Codificación de fuente y canal

Barrios, Iván

03313/5

Ingeniería en Computación

Teoría de la Información y Codificación

30/06/2025

ÍNDICE

1. Introducción

El presente informe tiene como objetivo el análisis, diseño y evaluación de métodos de codificación y compresión aplicados a sistemas de transmisión digital de información. A lo largo del trabajo práctico se abordan dos tipos fundamentales de codificación de canal: los **códigos correctores** y los **códigos detectores de errores**, así como también técnicas de **compresión sin pérdidas** basadas en codificación de Huffman aplicadas a fuentes extendidas.

En la primera parte se analiza el desempeño de un código lineal binario sistemático de bloque (14,10), evaluado tanto como **corrector de errores** (capaz de detectar y corregir errores en la transmisión), como también en su función de **detector**, donde se descartan las palabras que presentan errores sin intentar corregirlas. Se implementa la simulación de la transmisión sobre un canal AWGN con modulación BPSK, midiendo las tasas de error de bit (BER) y palabra (WER), y se comparan con las curvas teóricas y con la ganancia de codificación esperada.

En la segunda parte se implementa la **compresión de una imagen binaria** mediante la aplicación de la codificación de Huffman sobre **fuentes extendidas** de orden 2 y 3. Se evalúa la eficiencia de la compresión a través del cálculo del **largo promedio de los códigos** y la **tasa de compresión** alcanzada. Asimismo, se discute por qué es posible lograr compresión aun cuando los bits individuales parecen equiprobables.

Este trabajo permite aplicar conceptos centrales de teoría de la información y comunicaciones digitales, como **redundancia, entropía, codificación óptima y detección/corrección de errores**, integrándolos en simulaciones prácticas que reflejan el comportamiento real de los sistemas.

2. Simulación – Código Corrector de Errores

En esta sección se analiza el comportamiento de un código de bloque binario sistemático (14,10) utilizado como **corrector de errores**. El objetivo es evaluar su rendimiento frente a un canal ruidoso y comparar los resultados obtenidos por simulación con los valores teóricos esperados para sistemas sin codificación.

2.1 Diseño del código

Se implementó un código de bloque lineal con las siguientes características dadas por el enunciado:

- Longitud de palabra de código: n=14
- Longitud de palabra de fuente: k=10
- Código sistemático, con matriz generadora de la forma $G = [\ I_k \mid P_{k\ x\ (n-k)}\ ]$
- Matriz de control de paridad: $H = [\ P^T \mid I_{n-k}\ ]$

La matriz P fue diseñada manualmente para garantizar una distancia mínima $d_{min} = 3$ (se calcula a partir de que la cota de Hamming para el código nos permite solo tc = 1, lo cual vemos más adelante), condición necesaria para poder **corregir un error** y **detectar hasta dos** (tc = 1, td = 2). Esto se verificó analizando combinaciones lineales de filas de la matriz H^T, asegurando que ninguna suma de hasta dos filas produzca el vector nulo.

A partir de d_{min}, se calcularon y verificaron los siguientes parámetros:

- **Cantidad de errores corregibles**: $tc = \lfloor \frac{dmin-1}{2} \rfloor = 1$
- **Cantidad de errores detectables**: $td = dmin - 1 = 2$
- **Ganancia de codificación asintótica**: $Ga = \frac{k}{n} \lfloor \frac{dmin+1}{2} \rfloor = 1,52dB$

En el archivo *generacionMatrices.py* se puede ver cómo se diseñaron las matrices G y H. En este archivo también se hicieron los cálculos de los parámetros mencionados (tc, td y Ga) para verificar que el código (14,10) puede corregir 1 error para que sea corrector. Se vuelve a aclarar que no puede corregir 2 errores, ya que la **cota de Hamming** muestra que como máximo podemos corregir 1 error. Podemos ver en la **Figura 1** la ecuación de la cota de

Hamming que se cumple solo para tc = 1, ya que para tc = 2, **no** se cumple que $106 \leq 2^{14-10}$.

$$\sum_{i=0}^{t_c} \binom{n}{i} \leq 2^{n-k}$$

Figura 1: Cota de Hamming para un código (n, k) corrector de tc errores.

