Estructuras y Características de Desempeño de los Códigos LDPC (*Low-Density Parity-Check Codes*)

Por:

Wilson Giovanni Castañeda Barbosa

Tesis sometida como requisito parcial
para obtener el grado de

MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE ELECTRÓNICA

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla
Octubre del 2013

Supervisada por:

Dra. Gordana Jovanovic Dolecek

©INAOE 2013
Derechos Reservados
El autor otorga al INAOE el permiso de reproducir y distribuir copias de esta tesis en su totalidad o en partes.

RESUMEN

Este trabajo presenta el análisis de las estructuras y características de desempeño de los códigos LDPC (*Low-Density Parity-Check Codes*), los cuales son una clase de códigos lineales de bloque que proporcionan un rendimiento cercano a la Capacidad utilizando decodificación iterativa.

En principio se presenta la idea general de un sistema de comunicaciones y por medio de un diagrama de bloques se muestra cómo está conformado tal sistema. Después se sigue con la explicación del concepto de codificación y la razón por la cual se necesita de ésta para la transmisión o almacenamiento de información. Además se dan las definiciones de algunos parámetros utilizados en este campo, incluyendo el gran aporte de Shannon con sus teoremas.

Posteriormente se realiza una revisión general de las familias de códigos de corrección de error presentes en la literatura, definiendo sus estructuras y diferencias. Luego de esta revisión se da una introducción a los códigos LDPC, en donde se explica la forma de codificación y los distintos algoritmos de decodificación para estos códigos.

Finalmente, se realiza el análisis del tipo de códigos LDPC basados en arreglos y se muestran los resultados de las simulaciones realizadas. Se presenta la definición del fenómeno de *error-floor* y conjuntos absorbentes los cuales afectan el rendimiento de los códigos LDPC en las regiones de bajo BER (*Bit-Error Rate*). Se estudia el efecto de la cuantización en el rendimiento de los decodificadores de los códigos LDPC basados en arreglos. Así mismo, basados en el estudio anterior se analiza el resultado de la variación de los pesos de las columnas de las matrices de comprobación de paridad para dichos códigos.

Al final de la tesis se incluyen los códigos de los programas realizados, junto con una breve descripción de ellos, y se detalla un programa interactivo basado en MatLab para la enseñanza de comunicaciones digitales.

ABSTRACT

This thesis presents the analysis of structures and performance characteristics of LDPC codes (Low-Density Parity-Check), which are a class of linear block codes that provide performance close to Capacity using iterative decoding.

We first present the overview of a communications system and by means of a block diagram we show how such system is formed. Then the concept of coding is explained, along with the reasons of why this is required for transmission or storage of information. Moreover, we give the definitions of some the parameters used in this field, including the great contribution of Shannon with his theorems.

Additionally, we perform a general review of the families of error correcting codes in the literature, defining their structures and differences. After that, an introduction to LDPC codes is presented, where the coding and decoding algorithms for these codes are explained.

Finally, the analysis of the array-based LDPC codes is performed and the results of the simulations are shown. It is introduced the definition of error-floor phenomenon and absorbing sets which affect the performance of LDPC codes in the regions of low BER (*Bit-Error Rate*). Quantization effects in the performance of decoders of array-based LDPC codes are also studied. Then, based on the previous study we analyze the result of variations of the column weights of the parity check matrices for these codes.

At the end of the thesis, the codes of the implemented programs are included and an interactive program based on Matlab for teaching of digital communications is described.

AGRADECIMIENTOS

En primer lugar quiero plasmar mis agradecimientos a Jehová Dios, por cada una de sus enseñanzas, por brindarme la salud y las capacidades para alcanzar cada una de mis metas, por ayudarme a superar cada una de las dificultades y los problemas que se me presentaron y permitirme estar aquí dando un paso más hacia el éxito.

Al INAOE por darme la oportunidad de realizar mis estudios de postgrado, por el apoyo y cada uno de los servicios brindados.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por haberme otorgado la beca de maestría.

Al programa de becas UC-MEXUS-CONACYT, por apoyo brindado y por la oportunidad de realizar esta investigación.

A la Dra. Gordana Jovanovic Dolecek, por la asesoría brindada, por cada uno de sus consejos, por permitirme hacer parte de este proyecto y darme la oportunidad de aprender de este mundo de la investigación.

A la Dra. Lara Dolecek, por abrirme las puertas de su grupo de investigación en La Universidad de California, Los Angeles (UCLA), por el apoyo y por brindarme los conocimientos necesarios para avanzar en el desarrollo de mi trabajo.

A mis sinodales, por hacer parte de la revisión de este trabajo, por sus valiosos comentarios y sugerencias, por su disposición y colaboración.

A mi mamá Mary Luz Barbosa Amaya, a mi papá Jaime Antonio Castañeda Ferrucho, a mi hermana Viviana Marcela Castañeda Barbosa, a mi hermano Jaime Andrés Castañeda Barbosa y a toda mi familia en Colombia por brindarme ese apoyo incondicional, por ayudarme a crecer como persona y hacerme sentir todo ese amor y cariño a pesar de la distancia. A mi novia Vanessa Calderón por brindarme su amor, apoyo, paciencia y por estar presente en cada uno de los momentos vividos durante esta etapa. A mis amigos en Colombia y mis amigos acá en México, Marlon, Sergio, Daniel, Jorge, Diego, Andrés, Jericó, Luis, Pablo, Sandra, Don Marco, Doña Fanny, Don Reyes y muchos más que se convirtieron en mi familia durante estos dos años.

Wilson Giovanni Castañeda Barbosa.

Octubre 2013.

A Jehová Dios

CONTENIDO

RESUMEN 2

ABSTRACT 3

AGRADECIMIENTOS 4

PREFACIO 8

Capítulo 1 10

INTRODUCCIÓN 10

1.1 Canales sin memoria con entrada binaria 13

Capítulo 2 24

CODIFICACIÓN 24

2.1 Comunicaciones punto a punto 25

2.2 Transmisión de información sobre canales con ruido 27

2.3 Parámetros de un código 29

2.4 Teorema de codificación del canal 33

2.5 Teorema de la capacidad de información 35

Capítulo 3 43

CÓDIGOS DE CORRECCIÓN DE ERROR 43

3.1 Códigos de Bloque 43

3.1.1 Códigos Hamming 45

3.1.2 Códigos Reed-Solomon 46

3.1.3 Los Códigos BCH 49

3.2 Códigos de secuencia 51

3.2.1 Códigos Convolucionales 52

Capítulo 4 56

CÓDIGOS LDPC (Low-Density Parity-Check Codes) 56

4.1 Corrección de error usando comprobaciones de paridad 57

4.2 Códigos LDPC 62

4.2.1 Codificación de los códigos LDPC 66
4.2.2 Decodificación de los códigos LDPC 70
Capítulo 5 89
ANÁLISIS DE LOS CÓDIGOS LDPC, GENERACIÓN DEL CÓDIGO, EFECTO DE CUANTIZACIÓN Y VARIACIÓN DEL PESO DE COLUMNA DE H 89
5.1 Códigos lineales de bloque 89
5.2 Códigos LDPC 91
5.2.1 Construcción del Código "Códigos LDPC basados en arreglos (*Array-based LDPC Codes*)" 93
5.2.2 *Error-floor* en los códigos LDPC 95
5.2.3 Esquema de Cuantización 98
5.2.4 Diseño del Código "Variación del peso de las columnas de H" 103
CONCLUSIONES Y TRABAJO FUTURO 107
APÉNDICES 111
A.1 Generación de Codewords 111
A.2 Algoritmo de decodificación Suma-Producto 112
A.3 Algoritmo de decodificación Min-Sum 114
A.4 Generación de Matrices de comprobación de paridad para los códigos LDPC basados en arreglos 117
A.5 Programa Interactivo basado en MatLab para la enseñanza de comunicaciones digitales 118
A5.1 Descripción del Programa 118
A.5.2 Diagrama de constelación 119
A.5.3 Técnicas de modulación digital 120
A.5.4 Ruido Blanco Aditivo Gaussiano (AWGN) 120
A.5.5 Teorema de capacidad de canal 121
A.5.6 Probabilidad de error 122
A.5.7 Modulación M-aria 123
LISTA DE FIGURAS 126
REFERENCIAS 128

PREFACIO

Desde el primer sistema de comunicaciones digital diseñado a principios de los años 40, el cual se basó en la modulación de pulsos codificados (PCM), se asumió que la información podía transmitirse eficientemente en forma digital sobre canales de comunicación afectados por ruido, solo si la transmisión no se viera comprometida por dichos canales. Es así que se creyó que los efectos del ruido únicamente podían contrarrestarse al aumentar la potencia de la señal transmitida, asegurando que la relación señal a ruido fuera lo suficientemente alta. Shannon con su trabajo revolucionario, cambio el punto de vista que en ese entonces tenían las comunicaciones digitales, mostrando que era posible transmitir datos digitales con una confiabilidad arbitrariamente alta, sobre canales afectados con ruido, lo cual se obtenía mediante el uso de la codificación de los datos digitales a transmitir, utilizando un código de corrección de error. Grandes esfuerzos de investigación en este campo han traído consigo importantes resultados, como lo son los códigos LDPC (*Low-Density Parity-Check Codes*) propuestos por Gallager en 1962 en su tesis doctoral; pero que solo recibieron la atención adecuada hasta finales de los años. Este tipo de códigos de bloque presentan excelente rendimiento y su uso en diferentes aplicaciones ha aumentado notablemente, su análisis y el diseño de sus decodificadores han atraído una gran atención actualmente.

El objetivo de esta tesis es analizar la estructura y característica de desempeño de los códigos LDPC basados en arreglos (*array-based LDPC codes*) realizando modificaciones a su matriz de comprobación de paridad para observar el efecto en el rendimiento del decodificador del código.

Para lograr el objetivo de la tesis, primero se estudiaron los bloques que conforman un sistema de comunicaciones digital, los cuales se presentan en el Capítulo 1, en donde se define cada bloque y su función dentro del sistema. En el Capítulo 2 se estudia específicamente el bloque de

codificación, en donde se explica la necesidad de la codificación y cómo surgió la investigación en este campo. En el Capítulo 3 se presentan algunos códigos de corrección de error, mostrando la estructura de cada uno de ellos. Posteriormente en el Capítulo 4 se realiza el estudio de los códigos LDPC, empezando con sus fundamentos, codificación y los tipos de algoritmos de decodificación utilizados para estos códigos.

En base al estudio realizado en los capítulos anteriores, en el Capítulo 5 se desarrolla el análisis de los códigos LDPC. Se inicia con la representación de los codewords pertenecientes a una familia específica de códigos de bloque. A continuación se inicia el estudio de los códigos LDPC basados en arreglos, presentando su estructura y los parámetros que definen este tipo de códigos. Se introduce los conceptos de *error-floor*, conjuntos absorbentes y se explica el efecto que producen en el rendimiento de los decodificadores iterativos. Posteriormente se presenta el análisis de la elección del esquema de cuantización para los mensajes transmitidos. Así mismo se realizan simulaciones del efecto de la variación del peso de columna en las matrices de comprobación de paridad.

Al final de la tesis se presentan las conclusiones y el trabajo futuro. Por último, en el Apéndice se incluyen las descripciones de las funciones realizadas en MatLab y la descripción de programa interactivo basado en MatLab para la enseñanza de comunicaciones digitales.

Capítulo 1

INTRODUCCIÓN

En el presente capítulo se dará una revisión de la idea general de un sistema de comunicaciones y por medio de un diagrama de bloques se mostrará cómo está conformado tal sistema, realizando el estudio de cada bloque. Por último se presentarán tres diferentes tipos de canales (BSC, BEC y BI-AWGN) y se estudiarán sus principales características.

Recientemente, se ha visto un aumento en la demanda por los sistemas de transmisión y almacenamiento de datos digitales que sean eficientes y confiables; esta demanda se ha acelerado por la emergencia de redes de datos de gran escala y de alta velocidad en cuanto el intercambio, procesamiento y almacenamiento de información digital en diferentes campos de aplicación. Por tal motivo, se presentó la necesidad de una fusión entre los sistemas de comunicaciones y la tecnología de la información, en donde una de las preocupaciones mayores del diseñador de un sistema es el control de los errores, de forma que los datos se puedan reproducir con una alta fiabilidad.

En 1948, Shannon en un documento trascendente e histórico [1], establece que por medio de una correcta codificación de la información, los errores provocados por un canal con ruido o algún medio de almacenamiento pueden reducirse a un nivel deseado sin la necesidad de sacrificar la tasa de transmisión o el almacenamiento de la información, siempre que la tasa de la información sea menor que la capacidad del canal. A partir del trabajo presentado por Shannon, el diseño de métodos eficientes de codificación y decodificación para el control de errores en un canal con ruido, se ha convertido en un tema de investigación importante. Investigaciones recientes han logrado contribuir en la confiabilidad requerida en los sistemas digitales de alta velocidad, siendo la codificación para el control de errores parte integral del diseño de los sistemas modernos de comunicación y almacenamiento digital.

La transmisión y el almacenamiento de información tienen mucho en común. En los dos procesos se transfieren datos desde una *Fuente de información* a un *destino* (o usuario) [2]. La Figura 1.1 representa un sistema característico de transmisión o almacenamiento.

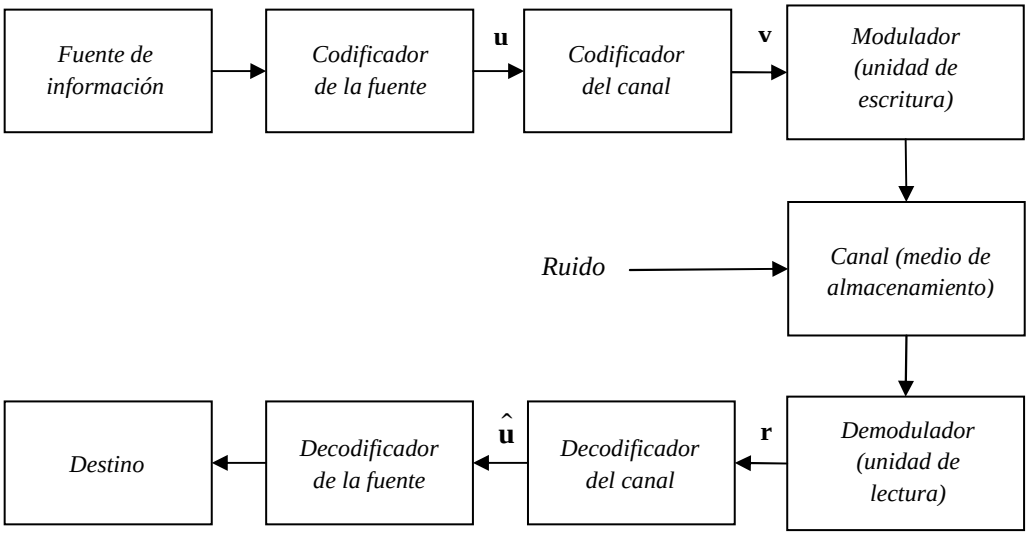


Figura 1.1 Sistema de comunicaciones típico.

La *Fuente de información* puede ser una persona o una máquina (un computador digital o un terminal de datos), mientras que la salida de la fuente puede ser una onda continua o una secuencia de símbolos discretos. El *Codificador de la fuente* (*source encoder*) convierte la salida de la fuente en una secuencia binaria de dígitos (bits) llamados la *secuencia de información* $\mathbf{u}$. En el caso de una fuente continua, el proceso involucra una conversión analógica-digital (A/D). El codificador de la fuente es idealmente diseñado de forma que (a) se minimiza el número de bits requeridos por unidad de tiempo que representan la salida de la fuente; y (b) la salida de la fuente puede ser inequívocamente reconstruida a partir de la secuencia de información $\mathbf{u}$. El *codificador del canal* transforma la secuencia de información $\mathbf{u}$ en una secuencia discreta codificada $\mathbf{v}$ conocida como *codeword*. En la mayoría de casos $\mathbf{v}$ es una secuencia binaria, aunque en algunas aplicaciones se utilizan códigos no binarios. La tarea de la codificación de canal es combatir al ambiente con

ruido, en el cual, los codewords serán transmitidos o en su caso almacenados. El *modulador* (o *unidad de escritura*) convierte cada símbolo de salida del codificador del canal en una forma de onda de duración T segundos, la cual es apropiada para la transmisión (o grabación); la forma de onda entra al *canal* (o *medio de almacenamiento*) y es afectada por el ruido. El *demodulador* (o *unidad de lectura*) procesa cada forma de onda de duración T segundos y produce una salida discreta (cuantizada) o una salida continua (no-cuantizada). La secuencia de salida del demodulador corresponde a la secuencia codificada $\mathbf{v}$, la cual es llamada la secuencia recibida $\mathbf{r}$. El *decodificador del canal* transforma la secuencia recibida $\mathbf{r}$ en una secuencia binaria $\widehat{\mathbf{u}}$ llamada la secuencia de información estimada. La estrategia de decodificación se basa en las reglas de codificación del canal y en las características del ruido del canal (o medio de almacenamiento). Idealmente la secuencia de información $\widehat{\mathbf{u}}$ estimada será una réplica de la secuencia de información $\mathbf{u}$, teniendo en cuenta que el ruido puede afectar y causar algunos errores de decodificación. El *decodificador de la fuente* convierte la secuencia de información estimada $\widehat{\mathbf{u}}$ en un estimado o una aproximación de la salida de la fuente y lo entrega al *destino*. En el caso de una fuente continua, el proceso presenta una conversión digital-analógica (D/A).

Este trabajo se enfoca específicamente en la codificación y decodificación de canal. Es posible combinar la fuente de información y el codificador de la fuente en una *fuente digital* con salida $\mathbf{u}$. De la misma forma el modulador (o unidad de escritura), el canal (o medio de almacenamiento) y el demodulador (o unidad de lectura) se combinan para tener una *codificación del canal* con entrada $\mathbf{v}$ y salida $\mathbf{r}$. Por último el decodificador de fuente y el destino pueden combinarse para obtener un sumidero digital con entrada $\widehat{\mathbf{u}}$. La Figura 1.2 muestra el diagrama de bloques que representa esta simplificación.

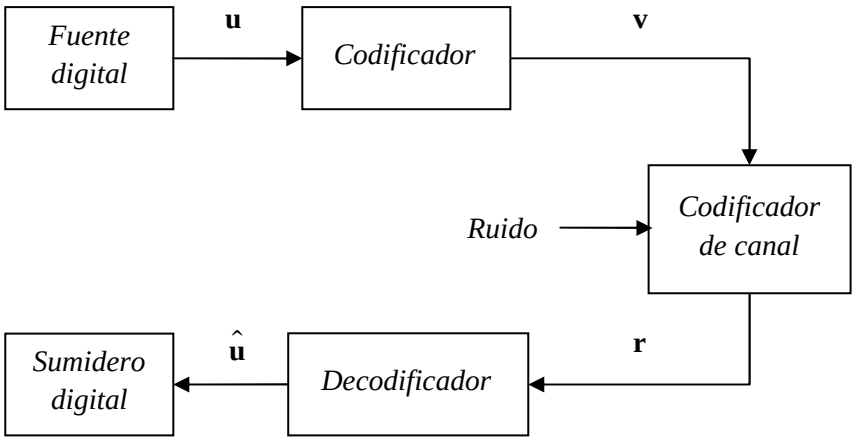


Figura 1.2 Modelo simplificado de un sistema de codificación.

Los códigos iterativos de corrección de error han encontrado una amplia aplicación en las comunicaciones celulares, radiodifusión digital, comunicaciones de espacio profundo y en redes LAN inalámbricas [3]. Este tratamiento autónomo de la corrección iterativa del error presenta todas las ideas claves necesarias para comprender, diseñar, implementar y analizar estos códigos poderosos.

Basados en el estudio de los bloques que conforman un sistema de comunicaciones digital, paralelo a este trabajo se desarrolló una herramienta basada en MatLab para la enseñanza y visualización de fundamentos de comunicaciones digitales [4].

A continuación se introducen las ideas básicas detrás de la corrección de errores. Es importante describir algunos de los canales frecuentemente utilizados para el análisis de un sistema de comunicaciones.

1.1 Canales sin memoria con entrada binaria

Un canal discreto es aquel en el cual se transmite un símbolo x de un conjunto discreto $\{X_1, X_2, \ldots, X_l\}$, conocido como el alfabeto fuente, y retorna un símbolo y de otro (posiblemente diferente) alfabeto discreto, $Y = \{Y_1, Y_2, \ldots, Y_m\}$. Un canal de entrada binaria transmite dos símbolos discretos, usualmente 0, 1, o 1, -1. Es posible que el canal entregue

símbolos binarios, pero igualmente puede dar salida a símbolos de un gran alfabeto discreto o un rango continuo de valores [3]. Desafortunadamente, los canales no siempre hacen el mapeo de un símbolo transmitido dado al mismo símbolo recibido (por lo cual es necesaria la corrección de error).

Un canal de comunicación puede ser modelado como un proceso aleatorio. Para un determinado símbolo x_i, el cual es transmitido en un tiempo i, tal que x_i es uno de los símbolos del conjunto X, es decir, $x_i = X_j \in \{X_1, X_2, \ldots, X_l\}$, la probabilidad de transición del canal $p(y|x) = p(y = Y_j | x = X_j)$ entrega la probabilidad que el símbolo retornado y_i en un tiempo i sea el símbolo Y_i del conjunto Y, es decir, $y_i = Y_j \in \{Y_1, Y_2, \ldots, Y_m\}$. Se dice que un canal es sin memoria si la salida del canal para cualquier instante de tiempo depende solamente de la entrada en ese instante de tiempo, no de símbolos previamente transmitidos. Más precisamente, para una secuencia de símbolos transmitidos $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ y símbolos recibidos $\mathbf{y} = [y_1, y_2, \ldots, y_N]$:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N} p(y_i | x_i). \tag{1.1}$$

Un canal sin memoria es por lo tanto completamente descrito por sus alfabetos de entrada y salida y la distribución de probabilidad condicional $p(y|x)$ para cada par entrada-salida.

Ejemplo 1.1

En el canal binario simétrico BSC (*Binary Symmetric Channel*) mostrado en la Figura 1.3 se transmite uno de dos símbolos, los dígitos binarios

$x \in \{0, 1\}$, y retorna uno de dos símbolos $y \in \{0, 1\}$. Este canal invierte el bit transmitido con una probabilidad de ϵ, es decir,

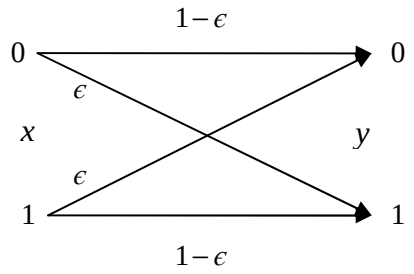


Figura 1.3 Canal binario simétrico (BSC).

con una probabilidad ϵ el símbolo de salida y entregado por el canal no es el símbolo que fue enviado y con probabilidad $1-\epsilon$, el símbolo y es el símbolo que fue enviado. El parámetro ϵ, es llamado la probabilidad cruzada (*crossover probability*) del canal. De esta forma se tiene que para el canal las probabilidades de transición son:

$$\begin{aligned} p(y=0|x=0) &= 1-\epsilon, \\ p(y=0|x=1) &= \epsilon, \\ p(y=1|x=0) &= \epsilon, \\ p(y=1|x=1) &= 1-\epsilon. \end{aligned} \qquad \text{(E.1.1.1)}$$

Un canal de entrada binaria es *simétrico* si ambos bits de entrada son afectados igualmente por el canal. Se puede ver fácilmente que el canal BSC es simétrico, debido a que $p(y=0|x=0) = p(y=1|x=1)$ y $p(y=0|x=1) = p(y=1|x=0)$.

En el decodificador, el símbolo y recibido del canal es usado para decodificar el símbolo x que fue enviado. En este caso el interés está en la probabilidad $p(x|y)$, es decir, dado que se recibe y ¿Qué tan

probable fue que x se envió? Se asume que cada bit tiene la misma probabilidad de ser transmitido. Así, para el canal binario simétrico, si $y=1$, la probabilidad que $x=1$ es la probabilidad que no ocurrió un error, es decir, $p(x=1|y=1)=1-\epsilon$, y la probabilidad que $x=0$ es la probabilidad de que el canal invirtió el bit transmitido $p(x=0|y=1)=\epsilon$. De la misma forma, $p(x=1|y=0)=\epsilon$ y $p(x=0|y=0)=1-\epsilon$.

Para una variable binaria x es fácil encontrar $p(x=1)$ dado $p(x=0)$, ya que $p(x=1)=1-p(x=0)$ y así solamente es necesario almacenar un valor de probabilidad para x. Las relaciones logarítmicas de funciones de verosimilitud LLR (*Log likelihood ratios*) son usadas para representar las métricas para una variable binaria por un único valor. La LLR está dada por:

$$L(x)=\log\frac{p(x=0)}{p(x=1)}, \tag{1.2}$$

donde en este documento se utiliza $\log$ para indicar el logaritmo en base e, o $\log_e$. Si $p(x=0)>p(x=1)$ entonces $L(x)$ es positivo. Además cuanto mayor sea la diferencia entre $p(x=0)$ y $p(x=1)$, es decir, tanto más seguros estamos que $p(x)=0$, el valor positivo para $L(x)$ es mayor. Por el contrario, si $p(x=1)>p(x=0)$ entonces $L(x)$ es negativo y, además, cuanto mayor sea la diferencia entre $p(x=0)$ y $p(x=1)$, el valor negativo para $L(x)$ es mayor. Por lo tanto el signo de $L(x)$ proporciona una decisión fuerte. Convirtiendo de regreso las LLR a probabilidades, se obtiene:

$$p(x=1)=\frac{p(x=1)/\,p(x=0)}{1+p(x=1)/\,p(x=0)}=\frac{e^{-L(x)}}{1+e^{-L(x)}} \tag{1.3}$$

y

$$p(x=1)=\frac{p(x=0)/\,p(x=1)}{1+p(x=0)/\,p(x=1)}=\frac{e^{L(x)}}{1+e^{L(x)}}. \tag{1.4}$$

Un beneficio de la representación logarítmica de las probabilidades es que cuando estas probabilidades necesitan ser multiplicadas, las razones de *log-likelihood* necesitan únicamente ser sumadas; esto puede reducir la complejidad de la implementación.

Ejemplo 1.2

Dado que las probabilidades $p(x|y)$ para el canal BSC son:

$$\begin{cases} p(x_i=1|y_i)=1-\epsilon \;\text{ y }\; p(x_i=0|y_i)=\epsilon & si\;\; y_i=1, \\ p(x_i=1|y_i)=\epsilon \;\text{ y }\; p(x_i=0|y_i)=1-\epsilon & si\;\; y_i=0, \end{cases} \tag{E.1.2.1}$$

las LLR recibidas para el $i-ésimo$ bit transmitido son:

$$\begin{aligned} R_i = L(x_i\,|\,y_i) &= \log\frac{p(x_i=0\,|\,y_i)}{p(x_i=1|\,y_i)} \\ &= \begin{cases} \log\epsilon/(1-\epsilon) & si\;\; y_i=1, \\ \log(1-\epsilon)/\epsilon & si\;\; y_i=0. \end{cases} \end{aligned} \tag{E.1.2.1}$$

Ejemplo 1.3

El canal binario de borrado BEC (*binary erasure channel*) el cual se muestra en la Figura 1.4 transmite uno de dos símbolos, usualmente dígitos binarios $x \in \{0, 1\}$. Sin embargo, el receptor, recibe el bit correcto o recibe un mensaje "e" de que el bit no se recibió (este fue borrado). El canal BEC borra un bit con probabilidad ε, llamada la probabilidad de borrado del canal. Así las probabilidades de transición de canal para el BEC son:

$$\begin{aligned} &p(y=0 \mid x=0)=1-\varepsilon, \\ &p(y=e \mid x=0)=\varepsilon, \\ &p(y=1 \mid x=0)=0, \\ &p(y=0 \mid x=1)=0, \\ &p(y=e \mid x=1)=\varepsilon, \\ &p(y=1 \mid x=1)=1-\varepsilon. \end{aligned} \tag{E.1.3.1}$$

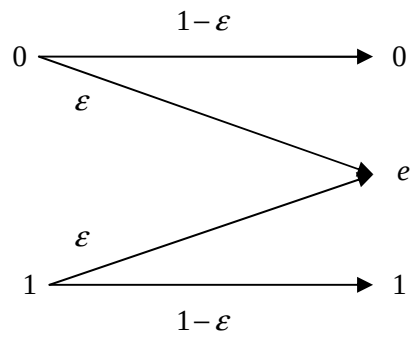


Figura 1.4 Canal de borrado binario (BEC).

