UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ANÁLISIS DE ESTRATEGIAS DE CODIFICACIÓN Y DECODIFICACIÓN PARA CÓDIGOS BCH UTILIZANDO UN SIMULADOR DE CANAL CON RUIDO

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:

MARIO OCTAVIO CARRAZCO DELGADO

DIRECTOR DE TESIS:

DR. VLADISLAV KHARTCHENKO

MÉXICO, D.F. AÑO 2010.

Í N D I C E G E N E R A L

AGRADECIMIENTOS

Deseo expresar mi reconocimiento al CONACyT por su apoyo al otorgarme una beca para realizar mis estudios de maestría y para desarrollar el presente trabajo de tesis.

Va mi gratitud a la Universidad Nacional Autónoma de México, donde realicé mis estudios en la carrera de Ingeniería Mecánica y Eléctrica y a la cual debo mi formación. También, quiero agradecer al Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas y a la Facultad de Estudios Superiores Cuautitlán, donde realicé mis estudios a nivel maestría, en especial a todos mis profesores del Posgrado en Ciencias de la Computación por haberme dado los conocimientos necesarios para realizar mis estudios de posgrado, muy en especial deseo agradecerle a mi tutor y director de tesis, el Dr Vladislav Khartchenko, por su infinita paciencia y apoyo, porque siempre tuvo tiempo para mí, por haberme encaminado a la realización del presente escrito, por enseñarme a no tener miedo a los campos finitos y a los extensos polinomios sobre un campo. Sin duda, el aprendizaje de las matemáticas fue una alegre experiencia.

A B S T R A C T

The objective of the research is to elaborate and validate a channel simulator with noise and distortion of many types the same as real transmissions: [(channel with noise AWGN (Additive White Gaussian Noise), acoustic noise, impulsive (transient) or interference, signal distortion (phase difference between input and output), out of reach or fading of the signal, echo, atmospheric noise (coming from electromagnetic radiations from solar and galactic origin), thermal noise (movement of free electrons inside a conductor), electrostatic noise (generated from the presence of voltage with current flow or without it), binary symmetric channel, (probabilistic) (abbreviation in English BSCp), etc.].

The intention to realise a thesis theoretical-practical is born from the necessity to protect the information, once we transmit information from one place to another or when we store information to be used later. These facts followed to do a system of coding-decoding able to detect and correct errors in a sequence of information, the difficulty of the construction was an important factor in the selection of the systems to use, BCH codes (Bose-Chaudhuri-Hocquenghen) were selected as very efficient, easy and cheap to be constructed, after comparing the error probabilities of BCH codes with the different error correction codes.

Foundation systems of transmission, modeling noise and distortion are presented. As well as the main problem of the code theory. It is also showed an analysis of the different strategies from construction, coding and decoding of codes BCH, their relation and difference with other error correction codes.

The simulator can be considered like a step for the electronic construction of a coder-decoder BCH, the idea is any person who wants to construct an electronic decoder, it will be very useful to know its computer behavior and to have the trustworthiness with a minimum of permissible error and this form to be able to apply them in the real life

Key words: transmission system, noise and distortion, modeling, error correction codes, finite fields, codes BCH.

R E S U M E N

El objetivo de esta investigación es elaborar y validar un simulador de canal con ruido y distorsión de varios tipos que se asemejan a las transmisiones reales: [canal con ruido AWGN (Additive White Gaussian Noise)**,** ruido acústico, impulsivo (transitorio) o interferencia, distorsión de la señal (diferencia de fase entre la entrada y la salida), perdida de línea o desvanecimiento de la señal, eco, ruido atmosférico (procedente de radiaciones electromagnéticas con origen solar y galáctico), ruido térmico (movimiento de los electrones libres en el interior de un conductor), ruido electrostático (generado por la presencia de voltaje con o sin flujo de corriente), canal simétrico binario (probabilístico) (siglas en inglés BSC_P), etc.].

La intención de realizar una tesis teórica-práctica nace de la necesidad de proteger la información, ya sea cuando transmitimos información de un lugar a otro o cuando la almacenamos para después usarla. Estos hechos llevaron a la realización de un sistema de codificación-decodificación capaz de detectar y corregir errores en una secuencia de información, la dificultad de la construcción constituyó un factor importante en la selección de los sistemas a utilizar, se eligieron varios códigos como los BCH (Bose-Chaudhuri-Hocquenghen) por ser muy eficientes, fáciles y baratos de construir, después de comparar las probabilidades de error de estos con los diferentes códigos correctores de errores.

Se presentan los fundamentos de sistemas de transmisión, modelación de ruido y distorsión. Así como el problema principal de la teoría de códigos. También se presenta un análisis de las diferentes estrategias de construcción, codificación y decodificación de los códigos BCH, su relación y diferencia con otros códigos correctores de errores.

Se puede considerar al simulador como un paso a la construcción electrónica de un codificador-decodificador BCH, la idea es que cualquier persona que quiera construir un decodificador electrónico, le sea de gran utilidad conocer el comportamiento computacional y tener la confiabilidad con un mínimo de error permisible y de esta forma poder aplicarlos en la vida real.

Palabras clave: sistema de transmisión, ruido y distorsión, modelación, códigos correctores de errores, campos finitos, códigos BCH.

P R E F A C I O

El origen de los códigos correctores de errores se remonta a finales de la década de 1940 con los trabajos germinales de Claude Elwood Shannon (Quién también fue uno de los primeros en representar la información en términos de bits), cuyo principal resultado es la publicación de "Una Teoría Matemática de la comunicación". El trabajo de Shannon, debe su importancia a que proporciona un estándar de rendimiento que no puede ser excedido por ningún canal de comunicaciones y también contempla los factores que limitan el rendimiento de un canal. Shannon demostró que dado un índice de entropía (promedio de información) y si la cantidad de información que se desea transmitir excede la capacidad de canal, entonces existirán errores inevitables e incorregibles durante la transmisión. Lo relevante de sus estudios, fue la demostración de que si el índice de entropía esta por debajo de la capacidad de canal, entonces existe una forma de codificar la información de modo que puede ser recibida sin errores. Aunque, en estos comienzos, la teoría de los códigos tenía un enfoque esencialmente probabilística. Shannon consideró que si existe redundancia[1] (bits sumados al mensaje codificado, de manera única), usando más símbolos y más palabras de las necesarias para transmitir un mensaje, probablemente esta redundancia es usada para mejorar la habilidad de reconocer mensajes confiablemente y comunicar diferentes tipos de información. Los sistemas que aplican esta redundancia generalmente son llamados códigos correctores de errores.

Esto marcó el nacimiento de la teoría de códigos, que es un campo de estudio relacionado con la transmisión de datos en un canal con ruido y/o distorsión y la recuperación de mensajes con errores, con lo que se abrió una importante línea de investigación en la elaboración de códigos y algoritmos de decodificación. El diseño de buenos códigos correctores de errores, desde el punto de vista teórico y práctico, es un importante problema en dicha teoría. Para determinar que tan bueno es un código teóricamente, se pueden analizar sus parámetros que frecuentemente son usados como una medida de eficiencia del mismo, mientras que desde el punto de vista práctico, un código debe de admitir una eficiente decodificación para que el código pueda ser considerado óptimo.

Posteriormente, con los primeros estudios sobre códigos cíclicos realizados por Prange en 1957, los códigos correctores de errores pasaron a un enfoque más algebraico. Desde entonces la teoría de códigos algebraica ha hecho grandes progresos en el estudio de códigos de corrección de errores que utilizan redundancia, aunque el problema de esta teoría surge en una aplicación de ingeniería es fascinante la utilización de técnicas algebraicas modernas y clásicas que involucran campos finitos, teoría de grupos y algebra de polinomios. Además, se relacionan con otras áreas como las que involucran Matemáticas Discretas, especialmente la Teoría Numérica y la Teoría de Diseños Experimentales. Surgiendo entonces muchos ejemplos prácticos (aún en uso hoy en día) como los siguientes: códigos lineales, códigos cíclicos, códigos de Hamming, códigos de Golay, códigos Reed-Muller, códigos BCH (Bose-Chaudhuri-Hocquenghen), los códigos de Reed-Solomon y Goppa (entre otros).

[1] El término "redundancia" se explica con detalle en la introducción y se desarrolla en capítulos posteriores.

El objetivo de este trabajo de tesis es señalar los códigos correctores de errores como una parte importante de un sistema de comunicaciones, debido a que el ruido y distorsión es inherente con probabilidad de provocar modificaciones en los bits que representan la información transmitida. Aunque, existen otras técnicas de asegurar la transmisión de datos digitales a través de un canal con ruido y/o distorsión. Estos se prefieren debido a que resulta en la mayoría de los casos ser el proceso más económico para desarrollar un sistema confiable. Por ejemplo, en muchos sistemas de comunicación una alternativa para el uso de codificación es simplemente proporcionar suficiente energía a la señal por unidad de información, para asegurar que la información no codificada sea transmitida con la seguridad que se requiere. Sin embargo, en muchos casos los códigos correctores de errores pueden proporcionar la seguridad requerida con menos energía que la operación no codificada y sería económicamente preferible esta solución a pesar de incrementar la complejidad del sistema. Por lo tanto, la teoría de códigos ha sido un fenómeno en crecimiento (por más de medio siglo) y un gran campo de investigación para los ingenieros en comunicaciones, científicos computacionales y matemáticos. El contenido del trabajo de esta tesis se menciona brevemente a continuación.

Estructura de la tesis:

Este trabajo se divide en cinco capítulos y tres apéndices. En cada proposición extraída de algún texto se indica la referencia correspondiente, por ejemplo [2; 222] significa que el resultado que le sigue puede encontrarse en la página 222 de la referencia [2].

En la introducción se presenta una idea general de la problemática de la transmisión de información en un sistema de comunicación y del uso de la redundancia (codificación) que es un método de la teoría de los códigos detectores-correctores de errores.

En el primer capítulo se proporciona una coherente y estructurada clasificación de los tipos de ruido y distorsión, como son: térmico, de disparo, acústico, electromagnético, transitorio, impulsivo y canal de distorsión. Además, se da una introducción de la teoría de la información y la teoría de la codificación ideal. El capítulo concluye con la modelación de algunos de los más importantes tipos de ruido.

El segundo capítulo está dirigido al estudio de los conceptos generales de los códigos lineales. Se presenta la definición de los códigos lineales detectores-correctores de errores como un subespacio de un espacio vectorial de dimensión finita sobre un campo finito, y se definen sus tres principales parámetros (longitud, dimensión y distancia mínima). Se definen también las matrices generadoras y de chequeo de paridad (parity check) de un código lineal, las cuales determinan completamente al mismo. Al recibir una palabra codificada, esto es un elemento del código, es necesario saber cómo decodificarla por lo que se muestra un método general para decodificar un código lineal, conocido como decodificación por síndrome. Enseguida se estudian ciertos tipos de códigos lineales: los llamados códigos de Hamming, con capacidad de detectar y corregir un error (que además tienen la máxima dimensión que puede tener un código lineal de longitud $(q^r - 1) / (q - 1)$, para $r \geq 2$, siendo q una potencia de la característica del campo finito) y también, se analizan los códigos Reed Muller (lineales pero no cíclicos). La mayoría de los códigos lineales que son usados en la práctica son cíclicos.

En el tercer capítulo se definen los códigos cíclicos, una clase especial de códigos lineales muy utilizados en la práctica, como se menciona al final del segundo capítulo. Se muestra una relación entre este tipo de códigos y el álgebra conmutativa y se hace uso de esta relación para calcular una matriz generadora para los mismos. Se analiza la manera de decodificar algunos códigos cíclicos conocidos como los famosos métodos de error trapping o captura del error y el de corrección de error burst o ráfaga.

En el cuarto capítulo se definen los códigos BCH, capaces de corregir una cantidad prefijada de errores y los códigos de Reed-Solomon un tipo de códigos BCH los cuales tienen los parámetros ideales de un código y muy buenos para corregir errores en forma de burst o ráfaga. Se presenta también, los códigos alternantes que son una larga e interesante familia que contiene los bien conocidos códigos Goppa.

En el quinto capítulo se presentan los resultados obtenidos de la simulación por computadora.

Presentación de conclusiones finalmente se presentan las conclusiones obtenidas del trabajo de tesis realizado.

Por último, para tener más claro un panorama de esta investigación de tesis se anexan los respectivos apéndices que sin duda serán de gran ayuda para el lector: Apéndice (A) "Cotas en teoría de códigos", Apéndice (B) "Factorización de polinomios en un campo finito" y Apéndice (C) "Códigos BCH generados por elementos primitivos".

Uso de la computadora:

Existen muchas herramientas disponibles en Internet para ser utilizadas como laboratorios de códigos correctores de errores, de modesto costo y gratuitas. Como el tutorial que ofrece comptut.pdf con una introducción a gap y magma y el laboratorio que ofrece Mathlab y Maple.

http://www.mathworks.com/
http://www.maplesoft.com/

Es recomendable visitar dichas páginas. Pero, debido a sus restricciones y limitaciones, este, fue un motivo más que dió lugar para la elaboración de está tesis. Durante el desarrollo de la presente tesis, así como en su conclusión, se uso la aplicación de las tecnologías computacionales como una herramienta para la optimización de cálculos de procesamiento, ordenamiento y análisis de los códigos correctores de errores. Además, del uso de la computadora para la aplicación de algoritmos programables que facilitan el cálculo algebraico.

Los programas que se elaboraron. Así como, el simulador fueron realizados en DEV C++, VISUAL C++ y JAVA en una plataforma ECLIPSE.

INTRODUCCIÓN

El problema principal

Cuando en un sistema de comunicación deseamos enviar información desde un lugar origen (emisor) hasta un lugar destino (receptor) o si deseamos almacenar información para que pueda ser usada posteriormente, frecuentemente nos encontramos con la tarea de convertir mensajes[2] desde un alfabeto origen para que este pueda ser transmitido a través de un canal. En la actualidad el método de transmisión de la información requiere traducir cada símbolo origen en una palabra binaria [palabra binaria significa una secuencia finita (a_1 a_2 ……. a_n), de uno(s) y cero(s)] para que esta pueda ser transmitida por un sistema o canal de comunicación de los que conocemos en la actualidad par trenzado, coaxial, fibra óptica, o medios no guiados como el aire, agua, vacío o incluso el espacio.

El problema que se tiene en la transmisión de la información es que en nuestro sistema de comunicación y dispositivos de almacenamiento de datos, no son completamente confiables ya que en la práctica existe el ruido y distorsión. Es decir, formas que producen interferencia o debilitamiento de la señal que dañan nuestra información original. Por ejemplo, un ***cero*** podría ser recibido como un ***uno***, y un ***uno*** podría ser recibido como un ***cero***. Debido a esto, procedemos a convertir nuestra información (codificar) a otra forma para que esta no pueda ser afectada durante su transmisión en un canal de comunicación y después decodificarla. Por último, habilitarla para el usuario receptor.

Si tomamos en cuenta que en la vida real el ruido y distorsión se encuentran en todas partes, la codificación implica cambiar el mensaje codificado a un conveniente código corrector de errores, con tan solo agregar información adicional llamada redundancia (una pequeña variación) para ser transmitido por un canal con ruido y/o distorsión para que estos errores puedan ser detectados y corregidos posteriormente. Un ejemplo de mensaje codificado es el utilizado para el código ASCII, el cual convierte cada carácter en un byte de 8 bits.

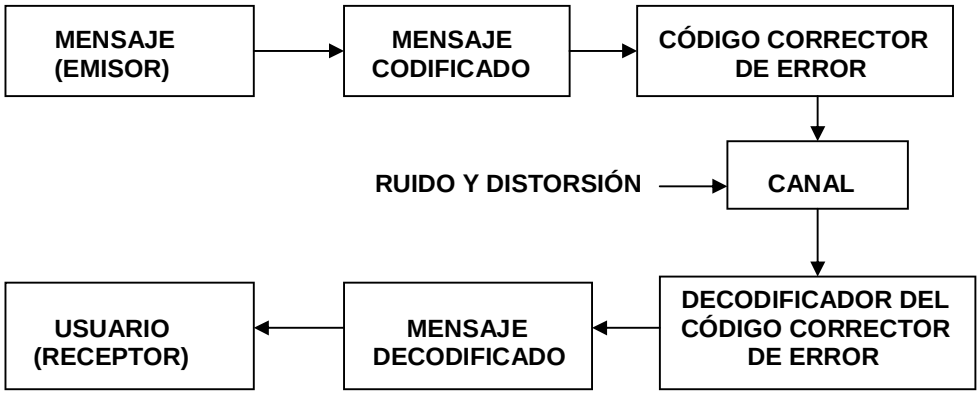


Figura 1 Esquema de canal de codificación.

[2] El término "mensaje" se utiliza preferentemente para referirse a la información (voz, datos, música, video, imágenes, etc.) antes de que haya sido modificada a una forma de onda y pueda estar sujeta a errores y/o distorsión del canal, y el término "señal" para el mensaje después que haya sido modificado a una forma de onda para su transmisión (ver figura 1 y posteriormente figura 1.7).

Otro problema, sumado a lo anterior, por lo cual se usan los códigos correctores de errores, es que algunos sistemas de telecomunicaciones no pueden tolerar el coste de la repetición de un mensaje cuando se corrompe por el camino. En su lugar, el mensaje debe ser corregido de alguna forma en el destino. En estos casos es adecuado el uso de un código corrector de errores.

Ejemplo 1. Supongamos que es enviada una sonda espacial a Marte con el objetivo de detectar los cuatro elementos clásicos griegos (agua, aire, fuego y tierra), los mensajes se codifican de la siguiente manera:

$$\text{agua} \rightarrow 00, \quad \text{aire} \rightarrow 01, \quad \text{fuego} \rightarrow 10 \quad \text{y} \quad \text{tierra} \rightarrow 11.$$

Suponiendo que el mensaje "agua", el cual se codifica como 00, es transmitido desde Marte, el mensaje puede llegar a ser distorsionado y puede ser recibido como 01. El usuario puede no darse cuenta que el mensaje fue corrompido.

Como se aprecia en el esquema del canal de codificación (ver figura 1) se puede codificar el mensaje nuevamente para convertirlo en un código corrector de errores, por tan solo introducir redundancia para que los errores puedan ser detectados y corregidos, como veremos en el siguiente ejemplo.

Ejemplo 2. Aplicamos redundancia al ejemplo 1 y observamos:

$$00 \rightarrow 000, \quad 01 \rightarrow 011, \quad 10 \rightarrow 101 \quad \text{y} \quad 11 \rightarrow 110.$$

Suponiendo nuevamente que el mensaje "agua", el cual se codificó como 00 y con redundancia como 000. Si existe un error en la transmisión el mensaje recibido podría ser 100, 010 o 001. De esta manera, podemos detectar un error ya que 100, 010 o 001 no se encuentran dentro de los mensajes codificados.

La codificación anterior no permite corregir los errores detectados. Por ejemplo, si suponemos que el mensaje recibido es 100 no sabemos si proviene de 000, 101 o 110. Sin embargo, si más redundancia es introducida, no solo podemos detectar errores sino que también podríamos corregirlos. Por ejemplo, si diseñamos el siguiente esquema de canal de codificación.

$$00 \rightarrow 00000, \quad 01 \rightarrow 01111, \quad 10 \rightarrow 10110 \quad \text{y} \quad 11 \rightarrow 11001.$$

Suponiendo nuevamente que el mensaje "agua", el cual se codificó como 00 y con redundancia como 00000. Si existe un error en la transmisión el mensaje recibido podría ser 10000, 01000, 00100, 00010 o 00001. Suponiendo que 10000 es recibido. Podemos decir, que 10000 proviene de 00000 por que hay al menos dos errores entre 10000 y los otros tres mensajes codificados 01111, 10110 y 11001.

Nota: entre más redundancia se aplique mayor será la pérdida en velocidad de transmisión de la información.

Ejemplo 3. Un método general para corregir r errores correctamente, es el de aplicar redundancia $2r + 1$ a un mensaje codificado de longitud k, donde $r \geq 1$. Por ejemplo, para 01 que tiene un valor de $k = 2$ y con $r = 2$. Se tiene que:

$$01 \quad \rightarrow \quad 0101010101$$

La decodificación se realiza por considerar las posiciones 1, 3, 5, 7 y 9 de la cadena recibida y tomar el bit que aparece más frecuentemente en estas posiciones, similarmente para las posiciones 2, 4, 6, 8 y 10 con el que obtenemos el segundo bit decodificado. Por ejemplo, si recibimos la siguiente cadena:

1100010101

Su decodificación es 01. Es claro que, en este caso especial, se pueden corregir hasta dos errores correctamente. En general, podemos corregir hasta r errores correctamente. Por obvias razones, este método es llamado código de repetición. El único problema con este método es que implica bastante perdida de velocidad en la transmisión de la información. En esta tesis, presentaremos métodos más eficientes.

La idea del canal de codificación es obtener códigos correctores de errores con las siguientes características deseables:

1.- Rápida codificación de mensajes.
2.- Fácil transmisión de los mensajes codificados
3.- Rápida y fácil decodificación de los mensajes recibidos.
4.- Máxima transferencia de información por unidad de tiempo.
5.- Máxima capacidad para detectar y corregir errores.

Los códigos correctores de errores han sido empleados para proteger la información en el Internet o en un teléfono celular (transmisión de datos, voz y video), grabación digital (música o video en un CD-ROM o DVD) y también han sido usados para corregir errores que ocurren en información trasmitida a través del espacio, etc.

El objetivo de la investigación es elaborar y validar un simulador de canal con ruido de varios tipos que se asemeja a las transmisiones reales y comparar las probabilidades de error de los códigos BCH con los diferentes códigos correctores de errores.

Existen diferentes tipos de decodificación para los cuales algunos códigos correctores de errores son más apropiados que otros, el problema es conocer el comportamiento de los mismos, bajo diferentes tipos de ruido y elegir el más apropiado de ellos. Es decir, que corrija el mayor número de errores.

Se aplicarán las tecnologías computacionales como una herramienta para la optimización de cálculos de procesamiento, ordenamiento y análisis de los códigos correctores de errores. Así como, el uso de la computadora para la aplicación de algoritmos programables que facilitan el cálculo algebraico, su implementación para minimizar el tiempo de ejecución, la asignación de altos rangos de datos y un rápido procesamiento.

CAPÍTULO 1

RUIDO Y DISTORSIÓN

1 RUIDO Y DISTORSIÓN

El ruido y la distorsión son los principales factores que limitan la calidad de un sistema de comunicación. Por lo tanto, la modelación, reducción y separación de los efectos del ruido y distorsión han sido la base de la teoría y práctica de un sistema de comunicación y del procesamiento de señales. La reducción de ruido y la separación de la distorsión en una señal, son problemas importantes en una comunicación y sus aplicaciones se encuentran en celulares móviles, reconocimiento de voz, procesamiento de textos, procesamiento de imágenes, procesamiento de señales medicas, radares, sonares y en muchas aplicaciones donde la señal no se encuentra libre de ruido y distorsión. En este capítulo, se analizarán las características y la modelación de diferentes tipos de ruido y distorsión que afectan a una señal en un sistema de comunicación.

1.1 RUIDO Y DISTORSIÓN.

El ruido puede ser definido como una señal no deseada que interfiere con la comunicación, el ruido por si solo es una señal que contiene información correspondiente al origen del ruido. Por ejemplo, el ruido que produce la máquina de un automóvil, la cual contiene información correspondiente a su motor. El ruido se puede producir de muchas formas, desde ruido de audio de frecuencias acústicas proveniente de movimiento, vibración o colisión debido a las revoluciones de los motores, movimientos de los vehículos, ventiladores de computadoras, “clicks” de encendido y apagado, gente platicando, viento y lluvia, etc. Hasta ruido de radio-frecuencia electromagnética que interfiere con la transmisión de voz, imagen y datos sobre el espectro de radio-frecuencia.

Señal de distorsión, se refiere a cualquier alteración no deseada de una señal con respecto a su forma original debido a las características no ideales del canal de la transmisión, reverberaciones, eco y perdida de la señal o atenuación. Los tipos de distorsión que pueden afectar una señal son muchos incluyendo la distorsión no lineal, la de frecuencia y la de fase.

La *distorsión no lineal*, es un término utilizado para describir el fenómeno de una relación no lineal entre las "entradas" y "salidas" de las señales, y hay varias subclases de esta distorsión. La *distorsión* al variar la amplitud (*distorsión de amplitud*). *La distorsión armónica* se debe a la adición de información con frecuencias armónicas por parte de un circuito o transductor, casi siempre en forma directamente proporcional a la amplitud de las entradas. La *distorsión de intermodulación* agrega frecuencias que no son por fuerza armónicos de las frecuencias componentes. La *distorsión de revoloteo (flutter)* se debe a desviaciones de la base de tiempo en mecanismos físicos, o en componentes electrónicos, como los osciladores. La *distorsión de frecuencia*, es un fenómeno en el que las salidas contienen frecuencias que no estaban presentes en las entradas. La *distorsión de fase*, se refiere a relaciones de fase que difieren entre las entradas y las salidas.

El ruido y la distorsión pueden causar errores de transmisión y corromper un proceso de comunicación. Por lo tanto, el procesamiento del ruido y distorsión es una parte importante de las modernas telecomunicaciones y sistemas de procesamientos de señales. Dependiendo de su origen el ruido y la distorsión pueden ser clasificados dentro de las siguientes categorías, que indican su naturaleza física:

a) Ruido electromagnético: representa todas las frecuencias y en particular a las radio frecuencias. Todos los aparatos eléctricos, transmisores y receptores de radio y televisión, los generadores electromagnéticos de ruido, etc.
b) Ruido acústico: su origen es a partir del movimiento, vibración o colisión. Es muy similar al ruido que se presenta en un ambiente de nuestras vidas cotidianas y con diferentes grados de magnitud. El ruido acústico es generado por movimiento de autos, aire acondicionado, ventiladores de computadora, tráfico, gente platicando, viento, lluvia, etc.
c) Ruido electrostático: generado por la presencia de un voltaje con o sin flujo de corriente. La iluminación fluorescente es el ejemplo más común de un ruido electrostático.
d) Canal de distorsión, eco y desvanecimiento: debido a características no ideales del canal de comunicación. Canales de radio, tales como las frecuencias de microondas usada por operadores de teléfonos celulares móviles son particularmente sensibles a las características de propagación y al ambiente del canal de comunicación.
e) Procesamiento de ruido: es el ruido que resulta del procesamiento digital/analógico de las señales. Ejemplos son: el ruido de cuantización en la codificación digital de voz o imágenes, o la pérdida de paquetes de datos en un sistema de comunicación de datos digitales.

Dependiendo de las características de frecuencia o tiempo, el ruido puede ser clasificado de la siguiente manera:

1.-Ruido de banda ancha: es el proceso de ruido con amplitud de banda estrecha que llegan a tener 50/60 Hz "Hum o Buzz"[3] de alimentación eléctrica.
2.-Ruido blanco: AWGN (Additive White Gaussian Noise) es un ruido puramente aleatorio que tiene un espectro de potencia plana. El ruido blanco teóricamente contiene todas las frecuencias en igual intensidad.
3.-Ruido blanco de banda limitada: es un ruido con espectro plano y limitado ancho de banda que usualmente cubre el espectro limitado de un dispositivo o señal de interés.
4.-Ruido coloreado: es un ruido no blanco o cualquier ruido de banda ancha el cual el espectro no tiene una forma plana; ejemplos son ruido rosa, ruido marrón y ruido autoregresivo.
5.-Ruido impulsivo: consiste en pulsos de ruido de corta duración de amplitud y duración aleatoria.
6.-Ruido transitorio: consiste de pulsos de ruido de larga duración.

[3] Zumbido constante, normalmente regular o periódico, producido por una red eléctrica simple o armónicamente complejo.

1.1.1 Ruido blanco.

El ruido blanco AWGN es una señal aleatoria que se caracteriza porque sus valores de señal en dos instantes de tiempo diferentes no guardan correlación estadística. Como consecuencia de ello, su densidad espectral de potencia es una constante, i.e, su gráfica es plana. Esto significa que la señal contiene todas las frecuencias y todas ellas tienen la misma potencia. Igual fenómeno ocurre con la luz blanca, lo que motiva la denominación. Un ruido que tiene la misma potencia para todas las frecuencias necesitaría ser de potencia infinita, aunque esto es solo un concepto teórico. Sin embargo, un ruido de banda limitada con densidad espectral de potencia plana que llegará a cubrir e interferir un sistema de comunicación de banda ancha, es llamado un proceso con ruido blanco. En la figura 1.1 podemos observar la forma que tiene el ruido blanco.

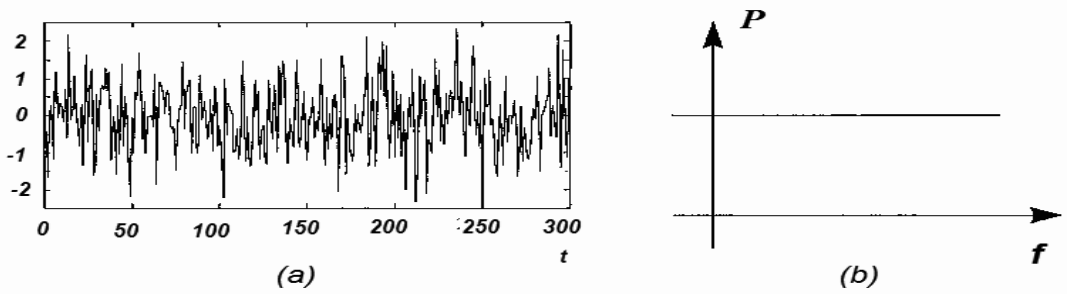


Figura 1.1 (a) Ruido blanco (t = tiempo) y (b) Su potencia espectral. Fuente: "Vaseghi, 2000".

1.1.2 Ruido coloreado.

El término ruido coloreado (correlacionado) se refiere a cualquier ruido de banda ancha con una densidad espectral de potencia que no sea blanco. Por ejemplo, la mayoría del ruido de audiofrecuencia, como el ruido del movimiento de vehículos, ruido de los ventiladores de computadoras, o el ruido de gente platicando, aparatos eléctricos, etc. Todos estos ejemplos poseen un espectro que no es blanco, con un predominio de bajas frecuencias y dependiendo de la forma que tenga la gráfica de la densidad espectral de potencia del ruido, se definen diferentes colores. También, un ruido blanco al pasar por un canal es "coloreado" debido a la forma del espectro del canal. Dos clásicas variedades de ruido coloreado se muestran en las figuras 1.2 y 1.3 ruido rosa y el ruido marrón respectivamente.

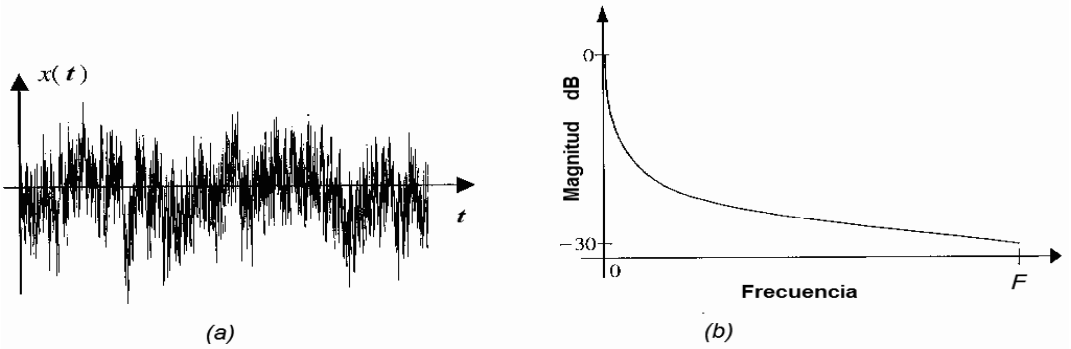


Figura 1.2 (a) Señal ruido rosa y (b) Su magnitud espectral. Fuente: "Vaseghi, 2000".

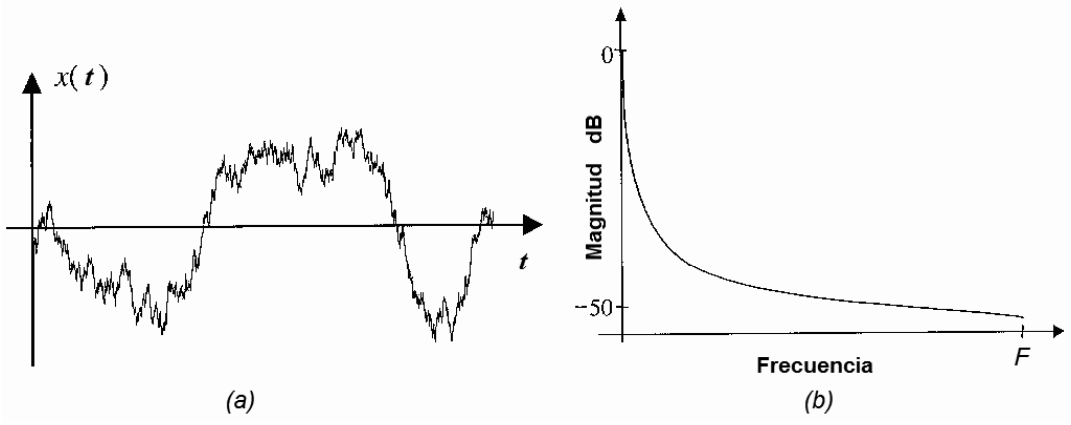


Figura 1.3 (a) Señal ruido marrón y (b) Su magnitud espectral.
Fuente: "Vaseghi, 2000".

1.1.3 Ruido impulsivo.

El ruido impulsivo es aquel ruido cuya intensidad aumenta bruscamente durante un impulso de duración corta, debido a una gran variedad de fuentes, como ruido de los interruptores, adversidades en el medio del canal del sistema de comunicación, surcos o degradación de superficie de grabaciones de audio, "clicks" de los teclados de las computadoras, etc. Por ejemplo, en el contexto de las señales de audio, podemos tener impulsos de 3 milisegundos de duración, la duración de este impulso es breve en comparación con el tiempo que transcurre entre un impulso y otro. Incide fundamentalmente en la transmisión de los datos, se debe básicamente a fuertes inducciones consecuencias de conmutaciones electromagnéticas.

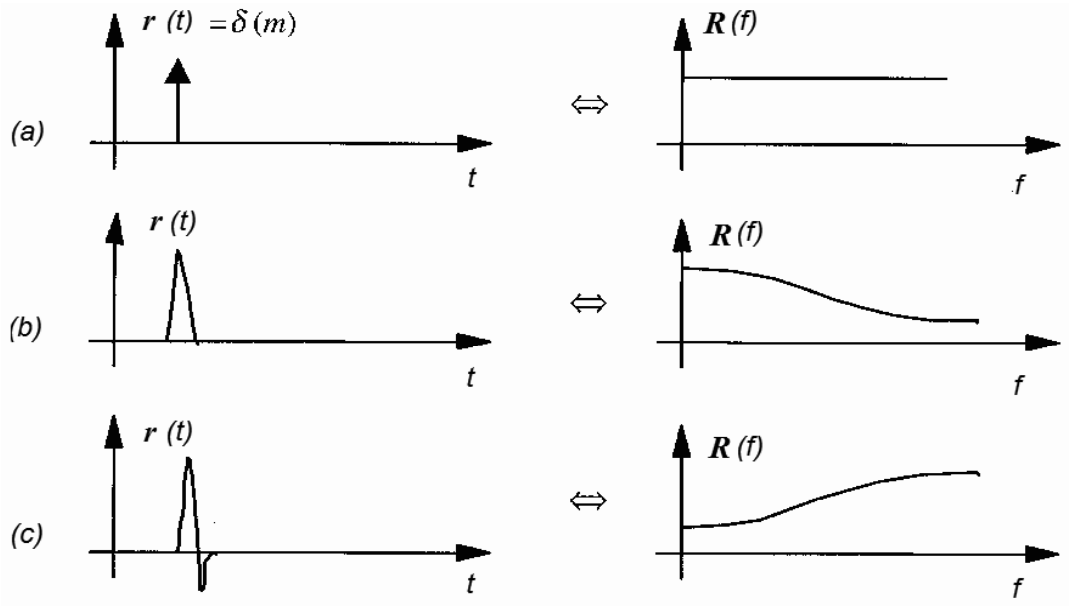


Figura 1.4 (a) Ruido impulsivo ideal y su espectro de frecuencia (t = tiempo y f=frecuencia), (b) y (c) Pulsos de duración corta y su respectivo espectro de frecuencia.
Fuente: "Vaseghi, 2000".

1.1.4 Ruido transitorio.

El ruido transitorio a diferencia de un ruido impulsivo tiene una larga duración y una alta proporción de energía que está contenida en bajas frecuencias y ocurre menos frecuentemente que un ruido impulsivo. Los orígenes del ruido transitorio son variados y pueden ser: electromagnéticos, acústicos o debido a defectos físicos cuando se tiene una información almacenada, como es el caso de los discos compactos. Los ejemplos del ruido transitorio incluyen el ruido por encendido y apagado en líneas telefónicas, ruido transitorio debido al encendido y apagado por la cercanía de aparatos eléctricos, sonidos "click" producidos por la computadora, ralladuras, defectos o daños en discos donde se guarda información almacenada, etc.

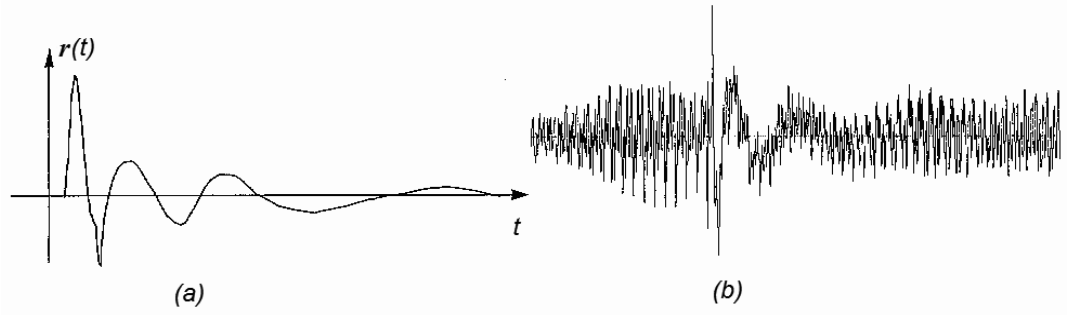


Figura 1.5 (a) Esquema de ruido transitorio de un gramófono y (b) Esquema promedio de un ruido transitorio obtenido de un gramófono. Fuente: "Vaseghi, 2000".

1.1.5 Ruido térmico.

El ruido de Johnson–Nyquist (ruido térmico, ruido de Johnson, o ruido de Nyquist)[4]. El concepto de ruido térmico tiene sus fundamentos en la termodinámica y está asociado con los movimientos aleatorios, dependientes de la temperatura, de las partículas libres, como las moléculas gaseosas en un recipiente o los electrones en un conductor. Aunque el promedio de estos movimientos aleatorios tiende a cero, son las fluctuaciones sobre este promedio las causantes del ruido térmico. Por ejemplo, los movimientos y choques de las moléculas gaseosas en un espacio confinado producen fluctuaciones aleatorias sobre el promedio de la presión. Como la temperatura aumenta, la energía cinética de las moléculas y el ruido térmico aumenta.

Similarmente, un conductor eléctrico contiene un gran número de electrones libres, junto con iones que vibran aleatoriamente alrededor de sus posiciones de equilibrio oponiéndose al movimiento de los electrones. El movimiento libre de los electrones constituye corrientes espontáneas aleatorias, o ruido térmico. Como la temperatura del conductor aumenta los electrones se mueven hacia los estados de mayor energía incrementándose los flujos de corriente aleatorios.

El ruido térmico es aproximadamente blanco, lo que significa que su densidad espectral de potencia es casi plana. Además, la amplitud de la señal sigue una distribución gaussiana.

---

[4] Este tipo de ruido fue medido por primera vez por John B. Johnson en 1928 en los Bell Labs. Comunicó su hallazgo a su compañero Harry Nyquist, quién elaboró la explicación técnica del fenómeno. 5 de marzo de 2009, http://es.wikipedia.org/wiki/Ruido_de_Johnson-Nyquist.

Para una resistencia metálica, el valor de voltaje instantáneo debido a ruido térmico es dado por:

$$V^2 = 4KTRB \tag{1.1}$$

donde k = $1.38x10^{-23}$ joules/kelvin (constante de Boltzmann), T es la temperatura absoluta en grados Kelvin, R es la resistencia en ohms y B es el ancho de banda. De la teoría de circuitos, la máxima potencia debida a un generador de ruido térmico, disipada por una resistencia R es dada por:

$$P_N = i^2 R = \left(\frac{V_{rms}}{2R}\right)^2 R = \frac{V^2}{4R} = KTB(w) \tag{1.2}$$

donde V_{rms} es la raíz del voltaje al cuadrado. La densidad de potencia espectral del ruido térmico esta dado por:

$$P_N(f) = \frac{KT}{2}(w/Hz)) \tag{1.3}$$

Conclusión: la temperatura de ruido especifica la potencia de ruido térmico en una resistencia acoplada.

1.1.6 Ruido de disparo.

El *ruido de disparo* es causado por la variación aleatoria de emisión de electrones que generan una fluctuación en el flujo promedio de partículas en el elemento de salida de un dispositivo electrónico, tal como un diodo, transistor de efecto de campo, transistor bipolar o tubo de vacío. El ruido de disparo fue observado por primera vez en la corriente del cátodo de los amplificadores de tubo de vacío y fue descrito por W. Schottky (1918)[5]. Dos ejemplos más, son el flujo de fotones en un rayo láser y el flujo de fotoelectrones emitidos en fotodiodos. Consideremos una corriente eléctrica como el flujo discreto de cargas eléctricas, si las cargas actúan independientemente, la corriente de fluctuación es dada por:

$$I_{NOISE}(rms) = (2eI_{dc}B)^{1/2} \tag{1.4}$$

donde (rms es la raíz media cuadrática) y la carga del electrón es e =$1.6x10^{-19}$ coulomb y *B* es el ancho de banda.

1.1.7 Ruido electromagnético.

Cada dispositivo eléctrico que genera, consume o transmite energía es una fuente potencial de ruido electromagnético e interferencias para otros sistemas. En general, la alta tensión o el nivel de corriente y la proximidad a circuitos o dispositivos eléctricos, producirán en su mayoría el ruido inducido. Las fuentes más comunes de ruido electromagnético son los transformadores, transmisores de radio y televisión, transmisores de microondas, teléfonos móviles, líneas de corriente alterna, encendido de motores y motores en marcha, generadores, relevadores, lámparas fluorescentes y hasta tormentas eléctricas. El ruido eléctrico de estas fuentes se puede dividir en dos tipos:

Ruido por campo electrostático,
Ruido por campo magnético.

[5] Vaseghi, Saeed V. *"Advanced Digital Signal Processing and Noise Reduction"*. New York: Wiley, Second Edition, 2000.

Los campos electrostáticos se generan por la presencia de tensión, con o sin paso de corriente. Las luces fluorescentes son una de las fuentes más comunes de ruido electrostático. Los campos magnéticos son creados tanto por el flujo de la corriente eléctrica como por la presencia de magnetismo permanente. Los motores y los transformadores, son ejemplos del primer caso y el campo magnético terrestre, es un ejemplo del segundo caso. En ocasiones, la mayoría de las fuentes de ruido producen combinaciones de ambos tipos de ruido, complicando el problema de reducción del ruido.

1.1.8 Ruido de distorsión.

En la transmisión de señales a través de un canal, las señales son producidas y distorsionadas debido a la frecuencia de respuesta y las características de atenuación del canal. Hay dos principales manifestaciones de las distorsiones del canal: distorsión de magnitud y distorsión de fase. Además, en radio comunicaciones tenemos el efecto de múltiples caminos, es cuando la señal transmitida toma diferentes rutas hasta el receptor. Esto da lugar a múltiples versiones de la señal con diferentes retrasos y atenuaciones. Las distorsiones del canal pueden degradar la señal o incluso interrumpir el proceso de comunicación.

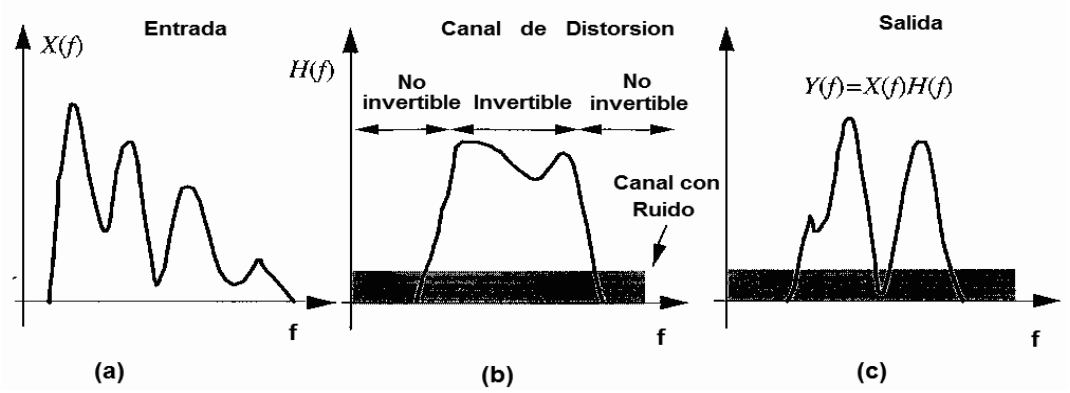


Figura 1.6 (a) Entrada del espectro de la señal, (b) El canal de frecuencia de respuesta y (c) El canal de salida.

Figura 1.6 (b) Ilustra la frecuencia de respuesta de un canal con una región invertible y dos no invertibles. En las regiones no invertibles, la frecuencia de la señal está fuertemente atenuada y cae en ruido. En la región invertible, la señal es distorsionada pero se puede recuperar. Para recuperar la señal distorsionada se utilizan ecualizadores y filtros que se han convertido en componentes esenciales en un moderno sistema de comunicación digital. Otra forma, de uso hoy día, es el de utilizar códigos correctores de errores, con los que se puede recuperar la señal si esta ha sido atenuada o distorsionada y que debido al costo que implicaría utilizar ecualizadores y/o filtros o cualquier otra forma de reducción de ruido, utilizar códigos correctores de errores son preferidos a pesar de incrementar la complejidad de la decodificación de una señal.

1.2 EL CONCEPTO DE CODIFICACIÓN IDEAL.

La contribución de la Teoría de Códigos y la comprensión de sus limitaciones requieren conocimientos de la Teoría de la Información, ya que sus teoremas importantes delimitan el funcionamiento de un sistema de comunicación. Debido a que la Teoría de la Información tiene alta relevancia con la Teoría de Códigos, es posible hoy en día, alcanzar el funcionamiento de los límites en la Teoría de la Información, el éxito ha sido por situar el problema de los códigos más completamente en el contexto de comunicación; para poder asimilar un problema de códigos más cercanamente en la detección de un problema en la señal transmitida[6], en vez de tratar con el problema de códigos principalmente como una combinación discreta.

Esta tesis es dedicada esencialmente al estudio de la codificación de la información y de su correspondiente decodificación, partiendo de simular la forma de onda a la que llamaremos señal, la cual es un método para comunicar información (ver cuadro con punto y línea de la figura 1.7) y con la cual se puede realizar la simulación de transmisión de información con diferentes tipos de ruido y distorsión.

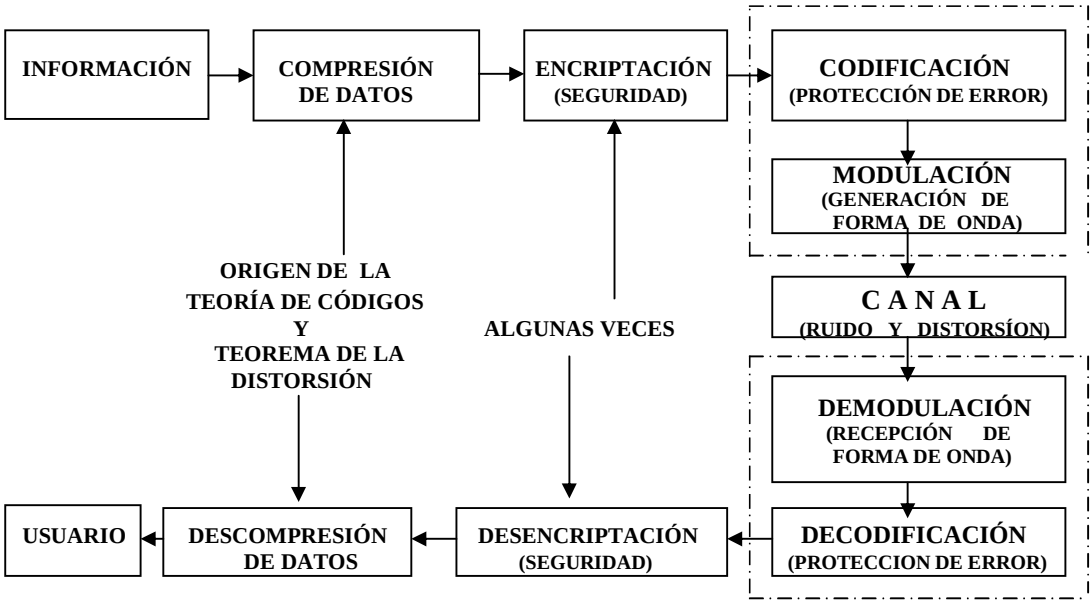


Figura 1.7 Esquema general de un sistema o canal de comunicación.

Algunos de los criterios para evaluar si un sistema de comunicación es ideal o perfecto son el costo, el ancho de banda del canal utilizado, la potencia del transmisor requerida y la demora a través del sistema. A continuación, analizaremos estos conceptos.

En sistemas digitales, el sistema óptimo se define como aquel que reduce al mínimo la probabilidad de error de bit[7] en la salida del sistema, sujeta a limitaciones sobre la energía transmitida y el ancho de banda del canal.

---

[6] Como se demostrará posteriormente diferentes problemas en la señal requerirán diferentes tipos de códigos correctores de errores.
[7] Algunos autores utilizan BER (Bit error rate) como un sinónimo de probabilidad de error de bit. BER se refiere al índice del número de bits recibidos incorrectamente con respecto del total de números de bits enviados.

Esto hace que surja una pregunta ¿es posible inventar un sistema de comunicación digital sin errores aún cuando se introduzca ruido?, Shannon (1948) contestó esta pregunta y la respuesta es si, conforme ciertas suposiciones. Antes de Shannon, Nyquist (1924) y R.V.L. Hartley (1928) realizaron estudios matemáticos sobre la capacidad de comunicación desarrollada por circuitos telegráficos, de hecho Shannon se basó en las ideas de Nyquist y Hartley sobre transmisión de información y de las siguientes definiciones.

Definición 1.2.1 La información enviada desde una fuente digital cuando se transmite el mensaje j-ésimo está dada por

$$I_j = \log_2\left(\frac{1}{P_j}\right) bits \tag{1.5}$$

donde P_j es la probabilidad de transmitir un mensaje j-ésimo.

Definición 1.2.2 La entropía *H* es el promedio de información, tal que, para un mensaje binario con probabilidades de salida p y 1-p, el promedio de información será dado por:

$$H = -p\log_2 p - (1-p)\log_2(1-p) bits. \tag{1.6}$$

donde p = 0.5, que se refiere a la máxima probabilidad de ocurrencia de cada uno de los símbolos. En general para m número de mensajes originales diferentes posibles, la medida de la información promedio de una fuente digital es

$$H = \sum_{j=1}^{m} P_j I_j = \sum_{j=1}^{m} P_j \log_2\left(\frac{1}{P_j}\right) bits \tag{1.7}$$

donde P_j es la probabilidad de enviar el mensaje j-ésimo.

Ejemplo 1.2.3 Determine el contenido de información de un mensaje compuesto de una palabra de 12 dígitos de longitud en la que cada dígito puede adoptar uno de cuatro niveles posibles. Se supone que la probabilidad de enviar cualquiera de los cuatro dígitos es igual y que el nivel en cualquier dígito no depende de los valores adquiridos por dígitos previos.