Los parámetros G, H, n, k, Ga se dejaron preparados para ser importados en los archivos necesarios de simulación.

2.2 Simulación del sistema

Para analizar el rendimiento del código, se implementó una simulación completa del sistema de transmisión digital. El flujo general es el siguiente:

1. Se generaron palabras de fuente aleatoria en cada fila de la matriz $U_{M\,x\,k}$ (en la simulación propuesta, se utilizo M=1000000, es decir M palabras de fuente).
2. Se codificaron en $V = UG$ con el código (14,10)
3. Se transmitieron mediante modulación BPSK por un canal AWGN (simulación del canal brindado por la cátedra en el archivo *modeloTransmision.py*).
4. Se aplicó detección dura, recibiendo VR y se corrigieron los errores usando el síndrome $S = VR.H^T$, obteniendo VE como las palabras de código estimadas.
5. Se decodificaron las palabras corregidas para obtener estimaciones de los mensajes fuente, siendo Ue las primeras k columnas de VR (porque se realizó codificación sistemática).

Ese flujo se puede ver también en la **Figura 2** de una forma gráfica. La implementacion se puede encontrar en el archivo *simulacion.py*, en la función *simular_ber_wer_corrector*, donde se recibe como parámetros la lista de $\frac{Eb}{N0}$ con la que se quiere simular, *A* que es la amplitud de los símbolos, *repeticiones* que es la cantidad de veces que se simula para obtener una mejor estimación y *M* que es la cantidad de palabras de fuente aleatoria a enviar.

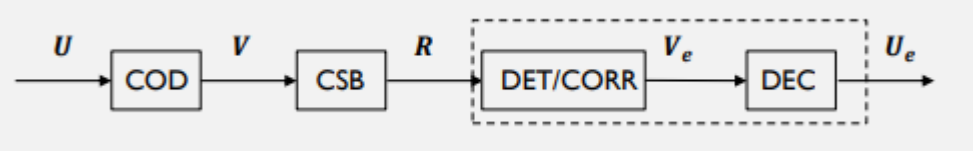

Figura 2 Flujo general de transmisión por canal mediante modulación BPSK por un canal AWGN

Para la simulación se usó una iteración para cada $\frac{Eb}{N0}$ en la lista propuesta (se utilizaron valores de 0 a 10 dB) en donde se calcula la probabilidad de error de bit de fuente y de palabra de código para cada $\frac{Eb}{N0}$, donde también se itera *repeticiones* veces con M cantidad de palabras de fuente.

Para cada valor de $\frac{Eb}{N0}$ se midieron: **BER** (tasa de error de bit): proporción de bits de fuente erróneos y **WER** (tasa de error de palabra): proporción de palabras de código erróneas. Estas dos métricas son las que devuelve la función.
Una vez obtenida la lista de valores de BER y WER para cada $\frac{Eb}{N0}$, se graficaron las curvas correspondientes para compararlas con la BER teórica sin codificación, $Peb = Q(\sqrt{\frac{2Eb}{N0}})$.

2.3 Resultados y comparación con teoría

Como puede observarse en la **Figura 3,** el uso del código (14,10) como corrector de errores permite **reducir significativamente la tasa de error de bit (BER)** en un amplio rango de valores de $\frac{Eb}{N0}$, especialmente a partir de valores intermedios hacia arriba. Esta mejora se traduce en un **desplazamiento hacia la izquierda** de la curva de BER codificado respecto de la curva teórica sin codificación, lo cual indica una **ganancia energética** efectiva del sistema.

Sin embargo, para valores bajos de $\frac{Eb}{N0}$, se observa que la curva de BER codificado queda **por encima de la curva teórica sin codificación**, lo cual indica un empeoramiento del rendimiento. Esto se debe a que, en presencia de mucho ruido, el número de errores introducidos por el canal supera la capacidad de corrección del código. En esas condiciones, el decodificador puede realizar **correcciones erróneas**, lo que termina aumentando la probabilidad de error respecto a un sistema sin codificación.

