Teoría de la información y codificación

Laboratorio N°1

Dolores Garro

Legajo: 03115/9

Ingeniería en Computación

Introducción

En el presente trabajo se realizó la simulación y análisis del comportamiento de sistemas de comunicación digital con *codificación de canal* y *codificación de fuente*.

En principio, se trabajó con un código de bloque lineal *(n,k) = (14,10)* provisto por la cátedra, simulando su funcionamiento como corrector-detector de errores, y luego únicamente como detector. Se compararon los resultados simulados con los teóricos, analizando las *probabilidades de error* y la *ganancia asintótica*.

Por otro lado, se utilizó el algoritmo de Huffman para comprimir la imagen "*logoFI.tif"* provista por la cátedra, la cual fue convertida a binario para su próxima codificación de fuente de orden 2 y 3, Se estimaron las probabilidades de los bloques, se construyó el árbol de codificación para ver la eficiencia de la codificación mediante el cálculo del *largo promedio* y la *tasa compresión*.

Se optó por utilizar Python como herramienta debido a una mayor experiencia previa en ese entorno, lo cual facilitó la implementación de los procedimientos requeridos.

A continuación, se detalla a lo largo del informe cada etapa de codificación realizada, con la interpretación de los resultados obtenidos.

Simulación de sistema de comunicación digital con codificación de canal

El sistema de codificación y decodificación de canal para comunicación digital puede representarse por el esquema de la Figura 1, en donde se ven cuatro etapas, las cuales se explicaran a continuación lo que realiza cada una y lo que se implementó en el código en Python.

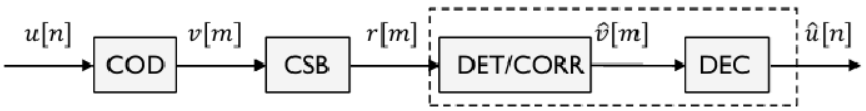

Figura 1: sistema de comunicación digital con codificación de canal

En la etapa de codificación, como entrada se tiene u[n], que representa la secuencia original de bits que se desea transmitir. En el código se generaron M mensajes aleatorios de una longitud de k = 10 bits. Esta etapa consiste en realizar la operación

$$v[m] = u[n] * G$$

Aquí se transforma cada bloque de *k* bits u[n] en un bloque de *n* bits v[m], multiplicando por la matriz generadora *G*

$$G_{kxn} = [I_n \mid P_{kx(n-k)}] = [I_{14} \mid P_{10x4)}]$$

$$G_{10x14} = [1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \mid 1\,1\,0\,0$$
$$0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \mid 0\,1\,1\,0$$
$$0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \mid 0\,0\,1\,1$$
$$0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \mid 1\,0\,0\,1$$
$$0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \mid 1\,0\,1\,1$$
$$0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \mid 1\,1\,0\,1$$
$$0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \mid 1\,1\,1\,0$$
$$0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0 \mid 0\,1\,1\,1$$
$$0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \mid 1\,1\,1\,1$$
$$0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \mid 0\,1\,0\,1\,]$$

La matriz *G* se construyó a partir de la fórmula indicada anteriormente y sabiendo que las filas de $P_{kx(n-k)}$ deben cumplir las siguientes condiciones

- Deben ser linealmente independientes
- Ninguna puede tener un único 1
- Ninguna puede ser completamente nula
- Deben ser todas distintas

Ya obtenido u[n], se pasa a la etapa Canal Simétrico Binario, en donde se simula el canal de transmisión con ruido. En este canal se introducen errores aleatorios en los bits transmitidos v[m], generando una secuencia recibida r[m].

En el código, el canal se implementó utilizando modulación BPSK y agregando ruido gaussiano con variaciones de Eb/N0. Se definieron los parámetros necesarios

$$A = 1, \quad E_s = A^2, \quad E_{bf} = E_s * (n/k)$$

en donde A es amplitud de la señal, Es energía del símbolo del canal, Ebf energía por bit de información

para calcular la densidad espectral de potencia de ruido *N0,* convirtiendo Eb/N0 a razón lineal

$$E_{bN0} = 10^{(E_{N0dB}/10)}$$

y determinar *N0* a partir de la *Ebf*

$$N_0 = E_{bf} / E_{bN0}$$

Se generó un vector gaussiano y por último se realizó la modulación BPSK y transmisión

$$s = (2V - 1) * A$$

$$r = s + ruido$$

Como último paso en la etapa de *CSB*, se aplicó una detección dura[1] para obtener la secuencia binaria recibida V_r, si la señal recibida *r* es positiva, el bit es un 1, si es negativa, es un 0. Ingresando en la etapa de detección y corrección de errores, se calcula el síndrome utilizando la H transpuesta

$$s = r[m] * H^T$$

si el síndrome es 0 significa que el bloque recibido no tiene errores, caso contrario se busca localizar el error y se intenta corregir.