En una cadena de 12 símbolos (dígitos) donde cada símbolo consiste en uno de los cuatro niveles, existen 4^{12} diferentes combinaciones o palabras que se pueden obtener. Como cada nivel es igualmente probable, todas las diferentes palabras son igualmente probables. De hecho

$$P_j = \frac{1}{4^{12}} = \left(\frac{1}{4}\right)^{12}; \quad \text{De otra manera,} \quad I_j = \log_2\left(\frac{1}{\left(\frac{1}{4}\right)^{12}}\right) = 24\ (bits\ ).$$

En este ejemplo se ve que el contenido de información en cualquiera de los posibles mensajes es igual a cualquier mensaje posible $I_j = 24$ bits. Por consiguiente, la información promedio *H* es 24 bits. Suponiendo que únicamente dos niveles (binarios) fueran permitidos para cada dígito y que todas las palabras fueran igualmente probables, entonces el contenido de información $I_j = 12$ bits y el promedio de información sería *H* = 12 bits.

Definición 1.2.4 La velocidad de la fuente esta dada por

$$R = \frac{H}{T} bits/seg \tag{1.8}$$

donde *H* es la entropía y T es el tiempo requerido para enviar el mensaje.

Con ello, Shannon demostró que en caso de una señal con ruido blanco la capacidad de un canal C (bits/seg) se puede calcular de modo que si la velocidad de transferencia de la información R (bits/seg) fuera menor que C, la probabilidad de errores en los bits tendería a cero, lo anterior es conocido como el teorema de Shannon-Hartley y su ecuación es

$$C = B\log_2\left(1+\frac{S}{N}\right) \tag{1.9}$$

donde B es el ancho de banda del canal en hertz (Hz) y S/N es la relación de potencia señal a ruido (watts/watts, no db).

En la ley de Shannon–Hartley se observan características de un canal de comunicación: potencia de la señal, nivel de ruido y ancho de banda, las cuales son usadas para derivar el canal de capacidad C, mientras la velocidad de transmisión se encuentre por debajo de C, la probabilidad de error en la información enviada puede ser baja o nula, usando la codificación para disminuir el costo de la potencia mientras se consume un mayor ancho de banda. Por ejemplo, suponiendo S/N=15 y B=3kHz resulta una C=12kbits/s. Alternativamente, se puede reducir S/N a 7 e incrementar B hasta 4 kHz para alcanzar la misma capacidad. De otra manera, para una capacidad fija de canal, el ancho de banda B se puede reducir a cambio de un aumento en S/N. Similarmente, para conocer y disminuir un índice de error de decodificación, se puede incrementar la potencia de la señal, o incrementar el ancho de banda del canal agregando códigos de corrección de errores, este último es preferido por reducir el costo del sistema a pesar de aumentar la complejidad del mismo.

Para una codificación ideal, el teorema de Shannon-Hartley (ecuación 1.9), determina la $S/N = E_b/N_o$ requerida (E_b es la energía requerida para transmitir un bit). Esto es, si la velocidad de la fuente es menor que la capacidad del canal, la codificación óptima permitirá que información original se decodifique en el receptor con $P_e \rightarrow 0$ aun cuando exista algo de ruido en el canal. A continuación se determinará la E_b/N_o requerida de modo que $P_e \rightarrow 0$ con la codificación ideal (desconocida). La señal codificada ideal no está limitada en cuanto al ancho de banda, así que, de acuerdo con la ecuación (1-9),

$$C = \lim_{B\to\alpha}\left\{B\log_2\left(1+\frac{S}{N}\right)\right\} = \lim_{B\to\alpha}\left\{B\log_2\left(1+\frac{E_b/T_b}{N_oB}\right)\right\}$$

$$C = \lim_{x\to 0}\left\{\frac{\log_2\left[1+\left(E_b/N_0T_b\right)x\right]}{x}\right\}$$

donde T_b es el tiempo requerido para enviar un bit, y N_0 es la potencia del ruido que ocurre dentro del ancho de banda de la señal.
La densidad espectral de potencia (PSD) es $P_n(f) = N_0/2$ y la potencia del ruido es:

$$N = \int_{-B}^{B} P_n(f)df = \int_{-B}^{B}\left(\frac{N_o}{2}\right)df = N_0 B \tag{1.10}$$

donde B es el ancho de banda de la señal. Para evaluar este límite se utiliza la regla de L'Hôpital.

$$C = \lim_{x\to 0}\left\{\frac{1}{1+(E_b/N_0T_b)x}\left(\frac{E_b}{N_0T_b}\right)\log_2 e\right\} = \frac{E_b}{N_0T_b 1n2} \tag{1.11}$$

Si se envía una señal a una velocidad cercana a la capacidad del canal, $P_e \to 0$ (es decir, el sistema óptimo). Por consiguiente, $1/T_b = C$, o, utilizando

$$\frac{1}{T_b} = \frac{E_b}{N_0T_b 1n\ 2} \qquad \text{o} \qquad E_b/N_0 = 1n\ 2 = -1.59\ \text{dB} \tag{1.12}$$

Para el caso de una codificación ideal, el teorema de Shannon-Hartley (ecuación 1.9), donde el valor mínimo Eb/No es -1.59 dB que permite que una señal pueda ser decodificada sin errores y este valor es llamado el *límite de Shannon*. De este modo, Shannon demostró una meta de rendimiento teórico (límite de Shannon) que hay que tratar de lograr con sistemas de comunicación prácticos. Los sistemas que se aproximan a esta meta por lo general incorporan códigos de corrección de errores. Por ejemplo, para un BPSK[8] sin codificar y BPSK codificado como un código Golay (23, 12) se puede observar en la figura 1.8 una ganancia de codificación de 1.33 dB es realizada para un BER de 10^{-3}. La ganancia de codificación se incrementa si la BER es más pequeña como se puede ver con una ganancia de codificación de 2.15 dB aplicada cuando $P_e=10^{-5}$.

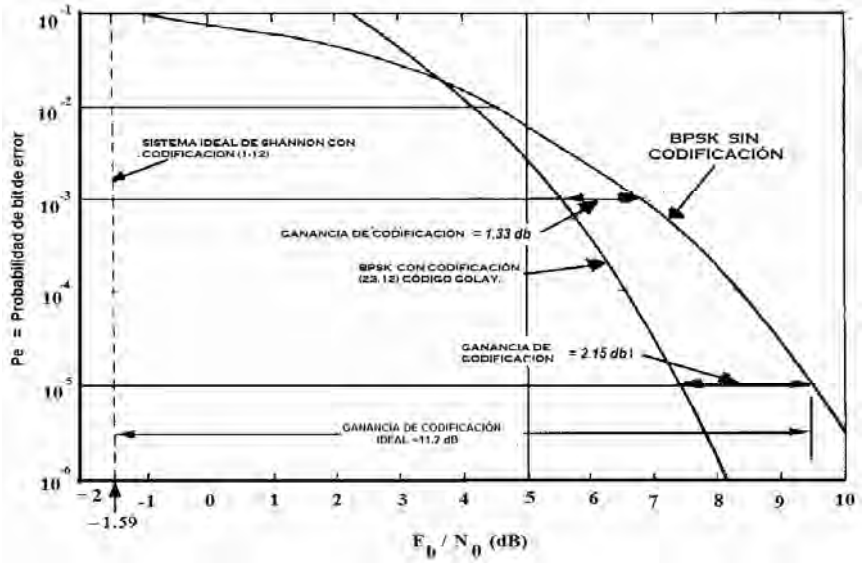


Figura 1.8 Rendimiento de los sistemas digitales, con y sin codificación.

[8] (BPKS, Binary phase-shift Keying) es una forma de modulación de la amplitud en la cual un bit es representado en una forma de onda para su transmisión y es llamado "phase-shift keying" por que su cambio de fase es de 180^0.

1.3 MODELACIÓN DEL RUIDO.

El objetivo de la modelación es caracterizar las estructuras y los parámetros de una señal o de un proceso de ruido. Para modelar un ruido adecuadamente, necesitaremos una estructura para modelar las características temporal y espectral del ruido, una exacta modelación del ruido es la clave para un sistema de comunicación confiable, ya que la clasificación de una señal con ruido y distorsión depende de la disponibilidad de una buena señal y un buen modelo de ruido y distorsión. Además, el uso de estos modelos dentro de un marco de la teoría Bayesiana. En este apartado consideraremos la modelación de algunos de los principales tipos de criterios usados para asegurar la calidad de las señales: modelo de estados finitos [Canal Simétrico Binario (Binary Simetric Channel o BSC) y Modelos De Markov Escondidos (hidden Markov model o HMM)], método Quasianalítico (QA), relación señal a ruido (SNR) y varios aspectos de producción de error.

1.3.1 Modelo de estados finitos.

El modelo de estados finitos cae dentro de dos categorías. La primera categoría, llamada modelo de estados finitos sin memoria, puede ser considerada un caso derivado de la clase general debido a que tales modelos tienen un singular estado. Los modelos sin memoria son usados para modelar la transmisión de errores desde la entrada hacia la salida, asumiendo que la probabilidad de error para cualquier símbolo de entrada no es afectado por lo que suceda para cualquier otro símbolo de entrada, tales modelos son aplicables a sistemas de comunicación con ruido AWGN y en difíciles sistemas de comunicación binaria, se asumiría que los errores de bit son no correlacionados[9].

En la segunda categoría, llamada modelo de estados finitos con memoria, se aplican a situaciones donde la transición[10] de los símbolos de entrada a los símbolos de salida son temporalmente correlacionados, i.e., la probabilidad de transición de los símbolos recibidos es correlacionado con la transición de la anterior y/o los siguientes símbolos. Este será el caso en los sistemas que tienen desvanecimiento, ruido impulsivo y ruido transitorio, etc. La correlación temporal de la variación de la amplitud de las señales debido a los casos anteriormente mencionados, causan errores en la transmisión y estos errores tienden a ocurrir en forma de burst o ráfaga, a estos canales se les llama canales de error burst o canales con memoria.

1.3.1.1 Modelo de estados finitos sin memoria: (Canal Simétrico Binario BSC)

Comenzaremos con algunas definiciones básicas.

Definición 1.3.1.1.1 Sea $A = \{a_1, a_2, ..., a_q\}$ un conjunto de tamaño q, al que llamaremos alfabeto del código y cuyos elementos son llamados símbolos del código.

(i) Una palabra q-aria de longitud n sobre A es una secuencia $w = w_1 w_2 ... w_n$ donde cada $w_i \in A$ para toda i. Igualmente, w es considerada como un vector $(w_1, ..., w_n)$.

(ii) Un código de bloque *q–ario* de longitud n sobre A es un conjunto no vacío C de palabras *q- arias* con la misma longitud n.

---

[9] En probabilidad y estadística, la **correlación** indica la fuerza y la dirección de una relación lineal entre dos variables aleatorias.

[10] Acción de pasar de un estado a otro, la **transición** de un estado a otro es gobernado por alguna regla probabilística la cual es parte del modelo.

(iii) Cada elemento de C es llamado palabra código en C.
(iv) El número de palabras código en C, denotado por $|C|$, es llamado tamaño de C.
(v) El promedio de información del código C de longitud n se define como $(\log_q |C|)/n$.
(vi) Un código de longitud n y tamaño M es llamado código-(n, M).

En la práctica, generalmente el alfabeto del código se toma como un campo finito de F_q de orden q. Además, un código sobre el alfabeto $F_2 = \{0, 1\}$ es llamado código binario, sobre el alfabeto $F_3 = \{0, 1, 2\}$ es llamado código ternario, etc.

Definición 1.3.1.1.2 Un canal de comunicación consiste de un canal con alfabeto finito $A= \{a_1, a_2,...., a_q\}$ y un conjunto de probabilidades del canal $P(a_j \text{ recibida} \mid a_i \text{ enviada})$, tales que

$$\sum_{j=1}^{q} P(a_j \text{recibida} \mid a_i \text{enviada}) = 1$$

para toda i. Donde $P(a_j \text{ recibida} \mid a_i \text{ enviada})$ es la probabilidad condicional de que el símbolo a_j es recibido, dado que el símbolo a_i fue enviado.

Definición 1.3.1.1.3 Un canal de comunicación se dice ser sin memoria si el resultado de cualquier transmisión es independiente del resultado de transmisiones previas; i.e., si $c= c_1c_2 \cdots c_n$ y $x = x_1 x_2 \cdots x_n$ son palabras de longitud n, entonces

$$P(x \text{ recibida} \mid c \text{ enviada}) = \prod_{i=1}^{n} P(x_i \text{ recibida} \mid c_i \text{ enviada}).$$

Definición 1.3.1.1.4 Un canal q-ario simétrico es un canal sin memoria con alfabeto de tamaño q tal que:

(i) Cada símbolo transmitido tiene la misma probabilidad p ($<1/2$) de ser recibido con error.
(ii) Si un símbolo se recibe con error, entonces cada uno de los $q - 1$ posibles errores es igualmente probable.

En particular, el canal simétrico binario (BSC) es un modelo de canal sin memoria el cual tiene un alfabeto $\{0, 1\}$ y con las siguientes probabilidades del canal

$$P(1 \text{ recibido} \mid 0 \text{ enviado}) = P(0 \text{ recibido} \mid 1 \text{ enviado}) = p,$$
$$P(0 \text{ recibido} \mid 0 \text{ enviado}) = P(1 \text{ recibido} \mid 1 \text{ enviado} = 1 - p.$$

Por lo tanto, la probabilidad de bit de error en un BSC es p. Este es llamado probabilidad de cruzamiento del BSC como lo muestra la siguiente figura.

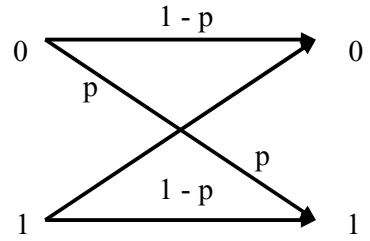


Figura 1.9 Canal Simétrico Binario (BSC).

Ejemplo 1.3.1.1.5 Suponemos que las palabras código de el código {000, 111} son enviados en un BSC con $p = 0.05$. Supóngase que 110 es recibido. Calcularemos la más parecida palabra código que fue enviada tan solo por calcular sus probabilidades de canal.

$$P(110\ recibido|000\ enviado) = P(1\ recibido|0\ enviado)^2 \times P(0\ recibido|0\ enviado)$$
$$= (0.05)^2 \times (0.95) = 0.002375,$$
$$P(110\ recibido|111\ enviado) = P(1\ recibido|1\ enviado)^2 \times P(0\ recibido|1\ enviado)$$
$$= (0.95)^2 \times (0.05) = 0.045125.$$

Como se puede observar, la segunda probabilidad es más grande que la primera y se concluye que 111 es la más parecida palabra código que se envió.

Una extensión del modelo BSC es el caso donde la entrada y la salida pertenecen a un alfabeto M-ario como se muestra en la figura 1.10 (b). Otro ejemplo que se muestra en la figura 1.10 (c) es usado para canales sin memoria en el cual la salida del canal es cuantizado a diferentes números de niveles. En estos modelos la probabilidad de transición es estimada por simular el promedio de datos como:

$$P(e) = \frac{n_{ij}}{N_i} \tag{1.13}$$

donde N_i es el número de ocurrencias de los símbolos de entrada *i-iesimo* y n_{ij} es el número de veces que los símbolos de entrada aparecen como las salidas jth.

La simulación del canal mostrado en la figura 1.10 también implica la generación de un número aleatorio para cada símbolo de entrada la cual determina la transición de entrada-salida de los símbolos del canal.

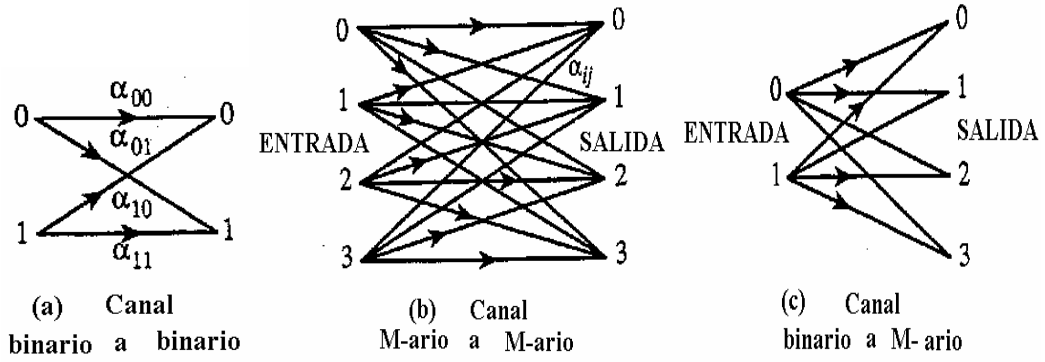


Figura 1.10 Otros ejemplos de modelo de canal discreto sin memoria.

1.3.1.2 Modelo de estados finitos con memoria: (Modelos Markovianos Escondidos HMM).

Para canales con memoria el modelo más comúnmente usado con variación de tiempo, es el modelo de estados finitos[11] de Markov, existen varias razones para su popularidad. Estos modelos son analíticamente tratables, su teoría es bien establecida en la literatura estadística y han sido aplicados para problemas de comunicación. Por ejemplo, para modelar salidas de información (voz y secuencia de imágenes) con origen discreto. Además, es una eficiente técnica computacional para simular o estimar errores provocados por ruido en un sistema de comunicación.

Consideremos un canal con desvanecimiento en el cual la señal recibida durante una gran parte del tiempo se mantiene aceptable pero, la otra parte se mantiene bajo el umbral de aceptación (desvanecimiento). Podemos asumir que el canal cambiará sus posiciones con respecto al tiempo en uno de los siguientes dos estados: (1) Un buen estado en el cual el nivel de la señal es suficientemente fuerte, tal que la probabilidad de error para un sistema de comunicación binaria sea casi cero y no produzca un error o cambio de símbolo. (2) Un mal estado en el cual el nivel de la señal recibida es tan bajo que la probabilidad de error produce un error o cambio de símbolo. El canal cambia de un buen estado a un mal estado y viceversa mientras el tiempo varía. El porcentaje de transición y la longitud de permanecer en cada uno de los dos estados depende de la correlación temporal del proceso de desvanecimiento. Si el tiempo es medido conforme van cambiando los símbolos (bits) en la señal recibida, entonces nosotros podemos construir un modelo de canal discreto (ver figura 1.11). Este es llamado modelo de Gilbert.

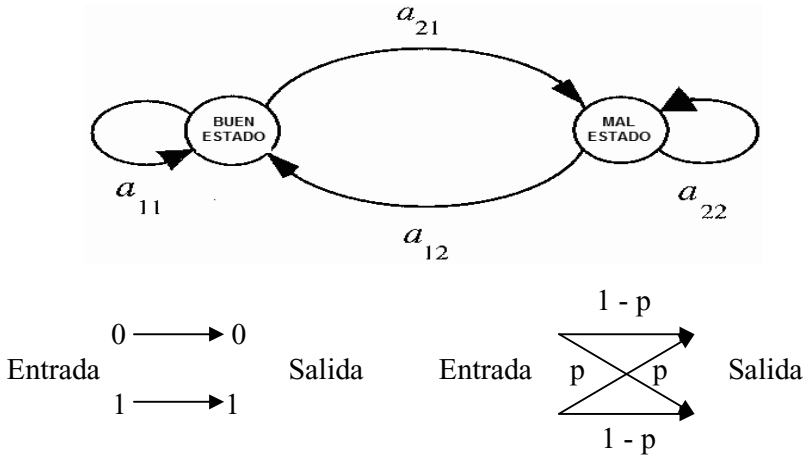


Figura 1.11 Ejemplo de un diagrama de transición para un modelo de Markov de dos estados (Modelo de Gilbert).

Al comienzo de cada símbolo (bit), el canal se encuentra en uno de los dos estados, si el canal se encuentra en un buen estado, el bit transmitido no cambia a un bit con error. De otra manera, si el canal está en estado malo el bit transmitido cambia a ser un bit con error. Para la transmisión de un nuevo bit el canal puede cambiar de estado o permanecer en el estado actual. Está transición entre los estados se lleva a cabo con un conjunto de probabilidades a_{ij} como se muestra en la figura 1.11.

[11] La nomenclatura "estados finitos" implica una visualización del canal (definido entre algunos dos puntos a y b) para encontrarse en una o varias condiciones, o "estados". La transición de un estado a otro es gobernado por alguna regla de probabilidad, la cual es parte de este modelo.

Notar que cuando el canal está en buen estado, no hay transmisión de errores y cuando el canal está en un mal estado, produce una alta concentración de errores. También, la secuencia del error será estadísticamente dependiente del estado del canal ya que el bit actual dependerá del estado en que se encuentre. Podemos notar de estos dos estados, que cada uno en particular corresponde a un modelo BSC diferente. Un modelo general puede ser construido con un largo número de estados, con muchos buenos y malos estados con grados de variación con lo que podemos incrementar más severamente el desvanecimiento, ruido impulsivo, ruido transitorio, etc. Tales aplicaciones guían a un modelo de Markov escondido.

El diagrama de transición para un modelo de dos estados mostrado en la figura 1.11, puede ser interpretado como un canal con dos estados: "buen estado" y "mal estado". La matriz de probabilidad de transición tiene la siguiente forma

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

En el estado bueno los errores ocurren con una pequeña probabilidad b_{12} mientras que en el estado malo la probabilidad de error de bit b_{22} es más grande. Las probabilidades apropiadas para que ningún error ocurra son $b_{11}=1 - b_{12}$ para el buen estado y $b_{21}=1 - b_{22}$ para el mal estado. De hecho, nosotros tenemos

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Debido a que $b_{i1}=1- b_{i2}$, la matriz B es completamente caracterizada por la probabilidad condicional del vector $b_1= (b_{12}, b_{22})$.

Considerando otro ejemplo; para n=3, el HMM representa un modelo de tres estados. El modelo de estado de la matriz de probabilidad de transición tiene la forma

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

En el estado uno, los errores ocurren con probabilidad $b_{12,}$ en el estado dos, la probabilidad de error de bit es b_{22} y en el estado tres, la probabilidad de bit de error es b_{32}. De hecho, nosotros tenemos

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

El diagrama de transición del modelo de tres estados es mostrado en la figura 1.12.

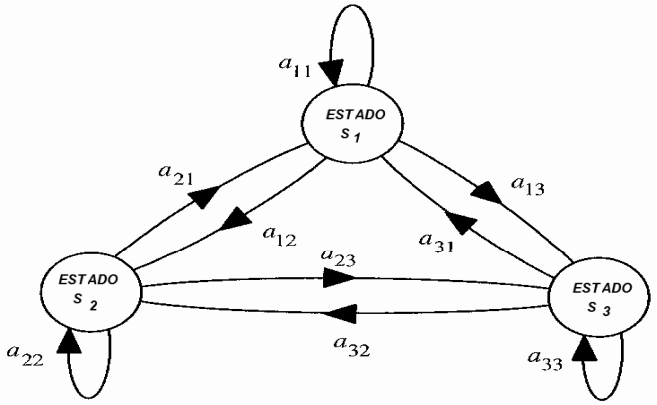


Figura 1.12 Modelo HMM con tres estados binarios.

1.3.1.2.1 Parámetros de un modelo de Markov escondido

Un modelo de Markov tiene los siguientes parámetros:

- Número de estados N**:** Usualmente se refiere al total de números de distintos eventos en el proceso de una señal.
- Estado en el tiempo $t = S_t$.
- Conjunto de probabilidades del estado:

$\pi_i(t)$ = Probabilidad de estar en el estado i en el tiempo t
$\pi_i(t) = P[S_t=i]$, $i=1,2,\ldots, N$

- Conjunto de probabilidades de transición de los estados.
  Probabilidad de ir desde el estado *i* en el tiempo *t* al estado *j* en el tiempo *t + 1*

$$= a_{ij}(t) = P[S_{t+1} = j \mid S_t = i], \qquad i, j = 1, 2,\ldots, N$$

(Generalmente el incremento del tiempo es igual a la duración del símbolo o bit).

- Conjunto de probabilidades de transición de entrada y salida (símbolo de error) para cada estado:

Símbolo de error: $E = \{e_1, e_2,\ldots, e_M\}$; Para el caso binario $E = \{1, 0\}$

$$b_i(e_k) = P[\text{símbolo de error } e_k \text{ ocurriendo} \mid S_t=i]$$

Los parámetros anteriores definen un proceso de Markov con tiempo discreto con un rango de transición igual al promedio de los símbolos utilizados en un sistema de comunicación, la salida del proceso consiste en dos secuencias: la secuencias de estados $[S_t]$ y una secuencia de símbolos de error $[E_t]$, donde t se refiere al tiempo y toma valores del conjunto de los enteros [0, 1, 2,…]. Usualmente, la entrada y la salida del canal junto con la secuencia de error pueden ser observadas, la secuencia de estados no es fácil ser observada físicamente en el canal. Por lo tanto, la secuencia de estados es llamada "escondida" o no visibles, a tales modelos de Markov, se les llama modelos de Markov escondidos o HMM. También, los modelos HMM son usados para evaluar la capacidad de un canal de comunicación y para el diseño de óptimas técnicas de códigos correctores de errores.

En la teoría de la comunicación se asume que el ruido blanco Gaussiano (AWGN) es un proceso estacionario[12]. Aunque para algunos problemas esto es valido y nos guía a una conveniencia matemática y útil solución, en la práctica el ruido frecuentemente es variante con el tiempo, correlacionado y no Gaussiano. Esto es parcialmente verdadero para un ruido de tipo impulsivo y para ruido acústico, los cuales son: no estacionarios y no Gaussianos y por lo tanto no pueden ser modelados como un AWGN usando un BSC. No estacionarios y procesos no Gaussianos de ruido pueden ser modelados por un modelo de Markov (HMM) escondido.

1.3.1.2.2 Procesos de Markov de primer orden.

La propiedad de Markov es definida como:

$$P\left[S_{t+1} \mid S_t, S_{t-1}, \ldots\right] = P\left[S_{t+1} \mid S_t, S_{t-1}, \ldots, S_{t-m-1}\right] \tag{1.14}$$

Para un proceso de Markov con memoria *m*. Un proceso de Markov de primer orden es

$$P\left[S_{t+1} \mid S_t, S_{t-1}, \ldots\right] = P\left[S_{t+1} \mid S_t\right] \tag{1.14a}$$

La propiedad de Markov implica que el futuro comportamiento del sistema depende del estado actual y de los *m* estados previos a este, pero no de los estados pasados *t*-*m*-1 o como el sistema lo obtuvo. En otras palabras, la descripción de la situación actual refleja totalmente toda la información que pueda influir en la evolución futura del proceso.

1.3.1.2.3 Modelo de Fritchman.

El modelo de Fritchman propuesto en 1997, es un modelo del tipo de HMM y es de gran interés por su utilidad para modelar errores en forma de "burst" en canales de radio comunicaciones móviles. El modelo de Fritchman divide a los N estados en k estados buenos y N - k estados malos. Los estados buenos representan la transmisión libre de errores y los estados malos siempre producen errores en la transmisión. La matriz A de estados de transición es representada de la siguiente manera

$$A = \begin{bmatrix} A_{BB} & A_{BM} \\ A_{MB} & A_{MM} \end{bmatrix}$$

Donde las submatrices representan las probabilidades de transición entre buenos y malos estados. Para este modelo es posible modelar los parámetros para errores en forma de burst y usar estas expresiones para estimar los parámetros del modelo (medido o simulado).

Sea $O = \{O_1, O_2, \ldots, O_T\}$ una secuencia de error, $O_k=1$ indica que la transmisión ha sufrido un error de bit $O_k= 0$ indica que la transmisión esta libre de errores. Además, la notación $(0^m|1)$ denota que el evento *m* o más consecutivas transmisiones libres de error son seguidas de un error y $(1^m|0)$ representa el evento de *m* o más consecutivos errores seguidos de un buen periodo. Fritchman mostró que las probabilidades de ocurrencia de estos dos eventos son dados por la siguiente suma:

[12] **Proceso estacionario** es un proceso estocástico cuya distribución de probabilidad en un instante de tiempo fijo o una posición fija es la misma para todos los instantes de tiempo o posiciones.

$$P(0^m \mid 1) = \sum_{i=1}^{k} f_i \lambda_i^{\,m-1} \tag{1.15}$$

y

$$P(1^m \mid 0) = \sum_{i=k+1}^{N} f_i \lambda_i^{\,m-1} \tag{1.16}$$

donde (λ_1, λ_2,…, λ_k) y (λ_{k+1}, λ_{k+2},…, λ_N) son los valores de A_{BB} y A_{MM} respectivamente y la f_i son funciones de a_{ij}.

De las ecuaciones (1.15) y (1.16) se puede obtener la probabilidad de obtener *m* ceros o unos con las siguientes ecuaciones

$$P(0^{m-1} \mid 1) - P(0^m \mid 1) = \sum_{i=1}^{k} f_i \lambda_i^{\,m-1}(1-\lambda_i) \tag{1.15a}$$

y

$$P(1^{m-1} \mid 0) - P(1^m \mid 0) = \sum_{i=k+1}^{N} f_i \lambda_i^{\,m-1}(1-\lambda_i) \tag{1.16a}$$

El modelo de Fritchman puede ser interpretado como un proceso de Markov con una matriz de probabilidad de transición de estado de la siguiente manera:

$$A = \begin{bmatrix} \Lambda_{BB} & A_{BM} \\ A_{MB} & \Lambda_{MM} \end{bmatrix}, \qquad \Lambda_{BB} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_K \end{bmatrix} \quad \text{y} \quad \Lambda_{MM} = \begin{bmatrix} \lambda_{K+1} & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}$$

Notar que en este modelo no hay transición de los conjuntos de N-k dentro de los estados buenos y no hay transición dentro de los estados malos, en el caso general la distribución libre de error $P(0^m|1)$ y la distribución de errores burst $P(1^m|0)$ no especifican dependencia estadística de las secuencias libres de errores y de los errores en burst. En el caso de un modelo con un estado malo (ver figura 1.13), Frichman demostró que:

$$P(0^m \mid 1) = \sum_{k=1}^{N-1} \frac{a_{Nk}(a_{kk})^m}{a_{kk}}, \qquad m \geq 1. \tag{1.17}$$

Con el cual se puede calcular el procedimiento para simular o medir la distribución libre de error de N-1 combinaciones de exponenciales *m* mostradas en la ecuación 1.17.

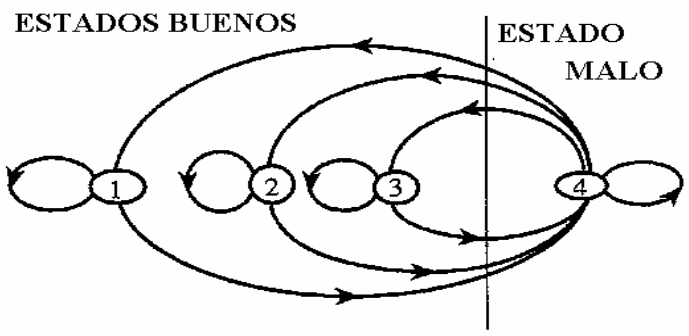


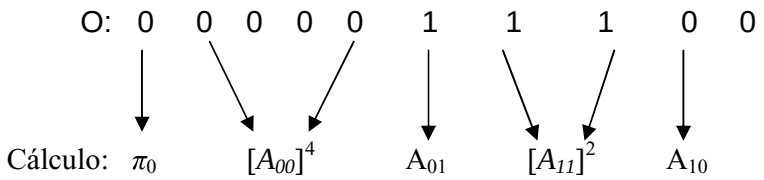
Figura 1.13 Modelo de Frichman con un estado malo y N-1 estados buenos.

1.3.1.2.4 Estimación de los parámetros del modelo de Markov.

Un procedimiento iterativo para estimar los parámetros del modelo de Markov $\Gamma=\{A, B\}$ dado un conjunto de secuencias de error $O=\{O_1,\ldots,O_t,\ldots,O_T\}$, es diseñado para maximizar la estimación donde Γ maximiza $P[O \mid \Gamma]$.

El método que presentaremos a continuación implica cálculos de variables hacia delante y hacia atrás usando la siguiente matriz (este método es para un canal binario con buenos y malos estados):

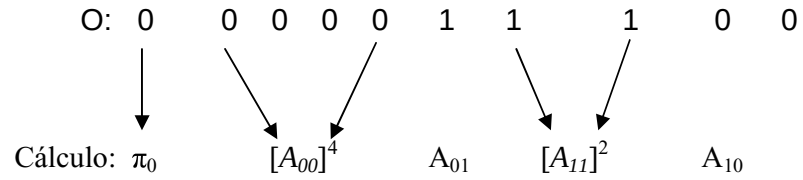
$$A = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}$$



En este método las potencias de las submatrices implicadas en los cálculos pueden ser precalculadas y reutilizadas. Otra modificación se basa en que el modelo general de Markov hay un parecido modelo de Frichman con buenos estados y malos estados y una matriz *A* que tiene la siguiente forma:

$$A = \begin{bmatrix} \Lambda_{00} & A_{01} \\ A_{10} & \Lambda_{11} \end{bmatrix}$$

donde Λ_{00} y Λ_{11} son las matrices diagonales. Con este modelo el canal permanece en el mismo estado durante errores ráfaga y cambia de estado únicamente al final de la ráfaga o burst. Por lo tanto, todas las variables son calculadas en el tiempo en que cambia el símbolo al del error, i.e., en el comienzo de cada ráfaga en lugar de hacerlo cada símbolo. El cálculo ahora toma la siguiente forma.



Notar que el cálculo de largos errores en burst de 1s y 0s implica elevar la potencia de la matriz diagonal en lugar de elevar las matrices de manera arbitraria, la multiplicación de las matrices ocurre únicamente en la transición de un error burst a otro. Por lo tanto, los cálculos son más eficientes y largos errores burst pueden ser procesados sin excesivo almacenamiento y requerimiento computacional.

1.3.2 Simulación Quasianalítica (QA)

Si el ruido es estrictamente aditivo a la señal recibida, y si la función de densidad de probabilidad del ruido en la señal es conocida analíticamente, puede no ser necesario simular el ruido completamente o contar los errores de bit. En lugar de esto puede ser suficiente simular únicamente la señal libre de ruido. Si tenemos el valor de cada señal libre de ruido y la función de densidad de probabilidad del ruido, es conceptualmente posible calcular la probabilidad que el ruido aditivo pudiera ser la causa de que la señal ruidosa cruzará el límite de decisión y causará un error de decisión.

Este método ha sido aplicado para receptores lineales expuestos al ruido AWGN como la principal forma de ruido, desafortunadamente QA es aplicable bajo ciertas condiciones restrictivas y no puede ser usado cuando estás condiciones no son conocidas. Sin embargo, las condiciones restrictivas satisfacen varias clases importantes de señales y es usada en muchos receptores de señal en muchos sistemas de comunicación prácticos.

Condiciones restrictivas

Estas condiciones no son absolutamente restrictivas (existen casos que pueden ser excepción) pero aplican en la gran mayoría de lo métodos QA.

- Únicamente se evalúa la detección símbolo por símbolo.
- El ruido es modelado y sumado a la señal.
- Los ruidos estáticos son conocidos en el receptor de señales.
- Los límites de decisión son conocidos.
- Los símbolos transmitidos correspondientes a cada señal son conocidos.
- Los procedimientos de evaluación son computacionalmente fáciles.
- El receptor es invariante en el tiempo.

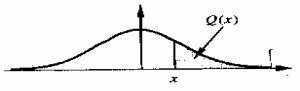
En QA simulación únicamente la señal es simulada no el ruido, y en la simulación de HMM todas las formas del ruido pueden ser simuladas y sumadas a la señal muestra por muestra. Por lo tanto, los requerimientos de una simulación QA se satisfacen con unos cientos de símbolos donde el método de HMM podría necesitar millones.

No existe un mejor camino para la aplicación del problema de estimación de la probabilidad de error de bit (BER), en la práctica existe una compensación entre la mano de obra de ingeniería que puede requerir un conjunto de problemas, la confianza de una técnica y la demanda computacional. Este último dependerá de los tipos de máquinas que sean usadas y de la naturaleza en particular. Consideramos que algunas técnicas pueden ser usadas en combinación. Sea el caso de una transmisión en el espacio donde podría usarse una simulación QA, si eso no nos pudiera dar una confianza, se pensaría en algún modo que esa sería una simulación inadecuada y se podría combinar con una simulación basada en el método HMM.

1.3.2.1 Probabilidad de error de bit (BER).

En la demodulación (ver figura 1-7) las decisiones para que una señal sea retransmitida o no, pueden ser realizadas con alguna probabilidad de error de lo contrario los códigos correctores de errores no serían necesarios. Para detectar un ruido Gaussiano en una transmisión binaria, la probabilidad puede ser expresada usando la función Q(x), que es la probabilidad que la unidad Gaussiana exceda la señal x.

$$Q(x) = P(N > x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-n^2/2} dn.$$



La función Q tiene las siguientes propiedades.

$$Q(x) = 1 - Q(-x) \qquad Q(0) = \frac{1}{2} \qquad Q(-\alpha) = 1 \qquad Q(\alpha) = 0$$

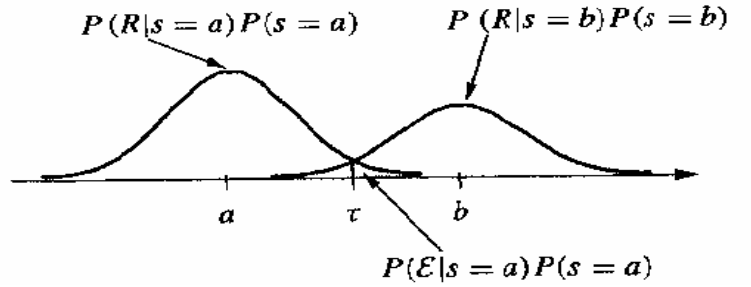


Figura 1.14 Distribución cuando dos señales son enviadas con ruido Gaussiano.

Supóngase que hay dos puntos *a* y *b* a lo largo de un eje, y que R = *s* + N, donde R es la señal, *s* es uno de los dos puntos y N es el ruido. Las distribuciónes P(R|s = a) P(s = a) y P(R|s = b) P(s = b) se muestran en la figura (1.14). Cuando la señal *a* es enviada, el error ocurre cuando R > τ. Donde ε es el evento de error. Un caso especial es cuando P(s=a) = P(s=b) = 1/2. El umbral de decisión es cuando τ = (*a*+*b*)/2. La distancia entre las dos señales es dada por d=|b - a| y la probabilidad de error puede ser escrita como:

$$P(\varepsilon) = Q\left(\frac{d}{2\sigma}\right). \tag{1.18}$$

Para un caso particular de señal BPSK, con $a = -\sqrt{E_b}, b = \sqrt{E_b}$ y $d = 2\sqrt{E_b}$, donde E_b es la energía requerida por bit. La varianza para un canal es expresado como $\sigma^2 = (N_0/2)$ y la probabilidad P(ε) es denotada como P_b "probabilidad de error de bit".

$$P_b = Q(E_b/\sigma) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right). \tag{1.19}$$

donde la cantidad E_b/N_0 es frecuentemente llamada bit de relación señal a ruido (SNR). N_0 representa la densidad espectral de la potencia de ruido. La transformación en decibeles puede darse con la siguiente ecuación.

$$E_b/N_0\, dB = 10\log_{10} E_b/N_0\,. \tag{1.20}$$

Para el caso de una simulación QA, con *k* bits de entrada y n bits de salidas, donde n > k y sea R = k/n el rango del código. Una transmisión con E_b Joules/bit, tenemos que la energía por bit codificado es:

$$E_C = R\, E_b \tag{1.21}$$

Frecuentemente se desea simular el método QA para un SNR en particular. Con $\gamma=E_b/N_o$ la cual denota la señal a ruido deseada simulada. Frecuentemente esto es expresado en decibeles dB, así que nosotros tenemos:

$$\gamma = 10^{(SNR,dB)/10} \tag{1.22}$$

Anteriormente comentamos que $\sigma^2 = N_0/2$ y conociendo γ, tenemos

$$\gamma = E_b / 2\sigma^2, \tag{1.23}$$

donde $$\sigma^2= E_b/2\,\gamma. \tag{1.24}$$

como $E_C = R\, E_b$, tenemos $$\sigma^2 = E_c/2R\,\gamma \tag{1.25}$$

Es común en la simulación, que el valor de la amplitud de la señal simulada sea $E_c=1$.

Con estos valores podemos tener la función de densidad que junto con la ecuación (1.19) caso particular de señal BPSK se puede calcular la probabilidad de error requerida para una simulación QA[13]. A continuación se presenta la gráfica para la probabilidad de error para R=1/3, R=1/2 y R=1.

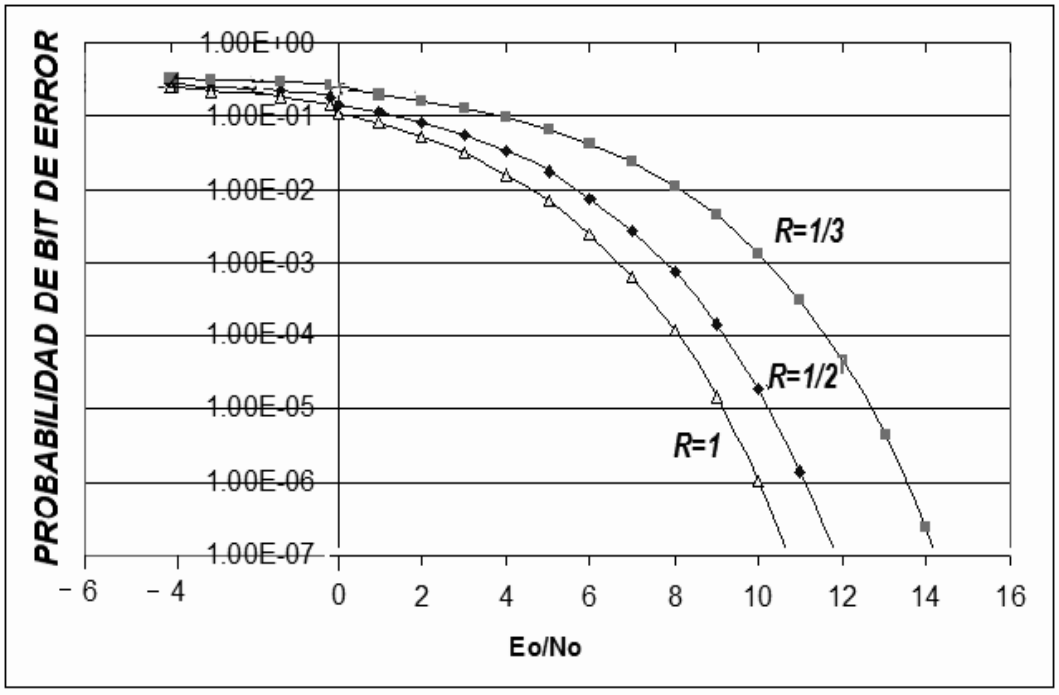


Figura 1.15 Probabilidad de bit de error para bits codificados.

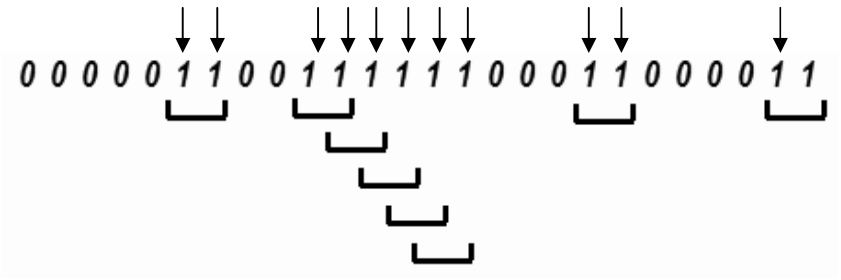
[13] Para otros tipos de señales digitales (OOK, PSK, QPSK Y MPSK), consultar [10; 378].

1.3.3 Otras formas de producir errores en una transmisión.

Ejemplo 1.3.3.1 Si se tiene una transmisión de 26 bits.

$$00000110011111100011000011 \rightarrow 0^5\ 1^2\ 0^2\ \ 1^6\ 0^3\ 1^2\ \ 0^4\ \ 1^2.$$

Calcular la probabilidad de tener dos bits erróneos consecutivos $P_e(11)$:
Contando el último par, hay 25 pares



$$\rightarrow \frac{8}{25}$$

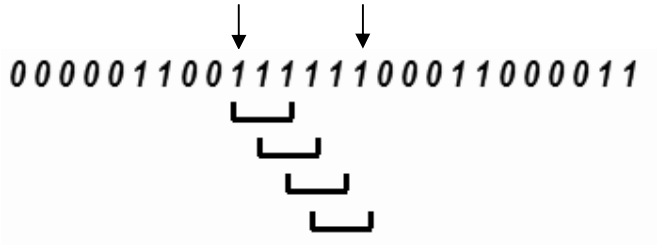
Para Pr (1 | 1) Dado que hay un bit erróneo, probabilidad de que el siguiente también lo sea es:

$$P_{(e)} = \frac{N^0\ Casos}{N^0\ Casos \quad Posibles} = \frac{8}{12} \approx \frac{P_e(11)}{P_e(1)} = \frac{\frac{8}{25}}{\frac{12}{26}}$$

Ejemplo 1.3.3.2 Si se tiene una transmisión de 26 bits.

$$00000110011111100011000011 \rightarrow 0^5\ 1^2\ 0^2\ \ 1^6\ 0^3\ 1^2\ \ 0^4\ \ 1^2.$$

Calcular la probabilidad de tener tres bits erróneos consecutivos $P_e(111)$:



$$\rightarrow \frac{4}{24} = \frac{2}{12}$$

Para Pr (111 | 11) Dado que hay dos bits erróneos, probabilidad de que el siguiente también lo sea es:

$$P_{(e)} = \frac{N^0\ Casos}{N^0\ Casos \quad Posibles} = \frac{P_e(111)}{P_e(11)} = \frac{\frac{4}{24}}{\frac{8}{25}}$$

CAPÍTULO 2

CÓDIGOS LINEALES

2. CÓDIGOS LINEALES

En este capítulo, trataremos con la teoría de códigos de bloque para detectar y corregir errores. Un código lineal de longitud *n* sobre un campo finito F_q es simplemente un subespacio del espacio vectorial F_q^n. Como los códigos lineales son espacios vectoriales, sus estructuras algebraicas frecuentemente los hacen más fáciles de describir y usar que los códigos no lineales. En la mayor parte de esta tesis, nuestra atención se enfoca en los códigos lineales sobre campos finitos. La noción de código de corrección de error fue descubierto por R.W. Hamming en 1950 [27]. Además, el concepto de código lineal fue enunciado primero, en 1956, por D. Slepian [17].

2.1 DETECCIÓN, CORRECCIÓN Y DECODIFICACIÓN DE ERROR.

En un canal de comunicación con codificación, donde únicamente las palabras son transmitidas. Suponiendo que una palabra **w** es recibida. Si **w** es una palabra código valida, se concluye que no hay error en la transmisión. De otra manera, se sabe que algunos errores han ocurrido. En este caso, se necesita una regla para encontrar la más parecida palabra código que fue enviada. Tal regla es conocida como la regla de decodificación. En este apartado, discutiremos los ejemplos de decodificación por máxima similitud y decodificación por distancias mínimas.

2.1.1 Decodificación por máxima similitud.

Supongamos que se envían las palabras código de un código C por medio de un canal de comunicación. Si la palabra **x** es recibida, podemos calcular las probabilidades del canal tales que

$$P(x \text{ recibida} \mid c \text{ enviada})$$

Para todas la palabras código $c \in C$. La *regla de decodificación por máxima similitud* concluye que c_x es la más similar palabra código transmitida si c_x tiene la mayor probabilidad de haber sido enviada.

$$P(x \text{ recibida} \mid c_x \text{ enviada}) = \max_{c \in C} P(x \text{ recibida} \mid c \text{ enviada})$$

Existen dos tipos de decodificación por máxima similitud.

(i) *La completa decodificación por máxima similitud*. Si una palabra *x* es recibida, decodificamos a la más parecida palabra código transmitida. Si existe más de una palabra, elegimos una arbitrariamente.

(ii) *La decodificación incompleta por máxima similitud*. Si una palabra *x* es recibida, decodificamos a la más parecida palabra código transmitida. Si existe más de una palabra, requerimos una retransmisión.

2.1.2 Distancia y peso de Hamming.

Supóngase que un código C es enviado en un BSC con probabilidad p < 0.5 (en la práctica, p es más pequeño que 0.5). Si una palabra **x** es recibida, entonces para cualquier palabra código **c** Є C el canal de probabilidad es dado por:

$$P(x \text{ recibida} \mid c \text{ enviada}) = p^e (1-p)^{n-e},$$

donde **n** es la longitud de **x** y **e** es el número de lugares en los cuales **x** y **c** difieren.

Ejemplo 2.1.2.1 Supóngase que se desea transmitir la cadena **c** = 100110 Є F_2^6, y supóngase también que la probabilidad de transmisión incorrecta es p = 0.07, así la probabilidad de transmitir ***c*** sin errores es $(0.93)^6 \approx 0.65$. La probabilidad de que la palabra enviada se reciba con error en la primera posición es, por eventos independientes, $(0.07)\ (0.93)^5 \approx 0.049$. Con ***e*** = 100000 se puede escribir **x** = **c**+**e**, entonces ***x*** es la palabra recibida y ***e***, se denomina el **patrón de error.**

Para calcular la probabilidad de que la palabra recibida difiera de la enviada en exactamente dos posiciones, se suman las probabilidades de cada patrón de error formado por la probabilidad de que la palabra enviada se reciba con error en la primera posición y la probabilidad de que la palabra enviada se reciba con error formado por dos unos y cuatro ceros. Es decir, $(0.07)\ (0.93)^5 \approx 0.049 + (0.07)^2(0.93)^4 \approx 0.004$. Igualmente, la probabilidad de que ocurran dos errores durante la transmisión es:

$$\binom{6}{2} (0.07)^2 (0.93)^4 \approx 0.055$$

Teorema 2.1.2.2 Sea c Є F_2^n. Para transmitir *c* por un canal binario simétrico con probabilidad de error p,
a) La probabilidad de recibir $x = c + e$, donde *e* es un patrón de error particular, formado por k unos y (n - k) ceros, es $p^k (1 - p)^{n-k}$.
b) La probabilidad de que se comentan k errores en la transmisión es[14]: $\binom{n}{k} p^k (1-p)^{n-k}$.

Ejemplo 2.1.2.3 La probabilidad de enviar la palabra codificada 110101101 y cometer a lo sumo un error en la transmisión es:

$$\underbrace{(1-p)^9}_{\substack{\text{Los nueve bits se transmiten en forma correcta.}}} + \underbrace{\binom{9}{1} p\ (1-p)^8}_{\substack{\text{Un bit ha cambiado en la transmisión y se detecta un error.}}}.$$

Para p = 0.001 obtenemos: $(0.999)^9 + \binom{9}{1}(0.001)(0.999)^8 = 0.999964$.

Como p < 0.5, tenemos que 1 - p > p, esta probabilidad es más larga para valores largos de n – ***e***. i.e., para pequeños valores de ***e***. Por lo tanto, esta probabilidad es maximizada por elegir una palabra código ***c*** para el cual ***e*** es tan pequeña como sea posible. Este valor ***e*** introduce la noción fundamental de la distancia de Hamming.

---

[14] El coeficiente binomial es $\binom{n}{i} = \frac{n!}{i!(n-i)!}$.