En cuanto a la **tasa de error de palabra (WER)**, ésta es naturalmente más alta que la BER, ya que un solo bit erróneo dentro de una palabra basta para considerarla incorrecta. La WER sigue una tendencia similar a la BER, disminuyendo con el aumento del $\frac{Eb}{N0}$, pero con valores absolutos mayores debido a esta definición.

No se muestra en este caso la **simulación** del sistema sin codificación, ya que coincide con la curva teórica y no aporta información adicional.

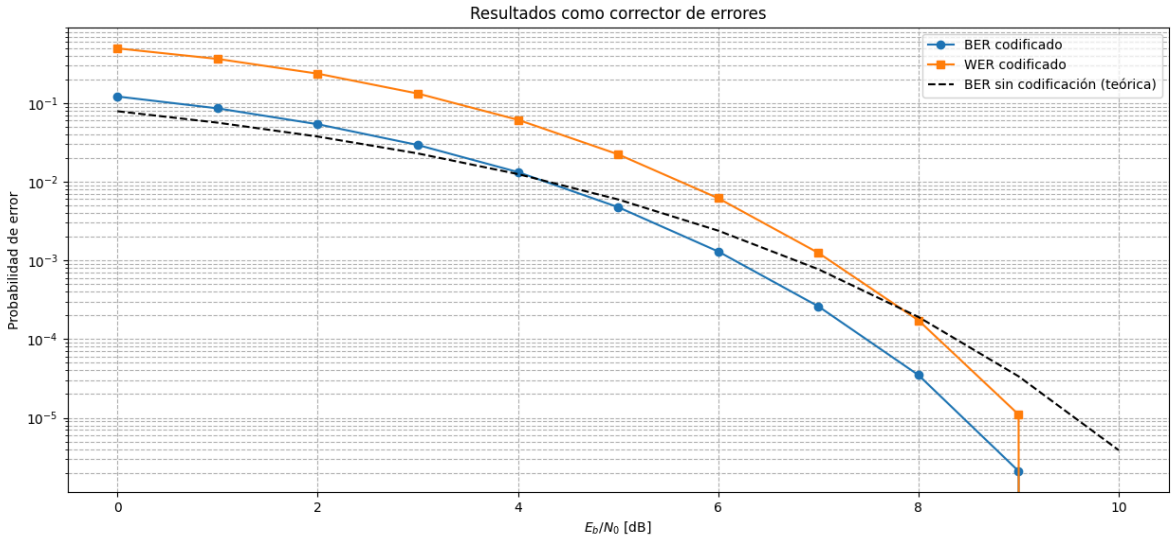

Figura 3 Curvas de tasa de error de bit (BER) y de palabra (WER) codificados y BER teórica sin codificación para código corrector

Para la **ganancia energética** ya mencionada previamente, podemos decir que es una medida que indica cuánta **eficiencia energética** se obtiene al usar un código de canal en lugar de transmitir sin codificación. Se define como la **diferencia horizontal** en dB entre la curva de BER sin codificación y la curva de BER codificado, para una misma tasa de error objetivo.

La ganancia teórica máxima que puede aportar un código está dada por la ganancia asintótica, calculada ya previamente en la sección de diseño mediante la fórmula:

$$Ga = \frac{k}{n}\left\lfloor\frac{dmin+1}{2}\right\rfloor = 1,52\ dB$$

En la **Figura 4** podemos ver este valor representado por la línea roja punteada horizontal, la cual podemos ver claramente que está por encima siempre de la curva de ganancia real del código calculada para cada $\frac{Eb}{N0}$ haciendo la **diferencia horizontal** en dB entre la curva de BER sin codificación y la curva de BER codificado, para una misma tasa de error objetivo. Esta ganancia real está representada por la curva azul en el gráfico y podemos observar que:

- Para valores bajos de $\frac{Eb}{N0}$, la ganancia es **negativa** o cercana a 0. Esto es esperable, ya que en esa región el código no puede corregir muchos errores y hasta puede introducir errores por malas correcciones.
- A partir de $\frac{Eb}{N0} \approx 5$, la ganancia comienza a estabilizarse, alcanzando un máximo cerca de **1 dB**.
- La ganancia real **no alcanza completamente** la ganancia asintótica (1.42 dB), aunque se aproxima razonablemente en los valores altos de SNR.

