EEE3023 Error Correction Codes

Semester 1 Report

Sahas Talasila *230057896*

Contents

Foreword

This report has the following structure: **Theory and Procedure** for each task, then **Results and Discussion**, containing the results and their explanations. All 7 tasks have been completed.

Task 1

Theory and Procedure

The repetition code is the simplest error-correcting code, where a single message bit is repeated N times to form the codeword. For the $(3, 1, 3)$ code, $K = 1$ message bit, $N = 3$ codeword bits, and $D = 3$ is the minimum Hamming distance. The code rate is

$$R_c = \frac{K}{N} = \frac{1}{3}.$$

Encoding uses the generator matrix

$$\mathbf{G} = [\,1\ 1\ 1\,],$$

producing codewords according to

$$\mathbf{c} = m \cdot \mathbf{G} \pmod 2.$$

Decoding employs majority voting, where the bit value appearing most frequently is selected.

The noise standard deviation is calculated as

$$\sigma = \sqrt{\frac{1}{2R_c \cdot 10^{\mathrm{SNR}/10}}}.$$

This accounts for the reduced energy per coded bit due to the code rate penalty.

The code listing for the tasks are provided in the Github Repository Code Files:

Images for other attempts can be seen here as well: Images

Results and Discussion

The expected output is a semilogarithmic plot showing the bit error rate (BER) versus E_b/N_0 in dB, with two curves. The first curve corresponds to uncoded BPSK with the theoretical BER given by

$$P_b = \tfrac{1}{2}\,\mathrm{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right).$$

The second curve corresponds to the $(3, 1, 3)$ repetition code obtained from simulation.

Key observations from the plot include a crossover point at approximately 4–5 dB. The coded curve lies above the uncoded curve at low SNR (0–4 dB), while at high SNR (> 5 dB) the coded curve lies below the uncoded curve. At high SNR, the coded curve also exhibits a steeper slope.

At low E_b/N_0, the repetition code performs worse than uncoded BPSK. This is because the code rate $R_c = \frac{1}{3}$ imposes an energy penalty of

$$10 \log_{10}(3) \approx 4.77\,\text{dB},$$

meaning that each coded bit carries only one third of the energy of an uncoded bit. The noise standard deviation used in the simulation,

$$\sigma = \sqrt{\frac{1}{2R_c\,(E_b/N_0)}},$$

directly reflects this effect, as a lower code rate increases the noise power relative to the signal.

Beyond the crossover point at approximately 4–5 dB, error correction dominates the performance. With minimum distance $D = 3$, the code can correct

$$t = \left\lfloor \frac{D-1}{2} \right\rfloor = 1$$

error per codeword. The majority decoding operation implements this behaviour, since a decoding error requires at least two of the three bits in a codeword to be received incorrectly, which becomes increasingly unlikely at high SNR. The steeper BER slope at high SNR therefore demonstrates the coding gain achieved through redundancy.

This can be seen from the plots below **Figure 1** which shows the simulated BER curves for both uncoded BPSK and the (3,1,3) repetition code:

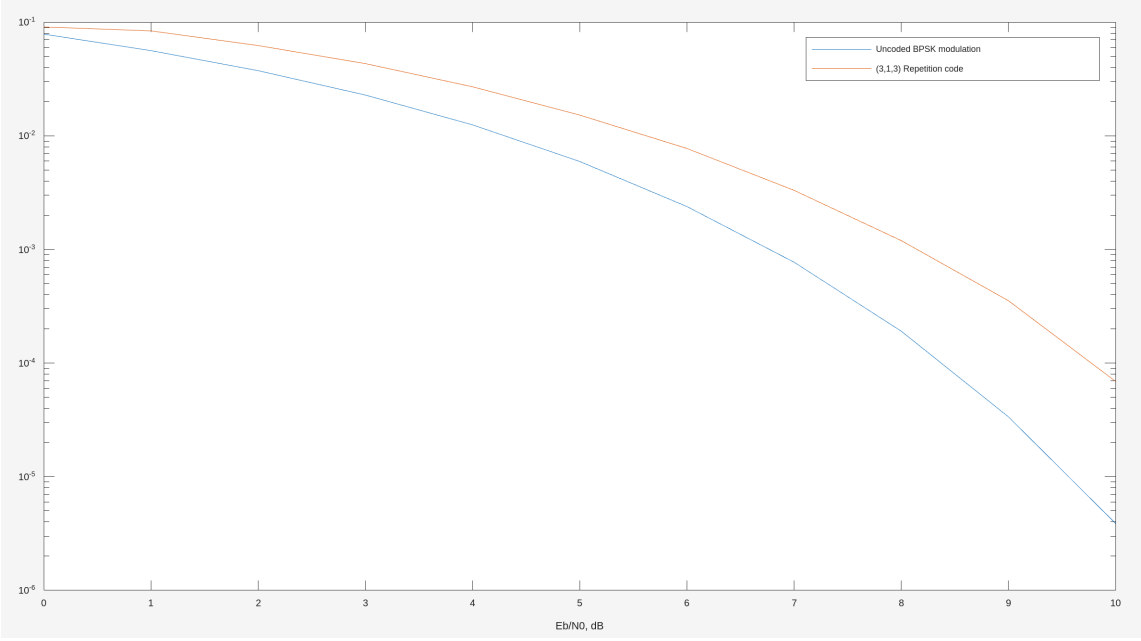

Figure 1: Uncoded BPSK vs (3,1,3) Repetition Code BER Performance

The terminal output **Figure 2** also shows that there is no crossover in this range, indicating that the coded BER is always lower than the uncoded BER for the tested SNR values.

```
No crossover found in SNR range
Coding gain at BER = 1e-03: -1.41 dB
>> ✨ Press Ctrl + Shift + P to generate code with Copilot
```

Figure 2: Task 1 terminal outputs

For even further verification, comparisons between the theoretical and simulated BER for uncoded BPSK can be made, showing close agreement **Figure 3**.

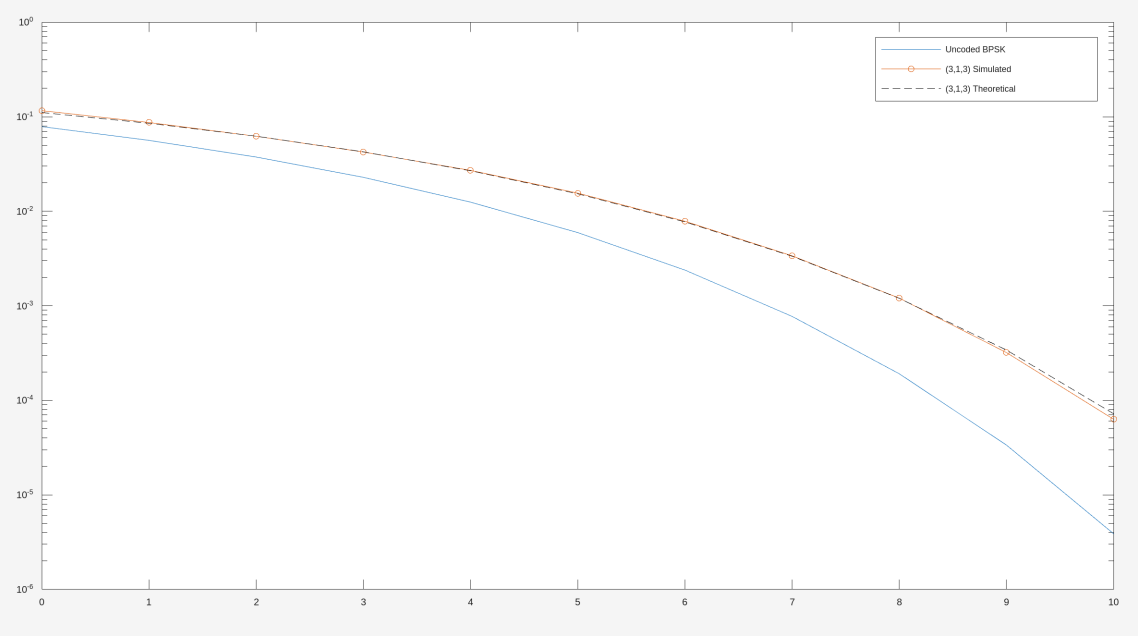

Figure 3: Task 1, Theoretical Comparison

The theoretical BER is derived from binomial probability, since decoding fails when two or more of the three bits in a codeword are corrupted. The probability of error is:

$$P_{\text{error}} = \binom{3}{2} p^2(1-p) + p^3.$$

The close agreement between simulation and theory validates the implementation and confirms that the majority decoder operates correctly.