Definición 2.1.2.4 Sean ***x*** y ***y*** palabras de longitud *n* sobre un alfabeto A. La distancia de Hamming de ***x*** a ***y***, denotada por d(***x***, ***y***), es definida como el número de lugares en las que ***x*** y ***y*** difieren. Si $\mathbf{x} = x_1 \cdots x_n$ y $\mathbf{y} = y_1 \cdots y_n$, entonces:

$$d(\mathbf{x}, \mathbf{y}) = d(x_1, y_1) + \cdots + d(x_n, y_n), \tag{2.1}$$

donde x_i y y_i son consideradas palabras de longitud 1, y

$$d(x_i, y_i) = \begin{cases} 1 & \text{si } x_i \neq y_i \\ 0 & \text{si } x_i = y_i. \end{cases}$$

Ejemplo 2.1.2.5 Sea A = {0, 1} y sean **x**=01010, **y**=01101 y **z**=11101. Entonces tenemos:

$$d(\mathbf{x}, \mathbf{y}) = 3,$$
$$d(\mathbf{y}, \mathbf{z}) = 1,$$
$$d(\mathbf{z}, \mathbf{x}) = 4.$$

Proposición 2.1.2.6 Sean **x**, **y** y **z** palabras de longitud n sobre A. Entonces, la distancia de Hamming satisface las siguientes propiedades:

(i) $0 \leq d(x, y) \leq n$,
(ii) $d(\mathbf{x}, \mathbf{y}) = 0$, si y solo si $\mathbf{x} = \mathbf{y}$,
(iii) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$,
(iv) *(desigualdad triangular)* $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

Demostración. (i), (ii) y (iii) son obtenidos de la definición de distancia de Hamming. Para (iv) cuando n = 1, tenemos:

Si $\mathbf{x} = \mathbf{z}$, entonces (iv) se cumple puesto que $d(\mathbf{x}, \mathbf{z}) = 0$.
Si $\mathbf{x} \neq \mathbf{z}$, entonces también $\mathbf{y} \neq \mathbf{x}$ o $\mathbf{y} \neq \mathbf{z}$, por lo que se cumple nuevamente (iv).

Definición 2.1.2.7 Para un código *C* que contiene al menos dos palabras, la *distancia mínima* de C, denotada como *d(C),* es

$$d(C) = mín\ \{d(\mathbf{x}, \mathbf{y}) : \mathbf{x},\ \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}.$$

Definición 2.1.2.8 Sea **x** una palabra en F_q^n. El peso de (Hamming) de **x**, denotado por *wt*(x), es el número de coordenadas o símbolos diferentes de cero en **x**; es decir,

$$wt(\mathbf{x}) = d(\mathbf{x}, 0),$$

donde **0** es la palabra cero.

Observación 2.1.2.9 Para cada elemento x de F_q, el peso de Hamming es definido como:

$$wt(x) = d(x, 0) = \begin{cases} 1 & \text{si } x_i \neq 0; \\ 0 & \text{si } x_i = 0. \end{cases}$$

Por lo tanto, escribimos $\mathbf{x} \in F_q^n$ como $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, el peso de Hamming de **x** puede ser definido como:

$$wt(\mathbf{x}) = wt(x_1) + wt(x_2) + \cdots + wt(x_n). \tag{2.2}$$

Lema 2.1.2.10 Si $x, y \in F_q^n$, entonces $d(x, y) = wt(\mathbf{x} - \mathbf{y})$.

Demostración. Para $\mathbf{x}, \mathbf{y} \in F_q$, $d(\mathbf{x}, \mathbf{y}) = 0$ si y solo si $\boldsymbol{x} = \boldsymbol{y}$, lo cual es cierto si y solo si $\mathbf{x} - \mathbf{y} = 0$ o, equivalentemente, $wt(\mathbf{x} - \mathbf{y}) = 0$. El lema 2.1.2.10 se sigue de (2.1) y (2.2).

Corolario 2.1.2.11 Sea **q** par (entonces, a = -a para toda $a \in F_q$). *Si* $x, y \in F_q^n$, entonces $d(x, y) = wt(x + y)$. Para $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ y $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ en F_q^n, sea

$$\mathbf{x} * \mathbf{y} = (x_1 y_1, x_2 y_2, \ldots, x_n y_n).$$

Lema 2.1.2.12 Si $\mathbf{x}, \mathbf{y} \in F_2^n$, entonces

$$wt(\mathbf{x} + \mathbf{y}) = wt(\mathbf{x}) + wt(\mathbf{y}) - 2wt(\mathbf{x} * \mathbf{y}), \tag{2.3}$$

Demostración. Por (2.2), es suficiente con probar que (2.3) se cumple para $\mathbf{x}, \mathbf{y} \in F_2$. Se puede verificar con la siguiente tabla 2.1.

x	**y**	**x** * **y**	wt (**x**) + wt (**y**) – 2wt (**x** * **y**)	wt (**x** + **y**)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	0

TABLA 2.1.

El lema 2.1.2.12 implica que $wt(\mathbf{x}) + wt(y) \geq wt(x+y)$ para $\mathbf{x}, \mathbf{y} \in F_2^n$, lo cual se cumple también sobre cualquier alfabeto F_q.

Definición 2.1.2.13 Sea C un código (no necesariamente lineal). El mínimo peso (Hamming) de C, denotado por wt(C), es el más pequeño de los pesos de palabra código C diferente de cero.

Teorema 2.1.2.14 Sea C un código lineal sobre F_q. Entonces d(C)= wt(C).

Demostración. Para cualquiera de las palabras **x**, **y** tenemos d(**x**, **y**) = wt(**x** - **y**). Por definición 2.1.2.7, existe **x´**, **y´** ∈ C tal que d(**x´**, **y´**) = d(C), así d(C) = d(**x´**,**y´**) = wt(**x´**- **y´**) ≥ wt(C), entonces **x´**- **y´** ∈ C. Contrariamente, existe un **z** ∈ C tal que wt(C) = wt(**z**), entonces wt(C) = wt(z) = d(**z**, 0) ≥ d(C).

Ejemplo 2.1.2.15 Sea $C = \{0000, 0101, 1010, 1111\}$ un código binario. Observamos que:

wt (0101) = 2,
wt (1010) = 2,
wt (1111) = 4.

Por lo tanto, d(C) = 2.

2.1.3 Decodificación por distancias mínimas /vecino más cercano.

Supongamos que las palabras código de un código C se envían por medio de un canal de comunicación. Si la palabra **x** es recibida, la *regla de decodificación por distancias mínimas o regla de decodificación de el vecino más cercano* decodificará **x** como $\boldsymbol{c_x}$ si $\boldsymbol{d(x, c_x)}$ es la mínima entre todas las palabras código en *C*, i.e.,

$$d(x, c_x) = \min_{c \in C} d(x, c). \tag{2.4}$$

De la misma manera, que en el caso de la decodificación por máxima similitud, podemos distinguir entre completa e incompleta decodificación para el caso de esta regla. Para una palabra recibida **x**, si dos o más palabras código $\boldsymbol{c_x}$ son mínimas e iguales (i.e, satisfacen 2.4), entonces la decodificación completa simplemente elige a una de ellas de forma arbitraria, mientras que en la decodificación incompleta se solicita una retransmisión.

Ejemplo 2.1.3.1 Suponemos las palabras código de un código binario

C = {01101, 00011, 10110, 11000}

son enviadas sobre un BSC y las siguientes palabras son recibidas x_1=10011 y x_2=11011. Entonces tenemos que:

Para x_1=10011.

d (10011,01101)=4,
d (10011,00011)=1,
d (10011,10110)=2,
d (10011,11000)=3.

Por usar el método de decodificación del vecino más cercano, decodificamos 10011 a 00011.

Para x_2=11011.

d (11011,01101)=3,
d (11011,00011)=2,
d (11011,10110)=3,
d (11011,11000)=2.

Por usar el método de decodificación del vecino más cercano, no podemos decodificar 11011 a 00011 o 11000.
Por lo tanto, se solicita una retransmisión.

Ejemplo 2.1.3.2 Para un código ternario C = {00122, 12201, 20110, 22000} usar el método de decodificación del vecino más cercano para decodificar la siguiente palabra código recibida 20120.

Para x = 20120.

d (20120, 00122)=2,
d (20120, 12201)=5,
d (20120, 20110)=1,
d (20120, 22000)=3.

Por usar el método de decodificación del vecino más cercano, decodificamos 20120 a 20110.

2.1.4 Relación de mínima distancia para detectar y corregir errores.

Dependiendo de los requerimientos de la aplicación, un decodificador puede ser diseñado para detectar errores únicamente, corregir errores únicamente o una combinación de detectar y corregir errores. Los parámetros de la distancia mínima nos proporcionan una métrica que se usa para predecir la capacidad de un código para detectar y corregir errores.

Definición 2.1.4.1 Un código de longitud *n*, tamaño *M* y distancia *d* se escribe como *C – (n, M, d)*. Los números *n*, *M* y *d* son llamados parámetros del código.

Ejemplo 2.1.4.2 Sea C = {00000, 00111, 11111} un código binario, con d(C) = 2, entonces:

d (00000, 00111) = 3,
d (00000, 11111) = 5,
d (00111, 11111) = 2.

Por lo tanto, C es un código binario (5, 3, 2).

Definición 2.1.4.3 Sea *u* un entero positivo. Un código *C* es *detector de u errores* si, siempre que una palabra código incurre en al menos un error pero a lo más *u* errores, la palabra resultante no es una palabra código. Un código C se dice *detector de exactamente u errores* si es detector de *u* errores, pero no detector de (u + 1) errores.

Ejemplo 2.1.4.4 Retomando el ejemplo 2.1.4.2, C es detector de un error ya que cambiando una posición de cualquier palabra código, no será otra palabra código.

00000 → 00111 necesita cambiar tres posiciones,
00000 → 11111 necesita cambiar cinco posiciones,
00111 → 11111 necesita cambiar dos posiciones.

De hecho, C es exactamente detector de 1 error, debido a que si cambiamos las dos primeras posiciones de 00111 el resultado sería la palabra código 11111 y por lo tanto, C no es un código detector de 2 errores.

Ejemplo 2.1.4.5 Sea C el código ternario *C* = {000000, 000111, 111222} un código (6, 3, 3), entonces C es detector de dos errores ya que cambiando dos posiciones de cualquier palabra código, no será otra palabra código.

000000 → 000111 necesita cambiar tres posiciones,
000000 → 111222 necesita cambiar seis posiciones,
000111 → 111222 necesita cambiar seis posiciones.

De hecho, C es exactamente detector de 2 errores, pues si cambiamos por 1 las últimas tres coordenadas de 000000 el resultado sería la palabra código 000111 y por lo tanto, *C* no es un código detector de 3 errores.

Teorema 2.1.4.6 Un código C es detector de *u* errores si y solo si d(C) ≥ *u* +1; i.e., un código con distancia *d* es exactamente un código detector de (*d* – 1) errores.

Demostración. Supongamos que d (C) ≥ *u* + 1. Si **c** ∈ C y **x** son tales como 1≤ d(**c**, **x**) ≤ *u* < d(C), entonces **x** ∉ C; Por lo tanto, C detecta *u* errores. De otra manera, si d(C) < *u* + 1, i.e., d (C) ≤ *u*, entonces existe $\mathbf{c}_1$, $\mathbf{c}_2$ ∈ C tales que 1≤ d ($\mathbf{c}_1$, $\mathbf{c}_2$) = d (C) ≤ *u*. Por lo que es posible que comencemos con $\mathbf{c}_1$ y se incurra en d(C) errores (donde 1≤ d(C) ≤ *u*) de tal modo que la palabra resultante sea $\mathbf{c}_2$, otra palabra código en C. Por lo tanto, C no es un código detector de *u* errores.

Definición 2.1.4.7 Sea *v* un entero positivo. Un código *C* se dice corrector de *v* errores si su decodificación por distancia mínima permite corregir *v* o menos errores, asumiendo que sea usada la regla de decodificación incompleta. Un código *C* es exactamente corrector de *v* errores si y solo si es corrector de *v* errores pero no será corrector de (*v* + 1) errores.

Ejemplo 2.1.4.8 Consideremos el código binario *C* = {000, 111}. Usando la regla de decodificación por distancias mínimas, observamos lo siguiente:

(i) Si 000 es enviado y un error ocurre en la transmisión, entonces recibimos 100, 010, o 001. Al emplear la regla de decodificación por distancias mínimas en cualquiera de los tres casos, codificaremos la palabra enviada como 000;

(ii) Si 111 es enviado y un error ocurre en la transmisión, entonces recibimos 110, 101, o 011. Al emplear la regla de decodificación por distancias mínimas en cualquiera de los tres casos, decodificaremos la palabra enviada como 111.

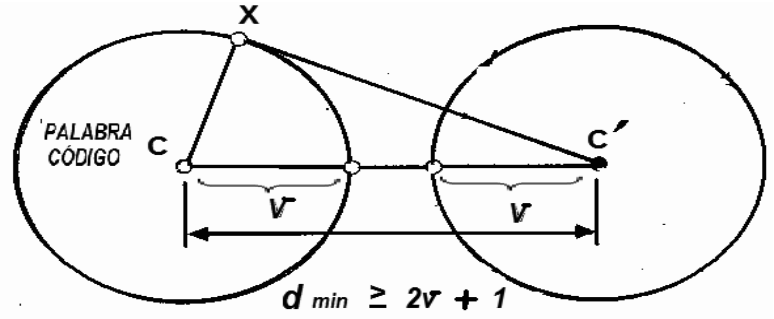
En todos los casos, es posible corregir un solo error. Por lo tanto, C es corrector de un error. Si al menos dos errores ocurrieran, la regla de decodificación por distancias mínimas podría producir una palabra código equivocada. Por ejemplo, si se envía 000 y 011 se recibe, entonces usando la regla de decodificación por distancias mínimas 011 será decodificada como 111. Por lo tanto, C es capaz de corregir exactamente un solo error.

Teorema 2.1.4.9 Un código C es corrector de *v* errores si y solo si d (C) ≥ 2*v* + 1, i.e., un código con distancia *d* es un código corrector de exactamente $\lfloor (d-1)/2 \rfloor$ errores. Aquí, $\lfloor x \rfloor$ es el mayor entero menor que o igual a x.

Demostración. Supongamos que d(C) ≥ 2*v* + 1. Sean **c** la palabra código enviada y **x** la palabra recibida. Si ocurrieran *v* o menos errores durante la transmisión, entonces d(**x**, **c**) ≤ *v*. Por lo tanto, para cualquier código **c**´ ∈ C, en donde **c** ≠ **c**´, se cumple que

$$\begin{aligned} d(\mathbf{x}, \mathbf{c}') &\geq d(\mathbf{c}, \mathbf{c}') - d(\mathbf{x}, \mathbf{c}) \\ &\geq 2v + 1 - v \\ &= v+1 \\ &> d(\mathbf{x}, \mathbf{c}). \end{aligned}$$

De hecho, ***x*** será decodificado correctamente a ***c*** si la regla de decodificación por distancias mínimas es usada. Por lo tanto, C corrige *v* errores (ver figura 2.1).



$d(x, c) < d(x, c') \geq d(c, c') - d(x, c)$

Figura 2.1 Decodificación de las esferas.

Supongamos que *C* corrige *v* errores. Si d*(C)* < *2v* + 1, entonces existen dos palabras distintas ***c*** y ***c´*** ∈ *C* con *d(**c**, **c´**) = d(C)* ≤ *2v*. Ahora demostraremos que, asumiendo que ***c*** es enviada y a lo más concurran *v* errores durante la transmisión, es posible que la regla de decodificación por distancia mínima decodifique la palabra recibida incorrectamente como ***c´*** o bien, que llegue a un empate. i.e., que haya dos distancias mínimas iguales y que en consecuencia no pueda corregirse ningún error si se emplea la decodificación incompleta por máxima similitud. Esto contradice la suposición de que *C* es un código corrector de *v* errores, lo cual queda demostrado que *d(C)* ≥ *2v*+1.

Notar que, si *d(**c**, **c'**)* < *v* + 1, entonces c puede convertirse en **c´** si se incurre a lo más en *v* errores, y esos errores no serán corregidos y ni siquiera serán detectados, puesto que ***c´*** esta nuevamente en *C*. Esto contradice la suposición de que *C* es corrector de *v* errores.

En general, un código puede corregir una combinación de t´ errores y detectar hasta **e** errores (**e** ≥ **t´**) si y solo si **e** + **t´** ≤ d_{min} - 1[15].

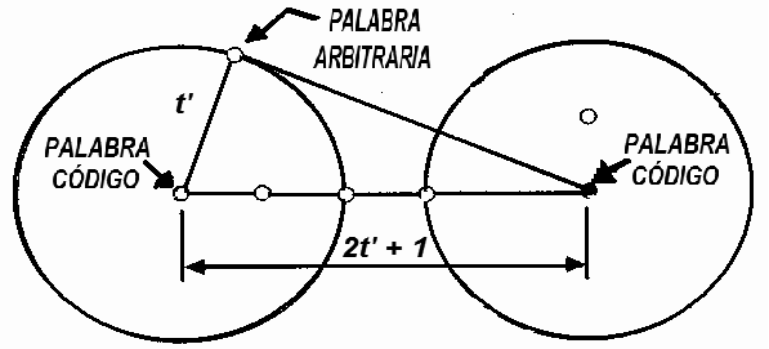


Figura 2.2 Código con distancia mínima de Hamming 2t´ + 1.

A continuación se muestra la tabla 2.2 con algunas posibles elecciones para detectar y corregir errores.

d_{min}	1	2	3		4		5		
e	0	1	2	1	3	2	4	3	2
t´	0	0	0	1	0	1	0	1	2

TABLA 2.2

[15] Se hace uso de **e** y **t** en lugar de *u* y *v* para diferenciar un patrón de error de una palabra código con errores usado frecuentemente en el diseño.

2.2 CÓDIGOS LINEALES.

Un código que forma un espacio vectorial es llamado código lineal. Nosotros describiremos un código lineal usando los parámetros [n, k] si las palabras código que pertenecen al código tienen longitud n y el código forma un espacio vectorial de dimensión k. En este apartado, se presentará la construcción de los códigos lineales usando matrices generadoras y considerando el problema de detectar y corregir errores en vectores recibidos. Debido a que un código lineal forma un espacio vectorial, existen técnicas para identificar vectores recibidos como palabras código que son mucho más eficientes que comparar los vectores con las palabras código uno por uno. Por dicha razón, para códigos con largo número de palabras código, los códigos lineales representan gran facilidad computacional para detectar y corregir errores que aquellos códigos que no forman una estructura de espacio vectorial (construcción, codificación y decodificación de códigos no lineales puede ser vista en [48]). Algunas razones por las que comúnmente se prefiere usar códigos lineales sobre códigos no lineales son los siguientes:

(i) Puesto que un código lineal es un espacio vectorial, éste puede ser descrito completamente por usar una base.
(ii) La distancia de un código lineal es igual al menor de los pesos de sus palabras código distintas de cero.
(iii) Los procedimientos para codificar y decodificar suelen ser mas rápidos y sencillos para códigos lineales que para aquellos no lineales.

2.2.1 Espacios vectoriales sobre campos finitos.

A continuación daremos algunas definiciones acerca de los espacios vectoriales sobre campos finitos.

Definición 2.2.1.1 Sea F_q el campo finito de orden q. Un conjunto no vacío V, junto con las operaciones de suma y multiplicación escalar para elementos del campo F_q, es un espacio vectorial o espacio lineal sobre F_q si satisface todas las siguientes condiciones. Para toda $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ y para, $\lambda, \mu \in F_q$:

(i) $\mathbf{u} + \mathbf{v} \in V$;
(ii) $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$;
(iii) existe un elemento $\mathbf{0} \in V$ con la propiedad de que $\mathbf{0} + \mathbf{v} = \mathbf{v} = \mathbf{v} + \mathbf{0}$ para toda $\mathbf{v} \in V$;
(iv) para cada $\mathbf{u} \in V$ existe un elemento de V, llamado $-\mathbf{u}$, tal que $\mathbf{u} + (-\mathbf{u}) = \mathbf{0} = (-\mathbf{u}) + \mathbf{u}$;
(v) $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$;
(vi) $\lambda\mathbf{v} \in V$;
(vii) $\lambda(\mathbf{u} + \mathbf{v}) = \lambda\mathbf{u} + \lambda\mathbf{v}$, $(\lambda + \mu)\mathbf{u} = \lambda\mathbf{u} + \mu\mathbf{u}$;
(viii) $(\lambda\mu)\mathbf{u} = \lambda(\mu\mathbf{u})$;
(ix) si 1 es la identidad multiplicativa en F_q, entonces $1\mathbf{u} = \mathbf{u}$.

Definición 2.2.1.2 Un subconjunto no vacío C de un espacio vectorial V es un subespacio de V si es también un espacio vectorial aplicando la suma y multiplicación escalar para V.

Ejemplo 2.2.1.3 El siguiente espacio vectorial sobre F_2 C = {(0, 0, 0, 0), (1, 0, 1, 0), (0, 1, 0, 1), (1, 1, 1, 1)}; es un subespacio de F_2^4.

Definición 2.2.1.4 Sea *V* un espacio vectorial sobre F_q. Una combinación lineal de $v_1,..., v_r \in V$ es un vector de la forma $\lambda_1 v_1 + \cdots + \lambda_r v_r$, donde $\lambda_1,..., \lambda_r \in F_q$ son escalares.

Definición 2.2.1.5 Sea *V* un espacio vectorial sobre F_q. El conjunto de vectores $\{v_1,..., v_r\}$ en V es linealmente independiente si

$$\lambda_1 v_1 + \cdots + \lambda_r v_r = 0 \Rightarrow \lambda_1 = \cdots = \lambda_r = 0.$$

El conjunto es linealmente dependiente si no es linealmente independiente; i.e., si hay $\lambda_1,..., \lambda_r \in F_q$, no todos cero, tales que $\lambda_1 v_1 + \cdots + \lambda_r v_r = 0$.

Ejemplo 2.2.1.6

(i) Cualquier conjunto S que contiene **0** es linealmente dependiente.

(ii) Para cualquier F_q, {(0, 0, 0, 1), (0, 0, 1, 0), (0, 1, 0, 0)} es linealmente independiente.

(iii) Para cualquier F_q, {(0, 0, 0, 1), (1, 0, 0, 0), (1, 0, 0, 1)} es linealmente dependiente.

Definición 2.2.1.7 *Sea V* un espacio vectorial sobre F_q y sea $S = \{v_1, v_2,..., v_k\}$ un subconjunto no vacío de *V*. La expansión lineal de S se define como

$$< S > = \{\lambda_1 v_1 + \cdots + \lambda_k v_k : \lambda_i \in F_q\}.$$

Si S = Ø, definimos $< S > = \{\mathbf{0}\}$. Es fácil verificar que $< S >$ es un subespacio de *V*, llamado el subespacio generado por *S*. Dando un subespacio *C* de *V*, un subconjunto *S* de *C* es llamado un conjunto generador de *C* si $C = < S >$.

Definición 2.2.1.8 Sea *V* un espacio vectorial sobre F_q. Un subconjunto no vacío $B = \{v_1, v_2,..., v_k\}$ de *V* es llamado una base de *V* si $V = < B >$ y *B* es linealmente independiente.

Es importante destacar que si $B = \{v_1, v_2,..., v_k\}$ es una base de *V*, entonces cualquier vector $\boldsymbol{v} \in V$ puede expresarse como una combinación lineal única de vectores en *B*. Es decir, existen únicas $\lambda_1, \lambda_2,..., \lambda_k \in F_q$ tales que

$$\boldsymbol{v} = \lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_k v_k.$$

Un espacio vectorial *V* sobre un campo finito F_q puede tener múltiples bases; pero todas esas bases contienen el mismo número de elementos. Este número de elementos es llamado dimensión de *V* sobre F_q, denotado como *dim (V)*.

Ejemplo 2.2.1.9 Si q=2, S = {0001, 0010, 0100} y V = <S>, entonces

$$V = \{0000, 0001, 0010, 0100, 0011, 0101, 0110, 0111\}.$$

Notar que S es linealmente independiente, entonces *dim (V)=3=k*. Observamos que *V* tiene q^k elementos; i.e., $|V| = 2^3 = 8$.

Definición 2.2.1.10 Sea $\boldsymbol{v} = (v_1, v_2,..., v_n)$, $\boldsymbol{w} = (w_1, w_2,..., w_n) \in F_q^n$.

(i) El producto escalar o producto euclidiano interno de *v* y *w* se define como

$$\boldsymbol{v} * \boldsymbol{w} = v_1w_2 + v_2w_2 + \cdots + v_nw_n \in F_q.$$

.

(ii) Los dos vectores $\boldsymbol{v}$ y $\boldsymbol{w}$ son ortogonales si $\mathbf{v} * \mathbf{w} = 0$.

(iii) Sea *S* un subconjunto no vacío de F_q^n. El complemento ortogonal $S^{\perp}$ de *S* se define como

$$S^{\perp} = \{\mathbf{v} \in F_q^n : \boldsymbol{v} * \mathbf{s} = 0 \text{ para toda s} \in S\}.$$

Si S = Ø, entonces definimos $S^{\perp} = F_q^n$.

Teorema 2.2.1.11 Sea S un subconjunto de F_q^n, entonces tenemos

$$dim (< S >) + dim (S^{\perp}) = n.$$

Demostración. Para el caso trivial es obvio que < S > = {0}. Sea $dim (< S >) = k \geq 1$ y supongamos que $\{\mathbf{v}_1,..., \mathbf{v}_k\}$ es una base de < S >. Necesitamos demostrar que $dim (S^{\perp}) = dim (<S>^{\perp}) = n - k$.
Notamos que $\mathbf{x} \in S^{\perp}$ si y solo si

$$v_1 * \mathbf{x} = \cdots = v_k * \mathbf{x} = 0,$$

Lo que es equivalente a decir que **x** satisface $A\boldsymbol{x}^T = 0$, donde *A* es la matriz de *k*x*n* cuyo i-esimo renglón es $\mathbf{v}_i$. Los renglones de *A* son linealmente independientes, de tal manera que $A\boldsymbol{x}^T = 0$ es un sistema lineal de *k* ecuaciones linealmente independientes de *n* variables. Del álgebra lineal, se sabe que dicho sistema admite un espacio de solución de *n - k*.

Ejemplo 2.2.1.12 Sea q = 2, n = 4 y S = {0100, 0101}. Entonces tenemos que

$$< S >= \{0000, 0100, 0001, 0101\}.$$

Notar que S es linealmente independiente, entonces dim $(<S>)=2$. Para encontrar $S^{\perp}$, sea $v = (v_1, v_2, v_3, v_4) \in S^{\perp}$. Entonces

$$v * (0, 1, 0, 0) = 0 \Rightarrow v_2 = 0,$$
$$v * (0, 1, 0, 1) = 0 \Rightarrow v_2 + v_4 = 0.$$

Por lo tanto, tenemos $v_2 = v_4 = 0$. Como v_1 y v_3 pueden ser 0 o 1, se concluye que

$$S^{\perp} = \{0000, 0010, 1000, 1010\}.$$

Enseguida notamos que {0010, 1000} es una base para $S^{\perp}$, por lo tanto dim $(S^{\perp}) = 2$. Por lo tanto, se puede verificar que:

$$\dim (< S >) + \dim (S^{\perp}) = 2 + 2 = 4 = n.$$

2.2.2 Códigos Lineales.

A continuación definiremos un código lineal y sus propiedades elementales.

Definición 2.2.2.1 Un *código lineal* C de longitud n sobre F_q es un subespacio de F_q^n.

Definición 2.2.2.2 Sea *C* un código lineal en F_q^n.

(i) El código dual de *C* es $C^{\perp}$, el complemento ortogonal del subespacio *C* de F_q^n.
(ii) La *dimensión* del código lineal *C* es la dimensión de *C* como un espacio vectorial sobre F_q, se denota como *dim (C).*

Teorema 2.2.2.3 Sea *C* un código lineal de longitud n sobre F_q. Entonces,

(i) $|C| = q^{\dim(C)}$, i.e., $\dim(C) = \log_q |C|$;
(ii) $C^{\perp}$ es un código lineal y $\dim(C) + \dim(C^{\perp}) = n$;
(iii) $(C^{\perp})^{\perp} = C$.

Demostración.

(i) Si $\{ c_1,..., c_k)$ es una base de C, entonces

$$C = \{\lambda_1 c_1 + \cdots + \lambda_k c_k : \lambda_1,\ldots,, \lambda_k \in F_q\}.$$

Como $|F_q| = q$, hay exactamente q posibilidades para cada $\lambda_1,\ldots, \lambda_k$; entonces, *C* tiene q^k *elementos donde dim(C)=k, es decir* $|C| = q^{dim(C)}$.

(ii) Verificamos que $S^{\perp}$ es siempre un subespacio del espacio vectorial F_q^n, para cualquier subconjunto *S de* F_q^n, y que $dim(< S >^{\perp}) = dim(S^{\perp}) = n - k$. *De* acuerdo al teorema 2.2.1.11 solo es necesario cambiar C = S.

(iii) Si tenemos que $dim(C) = dim((C^{\perp})^{\perp})$. Es suficiente con demostrar que $C \subseteq (C^{\perp})^{\perp}$. Sea $\boldsymbol{c} \in C$. Para mostrar que $\boldsymbol{c} \in (C^{\perp})^{\perp}$, necesitamos probar que $\boldsymbol{c} * \boldsymbol{x} = 0$ para toda $\boldsymbol{x} \in C^{\perp}$. Puesto que $\boldsymbol{c} \in C$ y $\boldsymbol{x} \in C^{\perp}$, por definición de $C^{\perp}$, se sigue que $\boldsymbol{c} * \boldsymbol{x} = 0$. Lo cual (iii) queda demostrado.

Observación 2.2.2.4 Un código lineal *C* de longitud *n* y de dimensión *k* sobre F_q es frecuentemente llamado código *q-ario [n, k]* o, si *q* es claro en el contexto, simplemente código *[n, k]*. También se denota como código lineal (n, q^k). Si la distancia *d* de C es conocida, los parámetros de un código lineal se denotan como *[n, k, d]*.

Definición 2.2.2.5 Sea *C* un código lineal

(i) *C es auto-ortogonal si* $C \subseteq C^{\perp}$.
(ii) *C es auto-dual si* $C = C^{\perp}$.

Proposición 2.2.2.6 La dimensión de un código auto-ortogonal de longitud n debe ser $\leq$ n/2, y la dimensión de un código-auto dual de longitud n es n/2.

Ejemplo 2.2.2.7 Determinar si el siguiente código es lineal. q=2 y C = {00000, 11110, 01111, 10001}. Del Teorema 2.2.2.3.
Para (i).- dim(C)= $\log_q |C| = \log_2 4 = 2$.
Para (ii).- $C^{\perp}$ = {00000, 10011, 10101, 00110, 11001, 01010, 01100, 11111}
$\dim(C^{\perp}) = \log_q |C^{\perp}| = \log_2 8 = 3 = k$. Por lo tanto, $\dim(C) + \dim(C^{\perp}) = 2 + 3 = 5 = n$.
Para (iii).- $\dim((C^{\perp})^{\perp}) = \dim(C) = \log_2 4 = 2$.

Ejemplo 2.2.2.8 Determinar si el siguiente código es lineal. q=2 y C = {0000, 1010, 0101, 1111}. Del Teorema 2.2.2.3.
Para (i).- dim(C)= $\log_q |C| = \log_2 4 = 2$.
Para (ii).- $C^{\perp}$ = {0000, 1010, 0101, 1111}.
$\dim(C^{\perp}) = \log_q |C^{\perp}| = \log_2 4 = 2 = k$. Por lo tanto, $\dim(C) + \dim(C^{\perp}) = 2 + 2 = 4 = n$.
Para (iii).- $\dim((C^{\perp})^{\perp}) = \dim(C) = \log_2 4 = 2$.
Nota: este código es autodual.

2.2.3 Bases para códigos lineales.

Como un código lineal es un espacio vectorial, todos sus elementos pueden ser descritos en términos de una base. En este apartado, describiremos tres algoritmos que proporcionan una base para un código lineal o bien para su dual. Las siguientes dos definiciones son aspectos fundamentales del álgebra lineal.

Definición 2.2.3.1 Sea A una matriz sobre F_q; una operación de renglón elemental valida en A es cualquiera de las siguientes tres operaciones:

(i) Intercambiar dos renglones,
(ii) Multiplicar un renglón por un escalar distinto de cero,
(iii) Reemplazar un renglón por su suma con otro renglón previamente multiplicado por un escalar.

Definición 2.2.3.2 Dos matrices son *equivalentes por renglón* si una puede ser obtenida a partir de la otra por una secuencia de operaciones de renglón elementales.

(i) Cualquier matriz M sobre F_q puede escribirse en la *forma escalonada por renglón* o en la *forma escalonada reducida por renglón* por medio de una secuencia de operaciones de renglón elementales. En otras palabras, una matriz es equivalente por renglón a otra matriz de la forma escalonada por renglón o forma escalonada reducida por renglón.
(ii) Para una matriz dada, su *forma escalonada reducida por renglón* es única, pero puede tener diferentes formas escalonadas por renglón.

A continuación se explican tres algoritmos para obtener la base de un código lineal.

Algoritmo 1.1

Entrada: Un subconjunto no vacío S de F_q^n .

Salida: Una base para C = < S >, el código lineal generado por *S*.
Descripción: Formar la matriz *A* cuyos renglones son las palabras en *S*. Por medio de operaciones de renglón elementales encontramos la forma escalonada por renglón de *A*. Después, los renglones distintos de cero de la forma escalonada por renglón de *A* forman una base de *C*.

Ejemplo 2.2.3.3 Sea q = 3. Encontrar una base para C = <S>, donde

$$S = \{12101, 20110, 01122, 11010\}$$

$$A = \begin{pmatrix} 12101 \\ 20110 \\ 01122 \\ 11010 \end{pmatrix} \rightarrow \begin{pmatrix} 12101 \\ 02211 \\ 01122 \\ 02212 \end{pmatrix} \rightarrow \begin{pmatrix} 12101 \\ 01122 \\ 00001 \\ 00000 \end{pmatrix}.$$

La matriz A esta es su la forma escalonada por renglón y {12101, 01122, 00001} es una base para C.

Algoritmo 1.2

Entrada: Un subconjunto no vacío S de F_q^n .

Salida: Una base para $C = < S >$, el código lineal generado por *S.*

Descripción: Formar la matriz *A* cuyos columnas son las palabras en S. Usar operaciones de renglón elementales para encontrar la forma escalonada por renglón de la matriz *A* y localizar las columnas líderes. Las columnas originales de *A* correspondientes a las columnas líderes y forman una base para C. La base que se obtiene en el Algoritmo 1.2 proporciona un subconjunto del conjunto dado S, lo cual no ocurre necesariamente en el Algoritmo 1.1.

Algoritmo 1.3

Entrada: Un subconjunto no vacío de S de F_q^n .

Salida: Una base para el código dual $C^{\perp}$, donde $C = < S >$.

Descripción: Formar la matriz *A* cuyos renglones son las palabras en *S*. Utilizar operaciones de renglón elementales para encontrar la forma escalonada reducida por renglón de la matriz *A.* Sea *G* la matriz de k x n que consta de todos los renglones distintos de cero de *A:*

$$A \rightarrow \begin{pmatrix} G \\ 0 \end{pmatrix}$$

donde *0* denota la matriz cero.

La matriz *G* contiene *k* columnas líderes. Permutar las columnas de *G* para formar

$$G' = (I_k \mid X),$$

donde I_k denota la matriz identidad de *k x k.* formar la matriz *H´* como sigue:

$$H' = (-X^T \mid I_{n-k}),$$

donde $X^{\perp}$ denota la transpuesta de X.

Aplicar la inversa de la permutación aplicada a las columnas de *G* a las columnas de *H´* para formar *H*. Después, los renglones de *H* forman una base de $C^{\perp}$. Notar que el Algoritmo 1.3 proporciona también una base para C puesto que incluye al Algoritmo 1.1.

2.2.4 Matriz generadora y matriz parity check.

Conociendo la base de un código lineal podemos describir las palabras del código mismo. En la teoría de códigos, se representa con frecuencia la base de un código lineal por medio de una matriz, llamada *matriz generadora* y la matriz que representa la base de su código dual es llamada *matriz parity check*. Estas matrices juegan un papel muy importante en la teoría de códigos ya que como se verá posteriormente son usadas para codificar y decodificar un código lineal.

Definición 2.2.4.1 (i) Una *matriz generadora* para un código lineal C es una matriz G cuyos renglones forman una base para C.
(ii) Una *matriz parity check* H para un código lineal C es la matriz generadora del código dual $C^{\perp}$.

Observaciones 2.2.4.2

(i) Si C es un código lineal con parámetros $[n, k]$, entonces su matriz generadora C es una matriz de k x n y su matriz parity check para C es una matriz $(n - k)$ x n.

(ii) El algoritmo 1.3 permite encontrar la matriz generadora y la matriz parity check de un código lineal.

(iii) Como el número de bases para un espacio vectorial usualmente es mayor a uno, la matriz generadora de un código lineal no es única, en consecuencia, tampoco lo es la matriz generadora de su código dual. Además, aun considerando una matriz generadora fija, al permutar sus renglones se obtendría una matriz generadora diferente igualmente valida para el código lineal.

(iv) Los renglones de la matriz generadora son linealmente independientes. Del mismo modo, los renglones de la matriz parity check. Para probar que la matriz G de k x n es en efecto una matriz generadora para el código lineal C con parámetros $[n, k]$, es suficiente con mostrar que los renglones de G son palabras de código de C y que son linealmente independientes.

Definición 2.2.4.3 (i) Una matriz generadora en la forma $(I_k \mid X)$ se dice que se encuentra en su *forma estándar*.

Lema 2.2.4.4 Sea C un código lineal [n, k] sobre F_q, con matriz generadora G. Entonces $\mathbf{v} \in F_q^n$ pertenece a $C^{\perp}$ si y solo si $\mathbf{v}$ es ortogonal para todo renglón de G; i.e., $\mathbf{v} \in C^{\perp} \Leftrightarrow \mathbf{v}G^T = 0$. En particular, dada una matriz H de (n – k) x n, entonces H es una matriz parity check para C si y solo si los renglones de H son linealmente independientes y $HG^T = 0$.

Demostración. Sea $\mathbf{r}_i$ el i-ésimo renglón de G. En particular, $\mathbf{r}_i \in C$ para toda $1 \leq i \leq k$, y toda $\mathbf{c} \in C$ puede ser escrito como

$$\mathbf{c} = \lambda_1 \mathbf{r}_1 + \cdots + \lambda_k \mathbf{r}_k,$$

donde $\lambda_1, \ldots, \lambda_k \in F_q$.

Si $\boldsymbol{v} \in C^{\perp}$, entonces $\boldsymbol{v} \cdot \boldsymbol{c} = 0$ para toda $\boldsymbol{c} \in C$. En particular, $\boldsymbol{v}$ es ortogonal a $\boldsymbol{r}_i$, para toda $1 \leq i \leq k$; i.e., $\boldsymbol{v}G^T = 0$. Por el contrario, si $\boldsymbol{v} \cdot \boldsymbol{r}_i = 0$ para toda $1 \leq i \leq k$, entonces para cualquier $\mathbf{c} = \lambda_1 \mathbf{r}_1 + \cdots + \lambda_k \mathbf{r}_k \in C$,

$$\boldsymbol{v} \cdot \boldsymbol{c} = \lambda_1(v \cdot \boldsymbol{r}_1) + \cdots + \lambda_k(v \cdot \boldsymbol{r}_k) = 0.$$

Para esta última parte, si H es una matriz parity check para el código C, entonces los renglones de H son linealmente independientes por definición. Puesto que los renglones de H son palabras código de $C^{\perp}$, se sigue de la declaración anterior que $HG^T = 0$.

Asimismo, si $HG^T = 0$, entonces la declaración anterior muestra que los renglones de H, y por tanto el espacio de renglones de H, están contenidos en $C^{\perp}$. Como los renglones de H son linealmente independientes, el espacio de renglones de H tiene dimensión $n - k$, por lo que el espacio de renglones de H es en realidad $C^{\perp}$. En otras palabras, H es una matriz parity check para C.

Una alternativa, pero equivalente forma de plantear el lema 2.2.4.4 anterior es la siguiente:

Sea C un código lineal [n, k] sobre F_q, con matriz parity check H. Entonces $\mathbf{v} \in F_q^n$ pertenece a C si y solo si v es ortogonal a cada renglón de H; i.e., $\mathbf{v} \in C \Leftrightarrow \mathbf{v}H^T = 0$. En particular, dada una matriz G de k x n, entonces G es una matriz generadora para C si y solo si los renglones de G son linealmente independientes y $GH^T = 0$.

Una de las consecuencias del lema 2.2.4.4 es el siguiente teorema que relaciona la distancia de un código lineal C con las propiedades de una matriz parity check de C.

Teorema 2.2.4.5 Sea C un código lineal y sea H una matriz parity check para C. Entonces

(i) C tiene distancia $\geq d$ si y solo si cualquier $d - 1$ columnas de H son linealmente independientes; y

(ii) C tiene distancia $\leq d$ si y solo si H tiene d columnas que son linealmente dependientes.

Demostración. *Para (i)* Sea $\boldsymbol{v} = (v_1, \ldots, v_n) \in C$ es una palabra de peso $\boldsymbol{e} > 0$. Supongamos que las coordenadas de $\boldsymbol{v}$ distintas de cero se encuentran en las posiciones $i_1, \ldots, i_e$, de manera que $v_j = 0$ si $j \notin \{i_1, \ldots, i_e\}$. Sea $\boldsymbol{c}_i$ (donde $1 \leq i \leq n$) la i-ésima columna de H.

Por el lema 2.2.4.4 C contiene la palabra $\boldsymbol{v} = (v_1, \ldots, v_n)$ de peso $\boldsymbol{e}$ (cuyas coordenadas distintas de cero son $v_{i_1}, \ldots, v_{i_e}$) si y solo si

$$0 = vH^T = v_{i_1} c_{i_1}^T + \cdots + v_{i_e} c_{i_e}^T,$$

lo cual es cierto si y solo si existen $\boldsymbol{e}$ columnas de H (concretamente, $c_{i_1}, \ldots, c_{i_e}$) que son linealmente dependientes.

Decimos que si la distancia de C es $\geq d$ es equivalente a decir que C no tiene ninguna palabra distinta de cero de peso $\leq d - 1$, es equivalente a decir nuevamente que cualesquiera $\leq d - 1$ columnas de H son linealmente independientes.
Para (ii) De manera semejante, decimos que la distancia de C es $\leq d$ equivale a decir que H tiene $\leq d$ columnas (y por lo tanto d columnas) que son linealmente dependientes.

Corolario 2.2.4.6 Sea C un código lineal y sea H una matriz parity check para C. Las siguientes afirmaciones son equivalentes:

(i) C tiene distancia d;
(ii) Cualesquiera d – 1 columnas de H son linealmente independientes y H tiene d columnas que son linealmente dependientes.

Ejemplo 2.2.4.7 Sea C un código lineal binario con matriz parity check**.**

$$H = \begin{pmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{pmatrix}$$

Puede apreciarse que no hay columnas cero y tampoco dos o tres columnas que sumen 0^T en H, así que tres columnas de H son linealmente independientes. Sin embargo, las columnas 1, 2, 3 y 6 suman a 0^T, y por lo tanto son linealmente dependientes. Como conclusión decimos que la distancia de C es $d = 4$.

Teorema 2.2.4.8 Si $G = (I_k \mid X)$ es la forma estándar de la matriz generadora de un código $C – [n, k]$, entonces la matriz parity check de C es $H = (-X^T \mid I_{n-k})$.

Demostración. Si la ecuación $HG^T=O$ se satisface. Por considerar las últimas $n - k$ coordenadas, es claro que los renglones de H son linealmente independientes. Por lo tanto, la conclusión se sigue del lema 2.2.4.4.

Ejemplo 2.2.4.9 Encontremos la matriz generadora y su matriz parity check en su forma estándar, para el código lineal binario $C=<S>$, donde S={0000000, 1011100, 0101110, 0010111, 1001011, 1100101, 1110010, 0111001}.

Utilizando la forma escalonada por renglón encontramos la matriz generadora G y utilizando la matriz escalonada reducida en su forma $G' = (I_k|X)$ que es la matriz generadora en su forma estándar.

$$C = \begin{pmatrix} 0000000 \\ 1011100 \\ 0101110 \\ 0010111 \\ 1001011 \\ 1100101 \\ 1110010 \\ 0111001 \end{pmatrix} \rightarrow \begin{pmatrix} 1011100 \\ 0101110 \\ 0010111 \\ 0000000 \\ 0000000 \\ 0000000 \\ 0000000 \\ 0000000 \end{pmatrix} = G$$

$$G' = \begin{pmatrix} 1001011 \\ 0101110 \\ 0010111 \end{pmatrix}$$

La matriz parity check debe ser
(n-k) x *n= (7-3)* x *n =4* x *7*
$H = (-X|I)$

$$X = \begin{pmatrix} 1011 \\ 1110 \\ 0111 \end{pmatrix} \qquad -X^T = \begin{pmatrix} 110 \\ 011 \\ 111 \\ 101 \end{pmatrix}$$

Por lo tanto: H es la matriz parity check y $H'_{(I_k|-X^T)}$ es su forma estándar.

$$H = \begin{pmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{pmatrix}, \qquad H'_{(I_k|-X^T)} = \begin{pmatrix} 1000110 \\ 0100011 \\ 0010111 \\ 0001101 \end{pmatrix}.$$

Se observa que:

1.- La matriz H tiene dimensión 4, puesto que H cuenta con cuatro renglones, por lo que $dim(C^{\perp}) = 4$.
2.- Del ejemplo 2.2.4.7 $d(C) = 4$.
3.- De la matriz generadora de C, se observa que $dim(C) = 3 = k$, por lo que el código binario C cuenta con $q^k = 2^3 = 8$ elementos o palabras código.
4.- Puede verificarse que $dim(C) + dim(C^{\perp}) = 3 + 4 = 7 = n$,
5.- Concluimos que los parámetros del código son C – [7, 3, 4], de acuerdo con la notación que hemos venido manejando.

2.2.5 Equivalencia de códigos lineales.

Ciertos códigos lineales pueden no tener una matriz generadora en su forma estándar, después de algunas operaciones de renglón elementales de las palabras código que la componen y, posiblemente, multiplicando ciertas coordenadas por escalares distintos de cero, podemos encontrar un nuevo código que si tenga una matriz generadora en su forma estándar.

Definición 2.2.5.1 Dos códigos (n, M) sobre F_q son equivalentes si uno puede obtenerse a partir del otro por medio de una combinación de operaciones del siguiente tipo:

(i) Permutación de los n dígitos de las palabras código;
(ii) Multiplicación de los símbolos que aparecen en una posición fija por un escalar distinto de cero.

Ejemplo 2.2.5.2 Sea q=3 y n=3. Considere el siguiente código ternario.

$$C = \{000, 011, 022\}.$$

Permutando la primera y segunda posición, se sigue por multiplicar la tercera posición por 2, obtenemos el siguiente código equivalente:

$$C' = \{000, 102, 201\}.$$

Teorema 2.2.5.3 Cualquier código lineal C es equivalente a un código lineal C´ con una matriz generadora en su forma estándar.

Demostración. Si G es una matriz generadora para C, pongamos G en una *forma escalonada reducida por renglón*. Acomodando las columnas de manera que las columnas líderes formen una matriz identidad. El resultado es la matriz, G', en su forma estándar que es la matriz generadora de un código C' equivalente al código C.

Ejemplo 2.2.5.4 Sea C un código lineal binario con la siguiente matriz generadora

$$G = \begin{pmatrix} 1100001 \\ 0010011 \\ 0001001 \end{pmatrix}$$

Acomodando las columnas en el orden 1, 3, 4, 2, 5, 6, 7 se produce la siguiente matriz

$$G' = \begin{pmatrix} 100\,|\,1001 \\ 010\,|\,0011 \\ 001\,|\,0001 \end{pmatrix}$$

Sea C′ el código generado por G′; entonces C′ es equivalente a C y C′ tiene una matriz generadora G′, la cual está en su forma estándar.

2.3 CODIFICACIÓN Y DECODIFICACIÓN DE UN CÓDIGO LINEAL.

2.3.1 Codificación de un código lineal.

Sea C un código lineal $[n, k, d]$ sobre el campo finito F_q. Cada palabra código de C puede representar una pieza de información, por lo que C puede representar q^k partes de información distintos. Una vez que se fija una base $\{\boldsymbol{r}_1,..., \boldsymbol{r}_k\}$ para C, cada palabra código $\mathbf{v}$, o, cada parte de información q^k, puede ser escrita de forma única por medio de una combinación lineal,

$$\mathbf{v} = u_1\boldsymbol{r}_1 + \cdots + u_k\boldsymbol{r}_k,$$

donde $u_1,..., u_k \in F_q$.

De manera equivalente, podemos establecer un conjunto G como matriz generadora de C cuyo *i-esimo* renglón es el vector $\mathbf{r}_1$ de la base elegida. Dado un vector $\boldsymbol{u} = (u_1,..., u_k) \in F_q^k$, entonces

$$\boldsymbol{v} = \boldsymbol{u}G = u_1\boldsymbol{r}_1 + \cdots + u_k\boldsymbol{r}_k$$

es una palabra código en C. Por otra parte, cualquier $\mathbf{v} \in$ C puede escribirse de forma única como $\boldsymbol{v} = \boldsymbol{u}G$, donde $\mathbf{u} = (u_1,...., u_k) \in F_q^k$. Por lo tanto, cada palabra $\mathbf{u} \in F_q^k$ puede ser codificada como $\boldsymbol{v} = \boldsymbol{u}G$. El procesos de representar los elementos de $\mathbf{u} \in F_q^k$ como palabras del código $\mathbf{v} = \boldsymbol{u}G$ en C se conoce como *codificación.*

Ejemplo 2.3.1.1 Sea el código binario [7, 3, 4] (ver ejemplo 2.2.4.9) con la siguiente matriz generadora

$$G' = \begin{pmatrix} 1001011 \\ 0101110 \\ 0010111 \end{pmatrix}.$$

Entonces, los mensajes son codificados por asignar las siguientes letras a las palabras F_q^3 como a continuación se describe

000	001	010	011	100	101	110	111
U	**N**	**I**	**V**	**E**	**R**	**S**	**O**

Ahora **v** = **u**G, para cada $\mathbf{u}_i$.

$$u_1 = u_1G = (000)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (0000000),$$

$$u_2 = u_2G = (001)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (0010111),$$

$$u_3 = u_3G = (010)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (0101110),$$

$$u_4 = u_4G = (011)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (0111001),$$

$$u_5 = u_5G = (100)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (1001011),$$

$$u_6 = u_6G = (101)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (1011100),$$

$$u_7 = u_7G = (110)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (1100101),$$

$$u_8 = u_8G = (111)\begin{pmatrix}1001011\\0101110\\0010111\end{pmatrix} = (1110010).$$

Por lo tanto, el mensaje **UNIVERSO** queda codificado como:

0000000	0010111	0101110	0111001	1001011	1011100	1100101	1110010
U	**N**	**I**	**V**	**E**	**R**	**S**	**O**

Algunas ventajas de tener G en forma estándar son las siguientes:

(i) Si un código lineal *C* tiene una matriz generadora *G* en su forma estándar, $G = (I \mid X)$, entonces el Algoritmo 1.3 a la vez proporciona

$$H = (- X^T \mid I)$$

como una matriz parity check para *C*.

(ii) Si un código lineal [*n, k, d*] tiene matriz generadora *G* en su forma estándar, $G = (I \mid X)$, entonces recuperar el mensaje $\boldsymbol{u}$ a partir de la palabra código $\boldsymbol{v} = \boldsymbol{u}G$ es trivial, puesto que

$$\boldsymbol{v} = \boldsymbol{u}G = \boldsymbol{u}\,(I \mid X) = (\boldsymbol{u}, \boldsymbol{u}X);$$

Es decir, los primeros *k* dígitos de la palabra código $\boldsymbol{v} = \boldsymbol{u}G$ dan el mensaje $\boldsymbol{u}$. Los dígitos $n - k$ restantes son comúnmente llamados *dígitos verificadores*. Los dígitos verificadores representan la *redundancia* la cual ha sido agregada al mensaje para su protección contra el ruido.

2.3.2 Decodificación de los códigos lineales.

En este apartado, se analizará el método de decodificación del vecino más cercano para un código lineal, así como una modificación que improvisa su ejecución cuando la longitud del código es demasiado larga. Además, se analizará el método de decodificación por síndromes recomendable para valores pequeños de n. Recordemos que para que un código sea práctico su algoritmo de decodificación deberá ser eficiente. A continuación se establece la definición de co-conjunto. Los co-conjuntos son muy importantes en los esquemas de decodificación para códigos correctores de errores.