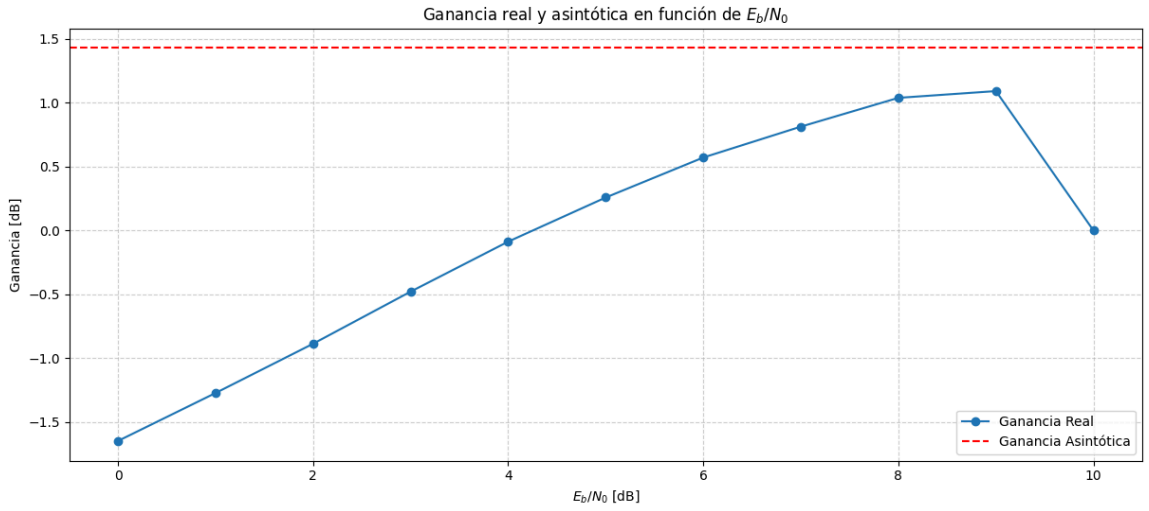

Figura 4 Curva de Ganancia real de simulación vs Ganancia asintótica para código corrector

La diferencia entre la **ganancia real** y la **asintótica** puede deberse a algunos factores:

- La ganancia asintótica es una **cota ideal**, válida solo para tasas de error muy bajas y con ciertas suposiciones teóricas.

- En la simulación real:
    - No se alcanza una BER lo suficientemente baja para aprovechar toda la capacidad del código.
    - Existen errores de corrección que degradan el rendimiento en ciertos rangos de SNR.
- Además, el número de muestras y la detección dura pueden afectar la estimación de la BER y por lo tanto el cálculo de la ganancia.

3. Simulación – Código Detector de Errores

En esta sección se evalúa el comportamiento del mismo código lineal (14,10), esta vez utilizado como **detector de errores**. A diferencia del modo corrector, en este caso el sistema **descarta las palabras recibidas** cuando el síndrome no es nulo, y solo se consideran como válidas aquellas que se reciben sin errores detectados.

3.1 Curvas de BER y WER

En la **Figura 5** se muestran las curvas simuladas de **BER codificado**, que es la tasa de error de bit medida solo sobre las palabras válidas (no descartadas), **WER codificado**, que es la fracción de palabras descartadas por detección de error y por último de **BER sin codificación (teórica)** que es la curva de referencia para BPSK en canal AWGN.

La curva de **BER codificado** muestra valores muy bajos en todo el rango de $\frac{Eb}{N0}$. Esto se debe a que solo se consideran las palabras que pasan la detección (síndrome nulo), las cuales tienen alta probabilidad de ser correctas.

En cambio, la **WER** es alta para valores bajos de $\frac{Eb}{N0}$, indicando que el sistema descarta una gran cantidad de palabras cuando el canal es ruidoso. A medida que el $\frac{Eb}{N0}$ aumenta, la WER desciende, ya que es menos probable que ocurran errores.

En valores altos de $\frac{Eb}{N0}$, ambas curvas convergen a valores muy bajos, como se espera de un sistema confiable.