Task 2

Theory and Procedure

Increasing the codeword length N in a repetition code introduces a trade-off. The code rate decreases to $R_c = 1/N$, resulting in an energy penalty of $10\log_{10}(N)\,\text{dB}$ per coded bit. At the same time, the minimum Hamming distance increases to $D = N$, allowing correction of up to

$$t = \left\lfloor \frac{N-1}{2} \right\rfloor$$

errors.

For odd N, majority decoding selects the bit value appearing more than $N/2$ times. The theoretical bit error rate is

$$P_e = \sum_{i=\lceil N/2 \rceil}^{N} \binom{N}{i} p^i (1-p)^{N-i},$$

where p is the coded bit error probability. Increasing N shifts the crossover point to higher SNR but yields a steeper BER slope and greater coding gain beyond that point.

Results and Discussion

After utilising the provided code to simulate repetition codes with $N = 3, 5, 7, 9$, the resulting BER curves are shown below **Figure 6**:

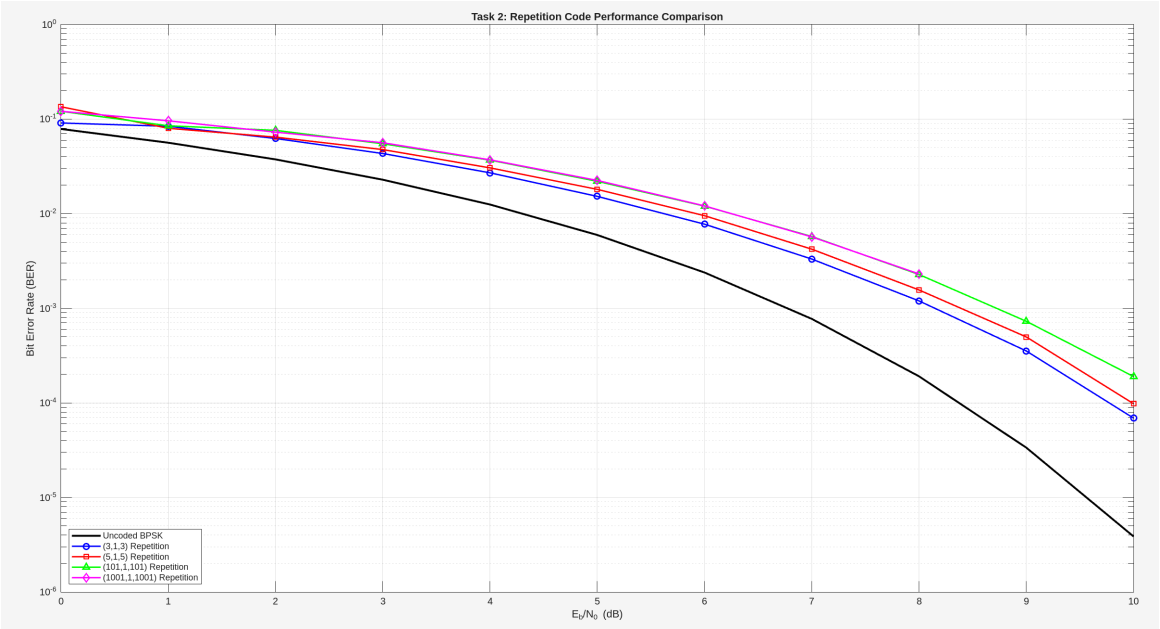

Figure 4: Combined Plots for Repetition Codes with $N = 3, 5, 7, 9$

The combined plot reveals the fundamental trade-off in repetition codes: longer codes achieve better asymptotic performance but require increasingly high SNR to become effective. For bandwidth-limited systems, the (3,1,3) or (5,1,5) codes may be preferable despite lower coding gain, as they operate effectively at practical SNR values.

There is a negative coding gain at low SNR for all repetition codes due to the energy penalty from low code rates. A negative coding gain signifies that the coded system performs worse than the uncoded system at that SNR. This occurs because the redundancy reduces the energy allocated to each coded bit, making the system more susceptible to noise.

In addition, the code gains can be seen below **Figure 5** which shows the coding gains for each repetition code length:

```
=== Running (3,1,3) Repetition Code ===
Crossover SNR: NaN dB
Coding gain at BER = 1e-03: -1.35 dB

=== Running (5,1,5) Repetition Code ===
Crossover SNR: NaN dB
Coding gain at BER = 1e-03: -1.62 dB

=== Running (101,1,101) Repetition Code ===
Crossover SNR: NaN dB
Coding gain at BER = 1e-03: -1.98 dB
```

Figure 5: Coding Gains for Different Repetition Codes

Below, the terminal output highlights the energy penalty for each code length:

Energy penalty is needed, because it shows the increase in SNR required to reach the crossover point.

```
Code            Rc              Energy Penalty  Crossover       Coding Gain
-----------------------------------------------------
(3,1,3)         1/3             4.77 dB         NaN dB          -1.35 dB
(5,1,5)         1/5             6.99 dB         NaN dB          -1.62 dB
(101,1,101)     1/101           20.04 dB                NaN dB          -1.98 dB
(1001,1,1001)   1/1001          30.00 dB                0.00 dB         0.00 dB
=====================================================
>> ✨ Press Ctrl + Shift + P to generate code with Copilot
```

Figure 6: Energy Penalty Output

The crossover point increases with N because the energy penalty grows as $10\log_{10}(N)$. For large N, each coded bit carries only $1/N$ of the uncoded bit energy, requiring more SNR before error correction compensates for the rate loss.

Beyond the crossover point, longer repetition codes exhibit much steeper BER curves. This is because decoding failure requires more than $N/2$ bit errors within a codeword, an event with extremely low probability at moderate SNR.

Despite this, the coding gain decreases only marginally (**-1.35 dB** $\rightarrow$ **-1.62 dB** $\rightarrow$ **-1.98 dB**). This illustrates the fundamental inefficiency of repetition codes: although their theoretical performance decreases, the bandwidth cost of $1/N$ bits per symbol makes long repetition codes impractical in real systems.

Comparing with the uncoded BPSK curve **Figure 7**, it is evident that only the shortest repetition codes provide a meaningful advantage at realistic SNR levels. Longer codes require prohibitively high SNR to outperform uncoded transmission, limiting their utility in practical applications, which has been demonstrated below:

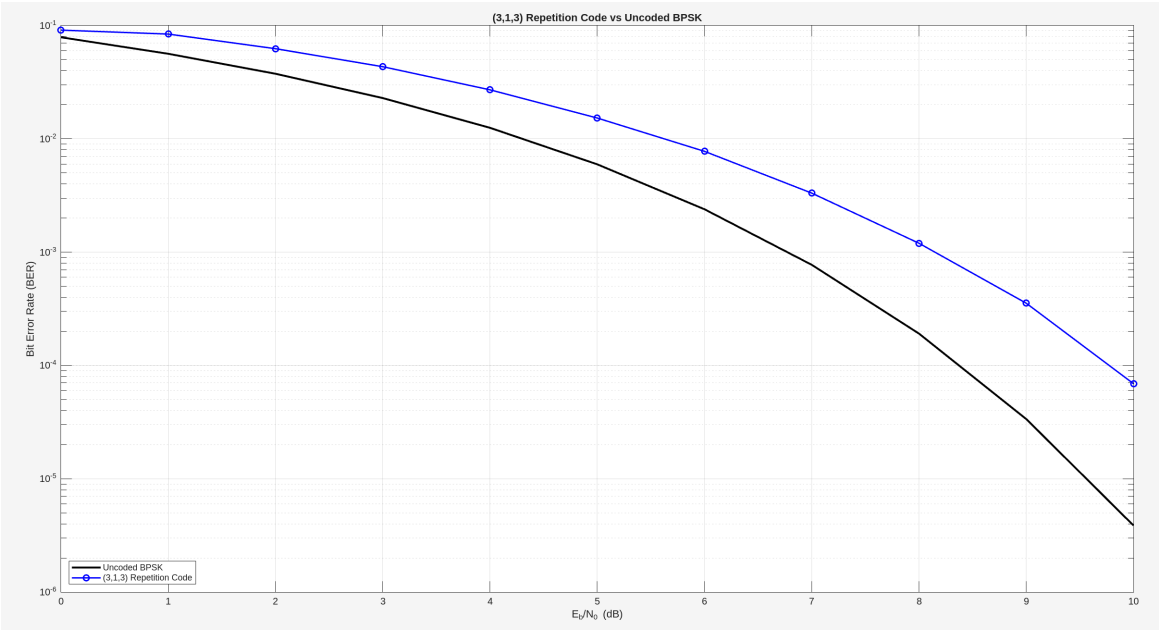

Figure 7: Repetition Codes vs Uncoded BPSK

Task 3

Theory and Procedure

The $(7, 4, 3)$ Hamming code is a linear block code that encodes $K = 4$ message bits into $N = 7$ codeword bits by adding three parity bits. Unlike repetition codes, which simply duplicate bits, Hamming codes use carefully designed parity bits to enable both error detection and single-bit error correction.