El canal BEC no invierte los bits, de esta forma si y es recibida como un 1 o un 0 el receptor puede estar completamente seguro del valor de x:

$$
\begin{aligned}
&p\left(x=0 \mid y=0\right)=1,\\
&p\left(x=1 \mid y=0\right)=0,\\
&p\left(x=0 \mid y=1\right)=0,\\
&p\left(x=1 \mid y=1\right)\ =1.
\end{aligned}
\tag{E.1.3.2}
$$

Sin embargo, si el canal ha borrado el bit trasmitido el receptor no tiene información acerca de x y puede solamente usar las probabilidades a priori de la fuente. Si la fuente es equi probable (es decir, los bits 1 y 0 tienen la misma probabilidad de haber sido enviados) el receptor solo puede hacer una adivinanza del cincuenta-cincuenta:

$$
\begin{aligned}
&p\left(x=0 \mid y=e\right)=0.5,\\
&p\left(x=1 \mid y=e\right)\ =0.5.
\end{aligned}
\tag{E.1.3.3}
$$

Así que, para este canal se tiene que las LLRs recibidas para el $i-ésimo$ bit transmitido son:

$$
\begin{aligned}
R_i = L\left(x_i \mid y_i\right) &= \log\frac{p\left(x_i=0 \mid y_i\right)}{p\left(x_i=1 \mid y_i\right)}\\
&= \begin{cases}
\log\frac{0}{1}=-\infty & si\ \ y_i=1,\\
\log\frac{1}{0}=\infty & si\ \ y_i=0,\\
\log\frac{0.5}{0.5}=0 & si\ \ y_i=1.
\end{cases}
\end{aligned}
\tag{E.1.3.4}
$$

Ahora se considera el canal, el cual es comúnmente utilizado para la mayoría de estudios de la teoría de codificación, éste canal presenta una entrada binaria con ruido aditivo modelado como muestras de una distribución de probabilidad Gaussiana.

Ejemplo 1.4

El canal de entrada binaria con ruido aditivo blanco Gaussiano BI-AWGN (*Binary-Input Additive White Gaussian Noise*) puede ser descrito por la ecuación:

$$y_i = \mu x_i + z_i, \qquad \text{(E.1.4.1)}$$

donde $x_i \in \{-1,+1\}$ es el $i-ésimo$ símbolo transmitido, y_i es el $i-ésimo$ símbolo recibido y z_i es la muestra de ruido aditivo de una variable aleatoria Gaussiana con media 0 y varianza σ^2, la cual algunas veces se escribe como $z_i = AWGN(0,\ \sigma)$. La función de densidad de probabilidad para z es:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-z^2/2\sigma^2}. \qquad \text{(E.1.4.2)}$$

Cuando se transmite un código binario sobre el canal BI-AWGN, los bits del codeword $c_i \in \{0,\ 1\}$ pueden ser mapeados a los símbolos $x_i \in \{-1,\ +1\}$ en uno de dos caminos: $\{0 \to 1,\ 1 \to -1\}$ o $\{0 \to -1,\ 1 \to 1\}$. Aunque la convención tradicional es $\{0 \to 1,\ 1 \to -1\}$, debido a que la aritmética módulo-2 sobre $\{0,\ 1\}$ mapea directamente a la multiplicación sobre $\{-1,\ +1\}$ cuando se usa este mapeo.

Las LLR recibidas para el canal BI-AWGN son:

$$
\begin{aligned}
R_i = L(x_i \mid y_i) &= \log \frac{p(x_i = 0 \mid y_i)}{p(x_i = 1 \mid y_i)} \\
&= \log \frac{p(x_i = 1 \mid y_i)}{p(x_i = -1 \mid y_i)} \\
&= \log \frac{p(y_i \mid x_i = 1)\, p(x_i = 1) / p(y_i)}{p(y_i \mid x_i = -1)\, p(x_i = -1) / p(y_i)} \\
&= \log \frac{p(y_i \mid x_i = 1)\, p(x_i = 1)}{p(y_i \mid x_i = -1)\, p(x_i = -1)},
\end{aligned}
\tag{1.5}
$$

donde se ha utilizado la regla de Bayes

$$
p(x_i \mid y_i) = p(x_i, y_i) / p(y_i) = p(y_i \mid x_i)\, p(x_i) / p(y_i) \tag{1.6}
$$

para sustituir $p(x_i = 1 \mid y_i)$ y $p(x_i = -1 \mid y_i)$. Si la fuente es equiprobable, entonces $p(x_i = 1 \mid y_i) = p(x_i = -1 \mid y_i)$, y se tiene:

$$
Ri = L(x_i \mid y_i) = \log \frac{p(y_i \mid x_i = 1)}{p(y_i \mid x_i = -1)}. \tag{1.7}
$$

Para el canal BI-AWGN:

$$
p(y_i \mid x_i = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2} \right), \tag{1.8}
$$

$$
p(y_i \mid x_i = -1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i + \mu)^2}{2\sigma^2} \right); \tag{1.9}
$$

de esta forma

$$
\begin{aligned}
Ri = L(x_i \mid y_i) &= \log \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i-\mu)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i+\mu)^2}{2\sigma^2}\right)} \\
&= \log \exp\left(-\frac{(y_i-\mu)^2}{2\sigma^2} + \frac{(y_i+\mu)^2}{2\sigma^2}\right) \\
&= \frac{1}{2\sigma^2}\left(-\left(y_i^2 - 2\mu y_i + \mu^2\right) + \left(y_i^2 + 2\mu y_i + \mu^2\right)\right) \\
&= \frac{2\mu}{\sigma^2} y_i.
\end{aligned}
\tag{1.10}
$$

El valor de LLR para un bit c_i es algunas veces llamado la *decisión suave* para c_i. Una *decisión fuerte* para c_i entregará $c_i = 0$, lo cual es equivalente a $x_i = 1$, si R_i es positivo y $c_i = 1$, lo cual es equivalente a $x_i = -1$, si R_i es negativo.

Cuando se considera el nivel de ruido relativo de un canal BI-AWGN, es conveniente asumir que $\mu = 1$ y ajustar σ para reflejar la calidad del ruido del canal. En este caso R_i puede ser escrita como

$$
R_i = \frac{2}{\sigma^2} y_i. \tag{1.11}
$$

Frecuentemente el nivel de ruido se expresa como la razón de la energía por símbolo transmitido E_s, y la densidad espectral de potencia de ruido N_0:

$$
\frac{E_s}{N_0} = \frac{2\mu}{\sigma^2}, \tag{1.12}
$$

y (E.1.4.1) se escribe algunas veces en la siguiente forma

$$
y_i = \sqrt{E_s} x_i + z_i. \tag{1.13}
$$

Cuando se usa la codificación de corrección de error sobre un canal BI-AWGN, una fracción k de los bits transmitidos corresponde a los bits en el mensaje y los restantes son bits extra agregados por el código 'redundancia'. Para canales que usan corrección de error, el nivel de ruido es frecuentemente expresado como la razón de la energía por bit del mensaje, E_b y N_0. La relación Señal a Ruido *SNR* (*Signal-to-Noise Ratio*):

$$\frac{E_b}{N_0} = \frac{1}{K}\frac{E_s}{N_0} = \frac{1}{K}\frac{\mu^2}{2\sigma^2}, \tag{1.14}$$

y la LLR recibida es frecuentemente dada como

$$Ri = L\left(x_i \mid y_i\right) = 4\frac{\sqrt{E_s}}{N_0} y_i = 4\frac{\sqrt{KE_b}}{N_0} y_i, \tag{1.15}$$

o, cuando se asume $\mu = 1$,

$$Ri = \frac{4}{N_0} y_i. \tag{1.16}$$

La SNR también puede ser expresada en dB:

$$\frac{E_b}{N_0}(\mathrm{dB}) = 10\log_{10}\frac{E_b}{N_0} = 10\log_{10}\frac{\mu^2}{2\sigma^2}. \tag{1.17}$$

Capítulo 2

CODIFICACIÓN

En este capítulo se explicará el concepto de codificación y la razón por la cual se necesita de ésta para la transmisión o almacenamiento de información. Se darán las definiciones de algunos parámetros utilizados en este campo. Por último se presentan los teoremas de Shannon y se mostrará la importancia de estos en la teoría de los sistemas de comunicaciones.

La publicación de Shannon [1] planteó una pregunta importante: ¿Cómo se puede transmitir información de una forma confiable y eficiente?, a esta pregunta el mismo Shannon dio una respuesta sencilla: “mediante la codificación”.

El avance llegó con un cambio del paradigma fundamental en el área de la codificación que tuvo lugar a principios de los años 90. En la teoría de codificación moderna, los códigos son vistos como grandes sistemas complejos descritos con modelos gráficos dispersos y aleatorios (*random sparse graphical models*), y la codificación, así como la decodificación se obtienen mediante algoritmos locales eficientes. Las interacciones locales de los bit de código (*codebits*) son simples, pero el código en general es sin embargo complejo (y por lo tanto suficientemente potente como para permitir una comunicación confiable), debido al gran número de interacciones. La idea de los códigos aleatorios está en el espíritu de la formulación original de Shannon. Los que es nuevo es la diseminación de la descripción y la naturaleza local de los algoritmos. Debido a esto la mayoría de los sistemas de comunicaciones inalámbricos han ya adoptado la codificación moderna [5].

El objetivo principal de la codificación está enfocado en el problema clásico de Shannon. Se desea transmitir un *mensaje* a través de un *canal con ruido* de modo que el *receptor* tenga la capacidad de determinar el

mensaje con una *alta probabilidad* sin importar el ruido del canal. Por consiguiente, los esquemas de baja complejidad que introduzcan un pequeño retardo y permitan una transmisión confiable para el límite extremo (capacidad de Shannon) son los de mayor interés.

2.1 Comunicaciones punto a punto

La Figura 2.1 muestra el caso práctico con el cual se puede iniciar el análisis de los sistemas de comunicaciones más complejos, el escenario de las comunicaciones punto a punto. En este caso, una *fuente* transmite su información (voz, audio, datos, etc.) al *destino*. Se desea una transmisión confiable, es decir, la información transmitida debe llegar con la menor *distorsión* posible al destino (número de bits errados, error medio cuadrático de distorsión, etc.)

Figura 2.1 Problema básico de las comunicaciones punto a punto.

Shannon formalizó el problema de las comunicaciones y mostró que el problema punto-a-punto puede ser descompuesto en dos problemas separados como se muestra en la Figura 2.2. El *codificador de la fuente* convierte la fuente en una secuencia de bits. Idealmente, el codificador de fuente remueve toda la redundancia de la fuente de manera que la secuencia de bits resultante tiene el menor número posible de bits, y aun así, representa la fuente con la suficiente precisión. El *codificador de canal* procesa la secuencia de bits y le agrega redundancia. Esta redundancia es cuidadosamente elegida con el fin de mitigar el ruido introducido por el canal.

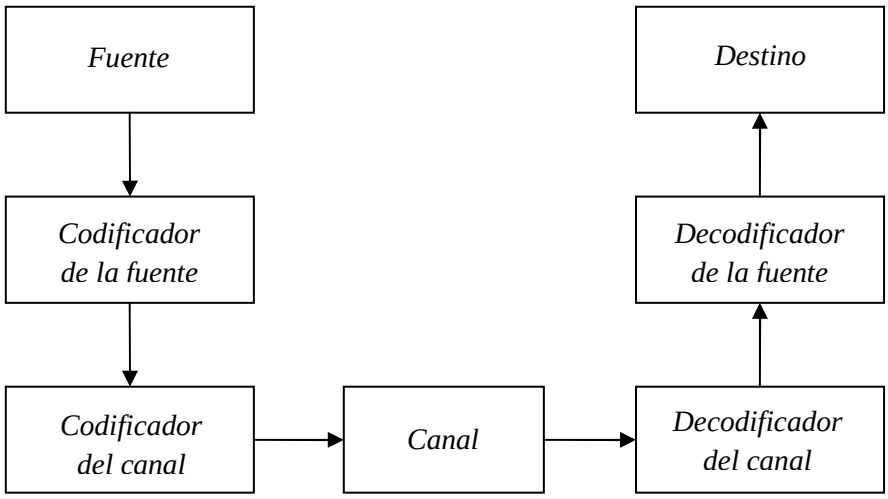


Figura 2.2 Problema básico de comunicaciones punto-a-punto desde el punto de vista del teorema de separación fuente-canal.

El teorema de codificación de fuente de Shannon afirma que, para una fuente y medida de distorsión dadas, existe una tasa mínima $R = R(d)$ (bits por símbolo de fuente emitidos), la cual es necesaria (y suficiente) para describir esta fuente con una distorsión que no exceda d. La gráfica de esta tasa R, como una función de la distorsión d, es usualmente llamada la curva *tasa-distorsión* (*rate-distortion*).

En la segunda etapa una cantidad apropiada de *redundancia* es agregada a estos bits de la fuente para protegerlos de los errores en el canal. Este proceso es llamado *codificación de canal* (*channel coding*). El teorema de codificación de canal de Shannon afirma la existencia de una tasa máxima (bits por uso de canal) para la cual la información transmitida es confiable, es decir, la disminución de la probabilidad de error sobre un canal dado. Esta tasa máxima es llamada la *capacidad* del canal y se denota por la letra C. En el receptor primero se decodifican los bits recibidos para determinar la información transmitida, luego se usan los bits decodificados para reconstruir la fuente. El teorema de *separación fuente-canal* de Shannon afirma que la fuente puede reconstruirse con una distorsión máxima d en el receptor, cuando $R(d) < C$; es decir, si la

tasa requerida para representar la fuente dada con la distorsión permitida es más pequeña que la capacidad del canal. Por el contrario, ningún esquema de codificación puede hacerlo mejor. Un gran beneficio del teorema de separación es que un enlace de comunicaciones puede ser usado por una gran variedad de fuentes: una buena solución de codificación de canal puede usarse con cualquier fuente. Actualmente todos los sistemas se basan en este principio.

Este trabajo no se enfoca al problema de la codificación de fuente y se considera que la codificación de fuente está realizada. La fuente emite una sucesión de bits independientes e idénticamente distribuidos, los cuales tienen la misma probabilidad de ser cero o uno. Bajo esta suposición, se verá cómo afrontar el problema de codificación de canal en una manera eficiente.

2.2 Transmisión de información sobre canales con ruido

Como ya se ha recalcado anteriormente, para transmitir información confiablemente sobre un canal con ruido a una tasa dada, es necesario agregar redundancia al mensaje, lo cual puede explotarse para combatir la distorsión introducida por el canal. Con el fin de aclarar algunos conceptos acerca de los canales con ruido presentados en el capítulo anterior, es necesario empezar con un caso especial como el que se presenta en el Ejemplo 2.1.

Ejemplo 2.1

Considerando nuevamente el canal simétrico binario con probabilidad cruzada ϵ de la Figura 1.3.

La entrada x y salida y en este caso son elementos del conjunto $\{\pm 1\}$. Un bit transmitido es recibido correctamente o es invertido; esto último ocurre

con una probabilidad ϵ, y los bits diferentes son invertidos o no invertidos independientemente. Considerando que $0<\epsilon<\frac{1}{2}$ sin pérdida de generalidad.

El canal BSC es el modelo genérico de un canal de entrada binaria sin memoria, en el cual se hacen decisiones fuertes en el extremo frontal del receptor, es decir, en donde el valor recibido es cuantizado en dos valores. A continuación se consideran dos casos específicos:

(a) Se considera la *transmisión sin codificar* sobre el canal BSC(ϵ). Se supone que los bits transmitidos son independientes y que $\mathbb{P}\{x=+1\}=\mathbb{P}\{x=-1\}=\frac{1}{2}$. Así que, se envían los bits de la fuente a través del canal sin agregar bits de redundancia. En el receptor se estima el bit transmitido x con base en la observación de y. Como se verá más adelante, la regla de decisión que minimiza la probabilidad de error de bit, llamada $\widehat{x}^{MAP}(y)$, es elegir el elemento del conjunto $\{\pm 1\}$, el cual maximiza $p_{X|Y}(x|y)$ para una y dada. Puesto que la probabilidad a priori sobre X es uniforme, una aplicación de la regla de *Bayes* muestra que esto es equivalente a maximizar $p_{Y|X}(y|x)$ para una y dada. Ya que $\epsilon<\frac{1}{2}$ concluimos que el estimador óptimo es $\widehat{x}^{MAP}(y)=y$. La probabilidad de que el estimado difiera del valor real, es decir $P_b=\mathbb{P}\{\widehat{x}^{MAP}(Y)\neq X\}$, es igual a ϵ. Dado que para cada bit de información que se quiere transmitir enviamos exactamente un bit sobre el canal, se dice que este esquema tiene *tasa* igual a 1. Esto permite concluir que con una transmisión sin codificar se puede alcanzar un par (tasa, $\mathrm{P_b}$) de (1, ϵ).

(b) Cuando la probabilidad de error ϵ es demasiado alta para una aplicación específica. La estrategia de transmisión más simple para disminuir esta probabilidad, es la codificación de repetición. Se asume que cada bit se repite k veces. Por simplicidad se elige k impar. Entonces, si X el bit a transmitirse, tiene valor x entonces la entrada al canal BSC(ϵ) es la k-tupla $x,\ldots,x$. Las k observaciones asociadas se denotan por $Y_1,\ldots,Y_k$. Es intuitivo y fácil de probar que el estimador que minimiza la probabilidad de error de bit está dado por la regla de mayoría

$$\widehat{x}^{MAP}(y_1,\ldots,y_k) = mayoría\{y_1,\ldots,y_k\}. \tag{2.1}$$

De lo anterior la probabilidad de error de bit está dada por

$$\begin{aligned} P_b &= \mathbb{P}\{\widehat{x}^{MAP}(Y) \neq X\} \overset{k \text{ odd}}{=} \mathbb{P}\{\text{al menos ocurran} \lceil k/2 \rceil \text{ errorres}\} \\ &= \sum_{i>k/2} \binom{k}{i} \epsilon^i (1-\epsilon)^{k-i}. \end{aligned} \tag{2.2}$$

Ya que por cada bit de información que se quiere transmitir se envían k bits sobre el canal, se dice que el esquema tiene una tasa de $\frac{1}{k}$. Así que con códigos de repetición se alcanzan los pares (tasa, $\mathrm{P_b}$), $\left(\frac{1}{k}, \sum_{i>k/2} \binom{k}{i} \epsilon^i (1-\epsilon)^{k-i}\right)$. Para que $\mathrm{P_b}$ se acerque a cero se tiene que elegir una k muy grande y como consecuencia de esto la tasa se acercara a cero a medida que incremente k.

2.3 Parámetros de un código

La información es inherentemente discreta, es natural y conveniente usar campos *finitos* para representarla. El campo más importante de este trabajo es el *campo binario* $\mathbb{F}_2$, el cual consiste de $\{0, 1\}$, con suma en

módulo dos (mod-2) y multiplicación en mod-2 $\left(0+0=1+1=0;\ 0+1=1;\ 0\cdot 0=1\cdot 0=0;\ 1\cdot 1=1\right)$. En otras palabras, al usar $\mathbb{F}_2$, la información se representa en términos de *bits* (secuencias de bits), una representación natural y conveniente para el propósito del procesamiento de datos. Se utiliza $\left|\mathbb{F}\right|$ para indicar el número de elementos del campo finito $\mathbb{F}$, por ejemplo $\left|\mathbb{F}_2\right|=2$. Como se verá más adelante, mediante el uso de las operaciones algebraicas en el codificador y en el decodificador se puede reducir significativamente la complejidad.

Definición 2.1 Un *código* C de *longitud* n y *cardinalidad* M sobre un campo $\mathbb{F}$, es una colección de M elementos de $\mathbb{F}^n$, es decir,

$$C\left(n,M\right)=\{x^{[1]},\ldots,x^{[M]}\},\ x^{[m]}\in\ \mathbb{F}^n,\ 1\leq m\leq M. \qquad (2.3)$$

Los elementos del código son llamados *codewords*. El parámetro n es llamado la *longitud del bloque*.

Ejemplo 2.2

Teniendo $\mathbb{F}=\mathbb{F}_2$. El *código de repetición* binario de longitud 3 se define como $C\left(n=3,\ M=2\right)=\{000,111\}$.

En este ejemplo se introducen los *códigos binarios*, es decir, códigos cuyos componentes son elementos de $\mathbb{F}_2=\left\{0,1\right\}$, pero algunas veces es más conveniente pensar en los dos elementos del campo como $\left\{\pm 1\right\}$. El mapeo estándar es $0\leftrightarrow 1$ y $1\leftrightarrow -1$. Es útil usar ambas notaciones.

Definición 2.2 La *tasa* de un código $C(n, M)$ es $R = \frac{1}{n}\log_{|\mathbb{F}|} M$. Es medida en *símbolos de información por símbolo transmitido*.

Definición 2.3 El conjunto de soporte de un codeword $x \in C$ es el conjunto de localizaciones $i \in [n] = \{1, \ldots, n\}$ tal que $x_i \neq 0$.

Definición 2.4 Considerando un código *binario* C, es decir, un código sobre $\mathbb{F}_2$. Se dice que un codeword $x \in C$ es *mínimo* si su conjunto de soporte no contiene el conjunto de soporte (distinto de cero) de cualquier otro codeword.

La distancia de Hamming y su derivación la distancia mínima de un código se introducen en las siguientes definiciones, son los parámetros centrales en toda la codificación clásica. Para nosotros ellas solo juegan un papel mínimo, así que no se hace una gran profundización de estas. Este es probablemente uno de los mayores factores que diferencian la codificación clásica y la codificación moderna.

Definición 2.5 El *peso de Hamming* de una palabra u, el cual se denota por $w(u)$, es igual al número de símbolos diferentes de cero en u, es decir, la cardinalidad del conjunto de soporte. La *distancia de Hamming* del par (u,v), la cual se denota como $d(u,v)$, es el número de posiciones en las cuales u difiere de v. Se tiene que $d(u,v) = d(u-v,0) = w(u-v)$, de tal forma que $d(u,v) = d(v,u)$ y $d(u,v) \geq 0$, con igualdad si y solo si $u = v$. Además, $d(.,.)$ satisface el *triángulo de desigualdad*:

$$d(u,v) \leq d(u,t) + d(t,v), \tag{2.4}$$

en donde las variables para la definición del peso y la distancia de Hamming $u, v, t \in \mathbb{F}^n$. En otras palabras, $d(.,.)$ es una *distancia* verdadera en el sentido matemático.

Definición 2.6 (Distancia mínima de un código). Siendo c un código, su *distancia mínima* $d(c)$ se define como:

$$d(c)=\min\{d(u,v):u,v\in c,u\neq v\}, \tag{2.5}$$

permitiendo que $x\in\mathbb{F}^n$ y $t\in\mathbb{N}$. Una *esfera* de radio t centrada en el punto x es el conjunto de todos los puntos en $\mathbb{F}^n$ que tienen una distancia a lo máximo t desde x. Si, para un código c con distancia mínima d, se colocan esferas de radios $t=\left\lfloor\frac{d-1}{2}\right\rfloor$ alrededor de cada codeword, entonces estas esferas están separadas. Esto se deduce del triángulo de desigualdad: si u y v son codewords y x es cualquier elemento en $\mathbb{F}^n$ entonces $d(c)\leq d(u,v)\leq d(u,x)+d(v,x)$. Si x está en la esfera de radio t alrededor de u entonces esto implica que $d(v,x)\geq d(c)-d(u,x)\geq\frac{d+1}{2}>t$. Es decir, x no está en la esfera de radio t alrededor de v. Además, por definición de d, t es el más grande de estos radios.

El radio t tiene un importante significado operacional que explica por qué la mayoría de la codificación clásica está centrada en la construcción de códigos con una gran distancia mínima. Concretamente, considerando el caso binario. Asumiendo un código $c(n,M,d)$, es decir, un código con M codewords de longitud n y de distancia mínima d, es transmitido sobre un BSC y asumiendo que se emplea un decodificador de *distancia limitada* con radio de decodificación t, $t\leq\left\lfloor\frac{d-1}{2}\right\rfloor$. Más precisamente, para una y dada, el decodificador elige $\widehat{x}^{BD}(y)$ definida por

$$\widehat{x}^{BD}(y)=\begin{cases}x\in c, \text{ si } d(x,y)\leq t,\\ \text{error}, \quad \text{si no existe tal } x,\end{cases} \tag{2.6}$$

donde por “error” el decodificador declara que no puede decodificarlo. Como se acaba de discutir, puede existir como *máximo una* $x \in c$ tal que $d(x,y) \leq t.$ Por lo tanto, si el peso del error no excede $t,$ entonces tal combinación encuentra la palabra correcta transmitida. A partir de esto, una gran t implica una gran resistencia en contra de los errores del canal. Permitiendo a $\delta = d/n$ denotar la distancia *normalizada* y considerando una tasa fija $r, 0 < r < 1,$

$$\delta^*(r) = \lim_{n \to \infty} \sup \max \left\{ \frac{d(C)}{n} : c \in \mathcal{C}\left(n, 2^{\lfloor nr \rfloor}\right) \right\}, \qquad (2.7)$$

donde $\mathcal{C}\left(n, 2^{\lfloor nr \rfloor}\right)$ denota el conjunto de todos los códigos de bloque binarios de longitud n que contienen al menos $2^{\lfloor nr \rfloor}$ codewords.

2.4 Teorema de codificación del canal

La presencia de ruido en el canal provoca errores entre las secuencias de datos de salida y entrada de los sistemas de comunicación digital. Si se tiene un canal de comunicación inalámbrico, como ejemplo de un canal con ruido; la probabilidad de error puede alcanzar un valor tan alto como $10^{-1},$ lo que quiere decir que en promedio sólo 9 de 10 bits transmitidos serán recibidos correctamente. Aunque en muchas aplicaciones, este nivel de *confiabilidad* es inadmisible. Probabilidades de error de 10^{-6} o menores, frecuentemente son un requisito necesario. Por este motivo, con el fin de alcanzar tales niveles de alto desempeño se recurre al uso de la *codificación de canal* [6].

Se utilizan los *códigos de bloque* para introducir algunas definiciones, los cuales se estudiaran más detalladamente en el capítulo siguiente. La secuencia del mensaje de este tipo de códigos se subdivide en bloques secuenciales cada uno de longitud de k bits, donde cada bloque de k bits

se mapea a un bloque de n bits, donde $n > k$. Así que, el número de bits de redundancia adicionados por el respectivo codificador a cada bloque transmitido es igual a $n-k$ bits. Por consiguiente la tasa de código R es:

$$R = \frac{k}{n} \tag{2.8}$$

Es obvio que para una k definida, la tasa del código R y la eficiencia de codificación del sistema, tienden a cero cuando la longitud de bloque n tiende a infinito.