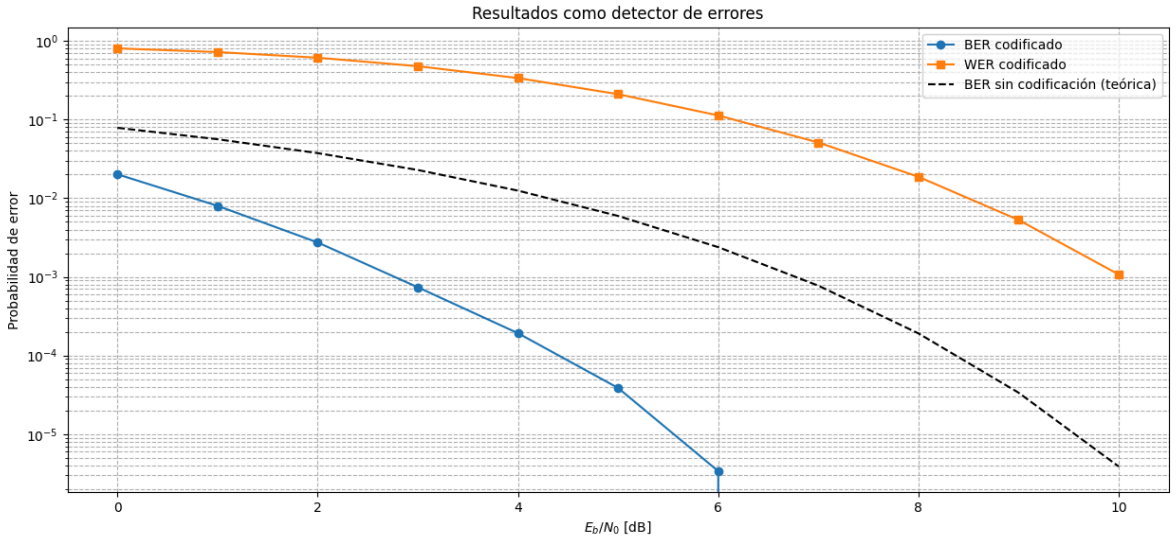

Figura 5 Curvas de tasa de error de bit (BER) y de palabra (WER) codificados y BER sin codificación para código detector

Cabe destacar que en el sistema detector evaluado no se contempla ningún mecanismo de retransmisión para las palabras descartadas. Es decir, las palabras con errores detectados son eliminadas sin volver a enviarse, lo cual impacta directamente en las métricas globales del sistema.

4. Codificación de Fuente con Huffman

Esta implementación tiene como objetivo analizar la eficiencia de la compresión sin pérdida mediante **codificación de Huffman**, aplicada a una **imagen binaria**. Para ello se considera una **fuente extendida** de orden 2 y 3, es decir, en lugar de codificar bits individuales, se agrupan de a 2 y 3 bits, respectivamente, antes de aplicar el algoritmo de codificación.

4.1 Procedimiento

Se utilizó la imagen binaria *'logo FI.tif'* provista por la cátedra. La imagen fue cargada y convertida en un vector plano de bits (0 y 1). Luego, para cada orden n (2 y 3):

1. Se agruparon los bits en bloques de n bits consecutivos.
2. Se contó la frecuencia relativa de aparición de cada símbolo posible para conocer de antemano las probabilidades de cada símbolo.

3. Se construyó un código óptimo de Huffman a partir de las probabilidades estimadas.
4. Se calculó el **largo promedio por símbolo** como: $L = \sum_i p_i . l_i$ con p_i la probabilidad de aparición de símbolo i y l_i la longitud en bits del código de Huffman.
5. Finalmente, se calculó la **tasa de compresión** como:
   $Tasa\ de\ compresión = \frac{bits\ sin\ comprimir\ por\ símbolo}{L} = \frac{n}{L.n} = \frac{1}{L}$