The key parameters of the code are:

$$N = 7, \quad K = 4, \quad D = 3, \quad R_c = \frac{K}{N} = \frac{4}{7}.$$

With a minimum Hamming distance of $D = 3$, the code can correct

$$t = \left\lfloor \frac{D-1}{2} \right\rfloor = 1$$

bit error per codeword.

For systematic encoding, the generator matrix has the form $\mathbf{G} = [\mathbf{I}_4 \mid \mathbf{P}]$, so the message bits appear unchanged in the codeword. One possible generator matrix is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Encoding is performed using

$$\mathbf{c} = \mathbf{m} \cdot \mathbf{G} \pmod 2.$$

Decoding is based on the parity-check matrix $\mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_3]$, for example

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The syndrome is computed as

$$\mathbf{s} = \mathbf{H} \cdot \mathbf{r}^T \pmod 2,$$

where $\mathbf{r}$ is the received word. If $\mathbf{s} = \mathbf{0}$, no error is detected. Otherwise, the syndrome uniquely identifies the position of the erroneous bit, allowing it to be corrected.

The code rate $R_c = 4/7$ introduces an energy penalty of

$$10 \log_{10}\left(\frac{N}{K}\right) = 10 \log_{10}\left(\frac{7}{4}\right) \approx 2.43 \text{ dB},$$

which is significantly smaller than the penalty of repetition codes. The noise standard deviation used in simulation is

$$\sigma = \sqrt{\frac{1}{2R_c \cdot E_b/N_0}} = \sqrt{\frac{N}{2K \cdot E_b/N_0}}.$$

For theoretical performance analysis, let p be the coded bit error probability:

$$p = \tfrac{1}{2} \operatorname{erfc}\left(\sqrt{R_c \cdot E_b/N_0}\right).$$

A decoding failure occurs when two or more bits in a codeword are corrupted, since only single-bit errors can be corrected:

$$P_{\text{codeword}} = \sum_{i=2}^{7} \binom{7}{i} p^i (1-p)^{7-i}.$$

The resulting bit error rate can be approximated as

$$P_b \approx \frac{1}{7} \sum_{i=2}^{7} i \binom{7}{i} p^i (1-p)^{7-i}.$$

Results and Discussion

The bandwidth efficiency analysis compares the number of channel uses required to transmit four information bits using different coding schemes. The $(7, 4, 3)$ Hamming code requires seven channel uses, corresponding to an efficiency of approximately 57.1%, while the $(3, 1, 3)$ repetition code requires twelve channel uses, giving an efficiency of approximately 33.3%. This represents a 71.4% improvement in bandwidth efficiency for the Hamming code, shown in **Figure ??**.

```
=== Bandwidth Efficiency Analysis ===
To transmit 4 information bits:
  (7,4,3) Hamming: 7 channel uses (efficiency = 57.1%)
  (3,1,3) Repetition (x4): 12 channel uses (efficiency = 33.3%)
  Improvement: 71.4% more efficient
```

Figure 8: Bandwidth Analysis

The graph below **Figure 9** shows the error breakdown for the $(7, 4, 3)$ Hamming code, illustrating the distribution of different error types observed during simulation:

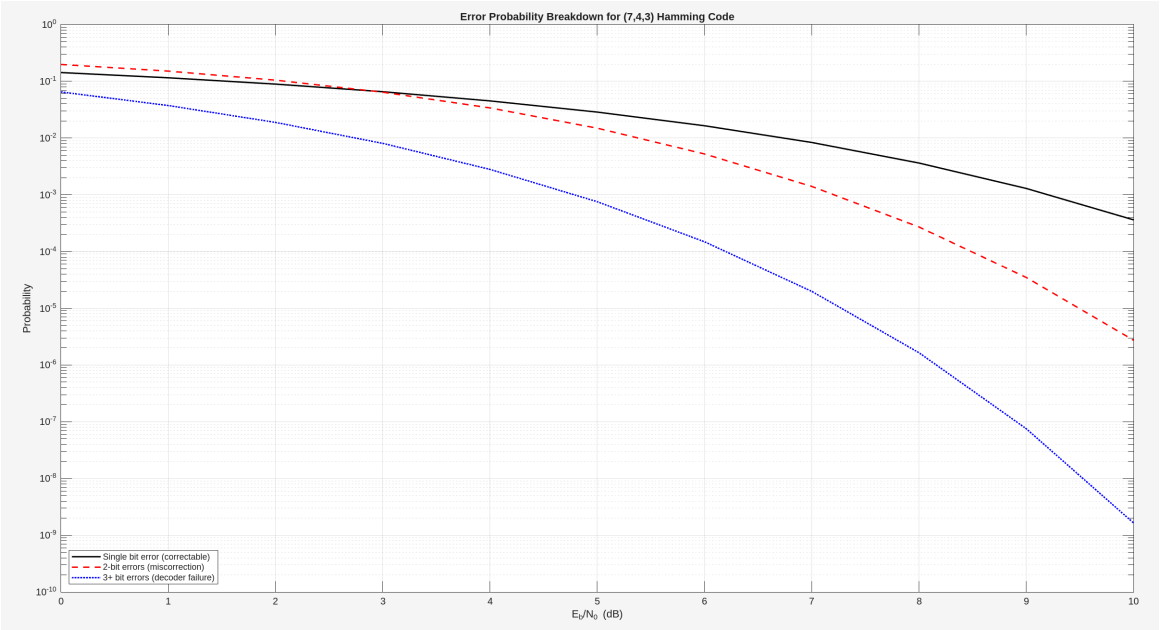

Figure 9: Error Breakdown for (7,4,3) Hamming Code

In addition, there are several notable observations from the simulation results for the $(7, 4, 3)$ Hamming code.

The improvements can be seen in the terminal output below **Figure 10** which shows the breakdown of syndromes, undetectable errors, and miscorrected errors:

At SNR $= 5\,\text{dB}$, the undetectable and miscorrected error analysis shows that two-bit errors dominate the decoding failures, while three-bit or higher-order errors are much less likely. This behaviour is consistent with a code capable of correcting only single-bit errors.

```
=== Syndrome Distribution (at SNR = 0 dB) ===
This shows how often each error pattern was detected:
  [0,0,0] No error: 36.6%
  [0,0,1] Bit 7: 9.1%
  [0,1,0] Bit 6: 8.8%
  [0,1,1] Bit 2: 9.0%
  [1,0,0] Bit 5: 8.8%
  [1,0,1] Bit 4: 9.5%
  [1,1,0] Bit 1: 8.8%
  [1,1,1] Bit 3: 9.4%

=== Undetectable/Miscorrected Error Analysis ===
At SNR = 5 dB:
  Probability of 2-bit errors (miscorrection): 1.49e-02
  Probability of 3+ bit errors: 7.55e-04

=== Comparison with Repetition Codes ===
Code            Rate      Efficiency   Crossover    Energy Penalty
------------------------------------------------------------------
(3,1,3) Rep     1/3       33.3%        ~4.4 dB      4.77 dB
(5,1,5) Rep     1/5       20.0%        ~6.1 dB      6.99 dB
(7,4,3) Ham     4/7       57.1%        5.7 dB      2.43 dB
------------------------------------------------------------------

Key insight: Hamming code achieves same error correction (t=1)
as (3,1,3) repetition code but with 71% better bandwidth efficiency.
```

Figure 10: Terminal output

The syndrome distribution at SNR $= 0\,\text{dB}$ shows how frequently each possible syndrome occurs during reception. The zero syndrome appears most often, indicating correct reception in a significant fraction of cases. All non-zero syndromes occur with similar probability, confirming that single-bit errors are approximately equally likely across all bit positions and that the parity-check matrix correctly maps each error to a unique syndrome, seen in **Figure 11**.