Una reconstrucción exacta de la secuencia de la fuente original en el destino requiere que la *probabilidad promedio del error de símbolo* sea arbitrariamente baja. Se supone una fuente discreta sin memoria con alfabeto de fuente L y entropía $H(L)$ bits por símbolo fuente. A su vez la fuente emite símbolos una vez cada T_S segundos. Por lo tanto, la *tasa de información promedio* de la fuente es $H(L)/T_S$ bits por segundo. El decodificador entrega símbolos decodificados al destino provenientes del alfabeto fuente L a la misma velocidad de la fuente de un símbolo, cada T_S segundos. El canal discreto sin memoria tiene una capacidad igual a C bits por uso del canal. Se supone que el canal es capaz de utilizarse una vez cada T_C segundos. Así que, la capacidad del canal por unidad de tiempo es C/T_C bits por segundo, lo cual representa la tasa máxima de transferencia de información por el canal.

De esta manera, el *teorema de codificación de canal* (Segundo Teorema de Shannon) para un canal discreto sin memoria se enuncia en dos partes [6]:

a) Se considera una fuente discreta sin memoria con un alfabeto L que tiene entropía $H(L)$ y produce símbolos cada T_S segundos. Sea un

canal discreto sin memoria con capacidad C, que se usa cada T_c segundos. Si

$$\frac{H(L)}{T_s} \leq \frac{C}{T_c} \tag{2.9}$$

existe un esquema de codificación para el cual la salida de la fuente puede transmitirse a través del canal y reconstruirse con una probabilidad de error arbitrariamente pequeña. El parámetro C/T_c es llamado la *tasa crítica*. Cuando se satisface la ecuación (2.9) con el signo de igualdad, se dice que el sistema transmitirá señales a la tasa crítica.

b) Inversamente, si

$$\frac{H(L)}{T_s} > \frac{C}{T_c} \tag{2.10}$$

no es posible transmitir información por el canal y reconstruirla con una probabilidad de error arbitrariamente pequeña.

2.5 Teorema de la capacidad de información

Se recurre a la información mutua para formular el teorema de la capacidad de información correspondiente a canales gaussianos limitados en potencia y limitados en banda. Para ser más específicos, se considera un proceso estacionario de media cero.

Este teorema de la capacidad de información es complementario al Segundo Teorema de Shannon y se aplica a un canal en el cual el ruido tiene una característica gaussiana, este teorema es conocido como el Tercer Teorema de Shannon.

En la Figura 2.3 se considera el modelo de un *canal gaussiano de tiempo discreto sin memoria*. Las muestras de la señal transmitida y el ruido gaussiano blanco aditivo con media cero y densidad espectral de potencia $N_0/2$, se denotan como X_k y N_k, respectivamente.

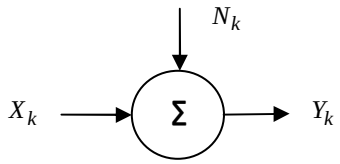


Figura 2.3 Modelo del canal gaussiano sin memoria en tiempo discreto.

donde la densidad espectral de potencia de ruido S_N se puede representar como se muestra en la Figura 2.4.

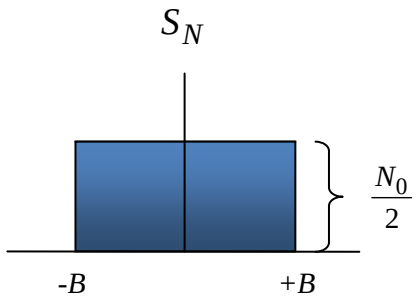


Figura 2.4 Densidad espectral de potencia del ruido AWGN.

La potencia promedio del sistema, la cual está dada por el área de la curva que representa la densidad espectral de potencia del ruido es:

$$P_N = 2B\frac{N_0}{2} = BN_0. \tag{2.11}$$

De este modo la capacidad de un canal con ancho de banda B [Hz], con densidad espectral de potencia de ruido AWGN $N_0/2$, está dada por

$$C = B\log_2\left(1+\frac{P}{P_N}\right)\frac{bit}{seg}. \tag{2.12}$$

P es la potencia promedio de la señal transmitida, P_N es la potencia del ruido y $\frac{P}{P_N}$ es la relación señal a ruido SNR (*Signal-to-Noise Ratio*).

Se puede notar que en la ecuación (2.12) se interrelacionan tres parámetros claves del sistema, el ancho de banda del canal, la potencia promedio transmitida (o equivalentemente, la potencia promedio de la señal recibida) y la densidad espectral de potencia del ruido a la salida del canal. La dependencia de la capacidad de información C del ancho de banda del canal B es lineal, mientras que la dependencia con respecto a la relación señal a ruido SNR es logarítmica. De este modo, es más fácil aumentar la capacidad de información de un canal de comunicación al expandir su ancho de banda, en lugar de incrementar la potencia transmitida para una potencia o varianza de ruido preestablecida.

El teorema implica que para una potencia promedio de la señal transmitida P y un ancho de banda de canal dado, se puede transmitir información a la tasa de C bits por segundo, como se define en (2.12), con una pequeña probabilidad de error mediante el uso de sistemas de codificación suficientemente complejos. No es posible transmitir información a una tasa superior a C bits/seg por medio de cualquier sistema de codificación. Por lo tanto, el teorema de la capacidad de canal define el *límite fundamental* sobre la tasa de transmisión libre de error para un canal Gaussiano limitado en banda y potencia. Sin embargo, para acercarse a este límite la señal transmitida debe tener propiedades estadísticas aproximadas a las del ruido blanco Gaussiano. El teorema de codificación de canal es el resultado más importante de la teoría de la información, especifica la capacidad del canal C como un límite fundamental sobre la velocidad a la cual puede ocurrir la transmisión de mensajes confiables sin errores por un canal discreto sin memoria. Sin embargo se debe tener en cuenta que el teorema de codificación de canal no indica cómo construir un buen código, si se satisface la ecuación (2.9), entonces existen buenos códigos. Además el teorema no tiene un resultado preciso para la probabilidad de error de símbolo después de decodificar la salida del canal. En su lugar, indica que la probabilidad de

error de símbolo tiende a cero cuando aumenta la longitud del código, siempre y cuando se satisfaga de nuevo la condición de la ecuación (2.9).

Por el momento se tiene una percepción intuitiva del teorema de la capacidad de información, se puede proseguir para explicar sus implicaciones en el contexto de un canal gaussiano que está limitado tanto en potencia como en ancho de banda. Sin embargo, en un ambiente útil, se necesita un marco de referencia ideal contra el cual se pueda valorar el desempeño de un sistema de comunicación práctico. Por este motivo se introduce la idea de un *sistema ideal* definido como aquel que transmite datos a una tasa de bits R_b igual a la capacidad de información C. Entonces es posible expresar la potencia promedio transmitida como,

$$P = E_b C, \tag{2.13}$$

donde E_b es la energía transmitida por bit. A partir de las ecuaciones (2.11), (2.12) y (2.13), la ecuación que define al sistema ideal es:

$$\frac{C}{B} = \log_2\left(1 + \frac{E_b}{N_0}\frac{C}{B}\right). \tag{2.14}$$

Equivalentemente, se puede definir la relación de la energía de la señal por bit a la densidad espectral de potencia del ruido E_b/N_0 en términos de la razón C/B para el sistema ideal como,

$$\frac{E_b}{N_0} = \frac{2^{C/B} - 1}{C/B}. \tag{2.15}$$

El *diagrama de eficiencia del ancho de banda* que se presenta en forma genérica en la Figura 2.5, muestra la eficiencia del ancho de banda R_b/B en función de E_b/N_0. Basándose en esta figura se puede observar que:

Para un ancho de banda infinito, la razón E_b/N_0, tiende al valor límite

$$\left(\frac{E_b}{N_0}\right)_\infty = \lim_{B\to\infty}\left(\frac{E_b}{N_0}\right) = \log 2 = 0.693. \qquad (2.16)$$

El valor de la ecuación (2.16) se le denomina como el *límite de Shannon* para un canal AWGN, suponiendo una tasa de código de cero. Este valor expresado en decibeles, es igual a -1.6 dB. El valor límite correspondiente de la capacidad se obtiene cuando el ancho de banda B del canal tiende a infinito, seguidamente se encuentra que:

$$C_\infty = \lim_{B\to\infty} C = \frac{P}{N_0}\log_2 e, \qquad (2.17)$$

donde e es la base del logaritmo natural.

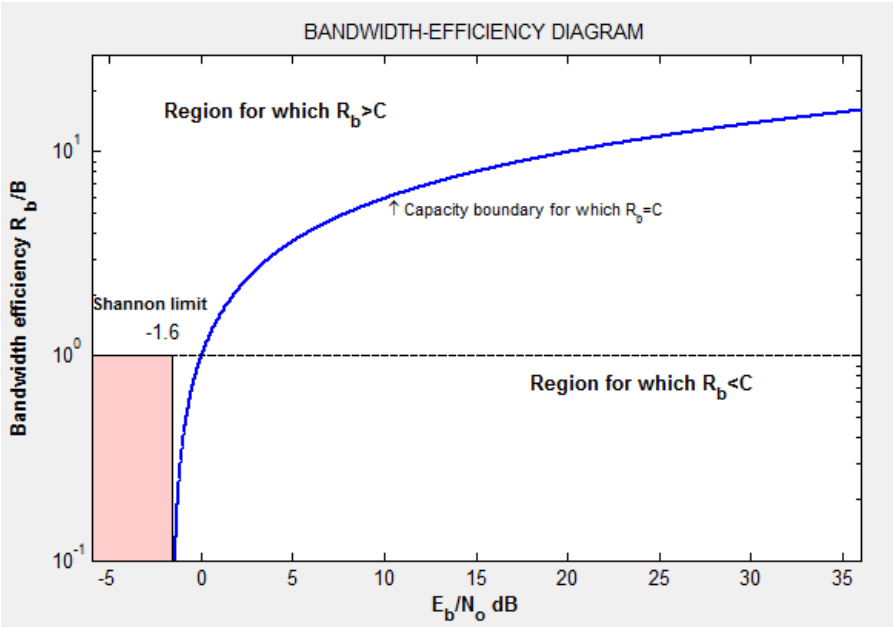


Figura 2.5 Diagrama de eficiencia del ancho de banda [4].

El *límite de la capacidad*, definido por la curva para la tasa de bits crítica $R_b = C$, separa las combinaciones de parámetros del sistema que tienen el potencial para soportar una transmisión libre de errores $(R_b < C)$ de las cuales no es posible este tipo de transmisión $(R_b < C)$.

Además, el diagrama subraya los compromisos potenciales entre E_b/N_0, R_b/B y la probabilidad de error de símbolo P_e. En particular, se observa el movimiento del punto de operación a lo largo de una línea horizontal como el intercambio de P_e en función de E_b/N_0 para una R_b/B fija. Por otra parte, es posible advertir el movimiento del punto de operación a lo largo de una línea vertical como el intercambio de P_e en función de R_b/B para una E_b/N_0 fija.

El límite de Shannon también puede definirse en términos de la E_b/N_0 requerida por el sistema ideal para que sea posible una transmisión libre de error. Específicamente, para el sistema ideal

$$P_e = \begin{cases} 0 & \left(E_b/N_0\right) \geq \ln 2 \\ 1 & \left(E_b/N_0\right) < \ln 2 \end{cases} \tag{2.18}$$

La Figura 2.6 describe el punto de vista de un sistema ideal. El límite entre la transmisión libre de error y la transmisión no confiable con posibles errores, definido por el límite de Shannon en la Figura 2.5, es análogo al límite de la capacidad en la Figura 2.6.

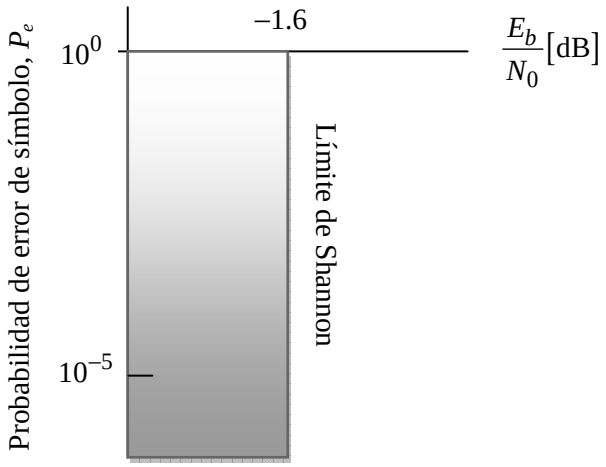


Figura 2.6 Diagrama de tasa de error.

En resumen los tres teoremas de Shannon proporcionan una información importante en los sistemas de comunicaciones. El *teorema de codificación de fuente*, primer teorema de Shannon, proporciona la herramienta matemática para valorar la compactación de datos, es decir, la comprensión sin pérdida de datos generados por una fuente discreta sin memoria, el teorema dice que es posible hacer que el número promedio de los elementos de un código binario (bits) por símbolo de fuente sea reducido, sin ser más pequeño que la entropía de la fuente medida en bits. La entropía de una fuente es una función de las probabilidades de los símbolos de la fuente que constituyen el alfabeto de la fuente. Ya que la entropía es una medida de incertidumbre, la entropía es máxima cuando la distribución de probabilidad asociada genera máxima incertidumbre. El *teorema de codificación de canal*, segundo teorema de Shannon, es el resultado más importante de la teoría de la información. Para un *canal binario simétrico*, el teorema dice que para cualquier tasa de código R menor o igual a la capacidad de canal C, existen códigos, tales que la probabilidad de error promedio es tan pequeña como se quiera. Un canal binario simétrico es la forma más simple de un canal sin memoria discreto. Es simétrico debido a que la probabilidad de recibir un 1 si un 0 fue enviado es igual a la probabilidad de recibir un 0 si un 1 fue enviado. La probabilidad de que ocurra un error es llamada *probabilidad de transición*. La probabilidad de transición p, es determinada no sólo mediante el ruido aditivo en el canal, sino también a partir del tipo de receptor usado. El valor de p únicamente define la capacidad del canal C. El *teorema de la capacidad de información*, tercer teorema de Shannon, dice que hay un máximo para la tasa o velocidad a la cual todos los sistemas de comunicaciones pueden operar confiablemente (es decir libre de errores) cuando el sistema es limitado en potencia. Esta velocidad máxima es llamada la *capacidad del canal*, medida en bits por segundo. Cuando el sistema opera a una velocidad superior a la capacidad del canal, está sentenciado a una alta probabilidad de error, sin importar la elección del

conjunto de señales usado para la transmisión o el receptor usado para el procesamiento de la señal recibida.

De este modo la tarea del diseñador de un sistema de comunicación digital es proveer un sistema rentable (*cost-effective*) para la transmisión de información desde un emisor (en una terminal del sistema) a una tasa y nivel de confiabilidad que son aceptables para un usuario (en el otro terminal del sistema). Los dos parámetros claves disponibles para el diseñador son la potencia de la señal transmitida y el ancho de banda del canal. Estos dos parámetros junto con la densidad espectral de potencia del ruido en el receptor, determinan la relación de la energía de la señal por la densidad espectral de potencia de bit-a-ruido. Consideraciones prácticas usualmente colocan un límite en el valor que se puede asignar a E_b/N_0. En consecuencia, en la práctica, frecuentemente se llega a un esquema de modulación y se encuentra que no es posible proporcionar una calidad de datos aceptable (es decir, un rendimiento de error suficientemente bajo). Para un E_b/N_0 fijo, la única opción practica disponible para cambiar de problemática a aceptable la calidad de los datos, es usar la *codificación*.

Por otro lado, otra motivación práctica para el uso de la codificación es reducir el E_b/N_0 para una tasa de error de bit fija. Esta reducción en E_b/N_0 puede, a su vez, ser explotada para reducir la potencia transmitida requerida o reducir los costos de hardware al exigir un tamaño de la antena más pequeño.

Capítulo 3

CÓDIGOS DE CORRECCIÓN DE ERROR

En este capítulo se realizará una revisión general de las familias de corrección de error presentes en la literatura. Además se presentarán sus estructuras y se indicará algunas de sus diferencias.

Los códigos de corrección de error ECC (*Error correcting codes*) son funciones que mapean una secuencia de datos *V* en una secuencia de datos *C*, usualmente más larga, conocida como la secuencia de código. Los ECC admiten funciones inversas capaces de recuperar la secuencia de datos *V*, independientemente de los posibles errores de transmisión en la secuencia *C*, siempre que la cantidad de símbolos con error no exceda la capacidad de corrección de error.

Típicamente, la codificación de una secuencia de datos puede realizarse de dos formas distintas: (a) codificación de bloque y (b) codificación de secuencia (*stream coding*). La primera consiste en una división de la secuencia de datos en bloques de longitud k y a cada uno de los cuales se le aplica una función de codificación que entrega bloques (*codewords*) de longitud N, estos codewords obtenidos se unen para formar la secuencia de código. Por otro lado, en la codificación de secuencia, la secuencia de datos es alimentada a una máquina de estados finita FSM (*Finite State Machine*), la cual da salida a uno o más símbolos de código para cada símbolo de datos de entrada.

3.1 Códigos de Bloque

En general, un código de bloque C es una función vectorial que asocia un vector de k elementos que pertenece a un conjunto A, a un vector de N elementos que pertenece a un conjunto L. Si se asume que la función es

inyectiva, la cardinalidad del conjunto de todos los codewords es igual a $\left[\mathrm{Card}(A)\right]^k$, donde $\mathrm{Card}(A)$ es la cardinalidad de A [7].

La tasa del código R_C, expresada en bits por unidad de símbolo, se define como:

$$R_C = \frac{K \log_2 \mathrm{Card}(A)}{N}. \tag{3.1}$$

Un código lineal de bloque sobre un campo finito de *Galois q-ario*, denotado como $\mathrm{GF}(q)$, es una función lineal vectorial, la cual asocia un vector k-dimensional $\boldsymbol{a}$ de elementos (el dato a ser codificado) en $\mathrm{GF}(q)$, a un vector N-dimensional $\boldsymbol{x}$ de elementos (el codeword) en $\mathrm{GF}(q)$, de acuerdo con la siguiente regla:

$$\boldsymbol{x} = G\boldsymbol{a}, \tag{3.2}$$

donde G es una matriz en $\mathrm{GF}(q)$, conocida como la matriz generación del código. Es posible demostrar que el conjunto de todos los codewords puede caracterizarse mediante la así llamada matriz de comprobación de paridad, usualmente denotada con H, la cual puede obtenerse mediante la manipulación apropiada de G. Un vector $\boldsymbol{x}$ es un codeword si y solo si

$$H\boldsymbol{x} = 0, \tag{3.4}$$

es decir, el conjunto de todos los codewords es definido como el espacio nulo de la matriz de comprobación de paridad H. En este trabajo solo se consideran códigos binarios; ya que en todos los casos los elementos de los datos y codewords estarán en $\mathrm{GF}(2)$.

Varias familias de códigos importantes pertenecen al conjunto de los códigos lineales de bloque, entre los cuales se tienen:

- Códigos cíclicos, que comprende los códigos Bose-Chaudhuri-Hocquenghem (BCH), los códigos Reed-Solomon, los códigos Hamming;
- Códigos LDPC.

Existen varias técnicas para la decodificación de los códigos lineales de bloque. En particular, es posible tener dos enfoques: (a) decodificación de salida-fuerte, la cual da salida a valores estimados de los codewords o de los símbolos de dato, y (b) decodificación de salida-suave, la cual da salida a las funciones de verosimilitud (*likelihood*) de cada codeword o símbolo de dato, es decir, un estimado de cuanto es la probabilidad de que un codeword o un símbolo de dato asuma cada valor posible. En este trabajo se considera solamente la decodificación de salida-suave, ya que este tipo de decodificación conduce a la construcción de receptores complejos basados en bloques de decodificación de entrada-suave, salida-suave más simples.

3.1.1 Códigos Hamming

Los códigos Hamming son códigos lineales de bloque binarios que pueden corregir un error o detectar dos errores.

Los códigos Hamming siempre tienen m bits de paridad $(m \geq 3)$ y $2^m - m - 1$ bits de mensaje. Por lo tanto, la tasa de código de los códigos Hamming es $\left(2^m - m - 1\right) / \left(2^m - 1\right)$. Todos los códigos Hamming tienen una distancia mínima de $d_{\min} = 3$. La matriz de comprobación de paridad H de un código Hamming se construye al tomar todos los vectores binarios diferentes de cero $2^m - 1$ de longitud m como las columnas de la matriz. En forma sistemática, las columnas de H están organizadas en la siguiente forma:

$$H = \left[\mathrm{I_m} \quad \mathrm{Q} \right], \tag{3.5}$$

Donde $\mathrm{I_m}$ es una matriz identidad $m \times m$, y la sub-matriz Q consiste de $2^m - m - 1$ columnas que son las m-tuplas o vectores de peso 2 o más.

La distancia mínima de un código Hamming es exactamente 3. Por tal motivo, el código es capaz de corregir todos los patrones de error con un solo error, o detectar todos los patrones de error de dos o menos errores.

3.1.2 Códigos Reed-Solomon

Los códigos Reed-Solomon (RS) fueron inventados en 1960 y siguen siendo una de las clases de código más aplicable y a pesar de su simplicidad o quizás debido a esto, son la base de las nuevas generaciones de códigos. El rango de una matriz de comprobación de paridad H del código $(n,\ n\text{-}r)$ con distancia mínima d no es menor que d-1 ya que cualquiera de sus d-1 columnas son linealmente independientes. Por otro lado, el rango de la matriz no es mayor que el número de filas, y por lo tanto para cualquier código $(n,\ n\text{-}r)$ se cumple que:

$$d \leq r + 1. \tag{3.6}$$

El límite dado en (3.6) se cumple para cualquier código (no necesariamente lineal)

$$M_q(n,\ d) \leq q^{n-d-1}. \tag{3.7}$$

El código que alcanza el límite (3.7) es un código óptimo y es llamado un *código separable con distancia máxima* MDS (*Maximum Distance Separable code*). Así que, en la matriz de comprobación de paridad $r \times n$ de un código MDS todas las r columnas son linealmente independientes. El ejemplo de este tipo de matriz de comprobación de paridad es la matriz $\mathrm{H} = \left[h_{ij}\right] = \left[\alpha_j^i\right],\ i = 0, \ldots,\ r-1,\ j = 0, \ldots,\ n-1,$ donde $\alpha_0, \ldots, \alpha_{n-1}$ son elementos distintos del campo $F_q(n \leq q)$. En otras palabras, la $i-ésima$

fila de la matriz H consiste de los valores del término z^i calculados en los puntos $\alpha_0,\ldots,\alpha_{n-1}$ del campo F_q:

$$\mathrm{H}_r = \begin{array}{c} 1 \\ z \\ \vdots \\ z^{r-1} \end{array} \left\| \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_0^{r-1} & \alpha_1^{r-1} & \cdots & \alpha_{n-1}^{r-1} \end{bmatrix} \right. . \tag{3.8}$$

Ahora se verá que todas las r columnas de (3.8) son linealmente independientes. Inicialmente, se supone que este no es el caso, y las columnas $\mathrm{h}_{j1},\ldots,\mathrm{h}_{jr}$ son linealmente dependientes. Esto significa que el $\det\left(\mathrm{h}_{j1},\ldots,\mathrm{h}_{jr}\right)=0.$ Por lo tanto, las filas de su determinante menor $\mathrm{H}_{j1,\ldots,jr}$ también son linealmente dependientes.

La dependencia lineal de las filas de este determinante menor con coeficientes $\lambda_0,\ldots,\lambda_{r-1}$ significa que el polinomio $\lambda_0+\lambda_0\cdot z+\ldots\lambda_{r-1}\cdot z^{r-1}$ de menor grado que r tiene r diferentes raíces $\mu_{j1},\ldots,\mu_{jr}$, y esto es imposible.

A partir del enunciado sobre la matriz (3.8) es fácil de deducir que en matrices $\mathrm{H}_{a,r}$ de forma más general con las filas definidas por los valores del termino $z^j\left(j=a,a+1,\ldots,a+r-1\right)$ en los elementos $\alpha_j\in F_q\setminus 0$:

$$\mathrm{H}_{a,r} = \begin{bmatrix} \alpha_0^a & \cdots & \cdots & \alpha_{n-1}^a \\ \vdots & \cdots & \cdots & \vdots \\ \alpha_0^{a+r-1} & \cdots & \cdots & \alpha_{n-1}^{a+r-1} \end{bmatrix}, \tag{3.9}$$

todas las r columnas son linealmente independientes. Los códigos definidos por las matrices de la forma presentada en (3.9) son conocidos como los códigos (RS) *Reed-Solomon*. Frecuentemente los códigos RS

son definidos como códigos cíclicos. Los elementos $\alpha_0,\ldots,\alpha_{n-1}$ forman un subgrupo. Este subgrupo debe ser cíclico debido a que el grupo bajo la multiplicación de todo el campo finito es el grupo cíclico. Por lo tanto, existe un elemento $\beta \in F_q$ tal que $\alpha_i = \beta^i, i = 0,\ldots,n-1.$ De esta manera, la matriz (3.9) puede convertirse en la forma:

$$\mathbf{H}_{a,r,\beta} = \begin{bmatrix} 1 & \beta^a & \cdots & \beta_{n-1}^{a(n-1)} \\ 1 & \beta^{a+1} & \cdots & \beta^{(a+1)(n-1)} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \beta^{a+r+1} & \cdots & \beta_{n-1}^{(a+r-1)(n-1)} \end{bmatrix} \tag{3.10}$$

Ya que el desplazamiento cíclico de cualquiera de las filas de la matriz $\mathrm{H}_{a,r,\beta}$ (este desplazamiento puede obtenerse mediante la multiplicación de las filas por $\beta^{a(n-1)}$) pertenece al espacio lineal de las filas de esta matriz, el código definido por la matriz $\mathrm{H}_{a,r,\beta}$ es un código cíclico. La condición para que el vector $\mathbf{v}$ sea codeword del código definido por la matriz $\mathrm{H}_{a,r,\beta}$ es la siguiente

$$\mathbf{H}_{a,r,\beta} \cdot \mathbf{v} = \mathbf{0}, \tag{3.11}$$

lo cual es equivalente a:

$$v\left(\beta^j\right) = 0, \;\; j = a,\ldots,a+r-1, \tag{3.12}$$

donde $v(\mathrm{x})$ es el polinomio correspondiente al vector $\mathbf{v}$. Por lo tanto, el polinomio generador de este código es:

$$g(x) = \mathrm{M.C.M}\left\{ f_{\beta^j}(x) \right\}, \;\; j = a,\ldots,a+r-1, \tag{3.13}$$

donde $\mathrm{M.C.M}$ es el mínimo común múltiplo y $f_\gamma(x)$ es el polinomio mínimo del elemento γ. En este caso se considera $f_\gamma(x)=x-\gamma$ y por lo tanto:

$$g(x)=\coprod_{j=a}^{a+r-1}\left(x-\beta^j\right). \tag{3.14}$$

El caso más interesante es cuando β es el elemento primitivo del campo F_q. Entonces $n=q-1$ y este tipo de códigos con mayor frecuencia son llamados códigos RS. El código definido por la matriz (3.8) tiene longitud q cuando $\{\alpha_0,\ldots,\alpha_{n-1}\}=F_q$ y es conocido como *código RS extendido-1*. Es posible extender este tipo de código por un símbolo más si se agrega ∞ al conjunto $\{\alpha_0,\ldots,\alpha_{n-1}\}=F_q$ con definición formal de $\infty^j=0$ para $j=0,1,\ldots,r-2$ y $\infty^{r-1}=1$.

3.1.3 Los Códigos BCH

Para una estimación de la distancia mínima la siguiente descripción de los códigos cíclicos es muy útil. Siendo $g(x)$ el polinomio generador del código cíclico $(n,\ k)$ V sobre el campo F_q. Se asume un código no trivial, esto es, $2<d<n$ y por lo tanto, n es el periodo del polinomio $g(x)$, esto es, $g(x)|x^n-1$ y $g(x)$ no divide $x^{n'}-1$ para $n'<n$. Siendo el código de longitud n y la característica p del campo F_q primos relativos. Entonces el polinomio x^n-1 no tiene raíces repetidas ya que el mínimo común denominador, $MCD\left(x^n-1,\frac{d}{dx}\left(x^n-1\right)\right)=1$. Por lo tanto, el polinomio $g(x)$ tampoco tiene raíces repetidas debido a que es el divisor del polinomio x^n-1. Siendo m el entero positivo mínimo tal que n divide a q^m-1.