4.2 Resultados y análisis

Al ejecutar el código, se obtuvieron los valores de la **Figura 7**, los cuales muestran los códigos resultantes, el largo promedio y la tasa de compresión considerando fuente extendida de orden 2 y orden 3.

```
Orden 2:
  - Largo promedio: 0.7583 bits/símbolo
  - Tasa de compresión: 1.3187
Códigos de Huffman por símbolo:
  00 → 01
  01 → 000
  10 → 001
  11 → 1

Orden 3:
  - Largo promedio: 0.5328 bits/símbolo
  - Tasa de compresión: 1.8770
Códigos de Huffman por símbolo:
  000 → 11
  001 → 10001
  010 → 100001
  011 → 1011
  100 → 1001
  101 → 100000
  110 → 1010
  111 → 0
```

Figura 7 Resultados de códigos resultantes, largo promedio y tasa de compresión tras simular la compresión

Pasando los resultados a la **Tabla 8**, la cual muestra que al extender la fuente de orden 2 a orden 3, se logra un **mayor grado de compresión**, reduciendo aún más el largo promedio por símbolo y aumentando la eficiencia.

Tabla 8: Valores de largo promedio y tasa de compresión para extensión n.

Orden de la fuente	Largo promedio	Tasa de compresión
2	0,7583 bits/símbolo	1,3187
3	0,5328 bits/símbolo	1,8770

El uso de una fuente extendida permite modelar **correlaciones entre bits consecutivos**. Si bien los bits individuales en una imagen binaria pueden parecer equiprobables (aproximadamente mitad blancos y mitad negros), al observar grupos de 2 o 3 bits se descubren **patrones más frecuentes**.

El código de Huffman aprovecha esta **redundancia estructural** para asignar códigos más cortos a los símbolos frecuentes y más largos a los poco frecuentes, lo que reduce el largo promedio y permite compresión.

Podemos ver además en la tabla los valores del largo promedio que a medida que aumenta la extensión de la fuente, se hace más chico el largo promedio. Esto **no** va a ser así hasta que el largo promedio sea 0, sino que se va achicando hasta que llega a un valor teórico (límite) dado por la entropía de la fuente (cuando la extensión n tiende a infinito).

Aunque los bits individuales tengan probabilidades similares $P(0) \approx P(1) \approx 0,5$, eso **no implica** que los grupos de bits (fuente extendida) sean equiprobables. Al agrupar bits, se pueden observar **dependencias estadísticas**: ciertos patrones ocurren más seguido que otros debido a estructuras o regularidades presentes en la imagen.

Por ejemplo, en una imagen con bloques negros y blancos, es probable que aparezcan más frecuentemente combinaciones que tengan bits repetidos en la secuencia, por ejemplo, 000 o 111. Esta **distribución no uniforme** de símbolos en la fuente extendida permite que Huffman logre **compresión**.

Conclusiones

Este trabajo permitió analizar y comparar distintas técnicas de codificación aplicadas a la transmisión y compresión de información.

En el primer ejercicio se diseñó un código (14,10) capaz de corregir un error y detectar dos. Su uso como corrector mostró una reducción significativa en la tasa de error de bit (BER) a partir de ciertos valores de $\frac{Eb}{N0}$, alcanzando una ganancia cercana a la asintótica. En cambio, al usarlo como detector, se logró una BER extremadamente baja sobre las palabras válidas, pero a costa de descartar una gran cantidad de información cuando el canal es ruidoso.

En el tercer ejercicio se aplicó la codificación de Huffman sobre una imagen binaria. Aunque los bits eran aproximadamente equiprobables, al extender la fuente se revelaron patrones estadísticos que permitieron lograr compresión. Se pudo ver claro como cada vez que se extendía la fuente, el largo promedio bajaba y la tasa de compresión aumentaba.

En conjunto, los resultados mostraron cómo la codificación de canal mejora la **confiabilidad** y la codificación de fuente mejora la **eficiencia**, ambas esenciales para optimizar un sistema de comunicación.

Bibliografía

[1] J. G. Proakis, Digital communications, McGraw-Hill, 3rd edition, 1995.

[2] S. Haykin Communication Systems, J. Wiley & Sons, 4th edition, 2001.

[3] S. M. Kay Fundamentals of Statistical Signals Processing. Volume II: Detection Theory., Prentice Hall, 1998.