Definición 2.3.2.0.1 Sea *C* un código lineal de longitud *n* sobre F_q y sea $\boldsymbol{u} \in F_q^n$ cualquier vector de longitud *n*; definimos el co-conjunto de *C* determinando por $\boldsymbol{u}$ como el conjunto

$$C + \boldsymbol{u} = \{v + \boldsymbol{u} : v \in C\} = (\boldsymbol{u} + C).$$

Teorema 2.3.2.0.2 Sea *C* un código lineal *[n, k, d]* sobre el campo finito F_q. Entonces,

(i) todo vector de F_q^n esta contenido en algún co-conjunto de C;

(ii) para todo $\mathbf{u} \in F_q^n$, $|C + \mathbf{u}| = |C| = q^k$;

(iii) para todos $\mathbf{u}, \mathbf{v} \in F_q^n$, $\mathbf{u} \in C + \mathbf{v}$ implica que $C + \mathbf{u} = C + \mathbf{v}$;

(iv) dos co-conjuntos son idénticos, o tiene intersección nula;

(v) hay q^{n-k} co-conjuntos diferentes de C;

(vi) para todos $\mathbf{u}, \mathbf{v} \in F_q^n$, $\mathbf{u} - \mathbf{v} \in C$ si y solo si $\mathbf{u}$ y $\mathbf{v}$ están en el mismo co-conjunto.

Demostración.

(i) El vector $\mathbf{v} \in F_q^n$ está claramente contenido en el co-conjunto C + **v**.

(ii) Por definición, C + **u** tiene a lo más | C | = q^k elementos. Dos elementos ***c*** + ***u*** *y* ***c'*** + ***u*** *de C* + ***u*** son iguales si y solo si ***c*** = ***c'***, por lo tanto | *C*+ ***u*** | = |*C*| = q^k.

(iii) Se sigue de la definición de C + **v** que C + **u** $\subseteq$C + **v**. Entonces, por (ii), C + **u** = C + **v**.

(iv) Consideramos dos co-conjuntos C + **u** y C + **v** y supongamos que **x** $\in$ (C + **u**) $\cap$(C + **v**). Como **x** $\in$ C + **u**, (iii) muestra que C + **u** = C + **x**. De manera similar, **x** $\in$ C + ***v***, se sigue que *C* + ***v*** = *C* + ***x***. Por lo tanto, *C* + ***u*** = *C* + ***v***.

(v) Se demuestra directamente de (i), (ii) y (iv).

(vi) Si **u** – **v** = **c** $\in$ C, entonces **u** = **c** + **v** $\in$ C + **v**, por lo que C + **u** = C + **v**. Por la prueba de (i), **u** $\in$ C + **u** y **v** $\in$ C + **v**, por lo que **u** y **v** están en el mismo co-conjunto. Por otra parte, supongamos **u**, **v** están ambos en el co-conjunto C + **x**. Entonces **u** = **c**+ **x** y **v** = **c**´+ x, para algún **c**, **c**´ $\in$ C. Por lo tanto, **u** - **v** = **c** - **c**´ $\in$ C.

Definición 2.3.2.0.3 A la palabra con el menor peso de Hamming en un co-conjunto se le llama *co-conjunto líder*.

Ejemplo 2.3.2.0.4 Los co-conjuntos del siguiente código lineal binario

$$C = \{0000, 1011, 0101, 1110\}$$

Son los siguientes:

0000 + C:	0000	1011	0101	1110
0001 + C:	0001	1010	0100	1111
0010 + C:	0010	1001	0111	1100
1000 + C:	1000	0011	1101	0110

Notar que los co-conjuntos líderes pertenecen a las palabras de menor peso y que en el caso de 0001 + C los co-conjuntos lideres pueden ser 0001 y 0100. Además, el arreglo anterior es llamado un arreglo estándar (Slepian [48, 60]).

2.3.2.1 Decodificación por el método del vecino más cercano.

Sea C un código lineal. Si la palabra código **v** es transmitida y la palabra **w** es recibida, de forma tal que hay un patrón de error que contiene **w**:

$$\mathbf{e} = \mathbf{w} - \mathbf{v} \in \mathbf{w} + C.$$

Entonces **w** – **e** = **v** $\in$ C, por lo que, empleando (vi) del Teorema 2.3.2.0.2, el patrón error **e** y la palabra recibida **w** se encuentran en el mismo co-conjunto.

Para los casos en que la probabilidad de error es pequeña, la decodificación por medio del vecino mas cercano funciona para la decodificación de códigos lineales de la siguiente manera. Una vez que es recibida la palabra **w**, elegimos la palabra **e** de menor peso en el co-conjunto **w** + C y concluimos que **v** = **w** – **e** es la palabra que fue transmitida originalmente.

Ejemplo 2.3.2.1.1 Sea q = 2 y C = {0000, 1011, 0101, 1110}. Decodificar la siguientes palabra recibida: (i) **w** = 1101. De acuerdo al ejemplo 2.3.2.0.4 **w** + C está en el cuarto co-conjunto. La palabra de menor peso en dicho co-conjunto es 1000. Por lo tanto, 1101-1000 = 1101+1000 = **0101** fue la más parecida palabra código transmitida.

Ejemplo 2.3.2.1.2 Sea q = 2 y C = {0000, 1011, 0101, 1110}. Decodificar la siguiente palabra recibida: (i) **w** = 1111. De acuerdo al ejemplo 2.3.2.0.4 **w** + C está en el segundo co-conjunto. Mientras que el segundo co-conjunto tiene dos palabras de menor peso 0001 y 0100 (en el arreglo elegimos 0001 como co-conjunto líder, si se hubiera elegido 0100 tendríamos una pequeña diferencia y esto debido a que utilizamos un arreglo en forma estándar para elegir los co-conjuntos). Si se realiza una decodificación completa se elige 0001, como el patrón de error, y concluimos que 1111 – 0001 = 1111 + 0001 = **1110** es la palabra código que fue enviada, la cual aparece en la parte superior del arreglo en forma estándar para C.

2.3.2.2 Decodificación por síndromes.

El esquema de decodificación basado en un arreglo en forma estándar trabaja razonablemente bien cuando la longitud *n* de un código lineal es pequeña, pero puede tomar una cantidad de tiempo considerable conforme *n* se hace grande. Puede ahorrarse tiempo haciendo uso de un síndrome para identificar el co-conjunto al que pertenece la palabra recibida. Como se presenta en este apartado, el uso de decodificación por síndrome representa una forma sencilla y fácil de decodificar los códigos correctores de errores. Pero no es recomendable prácticamente para códigos de gran tamaño.

Definición 2.3.2.2.1 Sea C un código lineal *[n, k, d]* sobre F_q y sea H una matriz parity check para *C*. Para cualquier **w** $\in F_q^n$, el síndrome de **w** es la palabra $S(\boldsymbol{w}) = \boldsymbol{w}H^T \in F_q^{n-k}$.

Teorema 2.3.2.2.2 Sea *C* y un código lineal *[n, k, d]* y sea H una matriz parity check para C. Para **u**, **v** $\in F_q^n$, tenemos que

(i) S(**u** + **v**) = S (**u**) + S (**v**) ;
(ii) S (**u**) = 0 si y solo si **u** es una palabra código en C;
(iii) S (**u**) = S (**v**) si y solo si **u** y **v** pertenecen al mismo co-conjunto de C.

Demostración.

(i) Es una consecuencia inmediata de la definición de síndrome (Definición 2.3.2.2.1).
(ii) Por la definición de síndrome. Tenemos que S(**u**) = 0 si y solo si $\boldsymbol{u}H^T = 0$.
(iii) Se sigue de (i), (ii) y del teorema 2.3.2.0.2 (vi).

Observación 2.3.2.2.3 Puesto que los síndromes se encuentran en F_q^{n-k}, existen a lo más q^{n-k} síndromes. El Teorema 2.3.2.0.2 (v) dice que hay q^{n-k} co-conjuntos, por lo que hay q^{n-k} correspondientes síndromes, todos distintos entre si. Por lo tanto, todos los vectores en F_q^{n-k} aparecen como síndromes.

Definición 2.3.2.2.4 La tabla que asocia cada co-conjunto líder con su síndrome se llama *tabla de síndromes* o *arreglo de decodificación estándar*.

Pasos para construir una tabla de síndromes asumiendo completa decodificación por el método del vecino más cercano.

Paso 1: Listar todos los co-conjunto para el código y elija para cada co-conjunto la palabra código de menor peso como el co-conjunto líder **u**.

Paso 2: Encontrar una matriz parity check H para el código y para cada co-conjunto líder **u**, calcular su síndrome $S(\mathbf{u}) = \mathbf{u}H^T$.

Para decodificación incompleta por el método del vecino más cercano, si encontramos más de una palabra de menor e igual peso en el co-conjunto, se solicita una retransmisión y se indica esto en la *tabla de síndromes* por medio de un guión – o $*$.

Una forma rápida de construir una tabla de síndrome, dada la matriz parity check H y la distancia del código C, es generar todos los patrones de error **e**, con

$$wt(e) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$$

como co-conjuntos líderes y calcular los síndromes $S(\mathbf{e})$ para cada uno de ellos.

Ejemplo 2.3.2.2.5 Para una completa decodificación por el método del vecino más cercano. Construir su tabla síndrome para el siguiente código lineal.

$$C = \{0000, 1011, 0101, 1110\}$$

Paso 1.- Elegimos las palabras código 0000, 0001, 0010 y 1000 como co-conjuntos líderes. Después, la matriz parity check para C es

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Paso 2.- Construimos la tabla síndrome para C. Con $S(u) = uH^T$.

Co-conjunto líder u	Síndrome $S_H(u) = u\,H^T$
0000	00
0001	01
0010	10
1000	11

Tabla 2.3 de síndromes.

Procedimiento para la decodificación por síndromes.

Paso 1: Para la palabra recibida **w**, calcule el síndrome S(**w**).

Paso 2: Encuentre el co-conjunto líder **u** próximo al síndrome S(**w**) = S(**u**) en la tabla de síndromes.

Paso 3: Decodifique **w** como **v** = **w** – **u**.

Ejemplo 2.3.2.2.6 Sea q=2 y C = {0000, 1011, 0101, 1110}. Sea **w**=1101 la palabra código transmitida. Calculamos el síndrome S(**w**) = $\mathbf{w}H^T$ =11. De la tabla 2.3 el próximo co-conjunto líder es 1000. Por lo tanto, 1101+1000 = **0101**. Este ejemplo puede ser verificado con ayuda del ejemplo 2.3.2.2.5.

Ejemplo 2.3.2.2.7 Sea C = {0000000, 1011100, 0101110, 0010111, 1001011, 1100101, 1110010, 0111001} un código lineal binario y **w** = 1110011 la palabra código transmitida. Entonces H es la matriz parity check

$$H = \begin{pmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{pmatrix}$$

Elegimos las palabras código que serán los co-conjuntos líderes y calculamos S(**u**) = $\mathbf{u}H^T$. Entonces

$$uH^T = \begin{pmatrix} 1000000 \\ 0100000 \\ 0010000 \\ 0001000 \\ 0000100 \\ 0000010 \\ 0000001 \end{pmatrix} * \begin{pmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \\ 1011 \\ 1110 \\ 0111 \end{pmatrix} = \begin{pmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \\ 1011 \\ 1110 \\ 0111 \end{pmatrix}$$

Elementos líder **u**	Síndrome S_H (u) = **u** H^T
0000000	0000
1000000	1000
0100000	0100
0010000	0010
0001000	0001
0000100	1011
0000010	1110
0000001	0111

Tabla 2.4 de síndromes.

Calculamos el síndrome S(**w**) = $\boldsymbol{w}H^T$ = 0111. De la tabla 2.4 el próximo co-conjunto líder es 0000001. Por lo tanto, 1110011 + 0000001 = **1110010** el cual pertenece a una palabra código de C. La decodificación puede ser corroborada con la siguiente tabla.

0+C	**0000000**	**1011100**	**1110010**	**1100101**	**0101110**	**0111001**	**0010111**	**1001011**
1000000+C	1000000	0011100	0110010	0100101	1101110	1111001	1010111	1001011
100000+C	0100000	1111100	1010010	1000101	0001110	0011001	0110111	1101011
10000+C	0010000	1001100	1100010	1110101	0111110	0101001	0000111	1011011
1000+C	0001000	1010100	1111010	1101101	0100110	0110001	0011111	1000011
100+C	0000100	1011000	1110110	1100001	0101010	0111101	0010011	1001111
10+C	0000010	1011110	1110000	1100111	0101100	0111011	0010101	1001001
1+C	0000001	1011101	**1110011**	1100100	0101111	0111000	0010110	1001010

Tabla 2.5 Arreglo en forma estándar para un código [7, 3, 4].

2.4 CONSTRUCCIÓN DE ALGUNOS DE LOS CÓDIGOS LINEALES.

En este apartado, analizaremos las reglas de propagación que describe la construcción de nuevos códigos basados en viejos códigos. La estrategia es construir códigos de gran tamaño o larga longitud a partir de códigos de menor tamaño o corta longitud. Desde el comienzo de la teoría de códigos, muchas reglas de propagación han sido propuestas y algunas han llegado a ser un importante estándar de construcción en teoría de códigos. Además, analizaremos la construcción de ciertos tipos de códigos de Hamming y de Reed muller.

2.4.1 Reglas de propagación.

Teorema 2.4.1.1 Supóngase que hay un código lineal *[n, k, d]* sobre $\mathbf{F}_q$. entonces:
(i) (longitud) existe un código lineal *[n + r, k, d]* sobre $\mathbf{F}_q$ para cualquier $r \geq 1$;
(ii) (subcódigos) existe un código lineal *[n, k - r, d]* sobre $\mathbf{F}_q$ para cualquier $1 \leq r \leq k-1$;
(iii) Existe un código lineal *[n – r, k, d - r]* sobre $\mathbf{F}_q$ para cualquier $1 \leq r \leq d-1$;
(iv) Existe un código lineal *[n, k, d - r]* sobre $\mathbf{F}_q$ para cualquier $1 \leq r \leq d-1$;
(v) Existe un código lineal *[n - r, k –r, d]* sobre $\mathbf{F}_q$ para cualquier $1 \leq r \leq k-1$.

Demostración. Sea *C* un código lineal *[n, k, d]* sobre $\mathbf{F}_q$.

(i) Para demostrar la existencia de un código lineal *[n+1, k, d]* sobre $\mathbf{F}_q$. Consideramos sumar una nueva coordenada 0 a todas las palabras código de C para formar un nuevo código

$$\{ (u_1,..., u_n, 0) : (u_1,..., u_n) \in C \}.$$

El código representa a un código lineal [*n + 1, k, d*] sobre $\mathbf{F}_q$.

(ii) Sea **c** una palabra código diferente de cero de C con wt(**c**) = *d*. Si consideramos la extensión de **c** para formar una base de *C:* $\{\boldsymbol{c}_1 = \boldsymbol{c},..., \boldsymbol{c}_k\}$. Considerando el nuevo código $< \{\boldsymbol{c}_1,..., \boldsymbol{c}_{k-r}\} >$ transformado por las primeras $k - r$ palabras código en la base. Por lo tanto, el nuevo código tiene los parámetros *[n, k - r, d]*.

(iii) Sea **c** $\in$ *C* una palabra código de peso d. Para cada palabra código de C, consideramos borrar el conjunto de *r* posiciones donde **c** tiene coordenadas diferentes de cero. Podemos observar que el nuevo código es un código lineal *[n - r, k, d - r]*.

(iv) El resultado es una consecuencia de (i) y (iii).

(v) Si $k = n$, entonces tenemos que $d = 1$. Por lo tanto, el espacio $\boldsymbol{F}_q^{n-r}$ es un código con parámetros *[n - r, k – r, d]*.

Consideramos que $k < n$. Para demostrarlo es suficiente demostrar la existencia de un código lineal *[n -1, k -1, d]* para $k \geq 2$. Sea C un código lineal *[n, k, d]* sobre $\mathbf{F}_q$. Asumimos que C tiene una matriz parity-check de la forma $H = (I_{n-k} \mid X)$. Si borramos la última columna de *H*, obtenemos la matriz H_1 con *(n-k)* x *(n-1)*. Entonces, los renglones de H_1 son linealmente independientes y cualquier *d-1* columnas de H_1 son linealmente independientes. De hecho, el código lineal con matriz parity-check H_1 tiene parámetros $[n -1, k - 1, d_1]$ con $d_1 \geq d$. Por lo tanto, podemos obtener de (iv), un código lineal *[n -1, k -1, d]*.

El teorema 2.4.1.1 produce códigos con parámetros no tan buenos como los códigos que los crearon. En la práctica, generalmente no se hacen nuevos códigos a partir de estas construcciones. Sin embargo, son muy útiles para el estudio y diseño de los códigos lineales. A continuación daremos algunos ejemplos de construcción de los códigos lineales y damos por entendido que su decodificación es la misma presentada en el apartado anterior.

Ejemplo 2.4.1.2 Un código Hamming binario de longitud 7 es un código lineal [7, 4, 3]. A partir del teorema 2.4.1.1, podemos obtener códigos lineales con parámetros [n, 4, 3] para $n \geq 7$ y también códigos lineales con parámetros [7, k, 3] para cualquier $1 \leq k \leq 4$.

Definición 2.4.1.3 Encontraremos nuevos códigos lineales por suma directa. Sea C_i un código lineal $[n_i, k_i, d_i]$ sobre $\mathbf{F}_q$ para i = 1, 2. Entonces la suma directa de C_1 y C_2 definido por

$$C_1 + C_2 = \{(\mathbf{c}_1, \mathbf{c}_2 : \mathbf{c}_1 \in C_1, \mathbf{c}_2 \in C_2)\}$$

es un código lineal $[n_1 + n_2, k_1 + k_2 , \min \{d_1, d_2\}]$ sobre $\mathbf{F}_q$.

Ejemplo 2.4.1.4 Sea $C_1 = \{000, 110, 101, 011\}$ un código lineal [3, 2, 2] binario, y sea $C_2 = \{0000, 1111\}$ un código lineal [4, 1, 4] binario. Entonces:

$$C_1 + C_2 = \{0000000, 1100000, 1010000, 0110000, 0001111, 1101111, 1011111, 0111111\}$$

es un código lineal [7, 3, 2] binario.

Definición 2.4.1.5 Códigos lineales con construcción (**u**, **u**+**v**). Sea C_i un código lineal $[n, k_i, d_i]$ sobre $\mathbf{F}_q$, para i = 1, 2. Entonces el código C definido por:

$$C = \{(\mathbf{u}, \mathbf{u} + \mathbf{v}) : \mathbf{u} \in C_1, \mathbf{v} \in C_2)\}$$

es un código lineal $[2n, k_1 + k_2 , \min \{2d_1, d_2\}]$ sobre $\mathbf{F}_q$.

Ejemplo 2.4.1.6 Sea $C_1 = \{000, 110, 101, 011\}$ un código lineal [3, 2, 2] binario, y sea $C_2 = \{000, 111\}$ un código lineal [3, 1, 3] binario. Entonces:

$$C_1 + C_2 = \{000000, 110110, 101101, 011011, 000111, 110001, 101010, 011100\}$$

es un código lineal [6, 3, 3] binario.

Definición 2.4.1.7 Sea A un código lineal [n, k, d]. Entonces el código C definido por $C = \{(\mathbf{c}, \mathbf{c}): \mathbf{c} \in A\} \cup \{(\mathbf{c}, 1 + \mathbf{c}): \mathbf{c} \in A)\}$ es un código lineal $[2n, k + 1, \min \{n, 2d\}]$ binario.

Ejemplo 2.4.1.8 Sea $A = \{00, 01, 10, 11\}$ un código lineal [2, 2, 1] binario. Entonces, de la definición 2.4.1.7 tenemos que

$$C = \{0000, 0101, 1010, 1111, 0011, 0110, 1001, 1100\}.$$

Por lo tanto, C es un código lineal [4, 3, 2] binario.

2.4.2 Códigos de Hamming.

Códigos de Hamming fueron descubiertos por R. W. Hamming y M. J. E. Golay. Forman una importante clase de códigos con interesantes propiedades y son fáciles de codificar y decodificar.

Definición 2.4.2.1 Sea $r \geq 2$. Un código lineal binario de longitud $n = 2^r-1$, con matriz parity check H y sus columnas consisten de todos los vectores diferentes de cero de $\boldsymbol{F}_2^r$, es llamado un código de Hamming binario de longitud 2^r-1. Se denota por Ham(r, 2).

Definición 2.4.2.2 El dual de un código binario de Hamming Ham(r, 2) es llamado un código simplex binario. Se denota por S(r, 2).

Definición 2.4.2.3 Propiedades de los códigos Hamming binarios.

(i) Todos los códigos Hamming binarios de longitud dada son equivalentes.
(ii) La dimensión de Ham(r, 2) es $k = 2^r - 1 - r$.
(iii) La distancia de Ham(r, 2) es d=3. Por lo tanto, Ham(r, 2) corrige únicamente un solo error.
(iv) Códigos Hamming binarios son códigos perfectos [ver Apéndice (A)].

Demostración.

(i) Para una longitud dada, cualquier matriz parity check puede ser obtenida de otra por realizar operaciones de columnas. Por lo tanto, el correspondiente código de Hamming es equivalente.
(ii) La matriz parity check H de Ham(r, 2), es una matriz $r \times (2^r-1)$. Por lo tanto, la dimensión de Ham(r, 2) es $2^r - 1 - r$.
(iii) Como ninguna de 2 columnas cualesquiera de H son iguales, entonces son linealmente independientes. Se dice, por el Corolario 2.2.4.6 que la distancia de Ham(r, 2) es igual a 3.
(iv) Verificamos que Ham(r, 2) satisface el límite de Hamming explicado en el Apéndice (A). Por lo tanto, se concluye que Ham(r, 2) es un código perfecto.

Ejemplo 2.4.2.4 El código Ham (3, 2) binario, tiene [7, 4, 3] como parámetros y corrige un error. El número de vectores en $\boldsymbol{F}_2^r$ que son garantizados a ser únicamente corregidos a un código de palabra es $16 \times (\binom{7}{0} + \binom{7}{1}) = 128$ y como $2^7=128$ vectores, es garantía de que cada vector en $\boldsymbol{F}_2^r$ sea únicamente corregible a una palabra código en Ham(3, 2), y por lo tanto el código es perfecto.

Definición 2.4.1.5 El código Hamming extendido binario, denotado por $\overline{\text{Ham}(r, 2)}$, es el código obtenido por sumar una coordenada al código Ham(r, 2) como se demostró en el teorema 2.4.1.1 (i).

Definición 2.4.1.6 Se denota Ham(r, q) a un q-ario código de Hamming con $r \geq 2$. Teniendo las siguientes propiedades:

(i) Ham(r, q) es un código con parámetros $[(q^r-1)/(q-1), ((q^r-1)/(q-1))-r, 3]$.
(ii) Ham(r, q) corrige exactamente un error.

Ejemplo 2.4.2.7 La matriz de un código ternario Ham (2, 3) es

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix}$$

con parámetros [4, 2, 3].

2.4.3 Códigos de Reed Muller.

Los códigos Reed Muller están entre los códigos más antiguos que son conocidos y tienen muchas aplicaciones prácticas. Para cada entero positivo **m** y **r** con $0 \leq \mathbf{r} \leq \mathbf{m}$, donde **r** es el orden del código Reed Muller denotado por $R(\mathbf{r}, \mathbf{m})$, el cuál es un código lineal binario con parámetros

$$\left[2^m, \quad \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{r}, \quad 2^{m-r} \right].$$

En efecto, $R(1, 5)$ fue usado por el Mariner 9 para transmitir imágenes blanco y negro desde el planeta Marte hacia la Tierra en 1972 [48; 118]. Los códigos Reed Muller admiten una especial decodificación llamada decodificación Reed.

Definición 2.4.3.1 El código Reed Muller de primer orden $R(1, m)$ son códigos binarios, definidos para toda $m \geq 1$, recursivamente como sigue:

(i) $R(1, 1) = F_2^2 = \{00, 01, 10, 11\}$.
(ii) *Para* $m \geq 1$.

$R(1, m+1) = \{(\mathbf{u}, \mathbf{u}) : \mathbf{u} \in R(1, m)\} \cup \{(\mathbf{u}, \mathbf{u}+1) : \mathbf{u} \in R(1, m)\}$.

Ejemplo 2.4.3.2 $R(1, 2) = \{0000, 0101, 1010, 1111, 0011, 0110, 1001, 1100\}$.
La matriz generadora de $R(1, 2)$ es:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Observar que el código $R(1, 2)$ es lineal y binario, pero no cíclico, los cuales analizaremos en el siguiente capítulo.

Proposición 2.4.3.3 Para $m \geq 1$, el código Reed Muller $R(1, m)$ es un código lineal binario con parámetros $[2^m, m+1, 2^{m-1}]$. En la cuál cada palabra código excepto el **0** y **1** tiene peso 2^{m-1}.

Definición 2.4.3.4 Para cualquier $r \geq 2$, el código Reed Muller $R(r, m)$ es considerado de orden r y longitud 2^m, para $m \geq r - 1$, definido recursivamente por:

$$R(r, m+1) = \begin{cases} F_2^{2^r} & \text{Si } m = r - 1, \\ \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in R(r, m), \ \mathbf{v} \in R(r-1, m)\} & \text{Si } m > r - 1. \end{cases}$$

Ejemplo 2.4.3.5 El código $R(1, 3)$ tiene longitud $2^3 = 8$. Su construcción será a partir de los vectores siguientes:

$$\begin{aligned} 1 \leftrightarrow 1 &= (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1) \\ v_3 \leftrightarrow v_3 &= (0\ 0\ 0\ 0\ 1\ 1\ 1\ 1) \\ v_2 \leftrightarrow v_2 &= (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1) \\ v_1 \leftrightarrow v_1 &= (0\ 1\ 0\ 1\ 0\ 1\ 0\ 1) \end{aligned}$$

Su construcción se asemeja a las características de una función booleana en el campo F_2.

Su matriz generadora es: $G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$

La matriz H_8 es una analogía de lo que conocemos como matriz parity check, tan solo por realizar la transformación Hadamard[16]. La columna 2 es v_1 (transformación Hadamard), la columna 3 es v_2 y la columna 5 es v_3.

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Algoritmo 2.1 Decodificación para códigos R (1, m)

Entrada: $r = (r_0, r_1, ..., r_2{}^m{}_{-1})$.
Salida: Máxima palabra código parecido **c**.
Encontrar la representación **R** = (r).
Calcular la transformación Hadamard $T = \mathbf{R}H_2{}^m$.
Encontrar la coordenada t_i con la magnitud más larga.
Sea i la expansión binaria $(i_m, i_{m-1}, \ldots, i_1)_2$

Si $(t_i > 0)$, entonces: $c = \sum_{j=1}^{m} i_j v_j$

Si no, entonces: $c = 1 + \sum_{j=1}^{m} i_j v_j$
FIN

Para r = [1, 0, 0, 1, 0, 0, 1, 0]
Pasos: 1.- Calculamos la transformación **R** = [-1, 1, 1, -1, 1, 1, -1, 1].
2.- Calcular **T**=**RH** = [2, -2, 2, -2, -2, 2, -2, -6].
3.- La máxima coordenada es: $t_7 = -6$, por lo tanto $(1, 1, 1)_2$.
4.- Como $t_7 < 0$, **c** = $1 + v_1 + v_2 + v_3$ = [1, 0, 0, 1, 0, 1, 1, 0].

[16] La transformación Hadamard convierte elementos binarios {0, 1} de r a binarios (+1 y -1) utilizada en decodificación de códigos Reed Muller ver [53, 381].

CAPÍTULO 3

CÓDIGOS CÍCLICOS

3. CÓDIGOS CÍCLICOS.

Como se explicó en el capítulo anterior, los códigos lineales pueden ser corregidos utilizando el arreglo estándar, aunque para códigos de gran tamaño el almacenamiento y el tiempo de computación pueden ser factores limitantes para que un código lineal sea óptimo. Además, aún no se ha visto ningún mecanismo para el cual la matriz generadora o la matriz parity check puedan ser diseñadas para una específica distancia mínima. En este capítulo se desarrollarán los códigos cíclicos, los cuales están basados en operaciones polinómicas y tienen una estructura algebraica con la cual se realiza la codificación y decodificación más eficiente.

Los primeros estudios sobre códigos cíclicos fueron realizados por Prange en 1957 [48, 133]. Desde entonces los códigos correctores de errores pasaron a un enfoque más algebraico y la teoría de códigos algebraica ha hecho grandes progresos en el estudio de códigos de corrección de errores que utilizan redundancia, surgiendo entonces muchos ejemplos prácticos que están entre los códigos cíclicos, como algunos de los códigos de Hamming, códigos de Golay, códigos BCH (Bose-Chaudhuri-Hocquenghen), o los códigos de Reed-Solomon y Goppa.

Se demostrará que un código cíclico es totalmente determinado por su polinomio generador y que la estructura algebraica para las operaciones en polinomios es la estructura algebraica de un anillo.

3.1 DEFINICIONES BÁSICAS.

3.1.1 Breve resumen de campos finitos y anillos.

La teoría de los campos finitos tiene sus orígenes en los siglos XVII y XVIII, con eminentes matemáticos tales como Pierre de Fermat (1601-1665) y Leonhard Euler (1707-1783) que contribuyeron a la estructura teórica de especiales campos finitos. La teoría general de campos finitos comienza con el trabajo de Carl Friedrich Gauss (1777-1855) y Evariste Galois (1811-1832), pero únicamente llega a ser de interés para matemáticos e ingenieros en recientes décadas, esto es debido a sus muchas aplicaciones a las matemáticas, ciencia computacional y teoría de la comunicación. Actualmente, la teoría de los campos finitos conjuntamente con la definición de un anillo ha llegado a ser muy importante en el uso de los códigos correctores de errores. En este capítulo se presenta un breve resumen de esta teoría. Para una completa introducción a los campos finitos se invita al lector a consultar [11].

Definición 3.1.1.1 Un campo es un conjunto no vacío F de elementos con dos operaciones "+" llamado suma y "·" llamado multiplicación los cuales satisfacen los siguientes axiomas. Para todo $a, b, c \in F$:

(i) F es cerrado bajo "+" y "·"; i.e., $a + b$ y $a \cdot b$ están en F.
(ii) Se cumple la ley conmutativa: $a + b = b + a$, $a \cdot b = b \cdot a$.
(iii) Se cumple la ley asociativa: $(a + b) + c = a + (b + c)$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
(iv) Se cumple la ley distributiva: $a \cdot (b + c) = a \cdot b + a \cdot c$.

Además, F contiene dos distintos elementos identidad que son el 0 y 1, donde el 0 es identidad de suma y el 1 es identidad multiplicativa y satisfacen los siguientes axiomas:

(v) $a + 0 = a$ para todo $a \in$ F.
(vi) $a \cdot 1 = a$ y $a \cdot 0 = 0$ para todo $a \in$ F.
(vii) Para cualquier a en F, existe un elemento inverso aditivo $(-a)$ en F tal que $a + (-a) = 0$.
(viii) Para cualquier $a \neq 0$ en F, existe un elemento inverso multiplicativo a^{-1} en F tal que $a \cdot a^{-1} = 1$.

Ejemplo 3.1.1.2 (i) Algunos campos conocidos, son el campo de los racionales **Q**,

$$Q := \left\{ \frac{a}{b} : a, b \text{ son enteros con } b \neq 0 \right\},$$

el campo de los reales **R** y el campo de los complejos **C**. Se puede comprobar que todos los axiomas en la definición 3.1.1.1 se satisfacen para los tres campos. De hecho, no estamos interesados en estos campos porque tienen un número infinito de elementos.

(ii) Denotamos por $\mathbf{Z}_2$ al conjunto {0, 1}. Definimos la suma y la multiplicación como en la figura 3.1. Después es fácil comprobar que $\mathbf{Z}_2$ es un campo y tiene únicamente dos elementos.

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

FIGURA 3.1 TABLA DE SUMA Y MULTIPLICACIÓN PARA Z_2.

Lema 3.1.1.3 Sea a, b cualquier elemento de un campo F. Entonces:

(i) $(-1) \cdot a = -a$;
(ii) $a \cdot b = 0$ implica que $a = 0$ o $b = 0$.

Demostración. (i) tenemos que

$(-1) \cdot a + a = (-1) \cdot a + a \cdot 1$ del axioma (vi),
$(-1) \cdot a + a \cdot 1 = ((-1) + 1) \cdot a$ del axioma (ii) y (iv),
$((-1) + 1) \cdot a = 0 \cdot a = 0$ del axioma (vii), (ii) y (vi).

(ii) Si $a \neq 0$, entonces:

$$\begin{array}{rcll} 0 & = & a^{-1}\cdot 0 & \\ a^{-1}\cdot 0 & = & a^{-1}\cdot (a\cdot b) & \text{del axioma (vi),} \\ a^{-1}\cdot (a\cdot b) & = & (a^{-1}\cdot a)\cdot b & \text{del axioma (iii),} \\ (a^{-1}\cdot a)\cdot b & = & 1\cdot b & \text{del axioma (ii) y (viii),} \\ 1\cdot b & = & b\cdot 1 = b & \text{del axioma (ii) y (vi).} \end{array}$$

Un campo que contiene únicamente muchos elementos finitos es llamado un campo finito. Un conjunto F que satisface los axiomas (i) al (vii) en la definición 3.1.1.1 es llamado un anillo conmutativo.

Ejemplo 3.1.1.4
(i) El conjunto de todos los enteros

$$Z := \{\ 0,\ \pm 1,\ \pm 2,\ ...\}$$

forman un anillo para la suma y la multiplicación, llamado el anillo de los enteros.
(ii) El conjunto de todos los polinomios sobre un campo F,

$$F[x] := \{\ a_0 + a_1 x + \cdots + a_n x^n : a_i \in F, n \geq 0\},$$

forman un anillo para la suma y multiplicación de polinomios.

Definición 3.1.1.5 Sea a, b y $m > 1$ enteros. Decimos que a es congruente a b módulo m, escrito como $a \equiv b \pmod m$, si $m \mid (a - b)$; i.e., m divide $a - b$.

Ejemplo 3.1.1.6
(i) $90 \equiv 30 \pmod{60}$ y $15 \equiv 3 \pmod{12}$.
(ii) $a \equiv 0 \pmod m$ se refiere a que $m \mid a$.
(iii) $a \equiv 0 \pmod 2$ se refiere a que a es par.
(iv) $a \equiv 1 \pmod 2$ se refiere a que a es impar.

Observación 3.1.1.7 Dado los enteros a y $m > 1$, por el algoritmo de la división se tiene que: $a = mq + b$, donde b es determinado únicamente por a y m, y $0 \leq b \leq m - 1$. Por lo tanto, cualquier entero a es congruente a exactamente uno de los 0, 1,…, $m - 1$ módulo m. El entero b (como se observa en la figura 3.1) es llamado el residuo principal de a dividido por m, denotado por (a (módulo m)).
Si $a \equiv b \pmod m$ y $c \equiv d \pmod m$, entonces nosotros tenemos

$$\begin{array}{l} a + c \equiv b + d \pmod m, \\ a - c \equiv b - d \pmod m, \\ a * c \equiv b * d \pmod m. \end{array}$$

Para un entero $m > 1$, denotamos por $\mathbf{Z}_m$ o $\mathbf{Z}/(m)$ al conjunto $\{0, 1,\ldots, m - 1\}$ y definimos la suma "+" y la multiplicación "$*$" en $\mathbf{Z}_m$ por:

$a + b$ = al residuo de $a + b$ dividido por m, i.e., ($a + b \pmod m$), y
$a * b$ = al residuo de $a\cdot b$ dividido por m, i.e., ($a \cdot b \pmod m$).

Es fácil mostrar que los axiomas (i) al (vii) de la definición 3.1.1.1 se satisfacen. Por lo tanto, $\mathbf{Z}_m$, junto con la suma "+" y la multiplicación "$*$", forman un anillo.

Ejemplo 3.1.1.8

(i) Módulo 2: En la figura 3.1, $\mathbf{Z}_2$ es exactamente un anillo definido para $m = 2$. En este caso, el axioma (viii) también se satisface. Por lo tanto, $\mathbf{Z}_2$ es un campo.

(ii) Módulo 4: En la figura 3.2 se construyeron las tablas de suma y multiplicación para $\mathbf{Z}_4$. Podemos ver que el axioma (viii) no se cumple para $\mathbf{Z}_4$ y como 2^{-1} no existe. Por lo tanto, $\mathbf{Z}_4$ no es un campo.

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

FIGURA 3.2 TABLAS DE SUMA Y MULTIPLICACIÓN PARA Z_4.

Conclusión: Nosotros encontramos de los ejemplos anteriores que $\mathbf{Z}_m$ es un campo para algunos enteros m y es precisamente un anillo para otros enteros.

Teorema 3.1.1.9 $\mathbf{Z}_m$ es un campo si y solo si m es un primo.

Demostración. Supóngase que m es un número compuesto y sea $m = a \cdot b$ para dos enteros $1 < a, b < m$. Por lo tanto, $a \neq 0$, $b \neq 0$. Sin embargo, $0 = m = a \cdot b$ en $\mathbf{Z}_m$. Esto es una contradicción al lema 3.1.1.3. Por lo tanto, $\mathbf{Z}_m$ no es un campo.

Ahora sea m un número primo. Para cualquier elemento diferente de cero $a \in \mathbf{Z}_m$, i.e., $0 < a < m$, nosotros conocemos que a es primo a m. Por lo tanto, existen dos enteros u, v con $0 \leq u \leq m - 1$ tales que $ua + vm = 1$, i.e., $ua \equiv 1 \pmod{m}$. Por lo tanto, $u = a^{-1}$. Esto implica que el axioma (viii) en la definición 3.1.1.1 también se satisface. Por lo tanto, $\mathbf{Z}_m$ es un campo.

Para un anillo $\mathbf{R}$ un entero $n \geq 1$ y $a \in \mathbf{R}$, denotamos por $n \cdot a$ el elemento

$$\sum_{i=1}^{n} a = \underbrace{a + a + \cdots + a}_{n}.$$

Definición 3.1.1.10 Sea F un campo. La característica de F es el menor entero positivo p tal que $p \cdot 1 = 0$, donde 1 es la identidad multiplicativa de F. Si p no existiera, se define la característica como 0.

Ejemplo 3.1.1.11

(i) Las características de **Q, R, C** son 0.
(ii) Las características del campo $\mathbf{Z}_p$ es p para cualquier valor primo p.

Conclusión: La característica de un campo es 0 o un número primo.

3.1.2 Anillos polinomiales.

Definición 3.1.2.1 Sea F un campo. El conjunto

$$F[x] := \left\{ \sum_{i=0}^{n} a_i x^i : a_i \in F, n \geq 0 \right\}$$

es llamado el anillo polinomial sobre F. (Si F es un anillo, los axiomas (i) al (viii) de la definición 3.1.1.1 se cumplen). Un elemento de F [x] se le llama polinomio sobre F.

Para un polinomio $f(x) = \sum_{i=0}^{n} a_i x^i$,

el entero n es llamado el grado de f(x), denotado por grad(f(x)). Además, un polinomio diferente de cero de grado n es dicho ser mónico si $a_n = 1$. Un polinomio f(x) de grado positivo se dice ser reducible sobre F si existen dos polinomios g(x) y h(x) sobre F tales que grad(g(x)) < grad(f(x)), grad (h(x)) < grad (f(x)) y f (x) = g(x)h(x). De otra manera, el polinomio F(x) de grado positivo es dicho ser irreducible sobre F.

Ejemplo 3.1.2.2 (i) El polinomio $f(x) = x^4 + 2x^6 \in \mathbf{Z}_3[x]$ es de grado 6. Es reducible debido a que $f(x) = x^4(1 + 2x^2)$.
(ii) El polinomio $g(x) = 1 + x + x^2 \in \mathbf{Z}_2[x]$ es de grado 2. Es irreducible. De otra manera, g(x) tendría un factor lineal x o x+1; i.e., 0 o 1 podrían ser una raíz de g(x), pero $g(0) = g(1) = 1 \in \mathbf{Z}_2$.

Definición 3.1.2.3 Sea $f(x) \in F[x]$ un polinomio de grado $n \geq 1$. Entonces, para cualquier polinomio de $g(x) \in F[x]$, existe un único par (s(x), r(x)) de polinomios con grad(r(x)) < grad(f(x)) o r(x) = 0 tal que g(x) = s(x)f(x) + r(x). El polinomio r (x) es llamado el residuo principal de g(x) dividido por f(x), denotado por (g(x) (mod f (x))).

Ejemplo 3.1.2.4 Sea $f(x) = 1 + x^2$ y $g(x) = x + 2x^4$ dos polinomios en $\mathbf{Z}_5[x]$. Tenemos que $g(x) = x + 2x^4 = (3 + 2x^2)(1 + x^2) + (2 + x) = (3 + 2x^2)\, f(x) + (2 + x)$, el residuo de g(x) dividido por f(x) es 2 + x.

Definición 3.1.2.5 Sea $f(x), g(x) \in F[x]$ dos polinomios diferentes de cero. El *más grande* o *mayor común divisor* de f(x) y g(x), denotado por mcd(f(x), g(x)), es el polinomio mónico de más alto grado el cual es divisor de ambos f(x) y g(x). En particular, decimos que f(x) es co-primo o primo a g(x) si mcd(f(x), g(x))=1. El *menor común múltiplo* de f(x) y g(x), denotado por mcm(f(x), g(x)), es el polinomio mónico de más bajo grado cual es múltiplo de ambos f(x) y g(x).

Observación 3.1.2.6 Si f(x) y g(x) tienen las siguientes factorizaciones:

$$f(x) = a \cdot p_1(x)^{e_1} \cdots p_n(x)^{e_n}, \quad g(x) = b \cdot p_1(x)^{d_1} \cdots p_n(x)^{d_n},$$

donde a, b $\in$ F, e_i, $d_i \geq 0$ y $p_i(x)$ son distintos polinomios mónicos irreducibles, entonces:

$$mcd\ (f(x), g(x)) = p_i(x)^{\min\{e_i, d_i\}} \cdots p_n(x)^{\min\{e_n, d_n\}}$$

y

$$mcm\ (f(x), g(x)) = p_i(x)^{máx\{e_i, d_i\}} \cdots p_n(x)^{máx\{e_n, d_n\}}.$$

Ejemplo 3.1.2.7 Si tenemos el siguiente polinomio binario $f_1(x) = (1+x)^2(1+x+x^4)^3$, $f_2(x)=(1+x)(1+x+x^2)^2$, $f_3(x) = x^2(1+x+x^4)$. Entonces, por la observación 3.1.2.6 tenemos:

$$\text{mcm}\,(f_1(x), f_2(x), f_3(x)) = x^2(1+x)^2(1+x+x^2)^2(1+x+x^4)^3.$$

A continuación presentaremos la tabla 3.1 que representa las analogías entre anillo de enteros **Z** y anillo polinomial F[x].

El anillo de los enteros Z	El anillo polinomial F[x]
Un entero m	Un polinomio f(x)
Un número primo p	Un polinomio irreducible p(x)
$Z_m = \{0,1,..., m-1\}$	$F[x]/(f(x)): \left\{ \sum_{1=0}^{n-1} a_i x^i : a_i \in F, n \geq 1 \right\}$
$a + b := (a + b(\text{mod } m))$	$g(x) + h(x) := (g(x) + h(x)(\text{mod } f(x)))$
$a * b := (ab(\text{mod } m))$	$g(x) * h(x) := (g(x)h(x)(\text{mod } f(x)))$
Z_m *es un anillo*	$F[x]/(f(x))$ *es un anillo*
Z_m *es un campo* $\Leftrightarrow$ *m es primo*.	$F[x]/(f(x))$ *es campo* $\Leftrightarrow$ $f(x)$ *es irreducibl e.*

TABLA 3.1 ANALOGÍAS ENTRE Z Y F[x].

Teorema 3.1.2.8 Sea f(x) un polinomio sobre el campo F de grado ≥ 1. Entonces F[x]/(f(x)), de acuerdo a la tabla 3.1 para la suma y multiplicación se forma un anillo. Además, F[x]/(f(x)) es un campo si y solo si f(x) es irreducible.

Demostración. Para verificar que F[x]/(f(x)) es un anillo, se aplica exactamente el mismo argumento que se aplicó para demostrar el Teorema 3.1.1.9.

+	0	1	x	1 + x
0	0	1	x	1 + x
1	1	0	1+x	x
x	x	1 + x	0	1
1 + x	1 + x	x	1	0

*	0	1	x	1 + x
0	0	0	0	0
1	0	1	x	1 + x
x	0	x	1	1 + x
1 + x	0	1 + x	1 + x	0

TABLA 3.2 SUMA Y MULTIPLICACIÓN PARA $Z_2[x]/(1+x^2)$.

Ejemplo 3.1.2.9 Si tenemos el anillo de la tabla 3.2 $Z_2[x] / (1 + x^2) = \{0, 1, x, 1+x\}$. Podemos observar de la tabla de multiplicación 3.2 que $Z_2[x] / (1 + x^2)$ no es un campo debido a que $(1 + x)(1+ x) = 0$.

+	0	1	x	1 + x
0	0	1	x	1 + x
1	1	0	1 + x	x
x	x	1 + x	0	1
1 + x	1 + x	x	1	0

*	0	1	x	1 + x
0	0	0	0	0
1	0	1	x	1 + x
x	0	x	1 + x	1
1 + x	0	1 + x	1	x

TABLA 3.3 SUMA Y MULTIPLICACIÓN PARA $Z_2[x]/(1+x+x^2)$.

Ejemplo 3.1.2.10 Si tenemos el anillo de la tabla 3.3 $Z_2[x] / (1+x+x^2) = \{0, 1, x, 1+x\}$. Podemos observar de la tabla de multiplicación 3.3 que $Z_2[x] / (1+x+x^2)$ si es un campo.

3.1.3 Definición de códigos cíclicos.

Dado un vector c = (c_0, c_1,…, c_{n-2}, c_{n-1}) $\in F_q^n$, y el vector

$$\mathbf{c}' = (c_{n-1}, c_0, c_1, \ldots, c_{n-2}).$$

Entonces, se dice que **c´** es la forma cíclica de c hacia la derecha. Un desplazamiento de r lugares a la derecha produce el siguiente vector (c_{n-r}, c_{n-r+1},…, c_{n-1}, c_0, c_1,…, c_{n-r-1}).

Definición 3.1.3.1 Un (n, k) código de bloque C es dicho ser **cíclico** si es lineal y si para cada palabra código c = (c_0, c_1,…, c_{n-1}) en C, su desplazamiento cíclico hacia la derecha c´= (c_{n-1}, c_0, c_1,…, c_{n-2}) pertenece también a C.

Para convertir la estructura combinatorial de los códigos cíclicos a su forma algebraica, se utiliza la siguiente correspondencia:

$$\text{TL: } F_q^n \rightarrow F_q[x] / (x^n-1), \quad (c_0, c_1, \ldots, c_{n-1}) \mapsto c_0 + c_1x + \cdots + c_{n-i}x^{n-1}. \tag{3.1}$$

Entonces, TL es una transformación lineal de espacios vectoriales sobre F_q. De aquí en adelante, identificaremos algunas veces F_q^n con $F_q[x] / (x^n - 1)$, y al vector c = (c_0, c_1,…, c_{n-1}) con el polinomio

$$F[x] := \left\{ c_0 + c_1x + \cdots + c_{n-1}x^{n-1} : c_i \in F, n \geq 0 \right\},$$

Por el teorema 3.1.2.8, conocemos que $F_q[x] / (x^n - 1)$ es un anillo. Pero no es un campo a menos que n = 1 (i.e., que sea irreducible). De hecho, tenemos una operación multiplicativa además de la suma en F_q^n.

Ejemplo 3.1.3.2 Sea el código cíclico C = {000, 110, 101, 011}, entonces

$$TL = \{0, \quad 1 + x, \quad 1 + x^2, \quad x + x^2\} \subset F_2[x]/(x^3 - 1),$$

donde TL es la transformación lineal.

Otra forma de desplazamiento, se da si nosotros multiplicamos x·c(x)

$$x \cdot c(x) = (c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + c_{n-1}x^n).$$

Para representar el desplazamiento cíclico, movemos el coeficiente de x^n a la posición del coeficiente constante, tan solo por aplicar el producto módulo x^n-1. Dividimos x·c(x) por x^n-1 y usamos el polinomio de la división con residuo, tenemos

$$x \cdot c(x) = c_{n-1}(x^n-1) + (c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + c_{n-1})$$

donde $(c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + c_{n-1})$ es el residuo principal. Por lo tanto, el residuo de dividir por x^n-1 es:

$$x \cdot c(x) \pmod{x^n-1} = c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1}.$$

3.1.4 Ideales en Anillos.

Definición 3.1.4.1 Sea ***R*** un anillo. Un subconjunto ***I*** no vacío de ***R*** es llamado ideal si

(i) $a + b$ y $a - b$ pertenecen a I, para toda a, b $\in I$;
(ii) $r \cdot a \in I$, para toda $r \in R$ y a $\in I$.

Ejemplo 3.1.4.2 En el anillo $\mathbf{F}_2[x]/(x^3-1)$, el subconjunto

$$I:=\{0,\ 1 + x,\ x + x^2,\ 1 + x^2\}$$

es un ideal.

De hecho, un ideal es similar a un subespacio, generado por un conjunto de vectores base, excepto que para crear un subespacio, los coeficientes son escalares, mientras que para un ideal, los coeficientes son polinomios. La dirección hacia lo cual queremos llegar es a afirmar que "los códigos cíclicos forman ideales en un anillo de polinomios".

Definición 3.1.4.3 Un ideal I en un anillo R es principal si existe alguna $g \in I$ tal que cada elemento $a \in I$ puede ser expresado como un producto $a = mg$ para algún $m \in R$. Para un ideal principal, el elemento g es llamado elemento generador. El ideal generado por g es denotado como $\langle g \rangle$:

$$\langle g \rangle = \{gr : r \in R\}.$$

El elemento g es llamado generador de I y I es dicho ser generado por g. Para un anillo R se llama anillo ideal principal si cada ideal de R es principal. Sea I un ideal en $F_q[x]/(x^n-1)$. Entonces

1.- Hay un único polinomio mónico $g(x) \in I$ de mínimo grado.
2.- I es principal con generador $g(x)$.
3.- $g(x)$ divide $(x^n - 1)$ en $F_q[x]$.

Ejemplo 3.1.4.4 En el ejemplo 3.1.4.2 el ideal I es principal. De hecho $I = \langle 1+x \rangle$. Notamos que

$$\begin{aligned} 0 \cdot (1+x) &= 1 + x^3 = 0, \\ 1 \cdot (1+x) &= 1 + x, \\ x \cdot (1+x) &= x + x^2, \\ x^2 \cdot (1+x) &= 1 + x^2. \end{aligned}$$

Ejemplo 3.1.4.5 Comprobar que I es un ideal generado por $g(x) = (1+x+x^3)$ en el anillo $\mathbf{F}_2[x]/(x^7-1)$. $I:=\{0, 1+x+x^3, x+x^2+x^4, x^2+x^3+x^5, x^3+x^4+x^6, 1+x^4+x^5, x+x^5+x^6, 1+x^2+x^6\}$

Comprobación.

$$\begin{aligned} 0 * (1+x+x^3) &= 0 = (1+x+x^3)(x^4+x^2+x+1) = x^7-1 \\ 1 * (1+x+x^3) &= 1+x+x^3 = (1+x+x^3)(x^4+x^2+x) \\ x * (1+x+x^3) &= x+x^2+x^4 = (1+x+x^3)(x^4+x^2+1) \\ x^2 * (1+x+x^3) &= x^2+x^3+x^5 = (1+x+x^3)(x^4+x+1) \\ x^3 * (1+x+x^3) &= x^3+x^4+x^6 = (1+x+x^3)(x^4+x^3+x^2+x+1) \\ x^4 * (1+x+x^3) &= 1+x^4+x^5 = (1+x+x^3)(x^4) \\ x^5 * (1+x+x^3) &= x+x^5+x^6 = (1+x+x^3)(x^4+x^3+x^1) \\ x^6 * (1+x+x^3) &= 1+x^2+x^6 = (1+x+x^3)(x^4+x^3+x^2) \end{aligned}$$

Por lo tanto, $I = <1 + x + x^3>$. Además, g(x) divide $(x^7 - 1)$ en $F_2[x]$.