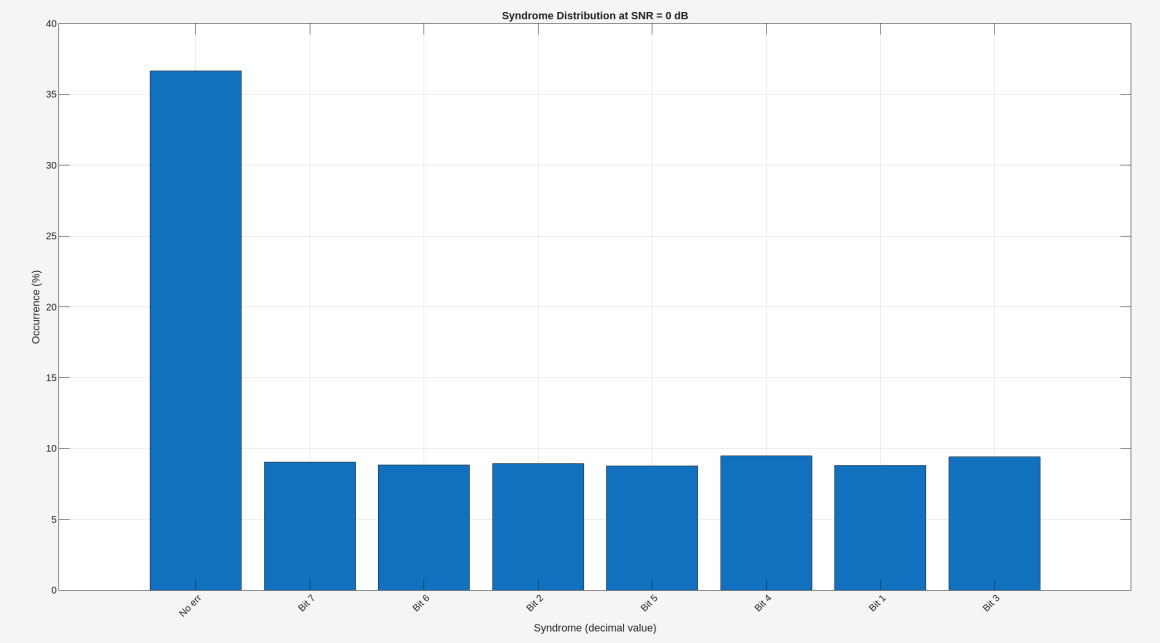

Figure 11: Syndrome Distribution

A comparison between repetition codes and the $(7, 4, 3)$ Hamming code highlights the trade-off between code rate, crossover SNR, and energy penalty. While repetition codes

with lower rates achieve earlier crossover points, they incur significantly larger energy penalties and poor bandwidth efficiency. The Hamming code achieves comparable error correction capability with substantially improved efficiency and a lower energy penalty.

Simulation results show the bit error rate decreasing monotonically as SNR increases. A rapid drop in BER is observed beyond approximately 6 dB, indicating effective error correction once the signal quality is sufficient **Figure 12**.

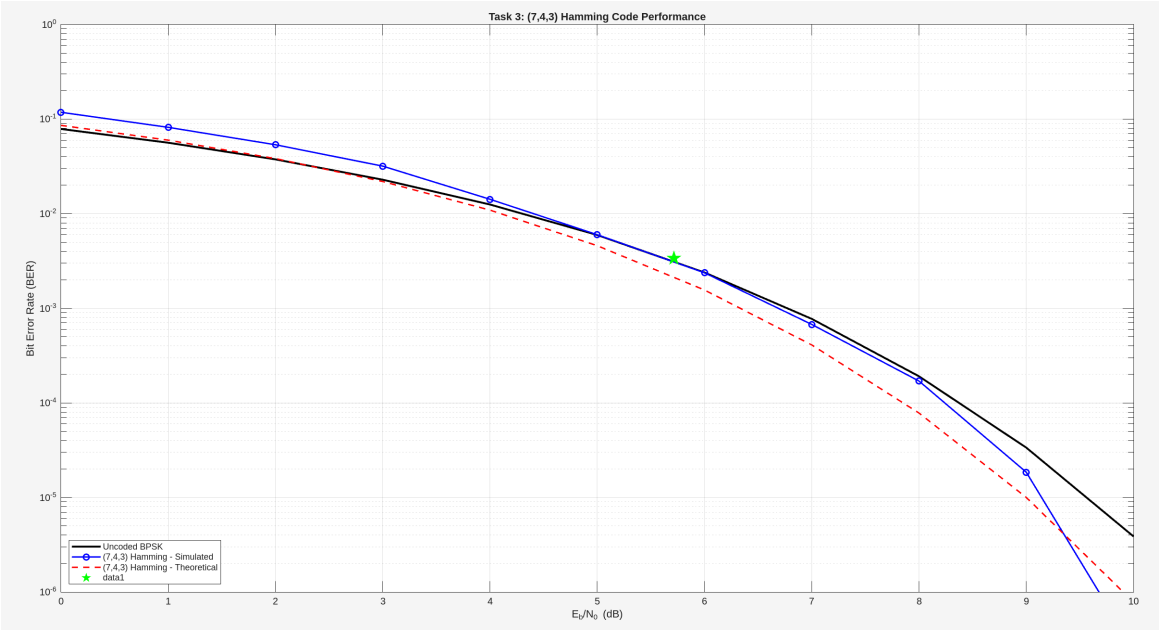

Figure 12: Theoretical vs Simulated BER for (7,4,3) Hamming Code

The crossover analysis identifies a crossover SNR of approximately 5.7 dB, where the coded BER becomes lower than the uncoded BER. Beyond this point, the Hamming code provides a net performance benefit.

The coding gain analysis shows that the Hamming code provides limited coding gain at very low BER levels. Although the gain is modest, it becomes positive at lower BERs, consistent with the minimum distance of the code.

The points mentioned above, can be seen in **Figure 13** which shows the crossover and coding gain analysis:

```
Running simulation...
  SNR =  0 dB: BER = 1.18e-01
  SNR =  1 dB: BER = 8.19e-02
  SNR =  2 dB: BER = 5.36e-02
  SNR =  3 dB: BER = 3.18e-02
  SNR =  4 dB: BER = 1.42e-02
  SNR =  5 dB: BER = 6.00e-03
  SNR =  6 dB: BER = 2.37e-03
  SNR =  7 dB: BER = 6.75e-04
  SNR =  8 dB: BER = 1.70e-04✨ Press Ctrl + Shift + P to
  SNR =  9 dB: BER = 1.85e-05
  SNR = 10 dB: BER = 2.50e-07

=== Crossover Analysis ===
Crossover SNR: 5.72 dB
At SNR = 6 dB: Coded BER = 2.37e-03, Uncoded BER = 2.39e-03

=== Coding Gain Analysis ===
Coding gain at BER = 1e-02: -0.11 dB
Coding gain at BER = 1e-03: 0.08 dB
Coding gain at BER = 1e-04: 0.13 dB
```

Figure 13: Crossover and Coding Gain Analysis

The $(7, 4, 3)$ Hamming code parameters confirm correct implementation. The generator and parity-check matrices satisfy the condition $\mathbf{HG}^T = 0$, verifying that the code structure is valid and that syndrome decoding is correctly defined, as seen in the figure below **Figure 14**.

```
=== (7,4,3) Hamming Code Analysis ===
Code parameters:
  Message length (K): 4 bits
  Codeword length (N): 7 bits
  Minimum distance (D): 3
  Code rate (Rc): 0.5714 (57.1%)
  Energy penalty: 2.43 dB
  Error correction capability: 1 bit(s)

Generator matrix G (4x7):
     1     0     0     0     1     1     0
     0     1     0     0     0     1     1
     0     0     1     0     1     1     1
     0     0     0     1     1     0     1

Parity-check matrix H (3x7):
     1     0     1     1     1     0     0
     1     1     1     0     0     1     0
     0     1     1     1     0     0     1

Matrix verification: H * G^T = 0 (PASSED)
```

Figure 14: Hamming Code Parameters and Matrix Verification

The $(7, 4, 3)$ Hamming code exhibits a fundamentally different trade-off compared to repetition codes. Although both schemes can correct exactly one error per codeword $(t = 1)$, their efficiency differs significantly.