En caso de error, como nuestro corrector es de errores simples ($T_c = 1$) la localización del error consiste en buscar el valor del síndrome en las filas de H^T . La fila X en la que se encuentre el valor del síndrome determina que el error está en el bit X de la palabra recibida.

En este trabajo de simulación se solicita realizar la codificación y decodificación de canal para comunicación digital como *corrector de errores* tanto como *detector de errores*. En el caso del corrector, se recorre la H^T y al encontrar el valor del síndrome en las filas de la matriz se invierte el bit. En el caso del detector se reporta en qué bit se detectó el error.

Como última etapa se encuentra la decodificación, en la que se recupera el mensaje original *u^[n]* a partir del bloque corregido *v^[m]* de forma sistemática, tomando directamente la parte correspondiente a los bits $n = 10$ de información originales.

Cálculo curvas de errores P_{ep} y P_{eb}

En el sistema simulado, se estudió la tasa de error de bit P_{eb} bit y de palabra P_{ep}, para diferentes relaciones señal-ruido E_b/N_0. Los resultados se compararon con la curva teórica del sistema sin codificación y con las expresiones teóricas proporcionadas.

Al utilizar un código de bloque (n,k) que corrige hasta t_c errores, la probabilidad de error de palabra se calcula como

[1] detección dura: si la señal recibida es positiva, se toma al bit como 1, caso contrario se toma al bit como 0.

$$P_{e,p} = \sum_{i=t_c+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

donde *p* es la probabilidad de error de bit.

Pep es probabilidad de error de tener error de palabra de código

Para p << 1, se puede aproximar a

$$P_{e,p} \simeq \binom{n}{t_c+1} p^{t_c+1}$$

por tomar el primer término de la sumatoria suponiendo que el término $i = t_c + 1$ de la sumatoria sea mayor que todos los siguientes.

La probabilidad de error de bit se ajusta mediante

$$P_{e,b} \simeq \frac{2t_c+1}{n} \binom{n}{t_c+1} p^{t_c+1}$$

Gráfica curvas de errores Pep y Peb

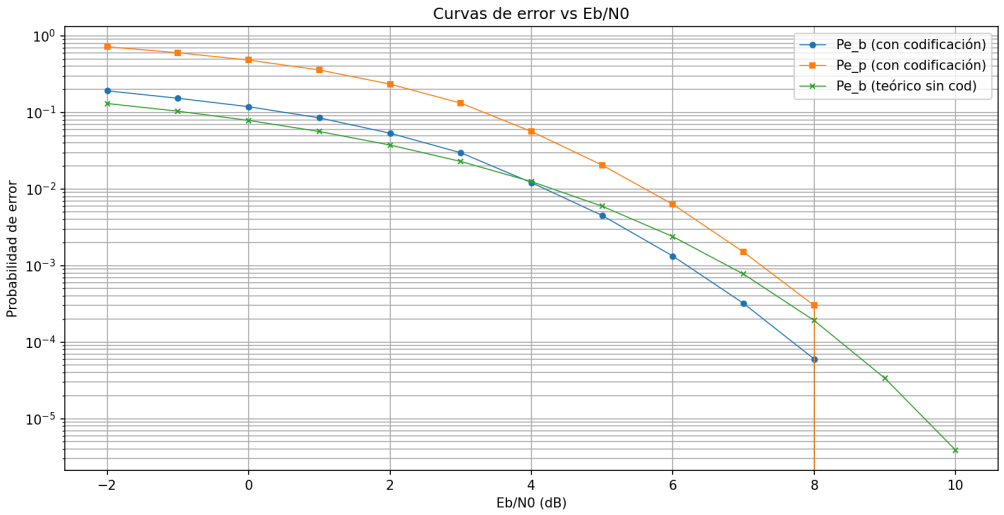

Figura 2: Curvas de error

Se puede ver en la Figura 2 los resultados de la simulación, cómo varían las probabilidades de error en función de la relación señal a ruido. La curva verde representa el *Peb* teórico sin codificación. Representa el rendimiento del sistema en un canal BPSK puro, sin introducir redundancia.