Comprobación.

$$
\begin{array}{r|l}
 & x^4 + x^2 + x + 1 \\
\hline
x^3 + x + 1 & x^7 - 1 \\
 & -x^7 - x^5 - x^4 \\
\hline
 & x^5 + x^4 + 1 \\
 & -x^5 - x^3 - x^2 \\
\hline
 & x^4 + x^3 + x^2 + 1 \\
 & -x^4 - x^2 - x \\
\hline
 & x^3 + x + 1 \\
 & -x^3 - x - 1 \\
\hline
 & 0 \quad 0 \quad 0
\end{array}
$$

Teorema 3.1.4.6 Los anillos **Z,** $\mathbf{F}_q[x]$ y $\mathbf{F}_q[x] / (x^n-1)$ son anillos ideales principales.

Demostración. Sea *I* un ideal de **Z**. Si $I = \{0\}$, entonces $I = < 0 >$ es un ideal principal. Asumimos que $I \neq \{0\}$ y sea *m* el entero positivo más pequeño en I. Sea *a* cualquier elemento de *I*. Por el algoritmo de la división, tenemos

$$a = qm + r \tag{3.2}$$

para algunos enteros *q* y $0 \leq r \leq m - 1$. La igualdad en (3.2) implica que *r* es también un elemento de *I* desde que $r = a - qm$. Esto hace que $r = 0$ por elegir *m*. Por lo tanto, $I = < m >$. Esto demuestra que **Z** es un anillo ideal principal.

Usando los mismos argumentos, podemos demostrar que $\mathbf{F}_q[x]$ es también un anillo ideal principal.

Para el caso de $\mathbf{F}_q[x] / (x^n-1)$ se aplican esencialmente los mismos argumentos. Si $I = \{0\}$, entonces $I = < 0 >$ es un ideal principal. Asumimos que $I \neq \{0\}$ y elegimos un polinomio diferente de cero g(x) de un ideal J con el menor grado. Para cualquier polinomio f(x) de J, tenemos que

$$f(x) = s(x)\, g(x) + r(x) \tag{3.3}$$

Para los polinomios s(x), g(x) $\in \mathbf{F}_q[x]$ con grad(r(x)) < grad(g(x)). Esto hace que r(x) = 0, como r(x) = f(x) - s(x)g(x) $\in J$ y g(x) tiene el más bajo grado para los polinomios diferentes de cero de *J*. Por lo tanto, $J = <g(x)>$, esto demuestra que $\mathbf{F}_q[x] / (x^n-1)$ es un anillo.

3.2 POLINOMIOS GENERADORES.

3.2.1 Polinomios generadores.

En este apartado, demostraremos la importancia de los ideales en los códigos cíclicos.

Teorema 3.2.1.1 Sea *I* un ideal diferente de cero en $\mathbf{F}_q[x]/(x^n - 1)$ y sea g(x) un polinomio mónico diferente de cero de menor grado en *I*. Entonces g(x) es un generador de *I* y divide $x^n - 1$.

Demostración. (Tomamos como referencia la demostración del Teorema 3.1.4.6). Consideramos el algoritmo de la división

$$x^n - 1 = s(x)g(x) + r(x)$$

con grad(r(x)) < grad(g(x)). Por lo tanto,

$$r(x) = (x^n - 1) - s(x)g(x)$$

es un elemento de I (notemos que $x^n - 1$ es el elemento cero de $\mathbf{F}_q[x]/(x^n - 1)$). Esto implica que r(x) = 0 ya que g(x) tiene el más bajo grado. Por lo tanto, g(x) es un divisor de $x^n - 1$.

Ejemplo 3.2.1.2 El conjunto $I = \{0, 1 + x^2, x + x^3, 1 + x + x^2 + x^3\}$ es un ideal en $\mathbf{F}_2[x]/(x^4 - 1)$. Su correspondiente código cíclico es $TL^{-1}(I) = \{0000, 1010, 0101, 1111\}$. Por lo tanto, el polinomio $1 + x^2$ es el de menor grado y divide a $x^4 - 1$.

Por el teorema 3.1.4.6, conocemos que cada ideal es principal en $\mathbf{F}_q[x]/(x^n-1)$, de hecho un código cíclico C es determinado por cualquiera de los generadores de TL(C). Generalmente hay más de un generador para un ideal de $\mathbf{F}_q[x]/(x^n - 1)$. Los siguientes resultados demuestran que el generador satisface ciertas propiedades adicionales.

Teorema 3.2.1.3 Hay un único polinomio mónico de menor grado en cada ideal *I* diferente de cero de $\mathbf{F}_q[x] / (x^n-1)$.

Demostración. Sea $g_i(x)$, i =1, 2, dos generadores mónicos distintos de menor grado del ideal *I*. Entonces, un múltiplo escalar de $g_1(x) - g_2(x)$ es un polinomio mónico diferente de cero y representa el grado más pequeño en *I*. Lo cual es una contradicción.

Definición 3.2.1.4 El polinomio mónico de menor grado de un ideal *I* diferente de cero de $\mathbf{F}_q[x]/(x^n-1)$ es llamado *polinomio generador* de *I*. Para un código cíclico C, el polinomio generador de TL(C) es también llamado el *polinomio generador* de C.

Ejemplo 3.2.1.5 El polinomio generador del siguiente código cíclico {000, 110, 011,101} es 1 + x.

Teorema 3.2.1.6 Cada divisor mónico de $x^n - 1$ es el *polinomio generador* de algún código cíclico en F_q^n.

Demostración. Sea g(x) un divisor mónico de $x^n - 1$ y sea *I* el ideal de < g(x) > de $\mathbf{F}_q[x]/(x^n-1)$ generado por g(x). Sea C el correspondiente código cíclico. Asumimos que h(x) es el polinomio generador de C. Entonces, existirá un polinomio b(x) tal que

$$h(x) \equiv g(x)b(x) \pmod{x^n - 1}.$$

De hecho, g(x) es un divisor de h(x). Por lo tanto, g(x) es el mismo que h(x) ya que h(x) tiene el menor grado y es mónico.

Ejemplo 3.2.1.7 Encontrar todos los código cíclico binarios de longitud 6 del polinomio $x^6 - 1 \in \mathbf{F}_2[x]$. Factorizamos de la siguiente forma

$$x^6 - 1 = (1 + x)^2(1 + x + x^2)^2.$$

Los divisores mónicos de $x^6 - 1$:

$$\begin{array}{ccc} 1, & 1 + x, & 1 + x + x^2, \\ (1 + x)^2, & (1 + x)(1 + x + x^2), & (1 + x)^2 (1 + x + x^2), \\ (1 + x + x^2)^2, & (1 + x)(1 + x + x^2)^2, & 1 + x^6. \end{array}$$

Por lo tanto, hay nueve códigos cíclicos binarios de longitud 6. Por ejemplo, el código cíclico correspondiente al polinomio $(1 + x + x^2)^2$ es

$$C = \{000000, 101010, 010101, 111111\}.$$

Conclusión: El número de códigos cíclicos de longitud n es determinado por la factorización de $x^n - 1$.

Teorema 3.2.1.8 Si $x^n - 1 \in F_q[x]$ tiene la siguiente factorización:

$$x^n - 1 = \prod_{i=1}^{r} p_i^{e_i}(x),$$

donde $p_1(x), p_2(x), \ldots, p_r(x)$ son distintos polinomios irreducibles mónicos y $e_i \geq 1$ para todo i = 1, 2,…, r. Por lo tanto, hay exactamente

$$\prod_{i=1}^{r} (e_i + 1)$$

códigos cíclicos de longitud n sobre F_q.

Demostración. El hecho de que número de códigos cíclicos es igual al número de polinomios irreducibles, se basa en decir, que existe una correspondencia uno a uno entre los códigos cíclicos en F_q^n y los divisores mónicos $x^n - 1 \in \mathbf{F}_q[x]$.

Ejemplo 3.2.1.9 Encontrar todos los códigos cíclicos binarios de longitud 7 en F_2^7**.** Por ejemplo, la descomposición en irreducibles mónicos de $x^7 - 1$ sobre F_2 es:

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1) = p_1(x)\cdot p_2(x)\cdot p_3(x).$$

De esta manera, como hay tres factores irreducibles hay $2^3 = 8$ códigos cíclicos binarios de longitud 7, dados por la siguiente tabla 3.4.

Los códigos C_1 y C_8 son los triviales.

	Ideal	polinomio generador	parámetros [n, k]
C_1=	1	1	[7, 7]
C_2 =	$x + 1$	$x + 1$	[7, 6]
C_3 =	$x^3 + x + 1$	$x^3 + x + 1$	[7, 4]
C_4 =	$x^3 + x^2 + 1$	$x^3 + x^2 + 1$	[7, 4]
C_5 =	$(x + 1) (x^3 + x + 1)$	$x^4 + x^3 + x^2 + 1$	[7, 3]
C_6 =	$(x + 1) (x^3 + x^2 + 1)$	$x^4 + x^2 + x + 1$	[7, 3]
C_7 =	$(x^3 + x + 1) (x^3 + x^2 + 1)$	$x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$	[7, 1]
C_8 =	$(x + 1) (x^3 + x + 1) (x^3 + x^2 + 1)$	$x^7 - 1$	[7, 0]

Tabla 3.4 Códigos cíclicos binarios de longitud 7

Las tablas 3.5 y 3.6 muestran la factorización de $x^n - 1$ y el número de q-arios códigos cíclicos de longitud ***n***, para $1 \leq n \leq 10$ y $\mathbf{q} = 2, 3$.

n	Factorización de $x^n - 1$	No. de códigos cíclicos
1	$1 + x$	2
2	$(1 + x)^2$	3
3	$(1 + x) (1 + x + x^2)$	4
4	$(1 + x)^4$	5
5	$(1 + x) (1 + x + x^2 + x^3 + x^4)$	4
6	$(1 + x)^2 (1 + x + x^2)^2$	9
7	$(1 + x) (1 + x^2 + x^3)(1 + x + x^3)$	8
8	$(1 + x)^8$	9
9	$(1 + x) (1 + x + x^2)(1 + x^3 + x^6)$	8
10	$(1 + x)^2 (1 + x + x^2 + x^3 + x^4)^2$	9
15	$(1+x)\ (1+x+ x^2)\ (1+x+ x^4)$ $(1+x^3+ x^4) (1+ x+ x^2 +x^3+ x^4)$	32

Tabla 3.5 Ejemplos de número de códigos cíclicos binarios, obtenidos a partir de su factorización.

n	Factorización de $x^n - 1$	No. de códigos cíclicos
1	$2 + x$	2
2	$(2 + x)(1 + x)$	4
3	$(2 + x)^3$	4
4	$(2+ x)(1 + x)(1 + x^2)$	8
5	$(2 + x) (1 + x + x^2 + x^3 + x^4)$	4
6	$(2 + x)^3(1 + x)^3$	16
7	$(2 + x) (1 + x + x^2+x^3+x^4+x^5+x^6)$	4
8	$(2 + x) (1 + x)(1 + x^2)(2+x+x^2)$ $(2 +2 x + x^2)$	32
9	$(2 + x)^9$	10
10	$(2 + x)(1 + x)(1 + x +x^2+x^3+x^4)$ $(1+2x +x^2 +2x^3 +x^4)$	16

Tabla 3.6 Ejemplos de número de códigos cíclicos ternarios, obtenidos a partir de su factorización.

3.2.2 Generadores y matrices parity check.

En este apartado, se demostrará que el *polinomio generador* determina a la matriz generadora y con esta matriz puede encontrarse la matriz parity check.

Teorema 3.2.2.1 Sea $g(x) = g_0 + g_1x + \cdots + g_{n-k}x^{n-k}$ el *polinomio generador* de un código cíclico C en F_q^n con grad (g(x)) = n - k. Entonces la matriz:

$$G=\begin{pmatrix} g(x) \\ xg(x) \\ \cdot \\ \cdot \\ \cdot \\ x^{k-1}g(x) \end{pmatrix} = \begin{pmatrix} g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & & & & & & & & & & \cdot \\ \cdot & \cdot & & & & & & & & & & \cdot \\ \cdot & \cdot & & & & & & & & & & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & g_0 & g_1 & \cdot & \cdot & \cdot & \cdot & g_{n-k} \end{pmatrix}$$

es una matriz generadora de C (notar que identificamos un vector con un polinomio).

Demostración. Será suficiente con demostrar que g(x), xg(x),…, $x^{k-1}g(x)$ forman una base de C (ver ejemplo 3.2.2.2). Es claro que son linealmente independientes sobre $\mathbf{F}_q$ con dim(C) = k.

Ejemplo 3.2.2.2 Consideramos el código cíclico binario de Hamming (7, 4) con el siguiente polinomio generador $g(x) = 1 + x^2 + x^3$. Entonces el código tiene un matriz generadora.

$$G=\begin{pmatrix} g_0 \\ x(g_1) \\ x^2(g_2) \\ x^3(g_3) \end{pmatrix} = \begin{pmatrix} 1\,0\,1\,1 & 0\,0\,0 \\ 0\,1\,0\,1 & 1\,0\,0 \\ 0\,0\,1\,0 & 1\,1\,0 \\ 0\,0\,0\,1 & 0\,1\,1 \end{pmatrix}.$$

La matriz anterior no esta en forma estándar, si el cuarto renglón es sumado al segundo renglón y la suma de los dos últimos renglones es sumado al primer renglón, nosotros formamos una matriz generadora en forma estándar:

$$G' = \begin{pmatrix} 1\,0\,0\,0 & 1\,0\,1 \\ 0\,1\,0\,0 & 1\,1\,1 \\ 0\,0\,1\,0 & 1\,1\,0 \\ 0\,0\,0\,1 & 0\,1\,1 \end{pmatrix}.$$

Teorema 3.2.2.3 Para un polinomio generador g(x) de un ideal $\mathbf{F}_q[x]/(x^n - 1)$ la dimensión de su correspondiente código cíclico es *k* si y solo si el grado de g(x) es *n* - *k*.

Demostración. Para cualquier palabra código, g(x)*a*(x) con *a*(x) Є $\mathbf{F}_q[x]/(x^n-1)$, tenemos

$$a(x)\, g(x) = u(x)\, (x^n-1) + v(x)$$

con grad(*v*(x)) < n. Haciendo, *v*(x)=*a*(x)g(x) - *u*(x) (x^n-1). Entonces, tenemos que g(x) divide a *v*(x). Escribimos *v*(x)=g(x)*b*(x) para algún polinomio *b*(x). Entonces, grad(*b*(x)) < *k*. Por lo tanto, *v*(x) esta en C. Esto demuestra que C es el mismo que <g(x)>. Por lo tanto, la dimensión del código es $\log_q |C|=k$. Es decir dim(C)=*k*.

Los códigos que satisfacen las siguientes dos propiedades se llaman códigos cíclicos.

1.- La suma de dos palabras código cualquiera es también una palabra código, lo dicho anteriormente es la propiedad de linealidad o espacio lineal.

2.-El resultado del desplazamiento cíclico de una palabra código es otra palabra código.

Desde que el código dual de un código cíclico C es también un código cíclico, podemos encontrar la matriz parity-check desde el polinomio generador del código dual.

Definición 3.2.2.4 Sea $h(x) = \sum_{i=0}^{k} a_i x^i$ un polinomio de grado k ($a_k \neq 0$) sobre F_q.

Definimos el *polinomio recíproco* $h_R(x)$ de h(x) por

$$h_R(x) := x^k h(1/x) = \sum_{i=0}^{k} a_{k-i} x^i .$$

Nota: si h(x) es un divisor de $x^n - 1$, entonces también lo es $h_R(x)$.

Ejemplo 3.2.2.5 Tenemos el siguiente polinomio $h(x) = 1 + x + x^3 \in \mathbf{F}_2[x]$ divisor de x^7-1. Entonces, $h_R(x) = 1 + x^2 + x^3$ también es divisor de x^7-1 (ver tabla 3.4).

Teorema 3.2.2.6 Sea g(x) el *polinomio generador* de un q-ario [n, k] código cíclico C. Donde $h(x) = (x^n-1)/g(x)$. Entonces $h_0^{-1} h_R(x)$ es el *polinomio generador* de $C^{\perp}$, donde h_0 es el término constante de h(x).

Demostración. Sea $g(x) = \sum_{i=0}^{n-1} g_i x^i$ y sea $h(x) = \sum_{i=0}^{n-1} h_i x^i$. Entonces

$$h_R(x) := (1/x^{n-k-1}) \sum_{i=0}^{n-1} h_{n-i-1} x^i ,$$

donde $K = \text{grad}(h(x))$.

Considerese el producto

$$\begin{aligned} 0 &\equiv g(x)h(x) \\ &\equiv (g_0h_0 + g_1h_{n-1} + \cdots + g_{n-1}h_1) + (g_0h_1 + g_1h_0 + \cdots + g_{n-1}h_2)\,x + (g_0h_2 + g_1h_1 + \cdots + \\ &\quad g_{n-1}h_3)x^2 + \cdots + (g_0h_{n-1} + g_1h_{n-2} + \cdots + g_{n-1}h_0)x^{n-1} (\text{mod } x^n-1). \end{aligned}$$

Por observar los coeficientes de las potencias de x (que deberían ser cero), obtenemos $g_i \cdot (h_{n-1}, h_{n-2}, \ldots, h_1, h_0) = 0$. Por lo tanto, $(h_{n-1}, h_{n-2}, \ldots, h_1, h_0)$ es una palabra código de $C^{\perp}$ desde que $(g_0, g_1, \ldots, g_{n-1})$ genera C por el teorema 3.2.2.1.

Por formar el desplazamiento cíclico del vector $(h_{n-1}, h_{n-2}, \ldots, h_1, h_0)$ para k+1 posiciones, se obtiene el vector correspondiente a $h_R(x)$. Esto implica que $h_R(x)$ es una palabra código $C^{\perp}$ y que además es cíclica.

Como $\text{grad}(h_R(x)) = \text{grad}(h(x)) = k$, el conjunto $\{h_R(x), x\,h_R(x), \ldots, x^{n-k-1} h_R(x)\}$ es una base de $C^{\perp}$. Por lo tanto, $C^{\perp}$ es generado por $h_R(x)$. De hecho, el polinomio mónico $h_0^{-1} h_R(x)$ es el polinomio generador de $C^{\perp}$.

Definición 3.2.2.7 Sea C un q-ario código cíclico de longitud n. Teniendo, $h(x) = (x^n - 1)/g(x)$. Entonces, $h_0^{-1} h_R(x)$ es llamado el *polinomio parity-check* de C, donde h_0 es el término constante de h(x).

Corolario 3.2.2.8 Sea C un código cíclico [n, k] con polinomio generador g(x). Teniendo $h(x) = (x^n - 1)/g(x)$. Sea $h(x) = h_0 + h_1 x + \cdots + h_k x^k$. Entonces, la siguiente matriz

$$H = \begin{pmatrix} h_R(x) \\ x h_R(x) \\ \cdot \\ \cdot \\ \cdot \\ x^{n-k-1} h_R(x) \end{pmatrix} = \begin{pmatrix} h_k & h_{k-1} & . & . & . & h_o & 0 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & h_k & h_{k-1} & . & . & . & h_o & 0 & 0 & \cdot & \cdot & 0 \\ \cdot & & & & & & & & & & \cdot & \\ \cdot & & & & & & & & & & \cdot & \\ \cdot & & & & & & & & & & \cdot & \\ 0 & 0 & \cdot & \cdot & \cdot & h_k & h_{k-1} & \cdot & \cdot & \cdot & \cdot & h_o \end{pmatrix}$$

es una *matriz parity check* de C.

Ejemplo 3.2.2.9 Sea C el código cíclico binario [7, 4] generado por $g(x) = 1 + x^2 + x^3$. Teniendo $h(x) = (x^7 - 1)/g(x) = 1 + x^2 + x^3 + x^4$. Entonces, $h_R(x) = 1 + x + x^2 + x^4$ es la *matriz parity check* de C. Por lo tanto,

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

es la *matriz parity check* de C.

Ejemplo 3.2.2.10 Sea el código cíclico binario [7, 4] generado por $g(x) = 1 + x + x^3$. Teniendo, $h(x) = (x^n - 1)/g(x) = 1 + x + x^2 + x^4$. Entonces, $h_R(x) = 1 + x^2 + x^3 + x^4$ es la *matriz parity check* de C. Por lo tanto,

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

es la *matriz parity check* de C.

3.3 DECODIFICACIÓN DE LOS CÓDIGOS CÍCLICOS.

3.3.1 Decodificación de los códigos cíclicos.

La decodificación de los códigos cíclicos consiste en los mismos tres pasos de la decodificación de los códigos lineales: calcular el síndrome; encontrar el síndrome correspondiente al error y corregir los errores. Debido a la estructura de los códigos cíclicos, los tres pasos para decodificarlos generalmente son más simples. Los códigos cíclicos consideran propiedades algebraicas y geométricas. Estas propiedades simplifican la decodificación de un código cíclico. En los códigos cíclicos se puede producir fácilmente la matriz parity check de la siguiente forma

$$H = (I_{n-k} \mid A)$$

tan solo por realizar operaciones de renglón elementales. Todos los síndromes que son calculados en este apartado serán calculados con la matriz parity check en su forma estándar.

Definición 3.3.1.1 Sea H = (I_{n-k} | A) ser una matriz parity check de un q–ario código cíclico C. Sea g(x) el polinomio generador de C. Entonces el síndrome de un vector **w** Є $\boldsymbol{F}_q^n$ es igual a ($w(x)$(mod g(x))); i.e., el residuo principal de $w(x)$ dividido por g(x).

Ejemplo 3.3.1.2 Consideramos el código binario de Hamming [7, 4, 3] con el polinomio generador g(x) = $1 + x^2 + x^3$. Entonces, por realizar operaciones de renglón elementales a la matriz de el ejemplo 3.2.2.9, obtenemos la matriz parity check $H = (I_3|A)$

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

Para la palabra **w** = 0110110, el síndrome es $\mathbf{s} = \mathbf{w}H^T = 010$. De otra manera,

$$w(x) = x + x^2 + x^4 + x^5 = x + x^2 g(x).$$

Por lo tanto, el residuo ($w(x)$ (mod g (x))) es x, el cual corresponde a la palabra 010.

Definición 3.3.1.3 El síndrome de una palabra recibida $w(x)$ puede ser determinada por el residuo $s(x)$ = ($w(x)$ (mod g (x))). Por lo tanto, $w(x)$ - $s(x)$ es una palabra código.

Corolario 3.3.1.4 Sea g(x) el polinomio generador de un código cíclico C. Para una palabra recibida $w(x)$, si el residuo $s(x)$ de w(x) dividido por g(x) tiene peso menor o igual a $\lfloor (d(C)-1)/2 \rfloor$, entonces $s(x)$ es el patrón de error de $w(x)$; i.e., $w(x)$ es decodificado a $w(x)$ - $s(x)$ por la regla de decodificación por máxima similitud.

Demostración 3.3.1.5 Si se conoce que $w(x)$ y $s(x)$ están en el mismo co-conjunto. Además, $s(x)$ es el co-conjunto líder. Entonces, su peso es de

$$wt(s(x)) \leq \lfloor (d(C)-1)/2 \rfloor .$$

Ejemplo 3.3.1.6 El residuo de $w(x) = x + x^2 + x^4 + x^5$ dividido por $g(x) = 1 + x^2 + x^3$ es x. Por lo tanto, $w(x)$ es decodificado a $w(x) - x = x^2 + x^4 + x^5 = 0010110$. Si la palabra $w_1(x) = 1 + x^2 + x^3 + x^4$ es recibida, entonces el residuo $(w_1(x) \pmod{g(x)})$ es $1 + x + x^2$. En este caso, podemos usar la decodificación por síndrome para obtener la palabra código $w_1(x) - x^4 = 1 + x^2 + x^3 = 1011000$ esto es debido a que la palabra 0000100 es el co-conjunto líder para el co-conjunto donde se encuentra $w_1(x)$.

Para algunas palabras código recibidas, podemos simplificar la decodificación por síndrome con ayuda de las propiedades algebraicas y geométricas de los códigos cíclicos. Como se puede apreciar, en el ejemplo anterior, para algunas palabras se puede decodificar directamente tan solo por obtener el residuo polinomial de las palabras. Sin embargo, para otras palabras podemos utilizar la decodificación por síndrome. A continuación describiremos la decodificación llamada error trapping o conocido también como el método de captura del error.

Lema 3.3.1.7 Sea C un q-ario código cíclico con polinomio generador g(x). Sea

$$s(x) = \sum_{i=0}^{n-k-1} s_i x^i$$

el síndrome de w(x). Entonces el síndrome del desplazamiento xw(x) es igual a $xs(x) - s_{n-k-1}g(x)$.

Demostración. Es suficiente con demostrar que $xs(x) - s_{n-k-1}g(x)$ es el residuo de xw(x) dividido por g(x). Como w(x) = q(x)g(x) + s(x). Entonces

$$xw(x) = xq(x)g(x) + xs(x) = (x\,q(x) + s_{n-k-1})g(x) + (xs(x) - s_{n-k-1}g(x)).$$

El resultado concluye que $\text{grad}(xs(x) - s_{n-k-1}g(x)) < n - k = \text{grad}(g(x))$.

Definición 3.3.1.8 Una corrida cíclica de 0 de longitud *l* de una n-tupla es una sucesión de *l* consecutivos componentes cero.

Ejemplo 3.3.1.9 e = (1, 3, 0, 0, 0, 0, 0, 1, 0) tiene una corrida cíclica de 0 de longitud 5.
Ejemplo 3.3.1.10 e = (0, 0, 1, 2, 0, 0, 0, 1, 0, 0) tiene una corrida cíclica de 0 de longitud 4.

Algoritmo de decodificación para códigos cíclicos.

Sea C un q-ario código cíclico [n, k, d] con polinomio generado g(x). Sea w(x) la palabra recibida con un patrón de error e(x), donde

$$wt(e(x)) \le \lfloor (d(C) - 1)/2 \rfloor$$

y e(x) tiene una corrida cíclica de 0 de longitud al menos k. La meta es determinar e(x).

Paso 1.- Calcule el síndrome de x^i w(x), para $i = 0, 1, 2, \ldots$, y denotamos por $s_i(x)$ el síndrome $(x^i\, w(x) \pmod{g(x)})$.

Paso 2.- Encontrar *m* tal que el peso del síndrome $s_m(x)$ para $x^m w(x)$ es menor que o igual a: $\lfloor (d(C) - 1)/2 \rfloor$.

Paso 3.- Calcule el residuo e(x) de $x^{n-m}\, s_m(x)$ dividido por $x^n - 1$. Decodifique $w(x) = w(x) - e(x)$.

Ejemplo 3.3.1.11 Consideramos la palabra recibida $w_1(x)$ = 1011100. Ahora, expresado en su forma polinómica $w_1(x) = 1 + x^2 + x^3 + x^4$. Hacemos la división por $g(x) = 1 + x^2 + x^3$, para obtener el síndrome $s_0(x)$. $w_1(x) = g(x) * (x) + (1 + x + x^2)$: Como $s_0(x) = 1 + x + x^2$ tiene peso 3 debemos seguir. La relación de recurrencia nos permite obtener $s_1(x) = xs_0(x) - (1)g(x) = 1 + x$ con peso 2. Buscamos el siguiente síndrome: $s_2(x) = x\, s_1(x) - (1)g(x) = x + x^2$ con peso *2*. Buscamos el siguiente síndrome: $s_3(x) = x\, s_2(x) - (1)g(x) = 1$ como el peso de $s_3(x)$ es 1, hemos terminado[17]. El error viene dado por $e(x) = x^{7-3}\, s_3(x) = x^4$. Entonces, se decodifica poniendo $w(x) = w_1(x) - e(x)$ = 1011100 - 0000100 = 1011000.

i	$s_i(x)$
0	$1 + x + x^3 + x^4 + x^6 + x^8 + x^9$
1	$1 + x^7 + x^8 + x^9$
2	$1 + x^2 + x^4 + x^5 + x^9$
3	$1 + x^2 + x^3 + x^4 + x^6 + x^8$
4	$x + x^3 + x^4 + x^5 + x^7 + x^9$
5	$1 + x + x^6$

Tabla 3.6

Ejemplo 3.3.1.12 Consideramos el código cíclico [15, 5] generado por $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$. Recibimos la palabra:

$$w_1(x) = 100101111011100 = 1 + x^3 + x^5 + x^6 + x^7 + x^8 + x^{10} + x^{11} + x^{12},$$

podemos comprobar desde la matriz parity check que la distancia mínima es 7. Por lo tanto, se puede corregir hasta un patrón de error de 3.

Polinomio generador = [1 1 1 0 1 1 0 0 1 0 1 0 0 0 0]; matriz parity check:

$$H=(I_{n-k} \mid A) = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&1&0&1&0&1 \\ 0&1&0&0&0&0&0&0&0&0&1&1&1&1&1 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&1&0&1 \\ 0&0&0&0&1&0&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&1&0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&1 \end{pmatrix}$$

Calculamos los síndromes $s_i(x)$ de $x^i\, w(x)$ hasta $wt(s_i(x)) \leq 3$ (ver tabla 3.6).

Decodificamos $w_1(x)$ = 100101111011100 a $w_1(x) - x^{10}\, s_5(x)$

$w_1(x) = 1 + x^3 + x^5 + x^6 + x^7 + x^8 + x^{10} + x^{11} + x^{12} - (x^{11} + x^{10} + x)$.

Por lo tanto, $w(x) = w_1(x) - e(x) = 1 + x + x^3 + x^5 + x^6 + x^7 + x^8 + x^{12}$.

La palabra decodificada es: $w(x) = 110101111000100$.

[17] El valor de (1)*g(x), (1) es debido a que el coeficiente lider de $s_i(x)$ es 1.

3.4 CÓDIGOS DE CORRECCIÓN DE ERROR EN BURST (RÁFAGAS).

3.4.1 Códigos de corrección de error en burst (ráfaga).

Hasta el momento ha sido posible corregir errores aleatorios. Sin embargo, como se comentó en el primer capítulo, hay ciertos canales de comunicación, como las líneas telefónicas y sistemas de almacenamiento magnético, los cuales son afectados por errores localizados sucesivamente en intervalos cortos y no de manera aleatoria. Tales errores son llamados ráfaga o burts. Los códigos cíclicos son muy eficientes para corregir errores burst, los códigos de esta clase son llamados códigos correctores de error burst. En este apartado, discutiremos algunas propiedades de estos códigos y un algoritmo de decodificación.

Definición 3.4.1.1 Un burst de longitud $l > 1$ es un vector binario con componentes diferentes de cero, con la primera y última posición diferente de cero. Un código es llamado *código de corrección de error burst de tamaño l* si y solo si corrige todos los errores burst de longitud l y menores que l.

Ejemplo 3.4.1.2 0011010000 es un burst de longitud 4 y 01000000000000100 es un burst de longitud 5.

Teorema 3.4.1.3 Un código lineal C es un código de corrección de error en forma de burst de tamaño l si y solo si el error burst de longitud l o menores a l se encuentran en distintos co-conjuntos de C.

Demostración. Si todos los errores burst de longitud $\leq l$ se encuentran en distintos co-conjuntos, entonces el error en forma de burst puede ser corregido a través de su síndrome. De otra manera, suponiendo que dos errores burst $\mathbf{b}_1$ y $\mathbf{b}_2$ de longitud $\leq l$ se encuentran en el mismo co-conjunto de C. La diferencia $\mathbf{c} = \mathbf{b}_1 - \mathbf{b}_2$ es una palabra código. De hecho, si $\mathbf{b}_1$ es recibido, entonces $\mathbf{b}_1$ podría ser decodificado a ambos $\mathbf{0}$ y $\mathbf{c}$.

Corolario 3.4.1.4 Sea C un código lineal de corrección de error burst de longitud l con parámetros [n, k]. Entonces

(i) Ningún burst diferente de 0 de longitud $\leq 2l$ puede ser una palabra código,

(ii) $n - k \geq 2l$ (límite de Reiger).

Demostración.

(i) Supongamos que existe una palabra código $\mathbf{c}$ la cual es un burst de longitud $\leq 2l$. Entonces, $\mathbf{c}$ es de la forma (0, 1, **u, v,** 1,0), donde $\mathbf{u}$ y $\mathbf{v}$ son dos palabras de longitud $\leq l - 1$. Por lo tanto, las palabras $\mathbf{w}$ = (0, 1, **u,** 0, 0, 0) y $\mathbf{c} - \mathbf{w}$ = (0, 0, 0**, v,** 1, 0) son dos burst de longitud $\leq l$. Están en el mismo co-conjunto. Esto es una contradicción al Teorema 3.4.1.3.

(ii) Sea $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n-k+1}$ el primer vector columna $n - k + 1$ de una matriz parity check de C. Entonces, se encuentra en F_2^{n-k} y son por lo tanto linealmente dependientes. De hecho, existe $c_1, c_2, \ldots, c_{n-k+1}$ que $\in \mathbf{F}_2$, no todos iguales a cero, tal que

$$\sum_{i=l}^{n-k+1} c_i u_i = 0.$$

Esto implica que $(c_1, c_2, \ldots, c_{n-k+1}, \mathbf{0})$ es una palabra código, y esta palabra código es un burst de longitud $\leq n - k + 1$. Por lo tanto, tenemos que $n - k + 1 > 2l$; i.e., $n - k + 1 \geq 2l$.

Un código lineal que corrige errores burst con parámetros [*n, k*] satisface $n - k \geq 2l$; es decir,

$$l \leq \left\lfloor \frac{n-k}{2} \right\rfloor.$$

Un código lineal de corrección de error burst es llamado óptimo si se le aplica el límite de Reiger.

Nota: la principal diferencia es que, en el caso de corrección de errores burst, no se requiere que el peso de un patrón de error sea menor que o igual a

$$\lfloor (d(C)-1)/2 \rfloor .$$

Algoritmo de decodificación para códigos de corrección de error ráfaga.

Sea C un q-ario código cíclico de parámetros [n, k, d] con polinomio generador g(x). Sea w(x) una palabra recibida con un patrón de error e(x) esto es un burst de error de longitud $\leq l$.

Paso 1.- Calcular el síndrome de x^i w(x), para $i = 1, 2,\ldots$, y denotamos por $s_i(x)$ al síndrome de x^i w(x).

Paso 2.- Encontrar *m* tal que síndrome para $x^m w(x)$ es un burst de longitud $\leq l$.

Paso 3.- Calcule el residuo e(x) de $x^{n-m} s_m(x)$ dividido por x^n-1. Decodifique $w(x) = w(x) - e(x)$.

i	$s_i(x)$
0	$1 + x + x^4 + x^5$
1	$1 + x^3 + x^5$
2	$1 + x^2 + x^3 + x^4$
3	$x + x^3 + x^4 + x^5$
4	$1 + x + x^3 + x^4 + x^5$
5	$1 + x^3 + x^4 + x^5$
6	$1 + x^2 + x^3 + x^4 + x^5$
7	$1 + x^2 + x^4 + x^5$
8	$1 + x^2 + x^5$
9	$1 + x^2$

Tabla 3.7

Ejemplo 3.4.1.5 Considere el código cíclico binario con parámetros [15, 9] generado por $g(x) = 1+x+x^2+x^3+x^6$. Podemos corregir errores burst de longitud ≤ 3. Suponiendo que recibimos la palabra $w(x)$ como:

$$w(x) = 111011101100000 = 1 + x + x^2 + x^4 + x^5 + x^6 + x^8 + x^9.$$

Calculamos el síndrome $s_i(x)$ o $x^i w(x)$ hasta que $s_m(x)$ sea una ráfaga de longitud ≤ 3. (Ver tabla 3.7). Decodificamos $w(x) = 111011101100000$ a

$$w(x) - x^6 s_9(x) = w(x) - x^6 - x^8 = 1 + x + x^2 + x^4 + x^5 + x^9 = 111011000100000.$$

CAPÍTULO 4

CÓDIGOS BCH

4. CÓDIGOS BCH

Hemos visto que los códigos de Hamming son códigos de mucha utilidad porque son lineales y perfectos, sin embargo, los códigos de Hamming no son ideales para situaciones en las cuales ocurren más de un bit de error en una transmisión. Lo anterior es debido a que los códigos de Hamming únicamente corrigen un solo error. De hecho, si más de un bit de error ocurre durante la transmisión de un código de Hamming, el vector recibido no será corregido a la palabra código que fue enviada. Existen códigos que en cierto sentido son generalizaciones de estos códigos y que son capaces de corregir **t** errores que ocurran durante la transmisión, para un entero dado **t**. Estos son los llamados códigos BCH (llamados así por sus descubridores Bose-Chaudhuri-Hocquenghem) mejor considerados como códigos cíclicos. Naturalmente la teoría de los códigos cíclicos, dada en el capítulo anterior, aplica a los códigos BCH. En particular, demostramos que un código cíclico es totalmente determinado por su polinomio generador. Sin embargo, es difícil obtener información de la distancia mínima de un código cíclico a partir de su polinomio generador, sobre todo si su formación es determinada por este último. De otra manera, si elegimos algunas propiedades de algunos polinomios generadores, entonces la información de la distancia mínima puede ser obtenida y un algoritmo simple de decodificación pudiera ser aplicado. Los códigos BCH binarios fueron descubiertos por A. Hocquenghem en 1959 [48; 159] e independientemente por R. C. Bose y D. K. Ray-Chaudhuri en 1960. La generalización al caso q-ario fue realizada por D. C. Gorenstein y N. Zierler en 1961. Entre los códigos BCH no binarios, la más importante subclase es la clase de los códigos Reed-Solomon descubiertos por I. S. Reed y G. Solomon [47].

4.1 CÓDIGOS BCH.

Definimos en el capítulo 3, el mínimo común múltiplo mcm ($f_1(x), f_2(x)$) de dos polinomios diferentes de cero $f_1(x), f_2(x) \in \mathbf{F}_q[x]$ como el polinomio mónico de mas bajo grado el cual es un múltiplo de $f_1(x)$ y $f_2(x)$. Suponemos que tenemos **t** polinomios diferentes de cero $f_1(x), f_2(x),\ldots, f_t(x) \in \mathbf{F}_q[x]$. El mínimo común múltiplo de $f_1(x), f_2(x),\ldots, f_t(x)$ es el polinomio mónico de más bajo grado el cual es múltiplo de, $f_1(x), f_2(x),\ldots, f_t(x)$, denotado por $mcm(f_1(x), f_2(x),\ldots, f_t(x))$.

Lema 4.1.1.1 Sea $f(x), f_1(x), f_2(x),\ldots, f_t(x)$ polinomios sobre $\mathbf{F}_q$. Si $f(x)$ es divisible por cada polinomio $f_i(x)$ para $i = 1,2,\ldots, \boldsymbol{t}$, entonces $f(x)$ es también divisible por mcm ($f_1(x), f_2(x),\ldots, f_t(x)$).

Demostración. Supongamos que $g(x)$ = mcm ($f_1(x), f_2(x),\ldots, f_t(x)$). Por el algoritmo de la división, existen dos polinomios $u(x)$ y $r(x)$ *sobre* $\mathbf{F}_q$ tal que grad ($r(x)$) < grad ($g(x)$) y

$$f(x) = u(x)g(x) + r(x).$$

De hecho, $r(x) = f(x) - u(x)g(x)$, y por lo tanto $r(x)$ es también divisible para toda $f_i(x)$. Desde que $g(x)$ tiene el más pequeño grado, esto hace que $r(x) = 0$.

Ejemplo 4.1.1.2 El polinomio $f(x) = x^{15} - 1 \in \mathbf{F}_2[x]$ es divisible por $f_1(x) = 1+x+x^2 \in \mathbf{F}_2[x]$, $f_2(x)=1 + x + x^4 \in \mathbf{F}_2[x]$ y $f_3(x) =(1+ x+x^2)(1+x^3+x^4) \in \mathbf{F}_2[x]$, respectivamente. Entonces $f(x)$ es también divisible por mcm $(f_1(x), f_2(x), f_3(x)) = (1 + x + x^2)(1+ x + x^4)(1 + x^3 + x^4)$.

Ejemplo 4.1.1.3 Del Apéndice B, donde analizamos a un elemento primitivo α de F_{q^m} y denotamos $M^{(i)}(x)$ al polinomio mínimal de α^i con respecto a F_q. Por el teorema B3 del Apéndice B, cada raíz β de $M^{(i)}(x)$ es un elemento de F_{q^m}, y por lo tanto β satisface $\beta^{q^m-1} - 1 = 0$; i.e., x - β es un divisor lineal de

$$X^{q^m-1} - 1.$$

Como $M^{(i)}(x)$ no tiene raíces múltiples. Por lo tanto, $M^{(i)}(x)$ es un divisor de $X^{q^m-1} - 1$. Por el lema 4.1.1.1, para un subconjunto $\boldsymbol{I}$ de Z_{q^m-1}, el mínimo común múltiplo $\text{mcm}(M^{(i)}(x))_{i \in I}$ es un divisor de

$$X^{q^m-1} - 1.$$

El ejemplo anterior es un método para encontrar algunos divisores de $X^{q^m-1} - 1$. Estos divisores pueden ser elegidos como polinomios generadores de códigos cíclicos de longitud $q^m - 1$.

Definición 4.1.1.4 Sea α un elemento primitivo de F_{q^m} y denotamos por $M^{(i)}(x)$ el polinomio mínimal de α^i con respecto a $\mathbf{F}_q$. Un código BCH (primitivo) sobre F_q de longitud $n = q^m - 1$ con distancia de diseño δ es un q-ario código cíclico generado por $g(x) = \text{mcm}\,(M^{(a)}(x), M^{(a+1)}(x),\ldots, M^{(a+\delta-2)}(x))$ para cualquier entero a. Además, si $a=1$ durante el procedimiento de diseño, el código BCH es llamado "narrow sense"[18].

4.1.2 DISEÑO Y PARÁMETROS DE UN CÓDIGO BCH.

Los códigos BCH son códigos cíclicos y por lo tanto pueden ser determinados por su polinomio generador. Un código BCH sobre $\mathbf{F}_q$ de longitud n, con distancia $\delta=2t+1$ capaz de corregir al menos t errores es determinado de la siguiente manera:

1. Determinar el más pequeño m tal que F_{q^m} tenga raíz primitiva n de unidad β.
2. Seleccionar un entero no negativo $a=1$.
3. Escribir una lista de $2t$ consecutivas potencias de β:
   $$\beta^a, \beta^{a+1}, \beta^{a+2t-1}.$$
   y determinar el polinomio mínimal en con respecto a $\mathbf{F}_q$ de cada potencia de β.
4. El polinomio generador $g(x)$ es el mínimo común múltiplo *(mcm)* de estos polinomios mínimales. El código BCH es un código cíclico $(n, n - \text{grad}(g(x))$.

Ejemplo 4.1.2.1 Se desea construir un código BCH de longitud 15 y que corrija 2 errores. Sean $n = 15 = 2^4 - 1$ para un código primitivo con $m = 4$ y sea β una raíz del polinomio primitivo $x^4 + x + 1$ en F_{q^4}. Si tomamos $a = 1$ (código narrow sense) y construimos un código binario BCH que corrija 2 errores, Entonces tenemos $t=2$ y $2t=4$ y las consecutivas potencias de β son: $\beta^1, \beta^2, \beta^3, \beta^4$. Por lo tanto, tenemos $\{\beta^1, \beta^2, \beta^4\}$ y $\{\beta^3\}$. Los cálculos de los polinomios mínimales son:

[18] "narrow sense" o "sentido estrecho"

$$M^{(0)}(x) = 1 + x$$
$$M^{(1)}(x) = M^{(2)}(x) = M^{(4)}(x) = M^{(8)}(x) = 1 + x + x^4$$
$$M^{(3)}(x) = M^{(6)}(x) = M^{(12)}(x) = M^{(9)}(x) = 1 + x + x^2 + x^3 + x^4$$
$$M^{(5)}(x) = M^{(10)}(x) = 1 + x + x^2$$
$$M^{(7)}(x) = M^{(14)}(x) = M^{(13)}(x) = M^{(11)}(x) = 1 + x^3 + x^4$$

y nuestros polinomios mínimales son:

$$M^{(1)}(x) = x^4+x+1 \quad y \quad M^{(3)}(x) = x^4+x^3+x^2+x+1.$$

Por lo tanto,

$$g(x)=mcm\,[M^{(1)}(x), M^{(3)}(x)] = (x^4+x+1)\,(x^4+x^3+x^2+x+1) = x^8+x^7+x^6+x^4+1.$$

Siendo, *grad (g(x)) = 8*, entonces tenemos un *(15, 7, 5)* código BCH.

Calculamos su código dual $h(x) = \dfrac{x^{15} - 1}{g(x)} = (x + 1)(x^2 + x + 1)(x^4 + x^3 + 1)$

y por lo tanto, $h(x)^{\perp} = x^7 (x^{-7}+x^{-6}+x^{-4}+1) = 1+x+x^3+x^7$ es su código dual.

Ejemplo 4.1.2.2 Se desea construir un código BCH de longitud 15 y que corrija 3 errores. Sean $n = 15 = 2^4 - 1$, para un código primitivo con $m=4$ y sea β una raíz del polinomio primitivo x^4+x+1 en F_{q^4}. Si tomamos $a=1$ (código narrow sense) y construimos un código binario BCH que corrija 3 errores. Entonces, tenemos $t=3$ y $2t=6$ y las consecutivas potencias de β son: $\beta^1, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6$. Por lo tanto, tenemos $\{\beta^1, \beta^2, \beta^4\}$, $\{\beta^3, \beta^6\}$ y $\{\beta^5\}$. Los cálculos de los polinomios mínimales son:

$$M^{(0)}(x) = 1 + x$$
$$M^{(1)}(x) = M^{(2)}(x) = M^{(4)}(x) = M^{(8)}(x) = 1 + x + x^4$$
$$M^{(3)}(x) = M^{(6)}(x) = M^{(12)}(x) = M^{(9)}(x) = 1 + x + x^2 + x^3 + x^4$$
$$M^{(5)}(x) = M^{(10)}(x) = 1 + x + x^2$$
$$M^{(7)}(x) = M^{(14)}(x) = M^{(13)}(x) = M^{(11)}(x) = 1 + x^3 + x^4$$

y nuestros polinomios mínimales son:

$$M^{(1)}(x) = 1+x+x^4, \quad M^{(3)}(x) = 1+x+x^2+x^3+x^4 \quad y \quad M^{(5)} = 1+x+x^2.$$

Por lo tanto:

$$g(x)=mcm\,[M^{(1)}(x), M^{(3)}(x),), M^{(5)}(x)]$$

$$g(x) = (1+x+x^4)\,(1+x+x^2+x^3+x^4)\,(1+x+x^2) = 1+x+x^2+x^4+x^5+x^8+x^{10}.$$

Siendo *grad (g(x))=10*, entonces tenemos un (15, 5, 7) código BCH. Quedando claro que la longitud de un código BCH es $q^m - 1$, a continuación demostraremos la forma de obtener la dimensión de un código BCH.

Teorema 4.1.2.3 La dimensión de un *q-ario* código BCH de longitud $q^m - 1$ generado por $g(x) := \text{mcm}\,(M^{(a)}(x), M^{(a+1)}(x), \ldots, M^{(a+\delta-2)}(x))$ es independiente de la elección del elemento primitivo α.

Demostración. Sea C_i el co-conjunto ciclotómico de $q^m - 1$ que contiene *i*. Suponemos

$$S = \bigcup_{i=a}^{a+\delta-2} C_i .$$

Entonces, tenemos

$$g(x) = mcm\left(\prod_{i\in C_a}\left(x-\alpha^i\right), \prod_{i\in C_{a+1}}\left(x-\alpha^i\right), \ldots, \prod_{i\in C_{a+\delta-2}}\left(x-\alpha^i\right)\right) = \prod_{i\in S}\left(x-\alpha^i\right).$$

Por lo tanto, la dimensión *k* es igual a q^m*- 1- grad* $(g(x)) = q^m$*-1-* $|S|$. Es obvio que el conjunto *S* es independiente de la elección de α.

De acuerdo a este resultado, la dimensión de un código BCH, se puede encontrar con la cardinalidad de

$$\bigcup_{i=a}^{a+\delta-2} C_i ,$$

donde C_i es el co-conjunto ciclotómico de *q* módulo q^m*- 1* que contiene *i*.

Ejemplo 4.1.2.4 Consideramos el siguiente co-conjunto ciclotómico de 2 módulo 15: $C_2 = \{1, 2, 4, 8\}$, $C_3 = \{3, 6, 12, 9\}$. Entonces la dimensión de un código BCH binario de longitud 15 de distancia designada 3 generado por $g(x) :=$ mcm $(M^{(2)}(x), M^{(3)}(x))$ es

$$15 - |C_2 \cup C_3| = 15 - 8 = 7.$$

Ejemplo 4.1.2.5 Considere el siguiente co-conjunto ciclotómico de 3 módulo 26:

$$C_1 = C_3 = \{1, 3, 9\}, \quad C_2 = \{2, 6, 18\}, \quad C_4 = \{4, 12, 10\}.$$

Entonces, la dimensión del código BCH ternario de longitud 26 y distancia del diseño 5 generado por

$$g(x) := \text{mcm}\,(M^{(1)}(x), M^{(2)}(x), M^{(3)}(x), M^{(4)}(x))$$

es

$$26 - |C_1 \cup C_2 \cup C_3 \cup C_4| = 26 - 9 = 17.$$

Ejemplo 4.1.2.6 Para $t \geq 1$, *t* y *2t* pertenecen al mismo co-conjunto ciclotómico de *2* módulo $2^m - 1$. Esto es equivalente a decir que $M^{(t)}(x) = M^{(2t)}(x)$. Por lo tanto,

$$\text{mcm}\,(M^{(1)}(x), \ldots, M^{(2t-1)}(x)) = \text{mcm}\, M^{(1)}(x), \ldots, M^{(2t)}(x)).$$

Es decir, los códigos BCH binarios (narrow sense) de longitud $2^m - 1$ con distancia de diseño $2t + 1$ son los mismos que los códigos BCH binarios (narrow sense) de longitud $2^m - 1$ con distancia de diseño $2t$, podemos considerar a los códigos BCH binarios (narrow sense) como aquellos que tienen una distancia de diseño impar.

Lema 4.1.2.7 Sea C un *q-ario* código cíclico de longitud n con polinomio generador *g(x).* Suponiendo que $\alpha_1,\ldots, \alpha_r$ son todas las raíces de *g(x)* y el polinomio g*(x)* es irreducible. Entonces un elemento *c(x)* de $\mathbf{F}_q[x]/(x^n - 1)$ es una palabra código de C si y solo si $c(\alpha_i) = 0$ para todo $i = 1,\ldots, r$.

Demostración. Si *c(x)* es una palabra código de *C*, entonces existe un polinomio *f(x)* tal que $c(x) = g(x) f(x)$. Por lo tanto, tenemos que $c(\alpha_i) = g(\alpha_i) f(\alpha_i) = 0$ para todo $i=1,\ldots, r$. De otra manera, si $c(\alpha_i) = 0$ para toda $i = 1,\ldots, r$, entonces *c(x)* es divisible por *g(x)* y como *g(x)* es irreducible ya que g(x) no tiene raíces múltiples, esto significa que *c(x)* es una palabra código de *C.*

Ejemplo 4.1.2.8 Consideramos el código de Hamming [7, 4] binario con polinomio generador $g(x) = 1 + x + x^3$. Como todos los elementos de $\boldsymbol{F}_8\backslash\{0, 1\}$ son raíces de $c(x) = 1+x+x^2+x^3+x^4+x^5+x^6 = (x^7-1)/(x-1)$, todas las raíces de *g(x)* también son raíces de *c(x).* De hecho, 1111111 es una palabra código.

A continuación el siguiente teorema explica el término "diseño de distancia".

Teorema 4.1.2.9 Un código BCH con distancia de diseño δ tiene una distancia mínima de al menos δ.