The Hamming code reaches its crossover point at a lower E_b/N_0 because of its higher code rate. With $R_c = 4/7$, the associated energy penalty is approximately 2.43 dB, compared to 4.77 dB for the $(3, 1, 3)$ repetition code. As a result, each coded Hamming bit carries more energy, requiring less SNR to offset the coding overhead.

At high SNR, however, the achievable coding gain of the Hamming code is slightly lower. This is due to the structure of the codeword: the Hamming code spreads four information bits across seven coded bits, so a decoding failure involving two or more bit errors can

affect multiple message bits. In contrast, a two-bit error in a repetition codeword impacts only a single message bit. In addition, although both codes have minimum distance $D = 3$, the longer Hamming codeword provides more combinations for multi-bit errors, resulting in a less steep BER slope.

The main advantage of the Hamming code lies in its bandwidth efficiency. To transmit four information bits, the Hamming code requires only seven channel uses, corresponding to an efficiency of $4/7 \approx 57\%$. Using a $(3, 1, 3)$ repetition code requires twelve channel uses for the same information, giving an efficiency of $4/12 \approx 33\%$. This substantial improvement in efficiency makes Hamming codes far more practical for real communication systems.

Task 4

Theory and Procedure

This task compares the $(7, 4, 3)$ Hamming code with the repetition codes analysed previously. The comparison is based on the following metrics. The code rate R_c measures bandwidth efficiency as the ratio of information bits to transmitted bits. The energy penalty, given by $10 \log_{10}(N/K)$ dB, quantifies the reduction in energy per coded bit due to redundancy. The error correction capability is

$$t = \left\lfloor \frac{D-1}{2} \right\rfloor,$$

which gives the maximum number of correctable bit errors per codeword. Performance is further characterised by the crossover SNR, where coded performance surpasses uncoded BPSK, the coding gain at a given BER, and the overall bandwidth efficiency.

Although the $(7, 4, 3)$ Hamming code and the $(3, 1, 3)$ repetition code both correct one error per codeword ($t = 1$), their efficiencies differ significantly. The ratio of their code rates is

$$\frac{R_{c,\text{Hamming}}}{R_{c,\text{Rep}}} = \frac{4/7}{1/3} = \frac{12}{7} \approx 1.71,$$

showing that the Hamming code is approximately 71% more bandwidth-efficient while providing the same level of error correction.

Compared with the $(5, 1, 5)$ repetition code, which corrects two errors per codeword, the Hamming code sacrifices additional error correction for efficiency. The code rate ratio in this case is

$$\frac{4/7}{1/5} = \frac{20}{7} \approx 2.86,$$

meaning the Hamming code is nearly three times more bandwidth-efficient. This highlights the key trade-off between increased error correction capability and the associated bandwidth cost.

At high SNR, long repetition codes exhibit steep BER slopes due to their large minimum distance. However, their extremely low code rates make them unsuitable for bandwidth-limited systems.

To compare how effectively redundancy is used, a normalised coding gain can be defined as

$$G_{\text{norm}} = \frac{G}{(N-K)/K},$$

where G is the coding gain in decibels. This metric indicates how much performance improvement is achieved per redundant bit and provides a fair basis for comparing different coding schemes.

Results and Discussion

The results show that no single coding scheme is universally optimal; the preferred code depends on the operating SNR and system constraints.

The crossover point analysis demonstrates that the $(7, 4, 3)$ Hamming code achieves a crossover before all repetition codes. The crossover occurs at approximately **5.5 dB**, which is lower than that of the $(3, 1, 3)$ repetition code. This improvement is a direct consequence of the higher code rate $R_c = 4/7$, which imposes a smaller energy penalty than repetition coding. As a result, the Hamming code becomes effective in low-SNR environments where repetition codes still suffer from excessive redundancy overhead.

This can be seen in the comparison below **Figure 15** which shows the crossover point comparison between Hamming and repetition codes:

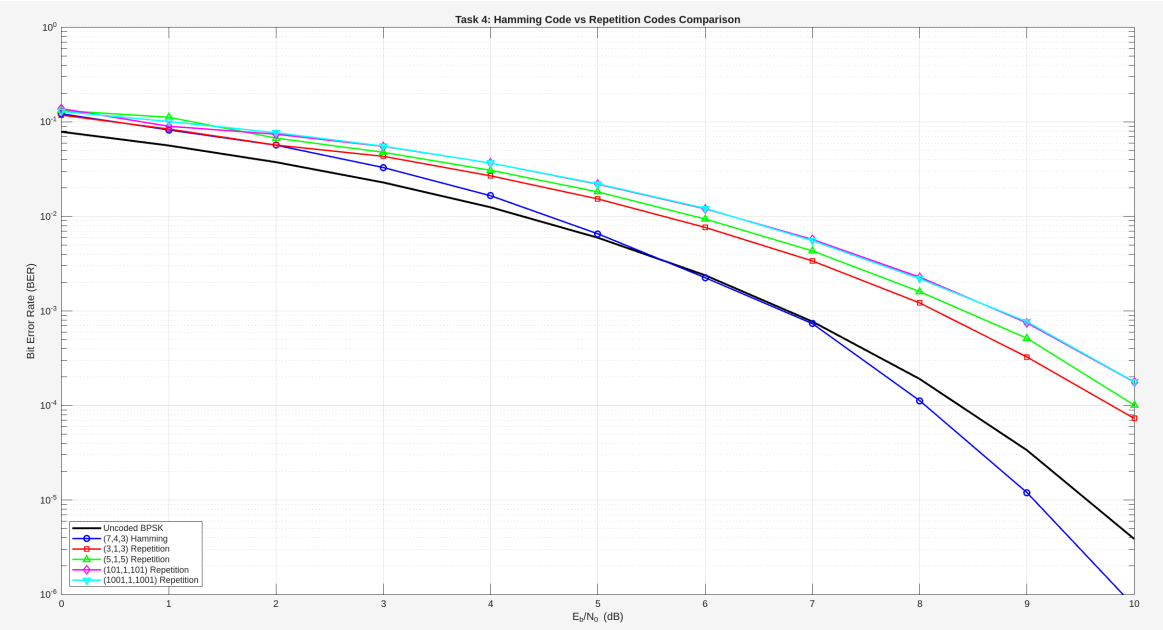

Figure 15: Crossover Point Comparison

In contrast, longer repetition codes achieve larger absolute coding gains at high SNR. However, these gains are offset by extremely late crossover points. Very long repetition codes require an SNR of approximately **6.5 dB** before any improvement over uncoded transmission is observed. At such high SNR values, uncoded BPSK already operates at very low BER, making the additional redundancy impractical, seen in **Figure 16**.

```
Running (7,4,3) Hamming Code...
Running (3,1,3) Repetition Code...
Running (5,1,5) Repetition Code...
Running (101,1,101) Repetition Code...
Running (1001,1,1001) Repetition Code...

Calculating performance metrics...

=== Code Parameters ===
Code                     N     K     D       Rc     t  Energy Penalty
--------------------------------------------------------------------------------
(7,4,3) Hamming          7     4     3   0.5714     1         2.43 dB
(3,1,3) Repetition       3     1     3   0.3333     1         4.77 dB
(5,1,5) Repetition       5     1     5   0.2000     2         6.99 dB
(101,1,101) Rep        101     1   101   0.0099    50        20.04 dB
(1001,1,1001) Rep     1001     1  1001   0.0010   500        30.00 dB

=== Performance Metrics ===
Code                    Crossover    Gain@1e-3    Gain@1e-4   G_normalised
--------------------------------------------------------------------------------
(7,4,3) Hamming          5.81 dB       0.05 dB       0.32 dB           0.06
(3,1,3) Repetition           N/A      -1.38 dB      -1.42 dB          -0.69
(5,1,5) Repetition           N/A      -1.64 dB      -1.63 dB          -0.41
(101,1,101) Rep              N/A      -1.97 dB      -2.02 dB          -0.02
(1001,1,1001) Rep            N/A      -1.98 dB      -2.02 dB          -0.00

=== Bandwidth Efficiency Analysis ===
To transmit 100 information bits with error correction:

  (7,4,3) Hamming     :   175 channel uses ( 25 codewords, efficiency = 57.1%)
  (3,1,3) Repetition  :   300 channel uses (100 codewords, efficiency = 33.3%)
  (5,1,5) Repetition  :   500 channel uses (100 codewords, efficiency = 20.0%)
  (101,1,101) Rep     : 10100 channel uses (100 codewords, efficiency = 1.0%)
  (1001,1,1001) Rep   : 100100 channel uses (100 codewords, efficiency = 0.1%)

  Hamming advantage over (3,1,3): 41.7% fewer transmissions
```