La curva azul muestra P_{eb} con codificación. Se observa por encima de la verde porque el uso de codificación distribuye la información en más bits y el canal introduce errores aleatorios. Sin embargo, a medida que E_b/N_0 aumenta, la codificación permite detectar y corregir errores, provocando que la curva azul descienda rápidamente, lo que implica que la codificación puede permitir alcanzar una probabilidad de error igual o menor a la que se lograría sin codificación, pero con menor energía de bit. De igual manera, se observa que decrece con el aumento de E_b/N_0, es decir que también tiene efectividad para mejorar la transmisión.

La curva naranja, representa P_{ep} representa con codificación, y refleja la probabilidad de error por palabra. Al igual que P_{eb} disminuye con el aumento de E_b/N_0. Se observa un descenso abrupto al alcanzar cierto umbral, indicando la eficacia del código para corregir a nivel de palabra. Esto se debe a que el código empleado tiene capacidad para corregir un número limitado de errores (uno por palabra). A partir de cierto umbral de E_b/N_0 cuando las condiciones del canal son mejores, el código logra corregir la mayoría de los errores rápidamente, haciendo que P_{ep} descienda rápidamente.

Ganancia asintótica

La ganancia asintótica G_a se da cuando la señal/ruido es infinita. Indica cuánta mejora en dB se logra en relación al sistema sin codificación. El resultado obtenido es

$$G_a = 1.429\ dB$$

lo cual es consistente para un código de bloque lineal (14,10), con capacidad de corrección de un error y decisión dura.

En la Figura 3 se compara la Ganancia real de código con la Ganancia asintótica *Ga*. Se observa que la ganancia real del código nunca supere la ganancia asintótica *Ga,* cumpliendo la condición teórica que establece *Ga* como límite superior de la mejora alcanzable. Esto se debe a que la ganancia asintótica representa el comportamiento ideal cuando la relación señal a ruido tiende a infinito (condiciones de canal perfectas). Por lo tanto, cualquier pico por encima de *Ga* observado en los gráficos reflejaría un error.

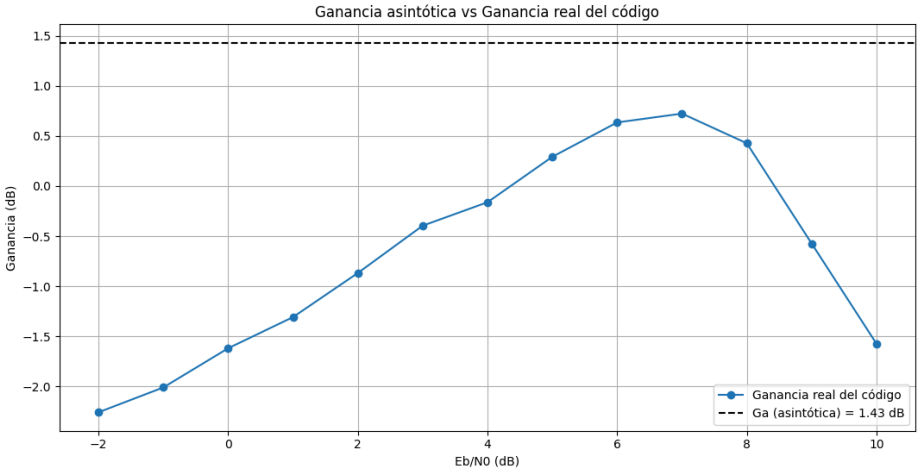

Figura 3: Ganancia asintótica vs real

Simulación de sistema de comunicación digital con codificación de fuente

En este ejercicio, se utilizó la codificación de Huffman para comprimir la imagen "*logo FI.tif*", debido a que el algoritmo permite reducir la cantidad de bits necesarios para representar la información sin pérdida.

En un principio, se transformó la imagen a una secuencia binaria y se asignaron códigos más cortos a los bloques de bits que se presentan con más frecuencia, para minimizar el tamaño total de la secuencia codificada. Esta etapa se realizó aplicando un umbral de 128 sobre los valores de intensidad. Los píxeles con valores mayores a 128 se consideran blancos (1) y los menores a 128 se consideran negros (0). Por último, estos valores fueron cargados a un vector unidimensional o una *secuencia de bits*.