Entonces F_{q^m} es el campo mínimo que contiene todas las raíces de $g(x)$. El siguiente teorema llamado límite BCH sostiene la verdad.

Teorema del límite BCH. Siendo α un elemento del campo F_{q^m} y $\alpha^a, \alpha^{a+1}, \ldots, \alpha^{a+s-1}$ las raíces del polinomio $g(x)$. Entonces la distancia mínima del código cíclico V con polinomio generador $g(x)$ no es más pequeña que $s+1$.

Prueba. Cualquier codeword $\mathbf{v} \in V$ puede considerarse como un polinomio $v(x)$ y los elementos $\alpha^a, \alpha^{a+1}, \ldots, \alpha^{a+s-1}$ son las raíces de este polinomio debido a que $g(x)$ es el divisor de $v(x)$, esto significa que $v\left(\alpha^j\right) = 0$ para $j = a, \ldots, a+s-1$. De acuerdo con (3.12) esto significa que el codeword $\mathbf{v}$ pertenece al código RS cíclico con distancia mínima $s+1$.

Frecuentemente este teorema es usado para el caso particular cuando α es el elemento primitivo del campo F_{q^m}. La asignación correspondiente dice que la distancia mínima de un código cíclico es mayor a la longitud del máximo “conjunto de raíces consecutivas”, esto es, el número máximo de potencias consecutivas $j_0, \ldots, j_{s-1}$, donde $\alpha^{j_0}, \ldots, \alpha^{j_{s-1}}$ son raíces del polinomio $g(x)$. El parámetro $s+1$ es llamado usualmente la *distancia diseñada* de un código BCH.

Existen una gran variedad de códigos de bloque, en este trabajo solo se da la definición y se presentan las características principales de algunos de ellos, ya que el centro de nuestro estudio son específicamente los códigos LDPC. Si se desea realizar un estudio más profundo puede consultar [8] y [9], cualquier libro de codificación de canal o tema relacionado.

3.2 Códigos de secuencia

La codificación de secuencia es una técnica de corrección de error que se aplica a secuencias de datos, en lugar de bloques de datos. La longitud del codeword es ilimitada, de modo que un codificador de secuencia típicamente funciona sobre una base de símbolo a símbolo, dando salida a un símbolo codificado por cada símbolo de dato en su entrada. Codificadores de secuencia prácticos están implementados por medio de FSM. En particular, una FSM es definida mediante dos funciones: (a) la *función de salida* y (b) la *función del estado siguiente*. Estas dos funciones están matemáticamente descritas como sigue:

$$c_k = f_o\left(a_k,\ s_k\right), \tag{3.15}$$

$$s_{k+1} = f_{ns}\left(a_k,\ s_k\right). \tag{3.16}$$

donde c_k denota el símbolo de salida para cada k, s_k denota el estado para cada k, a_k es el símbolo de datos de entrada para cada k, y $f_o(\cdot,\ \cdot)$ y $f_{ns}(\cdot,\ \cdot)$ denotan la función de salida y la función del estado siguiente, respectivamente. La secuencia de salida es formada por la secuencia $\{c_k\}$. Sin pérdida de generalidad, se supone que la secuencia de código tiene la misma longitud de la secuencia de información, la redundancia introducida por el codificador es explicada por la ampliación de la cardinalidad de símbolo.

La codificación lineal de secuencia es la técnica más popular de este tipo de codificación para la corrección de errores. Tales códigos son también conocidos como *códigos convolucionales* ya que la secuencia de salida puede verse como la convolución de la secuencia de datos de entrada con una secuencia de “respuesta al impulso”. La convolución se lleva a cabo utilizando algebra de campos finitos.

La decodificación de códigos de secuencia, de la misma forma que los códigos de bloque, puede realizarse mediante algoritmos de decodificación de salida-fuerte o salida-suave. El algoritmo de decodificación más importante para los códigos de secuencia es el bien conocido algoritmo de Viterbi (VA). El VA es óptimo, en el sentido que este calcula la secuencia de datos transmitida más probable dada la secuencia observable recibida. Las técnicas de decodificación/detección suaves para los códigos de secuencia comprenden el algoritmo de Viterbi de salida suave (SOVA por sus siglas en ingles *soft-output Viterbi algorithm*).

En general, en la codificación de secuencia no es posible calcular exactamente la probabilidad a posteriori (APP) de los símbolos de datos transmitidos (posiblemente infinitos). Sin embargo, el cálculo de la APP es posible, si se utiliza el codificador de secuencia para codificar una secuencia de longitud-finita, por medio del famoso algoritmo Bahl-Cocke-Jelinek-Raviv (BCJR), también conocido como el algoritmo forward backward (FB) [9].

3.2.1 Códigos Convolucionales

Los códigos convolucionales fueron introducidos por Elias en 1955 como una alternativa a los códigos de bloque. Él probo que es posible agregar redundancia a una secuencia de información a través del uso de registros de desplazamiento lineales. Wozencraft y Reiffen propusieron la decodificación secuencial como un método de decodificación eficiente para los códigos convolucionales con restricciones de longitudes grandes, y de esta forma estudios experimentales rápidamente empezaron a aparecer. En 1967 Viterbi propuso un algoritmo de decodificación de máxima verosimilitud (ML *Maximum likelihood*) que fue relativamente fácil de implementar para la decodificación de decisión-suave de los códigos convolucionales con restricciones de longitudes pequeñas. El algoritmo de Viterbi, junto con las versiones de decisión-suave de decodificación secuencial, llevo a la aplicación de los códigos convolucionales a

sistemas de comunicación de espacio-profundo y satelital en los los años 70. La importancia del algoritmo de Viterbi es que demostró ser relativamente fácil de implementar, dado que el codificador tiene un número pequeño de elementos de memoria. En 1974, Bahl, Cocke, Jelinek y Raviv (BCJR) introdujeron un algoritmo de decodificación de máxima probabilidad a posteriori (MAP) para los códigos convolucionales con probabilidades a priori desiguales para los bits de información. El algoritmo BCJR ha sido aplicado en años recientes para esquemas de decodificación iterativos de decisión-suave, en los cuales las probabilidades a priori de los bits de información cambian de iteración a iteración. En 1976 Ungerboeck y Csajka introdujeron el concepto modulación de trellis-codificada utilizando restricciones de longitud corta para códigos convolucionales de tasa $R = k/(k+1)$ junto con la decodificación Viterbi de decisión-suave. Estos esquemas forman la base para alcanzar una transmisión digital de alta-velocidad sobre canales telefónicos de banda-limitada. El trabajo de Ungerboeck generó interés en las propiedades de los codificadores de retroalimentación para los códigos convolucionales, ya que los códigos con alta tasa están representados más eficientemente en la forma de retroalimentación sistemática. Otra aplicación de los codificadores convolucionales de retroalimentación sistemática, es la llamada *codificación turbo*, la cual fue introducida por Berrou, Glavieux y Thitimajshima en 1993. Este esquema de codificación, el cual combina una concatenación paralela de dos codificadores de retroalimentación sistemáticos con una decodificación iterativa MAP, la cual es capaz de alcanzar moderadamente bajos BER, alrededor de 10^{-5} en un SNR cerca al límite de Shannon [2].

Los códigos convolucionales difieren de los códigos de bloque en que el codificador contiene memoria, y la salida del codificador en cualquier unidad de tiempo dada no solo depende de las entradas en esta unidad de tiempo, sino que también de algún número de entradas previas. Una tasa $R = k/n$ de un código convolucional con memoria de orden m puede

realizarse como un circuito secuencial lineal de k-entradas, n-salidas con memoria de entrada m; esto es, las entradas permanecen en el codificador para una unidad de tiempo adicional m después de ingresar. Típicamente, n y k son enteros pequeños, $k < n$, la secuencia de información es dividida en bloques de longitud k, y el codeword es dividido en bloques de longitud n. En el caso especial cuando $k = 1$, la secuencia de información no es dividida en bloques y se procesa continuamente. A diferencia de los códigos de bloque, se alcanzan grandes distancias mínimas y bajas probabilidades de error sin incrementar k y n, pero si se incrementa el orden m de la memoria.

3.2.1.1 Codificación de los códigos convolucionales

El diagrama de bloques de la Figura 3.1 Muestra un codificador de convolución lineal de tasa binaria de 1/2. La tasa del codificador es determinada por el hecho de que en la salida del decodificador se presentan dos bits por cada bit en la entrada. En general, un codificador con k bits de entrada y n bits de salida, tiene una tasa k/n. La tasa k/n se define como la tasa de código (Rc) del sistema.

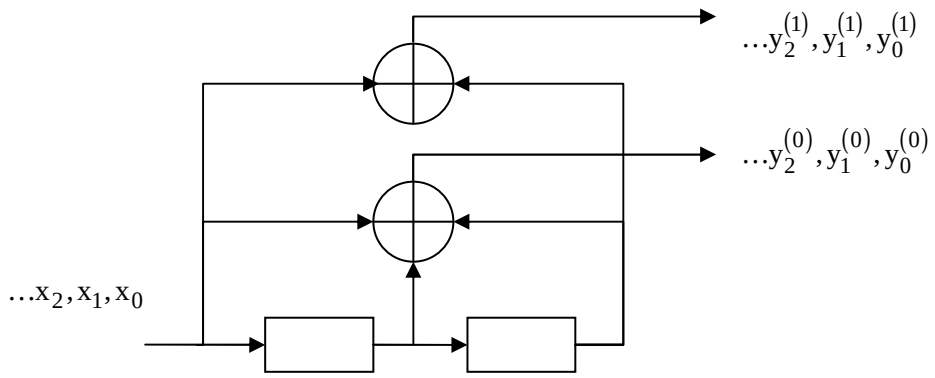


Figura 3.1 Codificador Convolucional.

Existe una forma para caracterizar la estructura de los códigos convolucionales, llamada secuencias del generador. Las secuencias del

generador se obtienen mediante una respuesta al impulso $g_j^{(i)}$ donde la salida $i-ésima$ del codificador es obtenida aplicando una función delta $d=(10000...)$ de secuencia de datos en la $j-ésima$ entrada.

Capítulo 4

CÓDIGOS LDPC (Low-Density Parity-Check Codes)

A través de este capítulo se presentarán las principales características de los códigos LDPC. Se mostrará los pasos necesarios para su codificación y se realizará un estudio de los principales algoritmos de decodificación y se explicará el funcionamiento del algoritmo de suma-producto mediante un ejemplo.

Los códigos LDPC son una clase de códigos de corrección de error propuestos por Gallager en su tesis de PhD en 1962 [10], 12 años después de que Hamming introdujera los primeros códigos de corrección de error (publicados en 1950). Los códigos Hamming y LDPC son códigos de bloque, los mensajes están divididos en bloques para ser codificados en el transmisor y de la misma forma decodificados como bloques separados en el receptor. Por un lado los códigos Hamming son cortos, muy bien estructurados y presentan una capacidad de corrección fija y conocida, mientras que los códigos LDPC presentan características opuestas, son usualmente largos y frecuentemente construidos pseudo-aleatoriamente con solo una noción probabilística de su rendimiento de corrección de error esperado. Después del trabajo de Gallager tuvieron que pasar treinta años para que los códigos LDPC fueran motivo de investigación, esto fue debido a sus altos requerimientos computacionales y de almacenamiento, difíciles de alcanzar en esa época.

Para iniciar el estudio y entender los fundamentos de los códigos LDPC, se presentan los bits de paridad como un medio para detectar, seguido del caso cuando se emplea más de un bit, el cual sirve para corrección de errores en datos digitales. Los códigos de bloque de corrección de errores son descritos como una combinación lineal de las ecuaciones de

comprobación de paridad (*parity-check equations*) y por lo tanto están definidos por su matriz de representación de comprobación de paridad (*parity-check matrix*).

4.1 Corrección de error usando comprobaciones de paridad

La idea esencial de la codificación de corrección de errores es aumentar los bits del mensaje (*message bits*) que van a ser comunicados con una deliberada incorporación de redundancia en forma de bits 'extra' con el fin de producir un codeword para el mensaje. Estos bits extra son agregados de tal forma que los codewords son lo suficientemente distintos uno del otro para permitir que el mensaje transmitido sea deducido correctamente en el receptor incluso cuando algunos bits en el codeword son afectados durante la transmisión sobre el canal [3].

El esquema más sencillo de codificación es un único código de comprobación de paridad (*single parity-check code*). Este código envuelve la suma de un único bit extra, llamado bit de comprobación de paridad (*parity-check bit*), para el mensaje binario; el valor de este bit depende de los bits en el mensaje. En un código de paridad par (*even-parity code*) el bit adicional agregado a cada mensaje asegura un número par de 1s en cada codeword.

Ejemplo 4.1

El vector de 7-bit ASCII para la letra A es $\left[1\ 0\ 0\ 0\ 0\ 0\ 1\right]$. Un bit de paridad es agregado como el octavo bit. El vector para A ya tiene un número par de 1s (dos) y así el valor del bit de paridad debe ser 0. Por consiguiente, el codeword para A es $\left[1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\right]$.

Explícitamente, para el código 7-bit ASCII más el bit de paridad se define un codeword c con la estructura: $\mathbf{c}=\left[c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8\right]$.

donde cada c_i es 1 o 0 y cada codeword satisface la siguiente restricción

$$c_1 \oplus c_2 \oplus c_3 \oplus c_4 \oplus c_5 \oplus c_6 \oplus c_7 \oplus c_8 = 0. \qquad (4.1)$$

La ecuación (4.1) es llamada una ecuación de comprobación de paridad. El símbolo $\oplus$ representa la suma o adición en modulo-2.

Ejemplo 4.2

Una letra del código ASCII de 7-bit es codificada con el código de comprobación de paridad único del Ejemplo 4.1. El codeword resultante es enviado a través de un canal ruidoso y se recibe el vector $y=\left[1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\right]$. Para comprobar si y es un codeword valido se prueba con (4.1):

$$\begin{aligned} & y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_8 \\ & = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1. \end{aligned} \qquad (E.4.2.1)$$

Ya que la suma es 1, la ecuación de comprobación de paridad no se satisface y y no es un codeword valido. Así que se ha detectado la presencia de al menos un error durante la transmisión.

Mientras la inversión de un único bit debido al ruido de canal se puede detectar fácilmente con un único código de comprobación de paridad, este código no es suficientemente potente para indicar cual bit, o tal vez cuales bits fueron invertidos. Por otra parte, ya que cualquier número par de inversiones de bits produce un vector que satisface la restricción (4.1), patrones de números pares de errores no se detectan por este código. La detección de más que un solo bit de error requiere o exige una mayor

redundancia en forma de bits de paridad adicionales. Estos códigos más sofisticados contienen múltiples ecuaciones de comprobación de paridad, cada una de las cuales debe satisfacerse por cada codeword en el código.

Ejemplo 4.3

Un código C consiste de todos los vectores de longitud 6 $\mathbf{c}=\left[c_1\ c_2\ c_3\ c_4\ c_5\ c_6\right]$, los cuales satisfacen las tres ecuaciones de comprobación de paridad

$$\begin{aligned} &c_1 \oplus c_2 \oplus c_4 = 0, \\ &c_1 \oplus c_2 \oplus c_3 \oplus c_6 = 0, \\ &c_2 \oplus c_3 \oplus c_5 = 0. \end{aligned} \tag{E.4.3.1}$$

Revisando el vector $\tilde{\mathbf{c}}=\left[1\ 0\ 0\ 0\ 0\ 1\right]$ vemos que

$$\begin{aligned} &c_1 \oplus c_2 \oplus c_4 = 1, \\ &c_1 \oplus c_2 \oplus c_3 \oplus c_6 = 0, \\ &c_2 \oplus c_3 \oplus c_5 = 0. \end{aligned} \tag{E.4.3.2}$$

Así que $\tilde{\mathbf{c}}$ no es un codeword valido para este código.

Las restricciones de codewords son frecuentemente escritas en forma de matriz, de esta forma las restricciones en (E.4.3.1) se convierten en:

$$\underbrace{\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}}_{H} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.2}$$

La matriz H es llamada la matriz de comprobación de paridad. Cada fila de H corresponde a una ecuación de comprobación de paridad y cada columna corresponde a un bit en el codeword. La entrada $(j,\ i)-ésima$ de H es 1 si el $i-ésimo$ bit del codeword está incluido en la $j-ésima$ ecuación de comprobación de paridad. De modo que, para un código binario con m restricciones de comprobación de paridad y una longitud de codeword N, la matriz de comprobación de paridad es una matriz binaria $m \times N$. En forma matricial un vector $\tilde{\mathbf{c}} = \left[c_1\ c_2\ c_3\ c_4\ c_5\ c_6\right]$ es un codeword valido para el código con matriz de comprobación de paridad H si y solo si este satisface la ecuación matricial

$$H\tilde{\mathbf{c}}^T = \mathbf{0}\ (\mathrm{mod}\ 2), \tag{4.3}$$

donde $\mathbf{0}$ es el vector de todo ceros de longitud-m y $\tilde{\mathbf{c}}^T$ denota la transpuesta de $\tilde{\mathbf{c}}$.

Existe más de una matriz de comprobación de paridad que puede describir un código particular; dos matrices de comprobación de paridad para el mismo código no tienen el mismo número de filas, y deben satisfacer (4.3) para cada codeword en el código.

Ejemplo 4.4

El código C en el Ejemplo 4.3 también puede ser descrito por cuatro ecuaciones de paridad:

$$\begin{aligned} &c_1 \oplus c_2 \oplus c_4 = 0, \\ &c_1 \oplus c_2 \oplus c_3 \oplus c_6 = 0, \\ &c_2 \oplus c_3 \oplus c_5 = 0, \\ &c_3 \oplus c_4 \oplus c_6 = 0. \end{aligned} \tag{E.4.4.1}$$

La ecuación extra es una combinación lineal de la primera con la segunda ecuación de comprobación de paridad y así es linealmente dependiente sobre las ecuaciones de comprobación de paridad existentes. (En la aritmética de módulo 2 una combinación lineal de las ecuaciones de comprobación de paridad es simplemente la suma módulo 2 de dos o más ecuaciones.) Es fácil notar que cualquier codeword que satisface $c_1 \oplus c_2 \oplus c_4 = 0$ y $c_1 \oplus c_2 \oplus c_3 \oplus c_6 = 0,$ también satisface $c_2 \oplus c_4 \oplus c_6 = 0.$

En los Ejemplos 4.3 y 4.4 hay tres ecuaciones linealmente independientes en seis variables, así que hay tres variables dependientes (una por cada ecuación). Estas son las variables correspondientes a los bits de comprobación de paridad. Las variables independientes restantes corresponden a los bits del mensaje. Entonces, para el código del Ejemplo 1.3, el número de bits del codeword es $N = 6,$ el número de bits del mensaje es $k = 3$ de esta forma tenemos una tasa de código $R = 1/2.$

Parece razonable que entre más restricciones de comprobación de paridad existan, menor será el número de codewords que pueden satisfacerlas. En realidad, esto es cierto siempre que el número de ecuaciones de comprobación de paridad sean linealmente independientes. Una ecuación extra que es linealmente dependiente en las ecuaciones existentes no añade ninguna restricción adicional para el conjunto de codeword entonces no reduce el número de variables independientes.

En general, un código puede tener cualquier número de restricciones de comprobación de paridad pero solo $N - k$ serán linealmente independientes, donde k es el número de bits del mensaje. En notación matricial $N - k$ es el rango de H sobre $\mathrm{GF}(2)$:

$$N - k = rango_2 H, \tag{4.4}$$

donde $rango_2 H$ es el número de filas en H que son linealmente independientes sobre $\mathrm{GF}(2)$.

4.2 Códigos LDPC

Como su nombre lo indica, los códigos LDPC son códigos de bloque con matrices de comprobación de paridad que contienen solo un número muy pequeño de entradas diferentes de cero. Esta escasez en H es esencial para una decodificación iterativa compleja que aumenta solo linealmente con la longitud del código.

Aparte del requerimiento que H sea dispersa, un código LDPC en sí no es diferente de cualquier otro código de bloque. De hecho, códigos clásicos de bloque pueden trabajar bien con algoritmos de decodificación iterativos si ellos pueden ser representados por una matriz de comprobación de paridad dispersa. Sin embargo, generalmente no es práctico la búsqueda de dicha matriz para un código existente. En cambio los códigos LDPC inicialmente están diseñados mediante la construcción de una matriz de comprobación de paridad dispersa y después se determina un codificador para el código.

La diferencia más grande entre los códigos LDPC y los clásicos códigos de bloque es la forma en cómo se decodifican. Los códigos clásicos de bloque son generalmente decodificados con ML como algoritmos de decodificación y así son usualmente cortos y diseñados algebraicamente para hacer esta tarea menos compleja. Sin embargo, los códigos LDPC son decodificados iterativamente usando una representación gráfica de su matriz de comprobación de paridad y por tal motivo son mucho más largos, menos estructurados, y diseñados con las propiedades de H como enfoque [3].

Una matriz de comprobación de paridad de un código LDPC es llamada *regular-*$(\omega_c,\ \omega_r)$ si cada bit del código es contenido en un número fijo ω_c

de comprobaciones de paridad y cada ecuación de comprobación de paridad contiene un número fijo ω_r de bits de código.

Ejemplo 4.5

Una matriz regular de comprobación de paridad, con $\omega_c = 2$ y $\omega_r = 3$, para el código del Ejemplo 4.3 es:

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \qquad \text{(E.4.5.1)}$$

Para una matriz irregular de comprobación de paridad se designa la fracción de columnas con peso i por v_i y la fracción de filas con peso i por h_i. Colectivamente el par $\mathbf{v}$ y $\mathbf{h}$ son llamados el grado de distribución (*degree distribution*) del código. En donde $\mathbf{v} = \left[v_1 \; v_2 \; v_3 \cdots \right]$ y $\mathbf{h} = \left[h_1 \; h_2 \; h_3 \cdots \right]$.

Ejemplo 4.6

La matriz de comprobación de paridad en (4.2) es irregular con grado de distribución $v_1 = 1/2,\ v_2 = 1/3,\ v_3 = 1/6,\ h_3 = 2/3\ y\ h_4 = 1/3.$

El número de 1s que tendrá la matriz de comprobación de paridad de un código LDPC regular es $m\omega_r = N\omega_c$. Mientras que para la matriz de un código irregular $m\left(\sum_i h_i i\right) = N\left(\sum_i v_i i\right)$.

Por lo tanto la tasa de un código de bloque está dada por:

$$R = \frac{k}{N} = \frac{N - rango_2 H}{N}. \tag{4.5}$$

Cuando la matriz de comprobación de paridad es de rango completo (*full rank*), es decir.

$$rango_2 H = m, \tag{4.6}$$

la tasa para un código regular puede escribirse como

$$r = \frac{k}{N} = \frac{N-m}{N} = \frac{N - N\omega_c / \omega_r}{N} = 1 - \frac{\omega_c}{\omega_r}, \tag{4.7}$$

y para un código irregular como

$$r = \frac{k}{N} = 1 - \frac{\sum_i v_i i}{\sum_i h_i i}. \tag{4.8}$$

En todos los casos, salvo algunos especialmente diseñados, las matrices de comprobación de paridad son de rango completo, o muy cerca a esto, y así la tasa de un código LDPC esta frecuentemente dada como

$$r \approx 1 - \frac{\omega_c}{\omega_r} \tag{4.9}$$

o

$$r \approx 1 - \frac{\sum_i v_i i}{\sum_i h_i i}, \tag{4.10}$$

incluso cuando el rango de H no ha sido explícitamente medido. En este caso r es llamada la *tasa diseñada* del código.

Note que las matrices de comprobación de paridad para los códigos LDPC son generalmente no sistemáticas. De modo que no es generalmente posible distinguir entre los bits del mensaje y los bits de paridad en los codewords LDPC.

Frecuentemente las matrices de comprobación de paridad de los códigos LDPC son representadas en forma gráfica por un gráfico de Tanner. El gráfico de Tanner consiste de dos conjuntos de nodos: N nodos para los bits de los codewords (llamados nodos de bit '*bit nodes*'), y m nodos para las ecuaciones de comprobación de paridad (llamados nodos de comprobación '*check nodes*'). Un borde (*edge*) une un nodo de bit a un nodo de comprobación si este bit está incluido en la correspondiente ecuación de comprobación de paridad, de tal manera que el número de bordes en el gráfico de Tanner es igual al número de 1s en la matriz de comprobación de paridad.

Ejemplo 4.7

El gráfico de Tanner de la matriz de comprobación de paridad en el Ejemplo 4.5 es mostrado en la Figura 4.1. Los nodos de bit son representados por los vértices circulares y los nodos de comprobación por los vértices cuadrados.

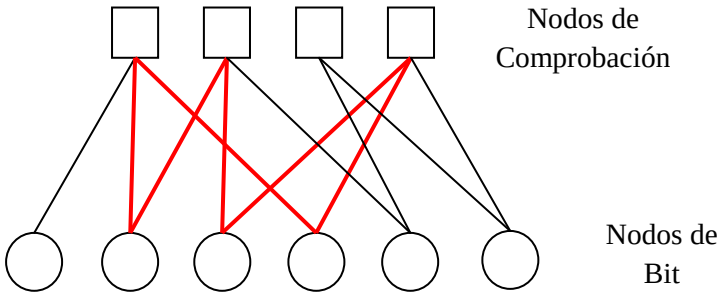


Figura 4.1 Representación en el gráfico de Tanner de la matriz de comprobación de paridad de (E.4.5.1). Un ciclo de longitud 6 (*6-cycle*) se muestra en color rojo.

Un ciclo en el gráfico Tanner es una secuencia de nodos las cual empieza y termina en el mismo nodo en el gráfico y no contiene otros nodos más de una vez. La longitud de un ciclo es el número de bordes que este contiene, y el *girth* de un gráfico es el tamaño de su ciclo más pequeño.

Ejemplo 4.8

Un ciclo de tamaño 6 es mostrado en color rojo en la Figura 4.1. No hay ciclos más pequeños en el gráfico Tanner, así que su *girth* es también 6.

4.2.1 Codificación de los códigos LDPC

La matriz de comprobación de paridad define el conjunto de codewords válidos (aquellos que satisfacen $\mathbf{c}H^T = \mathbf{0}$), pero no se ha visto aún como mapear los mensajes a estos codewords.

Ejemplo 4.9

Para distinguir entre los bits del mensaje y los bits de paridad en el codeword del Ejemplo 4.3, podemos reescribir las restricciones de comprobación de paridad de manera que cada una se resuelve para cada uno de los diferentes bits del codeword. Las restricciones del codeword del Ejemplo 4.3 se reescriben como

$$\begin{aligned} c_1 \oplus c_2 &= c_4, \\ c_2 \oplus c_3 &= c_5, \\ c_1 \oplus c_2 \oplus c_3 &= c_6. \end{aligned} \qquad \text{(E.4.9.1)}$$

Los bits del codeword c_4, c_5 y c_6 contienen los tres bits de comprobación de paridad mientras los bits del codeword c_1, c_2 y c_3 contienen los tres bits del mensaje. Los bits del mensaje son convencionalmente etiquetados $u_1, u_2, \ldots, u_k$, donde el vector $\mathbf{u} = \left[u_1 \; u_2 \cdots u_k\right]$ es el mensaje que va a ser comunicado. Este código es sistemático debido a que los primeros k bits del codeword contienen los bits del mensaje.

Escrito como en (E.4.9.1), las restricciones del codeword muestran como codificar el mensaje.