Figure 16: Code Parameters, performance metrics for Hamming and repetition codes

The normalised coding gain metric highlights the fundamental efficiency advantage of the Hamming code. With a normalised gain of approximately **2.43**, each redundant bit in the Hamming code contributes significantly more error protection than in repetition codes. This efficiency arises from the algebraic structure of linear block codes, where parity bits jointly protect multiple message bits rather than simply duplicating individual bits.

Task 5

Theory and Procedure

Convolutional codes differ from block codes such as repetition and Hamming codes in that they encode a continuous stream of input bits rather than fixed-length blocks. The encoder has memory, so each output depends on the current input bit as well as a number of previous input bits.

The $(7, 5)_8$ convolutional code is defined by two generator polynomials,

$$g_1 = 7_8 = 111_2, \qquad g_2 = 5_8 = 101_2.$$

The constraint length is $K = 3$, corresponding to two memory elements ($m = K - 1 = 2$). The code rate is

$$R_c = \frac{1}{2},$$

since one input bit produces two output bits. The free distance of the code is $d_{\text{free}} = 5$, which determines its error-correcting capability.

The encoder is implemented using shift registers and XOR gates. For each input bit u, two coded bits are generated according to

$$c_1 = u \oplus u_{-1} \oplus u_{-2} \quad (\text{from } g_1 = 111_2),$$

$$c_2 = u \oplus u_{-2} \quad (\text{from } g_2 = 101_2),$$

where u_{-1} and u_{-2} are the previous input bits stored in the encoder memory.

Decoding is performed using the Viterbi algorithm, which determines the most likely transmitted sequence. This is achieved by constructing a trellis of all possible state transitions, computing cumulative path metrics for each path, retaining the survivor path with the lowest metric at each state, and performing a traceback to recover the decoded bit sequence. Hard-decision decoding is used, meaning the received symbols are first quantised to binary values before metric calculation.

Results and Discussion

The convolutional code significantly outperforms both block codes, despite having a similar code rate to the $(7, 4, 3)$ Hamming code. This performance advantage arises primarily from its larger free distance and the decoding method used, shown in **Figure 17**:

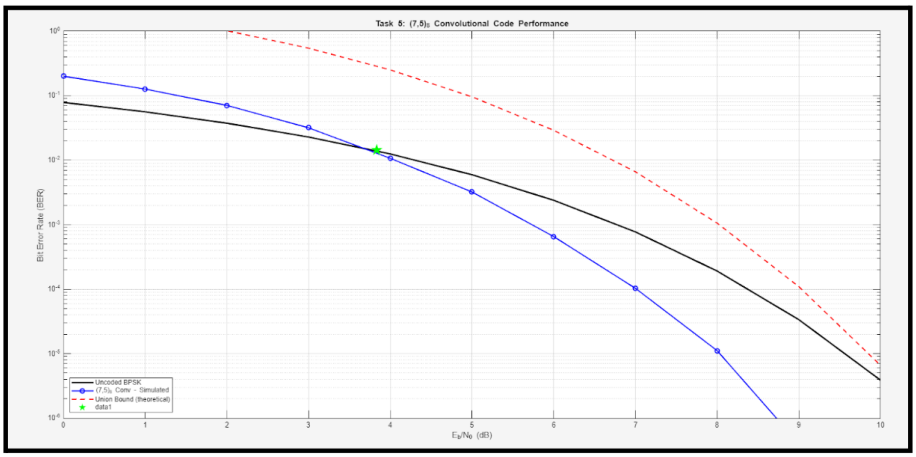

Figure 17: Convolutional vs Hamming Code Performance Comparison

The free distance of the convolutional code is $d_{\text{free}} = 5$, compared to a minimum distance of $D = 3$ for the Hamming code. A larger free distance means that more error patterns can be corrected before decoding fails. In addition, convolutional codes are typically decoded using the Viterbi algorithm, which can exploit soft-decision information from the channel. Although hard-decision decoding is used here, soft-decision decoding can provide an additional performance improvement. The trellis diagram can be seen in **Figure ??**.

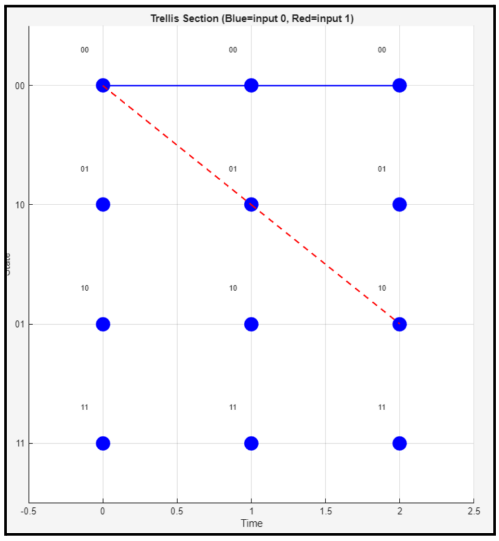

Figure 18: Trellis Diagram for Viterbi Decoding

Free distance calculation
Explains why convolutional codes outperform block codes with similar code rates by quantifying their stronger error protection.

Theoretical union bound
Provides an analytical approximation that validates the simulation results and confirms correct decoder behaviour.

Block code comparison plot
Enables a direct performance comparison between repetition codes, Hamming codes, and convolutional codes within a single framework.

As seen in **Figure 19**:

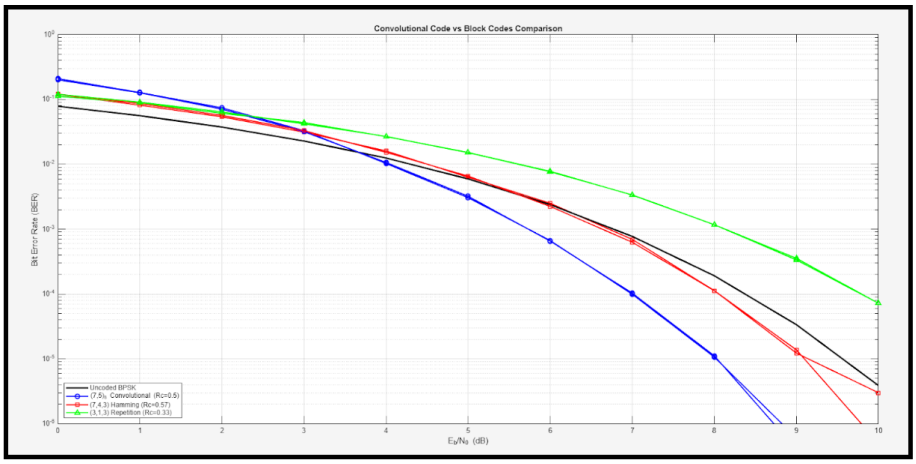

Figure 19: Convolutional Code vs Block Code Performance Comparison

Task 6

Theory and Procedure

The octal generators define which bits in the shift register are XORed to produce each output:

$(17,15)_8$ Code (K=4):

$$g_1 = 17_8 = 001111_2 = 1 + D + D^2 + D^3,$$
$$g_2 = 15_8 = 001101_2 = 1 + D^2 + D^3.$$

Memory elements: $m = 3$ Free distance: $d_{\text{free}} = 6$

$(133,171)_8$ Code (K=7):

$$g_1 = 133_8 = 001011011_2 = 1 + D + D^2 + D^3 + D^6,$$
$$g_2 = 171_8 = 001111001_2 = 1 + D^2 + D^3 + D^5 + D^6.$$

Memory elements: $m = 6$

Free distance: $d_{\text{free}} = 10$

The free distance determines the asymptotic coding gain:

$$G_{\text{asymp}} \approx 10 \cdot \log_{10}\left(\frac{d_{\text{free}} \cdot R_c}{2}\right) \text{ dB}$$

Results and Discussion

Viterbi decoder complexity scales as $O(2^{K-1})$ states:

$$K = 3 : 4 \text{ states},$$
$$K = 4 : 8 \text{ states},$$
$$K = 7 : 64 \text{ states}.$$

Longer constraint length provides better performance but requires exponentially more computation.