Se calcularon las probabilidades para estimar la distribución de los bloques de bits de la imagen. En el código esto se logró recorriendo la secuencia binaria en bloques de longitud 2 o 3 y contando cuántas veces aparece cada bloque. Luego, se realizó el cálculo de la probabilidad dividiendo la cantidad de veces que aparece el bloque por el número total de bloques generador en la secuencia.

Se construyó el árbol de Huffman para determinar los códigos binarios correspondientes a cada bloque de bits. Este proceso se realizó mediante la creación de una cola de prioridades denominada *heap*, la cual contiene nodos, cada uno representando un bloque de bits y su probabilidad de aparición.

Luego se construyó el árbol combinando los dos nodos menos probables, formando un nuevo nodo cuyo símbolo es la concatenación de los anteriores y su probabilidad es la suma de las probabilidades de cada uno. Esto se repite hasta que quede un único nodo, el cual será la raíz del árbol.

Finalmente, se generaron los códigos binarios recorriendo el árbol desde la raíz. Se asignó '0' a la rama izquierda y '1' a la rama derecha, formando el código completo para cada bloque de bits en función del camino recorrido desde la raíz hasta cada hoja.

De esta forma, se obtuvo un diccionario de códigos de Huffman que asigna a cada bloque de bits (o símbolo) un código binario de longitud variable, más corto para los símbolos más frecuentes y más largo para los menos probables, permitiendo así, minimizar el largo promedio de la secuencia codificada y mejorando la compresión de la imagen.

Una vez generado el diccionario, se codificó la secuencia binaria original de la imagen recorriendo la secuencia, tomando bloques de longitud igual al orden, y reemplazarlos por su correspondiente código de Huffman.

Se hizo una compresión de lista a través de un recorrer la secuencia binaria original generando bloques de bits igual al orden elegido.

Cálculo de Largo Promedio y tasa compresión

A continuación, se calcularon el largo promedio L y la tasa compresión para evaluar la eficiencia de la compresión obtenida mediante Huffman.

En primer lugar, se calculó el largo promedio de la secuencia codificada, que representa el número medio de bits necesarios para codificar cada bloque de la fuente. Se obtuvo de la fórmula

$$\bar{L} = \sum_{k=0}^{K-1} p_k \, l_k$$

donde:

- p_k es la probabilidad del símbolo k
- l_k es la longitud del código asignado a dicho símbolo
- K es el número total de símbolos

En segundo lugar, se determinó la tasa compresión, que mide cuánto se logró reducir el tamaño de la secuencia original luego de aplicar el algoritmo de compresión. Se calcula como $Tasa\ compresión\ =\ largo\ original\ /\ largo\ codificado$

Conclusiones

Tabla 1: Resultados obtenidos en simulación

RESULTADOS OBTENIDOS		
FUENTE EXTENDIDA	Orden 2	Orden 3
CÓDIGO DE HUFFMAN	{'01': '000', '10': '001', '00': '01', '11': '1'}	{'111': '0', '110': '1000', '100': '1001', '001': '1010', '101': '101100', '010': '101101', '011': '10111', '000': '11'}
LARGO PROMEDIO	1.518	1.606
TASA COMPRESIÓN	0.659	0.623

En la Tabla 1, se observan los resultados obtenidos en la simulación. Al comparar, vemos que al aumentar el orden, la tasa de compresión de orden 3 es menor a la de orden 2,

por lo que se logra una mejor compresión, ya que se aprovechan patrones de mayor longitud. Por otro lado, el largo promedio del orden 3 es mayor al de orden 2, pero se compensa por la mayor longitud de los bloques, que permite codificar más bits por bloques.

Cabe destacar que la tasa de compresión y el largo promedio son magnitudes inversas, a medida que el largo promedio disminuye (mayor eficiencia del código), la tasa de compresión aumenta, indicando una reducción más efectiva del tamaño del archivo. Esta relación es especialmente evidente al comparar los resultados de los dos órdenes, el aumento del orden permite un mayor aprovechamiento de los patrones y, aunque incrementa el largo promedio, reduce el tamaño relativo de la secuencia codificada.

En conclusión, la codificación de Huffman con fuentes extendidas permite reducir significativamente el tamaño de la información, acercándose a los límites teóricos de compresión.