Ejemplo 4.10

Usando las restricciones en (E.4.9.1) el mensaje $\mathbf{u} = \left[1 \; 0 \; 1\right]$ produce los bits de paridad

$$
\begin{aligned}
c_4 &= 1 \oplus 0 = 1, \\
c_5 &= 0 \oplus 1 = 1, \\
c_6 &= 1 \oplus 0 \oplus 1 = 0,
\end{aligned}
\tag{E.4.10.1}
$$

y entonces el codeword para este mensaje es $\mathbf{c} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$.

De nuevo estas limitaciones pueden escribirse en forma matricial:

$$
\begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}}_{G}. \tag{4.11}
$$

o

$$
\mathbf{c} = \mathbf{u}G, \tag{4.12}
$$

donde la matriz G es llamada la matriz generador del código. La entrada $(i, j)-ésima$ de G es 1 si el *i-ésimo* bit del mensaje juega un papel en la determinación del *j-ésimo* bit del codeword. Por ejemplo, la columna cuatro de G es 1 en la primera y segunda fila entonces tenemos:

$$
c_4 = u_1 \oplus u_2. \tag{4.13}
$$

Para códigos sistemáticos la matriz generador contiene la matriz identidad I_k de $k \times k$ como sus primeras k columnas, ya que el *i-ésimo* bit del mensaje es directamente mapeado al *i-ésimo* bit del codeword.

De (4.12) el conjunto de codewords para el código con generador G es el conjunto de todas las posibles combinaciones lineales de las filas de G. Así que, si G es la matriz generador para un código con matriz de comprobación de paridad H, entonces

$$GH^T = \mathbf{0} \pmod 2, \tag{4.14}$$

donde $\mathbf{0}$ es una matriz $k \times m$ de únicamente ceros.

Para un código binario con mensajes de k bits y codewords de longitud N, la matriz generador G es una matriz binaria $k \times N$. Un código con k bits de mensaje contiene 2^k codewords, los cuales son un subconjunto del total de las posibles combinaciones de los vectores binarios 2^N de longitud N.

Ejemplo 4.11

Sustituyendo cada uno de los distintos mensajes $u_1u_2u_3 = 000,\ 001, \ldots,\ 111$ en (4.12) se obtiene el siguiente conjunto de vectores codewords para el código del Ejemplo 4.3:

$$\begin{aligned}&[0\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 1\ 0\ 1\ 1], [0\ 1\ 0\ 1\ 1\ 1], [0\ 1\ 1\ 1\ 0\ 0],\\ &[1\ 0\ 0\ 1\ 0\ 1], [1\ 0\ 1\ 1\ 1\ 0], [1\ 1\ 0\ 0\ 1\ 0], [1\ 1\ 1\ 0\ 0\ 1].\end{aligned} \tag{E.4.11}$$

En general, una matriz generador para un código con matriz de comprobación de paridad H puede encontrarse realizando eliminación de Gauss-Jordan sobre H para obtenerla en la forma

$$H = \begin{bmatrix} A & I_{N-k} \end{bmatrix}, \tag{4.15}$$

donde A es una matriz binaria $(N-k) \times k$ y I_{N-k} es la matriz identidad de orden $(N-k)$. La matriz generador es entonces

$$G = \begin{bmatrix} I_k & A^T \end{bmatrix}. \tag{4.16}$$

4.2.1.1 (Casi) Codificación lineal en tiempo para códigos LDPC

En lugar de encontrar la matriz generador para H, un código LDPC puede codificarse usando la matriz de comprobación de paridad directamente mediante su transformación en forma triangular inferior y usando sustitución hacia atrás. La idea es hacer la mayor cantidad de transformaciones como sea posible utilizando únicamente permutaciones de filas y columnas para mantener H dispersa.

En primer lugar, usando solo permutaciones de fila y columna la matriz de comprobación de paridad se coloca en la forma triangular inferior aproximada (*approximate lower triangular form*)

$$H_t = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}, \tag{4.17}$$

donde la matriz T es una matriz triangular inferior de tamaño $(m-g)\times(m-g)$ (esto significa que T tiene 1s en la diagonal desde la esquina superior izquierda a la esquina inferior derecha, y todas las entradas por encima de la diagonal son 0). Si H_t es de rango completo entonces la matriz B es de tamaño $(m-g)\times g$ y A es de tamaño $(m-g)\times k$. Las g filas de H en C, D y E son llamadas el "*gap*" de la representación aproximada y entre más pequeña g, menor es la complejidad de codificación para el código.

4.2.2 Decodificación de los códigos LDPC

Se supone que un código ha sido enviado bajo un canal binario simétrico y uno o más de los bits del codeword pueden haber sido invertidos. La tarea del decodificador es detectar cualquier bit invertido y, si es posible, corregirlo.

Cada codeword en el código debe satisfacer (4.3), y así cualquier palabra recibida que no satisface esta ecuación debe tener errores. Por lo tanto, la

matriz de comprobación de paridad H reduce considerablemente la complejidad de la detección de errores.

Ejemplo 4.12

El codeword $\mathbf{c}=\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$ del código en el Ejemplo 4.3 fue enviado a través de un canal binario simétrico y se recibe el vector $\mathbf{y}=\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$. Sustituyendo en el lado izquierdo de la ecuación (4.3) se obtiene:

$$H\mathbf{y}^T=\begin{bmatrix}1 & 1 & 0 & 1 & 0 & 0\\ 1 & 1 & 1 & 0 & 0 & 1\\ 0 & 1 & 1 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}0\\0\\0\\0\\1\\0\end{bmatrix}=\begin{bmatrix}0\\0\\1\end{bmatrix}. \qquad \text{(E.4.12.1)}$$

El resultado no es cero y de esta forma el vector y no es un codeword de este código. Por consiguiente se concluye que ocurrieron errores de inversión de bit (*bit-flipping errors*) durante la transmisión.

El vector

$$\mathbf{s}=H\mathbf{y}^T, \qquad (4.18)$$

es llamado el síndrome de $\mathbf{y}$. El síndrome indica cuales restricciones de comprobación de paridad no han sido cumplidas por $\mathbf{y}$.

Ejemplo 4.13

El resultado de (E.4.12.1), es decir, el síndrome $\mathbf{s}$, indica que la tercera ecuación de comprobación de paridad representada en H no es cumplida por $\mathbf{y}$. Ya que esta ecuación de comprobación de paridad envuelve el segundo, tercero, y quinto bit del codeword se puede concluir que al menos uno de estos tres bits ha sido invertido por el canal.

Al utilizar una corrección de errores mediante comparación directa del vector recibido con cada uno de los codewords en el código garantiza la devolución del codeword más probable. Sin embargo, tal búsqueda exhaustiva sólo es factible cuando el número de bits del mensaje k es pequeño. Para códigos con k en el orden de miles se convierte computacionalmente demasiado costoso comparar el vector recibido directamente con cada uno de los 2^k codewords del código. Numerosas e ingeniosas soluciones han sido propuestas para hacer esta tarea menos compleja; que incluyen la elección códigos algebraicos, y la explotación de su estructura, así como la aceleración de la decodificación o, como para los códigos LDPC, la formulación de métodos de decodificación que no son de máxima verosimilitud (*maximum likelihood*), pero la pueden realizar muy bien con una complejidad reducida.

La clase de algoritmos utilizados para decodificar los códigos LDPC son colectivamente denominados algoritmos de paso-de-mensajes (*message-passing algorithms*), ya que su operación puede explicarse por el paso de mensajes a lo largo de los bordes del gráfico de Tanner. Cada nodo en el gráfico de Tanner funciona de forma aislada, teniendo acceso solo a los mensajes en los bordes conectados a este. Los algoritmos de paso de mensajes son un tipo de algoritmo de decodificación iterativa donde los mensajes pasan hacia atrás y hacia adelante entre los nodos de bit y los nodos de comprobación iterativamente hasta que se logre algún resultado (o el proceso sea detenido); ellos son nombrados o designados según el

tipo de mensaje que se pasa o por el tipo de operación realizada en los nodos.

En algunos algoritmos, tales como la decodificación de inversión-de-bit (*bit-flipping decoding*), los mensajes son binarios y en otros, tales como la decodificación de creencia de propagación (*belief-propagation decoding*) o suma-producto (*sum-product decoding*), los mensajes son probabilidades (o el logaritmo de la razón de sus probabilidades '*log likelihood ratios*') que representan un nivel de creencia sobre el valor de los bits del codeword.

4.2.2.1 Decodificación de paso de mensaje sobre un canal binario de borrado

Sobre un canal binario de borrado (BEC) un bit transmitido es recibido correctamente o completamente borrado con alguna probabilidad ϵ. Ya que los bits que son recibidos son siempre completamente correctos la tarea del decodificador es determinar el valor de los bits no conocidos.

Si existe una ecuación de comprobación de paridad que incluye un único bit borrado el valor correcto para el bit borrado puede determinarse por la elección del valor que satisface el requisito de paridad par.

Ejemplo 4.14

El código en el Ejemplo 4.3 incluye la ecuación de comprobación de paridad

$$c_1 \oplus c_2 \oplus c_4 = 0. \qquad \text{(E.1.14.1)}$$

Si el valor del bit c_1 se conoce que es 0 y el valor del bit c_2 es 1, entonces el valor del bit c_4 debe ser 1, para que c_1, c_2 y c_4 sean parte de un codeword válido para este código.

En el decodificador de mensaje de paso, cada nodo de comprobación puede determinar el valor de un bit borrado si este es el único bit borrado en su ecuación de comprobación de paridad. Los mensajes pasados a lo largo de los bordes del gráfico de Tanner son directos: un nodo de bit envía el mismo mensaje saliente a cada uno de los nodos de comprobación a los cuales está conectado. Este mensaje, etiquetado M_i para el $i-ésimo$ nodo de bit, declara el valor del bit como 1 o 0 si el valor es conocido o e si este es borrado. Si un nodo de comprobación recibe solo un mensaje e, este nodo puede calcular el valor del bit desconocido mediante la elección del valor que satisface la paridad.

Los nodos de comprobación envían de regreso diferentes mensajes a cada uno de los nodos de bit a los cuales están conectados. El mensaje $E_{j,i}$ del $j-ésimo$ nodo de comprobación al $i-ésimo$ nodo de bit declara que el valor del $i-ésimo$ bit es 1, 0 o e como se determina por el $j-ésimo$ nodo de comprobación. Si el nodo de bit de un bit borrado recibe un mensaje entrante 1 o 0, el nodo de bit cambia su valor al valor de mensaje entrante. En una iteración del decodificador de paso de mensaje (*message-passing decoding*) se envían los mensajes del nodo de bit al nodo de comprobación una vez y de vuelta se envían los mensajes del nodo de comprobación al nodo de bit una vez. Este proceso es repetido hasta que todos los valores de bit son conocidos, o hasta que algún número máximo de iteraciones del decodificador ha sido alcanzado y el decodificador se detiene (en la practica el decodificador también puede detenerse una vez que en una sola iteración no se encuentren bits borrados que se necesiten corregir, ya que no hay nuevas ecuaciones de comprobación de paridad de único borrado para las siguientes iteraciones).

Se usa la notación B_j para representar el conjunto de bits en la $j-ésima$ ecuación de comprobación de paridad de H. Entonces para la matriz de comprobación de paridad en el Ejemplo 4.5 tenemos

$$B_1 = \{1,2,4\},\ B_2 = \{2,3,5\},\ B_3 = \{1,5,6\},\ B_4 = \{3,4,6\}. \quad (4.19)$$

De la misma forma se usa la notación de A_i para representar las ecuaciones de comprobación de paridad para el $i-ésimo$ bit del código. Para el código en el Ejemplo 4.5 tenemos

$$\begin{aligned} A_1 &= \{1,3\},\ A_2 = \{1,2\},\ A_3 = \{2,4\}, \\ A_4 &= \{1,4\},\ A_5 = \{2,3\},\ A_6 = \{3,4\}. \end{aligned} \quad (4.20)$$

4.2.2.2 Decodificación de inversión-de-bit

Un algoritmo de decodificación de inversión de bit (*Bit-flipping decoding*) es el nombre dado a los algoritmos de paso de mensaje de decisión fuerte para códigos LDPC. El detector realiza una decisión binaria (fuerte) entre cada bit recibido y esta es pasada al decodificador. Para los algoritmos de inversión de bit los mensajes pasados a través de los bordes del gráfico Tanner son también binarios: un nodo de bit envía un mensaje declarando si este es 1 o 0, y cada nodo de comprobación envía un mensaje a cada nodo de bit a los cuales está conectado declarando su decisión sobre el valor de este bit (basados en la información disponible para el nodo de comprobación) o un mensaje indicando si su correspondiente ecuación de comprobación de paridad se satisface.

Para el algoritmo de inversión de bit que se describe aquí, el $j-ésimo$ nodo de comprobación determina su decisión sobre el $i-ésimo$ nodo de bit asumiendo que el $i-ésimo$ bit ha sido borrado y entonces elige el valor de 1 o 0 que cumple la $j-ésima$ ecuación de comprobación de paridad. (Esto significa que la decisión que realiza el nodo de

comprobación j acerca del nodo de bit i es basada solo en los valores de los otros bits incluidos en la $j-ésima$ ecuación de comprobación de paridad.) De esta forma el $j-ésimo$ nodo de comprobación determina un valor para el $i-ésimo$ bit que está completamente independiente del valor del $i-ésimo$ bit que se acaba de recibir (sin embargo, utiliza este valor en el cálculo de los valores para los otros bits en la ecuación de comprobación de paridad). El nodo de comprobación se dice que es la creación adicional, extrínseca, de la información sobre el $i-ésimo$ bit. Sin embargo, note que al contrario del algoritmo de decodificación de paso de mensaje para la corrección de errores, los valores de los bits son proporcionados al nodo de comprobación y de esta manera los valores de los bits calculados por el nodo de comprobación no son necesariamente correctos.

En el nodo de bit, toda la información extrínseca acerca de un bit es comparada con la información recibida del canal para determinar el valor del bit más probable. Si la mayoría de mensajes recibidos por un nodo de bit son diferentes de su valor recibido entonces el nodo de bit invierte ('*flips*') su valor actual. Este proceso es repetido hasta que todas las ecuaciones de comprobación de paridad se cumplidas o hasta que algún número máximo de iteraciones del decodificador sea alcanzado y el decodificador se detiene.

El decodificador de bit invertido se detiene inmediatamente siempre que un codeword válido ha sido encontrado mediante la comprobación de si la ecuación de comprobación de paridad es cumplida (es decir, $\hat{\mathbf{c}}H^T = \mathbf{s} = \mathbf{0}$ es un criterio de parada para la decodificación de inversión de bit). Esto es cierto para todos los decodificadores de paso de mensajes de los códigos LDPC y tiene dos importantes beneficios: en primer lugar, iteraciones adicionales se evitan una vez que una solución ha sido encontrada y, en segundo lugar, la falta de convergencia a un codeword siempre es detectada.

El algoritmo de inversión de bit se basa en el principio que un bit de codeword incorrecto estará contenido en un número grande de ecuaciones de comprobación de paridad sin otros bits incorrectos, cada uno que será capaz de calcular el valor correcto para este bit.

La matriz de baja densidad H ayuda a distribuir los bits en las comprobaciones de modo que es poco probable que dos ecuaciones contengan el mismo conjunto bits de codeword.

4.2.2.3 Decodificación Suma-Producto

El algoritmo de suma-producto es un algoritmo de paso de mensaje de decisión suave. Es similar al algoritmo de inversión de bit pero los mensajes que representan cada decisión (si el valor del bit es 1 o 0) son ahora probabilidades. Mientras la decodificación de inversión de bit acepta una decisión fuerte inicial sobre los bits recibidos como entrada, el algoritmo de suma-producto es un algoritmo de decisión suave que acepta como entrada la probabilidad para cada bit recibido. Las probabilidades de bit recibidas o de entrada al canal también son llamadas las probabilidades a priori para los bits recibidos, porque se conocen antes de que el decodificador LDPC sea operado.

Para el decodificador de suma-producto, la información extrínseca que pasa entre los nodos es también dada como probabilidades en lugar de decisiones fuertes. El mensaje extrínseco $E_{j,i}$ del nodo de comprobación j al nodo de bit i al cual está conectado es la opinión del nodo de comprobación j sobre la probabilidad que $c_i = 1$, basados en la información disponible para el nodo de comprobación j. Esto es, $E_{j,i}$ da la probabilidad de que $c_i = 1$, lo que causará que la ecuación de comprobación de paridad j se cumpla. (Note que $E_{j,i}$, no está definido si el bit i no está incluido en la comprobación j ya que la información extrínseca no es pasada entre los nodos i y j en este caso.) La

probabilidad de que una ecuación de comprobación de paridad se cumpla si $c_i = 1$, es la probabilidad de que un número impar de bits en la ecuación de comprobación de paridad sean 1s:

$$p_{j,i}^{ext} = \frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} \left(1 - 2P_{j,i'}\right), \tag{4.21}$$

donde $P_{j,i'}$, es el estimado actual disponible para el nodo de comprobación j de la probabilidad de que $c_{i'} = 1$. La probabilidad de que la ecuación de comprobación de paridad se cumpla si $c_i = 0$ es de esta manera $1 - p_{j,i}^{ext}$. Para una variable binaria x es fácil encontrar $p(x=1)$ dado $p(x=0)$, ya que $p(x=1) = 1 - p(x=0)$, y así necesitamos únicamente almacenar un valor de probabilidad para x. El logaritmo de las relaciones de verosimilitud (*Log likelihood ratios*) son usadas para representar las medidas para una variable binaria por medio de un único valor:

$$L(x) = \log \frac{p(x=0)}{p(x=1)}, \tag{4.22}$$

donde por $\log$ nos referimos a $\log_e$. El signo de $L(x)$ proporciona una decisión fuerte sobre x y la magnitud $|L(x)|$ es la confiabilidad de esta decisión. Convirtiendo de *log likelihood ratios* nuevamente a probabilidades,

$$\begin{aligned} p(x=1) &= \frac{e^{-L(x)}}{1 + e^{-L(x)}}, \\ p(x=0) &= \frac{e^{L(x)}}{1 + e^{L(x)}}. \end{aligned} \tag{4.23}$$

El beneficio de la representación logarítmica de las probabilidades, es que, cuando es necesario multiplicar las probabilidades, con el logaritmo de las relaciones de verosimilitud solo se necesita sumar; esto puede reducir la complejidad del decodificador de suma-producto.

Expresado como logaritmo de las relaciones de verosimilitud (*log likelihood ratio*), la información extrínseca del nodo de comprobación *j* al nodo de bit *i* es:

$$E_{j,i} = L\left(p_{j,i}^{ext}\right) = \log \frac{1 - p_{j,i}^{ext}}{p_{j,i}^{ext}}. \tag{4.24}$$

Ahora sustituyendo (4.23) en (4.24)

$$\begin{aligned} E_{j,i} &= \log \frac{\frac{1}{2} + \frac{1}{2}\prod_{i' \in B_j, i' \neq i}\left(1 - 2P_{j,i'}\right)}{\frac{1}{2} - \frac{1}{2}\prod_{i' \in B_j, i' \neq i}\left(1 - 2P_{j,i'}\right)} \\ &= \log \frac{1 + \prod_{i' \in B_j, i' \neq i}\left(1 - 2\frac{e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}\right)}{1 - \prod_{i' \in B_j, i' \neq i}\left(1 - 2\frac{e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}\right)} \\ &= \log \frac{1 + \prod_{i' \in B_j, i' \neq i}\frac{1 - e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}}{1 - \prod_{i' \in B_j, i' \neq i}\frac{1 - e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}}, \end{aligned} \tag{4.25}$$

donde

$$M_{j,i'} \triangleq L\left(P_{j,i'}\right) = \log \frac{1 - P_{j,i'}}{P_{j,i'}}. \tag{4.26}$$

Usando la relación

$$\tanh\frac{1}{2}\log\left(\frac{1-p}{p}\right)=1-2p, \tag{4.27}$$

se tiene

$$E_{j,i}=\log\frac{1+\prod_{i'\in B_j,i'\neq i}\tanh\left(M_{j,i'}/2\right)}{1-\prod_{i'\in B_j,i'\neq i}\tanh\left(M_{j,i'}/2\right)}. \tag{4.28}$$

Alternativamente, usando la relación

$$2\tanh^{-1}p=\log\frac{1+p}{1-p}, \tag{4.29}$$

(4.28) puede escribirse equivalentemente como

$$E_{j,i}=2\tanh^{-1}\prod_{i'\in B_j,i'\neq i}\tanh\left(M_{j,i'}/2\right). \tag{4.30}$$

Cada nodo de bit tiene acceso a las entradas LLR, R_i, y a las LLR de cada nodo de comprobación al cual esté conectado. El total de LLR $i-ésimo$ bit es la suma de estos LLRs:

$$L_i=L\left(P_i\right)=R_i+\sum_{j\in A_i}E_{j,i}. \tag{4.31}$$

Sin embargo, los mensajes enviados desde los nodos de bit a los nodos de comprobación, $M_{j,i}$, no son los valores completos de LLR para cada bit. Para evitar enviar de regreso a cada nodo de comprobación la información que ya tiene, el mensaje enviado desde el $i-ésimo$ nodo de bit al $j-ésimo$ nodo de comprobación es la suma en (4.31) sin la componente $E_{j,i}$ acabada de recibir del $j-ésimo$ nodo de comprobación:

$$M_{j,i} = \sum_{j' \in A_i, j' \neq j} E_{j',i} + R_i. \tag{4.32}$$

El objetivo de la decodificación de suma-producto es (a) calcular la probabilidad a posteriori (APP) para cada bit del codeword, $p_i = p\{c_i = 1 | \mathbf{s} = \mathbf{0}\}$, la cual es la probabilidad de que el $i-ésimo$ bit del codeword sea 1 condicional al evento $\mathbf{s}=\mathbf{0}$ (es decir, que todas las restricciones de comprobación de paridad se cumplan), y (b) seleccionar el valor decodificado para cada bit como el valor con el máximo de probabilidad APP. El algoritmo de suma-producto iterativamente calcula una aproximación del valor MAP (un decodificador que selecciona el codeword con máxima probabilidad a posteriori 'maximum APP probability' es un máximum a posteriori decoder 'MAP') para cada bit del código. Sin embargo, las probabilidades a posteriori devueltas por el decodificador de suma-producto son solo probabilidades MAP exactas si el gráfico de Tanner es libre de ciclos. En resumen, la información extrínseca obtenida de una restricción de comprobación de paridad en la primera iteración es independiente de la información de probabilidad a priori para ese bit (lo cual hace depender de las probabilidades a priori de los otros bits del codeword). La información extrínseca proporcionada al nodo de bit i en subsecuentes iteraciones permanece independiente de la probabilidad a priori original para c_i hasta que esta probabilidad se devuelve al nodo de bit i a través de un ciclo en el gráfico de Tanner. La correlación de la información extrínseca con la probabilidad original a priori es lo que impide que las probabilidades resultantes sean exactas.

Las entradas son las razones logarítmicas de verosimilitud (*log likelihood ratios*) para las probabilidades a priori del mensaje de cada canal,

$$R_i = \log \frac{p(c_i = 0 | y_i)}{p(c_i = 1 | y_i)}, \tag{4.33}$$

la matriz de comprobación de paridad H y el máximo número de iteraciones, $I_{\max}$. Las salidas del algoritmo son las probabilidades de bit a posteriori estimadas de los bits recibidos como el logaritmo de la relaciones de verosimilitud.

El decodificador de suma-producto inmediatamente se detiene siempre que un codeword válido ha sido encontrado mediante una revisión del cumplimiento de las ecuaciones de comprobación de paridad (por lo tanto $\hat{\mathbf{c}}H^T = \mathbf{0}$ es también un criterio de parada para la decodificación de suma-producto).

Ejemplo 4.15

El código LDPC del Ejemplo 4.5 es usado para codificar el codeword $\mathbf{c} = [1\ 0\ 0\ 1\ 0\ 1]$. El vector $\mathbf{c}$ es enviado a través de un BSC con probabilidad cruzada $\epsilon = 0.2$, y la señal recibida es $\mathbf{y} = [1\ 1\ 0\ 1\ 0\ 1]$.

Ya que el canal binario es simétrico la probabilidad de que al enviar un 0 se reciba un 1 es la probabilidad ϵ que se produzca un cruce, mientras la probabilidad de que al enviar un 1 se reciba un 1 es la probabilidad $1-\epsilon$ que no se produzca un cruce. De la misma forma, la probabilidad que se reciba un 0 cuando un 1 fue enviado es la probabilidad que ocurra un cruce, mientras la probabilidad que se reciba un 0 cuando un 0 fue enviado es la probabilidad que no ocurra un cruce. Por lo tanto, las LLR a priori para el BSC son:

$$R_i = \begin{cases} \log\dfrac{\epsilon}{1-\epsilon} & si\ y_i = 1, \\ \log\dfrac{\epsilon}{1-\epsilon} & si\ y_i = 1. \end{cases} \qquad \text{(E.4.15.1)}$$

Para este canal tenemos

$$\log\frac{\epsilon}{1-\epsilon}=\log\frac{0.2}{0.8}=-1.3863,$$
$$\log\frac{\epsilon}{1-\epsilon}=\log\frac{0.8}{0.2}=1.3863. \qquad \text{(E.4.15.2)}$$

y entonces

$$\mathbf{R}=\begin{bmatrix}-1.3863 & -1.3863 & 1.3863 & -1.3863 & 1.3863 & -1.3863\end{bmatrix}. \qquad \text{(E.4.15.3)}$$

Para comenzar la decodificación establecemos el número máximo de iteraciones a 3 y se alimenta con H y R. En la inicialización,

$$M_{j,i}=R_i. \qquad \text{(E.4.15.4)}$$

El primer bit es incluido en la primera y tercera comprobación y entonces $M_{1,1}$ y $M_{3,1}$ son inicializados a R_1:

$$M_{1,1}=R_1=-1.3863 \text{ y } M_{3,1}=R_1=-1.3863. \qquad \text{(E.4.15.5)}$$

Repitiendo esto para los bits restantes tenemos:

$$\begin{aligned}
&\text{para } i=2,\ M_{1,2}=M_{2,2}=R_2=-1.3863,\\
&\text{para } i=3,\ M_{2,3}=M_{4,3}=R_3=1.3863,\\
&\text{para } i=4,\ M_{1,4}=M_{4,4}=R_4=-1.3863,\\
&\text{para } i=5,\ M_{2,5}=M_{3,5}=R_5=1.3863,\\
&\text{para } i=6,\ M_{3,6}=M_{4,6}=R_6=-1.3863.
\end{aligned} \qquad \text{(E.4.15.6)}$$

Ahora se calculan las probabilidades extrínsecas para los mensajes comprobación a bit. La primera comprobación de paridad incluye el primer, segundo y cuarto bit y entonces la probabilidad extrínseca del primer nodo de comprobación al primer nodo de bit depende de las probabilidades del segundo y cuarto bit:

$$
\begin{aligned}
E_{1,1} &= \log \frac{1+\tanh\left(M_{1,2}/2\right)\tanh\left(M_{1,4}/2\right)}{1-\tanh\left(M_{1,2}/2\right)\tanh\left(M_{1,4}/2\right)} \\
&= \log \frac{1+\tanh\left(-1.3863/2\right)\tanh\left(-1.3863/2\right)}{1-\tanh\left(-1.3863/2\right)\tanh\left(-1.3863/2\right)} \\
&= 0.7538.
\end{aligned}
\qquad \text{(E.4.15.7)}
$$

Similarmente, la probabilidad extrínseca del primer nodo de comprobación al segundo nodo de bit depende de las probabilidades del primer y cuarto bit:

$$
\begin{aligned}
E_{1,1} &= \log \frac{1+\tanh\left(M_{1,1}/2\right)\tanh\left(M_{1,4}/2\right)}{1-\tanh\left(M_{1,1}/2\right)\tanh\left(M_{1,4}/2\right)} \\
&= 0.7538.
\end{aligned}
\qquad \text{(E.4.15.8)}
$$

y la probabilidad extrínseca del primer nodo de comprobación al cuarto nodo de bit depende de las LLR enviadas del primer y segundo nodo de bit al primer nodo de comprobación:

$$
\begin{aligned}
E_{1,1} &= \log \frac{1+\tanh\left(M_{1,1}/2\right)\tanh\left(M_{1,2}/2\right)}{1-\tanh\left(M_{1,1}/2\right)\tanh\left(M_{1,2}/2\right)} \\
&= 0.7538.
\end{aligned}
\qquad \text{(E.4.15.9)}
$$

Repitiendo para todas las comprobaciones dadas, obtenemos las LLR extrínsecas:

$$
E = \begin{bmatrix}
0.7538 & 0.7538 & \cdot & 0.7538 & \cdot & \cdot \\
\cdot & 0.7538 & -0.7538 & \cdot & -0.7538 & \cdot \\
-0.7538 & \cdot & \cdot & \cdot & 0.7538 & -0.7538 \\
\cdot & \cdot & 0.7538 & -0.7538 & \cdot & -0.7538
\end{bmatrix}. \qquad \text{(E.4.15.10)}
$$

Para ahorrar espacio las LLR extrínsecas están dadas en forma matricial, donde la $(j,\ i)-ésima$ entrada de E retiene el valor de $E_{j,i}$. Una entrada de punto indica que una LLR no existe para esa i y j.