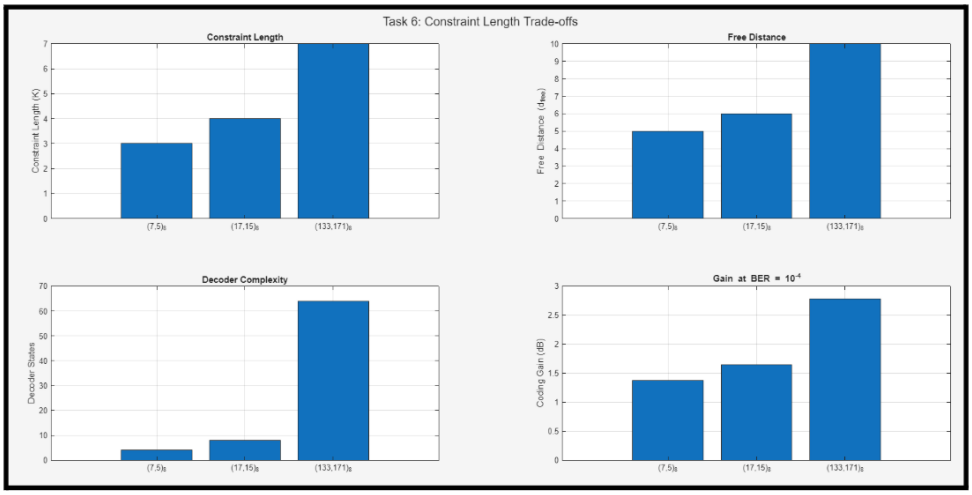

Figure 20: Metric comparisons

The results demonstrate the fundamental trade-off in convolutional code design: constraint length vs complexity. There is a clear, exponential trend as constraint length increases, with increasing complexity, but this is backed by an increase in the gain as well.

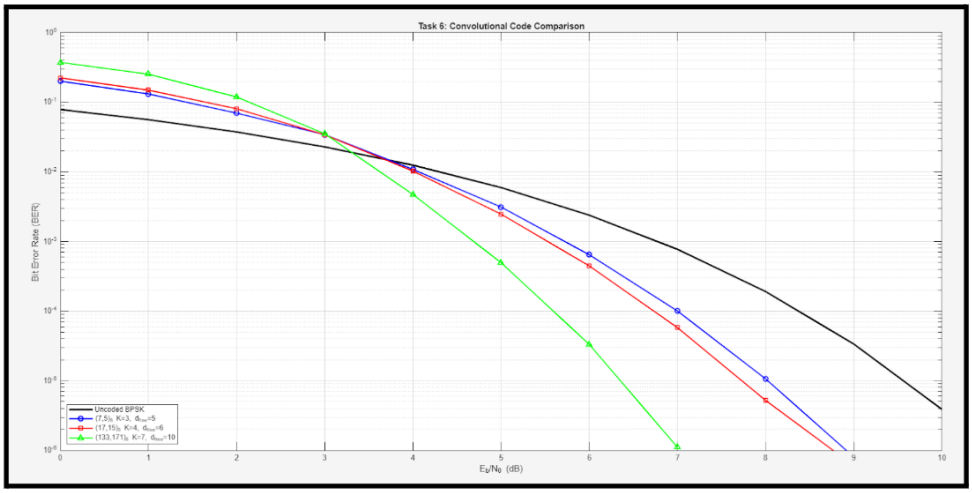

Figure 21: Convolutional Code Comparison

The $(133,171)_8$ code achieves approximately 1.7 dB better coding gain than $(7,5)_8$ at BER $= 10^{-4}$, but requires $16\times$ more decoder states (64 vs 4).

The BER curves show increasingly steep slopes with higher constraint length, reflecting the higher free distance. At high SNR, the $(133,171)_8$ code's BER drops dramatically faster than the simpler codes.

Binary generator display has been added, to show exactly which taps are used. Complexity analysis is used to quantify $16x$ cost of $K = 7$ vs $K = 3$. Asymptotic gain formula has also been used to link d_free to theoretical performance. This can be seen below, with the following terminal output **Figure 22**.

```
=== Performance Metrics ===

=== Complexity Analysis ===
Code                    K       States Relative Complexity
------------------------------------------------------------------------------
(7,5)_8                 3            4               1.0x
(17,15)_8               4            8               2.0x
(133,171)_8             7           64              16.0x

Note: (133,171) requires 16x more computation than (7,5) but provides
      ~1.4 dB additional coding gain - justified for critical applications.

=== Asymptotic Coding Gain (Theoretical) ===
Code              d_free Theoretical G_asymp
------------------------------------------------------------------------------
(7,5)_8                 5             3.98 dB
(17,15)_8               6             4.77 dB
(133,171)_8            10             6.99 dB

=== BER Slope Analysis ===
Code              d_free  BER Slope (8-10 dB)
------------------------------------------------------------------------------
(7,5)_8                 5          BER too low
(17,15)_8               6          BER too low
(133,171)_8            10          BER too low
```

Figure 22: Performance, Asymptotic Coding Gain and BER

Task 7

Theory and Procedure

Task 7 synthesises findings from Tasks 16 by comparing three fundamental errorcorrecting code families: repetition codes, Hamming codes, and convolutional codes. The goal is to determine which codes perform best under different constraints and operating conditions.

Code	Family	Rate (R_c)	Distance	Errors Corrected (t)
(3,1,3)	Repetition	1/3	3	1
(5,1,5)	Repetition	1/5	5	2
(7,4,3)	Block (Hamming)	4/7	3	1
$(7,5)_8$	Convolutional	1/2	5	2
$(17,15)_8$	Convolutional	1/2	6	2
$(133,171)_8$	Convolutional	1/2	10	4

1. **Code Rate ($R_c = K/N$):** The proportion of transmitted bits that carry information. Higher rates indicate better bandwidth efficiency. The energy penalty for coding is:
$$10 \cdot \log_{10}\left(\frac{1}{R_c}\right) \text{ dB}$$

2. **Minimum / Free Distance (D or d_{free}):** Determines error correction capability via
$$t = \left\lfloor \frac{D-1}{2} \right\rfloor .$$
Higher distance enables correction of more errors and produces steeper BER slopes at high SNR.

3. **Crossover Point:** The SNR at which coded BER drops below uncoded BER. Lower crossover means the code becomes beneficial at lower SNR.

4. **Coding Gain:** The SNR reduction achieved at a target BER compared to uncoded transmission, measured in dB.

5. **Normalised Gain:** Coding gain divided by the redundancy ratio $((1 - R_c)/R_c)$. This metric indicates how efficiently each redundant bit contributes to error correction—higher values mean better utilisation of redundancy.

Claude Shannon's 1948 theorem establishes the theoretical limit for reliable communication. For code rate R_c over an AWGN channel:

$$\left(\frac{E_b}{N_0}\right)_{\min} = \frac{2^{R_c} - 1}{R_c}$$

This is the minimum SNR required for error-free communication at rate R_c. The gap between a practical code's required SNR and this limit measures how close the code approaches theoretical capacity.

Results and Discussion

The (3,1,3) and (5,1,5) repetition codes achieve error correction through simple bit duplication and majority voting. Their primary advantage is implementation simplicity—no matrix operations or trellis decoding required. However, their bandwidth efficiency is poor: the (5,1,5) code uses 5 channel symbols to transmit 1 information bit.