Demostración. Sea $\boldsymbol{\alpha}$ un elemento primitivo de F_{q^m} y sea *C* un código BCH generado por $g(x) := mcm(M^{(a)}(x), M^{(a+1)}(x),\ldots,M^{(a+\delta-2)}(x))$. Es evidente que los elementos $\alpha^a,\ldots, \alpha^{a+\delta-2}$ son raíces de *g(x).*

Supóngase que la distancia mínima *d* de *C* es menor que δ. Entonces existe una palabra código diferente de cero $c(x)= c_o + c_1x + \cdots + c_{n-1}x^{n-1}$ tal que $wt(c(x)) = d < \delta$. Por el lema 4.1.2.7, tenemos que $c(\boldsymbol{\alpha}^i) = 0$ para todo $i= a,\ldots, a + \delta - 2$; i.e.,

$$\begin{pmatrix} 1 & \alpha^a & (\alpha^a)^2 & \cdots & (\alpha^a)^{n-1} \\ 1 & \alpha^{a+1} & (\alpha^{a+1})^2 & \cdots & (\alpha^{a+1})^{n-1} \\ 1 & \alpha^{a+2} & (\alpha^{a+2})^2 & \cdots & (\alpha^{a+2})^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \alpha^{a+\delta-2} & (\alpha^{a+\delta-2})^2 & \cdots & (\alpha^{a+\delta-2})^{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = 0.$$

Suponemos *c(x)* es $R = \{i_1,\ldots, i_d\}$, i.e., $c_j \neq 0$ si y solo si $j \in R$. Entonces tenemos que

$$\begin{pmatrix} (\alpha^a)^{i_1} & (\alpha^a)^{i_2} & (\alpha^a)^{i_3} & \cdots & (\alpha^a)^{i_d} \\ (\alpha^{a+1})^{i_1} & (\alpha^{a+1})^{i_2} & (\alpha^{a+1})^{i_3} & \cdots & (\alpha^{a+1})^{i_d} \\ (\alpha^{a+2})^{i_1} & (\alpha^{a+2})^{i_2} & (\alpha^{a+2})^{i_3} & \cdots & (\alpha^{a+2})^{i_d} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ (\alpha^{a+\delta-2})^{i_1} & (\alpha^{a+\delta-2})^{i_2} & (\alpha^{a+\delta-2})^{i_3} & \cdots & (\alpha^{a+\delta-2})^{i_d} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ c_{1_3} \\ \vdots \\ c_{i_d} \end{pmatrix} = 0.$$

Como $d \leq \delta - 1$, se obtiene el siguiente sistema de ecuaciones. Elegimos las primeras d ecuaciones y tendremos lo siguiente:

$$\begin{pmatrix} (\alpha^{a})^{i_1} & (\alpha^{a})^{i_2} & (\alpha^{a})^{i_3} & \cdots & (\alpha^{a})^{i_d} \\ (\alpha^{a+1})^{i_1} & (\alpha^{a+1})^{i_2} & (\alpha^{a+1})^{i_3} & \cdots & (\alpha^{a+1})^{i_d} \\ (\alpha^{a+2})^{i_1} & (\alpha^{a+2})^{i_2} & (\alpha^{a+2})^{i_3} & \cdots & (\alpha^{a+2})^{i_d} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ (\alpha^{a+d-1})^{i_1} & (\alpha^{a+d-1})^{i_2} & (\alpha^{a+d-1})^{i_3} & \cdots & (\alpha^{a+d-1})^{i_d} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ c_{1_3} \\ \vdots \\ c_{i_d} \end{pmatrix} = 0.$$

El determinante D de la matriz de coeficientes de las ecuaciones d es igual a:

$$D = \prod_{j=1}^{d} (\alpha^{a})^{i_j} \det \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \alpha^{i_3} & \cdots & \alpha^{i_d} \\ (\alpha^{2})^{i_1} & (\alpha^{2})^{i_2} & (\alpha^{2})^{i_3} & \cdots & (\alpha^{2})^{i_d} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ (\alpha^{d-1})^{i_1} & (\alpha^{d-1})^{i_2} & (\alpha^{d-1})^{i_3} & \cdots & (\alpha^{d-1})^{i_d} \end{pmatrix}$$

$$= \prod_{j=1}^{d} (\alpha^{a})^{i_j} \prod_{k>l} (\alpha^{i_k} - \alpha^{i_l}) \neq 0.$$

Esta última matriz es una matriz Vandermonde[19]. Para que se cumpla nuestra demostración la determinante deberá de ser igual a cero y en una matriz Vandermonde solo puede ser cero si $\alpha^{i_j} = \alpha^{i_k}$.

De hecho, los elementos del segundo renglón son todos distintos. Por lo tanto hay una contradicción a nuestra demostración. Concluimos que una palabra código con d < δ no puede existir.

Ejemplo 4.1.2.10 Sea α una raíz de $1 + x + x^3 \in \mathbf{F}_2[x]$, y sea C el código BCH binario de longitud 7 con distancia de diseño 4 generado por

$$g(x) = mcm(M^{(0)}(x), M^{(1)}(x), M^{(2)}(x)) = 1 + x^2 + x^3 + x^4.$$

Entonces $d(C) \leq wt(g(x)) = 4$. De otra manera, por el teorema 4.1.2.9, tenemos que $d(C) \geq 4$. Por lo tanto, $d(C) = 4$.

Ejemplo 4.1.2.11 Sea α una raíz de $1 + x + x^4 \in \mathbf{F}_2[x]$. Entonces α es un elemento primitivo de $\mathbf{F}_{16}$. Si consideramos el código BCH binario "narrow sense" de longitud 15 con distancia de diseño 7. Entonces el polinomio generador es:

$$\begin{aligned} g(x) &= mcm(M^{(1)}(x), M^{(2)}(x), M^{(3)}(x), M^{(4)}(x), M^{(5)}(x), M^{(6)}(x)) \\ &= M^{(1)}(x)M^{(3)}(x)M^{(5)}(x) \\ &= 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}. \end{aligned}$$

Por lo tanto, $d(C) \leq wt(g(x)) = 7$. De otra manera, por el teorema 4.1.2.9, tenemos que $d(C) \geq 7$. Por lo tanto, d (C) = 7.

[19] El lector puede consultar la definición de una matriz Vandermonde en [52; 117].

4.1.3 DECODIFICACIÓN DE UN CÓDIGO BCH.

Una decodificación algebraica para los códigos BCH tiene los siguientes pasos:

1. Cálculo del síndrome.
2. Determinar el polinomio localizador de error. Se puede decir que la raíces indican donde esta el error, existen muchos métodos para determinar el polinomio localizador de error los cuales incluyen el algoritmo de Peterson's para códigos BCH, el algoritmo de Berlekamp – Massey, algoritmo de Euclides, entre otros.
3. Encontrar las raíces del polinomio localizador de error.
   El uso de Chien search (exhaustiva búsqueda sobre los elementos en el campo) será tratado en esta tesis.

Por simplicidad, analizaremos únicamente la decodificación de códigos BCH binarios (narrow sense). Sea C un código BCH binario (narrow sense) de longitud $n=2^m-1$ con distancia designada $\delta=2t+1$ generado por $g(x):= mcm(M^{(1)}(x), M^{(2)}(x),\ldots, M^{(\delta-1)}(x))$, donde $M^{(i)}(x)$ es el polinomio mínimal de α^i con respecto a $\mathbf{F}_2$ para un elemento primitivo de α de F_{2^m}. Con

$$H=\begin{pmatrix} 1 & \alpha & (\alpha)^2 & \cdots & (\alpha)^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & \cdots & (\alpha^3)^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \alpha^{\delta-1} & (\alpha^{\delta-1})^2 & \cdots & (\alpha^{\delta-1})^{n-1} \end{pmatrix}. \qquad (4.1)$$

Podemos demostrar que una palabra $\boldsymbol{c} \in F_2^n$ es una palabra código de C si y solo si $cH^T = \mathbf{0}$. Por lo tanto, podemos definir al síndrome $S_H(\mathbf{w})$ de una palabra $\mathbf{w} \in F_2^n$ con respecto a H por wH^T.

Suponemos que $w(x) = w_0 + w_1x + \cdots + w_{n-1}x^{n-1}$ es una palabra recibida con el polinomio de error $e(x)$ con $wt(e(x)) \leq t$. Siendo $c(x) = w(x) - e(x)$, entonces $c(x)$ es una palabra código.

Paso 1: Cálculo de los síndromes. El síndrome de $w(x)$ es

$$(s_0, s_1,\ldots, s_{\delta-2}) := (w_0, w_1,\ldots, w_{n-1})H^T.$$

Es claro que $s_i = w(\boldsymbol{\alpha}^{i+1}) = e(\boldsymbol{\alpha}^{i+1})$ para toda $i = 0, 1,\ldots, \delta - 2$, siendo $\boldsymbol{\alpha}^{i+1}$ raíces de $g(x)$.

Considerando que los errores tienen las posiciones $i_0, i_1,\ldots, i_{l-1}$ con $l \leq t$; i.e.,

$$e(x) = x^{i_0} + x^{i_1} + \cdots + x^{i_{l-1}}. \qquad (4.2)$$

Entonces, se obtiene el siguiente sistema de ecuaciones

$$\begin{array}{ccccccccccc} \alpha^{i_0} & + & \alpha^{i_1} & + & \cdots & + & \alpha^{i_l} & = & s_0 & = & w(\alpha), \\ (\alpha^{i_0})^2 & + & (\alpha^{i_1})^2 & + & \cdots & + & (\alpha^{i_{l-1}})^2 & = & s_1 & = & w(\alpha^2), \\ & & \vdots & & \vdots & & \vdots & & \vdots & & \\ (\alpha^{i_0})^{\delta-1} & + & (\alpha^{i_1})^{\delta-1} & + & \cdots & + & (\alpha^{i_{l-1}})^{\delta-1} & = & s_{\delta-2} & = & w(\alpha^{\delta-1}). \end{array} \quad (4.3)$$

Cualquier método que resuelve el anterior sistema de ecuaciones es un algoritmo de decodificación para códigos BCH.

Paso 2: Encontrar el polinomio localizador de error.

Método 1.- Para

$$e(x) = x^{i_0} + x^{i_1} + \cdots + x^{i_{l-1}},$$

definimos el polinomio localizador del error por

$$\sigma(z) := \prod_{j=0}^{l-1} (1 - \alpha^{i_j} z).$$

Consideramos que la posición del error i_j puede ser encontrado a lo largo de todas las raíces de σ (z). Para este paso, se necesita encontrar el polinomio localizador de error σ(z). Suponemos que el síndrome polinomial es

$$s(z) = \sum_{j=0}^{\delta-2} s_j z^j, \quad (4.4)$$

si s(z) es un polinomio diferente de cero. Entonces existe un polinomio diferente de cero $r(z) \in F_{2^m}[z]$ tal que grad $(r(z)) \leq t-1$, $mcd(r(z), \sigma(z)) = 1$ y

$$r(z) \equiv s(z)\,\sigma(z) (\mathrm{mod}\ z^{\delta-1}). \quad (4.5)$$

Además, para cualquier par $(u(z), v(z))$ de polinomios diferentes de cero sobre F_{2^m} que satisfacen grad $u(z) \leq t-1$, grad $(v(z)) \leq t$ y

$$u(z) \equiv s(z)v(z) (\mathrm{mod}\ z^{\delta-1}), \quad (4.6)$$

tenemos que

$$\sigma(z) = \beta v(z), \qquad r(z) = \beta u(z), \quad (4.7)$$

para un elemento diferente de cero $\beta \in F_{2^m}$.

Para determinar el polinomio localizador de error σ (z), es suficiente resolver la congruencia polinomial (4.5). Este puede ser realizado por el algoritmo de Euclides. En este apartado presentaremos un ejercicio resuelto por el algoritmo de Euclides. Para un mejor entendimiento de su desarrollo, se invita al lector a prácticar previamente ejercicios similares en [41; 20] y como un complemento [42; 18].

Método 2.- Resolver el sistema de ecuaciones dada en la ecuación (4.3) (llamada funciones de suma de potencias simétricas) implica 2t ecuaciones y la posición de t desconocida y se resuelve como un sistema de ecuaciones no lineal. El problema sería tratado convenientemente introduciendo el polinomio localizador de error definido como

$$\Lambda(x) = \Lambda_t(x)^t + \Lambda_{t-1}(x)^{t-1} + \cdots + \Lambda_1(x) + \Lambda_0,$$

donde $\Lambda_0=1$ y las raíces de el polinomio localizador de error son los recíprocos de los localizadores del error. Para llegar a $\Lambda(x)$, tomamos en cuenta lo siguiente: El síndrome de la ecuación (4.4) y los coeficientes del polinomio localizador de error están relacionados para un entero k por:

$$\begin{aligned} S_k + \Lambda_1 S_{k-1} + \cdots + \Lambda_{k-1} S_1 + k\Lambda_k = 0 \qquad & \text{para} \quad 1 \le k \le t \\ S_k + \Lambda_1 S_{k-1} + \cdots + \Lambda_{t-1} S_{k-t+1} + \Lambda_t S_{k-t} = 0 \qquad & \text{para} \quad k > t. \end{aligned} \tag{4.8}$$

Es decir, para t errores tenemos:

$$\begin{aligned} &k=1: S_1 + \Lambda_1 = 0 \\ &k=2: S_2 + \Lambda_1 S_1 + 2\Lambda_2 = 0 \\ &\quad\vdots \\ &k=t: S_t + \Lambda_1 S_{t-1} + \Lambda_2 S_{t-2} + \cdots + \Lambda_{t-1} S_1 + t\Lambda_t = 0 \\ &\quad\vdots \\ &k=2t: S_{2t} + \Lambda_1 S_{2t-1} + \Lambda_2 S_{2t-2} + \cdots + \Lambda_t S_{2t-k} = 0 \end{aligned} \tag{4.9}$$

La relación entre los síndromes y los coeficientes del polinomio localizador de error para k > t es:

$$S_j = -\sum_{i=1}^{t} \Lambda_i S_{j-i} \tag{4.10}$$

La ecuación 4.8 puede ser expresado en una matriz de la siguiente forma:

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_t \\ S_2 & S_3 & \cdots & S_{t+1} \\ S_3 & S_4 & \cdots & S_{t+2} \\ \vdots & \vdots & \vdots & \vdots \\ S_t & S_{t+1} & \cdots & S_{2t-1} \end{bmatrix} \begin{bmatrix} \Lambda_t \\ \Lambda_{t-1} \\ \Lambda_{t-2} \\ \vdots \\ \Lambda_1 \end{bmatrix} = - \begin{bmatrix} S_{t+1} \\ S_{t+2} \\ \vdots \\ \vdots \\ S_{2t} \end{bmatrix}. \tag{4.11}$$

La matriz anterior se le conoce como una matriz Toeplitz de t x t. La solución de la matriz anterior se le conoce como el algoritmo de Peterson-Gorenstein-Zierler y resuelve para los coeficientes $\Lambda_1, \Lambda_2, \ldots, \Lambda_t$.

Para pequeños números de errores, se pueden aplicar fórmulas explícitas para los coeficientes de $\Lambda(x)$, los cuales pueden ser más eficientes son sugeridos por Peterson [53; 252] y se presentan a continuación:

Una corrección de error: $\Lambda_1 = S_1$.

Dos correcciones de error: $\Lambda_1 = S_1$, $\Lambda_2 = \dfrac{S_3 + S_1^3}{S_1}$.

Tres correcciones de error: $\Lambda_1 = S_1$,

$$\Lambda_2 = \frac{(S_3 + S_1^3 + S_5)}{(S_1^3 + S_3)}, \qquad \Lambda_3 = (S_1^3 + S_3) + S_1\Lambda_2.$$

Cuatro correcciones de error: $\Lambda_1 = S_1$, $\quad \Lambda_2 = \dfrac{S_1(S_7 + S_1^7) + S_3(S_1^5 + S_5)}{S_3(S_1^3 + S_3) + S_1(S_1^5 + S_5)}$,

$$\Lambda_3 = S_1^3 + S_3 + S_1\Lambda_2, \qquad \Lambda_4 = \frac{(S_5 + S_1^2 S_3) + (S_1^3 + S_3)\Lambda_2}{S_1}.$$

Nota: Para largos números de corrección, el algoritmo de Peterson es muy complejo ver [53; 252].

Método 3.- El algoritmo de Peterson mostrado en el método 2 involucra un fuerte desarrollo del algebra lineal para resolver la matriz 4.11. A continuación presentaremos el algoritmo de Berlekamp Massey el cual tiene una aplicación que comienza con un pequeño problema, e incrementa el problema hasta obtener una completa solución. Donde la complejidad computacional del algoritmo de Peterson es $O(t^3)$ el algoritmo de Berlekamp Massey es $O(t^2)$ y es dado por el siguiente pseudocódigo:

ALGORITMO 4.1
Decodificación: Berlekamp-Massey

Entradas: $S_1, S_2, ..., S_N$.
Inicio:
L=0 (longitud de registros lineales)
c(x)= 1 (polinomio de entrada)
p(x)= 1 (polinomio antes del último cambio de longitud)
l= 1 (l es k – m, la cantidad de cambio que se actualiza)
d_m= 1 (valor previo de la discrepancia)
for k= 1 to N

$d = S_k + \sum_{i=1}^{L} c_i S_{k-i}$ *(valor de la discrepancia)*
if *(d=0) (no hay cambio de polinomio)*
l = l + 1
else
if *(2 L ≥ k) (Entonces, el polinomio no cambia de longitud)*
$c(x) = c(x) - dd_m^{-1} x^l p(x)$
l = l + 1
else (actualizamos c(x))
t(x)= c(x) (cambio temporal)
$c(x) = c(x) - dd_m^{-1} x^l p(x)$
L = k - L
p(x) = t(x)
d_m = d
l = 1
end
end
end

Lema 4.1.3.1 Cuando la secuencia de los símbolos de entrada del algoritmo de Berlekamp-Massey son síndromes de un código BCH binario, Entonces el valor de la discrepancia d_m es igual a cero para valores pares de k, entonces no cambia el polinomio.

Como resultado del lema 4.1.3.1, nunca se actualiza para valores pares de k. Esto simplifica la complejidad del algoritmo a la mitad. El algoritmo para decodificar códigos BCH binarios se presenta a continuación:

ALGORITMO 4.2

Berlekamp-Massey para decodificación de códigos BCH binarios.

Entradas: $S_1, S_2, ..., S_N$, donde $N = 2t$.
Inicio:
L=0 (*longitud de registros lineales*)
c(x)= 1 (polinomio de entrada)
p(x)= 1 (polinomio antes del último cambio de longitud)
l= 1 (l es k – m, la cantidad de cambio que se actualiza)
d_m*= 1 (valor previo de la discrepancia)*
for k= 1 to N (con saltos de 2)

$d = S_k + \sum_{i=1}^{L} c_i S_{k-i}$ *(valor de la discrepancia)*

if *(d=0) (no hay cambio de polinomio)*
$l = l + 1$
else
if *(2* L ≥ *k) (Entonces, el polinomio no cambia de longitud)*

$c(x) = c(x) - dd_m^{-1} x^l p(x)$
$l = l + 1$
else (actualizamos c(x))
t(x)= c(x) (cambio temporal)
$c(x) = c(x) - dd_m^{-1} x^l p(x)$
$L = k - L$
p(x) = t(x)
$d_m = d$
$l = 1$
end
end
l=l+1 (conteo de valores de k)
end

Paso 3: Encontrar las raíces del polinomio localizador de error. Buscaremos todas las posibles raíces evaluando σ (z) a α^i, para todo i = 0, 1, 2,……. Entonces, todas las raíces

$$\alpha^{i_1}, \ldots, \alpha^{i_l}$$

de σ (z) serán encontradas. Por lo tanto, se obtendrá el error polinomial e(x) de la ecuación (4.2).

Ejemplo 4.1.3.2 Sea α una raíz de $g(x) = 1 + x + x^3 \in F_2[x]$. Entonces es un código de Hamming generado por $g(x) = mcm(M^{(1)}(x), M^{(2)}(x))$ con distancia diseñada $\delta=3$. Supóngase que $r(x)= 1 + x + x^2 + x^3$ es una palabra recibida.

Paso 1 Calculamos los síndromes

$$S_1(\alpha) = 1+\alpha+\alpha^2+\alpha^3 = \alpha^2$$
$$S_2(\alpha^2) = 1+\alpha^2+(\alpha^2)^2+(\alpha^3)^2 = \alpha^4$$

Paso 2 Encontrar el polinomio localizador del error. Resolvemos la siguiente congruencia polinomial:

$$r(z) \equiv s(z)\,\sigma(z) (mod\ z^2)$$

con $s(z)= \alpha^2 + \alpha^4 z$ con $\sigma(z)= 1+ \alpha^2 z$.

Otra forma de calcular el polinomio localizador de error es usar la tabla 4.1.

Fila	Q	R	U	V
-1	--	z^2	1	0
0	--	$s(z)$	0	1
1	$\alpha^3 z + \alpha$	α^3	1	$\alpha^3 z + \alpha$

Tabla 4.1 Algoritmo de Euclides para el ejemplo 4.1.3.2.

Paso 3 Encontrar las raíces del polinomio localizador del error.

De la congruencia polinomial $r(z) = \alpha^2$.

De la tabla *4.1* $r(z) = \alpha^2$.

Del algoritmo de Peterson $\Lambda(x) = S_1 = \alpha^2$.

Utilizando la transformación lineal del resultado anterior tenemos que el error se encuentra en

$$x = \alpha^2 = x^2$$
$$c(x) = r(x) - e(x)$$

	1	x	x^2	x^3	x^4	x^5	x^6
$r(x)$	1	1	1	1			
$e(x)$			1				
$c(x)$	1	1		1			

Tabla 4.2 Decodificación del ejemplo 4.1.2.2.

Podemos decir que $c(x) = 1+ x + x^3$ es la palabra enviada.

Ejemplo 4.1.3.3 Sea un código BCH [15, 5] que resulta del polinomio generador $g(x)=1+x+x^2+x^4+x^5+x^8+x^{10} \in F_2[x]$. Entonces el código es generado por $g(x)=M^{(1)}(x)M^{(3)}(x)M^{(5)}(x)$ *y* tiene una distancia de diseño $\delta=7$. Supóngase que $r(x)$=(101111110010000) es una palabra recibida.

Paso 1 Calculamos los síndromes:

$S_1 = r(\alpha)$

$= 1+ \alpha^2+ \alpha^3+ \alpha^4+ \alpha^5+ \alpha^6+ \alpha^7+ \alpha^{10}$

$= 1+ \alpha^2+ \alpha^3+ \alpha+1+ \alpha^2+\alpha+ \alpha^3+ \alpha^2+ \alpha^3+ \alpha+1+ \alpha^2+\alpha+1$

$= \alpha^3$

$S_2 = r(\alpha^2) = (r(\alpha^2))^2 = (\alpha^3)^2 = \alpha^6$
$S_3 = r(\alpha^3)$
$= 1 + \alpha^6 + \alpha^9 + \alpha^{12} + \alpha^{15} + \alpha^{18} + \alpha^{21} + \alpha^{30}$
$= 1 + \alpha^6 + \alpha^9 + \alpha^{12} + 1 + \alpha^3 + \alpha^6 + 1$
$= \alpha^3 + \alpha + \alpha^3 + \alpha^2 + \alpha + 1 + \alpha^3 + 1$
$= \alpha^6$
$S_4 = r(\alpha^4) = (r(\alpha))^4 = (\alpha^3)^4 = \alpha^{12}$
$S_5 = r(\alpha^5)$
$= 1 + \alpha^{10} + \alpha^{15} + \alpha^{20} + \alpha^{25} + \alpha^{30} + \alpha^{35} + \alpha^{50}$
$= 1 + \alpha^{10} + 1 + \alpha^5 + \alpha^{10} + 1 + \alpha^5 + \alpha^5$
$= \alpha^2 + \alpha + 1 = \alpha^{10}$
$S_6 = r(\alpha^6) = (r(\alpha^3))^2 = (\alpha^6)^2 = \alpha^{12}$

Paso 2 Encontrar el polinomio localizador del error. Por lo tanto, como $r(\alpha)$ corrige tres errores debemos encontrar σ_1, σ_2 y σ_3.

$$\begin{bmatrix} \alpha^3 & \alpha^6 & \alpha^6 \\ \alpha^6 & \alpha^6 & \alpha^{12} \\ \alpha^6 & \alpha^{12} & \alpha^{10} \end{bmatrix} \begin{bmatrix} \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^{12} \\ \alpha^{10} \\ \alpha^{12} \end{bmatrix}$$

Después, utilizamos la regla de Cramer para determinar σ_1, σ_2 y σ_3.

$$\begin{vmatrix} \alpha^3 & \alpha^6 & \alpha^6 \\ \alpha^6 & \alpha^6 & \alpha^{12} \\ \alpha^6 & \alpha^{12} & \alpha^{10} \end{vmatrix}$$
$= \alpha^{19} + \alpha^{24} + \alpha^{24} + \alpha^{18} + \alpha^{27} + \alpha^{22}$
$= \alpha^4 + \alpha^3 + \alpha^{12} + \alpha^7$
$= \alpha + 1 + \alpha^3 + \alpha^3 + \alpha^2 + \alpha + 1\ \alpha^3 + \alpha + 1$
$= \alpha^{12}$

$$\begin{vmatrix} \alpha^{12} & \alpha^6 & \alpha^6 \\ \alpha^{10} & \alpha^6 & \alpha^{12} \\ \alpha^{12} & \alpha^{12} & \alpha^{10} \end{vmatrix}$$
$= \alpha^{28} + \alpha^{30} + \alpha^{28} + \alpha^{24} + \alpha^{36} + \alpha^{26}$
$= 1 + \alpha^9 + \alpha^6 + \alpha^{11}$
$= 1 + \alpha^3 + \alpha + \alpha^3 + \alpha^2 + \alpha^3 + \alpha^2 + \alpha$
$= \alpha^{14}$

$$\sigma_3 = \frac{\alpha^{14}}{\alpha^{12}} = \alpha^2$$

$$\begin{vmatrix} \alpha^3 & \alpha^{12} & \alpha^6 \\ \alpha^6 & \alpha^{10} & \alpha^{12} \\ \alpha^6 & \alpha^{12} & \alpha^{10} \end{vmatrix}$$
$= \alpha^{23} + \alpha^{30} + \alpha^{24} + \alpha^{22} + \alpha^{27} + \alpha^{28}$
$= \alpha^8 + 1 + \alpha^9 + \alpha^7 + \alpha^{12} + \alpha^{13}$
$= \alpha^{10}$

$$\sigma_2 = \frac{\alpha^{10}}{\alpha^{12}} = \alpha^{13}$$

$$\begin{vmatrix} \alpha^3 & \alpha^{12} & \alpha^6 \\ \alpha^6 & \alpha^{10} & \alpha^{12} \\ \alpha^6 & \alpha^{12} & \alpha^{10} \end{vmatrix}$$
$= \alpha^{21} + \alpha^{22} + \alpha^{30} + \alpha^{24} + \alpha^{25} + \alpha^{24}$
$= \alpha^6 + \alpha^7 + 1 + \alpha^{10}$
$= \alpha^3 + \alpha^2 + \alpha^3 + \alpha + 1 + 1 + \alpha^2 + \alpha + 1$
$= 1$

$$\sigma_1 = \frac{1}{\alpha^{12}} = \alpha^3$$

Observar que si utilizamos el algoritmo de Peterson se tienen los mismos resultados:

$\Lambda_1 = \alpha^3$

$$\Lambda_2 = \frac{\left(S_1^2 S_3 + S_5\right)}{S_1^3 + S_3} = \frac{\alpha^6 \alpha^6 + \alpha^{10}}{\alpha^9 + \alpha^6} = \frac{\alpha^{12} + \alpha^{10}}{\alpha^9 + \alpha^6} = \frac{\alpha^3}{\alpha^5} = \alpha^{-2} = \alpha^{13}$$

$$\Lambda_3 = \left(S_1^3 + S_3\right) + S_1\Lambda_2 = \left(\alpha^9 + \alpha^6\right) + \alpha^3\alpha^{13} = \alpha^3 + \alpha + \alpha^3 + \alpha^2 + \alpha = \alpha^2$$

k	S_k	d_k	c(x)	L	p(x)	*l*	d	t(x)
1	α^3	α^3	$1+\alpha^3x$	1	1	2	α^3	1
3	α^6	α^5	$1+\alpha^3x+\alpha^2x^2$	2	$1+\alpha^3x$	2	α^5	$1+\alpha^3x$
5	α^{10}	α^4	$1+\alpha^3x+\alpha^{13}x^2+\alpha^2x^3$	3	$1+\alpha^3x+\alpha^2x^2$	2	α^4	$1+\alpha^3x+\alpha^2x^2$

Tabla 4.3 Algoritmo computacional Berlekamp–Massey para decodificar el código BCH [15, 5] binario que corrige tres errores para el Ejemplo 4.1.3.3.

El polinomio localizador de error es el siguiente:

$$E(z) = z^3 + \alpha^3 z^2 + \alpha^{13} z + \alpha^2.$$

Paso 3 Encontrar las raíces del polinomio localizador del error.

$$E(1) = z^3 + \alpha^3 z^2 + \alpha^{13} z + \alpha^2 = 0.$$
$$E(\alpha^5) = z^3 + \alpha^3 z^2 + \alpha^{13} z + \alpha^2 = 0.$$
$$E(\alpha^{12}) = z^3 + \alpha^3 z^2 + \alpha^{13} z + \alpha^2 = 0.$$

	1	x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}
r(x)	1	0	1	1	1	1	1	1	0	0	1	0	0	0	0
e (x)	1					1							1		
c(x)	0	0	1	1	1	0	1	1	0	0	1	0	1	0	0

Tabla 4.4 Decodificación del ejemplo 4.1.3.3.

Por lo tanto, se tiene que $c(x) = x^2+x^3+x^4+x^6+x^7+x^{10}+x^{12}$ y se puede comprobar que c(x) es múltiplo de g(x).

Ejemplo 4.1.3.4 Sea un código BCH [15, 5] que resulta del polinomio generador $g(x)=1+x+x^2+x^4+x^5+x^8+x^{10} \in F_2[x]$. Entonces, el código es generado por $g(x)=M^{(1)}(x)M^{(3)}(x)M^{(5)}(x)$ *y* tiene una distancia de diseño $\delta = 7$. Supóngase que $r(x)=(011111110101000)$ es una palabra recibida.

Paso 1 Calculamos los síndromes:

$S_1 = r(\alpha) = \alpha^3$.
$S_2 = r(\alpha^2) = (r(\alpha^2))^2 = (\alpha^3)^2 = \alpha^6$.
$S_3 = r(\alpha^3) = \alpha^4$.
$S_4 = r(\alpha^4) = (r(\alpha^3))^4 = (\alpha^3)^4 = \alpha^{12}$.
$S_5 = r(\alpha^5) = 0$.
$S_6 = r(\alpha^6) = (r(\alpha^4))^2 = \alpha^8$.

Paso 2 Encontrar el polinomio localizador del error. Por lo tanto como r(α) corrige tres errores debemos encontrar σ_1, σ_2 y σ_3.

$$\begin{bmatrix} \alpha^3 & \alpha^6 & \alpha^4 \\ \alpha^6 & \alpha^4 & \alpha^{12} \\ \alpha^4 & \alpha^{12} & 0 \end{bmatrix} \begin{bmatrix} \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^{12} \\ 0 \\ \alpha^8 \end{bmatrix}$$

Después, utilizamos la regla de Cramer para determinar σ_1, σ_2 y σ_3.

$$\begin{bmatrix} \alpha^3 & \alpha^6 & \alpha^4 \\ \alpha^6 & \alpha^4 & \alpha^{12} \\ \alpha^4 & \alpha^{12} & 0 \end{bmatrix} = 0.$$ Como el determinante da 0, entonces existen 2 errores.

Entonces utilizamos la regla de Cramer para determinar σ_1 y σ_2.

$$\begin{bmatrix} \alpha^3 & \alpha^6 \\ \alpha^6 & \alpha^4 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^4 \\ \alpha^{12} \end{bmatrix} \qquad \begin{bmatrix} \alpha^3 & \alpha^6 \\ \alpha^6 & \alpha^4 \end{bmatrix} = \alpha^2.$$

$$\begin{bmatrix} \alpha^4 & \alpha^6 \\ \alpha^{12} & \alpha^4 \end{bmatrix} = \alpha^{13}, \qquad \begin{bmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^{12} \end{bmatrix} = \alpha^3.$$

Los valores de σ_1 y σ_2 son: $\sigma_2 = \dfrac{\alpha^{13}}{\alpha^2} = \alpha^{11}$ y $\sigma_1 = \dfrac{\alpha^5}{\alpha^2} = \alpha^3$.

k	S_k	d_k	c(x)	L	p(x)	*l*	d	t(x)
1	α^3	α^3	$1+\alpha^3x$	1	1	2	α^3	1
3	α^6	α^5	$1+\alpha^3x+\alpha^{11}x^2$	2	$1+\alpha^3\alpha$	2	α^5	$1+\alpha^3x$

Tabla 4.5 Algoritmo computacional Berlekamp–Massey para decodificar el código BCH [15, 5] binario que corrige dos errores para el Ejemplo 4.1.3.4.

El polinomio localizador de error es el siguiente:

$$E(z) = z^2 + \boldsymbol{\alpha}^3 z + \boldsymbol{\alpha}^{11}.$$

Paso 3 Encontrar las raíces del polinomio localizador del error.

$$E(\boldsymbol{\alpha}^4) = z^2 + \boldsymbol{\alpha}^3 z + \boldsymbol{\alpha}^{11} = 0.$$
$$E(\boldsymbol{\alpha}^7) = z^2 + \boldsymbol{\alpha}^3 z + \boldsymbol{\alpha}^{11} = 0.$$

	1	x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}
r(x)	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0
e (x)					1			1							
c(x)	0	1	1	1	0	1	1	0	0	1	0	1	0	0	0

Tabla 4.6 Decodificación del ejemplo 4.1.3.4.

Por lo tanto, se tiene que $c(x) = x + x^2 + x^3 + x^5 + x^6 + x^9 + x^{11}$ y se puede comprobar que c(x) es múltiplo de g(x).

4.2 CÓDIGOS REED SOLOMON.

En agosto y septiembre de 1977, la NASA lanzó los satélites Voyager 1 y 2 desde Cabo Cañaveral, en Florida. Sus destinos iniciales eran Júpiter y Saturno, los satélites Voyager proporcionaron a la NASA imágenes de estos planetas y sus lunas que habían sido observadas. En enero de 1986, después de dejar Júpiter y Saturno, el Voyager 2 continúo su recorrido en nuestro sistema solar para transmitir éxitosamente datos e imágenes desde Urano y Néptuno duplicando aproximadamente la distancia hacia la Tierra, con lo que podemos decir que también se duplicaron los errores en la transmisión de bits. Sin el uso de un código Reed Solomon en la transmisión de estas imágenes proporcionadas por el Voyager 2, este no podría haber tenido éxito.

Fotografías trasmitidas desde el espacio hacia la Tierra son usualmente digitalizadas dentro de cadenas binarias y enviadas sobre un canal espacial. El Voyager 2 digitalizó sus imágenes a color en cadenas binarias de longitud 15,360,000 con velocidad alrededor de 44,800 bits por segundo, usando un complejo sistema de telecomunicaciones, estos bits fueron transmitidos uno a uno de regreso a la Tierra, donde las imágenes fueron después reconstruidas. Después de un considerable estudio, fue descubierto que los bits de error que ocurren en largas transmisiones a través del espacio generalmente ocurren en bursts o ráfaga. Para corregir estos errores en forma de bursts un nuevo sistema fue diseñado para que el Voyager 2 convirtiera imágenes dentro de cadenas binarias que utilizarán el método de corrección de error Reed Solomon. El proceso utilizando códigos Reed Solomon en la transmisión de datos fue todo un éxito. Después de dejar Urano, el Voyager 2 continúo su viaje a través del espacio, en agosto de 1989 transmitió datos e imágenes visuales a la Tierra acerca de Néptuno, actualmente el Voyager 2 esta en operación [41].

La más importante subclase de códigos BCH es la clase de códigos Reed Solomon, los cuales fueron descubiertos por I. S. Reed y G. Solomon independientemente del trabajo de A. Hocquenghem, R. C. Bose y D. K. Ray Chaudhuri quienes descubrieron los códigos BCH [48; 171].

4.2.1 CÓDIGOS REED SOLOMON.

Consideramos C como un q-ario código BCH de longitud $q^m - 1$ generado por $g(x) := \text{mcm}\,(M^{(a)}(x), M^{(a+1)}(x),\ldots,(M^{a+\delta-2)}(x))$, donde $M^{(i)}(x)$ es el polinomio mínimal de α^i con respecto a $\mathbf{F}_q$ para un elemento primitivo α de F_{q^m}. Si m=1, obtenemos un q-ario código BCH de longitud q - 1. En este caso, $\boldsymbol{\alpha}$ es un elemento primitivo F_q y, además, el polinomio mínimal de $\boldsymbol{\alpha}^i$ con respecto a $\mathbf{F}_q$ es $x - \alpha^i$. Es decir, para $\delta \leq q - 1$, el polinomio generador es:

$$g(x) = \text{mcm}\,(x - \boldsymbol{\alpha}^a, x - \boldsymbol{\alpha}^{a+1}, \ldots, x - \boldsymbol{\alpha}^{a+\delta-2})$$
$$= (x - \boldsymbol{\alpha}^a), (x - \boldsymbol{\alpha}^{a+1}) \cdots (x - \boldsymbol{\alpha}^{a+\delta-2})$$

desde que $\boldsymbol{\alpha}^a, \boldsymbol{\alpha}^{a+1},\ldots, \boldsymbol{\alpha}^{a+\delta-2}$ son parejas distintas.

Definición 4.2.1.1 Un q-ario código Reed Solomon es un *q–ario* código BCH de longitud q – 1 generado por:

$$g(x) = (x - \boldsymbol{\alpha}^{a+1})(x - \boldsymbol{\alpha}^{a+2}) \cdots (x - \boldsymbol{\alpha}^{a+\delta-1}),$$

con $\mathbf{a} \geq 0$ y $2 \leq \boldsymbol{\delta} \leq q - 1$, donde $\boldsymbol{\alpha}$ es un elemento primitivo de $\mathbf{F}_q$. Observar que no se considera un código Reed Solomon binario, ya que la longitud es q – 1 = 1.

Ejemplo 4.2.1.2 Consideramos un 7-ario código Reed Solomon de longitud 6 con un polinomio generador $g(x) = (x - 3)(x - 3^2)(x - 3^3) = 6 + x + 3x^2 + x^3$. Este es un 7-ario código [6, 3]. La matriz generadora g(x) tiene la siguiente forma:

$$G = \begin{pmatrix} 6 & 1 & 3 & 1 & 0 & 0 \\ 0 & 6 & 1 & 3 & 1 & 0 \\ 0 & 0 & 6 & 1 & 3 & 1 \end{pmatrix}.$$

La matriz parity-check es formada de $h(x) = (x^6 - 1)/g(x) = 1 + x + 4x^2 + x^3$, con $h_R = 1 + 4x + x^2 + x^3$.

$$H = \begin{pmatrix} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 & 1 \end{pmatrix}.$$

Puede comprobarse a partir de la matriz parity check que la distancia mínima es 4. Por lo tanto, este es un 7-ario código [6, 3, 4]-MDS[20].

Ejemplo 4.2.1.3 Consideramos un 8-ario código Reed Solomon de longitud 7 con un polinomio generador $g(x) = (x - \alpha)(x - \alpha^2) = 1 + \alpha + (\alpha^2 + \alpha)x + x^2$, donde α es una raíz de $1 + x + x^3 \in \mathbf{F}_2[x]$. Este es un 8-ario código [7, 5], con la siguiente matriz generadora formada por g(x):

$$G = \begin{pmatrix} \alpha+1 & \alpha^2+\alpha & 1 & 0 & 0 & 0 & 0 \\ 0 & \alpha+1 & \alpha^2+\alpha & 1 & 0 & 0 & 0 \\ 0 & 0 & \alpha+1 & \alpha^2+\alpha & 1 & 0 & 0 \\ 0 & 0 & 0 & \alpha+1 & \alpha^2+\alpha & 1 & 0 \\ 0 & 0 & 0 & 0 & \alpha+1 & \alpha^2+\alpha & 1 \end{pmatrix}.$$

Su matriz parity check:

$$H = \begin{pmatrix} 1 & \alpha^4 & 1+\alpha^4 & 1+\alpha^4 & 1+\alpha^4 & \alpha^4 & 0 \\ 0 & 1 & 1 & 1 & 1+\alpha^4 & 1+\alpha^4 & \alpha^4 \end{pmatrix}$$

se obtiene de $h(x) = (x^7 - 1)/g(x) = \alpha^4 + (1 + \alpha^4)x + (1 + \alpha^4)x^2 + x^3 + \alpha^4 x^4 + x^5$. Puede comprobarse a partir de la matriz parity check que la distancia mínima es 3. Por lo tanto, este es un código 8-ario [7, 5, 3] MDS.

Teorema 4.2.1.4 Los códigos Reed Solomon son MDS; i.e., un q-ario código Reed-Solomon de longitud q - 1 generado por

$$g(x) = \prod_{i=a+1}^{a+\delta-1} (x - \alpha^i)$$

es un código cíclico [q - 1, q - δ, δ] para cualquier $2 \le \delta \le q - 1$.

Demostración. Como el grado de g(x) es δ – 1, la dimensión del código es exactamente k := q – 1 – (δ - 1) = q – δ. Por el teorema 4.1.2.9 la distancia mínima es al menos δ. De otra manera, la distancia mínima es a lo máximo (q – 1) + 1 – k = δ [ver cota de Singleton en el Apéndice (A)].

[20] Los códigos MDS (máxima distancia separable) y la cota de Singleton se describen en el Apéndice (A).

Ejemplo 4.2.1.5 Sea α una raíz de $1 + x + x^4 \in \mathbf{F}_2[x]$. Entonces, α es un elemento primitivo de $\mathbf{F}_{16}$. Consideramos el código Reed Solomon generado por $g(x) = (x - \alpha^3)(x - \alpha^4)(x - \alpha^5)(x - \alpha^6)$. Observamos que no es fácil obtener la distancia mínima a partir de su matriz parity check. Sin embargo, por el teorema 4.2.1.4, Este es un código cíclico [15, 11, 5] sobre $\mathbf{F}_{16}$.

Definición 4.2.1.6 Sea C un q-ario código Reed Solomon generado por

$$g(x) = \prod_{i=1}^{\delta-1} \left(x - \alpha^i\right)$$

con $2 \leq \delta \leq q - 1$. Entonces, el código extendido $\overline{C}$ es también MDS.

Ejemplo 4.2.1.7 Consideramos el 7-ario código Reed Solomon [6, 3, 4]. El cual tiene la siguiente matriz parity check de código extendido

$$\begin{pmatrix} 1 & 4 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Por lo tanto, por el Corolario 2.2.4.6, el código extendido tiene una distancia mínima 5, y de hecho es un código [7, 3, 5] MDS.

Ejemplo 4.2.1.8 Consideramos el 8-ario código Reed Solomon [7, 5, 3]. El cual tiene la siguiente matriz parity check de código extendido.

$$\begin{pmatrix} 1 & \alpha^4 & 1 & 1+\alpha^4 & 1+\alpha^4 & \alpha^4 & 0 & 0 \\ 0 & 1 & \alpha^4 & 1 & 1+\alpha^4 & 1+\alpha^4 & \alpha^4 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Por lo tanto, por el Corolario 2.2.4.6, el código extendido tiene una distancia mínima 4, de hecho es un código [8, 5, 4] MDS.

Como se demostró anteriormente, los códigos Reed Solomon son MDS. Por lo tanto, tienen muy buenos parámetros. Desafortunadamente, Reed Solomon son códigos no binarios, mientras que para la práctica generalmente son utilizados los códigos binarios. A continuación se presenta la técnica de concatenación la cual se usa para producir códigos binarios a partir de los códigos Reed Solomon sobre extensiones del campo $\mathbf{F}_2$.

Sea C un código Reed Solomon [n, k] sobre F_{q^m}, donde $n = 2^m - 1$. Aplicando la técnica de la concatenación, concatenamos C con el código trivial F_2^m. Sea $\alpha_1, \ldots, \alpha_m$ una base en $\boldsymbol{F}_2$ de F_{2^m} y consideramos la transformación ϕ como:

$$\phi : F_{2^m} \rightarrow F_2^{\,m}.$$

donde

$$u_1\alpha_1 + u_2\alpha_2 + \cdots + u_m\alpha_m \mapsto (u_i, u_2, \ldots, u_m).$$

Teorema 4.2.1.9 Sea C un código Reed Solomon [n, k] sobre F_{2^m}, donde n = 2^m - 1. Entonces

$$\phi*(C):\{(\phi(c_0),...,\phi(c_{n-1})):(c_0,...,c_{n-1})\in C\}$$

es un código binario [mn, mk] con distancia mínima por lo menos n – k + 1.

Ejemplo 4.2.1.10 Consideramos el 8-ario código C Reed Solomon generado por

$$g(x)=\prod_{i=1}^{6}(x-\alpha^i)=\sum_{i=0}^{6}x^i,$$

donde **α** es una raíz de 1 + x + x^3. Por lo tanto, C= {a (1, 1, 1, 1, 1, 1, 1) : a Є $\mathbf{F}_8$} es el 8-ario código trivial [7, 1, 7]-MDS. El código **ø***(C) es un código lineal binario [21, 3, 7] transformado por

100 100 … 100, 010 010 … 010, 001 001 … 001.

Para un código Reed Solomon C, el código **ø***(C) no puede corregir demasiados errores aleatorios debido a que la distancia mínima no es muy grande. Sin embargo, puede corregir muchos errores en forma de burst o ráfaga.

Teorema 4.2.1.11 Sea C un código Reed Solomon [2^m – 1, k] sobre F_{2^m}. Entonces, el código **ø***(C) puede corregir

$$m\lfloor(n-k)/2\rfloor-m+1$$

errores en forma de burst, donde n = 2^m - 1 es la longitud del código.

Demostración. Suponemos $l=m\lfloor(n-k)/2\rfloor-m+1$. Por el teorema 3.4.1.3, es suficiente demostrar que todos los errores en forma de burst de longitud l o menos se encuentran en diferentes co-conjuntos.
Sea e_1, e_2 Є F_2^{mn} dos errores burst de longitud l o menores colocados en el mismo co-conjunto de **ø***(C). Sea $\mathbf{c}_i$ una representación de $\mathbf{e_i}$ bajo la transformación **ø***, i.e,. **ø***($\mathbf{c}_i$) = $\mathbf{e}_i$ para i = 1, 2. Entonces, está claro que:

$$\begin{aligned} wt(c_i) &\le \left\lceil\frac{l-1}{m}\right\rceil+1\\ &= \left\lfloor\frac{n-k}{2}\right\rfloor\\ &= \left\lfloor\frac{d(C)-1}{2}\right\rfloor \end{aligned}$$

(Debido a que C es un código MDS),

Para i = 1, 2, y $\mathbf{c}_1$, $\mathbf{c}_2$ en el mismo co-conjunto de C. Entonces $\mathbf{c}_1$ = $\mathbf{c}_2$ esto significa que $\mathbf{e_1}$ = $\mathbf{e_2}$ debido a que **ø***es inyectiva.

Ejemplo 4.2.1.12 Consideramos el 8-ario código Reed Solomon [7, 3, 5], el código **ø***(C) es un código lineal binario [21, 9]. Puede corregir l errores burst.

$$l=3\left\lfloor\frac{7-3}{2}\right\rfloor-3+1=4.$$

4.2.2 DECODIFICACIÓN DE UN CÓDIGO REED – SOLOMON.

Una decodificación algebraica para los códigos Reed Solomon tiene los siguientes pasos:

1. Cálcular los síndromes.
2. Determinar el polinomio localizador de error. Se puede decir que la raíces indican donde esta el error, existen muchos métodos para determinar el polinomio localizador de error los cuales incluyen el algoritmo de Peterson's con la regla de Cramer, el algoritmo de Berlekamp – Massey, algoritmo de Euclides, entre otros.
3. Encontrar las raíces del polinomio localizador de error. El uso de Chien search (exhaustiva búsqueda sobre los elementos en el campo).
4. Encontrar las posiciones de error in r(x) por encontrar las raíces de V(z). Para encontrar los coeficientes de e(x) en dichas posiciones del error, denotaremos el término x^{-i} en e(x) por e_{-i} en la siguiente fórmula:

$$e_{-i} = \frac{R(a^i)}{V'(\alpha^i)}.$$

Ejemplo 4.2.2.1 Elegimos el polinomio primitivo $f(x) = 1 + x + x^4 \in Z_2[x]$ donde representaremos el campo $F_{q^m} = Z_q[x]/\bmod f(x)$ donde f(x) es un polinomio irreducible (Ver Apéndice ejemplo B.6). Para construir un código Reed – Solomon [15, 2] que corrija 2 errores; i.e., con distancia 5, elegimos el siguiente polinomio generador.

$$\begin{aligned} g(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) \\ &= (x^2+(\alpha^2+\alpha)x+\alpha^3)(\alpha^2+(\alpha^4+\alpha^3)x+\alpha^7) \\ &= (x^2+\alpha^5 x+\alpha^3)(x^2+\alpha^7 x+\alpha^7) \\ &= x^4+(\alpha^7+\alpha^5)x^3+(\alpha^7+\alpha^{12}+\alpha^3)x^2+(\alpha^{12+}\alpha^{10})x+\alpha^{10} \\ &= x^4+\alpha^{13}x^3+\alpha^6 x^2+\alpha^3 x+\alpha^{10} \end{aligned}$$

Por aplicar, la técnica de concatenación descrita anteriormente convertimos el código Reed-Solomon en un código binario. Por el Teorema 4.2.1.4, el código Reed-Solomon [15, 11, 5] y con la concatenación obtenemos el código lineal binario [60, 44, 5], lo que nos da un total de 17592186044416 palabras código.
Una palabra *C* es transmitida y el vector binario recibido es el siguiente:
(0000 0000 0000 1111 0110 0001 1011 0111 0100 0111 0010 1001 1010 1110 0000)

$$\begin{aligned} r(x) &= (1+\alpha+\alpha^2+\alpha^3)x^3+(\alpha+\alpha^2)x^4+\alpha^3x^5+(1+\alpha^2+\alpha^3)x^6+(\alpha+\alpha^2+\alpha^3)x^7+\alpha x^8+ \\ &\quad (\alpha+\alpha^2+\alpha^3)x^9+\alpha^2x^{10}+(1+\alpha^3)x^{11}+(1+\alpha^2)x^{12}+(1+\alpha+\alpha^2)x^{13}. \\ r(x) &= \alpha^{12}x^3+\alpha^5x^4+\alpha^3x^5+\alpha^{13}x^6+\alpha^{11}x^7+\alpha x^8+\alpha^{11}x^9+\alpha^2x^{10}+\alpha^{14}x^{11}+\alpha^8x^{12}+\alpha^{10}x^{13} \end{aligned}$$

A continuación, debido a que *C* corrige *t=2* errores en *r(x)*, podemos calcular los primeros cuatro síndromes de la siguiente manera:
Paso 1 Calcular los síndromes.

$$\begin{aligned} S_1 &= r(\alpha) \\ &= \alpha^{15}+\alpha^9+\alpha^8+\alpha^{19}+\alpha^{18}+\alpha^9+\alpha^{20}+\alpha^{12}+\alpha^{25}+\alpha^{20}+\alpha^{23} \\ &= 1+\alpha^9+\alpha^8+\alpha^4+\alpha^3+\alpha^9+\alpha^5+\alpha^{12}+\alpha^{10}+\alpha^5+\alpha^8 \\ &= 1+\alpha+1+\alpha^3+\alpha^3+\alpha^2+\alpha+1+\alpha^2+\alpha+1 \\ &= \alpha. \end{aligned}$$

$S_2 = r(\alpha^2)$
$= \alpha^{18} + \alpha^{13} + \alpha^{13} + \alpha^{25} + \alpha^{25} + \alpha^{17} + \alpha^{29} + \alpha^{22} + \alpha^{36} + \alpha^{32} + \alpha^{36}$
$= \alpha^3 + \alpha^2 + \alpha^{14} + \alpha^7 + \alpha^2$
$= \alpha^3 + \alpha^3 + 1 + \alpha^3 + \alpha + 1$
$= \alpha^9$.
$S_3 = r(\alpha^3)$
$= \alpha^{21} + \alpha^{17} + \alpha^{18} + \alpha^{31} + \alpha^{32} + \alpha^{25} + \alpha^{38} + \alpha^{32} + \alpha^{22} + \alpha^{47} + \alpha^{44} + \alpha^{49}$
$= \alpha^6 + \alpha^2 + \alpha^3 + \alpha + \alpha^{10} + \alpha^8 + \alpha^2 + \alpha^{14} + \alpha^4$
$= \alpha^3 + \alpha^2 + \alpha^3 + \alpha + \alpha^2 + \alpha + 1 + \alpha^2 + 1 + \alpha^3 + 1 + \alpha + 1$
$= \alpha^{11}$.
$S_4 = r(\alpha^4)$
$= \alpha^{24} + \alpha^{21} + \alpha^{23} + \alpha^{37} + \alpha^{39} + \alpha^{33} + \alpha^{47} + \alpha^{42} + \alpha^{58} + \alpha^{56} + \alpha^{62}$
$= \alpha^9 + \alpha^6 + \alpha^8 + \alpha^7 + \alpha^9 + \alpha^3 + \alpha^2 + \alpha^{12} + \alpha^{13} + \alpha^{11} + \alpha^2$
$= \alpha^3 + \alpha^2 + \alpha^2 + 1 + \alpha^3 + \alpha + 1 + \alpha^3 + \alpha^3 + \alpha^2 + \alpha + 1 + \alpha^3 + \alpha^2 + 1 + \alpha^3 + \alpha^2 + \alpha$
$= \alpha^5$.