Para verificar si hay un codeword válido, se calcula las probabilidades estimadas a posteriori para cada bit, mediante una decisión fuerte y se comprueba el síndrome s. El primer bit tiene LLR extrínsecas de la primera y tercera comprobación y una LLR intrínseca del canal. El total de LLR es su suma:

$$\begin{aligned} L1 &= R_1 + E_{1,1} + E_{3,1} \\ &= -1.3863 + 0.7538 - 0.7538 = -1.3863. \end{aligned} \tag{E.4.15.11}$$

La LLR del canal es negativa y las LLR extrínsecas se cancelan, lo cual indica que el primer bit es 1. De esta forma la decisión sobre el primer bit se mantiene. Se repite este procedimiento para los siguientes bits:

$$\begin{aligned} L_2 &= R_2 + E_{1,2} + E_{2,2} = 0.1213, \\ L_3 &= R_3 + E_{2,3} + E_{4,3} = 1.3863, \\ L_4 &= R_4 + E_{1,4} + E_{4,4} = -1.3863, \\ L_5 &= R_5 + E_{2,5} + E_{3,5} = 1.3863, \\ L_6 &= R_6 + E_{3,6} + E_{4,6} = -2.8939. \end{aligned} \tag{E.4.15.12}$$

Al realizar una decisión fuerte sobre los bits recibidos (la cual es simplemente dada por los signos de las LLR) se obtiene

$$\hat{\mathbf{c}} = [1\ 0\ 0\ 1\ 0\ 1]. \tag{E.4.15.13}$$

Para verificar si $\hat{\mathbf{c}}$ es un codeword válido, se tiene

$$s = \hat{\mathbf{c}}H' = [1\ 0\ 0\ 1\ 0\ 1]\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [0\ 0\ 0\ 0]. \tag{E.4.15.14}$$

Ya que $\boldsymbol{s}$ es cero, $\hat{\mathbf{c}}$ es un codeword válido y la decodificación se detiene, retornando $\hat{\mathbf{c}}$ como la palabra decodificada.

El algoritmo de suma-producto puede ser modificado para reducir la complejidad de implementación del decodificador. Esto puede hacerse alterando (4.30)

$$E_{j,i} = 2\tanh^{-1}\prod_{i'\in B_j, i'\neq i}\tanh\left(M_{j,i'}/2\right), \qquad (4.34)$$

de tal manera que se reemplaza el termino de producto por una suma. Por simplicidad escribiremos

$$\prod_{i'} \triangleq \prod_{i'\in B_j, i'\neq i}, \qquad (4.35)$$

de aquí en adelante.

En primer lugar, $M_{j,i'}$ puede factorizarse como:

$$M_{j,i'} = \alpha_{j,i'}\beta_{j,i'}, \qquad (4.36)$$

donde

$$\begin{aligned} \alpha_{j,i'} &= \operatorname{sign} M_{j,i'}, \\ \beta_{j,i'} &= \left|M_{j,i'}\right|. \end{aligned} \qquad (4.37)$$

Usando esta notación se tiene que

$$\prod_{i'}\tanh\left(M_{j,i'}/2\right) = \prod_{i'}\alpha j,i'\prod_{i'}\tanh\left(\beta_{j,i'}/2\right). \qquad (4.38)$$

Entonces (4.30) se convierte en

$$
\begin{aligned}
E_{j,i} &= 2\tanh^{-1}\left(\prod_{i'}\alpha_{j,i'}\prod_{i'}\tanh\left(\beta_{j,i'}/2\right)\right) \\
&= \left(\prod_{i'}\alpha_{j,i'}\right)2\tanh^{-1}\prod_{i'}\tanh\left(\beta_{j,i'}/2\right).
\end{aligned}
\tag{4.39}
$$

La ecuación (4.39) puede ahora ser reordenada para reemplazar el producto por una suma:

$$
\begin{aligned}
E_{j,i} &= \left(\prod_{i'}\alpha_{j,i'}\right)2\tanh^{-1}\log^{-1}\log\prod_{i'}\tanh\left(\beta_{j,i'}/2\right) \\
&= \left(\prod_{i'}\alpha_{j,i'}\right)2\tanh^{-1}\log^{-1}\sum_{i'}\log\tanh\left(\beta_{j,i'}/2\right).
\end{aligned}
\tag{4.40}
$$

Después, se define

$$
\phi(x) = -\log\tanh\frac{x}{2} = \log\frac{e^{x}+1}{e^{x}-1}, \tag{4.41}
$$

y note que, dado que

$$
\phi\left(\phi(x)\right) = \log\frac{e^{\phi(x)}+1}{e^{\phi(x)}-1} = x. \tag{4.42}
$$

Se tiene que $\phi^{-1} = \phi$. Finalmente, (4.40) se convierte en

$$
E_{j,i} = \left(\prod_{i'}\alpha_{j,i'}\right)\phi\left(\sum_{i'}\phi\left(\beta_{j,i'}\right)\right). \tag{4.43}
$$

El producto de los signos puede ser calculado utilizando suma modulo-2 de las decisiones fuertes sobre cada $M_{j,i'}$, mientras la función ϕ puede ser implementada fácilmente usando una tabla de consulta (*lookup table*).

Alternativamente, el algoritmo min-sum simplifica el cálculo de (4.39) aún más por reconocer que el término correspondiente al más pequeño $M_{j,i'}$, domina el término del producto y entonces el producto puede aproximarse mediante un mínimo:

$$E_{j,i} \approx \prod_{i'} \operatorname{sign} M_{j,i'} \underbrace{\min}_{i'} \left| M_{j,i'} \right|. \tag{4.44}$$

De nuevo, el producto de los signos puede calcularse mediante la adición modulo-2 de las decisiones fuertes sobre cada $M_{j,i'}$, y así el algoritmo min-sum resultante requiere solo sumas y el cálculo de mínimos.

Capítulo 5

ANÁLISIS DE LOS CÓDIGOS LDPC, GENERACIÓN DEL CÓDIGO, EFECTO DE CUANTIZACIÓN Y VARIACIÓN DEL PESO DE COLUMNA DE H

En este capítulo se realizará el análisis de los códigos LDPC basados en arreglos y se mostrará los resultados de las distintas simulaciones realizadas. Además se presentarán las definiciones de error-floor y conjuntos absorbentes, junto con el estudio del efecto de la cuantización de los mensajes en el rendimiento de los decodificadores de los códigos LDPC basados en arreglos.

5.1 Códigos lineales de bloque

Para iniciar es importante tener claro que es un código de bloque. Por consiguiente, un código C es la colección de codewords c, $C=\left\{c : c\cdot H^T=0\right\}$. En donde H es la matriz de comprobación de paridad para tal código.

Ejemplo 5.1

Encontrar el código, su respectiva lista de codewords y calcular la distancia mínima para la siguiente matriz de comprobación de paridad H:

$$H=\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \qquad \text{(E.5.1.1)}$$

Utilizando un programa sencillo en MatLab se obtiene que el código correspondiente para la matriz H dada, está conformado por la colección de codewords siguiente:

codewords =

0	0	0	0	0	0	0
0	0	0	1	1	0	1
0	0	1	0	1	1	1
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	1	0	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	1	1
1	0	1	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	1	0	1
1	1	0	1	0	0	0
1	1	1	0	0	1	0
1	1	1	1	1	1	1

Donde cada codeword cumple con la restricción $c \cdot H^T = 0$.

La distancia mínima para este código es 3, ya que el menor número de bits con valor distinto de cero en cualquiera de los codewords del código es 3 (excluyendo el codeword que contiene solo elementos cero).

El código dado por el Ejemplo 5.1 es un código de bloque lineal y representa el código lineal de bloque (7,4), donde la longitud del codeword es N=7 y el tamaño del mensaje es k=4, este código también se puede obtener por medio de la matriz generador G:

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(E.5.1.2)}$$

El código lineal de bloque es representado por su matriz generador G de dimensión $k \times N$, con la cual se realiza una multiplicación módulo 2 por los vectores de símbolo de mensaje m de k bits para producir los codewords de N bits.

donde,

$$c = mG. \qquad \text{(E.5.1.3)}$$

Por lo tanto, el codificador y decodificador de un esquema de codificación lineal de bloque utiliza multiplicaciones de matrices con las matrices generador/comprobación de paridad, en lugar de utilizar métodos de búsqueda de tablas (*table lookup methods*) con todas las $2^K \times N$ matrices de codewords.

5.2 Códigos LDPC

La construcción de los códigos LDPC simplemente envuelve reemplazar un número pequeño de los valores de una matriz por 1s, de tal forma que

las filas y columnas tengan la distribución de grado requerida. En muchos casos, la asignación de las entradas aleatoriamente en H producirá un código LDPC razonable. Sin embargo, la construcción de H puede afectar el rendimiento del decodificador de Suma-Producto, de manera significativa para algunos códigos, y también la complejidad de la implementación del código.

Mientras no hay una fórmula para un 'buen' código LDPC, hay un número de principios que informan y muestran el camino a seguir al diseñador de códigos. Las primeras decisiones obviamente son el grado de distribución a elegir y como construir la matriz con estos grados, es decir, pseudo-aleatoriamente o con algún tipo de estructura. Cualquiera que sea la construcción elegida, las características a considerar incluyen el *girth* del gráfico de Tanner y la mínima distancia del código [3].

El análisis se basa en las propiedades de un código LDPC, las cuales influyen en el rendimiento de la decodificación iterativa. La selección de las distribuciones de grado se le conoce comúnmente como '*diseño del código*' y los métodos para asignar las ubicaciones de las entradas de valor 1 en la matriz de comprobación de paridad están referidos usualmente como '*construcción del código*'. Es importante tener en cuenta que, cuando se consideran las propiedades de un código LDPC, frecuentemente en realidad se están teniendo en cuenta las propiedades de una matriz de comprobación de paridad para una elección en particular para este código. La elección de una matriz de comprobación diferente para el mismo código puede comportarse de forma distinta al utilizar la decodificación de suma-producto. En contraste a esto, con la decodificación ML la probabilidad de una decodificación incorrecta es determinada únicamente por la distribución del peso de los codewords, que a su vez es independiente de elegir la matriz generador o la matriz de comprobación de paridad para representar el código.

5.2.1 Construcción del Código "Códigos LDPC basados en arreglos (*Array-based LDPC Codes*)"

Los códigos *Array-based* son un tipo de códigos regulares LDPC estructurados [11], los cuales están parametrizados por un par de enteros (p,γ), tal que $\gamma \leq p$, donde p es número primo diferente de 2 (primo impar). Este tipo de códigos se representan por medio de la matriz de comprobación $H_{p,\gamma}$, dada por:

$$H_{p,\gamma}, = \begin{bmatrix} I & I & I & \cdots & I \\ I & \sigma & \sigma^2 & \cdots & \sigma^{p-1} \\ I & \sigma^2 & \sigma^4 & \cdots & \sigma^{2(p-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ I & \sigma^{(\gamma-1)} & \sigma^{(\gamma-1)2} & \cdots & \sigma^{(\gamma-1)(p-1)} \end{bmatrix}, \tag{5.1}$$

donde σ es una matriz de permutación de dimensión $p \times p$ de la forma:

$$\sigma = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{5.2}$$

La tasa de este tipo de códigos es $R = 1 - \dfrac{\gamma p - \gamma + 1}{p^2}$.

A partir de la estructura presentada en la ecuación (5.1) se construyeron matrices para los códigos LDPC basados en arreglos de longitud N=289 con p=17 y N=2209 con p=47. Además, se utilizó el mismo procedimiento para la construcción de una matriz de comprobación de paridad de longitud N=1296 con p=36 la cual tiene las características compatibles con el estándar IEEE 802.11n [12].

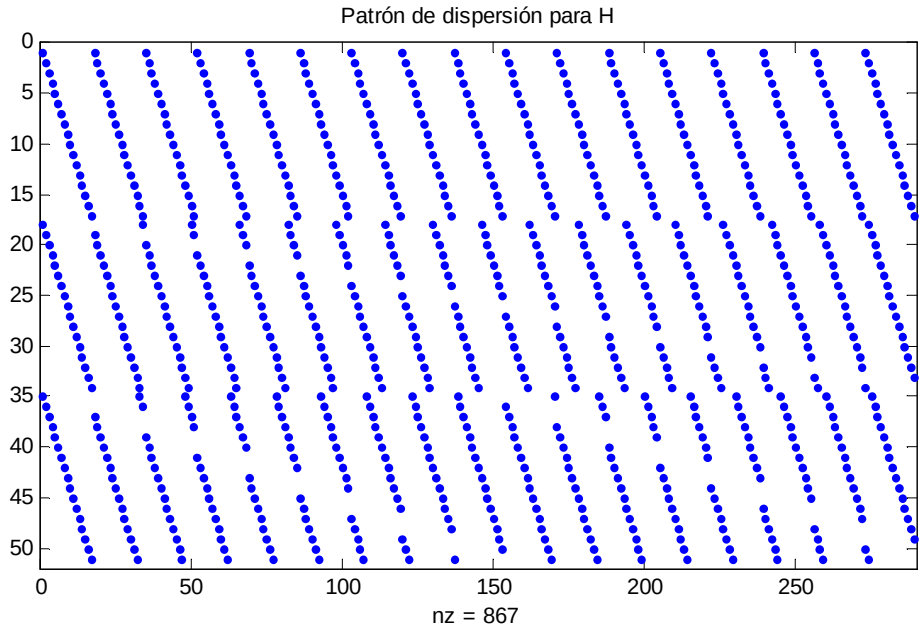


Figura 5.1 Densidad de 1s para la matriz del código LDPC basado en arreglos de longitud N=289 con $p=17$ y $\gamma=3$.

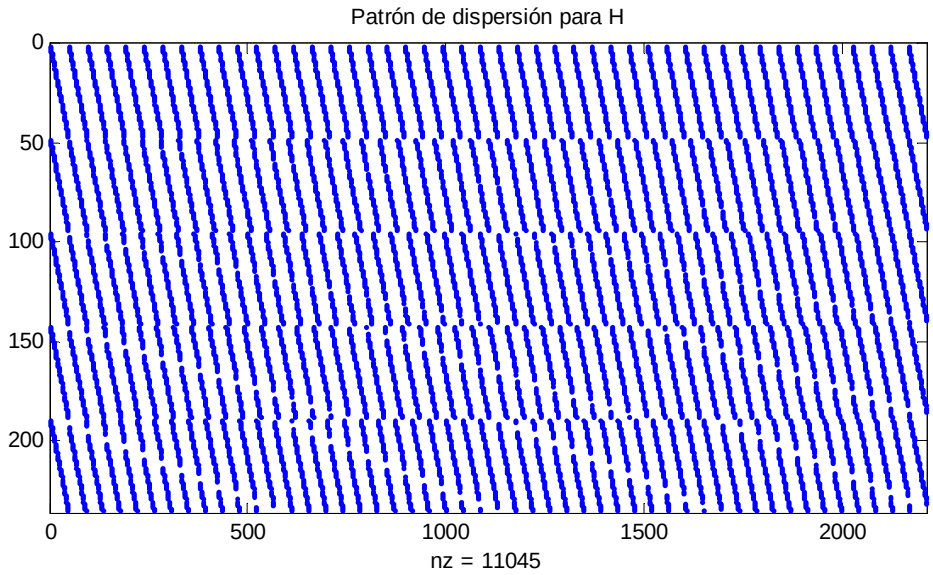


Figura 5.2 Densidad de 1s para la matriz del código LDPC basado en arreglos de longitud N=2209 con $p=17$ y $\gamma=5$.

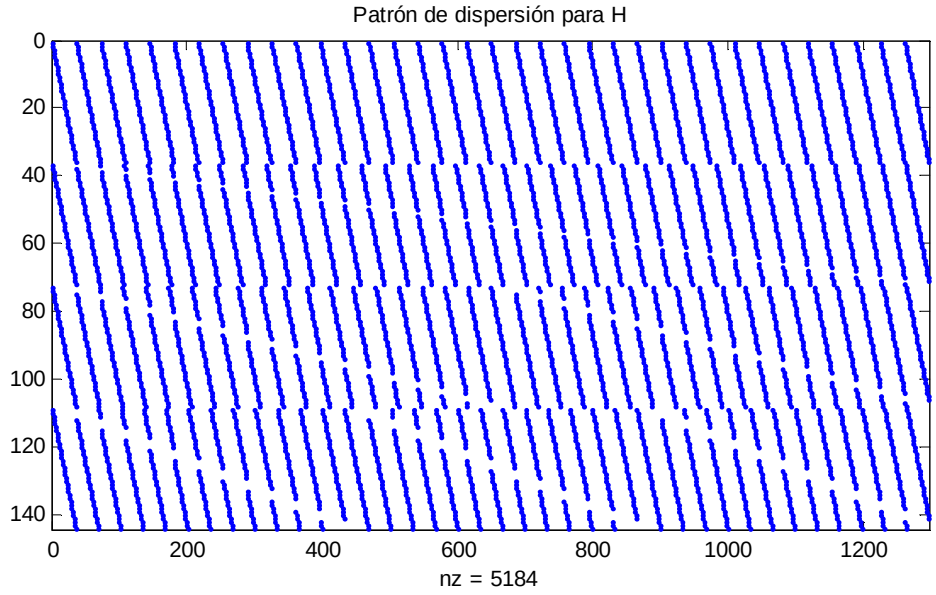


Figura 5.3 Densidad de 1s para la matriz del código LDPC basado en arreglos de longitud N=1296 con $p = 36$ y $\gamma = 4$.

Las Figuras 5.1, 5.2, 5.3 muestra el patrón de dispersión (elementos distintos de cero) para las matrices de comprobación de paridad de los códigos LDPC basados en arreglos con codewords de longitud N=289 con un peso de columna de 3, N =2209 con un peso de columna de 5 y N=1296 con un peso de columna de 4, respectivamente. Se puede observar una de las características principales de este tipo códigos, la cual es que presenta una baja densidad de elementos distintos de cero en sus matrices representativas.

5.2.2 *Error-floor* en los códigos LDPC

Los códigos LDPC son famosos por su rendimiento destacado en regiones de relación señal-a-ruido (SNR) moderadas. Sin embargo, para una SNR suficientemente alta, algunos de estos códigos presentan un cambio rápido en la pendiente de la curva que representa la relación de error de bit (BER *bit-error rate*) versus la SNR. Esta variación en la pendiente de la curva de la BER es conocida como el *error-floor* del código. Para una gran variedad de códigos, incluyendo los códigos LDPC basados en arreglos, los error-floors aparecen en valores elevados de

BER. Este fenómeno impide el uso de este tipo de códigos en sistemas de comunicación de almacenamiento y espacio profundo que necesiten enormemente bajo BER. Para muchos códigos, los error-floors no se pueden observar directamente, debido a que están fuera del alcance de las técnicas de simulación estándar.

Se han realizado muchos trabajos enfocados en caracterizar el error floor en los códigos LDPC. Se demostró que este comportamiento está asociado con la suboptimalidad de los decodificadores de paso-de-mensaje prácticos sobre los gráficos con ciclos. Mackay y Postol [13] reconocieron que algunas clases de no-codewords, a los cuales llamaron codewords-cercanos (*near-codewords*), pueden resultar en una falla en el decodificador, particularmente en (a, b) codeword-cercano es una cadena binaria de peso a cuyo síndrome tiene peso b (para la cadena x recibida, el síndrome s es definido como $s = Hx^t$).

5.2.2.1 Conjuntos Absorbentes (*Absorbing Sets*)

Una representación conveniente de la matriz de comprobación de paridad H de un código binario lineal está dada en términos de su factor o gráfico de Tanner. [14]. En particular, $G_H = (V, F, E)$ denota un gráfico bipartito, en el cual el conjunto de vértices V se asocia con los N bits del código (columnas de H) y el conjunto F corresponde a las m comprobaciones del código (filas de H). El conjunto de bordes (*edge set*) E se define por la estructura de H: en particular, existe un borde $e(i, j) \in E$ si y solo si $i \in V$ y $j \in F$. Los elementos de V son llamados "*nodos de bit*" y los pertenecientes a F se les conoce como "*nodos de comprobación*".

Para los códigos LDPC basados en arreglos, el factor gráfico asociado con $H_{p,\gamma}$ no presenta ciclos de longitud cuatro, y de esta forma el *girth* es como mínimo seis. Para cualquier subconjunto D de V, N_D denota el

subconjunto de los nodos de comprobación vecinos a los elementos de D. Para cualquier subconjunto D de V, $\mathcal{E}(D)$ (respectivamente $O(D)$) son el conjunto de vértices vecinos D en F en el gráfico G con grado par (impar) con respecto a D.

Definición 1: Dado un par de enteros (a, b), un conjunto absorbente (a, b) es un subconjunto $D \subseteq V$ de tamaño a, con $O(D)$ de tamaño b con la propiedad de que cada elemento de D tiene estrictamente menos vecinos en $O(D)$ que en $F \setminus O(D)$. Se dice que un conjunto absorbente (a, b) D, es un conjunto absorbente completo (a, b), si además, todos los nodos de bit en $V \setminus D$ tiene estrictamente más vecinos en $F \setminus O(D)$ que en $O(D)$.

Un ejemplo de un conjunto absorbente (4, 4) está dado en la Figura 5.4 donde los círculos oscuros representan el conjunto de bits D, los cuadros oscuros constituyen el subconjunto $O(D)$, los cuadros blancos representan el conjunto $\mathcal{E}(D)$, $E(D,O(D))$ está dado por las líneas sólidas y $E(D,\mathcal{E}(D))$ por las líneas punteadas. Se observa que cada elemento en D tiene más vecinos con grado par, que los que corresponden al grado impar. Todos los nodos de comprobación que no están en la figura son denotados por cuadros vacíos. Para que este conjunto sea un conjunto absorbente completo, cada nodo de bit que no está en la figura debe tener también estrictamente más cuadros vacíos como vecinos que cuadros oscuros. Nótese que $D \subseteq V$ es un conjunto absorbente completo si y solo si para todo $v \in V$, $\mathrm{wt}(Hx_{D \triangle v}) > \mathrm{wt}(Hx_D) = b$, donde $D \triangle v$ denota la diferencia simétrica entre D y $\{v\}$, $\mathrm{wt}(y)$ es el peso de Hamming de una cadena binaria con soporte D.

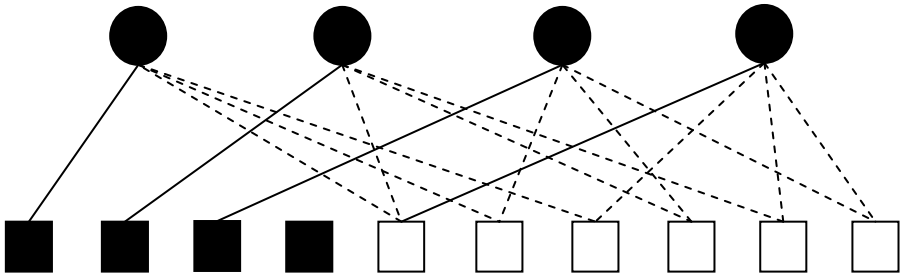


Figura 5.4 Ejemplo del conjunto absorbente (4, 4).

El nombre de “conjunto absorbente” indica la naturaleza absorbente de estos objetos combinatorios, en el sentido que actúan como estados de mínimos locales para el algoritmo de decodificación: al entrar en esta configuración el algoritmo de decodificación es absorbido en ésta. Para el caso especial del algoritmo de inversión de bit (*bit-flipping algorithm*), la configuración descrita como conjunto absorbente completo, es estable, ya que cada nodo de bit recibe estrictamente más mensajes de los nodos de comprobación vecinos que refuerzan su valor, que mensajes que mensajes que indiquen el valor opuesto del bit. Sin embargo, los conjuntos absorbentes también controlan el comportamiento de decodificadores de paso de mensaje más sofisticados, tales como el algoritmo de Suma-Producto.

5.2.3 Esquema de Cuantización

Con el fin de reducir la complejidad de la implementación, los mensajes en un decodificador práctico de paso de mensajes MP son cuantizados. El esquema de cuantización es denotado como $\mathrm{Q}a.b$, en donde a representa la parte entera, b la parte fraccional y un bit adicional es utilizado para la representación del signo, de tal manera que el número total de bits utilizados para la cuantización es $a+b+1$. El uso del esquema de cuantización conduce a un efecto en el rendimiento, el cual se analizará con los resultados de las simulaciones.

Basados en el análisis y los resultados presentados en [15] (efecto de los conjuntos absorbentes y esquema de cuantización), se realizaron las siguientes simulaciones para observar que tanto influye en el rendimiento del código, la elección de los valores a y b para el esquema de cuantización. En este caso se utilizó la matriz de comprobación $H_{p,\gamma}$ para el código LDPC basado en arreglos con longitud N=2209 con $p = 47$ y $\gamma = 5$.

Es claro que a medida que se aumentan los valores de a y b, la complejidad de los cálculos es mayor. La Figura 5.5 muestra una comparación del rendimiento del decodificador para a+b=6, de la misma manera la Figura 5.6 presenta la comparación del rendimiento del decodificador para el caso cuando a+b =7.

Se puede observar que existe un mejor comportamiento en el rendimiento cuando el valor de a cambia de 3 a 4, de hecho la diferencia es significativa. Por otro lado cuando el valor de a cambia de 4 a 5 las curvas de rendimiento presentan un comportamiento muy parecido, aunque se presenta un cambio en el perfil de error. Además, cabe resaltar que el tiempo de simulación necesario para obtener las curvas del rendimiento fue mayor cuando se utilizó el de a=3, en los casos de a=4 y a=5 el tiempo de simulación fue similar. Como se presenta en [15], existen configuraciones de ruido que causan ciertos errores de conjuntos absorbentes dependiendo del esquema de cuantización elegido. Sin embargo dichos errores se pueden corregir por completo o convertir en otra configuración de conjunto absorbente (por medio de alguna relación topológica de estos, la cual se puede analizar en el gráfico de Tanner) con menos efecto negativo en el rendimiento del decodificador.

Teniendo en cuenta que el mejor rendimiento del decodificador se presenta cuando a=4, el siguiente paso es observar el comportamiento para distintos valores de b. Para este análisis inicialmente se utiliza la

matriz $H_{p,\gamma}$ que representa el código LDPC basado en arreglos con N=2209 con $p=47$ y $\gamma=5$. Los resultados de esta simulación se presentan en la Figura 5.7.