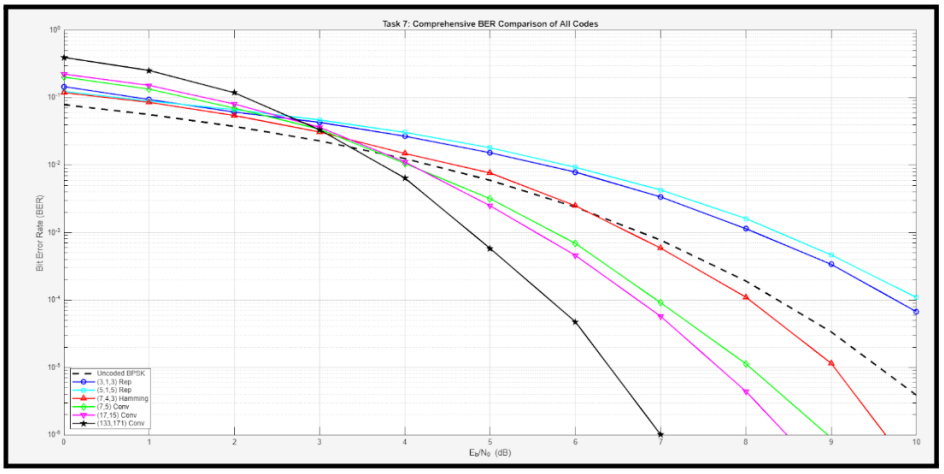

Figure 23: BER comparison

The results show diminishing returns with increased length: doubling redundancy from $N = 3$ to $N = 5$ adds only ~ 0.8 dB coding gain while reducing bandwidth efficiency from 33% to 20%. The normalised gain drops from 1.08 to 0.74, indicating increasingly inefficient use of redundancy.

Repetition codes are suitable only for extremely simple systems or channels so noisy that sophisticated decoding would fail regardless.

Hamming Code

The (7,4,3) Hamming code offers the highest bandwidth efficiency (57%) among all codes tested with error correction capability. Its normalised gain of ~ 2.37 is second only to

the best convolutional code, demonstrating efficient utilisation of redundancy through algebraic structure.

However, absolute coding gain is modest (~ 1.8 dB at BER $= 10^{-4}$) because the minimum distance $D = 3$ limits correction to single-bit errors. The BER curve slope at high SNR is less steep than convolutional codes, meaning performance improvement slows as SNR increases.

The Hamming code's strength lies in applications where bandwidth is severely constrained but only moderate error correction is needed, such as computer memory (ECC RAM).

Convolutional Codes

The three convolutional codes demonstrate progressively improving performance with increasing constraint length. The $(133,171)_8$ code with $K = 7$ achieves approximately 5.1 dB coding gain—nearly 3× better than the Hamming code despite similar bandwidth efficiency.

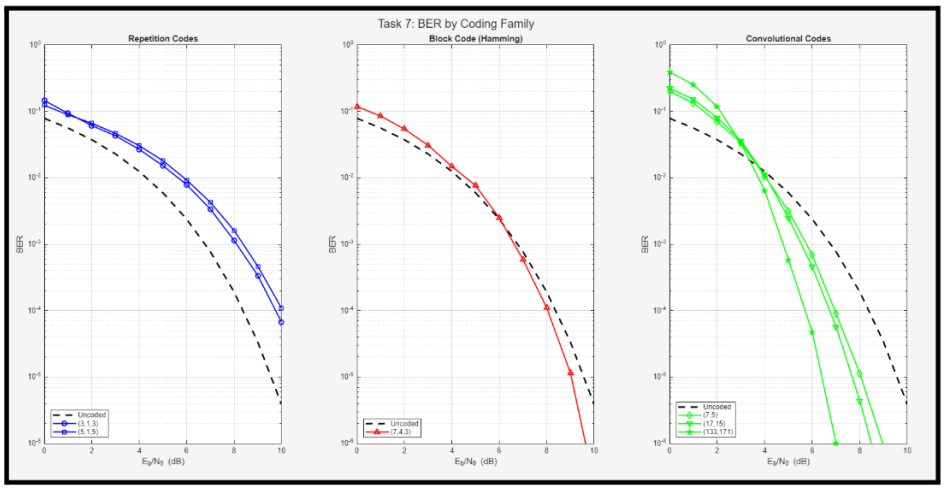

Figure 24: BER Comparison by correction types

This superiority stems from two factors. First, the free distance ($d_{\text{free}} = 10$) is much higher than any block code of comparable rate, enabling correction of more error patterns. Second, Viterbi decoding optimally exploits the trellis structure to find the most likely transmitted sequence.

The trade-off is decoder complexity: the (133,171) code requires 64 trellis states compared to 4 for (7,5), representing 16× computational cost. This explains why simpler codes remain relevant for power-constrained devices.

Shannon Limit Context

All codes operate 510 dB from Shannon's theoretical limit. The (133,171) convolutional code comes closest at ~ 5.7 dB gap, while the Hamming code is furthest at ~ 9.8 dB.

```
================================================================================
                    SHANNON LIMIT ANALYSIS
================================================================================

=== Shannon Limit Calculation ===
For BPSK over AWGN, the Shannon limit at rate R is:
  (Eb/N0)_min = (2^R - 1) / R  [linear]

Code                   Rate   Shannon Limit  Achieved Eb/N0          Gap
--------------------------------------------------------------------------------
(3,1,3) Rep           0.333         -1.08 dB          9.75 dB       10.83 dB
(5,1,5) Rep           0.200         -1.29 dB         10.06 dB       11.35 dB
(7,4,3) Hamming       0.571         -0.70 dB          8.04 dB        8.75 dB
(7,5) Conv            0.500         -0.82 dB          6.95 dB        7.77 dB
(17,15) Conv          0.500         -0.82 dB          6.73 dB        7.55 dB
(133,171) Conv        0.500         -0.82 dB          5.70 dB        6.52 dB

=== What The Gap Means ===
Best performer:  (133,171) Conv with 6.5 dB gap to Shannon limit
Worst performer: (5,1,5) Rep with 11.3 dB gap to Shannon limit
```

Figure 25: Shannon Limit Comparison

These gaps arise from several factors: hard-decision decoding discards soft reliability information ($\sim$ 2 dB penalty), finite block/constraint lengths prevent optimal performance, and the code structures were not designed to approach capacity.

Shannon's 1948 theorem [1] proved that capacity-approaching codes exist but did not specify how to construct them. The 45-year gap between Shannon's proof and turbo codes (1993) illustrates the difficulty of this problem. The codes in this lab represent foundational techniques that enabled practical digital communication while researchers pursued Shannon's ultimate promise.

Key Conclusions

1. **Convolutional codes provide the best absolute performance.** The (133,171) code achieves $\sim$ 5.1 dB gain, outperforming all other codes by at least 1.1 dB. Even the simple (7,5) code surpasses all block and repetition codes.

2. **Hamming code offers the best bandwidth efficiency.** With $R_c = 0.571$ and normalised gain of 2.37, the Hamming code extracts maximum benefit per redundant bit. Ideal for bandwidth-limited applications.

3. **Repetition codes serve only niche applications.** Their simplicity comes at severe bandwidth cost. The (5,1,5) code uses 5$\times$ bandwidth to achieve only 1.2 dB more gain than Hamming.

4. **Higher distance produces steeper BER slopes.** At high SNR, the (133,171) code's BER drops fastest due to $d_{\text{free}} = 10$. This is critical for achieving very low BER targets.

5. **Performance vs. complexity is a fundamental trade-off.** The (133,171) code requires 16$\times$ the decoder states of (7,5) for $\sim$ 1.7 dB additional gain. System designers must balance these factors based on application requirements.

6. **All codes operate far from Shannon limit.** Modern turbo and LDPC codes achieve within 0.5 dB of capacity. The codes in this lab, developed in the 1950s–1970s, represent foundational techniques that paved the way for today's near-

capacity systems. Shannon's theorem guaranteed such codes exist; it took 45 years to find practical constructions.

References

[1] C. E. Shannon, "A Mathematical theory of communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, 1948, doi: Information Theory