Paso 2. Construir la tabla del algoritmo de Euclides[21] para $z^4 = s(z)\, q_1 + r_1$, siendo $S(z) = \alpha + \alpha^9 z + \alpha^{11} z^2 + \alpha^5 z^3$.

Fila	Q	R	U	V
-1	--	z^4	1	0
0	--	S(z)	0	1
1	$\alpha^{10} z + \alpha$	$\alpha^6 z^2 + \alpha^{14} z + \alpha^2$	1	$\alpha^{10} z + \alpha$
2	$\alpha^{14} z + \alpha^{13}$	$\alpha^{10} z + \alpha^4$	$\alpha^{14} z + \alpha^{13}$	$\alpha^9 z^2 + \alpha^2 z + \alpha^3$

Tabla 4.7 Algoritmo de Euclides para el ejemplo 4.1.2.2.

De la tabla 4.7, tenemos que $V(z) = \alpha^9 z^2 + \alpha^2 z + \alpha^3$ y $R(z) = a^{10} z + a^4$

Paso 3. Evaluando en las sucesivas potencias de α para V(z), se obtiene que las raíces de V(z) son α^4 y α^5. Como resultado, las posiciones en r(x) que contiene los errores son $x^{-4} = x^{11}$ y $x^{-5} = x^{10}$.

Paso 4. Para encontrar los coeficientes del error polinomial e(x) de dichas posiciones de error, primero notamos que $V'(z) = \alpha^2$. Después, encontramos los coeficientes e(x), de la siguiente manera:

$$e_{10} = \frac{R(\alpha^5)}{V'(\alpha^5)} = \left(\frac{\alpha^{15} + \alpha^4}{\alpha^2}\right) = \left(\frac{\alpha}{\alpha^2}\right) = \alpha^{14} \qquad y \qquad e_{10} = \frac{R(\alpha^4)}{V'(\alpha^4)} = \left(\frac{\alpha^{14} + \alpha^4}{\alpha^2}\right) = \left(\frac{\alpha^9}{\alpha^2}\right) = \alpha^7$$

Por lo tanto, $e(x) = \boldsymbol{\alpha^{14} x^{10} + \alpha^7 x^{11}}$
$c(x) = r(x) + e(x) = \alpha^{12} x^3 + \alpha^5 x^4 + \alpha^3 x^5 + \alpha^{13} x^6 + \alpha^{11} x^7 + \alpha x^8 + \alpha^{11} x^9 + \alpha^2 x^{10}$
$+ \alpha^{14} x^{11} + \alpha^8 x^{12} + \alpha^{10} x^{13} + \boldsymbol{\alpha^{14} x^{10} + \alpha^7 x^{11}}$.
$= \alpha^{12} x^3 + \alpha^5 x^4 + \alpha^3 x^5 + \alpha^{13} x^6 + \alpha^{11} x^7 + \alpha x^8 + \alpha^{11} x^9 + \alpha^{13} x^{10}$
$+ \alpha x^{11} + \alpha^8 x^{12} + \alpha^{10} x^{13}$.

Se puede verificar que la palabra código es: $c(x) = (\alpha^{10} x^9 + \alpha^3 x^5 + \alpha^2 x^3)\, g(x)$ y por el teorema 4.2.1.11 el código Reed-Solomon [15, 11, 5] corrige 5 errores en forma de burst o ráfaga.

[21] Como en el caso de los códigos BCH, para construir la tabla del algoritmo de Euclides, se invita al lector a revisar [41, 20] y como un complemento [42, 18].

4.2.3 CÓDIGOS REED-SOLOMON GENERALIZADOS.

Recordamos de la construcción de los códigos Reed-Solomon, que las palabras código son obtenidas por

$$c = (f(1),\ f(\alpha),..., f(\alpha^{n-1})). \tag{4.12}$$

Ahora elegimos un vector v = (v_1, v_2,…, v_n) cuyos elementos son diferentes de cero. Entonces una generalización de (4.12) es

$$c = (v_1\ f(1), v_2\ f(\alpha),..., v_n\ f(\alpha^{n-1})).$$

La definición de lo anterior quedaría de la siguiente manera:

Definición 4.2.3.1 Sea $n \le q$. Con $\boldsymbol{\alpha} = (\alpha_1, \alpha_2,\ldots, \alpha_n)$, donde α_i $(1 \le i \le n)$ son elementos distintos de $\mathbf{F}_q$. Sea $\mathbf{v} = (v_1, v_2,\ldots, v_n)$, donde $v_i \in F_q$ $1 \le i \le n$. Para $k \le n$, el código Reed-Solomon generalizado GRS_k ($\boldsymbol{\alpha}$, $\mathbf{v}$) es definido, de la siguiente manera:

$$\{(v_1 f(\alpha_1), v_2 f(\alpha_2),..., v_n f(\alpha_n)) \ :\ f(x) \in F_q[x] \ \ y \ \ grad(f(x)) < k\}.$$

Los elementos $\alpha_1, \alpha_2,\ldots, \alpha_n$ son llamados localizadores de código de GRS_k ($\boldsymbol{\alpha}$, $\mathbf{v}$).

Teorema 4.2.3.2 El código Reed-Solomon generalizado GRS_k ($\boldsymbol{\alpha}$, $\mathbf{v}$) tiene los siguientes parámetros: *[n, k, n – k + 1],* por lo tanto, es un código MDS. Su demostración es la misma utilizada para un código Reed-Solomon no generalizado, demostrado anteriormente con el teorema del límite de Singleton del Apéndice (A).

Corolario 4.2.3.3 Una matriz parity-check de GRS_{n-K} ($\boldsymbol{\alpha}$, $\mathbf{v}'$) $\subseteq$ GRS_K ($\boldsymbol{\alpha}$, $\mathbf{v}$) $^{\perp}$ es:

$$\begin{pmatrix} v' & v'_2 & \cdots & v'_n \\ v'_1\alpha_1 & v'_2\alpha_2 & \cdots & v'_n\alpha_n \\ v'_1\alpha_1^2 & v'_2\alpha_2^2 & \cdots & v'_n\alpha_n^2 \\ \vdots & & \vdots & \vdots \\ v'_1\alpha_1^{n-k-1} & v'_2\alpha_2^{n-k-1} & \cdots & v'_n\alpha_n^{n-k-1} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha^1 & \alpha^2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & & \vdots & \vdots \\ \alpha_1^{n-k-1} & \alpha_2^{n-k-1} & \cdots & \alpha_n^{n-k-1} \end{pmatrix} \qquad \begin{pmatrix} v'_1 & 0 & \cdots & 0 \\ 0 & v'_2 & \cdots & 0 \\ & & \cdots & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & v'_n \end{pmatrix}.$$

4.3 CÓDIGOS ALTERNANTES (GOPPA).

4.3.1 Códigos alternantes.

V. D. Goppa describió una nueva e interesante clase de códigos lineales de corrección de error, llamados códigos Goppa, en los comienzos de los 70s. Esta clase de códigos incluye los códigos BCH narrow-sense. Los códigos Goppa son considerados la subclase más interesante de los códigos alternantes, introducidos por H. J. Helgert en 1974. La clase de los códigos alternantes surgen de los códigos Reed-Solomon generalizados, vistos en la sección anterior. Los códigos alternantes son una larga familia de códigos que contienen a los bien conocidos códigos Hamming, BCH y Goppa.

En esta sección se usará la misma notación que para los códigos Reed-Solomon generalizados, excepto que ahora son definidos sobre F_{q^m} para cualquier $m \geq 1$.

Definición 4.3.1.1 Un código alternante $A_K(\alpha, v')$ sobre $\mathbf{F}_q$ es el subcampo del subcódigo de $GRS_K(\alpha , \mathbf{v})|\mathbf{F}_q$, donde $GRS_K(\alpha , \mathbf{v})$ es un código Reed-Solomon generalizado sobre F_{q^m}, para algún $m \geq 1$.

Proposición 4.3.1.2 El código alternante $A_K(\boldsymbol{\alpha}, \mathbf{v}')$ tiene parámetros $[n, k', d]$, donde $mk - (m-1)n \leq k' \leq k$ y $d \geq n - k + 1$.

Demostración. Por el teorema 4.2.3.2, $GRS_K(\boldsymbol{\alpha}, \mathbf{v})$ tiene parámetros $[n, k, n - k + 1]$. *Por lo tanto,* $A_K(\boldsymbol{\alpha}, \mathbf{v}')$ tiene longitud n, y su dimensión es k' satisfaciendo trivialmente $k' \leq k$.

Observación 4.3.1.3 De la definición 4.3.1.1 y del corolario 4.2.3.3 tenemos que $A_K(\boldsymbol{\alpha}, \mathbf{v}')$ esta definida para: $\{c \in F_q^n : cH^T = 0\}$,

donde H es la *matriz parity check. Como* H esta determinada por $\boldsymbol{\alpha}$ y $\mathbf{v}'$, es apropiado que la notación de un código alternante sea expresada en términos de $\boldsymbol{\alpha}$ y $\mathbf{v}'$.

Ejemplo 4.3.1.4 Para q *y* m, Recordemos que para un código *BCH* sobre $\mathbf{F}_q$ es un código que consiste de todas las palabras código $\mathbf{c} \in F_q^n$ que satisface $cH^T = \mathbf{0}$, donde

$$H' = \begin{pmatrix} 1 & \alpha^{a} & \alpha^{2a} & \cdots & \alpha^{a(n-1)} \\ 1 & \alpha^{a+1} & \alpha^{2(a+1)} & \cdots & \alpha^{(a+1)(n-1)} \\ 1 & \alpha^{a+2} & \alpha^{2(a+2)} & \cdots & \alpha^{(a+2)(n-1)} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & \alpha^{a+\delta-2} & \alpha^{2(\alpha+\delta-2)} & \cdots & \alpha^{(a+\delta-2)(n-1)} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^{2} & \cdots & \alpha^{n-1} \\ 1 & \alpha^{2} & \alpha^{4} & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & \alpha^{\delta-2} & \alpha^{2(\delta-2)} & \cdots & \alpha^{(\delta-2)(n-1)} \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \alpha^{a} & \cdots & 0 \\ & & \cdots & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha^{a(n-1)} \end{pmatrix}$$

Por lo tanto, un código BCH es también un código alternante.

Ejemplo 4.3.1.5 Sea $q = 2$ y $m = 3$, y sea $n = 6$. Sea α un elemento primitivo de $\mathbf{F}_8$ que satisface $\alpha^3 + \alpha + 1 = 0$. Tomamos $\boldsymbol{v}' = (1,\ldots,1)$ y $\boldsymbol{\alpha} = (\alpha, \alpha^2, \ldots, \alpha^6)$. Entonces

$$A_3(\alpha, v') = \{c \in F_2^6 : cH^T = 0\},$$

donde

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{pmatrix}.$$

Entonces, para $H = (1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2})$ y $\overline{H}$ es una matriz m x $(2^m - 2)$.

$$\overline{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

El cual tiene la siguiente forma reducida:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Por lo tanto, $A_3(\boldsymbol{\alpha}, \mathbf{v}')$ tiene una matriz generadora:

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

Por lo tanto, es un código lineal [6, 2, 4].

4.3.2 Códigos Goppa.

Los códigos Goppa son también usados en la criptografía, el criptosistema McEliece y el criptosistema Niederreiter son ejemplos de criptosistemas de clave pública que usan códigos Goppa.

Definición 4.3.2.1 Sea g(z) un polinomio en F_{q^m} [z] para cualquier *m,* y sea $L = \{\alpha_1,\ldots, \alpha_n\}$ un subconjunto de F_{q^m} tal que $L \cap \{g(z)\} = \emptyset$ Para $\mathbf{c} = (c_1, \ldots, c_n) \in F_q^n$, siendo

$$R_C(z) = \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i}.$$

El código Goppa Γ(L , g) es definido como:

$$\Gamma(L,\ g) = \{c \in F_q^n : R_C(z) \equiv 0 (\text{mod}\ \ g(z))\}.$$

El polinomio g(z) es llamado polinomio Goppa. Cuando g(z) es irreducible, Γ(L , g) es llamado un código Goppa irreducible.

Observación 4.3.2.2 Notar que

$$(z - \alpha_i)\left(-\frac{g(z) - g(\alpha_i)}{z - \alpha_i} g(\alpha_i)^{-1}\right) \equiv 1 \pmod{g(z)},$$

y como $(g(z) - g(\alpha_i))/(z - \alpha_i)\ g(z)$ es un polinomio, entonces $1/(z - \alpha_i)$ puede ser visto como un polinomio; i.e.,

$$\frac{1}{z - \alpha_i} \equiv -\frac{g(z) - g(\alpha_i)}{z - \alpha_i} g(\alpha_i)^{-1} \pmod{g(z)},$$

Por lo tanto, la congruencia $R_C(z) \equiv 0$ (mod g(z)) en la definición de Γ(L , g) significa que g(z) divide al polinomio

$$\sum_{i=1}^{n} c_i \frac{g(z) - g(\alpha_i)}{z - \alpha_i} g(\alpha_i)^{-1}.$$

Sin embargo, notar que $(g(z) - g(\alpha_i))/(z - \alpha_i)$ es un polinomio de grado $< t$ si *g(z)* tiene grado t, se dice que $\mathbf{c} \in \Gamma(L,g)$ si y solo si

$$\sum_{i=1}^{n} c_i \frac{g(z) - g(\alpha_i)}{z - \alpha_i} g(\alpha_i)^{-1} = 0$$

Es un polinomio.

Proposición 4.3.2.3 Para un polinomio Goppa g(z) de grado *t* y $L = \{\alpha_1,\ldots,\alpha_n\}$, tenemos

$$\Gamma(L, g) = \{c \in F_q^n : cH^{\mathrm{T}} = 0\},$$

donde

$$H = \begin{pmatrix} g(\alpha_1)^{-1} & \cdots & g(\alpha_n)^{-1} \\ \alpha_1 g(\alpha_1)^{-1} & \cdots & \alpha_n g(\alpha_n)^{-1} \\ \vdots & \cdots & \vdots \\ \alpha_1^{t-1} g(\alpha_1)^{-1} & \cdots & \alpha_n^{t-1} g(\alpha_n)^{-1} \end{pmatrix}.$$

Podemos ver fácilmente que H´puede ser descompuesta por:

$$\begin{pmatrix} g_t & 0 & \cdots & 0 \\ g_{t-1} & g_t & \cdots & 0 \\ g_{t-2} & g_{t-1} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{pmatrix}\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix}\begin{pmatrix} v_1' & 0 & \cdots & 0 \\ 0 & v_2' & \cdots & 0 \\ & & \cdots & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & v_n' \end{pmatrix},$$

donde $\boldsymbol{v}'_i = g(\alpha_i)^{-1}$ para $1 \le i \le n$. Decimos que $\mathbf{c} \in \Gamma(L, g)$ si y solo si $\mathbf{c}H^{\mathrm{T}} = 0$, donde

$$H = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix}\begin{pmatrix} g(\alpha_1)^{-1} & 0 & \cdots & 0 \\ 0 & g(\alpha_2)^{-1} & \cdots & 0 \\ & & \cdots & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & g(\alpha_n)^{-1} \end{pmatrix}.$$

Ejemplo 4.3.2.4 Para cualquier q, tenemos $g(z) = z^t$ y sea $L = \{1, \alpha^{-1}, \alpha^{-2}, \ldots, \alpha^{-(q^m-2)}\}$, donde α es un elemento primitivo de F_{q^m}. Por lo tanto $n = q^m - 1$. Entonces $\Gamma(L, g) = \{\mathrm{c} \in F_q^n : \mathbf{c}H^T = \mathbf{0}\}$, donde

$$H = \begin{pmatrix} 1 & \alpha^t & \alpha^{2t} & \cdots & \alpha^{(n-1)t} \\ 1 & \alpha^{t-1} & \alpha^{2(t-1)} & \cdots & \alpha^{(n-1)(t-1)} \\ \vdots & & \vdots & \ddots & \vdots \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \end{pmatrix}.$$

Podemos observar que $\Gamma(L, g)$ es precisamente un código BCH narrow sense.

Ejemplo 4.3.2.5 Sea $q = 2$ y tenemos $g(z) = \alpha^3 + z + z^2$, donde α es un elemento primitivo de $\mathbf{F}_8$ que satisface $\alpha^3 + \alpha + 1 = 0$. Sea $L = \mathbf{F}_8$, Por lo tanto, $n = 8$ y $m = 3$. Entonces

$$\Gamma(L, g) = \{c \in F_2^8 : cH^{\mathrm{T}} = 0\},$$

donde

$$H = \begin{pmatrix} \alpha^4 & \alpha^4 & \alpha & 1 & \alpha & \alpha^2 & \alpha^2 & 1 \\ 0 & \alpha^4 & \alpha^2 & \alpha^2 & \alpha^4 & \alpha^6 & 1 & \alpha^6 \end{pmatrix}.$$

Reemplazando cada entrada en H por un vector columna en F_2^3, obtenemos una matriz $\overline{H}$ la cual tiene la siguiente forma escalonada reducida por renglón:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix},$$

Asi, $\Gamma(L, g)$ tiene la siguiente matriz generadora: $\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$.

Por lo tanto $\Gamma(L, g)$ es un código lineal y tiene parámetros [8, 2, 5].

CAPÍTULO 5

PRESENTACIÓN DEL SIMULADOR

5. PRESENTACIÓN DEL SIMULADOR.

El propósito de este capítulo, es simular una secuencia de información de los códigos BCH y analizar sus estrategias de codificación y decodificación, comparándolos con otros códigos correctores de errores, en un canal con ruido y/o distorsión.

5.1 Simulación de errores.

Se realizó la simulación de un sistema de comunicación digital, simulando errores con los diferentes métodos de simulación de ruido y/o distorsión y de producción de errores que se trataron en el capítulo 1. Los cuales son:

Método 1.- Errores independientes (BSC).
Método 2.- Simulación Quasianalítica (QA).
Método 3.- Dos estados HMM.
Método 4.- Estimación de parámetros HMM.
Método 5.- Tres estados HMM.

De acuerdo a nuestro modelo de estados finitos, el canal discreto es simétrico (un cero y un uno que representan los bits, son afectados de igual manera). Por lo tanto, pueden ser descritos por una secuencia de errores de bit. La secuencia simulada de error de bit para errores en forma de burst puede ser planteada con un modelo de Markov escondido (HMM).

Para el canal binario un HMM es definido por (S, Π, A, B), donde S = {s_1, s_2,..., s_n} es el conjunto de estados, Π es el vector de probabilidad inicial del estado, $A = \lfloor a_{ij} \rfloor_{n,n}$ es la matriz de probabilidad de transición de los estados ($a_{ij} = \Pr\{s_j | s_i\}$) y $B = \lfloor b_{ik} \rfloor_{n,m}$ es una matriz de probabilidad de transición de cambio de símbolo (bit de error). Los diagramas de transición para dos y tres estados son presentados a continuacion:

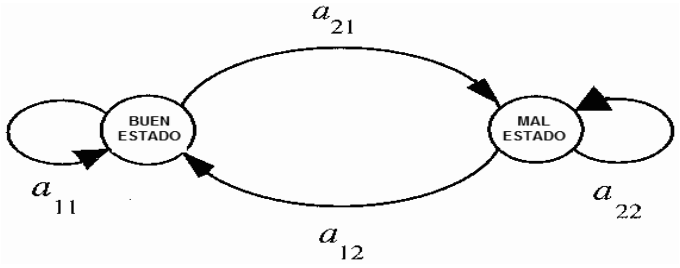


Figura 5.1 Diagrama de transición para HMM de dos estados (Modelo de Gilbert).

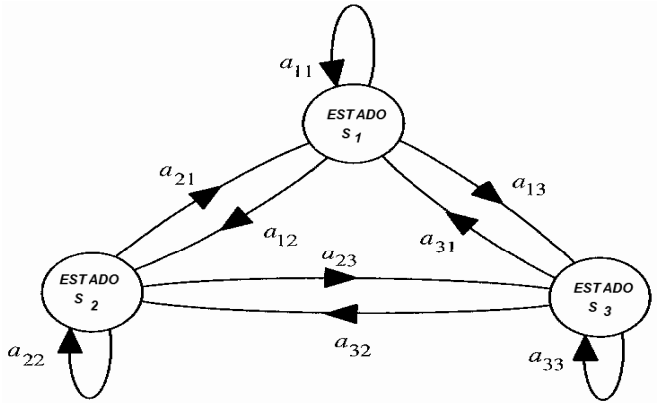


Figura 5.2 Modelo HMM con tres estados.

A continuación analizaremos los siguientes parámetros para un canal discreto.

Para un modelo de dos estados:

$$\Pi = \begin{bmatrix} 0.99 \\ 0.01 \end{bmatrix}; \qquad A = \begin{bmatrix} 0.99 & 0.13 \\ 0.01 & 0.87 \end{bmatrix}; \qquad B = \begin{bmatrix} 1.00 & 0.00 \\ 0.57 & 0.43 \end{bmatrix}.$$

Para un modelo con tres estados:

$$\Pi = \begin{bmatrix} 0.97 \\ 0.01 \\ 0.02 \end{bmatrix}; \qquad A = \begin{bmatrix} 0.98 & 0.00 & 0.19 \\ 0.01 & 0.65 & 0.00 \\ 0.01 & 0.35 & 0.81 \end{bmatrix}; \qquad B = \begin{bmatrix} 1.00 & 0.00 \\ 0.45 & 0.55 \\ 0.61 & 0.39 \end{bmatrix}$$

5.1.1 Distribución de probabilidad de intervalos libres de error.

Para un modelo de Markov escondido el intervalo libre de error *l* tiene una matriz geométrica de distribución de probabilidad acumulativa[22] dado por

$$\frac{\Pi P(1)[1 - P^{l}(0)]}{\Pi P(1)} \tag{5.1}$$

donde definimos

$$P(e) = AC(e), \qquad e = 0, 1$$

y C(e) es una matriz diagonal n x n, la cual sus elementos diferentes de cero son las columnas de la matriz B. Por ejemplo, si e = 0, las entradas de la diagonal en C(0) son (b_{11}, b_{21},…, b_{n1}) y para e = 1, las entradas de la diagonal en c(1) son (b_{12}, b_{22},…, b_{n2}).

Los resultados de calcular esta probabilidad para errores independientes (sin memoria) y para los modelos de dos y tres estados con los parámetros antes analizados, son mostrados a continuación (figura 5.3):

---

[22] [34; 842] Jeruchim Michel C. Philip Balaban and K. Sam Shanmugan. Simulation of Communication Systems. New York: Klumer Academic/Plenum Publishers, Second Edition, 2000.

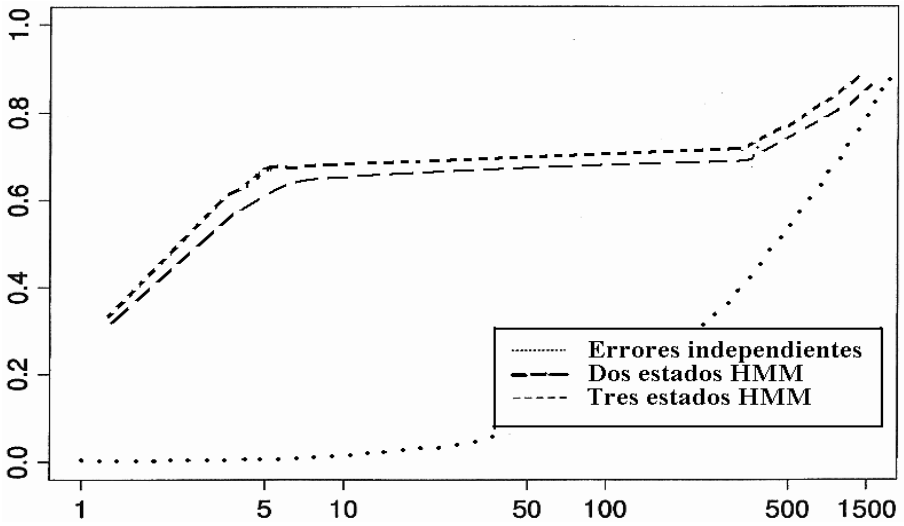
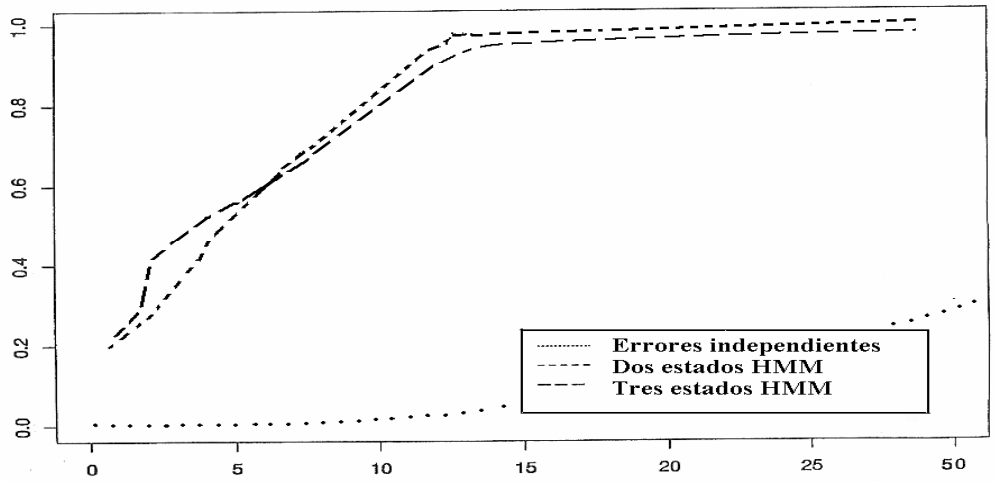


Figura 5.3 Distribución de probabilidad acumulativa de intervalos libres de error.

5.1.2 Modelación del canal de error discreto.

Ahora describiremos los resultados encontrados para un HMM para producir una secuencia de error que simula ruido para los parámetros antes analizados y comparándolo, con la simulación de errores independientes. Obteniendo la siguiente gráfica de probabilidades acumulativas:



Gráfica 5.4 Número de errores en un block de secuencia de 1000 bits.

Para dos y tres estados la figura mostrada anteriormente es muy similar a la obtenida por los autores Jeruchim Michel C., Philip Balaban y K. Sam Shanmugan en [34; 842][23] la cuales son presentadas en las figuras 5.5 y 5.6, para un canal de comunicación inalámbrico llamado "reverse radio link" para simular una transmisión telefónica de usuarios móviles a su estación base, y llamado "fordward radio link" para una transmisión telefónica de la estación base al usuario, respectivamente.

[23] [34; 842] Jeruchim Michel C. Philip Balaban and K. Sam Shanmugan. Simulation of Communication Systems. New York: Klumer Academic/Plenum Publishers, Second Edition, 2000.

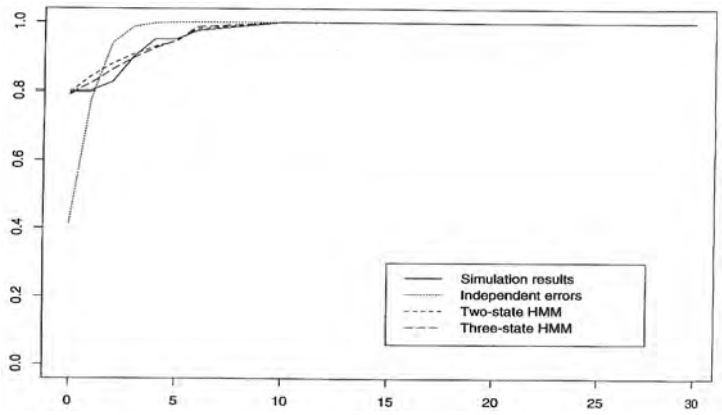


Figura 5.5 Probabilidades acumulativas de un número de errores en un block de 576 bits "reverse radio link". Fuente: "Jeruchim Michel C. Philip Balaban and K. Sam Shanmugan, 2000".

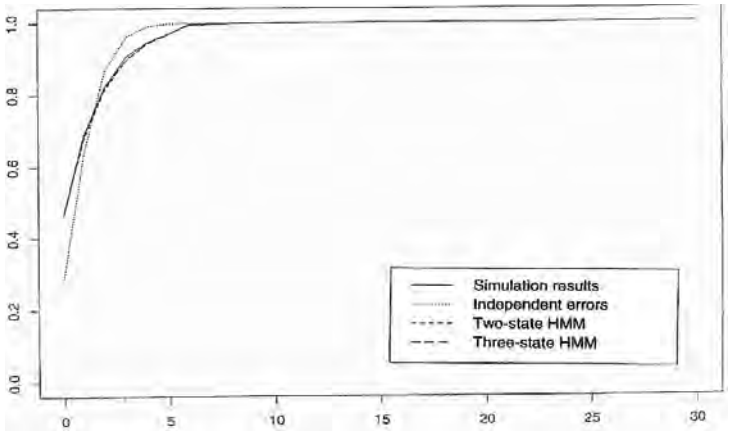


Figura 5.6 Probabilidades acumulativas de un número de errores en un block de 576 bits "fordward radio link". Fuente: "Jeruchim Michel C. Philip Balaban and K. Sam Shanmugan, 2000".

Aunque los autores presentan la simulación con 576 bits, que implican dispositivos digitales (moduladores, esparcidores de datos, filtros, etc.) y un AWGN con la aplicación de una SNR = 6 dB que hacen simular desvanecimiento de la señal e interferencia, los valores obtenidos para dos y tres estados con nuestro simulador, proporcionan los mismos resultados que en el caso de estudio IV de los autores antes mencionados. Por lo tanto, podemos concluir que la simulación es correcta.

Un importante parámetro del modelo de Markov es el número de estados. El problema de elegir el número de estados se soluciona usando límites confidenciales. De otra manera, el número de estados puede ser seleccionado dependiendo de la aplicación del modelo. Con los datos obtenidos del simulador se concluye que, este es un canal discreto, el cual es computacionalmente costoso y es impráctico usarlo para evaluar el funcionamiento de dispositivos digitales (tales como codificadores correctores/detectores de error, decodificadores, entrelazadores, etc.).

Sin embargo, nuestro simulador puede ser usado para generar suficientes errores en forma de bit con un modelo de estados finitos. A lo largo de esta investigación se ha ido demostrando que para un sistema de comunicación, el modelo de estados finitos proporciona una amplia variedad de procesos aleatorios y son muy buenos candidatos para modelar el ruido y distorsión.

5.2 Ténicas de Entrelazado (Entrelazado y desentrelazador de Bloques).

El entrelazado es considerado una técnica sencilla y eficiente para superar burst de error, este toma una secuencia de símbolos y los combina. En el receptor, la secuencia es combinada nuevamente obteniendo el orden original utilizando un desentrelazador. Los entrelazadores son eficaces para contender errores en forma de burst, los errores en forma de burst tienen la característica de ser varios errores aleatorios juntos o próximos uno de otro, dicha proximidad puede ser esparcida si utilizamos un entrelazador que convierte los errores en forma de burst en errores aleatorios.

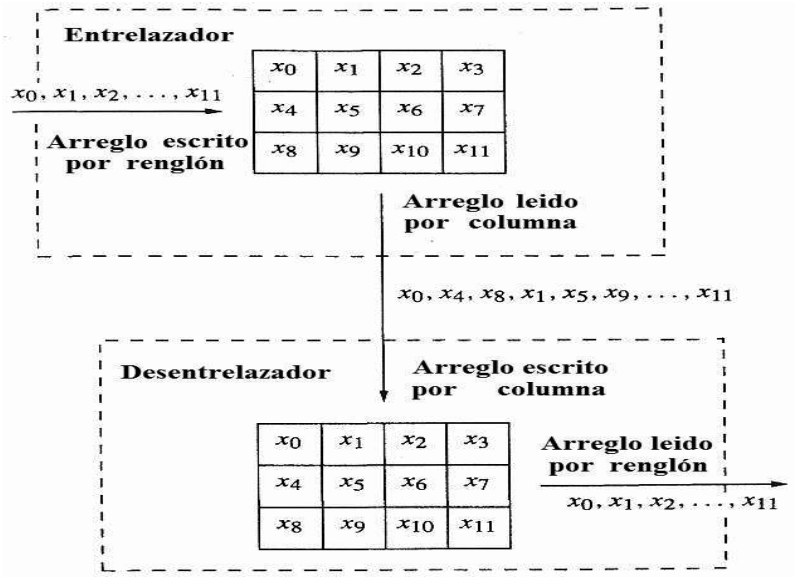


Figura 5.7 Entrelazador y Desentrelazador.

La forma más común de entrelazar, es utilizando un entrelazador de bloque. Esto es simplemente un arreglo de N x M el cual puede ser leído y escrito de forma diferente. Típicamente la secuencia de entrada de símbolos es escrita en el entrelazador en forma de renglón y leida en forma de columna. La figura 5.7 muestra un entrelazador 3 x 4. La secuencia de entrada $x_{0,}\ x_{1,\ldots,}\ x_{11}$ es leída en forma de renglón en el arreglo como se muestra en la figura 5.7, y a la salida del entrelazador la secuencia es leída como:

$$x_0,\ x_4,\ x_8, x_1, x_5, x_9, x_2, x_6, x_{10}, x_3, x_7, x_{11}.$$

Frecuentemente la amplitud M es elegida como la longitud de una palabra código.

En general cuando un burst de longitud ***l*** es desentrelazado, causa un máximo de $\lceil l/N \rceil$ errores entre las palabras código recibidas. Si el código usado puede corregir hasta **t** errores, la decodificación fracasará si la longitud del burst es mayor que N**t** + 1.

La eficiencia de un entrelazador es definida como la relación de la longitud del más pequeño burst de error que excede la capacidad de corrección a la cantidad de memoria en el entrelazador. Para un entrelazador de bloque la eficiencia es:

$$\mathrm{E} = \frac{Nt+1}{NM} \approx \frac{t}{M}. \qquad (5.2)$$

5.2.1 Entrelazador y desentrelazador convolucional.

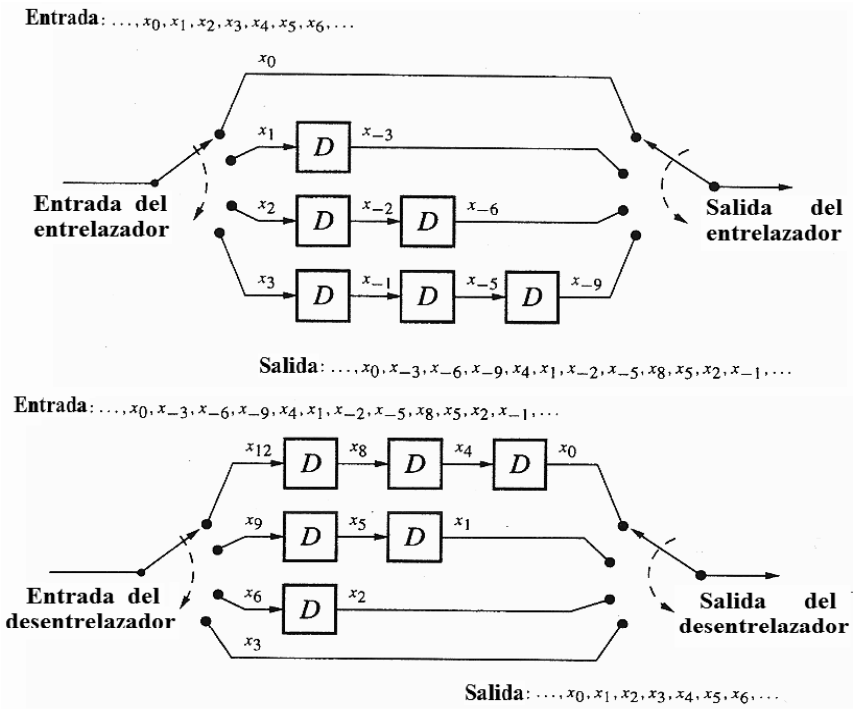


Figura 5.8 Entrelazador y desentrelazador convolucional.

El entrelazador y desentrelazador convolucional es el mostrado en la figura 5.8. Esta figura muestra la cadena de entrada y el estado de entrelazado en un particular tiempo, dichas líneas de retraso incrementan sucesivamente la longitud. Los parámetros del entrelazador convolucional son (M, D), donde M es el número de líneas de retraso y D es el número de muestras que cada elemento de retraso introduce. Quedando claro que los símbolos de entrada del entrelazador son separados por MD símbolos. Si M es más grande que o igual a la longitud del código, entonces cada símbolo en la palabra código es colocada en una diferente línea de retraso. Si un burst de longitud ***l*** ocurre, entonces

$$\lceil l/(MD+1) \rceil$$

errores pueden ser introducidos dentro de las palabras código del desentrelazador. Para un código corrector de **t** errores, la decodificación fracasará si la longitud del burst es mayor que (MD + 1)(**t** - 1) + 1. La eficiencia esta dada por:

$$\mathrm{E} = \frac{(MD+1)(t-1)+1}{DM(M-1)/2} \approx \frac{2t}{M-1}. \tag{5.3}$$

Comparando con el entrelazador y desentrelazador de bloque, los convolucionales son aproximadamente el doble de eficientes. Sin embargo, el tiempo es un factor que debemos tomar en cuenta en su diseño. De otra manera, podemos elegir un código que corrija más errores y de dicha forma evitar el tiempo de retraso de un entrelazador y desentrelazador convolucional.

5.3 Validación de los códigos correctores de errores.

En este apartado, analizaremos diferentes códigos correctores de errores con nuestro simulador de canal con ruido y/o distorsión.

Ejemplo 5.1 Los códigos cíclicos BCH [15, 5, 7] y Golay [23, 12, 7] son transmitidos a través de un canal con ruido, donde los errores son generados por un modelo de Markov escondido de dos estados. Se utilizan los algoritmos de decodificación para el método 1 error trapping y para el método 2 la decodificación para códigos cíclicos de corrección de error burst. Corremos el programa y obtenemos lo siguiente:

```
Escribe tu texto: PUMA
------------------------------------------------------------------------
Tu texto es: PUMA
------------------------------------------------------------------------
Tu texto en codigo BCH [15, 5, 7] con 60 digitos es:
100110101111000110001001101011011001010000111111011001010000

Tu texto en codigo Golay [23, 12, 7] con 92 digitos es:
110000100001001110000001100011101001001100011011010010111100110001100
111101110010000110000
------------------------------------------------------------------------
Escribe los porcentajes de estados:
Porcentaje de transicion Bueno a Malo:
95
Porcentaje de quedarse en el estado Bueno:
5
------------------------------------------------------------------------
Porcentaje de transicion Malo a Bueno:
20
Porcentaje de quedarse en el estado Malo:
80
------------------------------------------------------------------------
Porcentaje de no cambio de simbolo del estado Bueno:
100
Porcentaje de cambio de simbolo del estado Bueno:
0
------------------------------------------------------------------------
Porcentaje de no cambio de simbolo del estado Malo:
70
Porcentaje de cambio de simbolo del estado Malo:
30
------------------------------------------------------------------------
Tu texto modificado para el codigo BCH [15, 5, 7] es:
100XXXXX1111000110001001101011011001010000111111011001010XXX0
Tu texto modificado para el codigo Golay [23, 12, 7] es:
110XXXXX000100111000000110001110100100110001101101001011XXX01100011001
111101110010000XX0X000
------------------------------------------------------------------------
Total de errores para la cadena binaria de 15 es:  8
Total de errores para la cadena binaria de 23 es: 11
```

```
Decodificacion  1 para el codigo BCH [15, 5, 7]: ?uma
Decodificacion  2 para el codigo BCH [15, 5, 7]: puma
Decodificacion  1 para el codigo Golay [23, 12, 7] es: ?UMA
Decodificacion  2 para el codigo Golay [23, 12, 7] es: PUMA
```

En el ejemplo 5.1 se simula la transmisión del texto "PUMA" a través de un canal con ruido y/o distorsión, para el cual genera 8 y 11 errores. El texto es codificado y decodificado para los códigos cíclicos BCH [15, 5, 7][24] y Golay [23, 12, 7]. En la decodificación 1 se utiliza el método de error trapping (visto en el apartado 3.3 decodificación de los códigos cíclicos) el cual corrige 3 errores aleatorios y en la decodificación 2 se utiliza el método de corrección de error burst (visto en el apartado 3.4, códigos de corrección de error en burst (o ráfagas)). La siguiente ecuación dada en el capítulo 3, nos permite demostrar el número de errores que corrige la corrección de error burst

$$l \leq \left\lfloor \frac{n-k}{2} \right\rfloor$$

Para el caso de los códigos cíclicos BCH [15, 5, 7]. Tenemos,

$$l = \left\lfloor \frac{15-5}{2} \right\rfloor = 5.$$

Para el código cíclico Golay [23, 12, 7] procedemos de igual manera y tenemos $l = 5$. Es decir, puede corregir hasta 5 errores en forma de burst.

Con lo cual queda demostrado que con un canal de dos estados que utiliza modelos de Markov escondido, se pueden generar errores aleatorios que simulan ruido y/o distorsión. Además, existen diferentes tipos de decodificación para errores aleatorios y para errores en forma de burst "en el mismo código cíclico".

En está investigación, se mencionaron los factores que han ayudado a un considerable desarrollo en la teoría de códigos. La información que se origina en la fuente de datos es codificada sumandosele dígitos redundantes, transmitida sobre el canal y decodificada para aparecer en su forma original en el receptor de datos. Los dígitos afectados por el simulador de canal con ruido y/o distorsión son corregidos con la ayuda de los códigos correctores de errores, esto tiene como consecuencia la creación de una gran cantidad de los mismos. Una vez establecido esto, surge el problema de que código utilizar para un canal con ruido y/o distorsión determinado.

Aunque nuestro trabajo de investigación se orienta específicamente a la clase de códigos BCH, se desea señalar que el diseño de un código se realiza tomando en cuenta las restricciones que nos imponga el caso. Para esto definiremos algunos conceptos que nos ayudarán a comprender dichas restricciones y evaluar las propiedades de un código. La rentabilidad esta definida como la razón del número de dígitos que fueron recibidos correctamente al total de dígitos enviados; esto también puede ser expresado como 1 – P (e), donde P (e) es la probabilidad de que un dígito enviado sea incorrecto. La eficiencia del código esta definida como la razón del número de dígitos de información al número de dígitos enviados. Por lo tanto, la redundancia del código esta definida como (1 – eficiencia del código).

Desafortunadamente no existe algún procedimiento sistemático para sintetizar que un código sea óptimo en todos los sentidos. Pero podemos evaluarlos como lo muestra la figura 5.9. Donde la probabilidad de error podría ser igual a la probabilidad de que nuestro mensaje sea o no retransmitido.

[24] El código BCH [15, 5, 7] solo maneja letras minúsculas para su codificación y decodificación, debido a su tamaño.

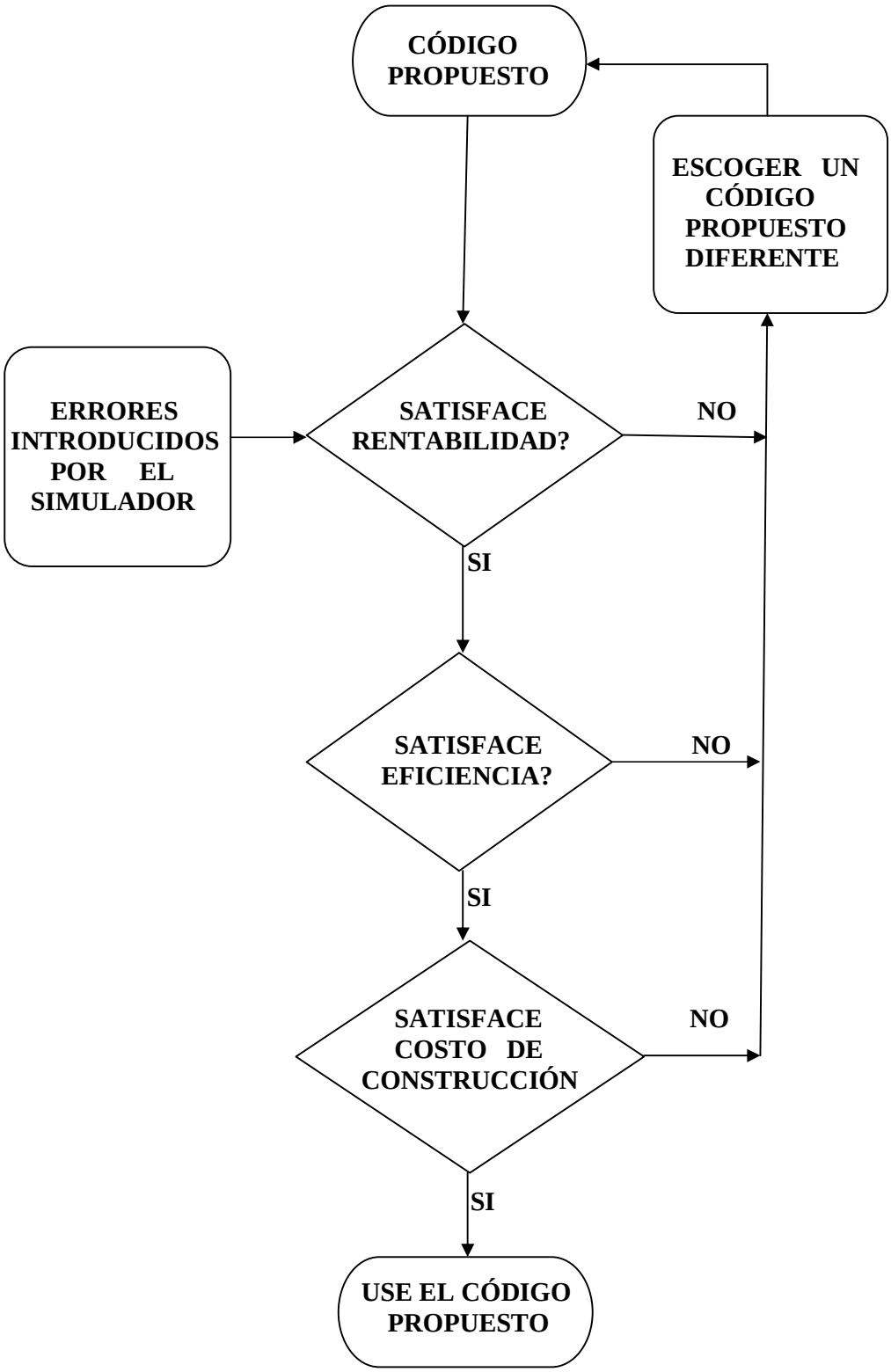


Figura 5.9 Criterio de selección de un código corrector de errores.

La tabla siguiente proporciona los métodos de decodificación utilizados para la simulación de una transmisión.

CÓDIGO	MÉTODO DE DECODIFICACIÓN	PALABRAS CÓDIGO
BCH [15, 5, 7]	**DECODIFICACIÓN ALGEBRAICA (SOLUCIÓN DE ECUACIONES)**	**32**
BCH [31, 6, 15]	**DECODIFICACIÓN ALGEBRAICA (SOLUCIÓN DE ECUACIONES)**	**64**
REED MULLER R[1, 5]	**DECODIFICACIÓN EXHAUSTIVA**	**64**
REED SOLOMON [15, 11, 5] CONCATENADO ES [60, 44, 5] ERRORES EN FORMA DE BURST 5	**DECODIFICACIÓN ALGEBRAICA (SOLUCIÓN DE ECUACIONES)**	**150**
REED SOLOMON [15, 9, 7] CONCATENADO ES [60, 36, 7] ERRORES EN FORMA DE BURST 9	**DECODIFICACIÓN ALGEBRAICA (SOLUCIÓN DE ECUACIONES)**	**150**
BCH [63, 7, 31]	**DECODIFICACIÓN EXHAUSTIVA**	**150**

Tabla 5.1 Métodos de decodificación para la simulación.

A continuación se presentan los resultados obtenidos por el simulador con diferentes probabilidades de error y diferentes códigos correctores de errores, entre ellos los códigos BCH (valores de criterio de selección **E**=Excelente, **B**=Bueno, **R**=Regular y **D**=Deficiente).

PROBABILIDAD	0.05			0.1			0.15			0.2			0.25			0.3		
	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO
BCH [15, 5, 3]	2	7	E	8	7	E	4	5	B	15	3	R	14	6	D	17	9	D
BCH [31, 6, 7]	5	45	E	16	308	E	16	311	B	28	1893	R	37	3731	R	45	5444	D
REED MULLER R[1, 5]	5	3	E	16	2	E	18	2	B	28	2	R	39	2	R	46	2	D
REED SOLOMON [15, 11, 5] CONCATENADO ES [60, 44, 5] ERRORES EN FORMA DE BURST 5	9	10	E	28	10	B	36	26	R	43	20	D	74	23	D	80	43	D
REED SOLOMON [15, 9, 7] CONCATENADO ES [60, 36, 7] ERRORES EN FORMA DE BURST 9	9	10	E	28	11	E	36	28	E	43	35	B	74	37	R	80	44	D
BCH [63, 7, 15]	9	3	E	29	2	E	38	2	B	46	3	R	78	2	R	84	3	D

Tabla 5.2 Valores obtenidos para una simulación de transmisión del texto "UNAM" con errores independientes.

PROBABILIDAD	0.05			0.1			0.2			0.3			0.4			0.5		
	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO
BCH [15, 5, 3]	1	3	E	0	9	E	6	5	B	4	3	R	10	7	D	10	5	D
BCH [31, 6, 7]	1	8	E	2	20	E	8	28	B	11	363	R	10	274	D	12	153	D
REED MULLER R[1, 5]	1	2	E	2	13	E	8	2	B	11	3	R	10	1	D	12	1	D
REED SOLOMON [15, 9, 7] CONCATENADO ES [60, 36, 7] ERRORES EN FORMA DE BURST 9	2	25	E	2	30	E	10	12	E	14	5	B	12	7	R	19	29	D
BCH [63, 7, 15]	2	3	E	2	9	E	11	3	E	16	1	B	12	1	R	19	2	D

Tabla 5.3 Valores obtenidos para una simulación de transmisión del texto "UNAM" para dos estados.