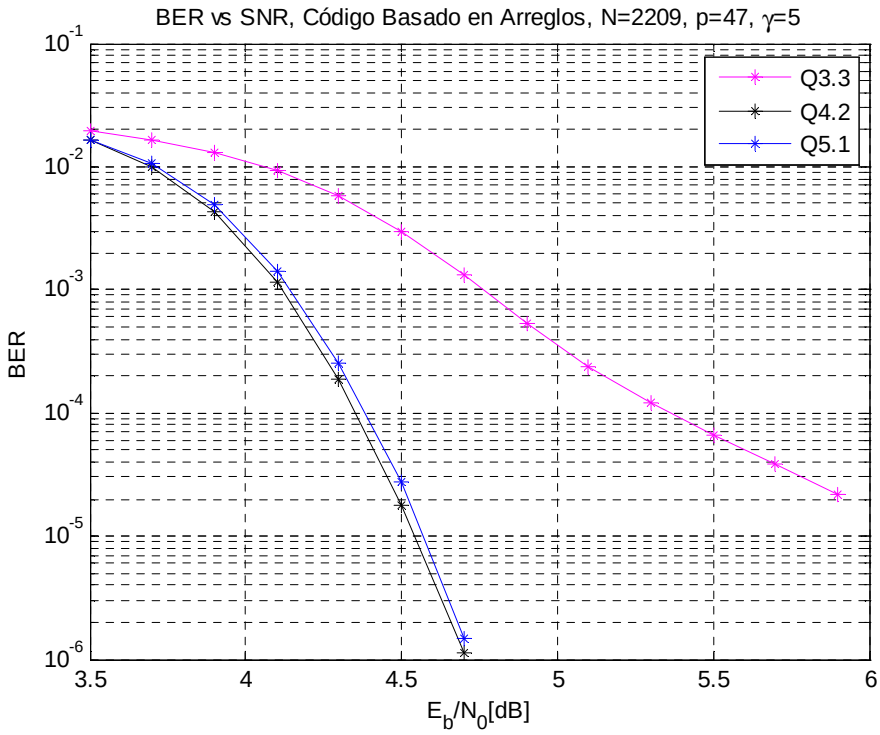


Figura 5.5 Comparación del rendimiento del decodificador para diferentes valores de cuantización.

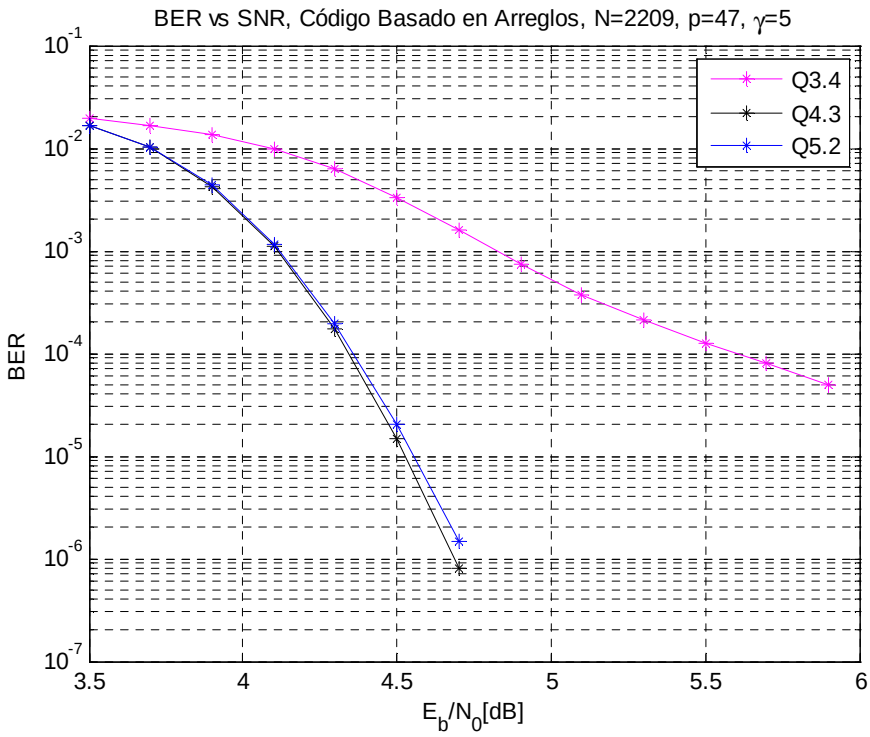


Figura 5.6 Comparación del rendimiento del decodificador para diferentes valores de cuantización.

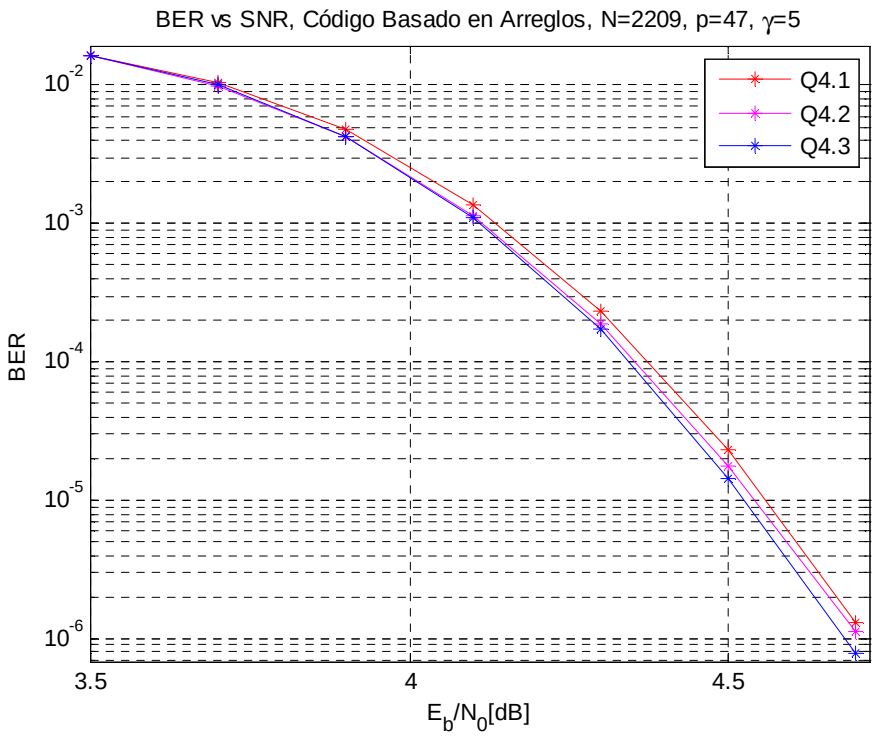


Figura 5.7 Comparación del rendimiento del decodificador para diferentes valores de cuantización $Q4.b$.

La Figura 5.7 muestra el rendimiento del decodificador para una cuantización $Qa.b$, donde a=4 y b toma los valores de 1,2 y 3. En este caso, se observa que la variación del parámetro b no afecta de manera significativa el rendimiento.

Ahora se utiliza la matriz $H_{p,\gamma}$ que representa el código LDPC basado en arreglos con N=1296 con $p=36$ y $\gamma=3$. La Figura 5.8 muestra los resultados de la simulación para este código.

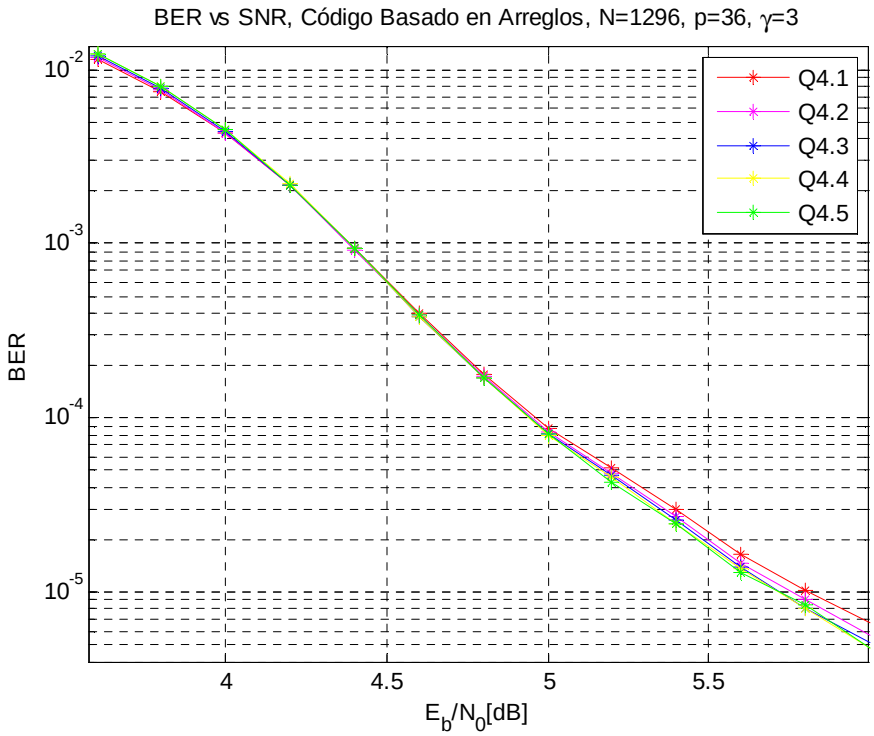


Figura 5.8 Comparación del rendimiento del decodificador para diferentes valores de cuantización $Q4.b$.

A partir de estos resultados se logra corroborar que una variación del valor b que representa la parte decimal de la cuantización, no tiene un efecto predominante en el rendimiento del decodificador, cuando el parámetro a es fijo, en este caso a=4. De hecho se observa en las figuras 5.7 y 5.8 que para los valores de BER y SNR presentados, el rendimiento mejora levemente a medida que se aumenta el valor de b, aunque en [15]

este efecto cambia, para regiones muy bajas de BER a medida que aumenta el valor de b, el rendimiento del decodificador disminuye.

Para nuestro caso no fue posible obtener valores de BER muy bajos debido al tiempo de simulación necesario para obtener estos valores y a la falta de un equipo de cómputo potente necesario para este tipo de simulaciones.

5.2.4 Diseño del Código "Variación del peso de las columnas de H"

En primer lugar, se busca determinar el papel del peso de una columna en el rendimiento de decodificación de los códigos LDPC basados en arreglos, para un esquema de cuantización dado. Para este análisis nos basamos en los resultados obtenidos previamente y en [15]. Por lo cual, se determina utilizar un esquema de cuantización de $\mathrm{Q}4.2$.

Se consideran los códigos LDPC basados en arreglos, se varía el peso de la columna ω_{c} (*column weight*) y el peso de la fila ω_{r} (*row weight*) se mantiene en un valor fijo.

A continuación en la Figura 5.9, se presenta la curva de rendimiento para el código LDPC basado en arreglos con una longitud N=289, en donde el peso de columna varia $\gamma = 3$, $\gamma = 4$ y $\gamma = 5$.

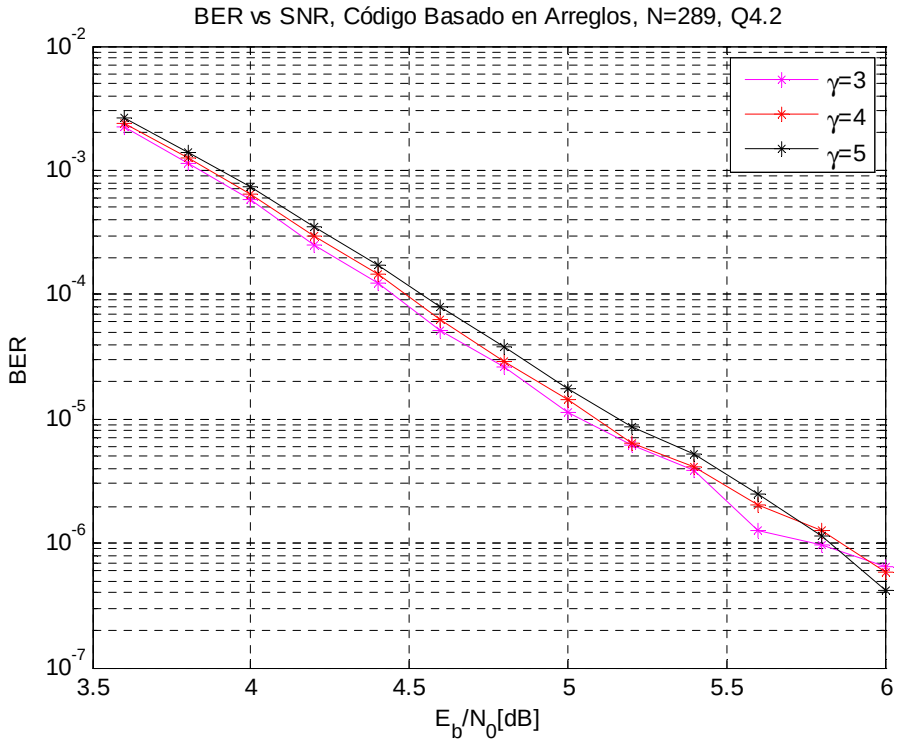


Figura 5.9 Comparación del rendimiento del decodificador para diferentes valores de peso de columna.

De la misma forma que en la figura anterior, la Figura 5.10 presenta el comportamiento del rendimiento de un código cuando se varia el peso de columna para valores de $\gamma = 3$, $\gamma = 4$ y $\gamma = 5$, pero en este caso el código utilizado para las simulaciones es el código LDPC basado en arreglos con una longitud N=1296.

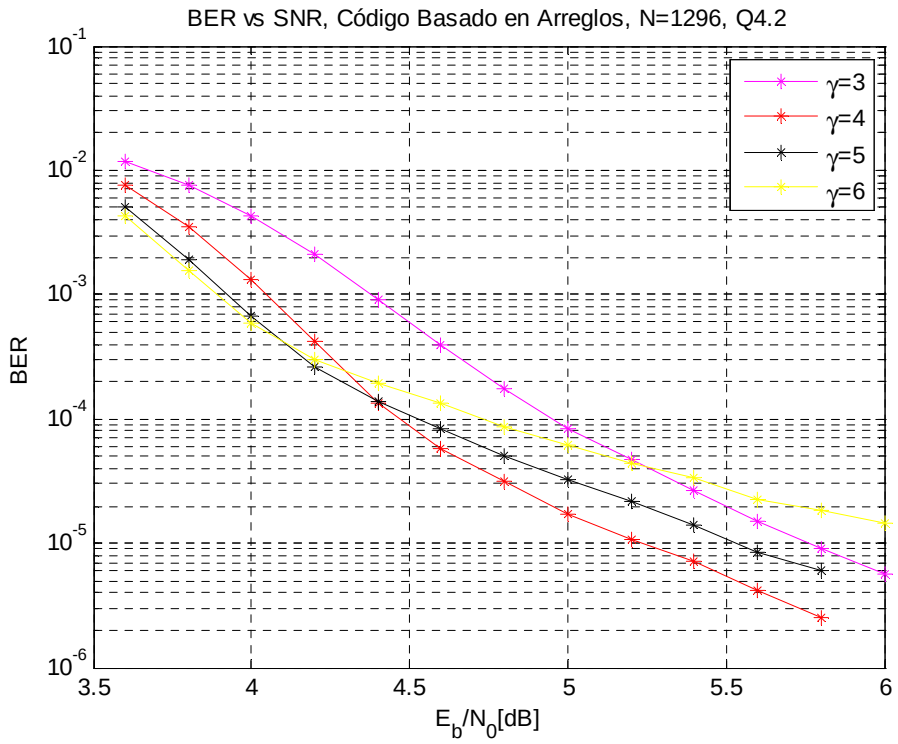


Figura 5.10 Comparación del rendimiento del decodificador para diferentes valores de peso de columna.

Las figuras 5.9 y 5.10 muestran la simulación del rendimiento de códigos regulares LDPC basados en arreglos, con una variación del peso de columna, sobre un canal AWGN. Para estos códigos se observa que si el peso de columna aumenta, existe una ligera ventaja en rendimiento para valores moderados de SNR. Sin embargo, para niveles de alto ruido el efecto del incremento del peso de columna es reducir el rendimiento de la corrección de errores. El efecto negativo de pesos de columna grandes en canales de alto ruido puede explicarse observando que si dos bits del código en una comprobación de paridad dada están en error (o borrados) es más difícil (o imposible) determinar y corregir los bits erróneos usando esta ecuación de comprobación de paridad. En los códigos con un gran peso de columna, el peso de fila debe también incrementar proporcionalmente, por lo que cada ecuación de comprobación de paridad incluye más bits del codeword, incrementando la probabilidad (*likelihood*) de que dos bits en cualquier comprobación de paridad dada estén en error. Este es un problema menor en canales con alta relación señal a

ruido ‘SNR’ (*signal-to-noise-ratio channels*), donde solo muy pocos bits del código son afectados por el ruido. En realidad, en estos canales un gran peso de columna puede mejorar la velocidad de convergencia de la decodificación de suma-producto además de mejorar la distancia mínima del código.

CONCLUSIONES Y TRABAJO FUTURO

El objetivo principal de este trabajo era el análisis de las estructuras y características de desempeño de los códigos LDPC.

Para cumplir con este objetivo inicialmente se desarrolló un programa interactivo basado en MatLab para la enseñanza de comunicaciones digitales. Con este programa se ilustraron de manera amigable, clara y concisa los fundamentos básicos relacionados con los sistemas de comunicaciones digitales, reforzando así los conocimientos en los conceptos de modulación digital, relación señal a ruido, probabilidad de error y el teorema de capacidad de Shannon, debido a que el programa permite visualizar e interactuar con estos temas.

Posteriormente se realizó una investigación abarcando los temas de codificación canal y de los códigos de corrección de error. En base a esta investigación se observó que gracias al uso de la codificación de canal en un sistema de comunicación se reduce la probabilidad de error de manera significativa, lo cual lleva a la transmisión de información de manera confiable en canales con presencia de ruido. De la misma forma es posible reducir la potencia de transmisión para una tasa de error de bit fija teniendo un sistema cuya información está codificada. Además se analizó la diferencia en la estructura entre los códigos de bloque y los códigos convolucionales. En la codificación de bloque se entregan bloques de datos de gran longitud a partir de un bloque con menor cantidad de datos, mientras que la codificación convolucional se realiza mediante una máquina de estados finita que da salida a uno o más símbolos de código por cada dato de entrada.

Continuando con este proceso de investigación se llegó al análisis de los códigos LDPC, estudiando su codificación y los principales algoritmos de

decodificación utilizados. A partir de esto se encontraron las principales diferencias con respecto a las otras clases de códigos bloque, por ejemplo, la matriz de comprobación de baja densidad, que es de gran importancia para el proceso de decodificación. Una de las características más importantes de los códigos LDPC es que su rendimiento es muy cercano al límite de capacidad. Se observaron las ventajas que se tienen al utilizar un algoritmo de decodificación suave, estando entre las más importantes el hecho de que dicho algoritmo puede utilizar operaciones logarítmicas basadas en probabilidades en lugar de multiplicaciones. De esta forma se logró reducir la complejidad de las simulaciones realizadas. Aunque las simulaciones presentadas en este trabajo tomaron un tiempo considerable, al utilizar algoritmos que no estén basados en la relación logarítmica los resultados tardarían mucho más tiempo.

Para obtener las curvas de SNR (*Signal to Noise Ratio*) versus BER (*Bit-Error Rate*) se necesitó de un tiempo considerable. Para longitudes de código pequeñas los resultados se obtuvieron en un promedio de dos semanas, para longitudes intermedias alrededor de 3 semanas y para la longitud de código más grande utilizada (N=2209), las simulaciones tardaron en promedio un mes. Por tal motivo el tiempo no fue el suficiente para obtener los resultados para SNR mayores. Por consiguiente no fue posible obtener valores de BER más bajos, los cuales permitirían el análisis del error-floor. Como se muestra en algunas investigaciones recientes en este campo, para llevar a cabo estas simulaciones se necesitan equipos de cómputo potentes y/o implementaciones del decodificador en FPGA, con los cuales se podrían obtener resultados de BER mucho más bajos en alrededor de dos semanas para longitudes de código del orden de 2000.

Durante este trabajo se desarrolló un programa para obtener la familia de codewords de un código para una matriz de comprobación de paridad dada, junto con el algoritmo de decodificación de suma-producto y un programa para construir y obtener cualquier matriz de comprobación de

paridad para un código LDPC basado en arreglos con (p,γ) dados. Este tipo de códigos se diseñan en base a una matriz de permutación, lo que permite que su matriz de comprobación de paridad sea construida fácilmente.

Se estudió el efecto de la elección de un esquema de cuantización sobre el rendimiento del decodificador del código realizando extensas simulaciones. Es así que partiendo del mejor resultado obtenido del rendimiento para un esquema de cuantización y basados en investigaciones recientes se decidió utilizar una cuantización Q4.2 para el análisis posterior, en donde se utilizan cuatro bits para representar la parte entera, dos para la parte fraccionaria y un bit adicional para representar el signo del mensaje transmitido entre los nodos del decodificador.

Los resultados empíricos obtenidos del análisis de la variación del peso de la columna de una matriz de comprobación de paridad muestran que al utilizar una matriz con longitud corta (N=289) esta variación no influye de manera significativa en el rendimiento del código. Sin embargo, para una matriz con una longitud de N=1296 esta variación es más significativa, presentando un rendimiento pobre para un peso de columna de 3 en regiones de SNR bajas. Para un peso de 6 el rendimiento se ve muy afectado en regiones de SNR mayores a 5 dB presentando el peor caso obtenido para esta región. El peso de columna con mejor rendimiento para esta matriz fue 4, presentando en buen rendimiento para bajos y altos valores de SNR en comparación a los otros resultados.

A partir de las simulaciones obtenidas y del tiempo requerido para obtener estos resultados, se puede definir un esquema de cuantización (Q4.2) para futuras implementaciones de decodificadores de paso de mensaje, lo cual reduce la complejidad de tales implementaciones. Además, se puede partir de un valor de peso de columna igual a 4 para el diseño de matrices de comprobación de longitud moderada.

Este trabajo es la base para las posibles investigaciones futuras en el área de la codificación y códigos LDPC, como por ejemplo el análisis de los conjuntos absorbentes, culpables del efecto negativo en el rendimiento de los decodificadores para los códigos LDPC en las regiones de alta SNR. Así mismo la implementación de los distintos tipos de decodificadores en FPGA podría facilitar la observación del comportamiento de los códigos LDPC y su respectivo análisis.

APÉNDICES

A.1 Generación de Codewords

El siguiente código sencillo fue desarrollado en MatLab para generar el conjunto de codewords pertenecientes a un código dependiendo de su matriz de comprobación de paridad H adicionalmente se encuentra la distancia mínima del código.

```
%% CODEWORDS
k=7;  %Número de bits de la secuencia.
M=2^k; %Número de filas de la Matriz H.

H=[1 0 0 1 0 1 1; 0 1 0 1 1 1 0; 0 0 1 0 1 1 1];    %Matriz
de comprobación de paridad.

[l,n]=size(H);
binario=dec2bin(0:M-1,k);
binario2=reshape(binario',1,k*M);

%%% Para generar todas las posibles combinaciones
for m=1:M*k
    vec_binario(m)=str2num(binario2(m));
end
vec_final=vec2mat(vec_binario,k);

r=1;
j=1;
d=1;

%%% Producto punto entre H y las secuencias binarias
for i=1:l
    for renglon=1:M
        resultado=vec_final(renglon,:).*H(i,:);
        s=0;
        for sum=1:k
            s=s+resultado(sum);
            if mod (s,2)==0
                s=0;
            else
                s=1;
            end
        end
        p(i,renglon)=s;
    end
end

%%% Identificando los codewords
for ii=1:M
    suma=0;
    for x=1:l
```

```
        suma=suma+p(x,ii);
    end
    if suma==0
        codewords(j,1:k)=vec_final(ii,:);
        j=j+1;
     end
end

%%% Para encontrar la distancia mínima
[l2,n2]=size(codewords);
 for sum1=1:l2
     s1=0;
     for sum2=1:k
        s1=s1+codewords(sum1,sum2);
     end
     p1(sum1)=s1;
     if p1(sum1)~=0
         aux1(d)=p1(sum1);
         d=d+1;
     end
 end
codewords
minimum_distance=min(aux1);
```

A.2 Algoritmo de decodificación Suma-Producto

En [3] se presenta el Algoritmo de Suma-Producto, donde las estradas son relaciones logarítmicas de verosimilitud para las probabilidades a priori del mensaje de cada canal. Basados en esto se desarrolló un programa en MatLab que realiza el proceso de decodificación de Suma-Producto que se muestra a continuación:

```
function vhat=Decodificador_Suma_Producto(H,R)

H=[1 1 0 1 0 0; 0 1 1 0 1 0; 1 0 0 0 1 1; 1 0 1 1 1 0]
R=[-1.3863 1.3863 -1.3863 1.3863 -1.3863 -1.3863];
Imax=200;        % Numero máximo de iteraciones
[m,N]=size (H); % m=filas de H, N=columnas de H

%Para obtener Bj, conjunto de bits en la jth ecuación de
paridad de H (1's de filas de H)
for i=1:m;
    auxB=1;
    for j=1:N;
        if H(i,j)==1;
            B(i,auxB)=j;
            auxB=auxB+1;
        end
    end
```

```
end
B
[mB,NB]=size (B);

%Para obtener Ai, representa las ecuaciones de paridad para
el ith bit del codigo (1's de columnas de H)
for j=1:N;
    auxA=1;
    for i=1:m;
        if H(i,j)==1;
            A(j,auxA)=i;
            auxA=auxA+1;
        end
    end
end
[mA,NA]=size (A);

%% INICIALIZACION
%Para obtener Mji
for i=1:N;
    for j=1:m;
         M(j,i)=R(i);
     end
end


for I=0:1:Imax  ;
%% MENSAJES DE COMPROBACIÓN
    for jB=1:mB;
        for iB1=1:NB;
            producto=1;
            comp1=B(jB,iB1);
            for iB2=1:NB;
                comp2=B(jB,iB2);
                if comp1~=comp2;
                    j=jB;
                    i=comp2;
                    producto=tanh(M(j,i)/2)*producto;
                end
            end
            j=jB;
            i=comp1;
            E(j,i)=log((1+producto)/(1-producto));
        end
    end

%% PRUEBA
    for i=1:N;
        suma=0;
        for jA=1:NA;
            j=A(i,jA);
            suma=E(j,i)+suma;
        end
```

```
        L(i)=suma+R(i)
        if L(i)<=0;
            vhat(i)=1;
        else
            vhat(i)=0;
        end
    end

%Para comprobar si c es un código de palabra válido
    s=rem((vhat*H'),2);
    if s==0 ;
        disp ('Codeword Válido')
        break;

%% BIT MESSAGES
    else
        for iA=1:mA;
           for jA1=1:NA;
                suma=0;
                comp1=A(iA,jA);
                for jA2=1:NA;
                    comp2=A(iA,jA2);
                    if comp1~=comp2;
                        j=comp2;
                        i=iA;
                        suma=E(j, i)+suma;
                    end
                    j=comp1;
                    i=iA;
                    M(j,i)=suma+R(i);
                end
           end
        end
    end
end
```

A.3 Algoritmo de decodificación Min-Sum

Realizando algunas modificaciones al algoritmo de suma-producto se puede obtener el algoritmo min-sum

```
function vhat=Decodificador_min_sum(H,R)

H=[1 1 0 1 0 0; 0 1 1 0 1 0; 1 0 0 0 1 1; 0 0 1 1 0 1]
R=[-1.3863 1.3863 -1.3863 1.3863 -1.3863 -1.3863];
Imax=50;        % Numero maximo de iteraciones
[m,N]=size (H); % m=filas de H, N=columnas de H

%Para obtener Bj, conjunto de bits en la jth ecuacion de
paridad de H (1's de filas de H)
for i=1:m;
```

```
    auxB=1;
    for j=1:N;
        if H(i,j)==1;
            B(i,auxB)=j;
            auxB=auxB+1;
        end
    end
end
B
[mB,NB]=size (B);

%Para obtener Ai, representa las ecuaciones de paridad para
el ith bit del codigo (1's de columnas de H)
for j=1:N;
    auxA=1;
    for i=1:m;
        if H(i,j)==1;
            A(j,auxA)=i;
            auxA=auxA+1;
        end
    end
end
[mA,NA]=size (A);

%% INICIALIZACION
%Para obtener Mji
for i=1:N;
    for j=1:m;
         M(j,i)=R(i);
     end
end


for I=0:1:Imax  ;
%% MENSAJES DE COMPROBACIÓN
    for jB=1:mB;
        for iB1=1:NB;
            producto=1;
            s=1;
            comp1=B(jB,iB1);
            for iB2=1:NB;
                comp2=B(jB,iB2);
                if comp1~=comp2;
                    j=jB;
                    i=comp2;
                    producto=sign(M(j,i))*producto
                    iB2
                    x(s)=abs(M(j,i))
                    s=s+1;
                end
            end
            j=jB;
            i=comp1;
```

```
            E(j,i)=producto*min(x)
        end
    end

%% PRUEBA
    for i=1:N;
        suma=0;
        for jA=1:NA;
            j=A(i,jA);
            suma=E(j,i)+suma;
        end
        L(i)=suma+R(i)
        if L(i)<=0;
            vhat(i)=1;
        else
            vhat(i)=0;
        end
    end

%Para comprobar si c es un código de palabra válido
    s=rem((vhat*H'),2);
    if s==0 ;
        disp ('Codeword Válido')
        break;

%% MENSAJES DE BIT
    else
        for iA=1:mA;
           for jA1=1:NA;
                suma=0;
                comp1=A(iA,jA);

                for jA2=1:NA;
                    comp2=A(iA,jA2);
                    if comp1~=comp2;
                        j=comp2;
                        i=iA;
                        suma=E(j, i)+suma;
                    end
                    j=comp1;
                    i=iA;
                    M(j,i)=suma+R(i);
                end
           end
        end
    end
end
```

A.4 Generación de Matrices de comprobación de paridad para los códigos LDPC basados en arreglos

```
%% Para crear la Matriz array-based
clear all;close all;clc;

p=36
y=4
r=eye(p+1);
r(1,:)=r(p+1,:);
sigma=r(1:p,2:p+1);

E(1,:)=zeros (1,p-1);
x=0;
for i=1:p
    E(2,i)=x;
    for m=3:y
        E(m,i)=(m-1)*x;
    end
    x=x+1;
end

H=zeros (y*p,p*p);

for renglon=1:y
    for columna=1:p
        H((renglon-1)*p+1:(renglon-1)*p+p,(columna-1)*p+1:(columna-1)*p+p)=sigma^E(renglon,columna);

    end
End

%% Para crear la matriz en un archivo .txt
flag=1;
[r,c,v]=find(H'==1);
[M,N]=size(H);

ft = fopen('Longitud1296_6.txt','w');
fprintf(ft, '%i %i \n',N,M);

for i=1:M
    fprintf(ft, '%i \t',p);
    fprintf(ft, '\t');
end
fprintf(ft, '\n')

paridad=zeros(p*y,p);
for i=1:p*y
    paridad(i,:)=r(flag:flag+p-1)';
    flag=flag+p;
end
```

```
for i=1:p*y

   for j=1:p
    fprintf(ft, '%i \t', paridad(i,j))
    fprintf(ft, '\t');
   end
   fprintf(ft, '\n')
end
fclose(ft);
```

A.5 Programa Interactivo basado en MatLab para la enseñanza de comunicaciones digitales

Durante el estudio de los diferentes bloques que conforman un sistema de comunicaciones digitales se desarrolló un Demo programa que presenta el estudio de diferentes temas relacionados con el sistema.