PROBABILIDAD	0.05			0.1			0.2			0.3			0.4			0.5		
	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO	ERRORES	TIEMPO ms	CRITERIO
BCH [15, 5, 3]	1	3	E	0	1	E	6	5	E	4	2	B	10	2	R	10	3	D
BCH [31, 6, 7]	1	8	E	2	5	E	8	21	E	11	19	B	10	14	R	12	10	D
REED MULLER R[1, 5]	1	2	E	2	1	E	8	3	E	11	1	B	10	2	R	12	1	D
REED SOLOMON [15, 9, 7] CONCATENADO ES [60, 36, 7] ERRORES EN FORMA DE BURST 9	2	19	E	2	9	E	10	12	E	14	3	E	12	3	B	19	5	R
BCH [63, 7, 15]	2	3	E	2	1	E	11	5	E	16	1	E	12	3	B	19	3	R

Tabla 5.4 Valores obtenidos para una simulación de transmisión del texto "UNAM" para dos estados utilizando la técnica del entrelazado.

Los resultados obtenidos por el simulador nos proporcionan un cuadro comparativo de los diferentes códigos correctores de errores con diferentes métodos de decodificación. Es evidente, que el código Reed Solomon concatenado y utilizando el entrelazado de bloque, es el mejor para corregir errores en forma de burst y proporciona una gran cantidad de palabras código.

Mientras, que trabajar con los códigos BCH utilizando más palabras código y una mayor distancia, genera mayor complejidad y tiempo computacional, con los códigos Reed Solomon es justificada la complejidad por tener una mayor distancia para corregir errores en forma de burst que con los BCH.

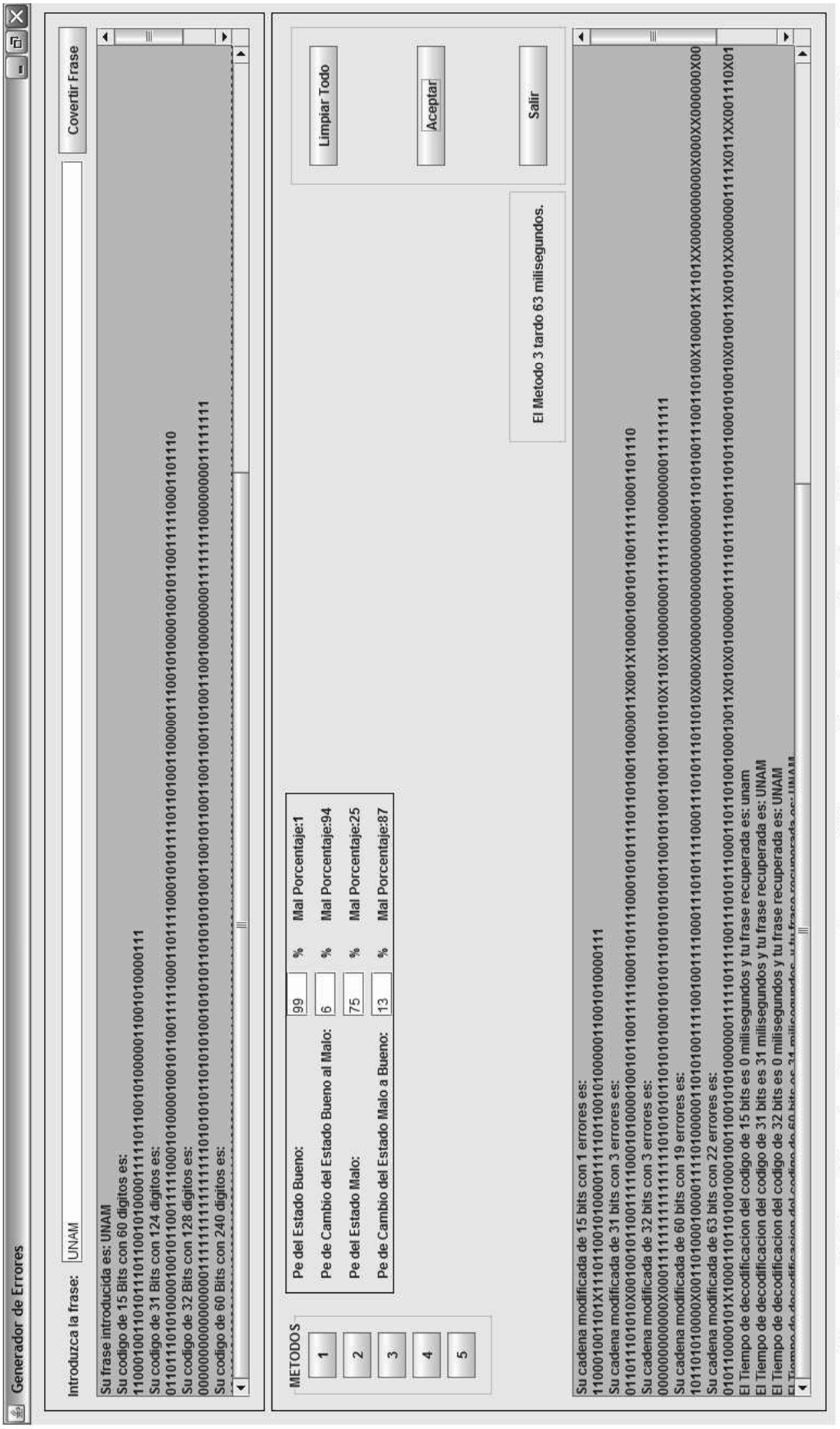


Figura 5.10 Simulación de canal con ruido y su implementación utilizando diferentes códigos BCH.

C O N C L U S I O N E S

En este trabajo de investigación se estableció la importancia de los códigos correctores de errores para aumentar la eficiencia y seguridad de la transferencia de información de un lugar a otro o si deseamos almacenar información para ser usada posteriormente. Estos códigos pueden auxiliar en las tareas de diseño de sistemas de comunicaciones, cuyo objetivo es el de proporcionar un sistema costo-eficiencia para realizar transmisiones de información de acuerdo a las necesidades del usuario. Los otros parámetros clave de un diseño de sistemas de comunicación, son el ancho de banda de transmisión, la potencia de la señal y la complejidad de la implementación elegida. El índice de transmisión de información y la seguridad de la información enviada están típicamente determinados por los requerimientos del usuario.

El ancho de banda de transmisión esta generalmente restringido por factores específicos del medio de transmisión usado, por ejemplo 3.1 kHz para telefonía. Similarmente, existen anchos de banda estándares para canales individuales o circuitos de radio terrestres y enlaces satélitales, debidos al establecimiento de reglas gubernamentales en la utilización del espectro radio eléctrico. En otros casos, el ancho de banda no está restringido tan rigurosamente, ejemplos son los enlaces hacia y desde vehículos en el espacio, donde los anchos de banda son grandes para los pocos enlaces individuales, estos pueden ser elegidos libremente sin preocuparse de posibles interferencias con otros usuarios.

Por lo tanto, la potencia de la señal y la complejidad de la implementación, son características del sistema a discutir en la tarea del diseño. Estas características representan factores de costo para el diseño a considerar. Por ejemplo, en la mayoría de los sistemas, un nivel deseable de seguridad de la información enviada puede ser alcanzado proporcionando suficiente potencia a la señal transmitida para superar las perturbaciones que producen el ruido y/o distorsión, los cuales causan errores.

Una alternativa para alcanzar el mismo objetivo es agregar redundancia a la información transmitida en forma de códigos correctores de errores. Sin embargo, el uso de codificación aumenta la complejidad del sistema, particularmente para la implementación de las operaciones de decodificación.

Como fue mencionado anteriormente, el trabajo de Shannon mostró que en un canal de comunicación hay potencia de la señal, nivel de ruido y ancho de banda, las cuales son usadas para derivar el canal de capacidad C, mientras la velocidad de transmisión se encuentre por debajo de C, la probabilidad de error en la información enviada puede ser baja o nula, usando la codificación para disminuir el costo de la potencia mientras se consume un mayor ancho de banda.

La aplicación de los resultados de la teoría de la codificación en las situaciones prácticas requiere la consideración de dos factores. Primero, la necesidad de aumentar la velocidad a la cual se transmiten los datos puede producir un aumento en la probabilidad de error de cada dato y se debe hacer una evaluación de la efectividad del código expandido con este aumento en el índice de error. Segundo, la suposición de que las probabilidades de error en los dígitos son independientes.

Por ejemplo, un canal sujeto al desvanecimiento producirá índices de error significativamente mayores durante los instantes en que el desvanecimiento es más severo y bloques enteros de dígitos pueden ser erróneos (la cual es una característica de los errores en forma de burst o ráfaga), provocando la posible retransmisión del mensaje.

En esta investigación, se explicó que el problema principal en la transmisión de información es el ruido y/o distorsión el cual es un factor de imprecisión o incertidumbre en la recepción de mensajes o lectura de información y puede ser simulado con los modelos de estados finitos [Canal Simétrico Binario (Binary Simetric Channel o BSC) y Modelos De Markov Escondidos (hidden Markov model o HMM)], método Quasianalítico (QA), relación señal a ruido (SNR) y varios aspectos de producción de error. Siendo el HMM el ideal para simular errores en forma de burst.

Se establecieron los fundamentos de la teoría de códigos lineales sobre campos finitos, así como las definiciones de los conceptos de espacio vectorial, campo finito, distancia y peso de Hamming. También, se definen sus tres principales parámetros (longitud, dimensión y distancia mínima) donde se explicó que el problema principal en la teoría de códigos consiste en encontrar códigos óptimos para parámetros dados. Es decir, determinar el valor máximo de M para el que existe un código de longitud n, tamaño M y distancia mínima d sobre un alfabeto A de tamaño $q > 1$.

Además, se analizaron las características de implementar el método de codificación para códigos lineales. Asimismo, los diferentes métodos de decodificación (por máxima similitud, por distancias mínimas o vecino más cercano y por síndromes) y los posibles problemas a enfrentar cuando se tiene una decodificación incorrecta. Se definen también las matrices generadoras y de chequeo de paridad (parity check) de un código lineal, las cuales determinan completamente al mismo. Se analizaron las reglas de propagación que describen la construcción de nuevos códigos basados en viejos códigos.

Se presentó una breve introducción a los anillos polinomiales y la importancia de los ideales en los códigos cíclicos. Se demostró que un código cíclico es totalmente determinado por su polinomio generador y que la estructura algebraica para las operaciones en polinomios es la misma estructura de un anillo. Se analizó la manera de decodificar algunos códigos cíclicos conocidos como los famosos métodos de error trapping o captura del error y el de corrección de error burst o ráfaga.

Por otra parte, en el marco de la teoría clásica de códigos se presenta el Apéndice (A) donde se explica con ejemplos y gráficamente algunas de las cotas más sobresalientes para posibles valores de M, las cuales hacemos mención durante toda la tesis: la cota de la esfera, la cota de Gilbert – Varshamov, la cota de Hamming (para códigos perfectos), la cota de Singleton (el cual es conocido como máxima distancia separable o MDS), la cota de Plotkin, cota de de Griesmer (la cual aplica específicamente para códigos lineales) y la cota de Elias.

Se demostró que los códigos BCH, capaces de corregir una cantidad prefijada de errores, pueden ser determinados por elegir algunas propiedades de los polinomios generadores (haciendo uso de la factorización de polinomios $x^n - 1$ que se estudia en el Apéndice (B)) y con la cual la información de la distancia mínima puede ser obtenida. Por lo tanto, un algoritmo de decodificación puede ser aplicado.

Una decodificación algebraica para los códigos BCH tiene los siguientes pasos: 1.-Cálculo del síndrome. 2.- Determinar el polinomio localizador de error. (Se mencionaron muchos métodos para determinar el polinomio localizador de error de los cuales se concluye que el mejor método en términos de rápidez es el algoritmo de Berlekamp – Massey mencionados como los algoritmos de decodificación 4.1 y 4.2) y 3.-Encontrar las raíces del polinomio localizador de error. También, se hace referencia al Apéndice (C) donde se da una tabla para que el lector pueda hacer sus propios diseños y comparativos de los códigos BCH.

Se presentaron y analizaron los códigos Reed-Solomon, un tipo de códigos BCH los cuales tienen los parámetros ideales de un código y muy buenos para corregir errores en forma de burst o ráfaga. Por dicha razón, los códigos Reed Solomon son MDS y se consideran BCH por que trabajan en las extenciones de su campo. Es decir, las palabras código Reed Solomon están en $F[x]$ en lugar de $Z_2[x]$. Se demostró que estos códigos al utilizar la regla de concatenación también incrementan su capacidad para corregir errores y su tamaño excelentemente, siendo el mejor de los códigos correctores de errores. Aunque, su decodificación es más compleja. Se mencionan también, los códigos alternantes que son una larga e interesante familia que contiene los bien conocidos códigos Goppa y tienen excelentes propiedades criptográficas.

En el capítulo 5, se demostró prácticamente una simulación de un sistema de comunicaciones, en este caso, la transmisión de un texto, en una computadora Vaio con procesador Core 2 Duo a 2.27 Ghz y memoria RAM 4GB y los programas fueron desarrollados en java en una plataforma NetBeans IDE 6.7.1 y plataforma eclipse 4.1. Los resultados mostados en las tablas 5.1, 5.2, 5.3 y 5.4 nos muestran que el mejor código corrector de error es el Reed Solomon, el cual corrige una gran cantidad de errores para diferentes probabilidades, su decodificación es más compleja y requiere mayor tiempo computacional. Finalmente, se expuso la técnica de entrelazado, la cual ayuda a cualquier código corrector de error a mejorar la capacidad de corregir errores, tan solo por separar los errores en forma de burst.

Para finalizar, cabe mencionar que, en relación a los algoritmos de decodificación, la computación en paralelo, abre la posibilidad de poder procesar, ordenar y analizar códigos correctores de errores de gran tamaño, lo cual despierta un interés para futuros trabajos previos. Se concluye, invitando al lector a utilizar el simulador propuesto para comprobar sus códigos correctores de errores ya sean propios o diseñados de los ya existentes, así como sus propios algoritmos de decodificación.

"POR MI RAZA HABLARÁ EL ESPÍRITU"

A P É N D I C E (A)

En este apartado, estudiaremos los límites superiores y los límites inferiores, los cuales nos dan un mejor entendimiento de la función $A_q(n, d)$ con la cual denotamos a un código lineal, el estudio de $A_q(n, d)$ es el problema principal de la teoría de códigos. Es decir, encontrar el más largo código de una longitud dada y mínima distancia. En los capítulos anteriores (principalmente el 4 que se refiere a los códigos BCH) hemos aprendido que buenos códigos son largos, o más precisamente, dado un canal con cierta probabilidad de error p, podemos reducir la probabilidad de error por buscar una secuencia de códigos incrementando la longitud n. Claramente el número promedio de errores en una palabra recibida es np y por lo tanto d (distancia) debe crecer tan rápido como 2np para corregir estos errores. Esto explica la importancia de *R(C)* que representa el máximo promedio de información o palabras código mientras se mantiene una distancia relativa *δ*. Una distancia mínima relativa implica una relativa baja capacidad de corrección de error. Entonces denotamos R(C) y δ(C) como:

$$R(C) = \frac{\log_q M}{n} \qquad y \qquad \delta(C) = \frac{(d-1)}{n}.$$

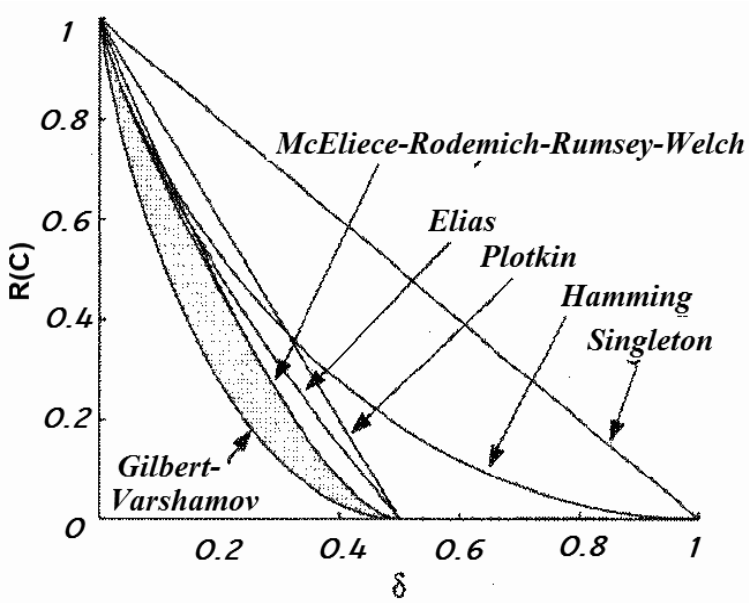


Figura A-1 Comparación de la cota inferior (Gilbert Varshamov) y varios límites superiores.

Para límites superiores de largos valores de n, el mejor límite superior es el de McEliece-Rodemich-Rumsey-Welch, el cual se obtiene por implementar la programación lineal [53; 413]. Otros límites superiores que se aprecian en la figura A-1 son el límite de Elias, Plotkin, Hamming y Singleton. El límite inferior es representado por la cota Gilbert Varshamov.

Consideremos ahora el siguiente código binario de repetición de longitud *n*

$$C = \left\{ \underbrace{00\ \cdots\ 0}_{n}, \underbrace{11\ \cdots\ 1}_{n} \right\}.$$

Es claro, que (n, M, d) = (n, 2, n) y que su distancia es *n*, por lo que *C* un es código con parámetros [*n, k, d*] = [*n, 1, n*], i.e., (*n, M = q^k, d) =(n, 2, n).* Entonces

$$R(C) = \frac{\log_2(2)}{n} = \frac{1}{n} \to 0, \qquad y \qquad \delta(C) = \frac{(n-1)}{n} \to 1.$$

En este caso, conforme $n \to \infty$, $R(C) \to 0$ mientras que $\delta(C) \to 1$. Esto es, conforme aumenta la longitud de la palabra, el código adquiere mayor capacidad para corregir errores a costa de sacrificar su eficiencia, es reflejado por el bajo número de palabras código.

Definición A.1 Dado un alfabeto *A* de tamaño *q (q > 1)* y de valores *n* y *d*, denotemos $A_q(n, d)$ al mayor valor posible de *M* para el que existe un código *(n, M, d)* sobre *A. Es decir*,

$$A_q(n, d) = \text{máx } \{M: \text{existe un código } (n, M, d) \text{ sobre } A\}.$$

Un código *(n, M, d)* cuyo tamaño *M* es el máximo, i.e., $M = A_q(n, d)$ se llama código óptimo. Es importante señalar que el valor $A_q(n, d)$ depende sólo de *q, n y d* y no de *A.*

Cotas inferiores

Teorema A.2 (**Cota de la esfera**). Para un entero $q > 1$ y enteros, *n, d* tales que $1 \leq d \leq n$, tenemos que

$$\frac{q^n}{\sum_{t=0}^{d-1} \binom{n}{i} (q-1)^i} \leq A_q(n, d). \tag{A1}$$

Demostración *Sea* $C = \{c_1, c_2, \ldots, c_M\}$ un código óptimo *(n, M, d)* sobre *A* con $|A|=q$, *i.e., $M = A_q(n, d)$.* Puesto que *C* tiene el mismo tamaño, no puede haber palabra en A^n cuya distancia a alguna palabra en *C* sea mayor o igual a *d.* Si existiera tal palabra, simplemente se incluiría en C, y obtendríamos (n, M+1, d). Esto significa que para todo vector $\boldsymbol{x} \in A^n$ existe al menos una palabra $c_i \in C$ tal que $d(\boldsymbol{x}, \boldsymbol{c_i})$ es a lo mas *d-1*, esto es $\boldsymbol{x} \in S_A(c_i, d-1)$. Por lo tanto, toda la palabra A^n se encuentra en al menos en alguna de las esferas $S_A(c_i, d-1)$. i.e.,

$$A^n \subseteq \bigcup_{4=1}^{M} S_A(c_1, d-1).$$

Como $|A^n| = q^n$ y $|S_A(c_i, d-1)| = V_q^n(d-1)$, para cualquier *i* tenemos que

$$q \leq M \cdot V_q^n(d-1),$$

lo cual implica que $\frac{q^n}{V_q^n(d-1)} \leq M = A_q(n, d)$.

Ejemplo Probaremos que $A_2\,(5, 4) = 2$. La cota de la esfera muestra que $A_2\,(5, 4) \geq 2$. Por la demostración del teorema A.2, tenemos, $A_2\,(5, 4) = A_2\,(4, 3)$, de manera que buscaremos probar $A_2\,(5, 3) \leq 2$. Sea C un código binario (4, M, 3) y sea (x_1, x_2, x_3, x_4) una palabra de C. Dado que $d(C)=3$, las demás palabras en C deben ser de alguna de la siguientes formas

$$(x_1, \overline{x_2}, \overline{x_3}, \overline{x_4}), (\overline{x_1}, x_2, \overline{x_3}, \overline{x_4}), (\overline{x_1}, \overline{x_2}, x_3, \overline{x_4}),$$
$$(\overline{x_1}, \overline{x_2}, \overline{x_3}, x_4), (\overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_4}),$$

donde $\overline{x_4}$ se define como

$$\overline{x_1} = \begin{cases} 1 & \text{si} \quad x_1 = 0; \\ 0 & \text{si} \quad x_1 = 1. \end{cases}$$

Sin embargo, ningún par de estas cinco palabras tiene distancia 3 (o mayor) entre si, de manera que solo una de ellas puede ser incluida en el código C. Por lo tanto $M \leq 2$, lo cual implica que $A_2\,(4, 3) \leq 2$. Por lo tanto, $A_2\,(5, 4) = A_2\,(4, 3) = 2$.

(**La cota de Gilbert – Varshamov**). La cota de Gilbert – Varshamov es una cota inferior para códigos lineales conocida desde 1950.

Teorema A.3 Sean n, k y d enteros que satisfacen $2 \leq d \leq n$ y $1 \leq k \leq n$. Si

$$\sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i < q^{n-k}, \tag{A2}$$

entonces existe un código lineal con parámetro [n, k] sobre F_q con distancia mínima mayor o igual a d.

Demostración Probaremos que, si (A2) se satisface existe una matriz H de $(n - k)$ x n sobre F_q tal que todas las $d - 1$ columnas de H son linealmente independiente. Sea c_j la cual denota la jth columna de H. Para c_1 cualquier vector distinto de cero en F_q^{n-k}. Sea c_2 cualquier vector que no se encuentre en la expansión de c_1, esto es, que no pueda ser generado por medio de alguna combinación lineal de c_1. Para cualquier $2 \leq j \leq n$, sea c_j cualquier vector que no se encuentra en la expansión lineal de $d - 2$ (o menos) vectores $c_1, c_2, \ldots, c_{j-1}$. El número de vectores en la ecuación lineal de d – 2 de las palabras $c_2, \ldots, c_{j-1}$ $(2 \leq j \leq n)$ está dada por

$$\sum_{i=0}^{d-2} \binom{j-1}{i} (q-1)^i \leq \sum_{i=0}^{d-2} (q-1)^i < q^{n-k}.$$

Por lo tanto, el vector c_j $(2 \leq j \leq n)$ siempre puede encontrarse.

Ejemplo Sea q = 2, n = 5 y d = 4. Del teorema A.3 encontramos $A_2\,(4, 3) = 5$ y por lo tanto $A_2\,(5, 4) = 2$ lo que garantiza la existencia de un código $[5, 1, 4]$.

Cotas superiores

Teorema A.4 (**cota de Hamming**). Para un entero $q > 1$ y siendo n, d enteros tales que $1 \le d \le n$, tenemos que

$$A_q(n, d) \le \frac{q^n}{\sum_{i=0}^{\left\lfloor \frac{d-1}{2} \right\rfloor} \binom{n}{i} (q-1)^i}. \qquad \text{(A3)}$$

Demostración Sea $C = \{c_1, c_2, \ldots, c_m\}$ un código óptimo de parámetros (n, M, d) sobre A (con $|A| = q$), de manera que $M = A_q(n, d)$. Sea

$$e = \lfloor (d-1)/2 \rfloor ;$$

Entonces, las esferas $S_A(c_i, e)$ estan disjuntas. Por está razón, tenemos que.

$$\bigcup_{i=1}^{M} S_A(c_i, e) \subseteq A^n ,$$

donde la unión de esferas es disjuntas. Puesto que $|A^n| = q^n$ y $|S_A(c_i, e)| = V_q^n(e)$ para cualquier i, se cumple que

$$M \cdot V_q^n(e) \le q^n ,$$

lo cual implica que

$$A_q(n, d) = M \le \frac{q^n}{V_q^n(e)} = \frac{q^n}{V_q^n(\lfloor (d-1)/2 \rfloor)}.$$

Completando así la prueba.

Ejemplo Sea n = 23 y d = 7 con t = 3. Entonces, decimos que el código se llamará perfecto y tiene máximo número de palabras, si se cumple la siguiente igualdad

$$|C| \times \left(\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t} \right) = 2^n ,$$

donde $|C| = A_q(n, d)$, que es el número máximo de palabras código, entonces tenemos

$$|C| = \frac{2^{23}}{\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3}} = 2^{12} = 4096$$

Por lo tanto,

$$2^{12} \times 2^{11} = 2^{23} = 2^n .$$

El código es un código Golay (23, 12, 7) código perfecto que corrige 3 errores.

Teorema A.5 (**Cota de Singleton**). Para cualquier entero q > 1, cualquier entero positivo n y cualquier entero d tal que $1 \le d \le n$, se cumple que

$$A_q(n, d) \le q^{n-d+1}. \tag{A4}$$

Particularmente, cuando q es una potencia prima, los parámetros $[n, k, d]$ de cualquier código lineal sobre F_q satisfacen

$$k + d \le n + 1.$$

Usualmente un código [n, k, n – k + 1] es llamado código de máxima distancia separable MDS.

Demostración Consideremos un código con parámetros (n, M, d) sobre alfabeto A de tamaño q, donde $M = A_q(n, d)$. Denotemos c_i, a cada palabra C. Eliminamos las últimas $d - 1$ coordenadas de cada palabra en C.

$$c_i = \overbrace{\underbrace{x_1 \cdots x_{n-d+1}}_{n-(d-1)=n-d+1} \underbrace{x_{n-d+2} \cdots x_n}_{d-1}}^{n}$$

Como la distancia de C es d, al eliminar $d - 1$ coordenadas en todas sus palabras se tiene un código de longitud $n – d + 1$, y todas las palabras código son distintas entre el máximo número de palabras de longitud n – d + 1 es q^{n-d+1}.

Ejemplo Sea q = 2, n = 13, d = 5. Entonces tenemos de (A.5) A(13, 5) ≤ 512 este limite es llamado MDS[25].

Teorema A.6 (**cota de Plotkin**). Sea $q > 1$ un entero y supongamos que n y d satisfacen $r\, n < d$, donde $r = 1 - q^{-1}$. Entonces,

$$A_q(n, d) \le \left\lfloor \frac{d}{d - rn} \right\rfloor. \tag{A5}$$

(Cota de Plotkin para códigos binarios)

(i) Cuando d es par, $$A_2(n, d) \le \begin{cases} 2 \lfloor d/(2d - n) \rfloor & \text{para } n < 2d, \\ 4d & \text{para } n = 2d. \end{cases}$$

(ii) Cuando d es impar

$$A_2(n, d) \le \begin{cases} 2 \lfloor (d + 1)/(2d + 1 - n) \rfloor & \text{para } n \le 2d+1, \\ 4d + 4 & \text{para } n = 2d + 1. \end{cases}$$

Dado $A_2(8, 5) \le 5$ Es el resultado obtenido con la ecuación (A5).
Dado $A_2(8, 5) \le 4$ Es el resultado obtenido cuando d es impar (mejor límite).

[25] MDS: Máxima distancia separable.

Teorema A.7 Teorema (**Cota de Griesmer**). Para un código con parámetros [n, k, d] sobre F_q donde k ≥ 1. Tenemos

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{q^i} \right\rceil . \qquad \text{(A6)}$$

Ejemplo Sea q=2, n=15 y d =7. De (A6), como $\sum_{i=0}^{4} \left\lceil \frac{7}{2^i} \right\rceil = 15$.

Tenemos que [15, k, 7]. Entonces el código tiene k ≤ 5, entonces se tiene un código [15, 5, 7].

Teorema A.8 (**Cota de Elías**). Sea q, n, d, r ∈ N, q ≥ 2, θ = 1 – q^{-1} y con r ≤ θn y r^2 - 2θnr + θnd > 0. Entonces

$$A(n,\ d) \leq \frac{\theta n d}{r^2 - 2\theta n r + \theta n d} \cdot \frac{q^n}{V_q(n,r)} . \qquad \text{(A7)}$$

Ejemplo Sea q = 2, n = 13, d = 5, con θ = ½. Realizamos la estimación para A (14, 6). El resultado es:

$$A(13,\ 5) = A(14,\ 6) \leq \frac{42}{r^2 - 14r + 42} \cdot \frac{2^{14}}{\sum_{i \leq r} \binom{14}{i}} .$$

La mejor elección es r = 3. Por lo tanto tenemos que A (13, 5) ≤ 162.

A P É N D I C E (B)

En este apartado, estudiaremos la forma de factorizar polinomios $x^n - 1$. Partimos que se requieren conocimientos de elementos primitivos de un campo finito $\mathbf{F}_q$, polinomios minimales (reducibles e irreducibles), co-conjuntos ciclotómicos, etc. Estos conceptos e inclusive las demostraciones de los teoremas presentados a continuación, pueden ser analizados por el lector en cualquier bibliografía de códigos ya que en su mayoría dedican un apartado para este tema.

Ejemplo B.1 Primero representaremos los campos $F_{q^m} = Z_q[x]/\bmod f(x)$ donde f(x) es un polinomio irreducible de grado n > 1. Cada elemento diferente de cero es expresado como: (a) un polinomio de α y (b) una potencia de α (ver tabla B-1). Para factorizar x^7-1 sobre $\mathbf{F}_2$ tenemos n = 7 y q = 2. Por lo tanto, encontramos m = 3 tal que $n|q^m$-1 sobre $\mathbf{F}_2$. Después, encontramos $f(x) = 1 + x + x^3$ irreducible en $\mathbf{F}_2$ tal que $f(\boldsymbol{\alpha}) = 0$.

Potencia de α	Polinomio de α	Representación m-tupla binaria
α^0	1	100
α^1	α	010
α^2	α^2	001
α^3	α + 1	110
α^4	α^2 + α	011
α^5	α^2 + α + 1	111
α^6	α^2 + 1	101
α^7	1	100

Tabla B-1 F_{2^3} **generado por f(x) = 1 + x + x^3.**

Como encontramos que el polinomio mínimal de un elemento primitivo $\alpha \in F_{q^m}$, si deseamos encontrar el polinomio mínimal de α^i, para cualquier *i*. Primero, comenzaremos con la definición de un co-conjunto ciclotómico.

Definicion B.2 Sea n co-primo a q. El co-conjunto ciclotómico de q módulo n conteniendo *i* es definido por

$$C_i = \{(i \cdot q^j \,(\bmod n)) \in z_n : j = 0,\ 1,\ \ldots\}.$$

Un subconjunto $\{i_1,\ldots, i_t\}$ de Z_n es llamado un completo conjunto representativo de co-conjuntos ciclotómicos q módulo n si $C_{i_1},\ldots,C_{i_t}$ son distintos y $U_{j=1}^{t} C_{i_j} = Z_n$.

Para nuestro ejemplo de la definición B.2, los co-conjuntos ciclotómicos 2 mod 7 son: $C_0 = \{0\}$, $C_1 = \{1, 2, 4\}$ y $C_3 = \{3, 6, 5\}$.

Teorema B.3 Sea α un elemento primitivo de F_{q^m}. Entonces el polinomio mínimal de α^i con respecto a F_q es: $$M^{(i)}(x) = \prod_{j \in C_i} (x - \alpha^j),$$
donde C_i es el único co-conjunto ciclotómico de q módulo $q^m - 1$ que contiene *i*.

Teorema B.4 Sea n un entero positivo con mcd[26](q, n) =1. Suponemos que m es un entero positivo que satisface n| (q^m-1). Sea α un elemento primitivo de F_{q^m} y sea $M^{(j)}(x)$ el polinomio mínimal de α^j con respecto a F_q. Sea $\{S_1, \ldots, S_t\}$ un completo conjunto representativo de co-conjuntos ciclotómicos de q módulo n. Entonces el polinomio x^n-1 tiene la siguiente factorización con polinomios irreducibles mónicos sobre F_q:

$$x^n - 1 = \prod_{i=1}^{t} M^{((q^m-1)s_i/n)}(x).$$

Demostración Suponemos $r = (q^m - 1)/n$. Entonces α^r es una raíz primitiva nth de unidad, y por lo tanto todas las raíces de $x^n - 1$ son 1, α^r, α^{2r}, $\alpha^{(n-1)r}$. Por lo tanto, los polinomios $M^{(ir)}(x)$ son divisores de x^n-1, para todo $0 \le i \le n - 1$. Entonces, tenemos

$$x^n - 1 = mcm\ (M^{(0)}(x), M^{(r)}(x), M^{(2r)}(x),..., M^{((n-1)r)}(x)).$$

Para determinar la factorización de x^n-1, es suficiente determinar todos los distintos polinomios entre $M^{(0)}(x)$, $M^{(r)}(x)$, $M^{(2r)}(x)$, … , $M^{((n-1)r)}(x)$. Si conocemos que el grado del polinomio minimal α^i es igual al tamaño del co-conjunto ciclotómico que contiene *i*. Entonces, $M^{(ir)}(x) = M^{(jr)}(x)$ si y solo si *ir* y *ij* se encuentran en el mismo co-conjunto ciclotómico de q módulo $q^m - 1= rn$; i.e., i y j estan en el mismo co-conjunto ciclotómico de q módulo n. Esto implica que todos los distintos polinomios entre $M^{(0)}(x)$, $M^{(r)}(x)$, $M^{(2r)}(x)$, … , $M^{((n-1)r)}(x)$ son

$$(M^{(S_1 r)}(x),\ M^{(S_2 r)}(x),\ \ ...,\ \ M^{(S_t r)}(x)).$$

Con lo que el teorema **B.4** queda demostrado.

Corolario B.5 Sea n un entero positivo con mcd (q, n) = 1. Entonces el número de factores irreducibles mónicos de $x^n - 1$ sobre F_q es igual al número de co-conjuntos ciclotómicos de q módulo n.

Entonces, para el ejemplo **B.1** tenemos que:

$$M^{(o)}(x) = \prod_{j \in C_i} (x - \alpha^0) = x + 1$$
$$M^{(1)}(x) = \prod_{j \in C_i} (x - \alpha^j) = (x - \alpha^1)(x - \alpha^2)(x - \alpha^4) = x^3 + x + 1$$
$$M^{(3)}(x) = \prod_{j \in C_i} (x - \alpha^j) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^5) = x^3 + x^2 + 1$$

La factorización es: $x^7 - 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1)$

[26] mcd: se refiere al máximo común divisor y mcm: se refiere al mínimo común multiplo.

Ejemplo B.6 Para factorizar $x^{15} - 1$ sobre $\mathbf{F}_2$ tenemos n = 15 y q = 2. Por lo tanto, encontramos m = 4 tal que $n|q^m-1$ sobre $\mathbf{F}_2$. Después, encontramos $f(x) = 1 + x + x^4$ irreducible en $\mathbf{F}_2$ (Ver tabla B-2), tal que $f(\alpha) = 0$.

Potencia de α	Polinomio de α	Representación m-tupla binaria
α^0	1	1000
α^1	α	0100
α^2	α^2	0010
α^3	α^3	0001
α^4	$\alpha+1$	1100
α^5	$\alpha^2+\alpha$	0110
α^6	$\alpha^3+\alpha^2$	0011
α^7	$\alpha^3+\alpha+1$	1101
α^8	α^2+1	1010
α^9	$\alpha^3+\alpha$	0101
α^{10}	$\alpha^2+\alpha+1$	1110
α^{11}	$\alpha^3+\alpha^2+\alpha$	0111
α^{12}	$\alpha^3+\alpha^2+\alpha+1$	1111
α^{13}	$\alpha^3+\alpha^2+1$	1011
α^{14}	α^3+1	1001
α^{15}	1	1000

Tabla B-2 F_{2^4} **generado por f(x) = 1 + x + x^4.**

Para nuestro ejemplo de la definición B.2 los co-conjuntos ciclotómicos 2 mod 15 son:

$C_0 = \{0\}$, $C_1 = \{1, 2,4, 8\}$, $C_3 = \{3, 6, 12, 9\}$, $C_5 = \{5, 10\}$ y $C_7 = \{7, 14, 13, 11\}$.

Para el ejemplo **B.6** tenemos que los co-conjuntos representativos son {0, 1, 3, 5, 7}. Entonces, se tiene

$$M^{(o)}(x) = \prod_{j \in C_i} (x - \alpha^0) = x + 1$$

$$M^{(1)}(x) = \prod_{j \in C_i} (x - \alpha^j) = (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8) = x^4 + x + 1$$

$$M^{(3)}(x) = \prod_{j \in C_i} (x - \alpha^j) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^9) = x^4 + x^3 + x^2 + x + 1$$

$$M^{(5)}(x) = \prod_{j \in C_i} (x - \alpha^j) = (x - \alpha^5)(x - \alpha^{10}) = x^2 + x + 1$$

$$M^{(7)}(x) = \prod_{j \in C_i} (x - \alpha^j) = (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{13})(x - \alpha^{11}) = x^4 + x^3 + 1$$

La factorización es:

$$x^{15} - 1 = (x+1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x^4 + x^3 + 1).$$

El método anterior para obtener la factorización de un polinomio x^n-1, junto con los resultados obtenidos son muy útiles cuando estudiamos códigos BCH[27].

---

[27] Se invita al lector a revisar [47] Apéndices A y B el cual ofrece listas para polinomios minimales hasta F_{2^m} con $1 < m \leq 10$.

A P É N D I C E (C)

Tabla C-1 Códigos BCH generados por elementos primitivos de orden menor que 2^{10}.

n	k	t	n	k	t	n	k	t	n	k	t	n	k	t
7	4	1		123	19		184	45		698	35		183	119
15	11	1		115	21		175	46		688	36		173	122
	7	2		107	22		166	47		678	37		163	123
	5	3		99	23		157	51		668	38		153	125
31	26	1		91	25		148	53		658	39		143	126
	21	2		87	26		139	54		648	41		133	127
	16	3		79	27		130	55		638	42		123	170
	11	5		71	29		121	58		628	43		121	171
	6	7		63	30		112	59		618	44		111	173
63	57	1		55	31		103	61		608	45		101	175
	51	2		47	42		94	62		598	46		91	181
	45	3		45	43		85	63		588	47		86	183
	39	4		37	45		76	85		578	49		76	187
	36	5		29	47		67	87		573	50		66	189
	30	6		21	55		58	91		563	51		56	191
	24	7		13	59		49	9	1023	553	52		46	219
	18	10		9	63		40	95		543	53		36	223
	16	11	511	502	1		31	109		533	54		26	239
	10	13		493	2		28	111		523	55		16	147
	7	15		484	3		19	119		513	57		11	255
127	120	1		475	4		10	121		503	58			
	113	2		466	5	1023	1013	1		493	59			
	106	3		457	6		1003	2		483	60			
	99	4		448	7		993	3		473	61			
	92	5		439	8		983	4		463	62			
	85	6		430	9		973	5		453	63			
	78	7		421	10		963	6		443	73			
	71	9		412	11		953	7		433	74			
	64	10		403	12		943	8		423	75			
	57	11		394	13		933	9		413	77			
	50	13		385	14		923	10		403	78			
	43	14		376	15		913	11		393	79			
	36	15		367	16		903	12		383	82			
	29	21		358	18		893	13		378	83			
	22	23		349	19		883	14		368	85			
	15	27		340	20		873	15		358	86			
	8	31		331	21		863	16		348	87			
255	247	1		322	22		858	17		338	89			
	239	2		313	23	1023	848	18		328	90			
	231	3		304	25		838	19		318	91			
	223	4		295	26		828	20		308	93			
	215	5		286	27		818	21		298	94			
	207	6		277	28		808	22		288	95			
	199	7	511	268	29		798	23		278	102			
	191	8		259	30		788	24	1023	268	103			
	187	9		250	31		778	25		258	106			
	179	10		241	36		768	26		248	107			
	171	11		238	37		758	27		238	109			
255	163	12		229	38		748	28		228	110			
	155	13		220	39		738	29		218	111			
	147	14		211	41		728	30		208	115			
	139	15		202	42		718	31		203	117			
	131	18		193	43		708	34		193	118			

B I B L I O G R A F Í A

[1] Adámek, Jirí . *Foundations of Coding (Theory and Applications of Error-Correcting Codes with an Introduction to Cryptography and Information Theory).* USA: John Wiley, 1991.

[2] _____. M.L. Agarwal & Seema Gupta. "*BCH codes and Desings*". Department of Statistics University of Delhi, Elsevier Science. 12 September 2003, pp. 20-22.

[3] Anderson, John B. and Seshadri Mohan. *Source and Channel Coding (An Algorithmic Approach).* Massachusetts: Kluwer Academic Publishers, 1991.

[4] _____. A. A Andrade, J.C. Interlando and R. Palazzo Jr. "*Alternant and BCH codes over certain ring"* Computational and Applied Mathematics vol.22 No. 2, 2003, pp. 233-247.

[5] Ayres, Frank JR. *Theory and Problems of Modern Algebra.* New York: Shaum Publisching, 1065.

[6] Berlekamp, R. Elwyn. *Algebraic Coding Theory.* New York: MacGraw-Hill, 1968.

[7] Berlekamp, R. Elwyn. *The Development of Coding Theory.* New York: IEEE Press, 1974.

[8] Bierbrauer, Juergen. *Introduction to Coding Theory.* USA: Chapman & Hall / CRC, 2005.

[9] Blahut E. Richard. *Theory and Practice of Error Control Codes.* New York: Addison-Wesley, 1993.

[10] Bossert, Martin. *Channel Coding for Telecommunications* .New York: John Wiley, 1999.

[11] Buchmann. J, T Høhldt, H. Stichtenoch and H. Tapia. *Coding Theory, Cryptography and Related Areas.* Germany: Springer, 2000.

[12] _____. Fabien Buda, Juing Fang, Philippe Sehier. "*Sofh Decoding of BCH Codes Applied to Multilevel Modulation Codes For Rayleigh Fading Channels"* IEEE Transactions on Circuits and Systems-II Express Briefs, vol. 55 No. 5 May 2008. pp. 32-36.

[13] Couch Leon W. II. *Modern Communication Systems* (Principles and Applications). New Jersey: Prentice Hall, 1995.

[14] Csiszár, Imre and Janos Körner. *Information Theory (Coding Theorems for Discrete Memoryless Systems).* Budapest: Academic Press, 1981.

[15] _____. Darius Dabiri and Ian F. Blake. "*Fast Parallel Algorithms for Decoding Reed-Solomon Codes Based on Remainder Polynomials"* IEEE Transactions on Information Theory, vol 41, No. 4 July 1995, pp. 873-885.

[16] P. J. Deitel and H. M. Deitel. *Java How to Program*. New Jersey: Pearson/Prentice Hall, Seventh Edition, 2007.

[17] Descamps, Sebastià Xambó. *Block Error-Correcting Codes.* Germany: Springer, 2003.

[18] _____. Dianwu Yue and Hongbo Zhu. "*On the Minimum Distance of Composite-Length BCH Codes"*. IEEE Communications Letters, vol. 3, No. 9, Septembre 1999. pp. 269-271.

[19] Embree Paul M. and Kimble Bruce. *C Language Algorithms for Digital Signal Processing.* New Jersey: Prentice Hall, 1991.

[20] Ferrel G. Stremler. *Introduction to Communication Systems.* USA: Addison-Wesley, Second Edition, 1982.

[21] Fishman, George S. *Monte Carlo: Concepts, Algorithms, and Applications.* New York: Springer, 1996.

[22] Frerking, Marvin E. *Digital Signal Processing in Communication Systems.* New York: Chapman & Hall, 1994.

[23] Gardner, Floyd M. and John D. Baker. *Simulation Techniques (Models of Comunication Signals and Processes).* New York: Wiley, 1997.

[24] Gilat, Amos. *Matlab an Introduction with Applications.* USA: John Wiley & Sons, Third Edition, 2008.

[25] Grimaldi, Ralph P. *Matemáticas Discretas y Combinatoria (Una Introducción con Aplicaciones).* Addison Wesley, Tercera Edición, 1998.

[26] Hamidreza S. Jamali and Tho Le-Ngoc. *Coded-Modulation Techniques For Fading Channels.* Massachusetts: Klumer Academic Publishers, 1994.

[27] Hamming, R.W.*Coding and Information Theory.* USA: Prentice-Hall,1980.

[28] Haykin, Simon and Van Veen Barry. *Signals and Systems.* New York: John Wiley & Sons.

[29] Herrera Pérez, Enrique. *Comunicaciones II Comunicación Digital y Ruido.* México: Limusa, 2005.

[30] Hoffman, D.G. D.A. Leonard, C.C. Lindner, K.T. Phelps, C.A. Rodger and J.R. Wall. *Coding Theory (The Essentials).* New York: Marcel Dekker, 1991.

[31] _____. B. Honary, A. Moinian and B. Ammar. "*Construction of well- structured quasi-cyclic Iow-density parity check codes"*. IEEE Proc.-Commun., vol 152, No. 6 December 2005, pp. 1081-1085.

[32] Ibe. Oliver C. *Fundamentals of Applied Probability and Random Processes.* Academic Press, 2005.

[33] Jayant N. S. and Peter Noll. *Digital Coding of Waveforms (Principles and Applications to Speech and Video)*. New Jersey: Prentice Hall, 1984.

[34] Jeruchim Michel C. Philip Balaban and K. Sam Shanmugan. *Simulation of Communication Systems. New York: Klumer Academic/Plenum Publishers, Second Edition, 2000.*

[35] Johnson, R. Johnny. *Introduction to Digital Signal Processing.* New Jersey: Prentice Hall, 1989.

[36] Johnsonbaugh, Richard . *Matemáticas Discretas*. México: Pearson/ Prentice Hall, Sexta Edición, 2005.

[37] Kamen. *Introduction to Signals and Systems* 1 *and* 2.

[38] Karl John H. *An Introduction to Digital Signar Processing.* USA: Academic Press, 1989.

[39] _____. Tadao Kasami. "*A Decoding Procedure for Multiple-Error- Correcting Cyclic Codes"* IEEE, pp. 134-138.

[40] Key Steven M. *Fundamentals of Statiscal Signal Processing: Estimation Theory.* New Jersey: Prentice Hall, 1993.

[41] Klima, Richard E. Neil P. Sigmon and Ernest L Stitzinger. *Applications of Abstract Algebra with Maple and MATLAB.* USA: Chapman & Hall, Second Edition, 2007.

[42] Knuth, Donald E. *Fundamental Algorithms.* Reading, MA: Addison-Wesley, Second Edition, 1973, vol. 1.

[43] _____. Ralf Koetter and Alexander Vardy. "*Algebraic Soft-Decisión Decoding of Reed Solomon Codes"*. IEEE Transactions on Information Theory, vol 49, No. 11 November 2003, pp. 2809-2825.

[44] _____. Keiichiro Koga." *A Simple Decoding of BCH Codes Over GF (2^m)"*. IEEE Transactions on Communications, vol 46, No. 6, June 1998, pp. 709-716.

[45] Larson Roland E. and Hostetler Robert P. *Calculus with Analytic Geometry.* USA: MacGraw-Hill, 1982.

[46] Lee, L.H. Charles. *Error-Control Block Codes for Communications Engineers.* Boston/London: Artech House.

[47] Lin Shu and Daniel J. Costello, Jr. *Error Control Coding: Fundamentals and Applications.* New Jersey: Prentice-Hall.

[48] Ling San and Xin Chaoping. *Coding Theory (A First Course).* Cambridge University Press, First published, 2004.

[49] Lint J.H. van. *Introduction to Coding Theory*. Germany: Springer, Third Edition, 1999.

[50] Lipschutz, Seymour. and Lars Lipson, Marc. *Theory and Problems of Linear Algebra.* USA: MacGraw Hill, Third Edition, 2001.

[51] MacDonald Iain L. and Zucchini Walter. *Hidden Markov and Other Models for Discrete Valued Time Series.* USA: Chapman & Hall, 1997.

[52] MacWilliams, J. Florence and N. J. A Sloane. *The Theory of Error-Correcting Codes.* Netherlands: Elsevier/North-Holland, Second Printing, 1978.

[53] Moon, Todd K. *Error Correction Coding (Mathematical Methods and Algorithms).* New Jersey: Wiley, 2005

[54] Morelos – Zaragoza, Robert H. *The Art of Error Correcting Coding.* New York: John Wiley & Sons.

[55] Ooi**,** James M. *Coding for Channels with Feedback.* Massachusetts: Klumer Academic Publishers, 1998.

[56] Papoulis A. *Probability, Random Variables, and Stochastic Processes.* New York: McGraw Hill, Second Edition, 1984.

[57] _____. Cecilio Pimentel. "*Generating Series for Error Statistics of Block Codes on Channels whith Memory"*. Grupo de Pesquisa em Comunicações – CODEC Departamento de Electrônica e Sistemas Universidade Federal de Pernambuco – UFPE

[58] _____. Ricardo A. Podesta. "*Códigos Cíclicos"* Jornadas de Criptografia y Códigos Autocorrectores (JCCA), 20-24 de Noviembre 2006, Mar del Plata Argentina. pp. 1-29. Bajada en Marzo 2009.

[59] Poli Alain and Llorenç Huguet. *Error Correcting Codes (Theory and Applications).* Paris: Masson/Prentice Hall, 1989.

[60] Reed, I. X. Yin, and T.-K. Truong, *"Algebraic Decoding of (32,16,8) Quadratic Residue Code,"* IEEE Trans. Information Theory, vol. 36, no. 4, pp 876-880, July 1990.

[61] _____. Josep Rifa. "*A new algebraic algorithm to decode the ternary Golay code"* Deparment of Computer Sciences, Universitat Autònoma de Barcelona. Information Processing Letters 68 (1998) 271-278.

[62] Roman Steven. *Coding and Information Theory.* New York: Springer, 1992.

[63] Rong X. Li. *Probability, Random Signals, and Statistics.* USA: CRC Press, 1999.

[64] Rosengrant M. A. *Introduction to Telecommunications.* USA: Pearson/Prentice Hall, Second Edition, 2007

[65] Schwartz, Mischa. *Information, Trasmission, Modulation, and Noise.* USA: McGraw Hill, Fourth Edition, 1990.

[66] Shanmugan and P. Balaban. "*A Modified Monte Carlo Simulation Technique for Evaluation of E K. Error Rate in Digital Communication System"*. IEEE Trans. *Comm,* vol. 28, no. 11, pp. 1916-1924, Nov.1980.

[67] _____. G. Sharma, A. Dholakia, and A. Hassan. "*Simulation of Error Trapping Decoders on a Fading Channel"*. Proc. IEEE Vehicular Technology Conference, Atlanta, GA, Apr.-1 May 1996, vol. 2, pp. 1361-1365.

[68] Stremler, Ferrel G. *Introducción a los Sistemas de Comunicación.* Wilmington, Delaware, USA: Addison-Wesley Iberoamericana, Tercera Edición, 1993.

[69] Thompson. M. Thomas. *From Error Correcting Codes Trough Sphere Packings to Simple Groups.* USA: The Mathematical Association of America, 1983.

[70] Vaseghi, Saeed V. *Advanced Digital Signal Processing and Noise Reduction.* New York: Wiley, Second Edition, 2000.

[71] Volodia, Blinovsky. *Asymptotic Combinatorial Coding Theory.* Massachusetts: Klumer Academic Publishers, 1997

[72] Vucetic Branka and Jinhong Yuan. *Space-Time Coding.* England: John Wiley & Sons, 2003.

[73] _____. Hank Wallace. "*Error Detection and Correction Using the BCH Code"*. 2001, pp. 1-21.

[74] Warner, Sheth. *Classical Modern Algebra.* New Jersey: Prentice – Hall, 1971.

[75] Wickless W. J. *A First Graduate Course in Abstract Algebra.* New York: Marcel Dekker, 2004.

[76] Wiggert Djimitri. *Codes for Error Control and Synchronization.* USA: Artech House, 1998.

[77] Wiggert Ph. D, Djimitri. *Error-Control Coding and Applications.* Massachusetts: Artech House, 1997

[78] Xia Gen, Xiang. *Modulated Coding for Intersybol Interference Channels.* New York: Marcel Dekker, 2001.