A5.1 Descripción del Programa

En la pantalla principal del software desarrollado se muestra una breve introducción sobre comunicaciones digitales (ver Fig. 1). Además, se tiene un menú en la parte superior derecha en donde se selecciona una de las siete secciones disponibles. La parte inferior derecha posee los botones de “OK” y “EXIT”, cuyas respectivas tareas son cerrar el programa o ejecutar la función deseada una vez ingresados los datos. El menú y los dos botones descritos anteriormente están siempre disponibles.

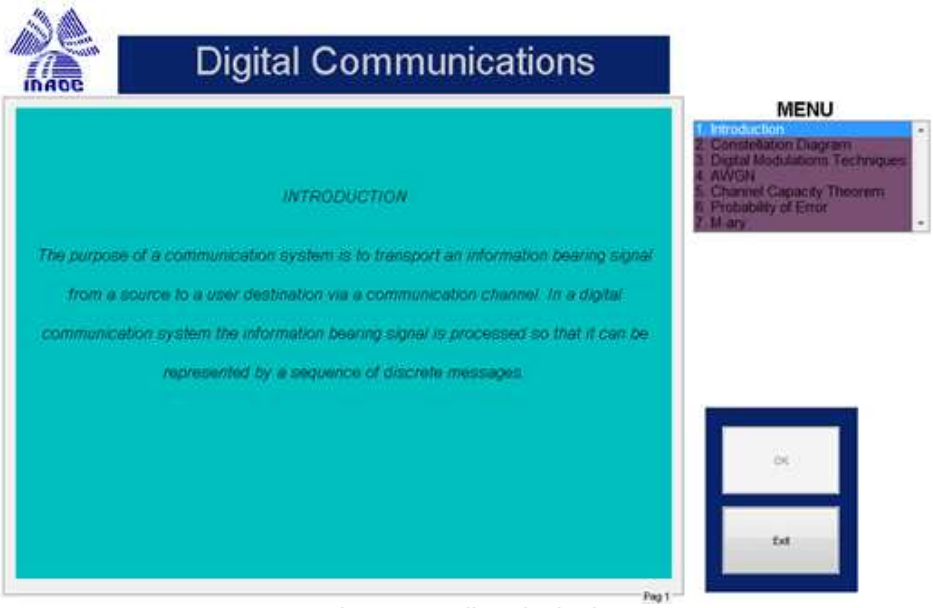


Fig. 1 Pantalla principal.

A continuación se describen cada una de las interfaces que conforman el software.

A.5.2 Diagrama de constelación

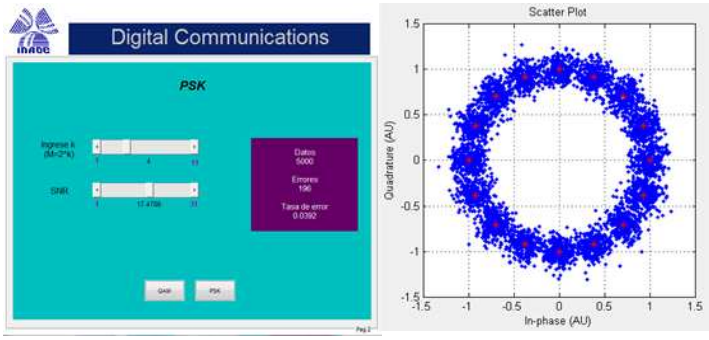
En esta sección se realiza la representación de una señal en un diagrama de constelación. Esta sección posee un submenú para elegir entre dos opciones de esquemas de modulación: PSK y QAM. Una vez elegida cualquiera de las opciones, aparece una nueva interfaz en la que es necesario ingresar dos parámetros:

- SNR. Puede variar en el rango desde 1 hasta 31.
- *k*. Es el número de bits y puede variar en el rango desde 1 hasta 11. A partir de *k* se calcula el número de señales que se pueden representar: $M=2^k$.

En la interfaz también aparece un recuadro en el cual se muestran los siguientes resultados: datos, errores y tasa de error.

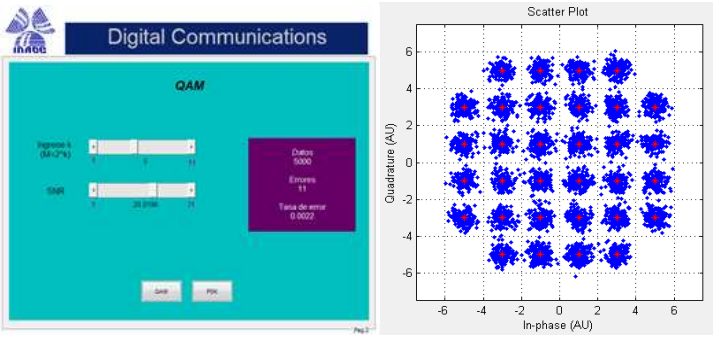
Ejemplo 1. Encontrar el diagrama de constelación para un esquema de modulación PSK con SNR=17.4706dB y número de bits *k*=4. La Fig. 2 muestra los resultados obtenidos.

Ejemplo 2. Encontrar el diagrama de constelación para un esquema de modulación QAM con SNR=17.4706dB y número de bits *k*=4. La Fig. 3 muestra los resultados obtenidos.



(a) (b)

Fig. 2 Ejemplo 1 a) Interfaz de usuario y b) Diagrama de constelación.



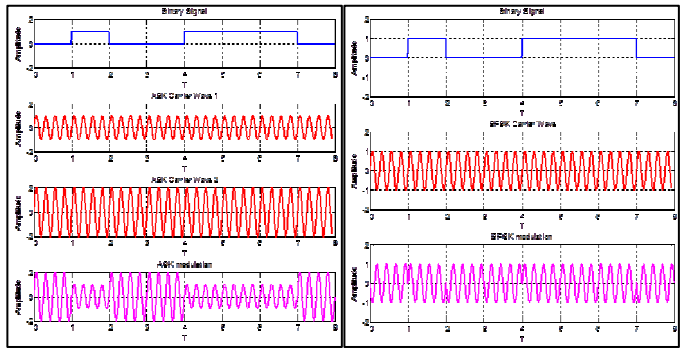
(a) (b)

Fig. 3 Ejemplo 2 a) Interfaz de usuario y b) Diagrama de constelación.

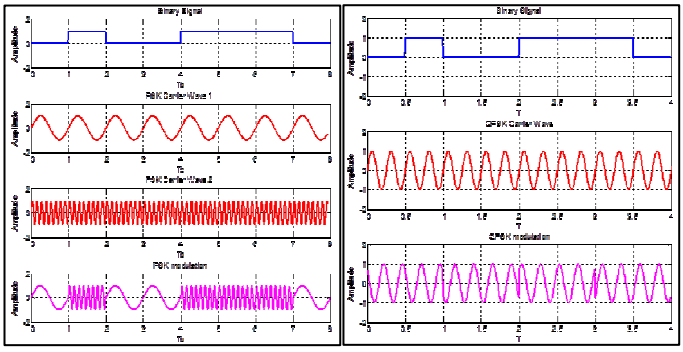
A.5.3 Técnicas de modulación digital

En esta interfaz se elige el tipo de modulación digital para realizar su representación gráfica. Las siguientes opciones se muestran en el submenú: ASK, FSK, PSK y QPSK. Al seleccionar la opción, aparece una nueva interfaz en la que es necesario ingresar una cadena de unos y ceros para ser representada.

Ejemplo 3. Representar gráficamente la cadena de bits 01001110 para cada uno de los diferentes esquemas de modulación disponibles. La Fig. 4 muestra los resultados obtenidos.



(a) (b)



(c) (d)

Fig. 4 Ejemplo 3, a) ASK, b) PSK, c) FSK y d) QPSK.
Señal binaria: azul, señales involucradas en la modulación: rojo y señal modulada: magenta.

A.5.4 Ruido Blanco Aditivo Gaussiano (AWGN)

Esta interfaz es muy parecida a la anterior, ya que se presentan las mismas opciones de esquemas de modulación en el submenú. El enfoque en este punto es visualizar la señal modulada original y la señal afectada por el ruido del canal. Al elegir cualquiera de las opciones aparece una nueva interfaz en la que es necesario ingresar, además de la cadena de bits, la SNR deseada.

Ejemplo 4. Representar gráficamente cada uno de los diferentes esquemas de modulación con y sin ruido para la cadena de bits 01001110. La Fig. 5 muestra los resultados obtenidos.

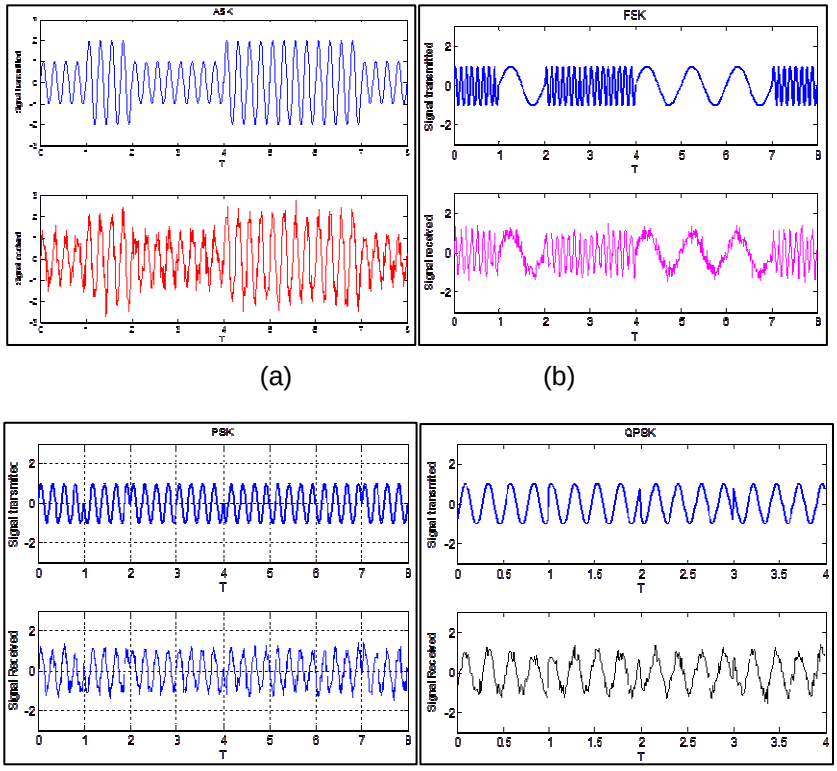
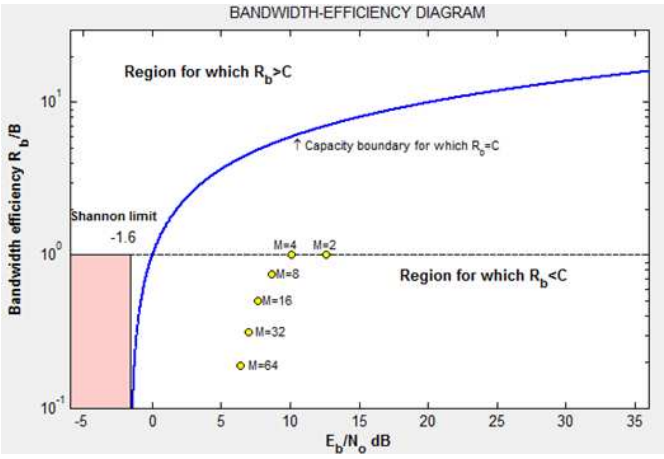


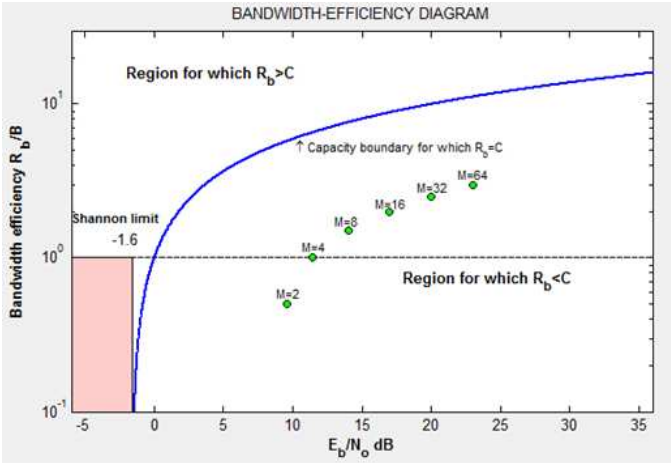
Fig. 5 Ejemplo 4, a) ASK, b) FSK, c) PSK y d) QPSK.

A.5.5 Teorema de capacidad de canal

Esta interfaz muestra un submenú con dos opciones: FSK y PSK. Al seleccionar una de las opciones, se muestra una gráfica de eficiencia de ancho de banda donde se presenta el Límite de Shannon para visualizar el desempeño de ambos tipos de modulación, cada uno para valores de *M*=2, 4, 8, 16, 32 y 64, donde cada punto corresponde a una probabilidad de error de símbolo promedio 10^{-5}. La Fig. 6 muestras las gráficas correspondientes donde también se incluye la curva de capacidad para un sistema ideal.



(a)

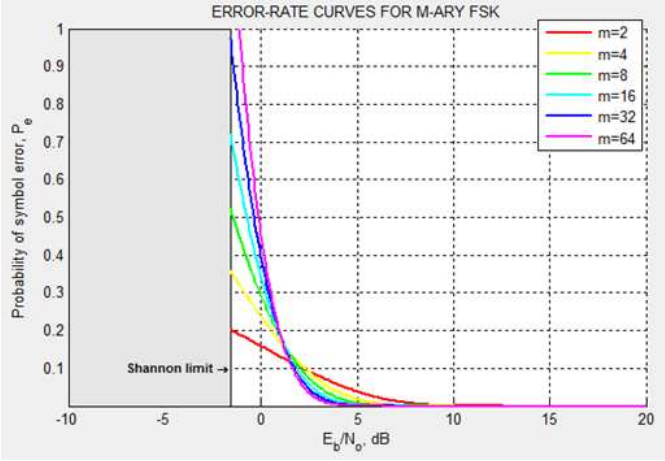


(b)

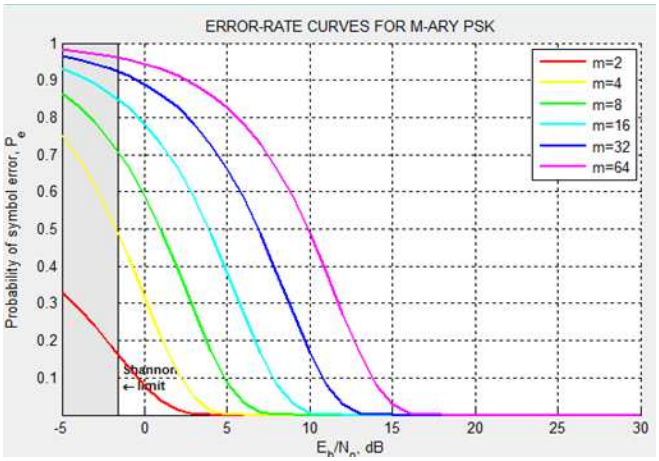
Fig. 6 Diagrama de Eficiencia de ancho de Banda a) FSK b) PSK.

A.5.6 Probabilidad de error

Al igual que la interfaz anterior, el submenú tiene las opciones: FSK y PSK. Al seleccionar una de estas, una gráfica de tasa de error es mostrada para distintos valores de *M*, en donde se observa el comportamiento de los dos tipos de modulación en términos de la probabilidad de error. Las respectivas gráficas se muestran en la Fig. 7.



(a)



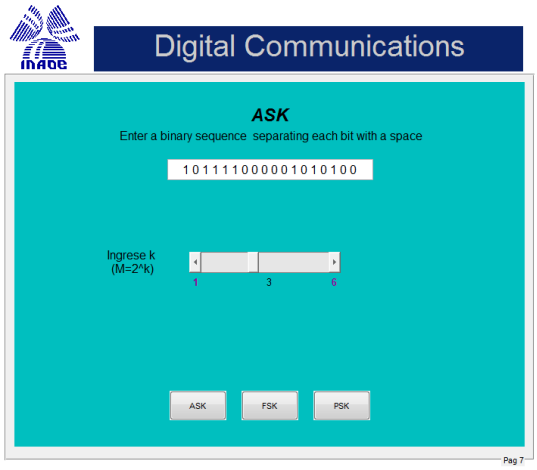
(b)

Fig. 7 Probabilidad de error a) FSK b) PSK.

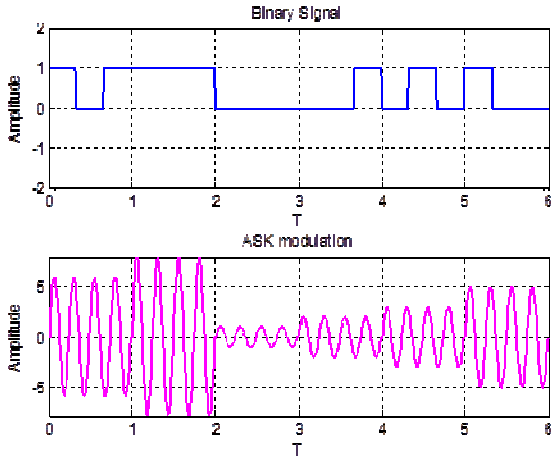
A.5.7 Modulación M-aria

Por último, además del tipo de modulación (ASK, FSK, PSK), en esta interfaz se elige el número de bits para representar la señal en el rango desde 1 hasta 6.

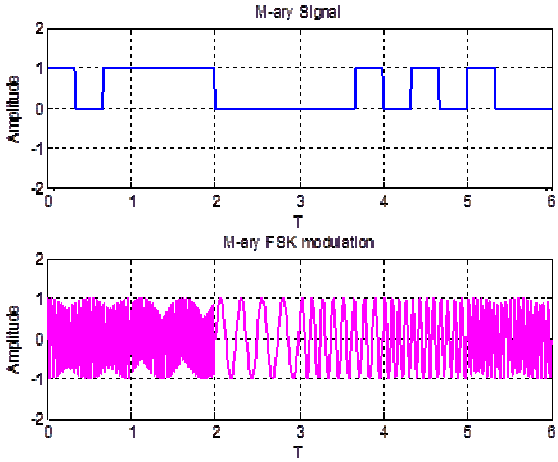
Ejemplo 5. Representar gráficamente la cadena de bits 101111000001010100 con *k*=3, para cada esquema de modulación. La Fig. 8 muestra los resultados obtenidos.


Digital Communications
ASK
Enter a binary sequence separating each bit with a space
1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 0
Ingrese k
(M=2^k)
1
3
6
ASK
FSK
PSK
Pag 7

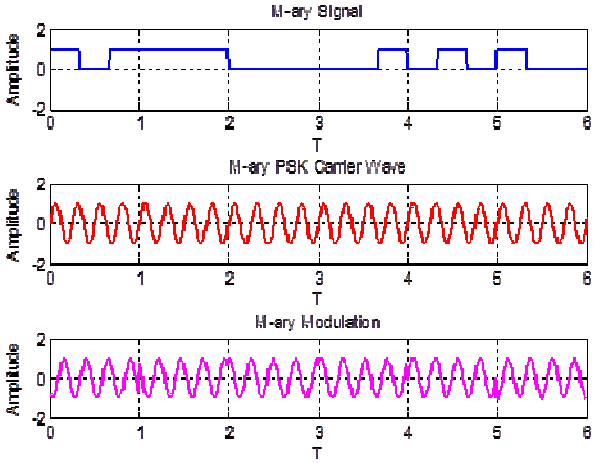
(a)


Binary Signal
Amplitude
T
ASK modulation
Amplitude
T

(b)


M-ary Signal
Amplitude
T
M-ary FSK modulation
Amplitude
T

(c)



(d)

Fig. 8 Modulación *M*-aria a) Interfaz b) ASK c) FSK y d) PSK.

LISTA DE FIGURAS

Figura 1.1 Sistema de comunicaciones típico..... 11

Figura 1.2 Modelo simplificado de un sistema de codificación..... 13

Figura 1.3 Canal binario simétrico (BSC)..... 15

Figura 1.4 Canal de borrado binario (BEC)..... 18

Figura 2.1 Problema básico de las comunicaciones punto a punto. 25

Figura 2.2 Problema básico de comunicaciones punto-a-punto desde el punto de vista del teorema de separación fuente-canal..... 26

Figura 2.3 Modelo del canal gaussiano sin memoria en tiempo discreto. 36

Figura 2.4 Densidad espectral de potencia del ruido AWGN..... 36

Figura 2.5 Diagrama de eficiencia del ancho de banda [4]..... 39

Figura 2.6 Diagrama de tasa de error..... 40

Figura 3.1 Codificador Convolucional. 54

Figura 4.1 Representación en el gráfico de Tanner de la matriz de comprobación de paridad de (E.4.5.1). Un ciclo de longitud 6 (*6-cycle*) se muestra en color rojo. 66

Figura 5.1 Densidad de 1s para la matriz del código LDPC basado en arreglos de longitud N=289 con $p=17$ y $\gamma=3$. 94

Figura 5.2 Densidad de 1s para la matriz del código LDPC basado en arreglos de longitud N=2209 con $p=17$ y $\gamma=5$. 94

Figura 5.3 Densidad de 1s para la matriz del código LDPC basado en arreglos de longitud N=1296 con $p=36$ y $\gamma=4$. 95

Figura 5.4 Ejemplo del conjunto absorbente (4, 4). 98

Figura 5.5 Comparación del rendimiento del decodificador para diferentes valores de cuantización. 100

Figura 5.6 Comparación del rendimiento del decodificador para diferentes valores de cuantización. 101

Figura 5.7 Comparación del rendimiento del decodificador para diferentes valores de cuantización $Q4.b$. 101

Figura 5.8 Comparación del rendimiento del decodificador para diferentes valores de cuantización $Q4.b$. 102

Figura 5.9 Comparación del rendimiento del decodificador para diferentes valores de peso de columna. 104

Figura 5.10 Comparación del rendimiento del decodificador para diferentes valores de peso de columna. 105

REFERENCIAS

[1] C. E. Shannon, “A mathematical theory of communication,” Bell System Technical Journal, vol. 27, pp. 379-423, 1948.

[2] S. Lin, D. J. Costello, “Error Control Coding,” Pretince Hall, 2nd Edition, 2004.

[3] S. J. Johnson, “Iterative Error Correction - Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes,” Cambridge University, 2010.

[4] W. G. Castaneda, V. Calderon, G. J. Dolecek, “Programa Interactivo Basado en MatLab para la Enseñanza de Comunicaciones Digitales,” Cumbre Internacional de las Ingenierías, 2013.

[5] T. Richardson, R. Urbanke, “Modern Coding Theory,” Cambridge University, 2008.

[6] S. Haykin, “Digital Communications,” John Wiley & Sons, Universidad McMaster, 4th Edition, 2001.

[7] M. Franceschini, G. Ferrari, R. Raheli, “LDPC Coded Modulations,” Springer, 2009.

[8] Y. Jiang, “A Practical Guide to Error-Control Coding Using MATLAB,” Artech House, 2010.

[9] E. Krouk, S. Semenov, “Modulation and Coding Techniques in Wireless Communications,” John Wiley & Sons, 2011.

[10] R. G. Gallager. *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[11] J.L Fan, “Array-codes as low-density parity-check codes,” Second Int. Symp. On Turbo Codes, Brest, France, pp. 543-46, Sept. 2000.

[12] IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements.

[13] D. J. C. MacKay and M. S. Postol, “Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes,” Electron. Notes in Theo. Comput. Sci., vol. 74, pp. 97-104, 2003.

[14] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, B. Nikolic, “Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes,” IEEE Trans. on Inform. Theory, vol. 56, no. 1, pp. 181-201, Jan. 2010.

[15] B. Amiri, S. G. Srinivasa, L. Dolecek, “Quantization, Absorbing Regions and Practical Message Passing Decoders,” Signals, Systems and Computers (ASILOMAR), pp. 1255-1259, 2012.