INTRODUCCIÓN A LA TEORÍA DE CÓDIGOS CORRECTORES DE ERRORES

John Hermes Castillo Gómez
John Jairo López Santander
Hamilton Mauricio Ruiz

Editorial
Universidad de Nariño

êditorial
Universidad de **Nariño**

INTRODUCCIÓN A LA TEORÍA DE CÓDIGOS CORRECTORES DE ERRORES

INTRODUCCIÓN A LA TEORÍA DE CÓDIGOS CORRECTORES DE ERRORES

John H. Castillo
John Jairo López Santander
Hamilton Mauricio Ruiz

Grupo de investigación
Álgebra, Teoría de Números y Aplicaciones: ERM

êditorial
Universidad de Nariño

Castillo Gómez, John Hermes
Introducción a la teoría de códigos correctores de errores / John Hermes Castillo Gómez ... [y otros]. – San Juan de Pasto : Editorial Universidad de Nariño, 2025

216 páginas : ilustraciones, tablas

Incluye referencias bibliográficas p. 207-210 y reseña de los autores p. 214
ISBN: 978-628-7771-20-8

1. Correctores de errores--Códigos 2. Teoría de códigos 3. Códigos de bloque—Teoría y ejercicios 4. Códigos lineales—Teoría y ejercicios 5. Códigos cíclicos—Teoría y ejercicios I. López Santander, John Jairo II. Ruiz, Hamilton Mauricio

005.72 C352in – SCDD-Ed. 22

SECCIÓN DE BIBLIOTECA

Introducción a la teoría de códigos correctores de errores

© Editorial Universidad de Nariño

© John Hermes Castillo Gómez
jhcastillo@udenar.edu.co
John Jairo López Santander
jlopez12@tulane.edu
Hamilton Mauricio Ruiz
hamilton.ruiz@esap.edu.co

ISBN:978-628-7771-20-8

Corrector de estilo: Germán Chaves Jurado
Diseño y diagramación: John Hermes Castillo Gómez

Fecha de publicación: Marzo 2025

San Juan de Pasto, Nariño, Colombia

Prohibida la reproducción total o parcial, por cualquier medio o con cualquier propósito, sin autorización escrita de los Autores o de la Editorial Universidad de Nariño.

Índice general

Índice de figuras

Índice de tablas

Prefacio

En este texto ofrecemos una introducción a los conceptos básicos de la teoría de códigos correctores de errores, también conocida como teoría de códigos. El principal objetivo de este manuscrito es incentivar el estudio de esta área entre los lectores de habla hispana, en especial en nuestro país. Aquí brindamos un acercamiento a esta área con las demostraciones de algunos de sus resultados fundamentales, con ejemplos y ejercicios para que el lector aplique los conceptos estudiados. Las motivaciones iniciales para la elaboración de este libro se originaron en la tesis de pregrado [23] y en el tercer capítulo del libro [35].

Al lector interesado en profundizar sus conocimientos en estos temas, le recomendamos los textos: "The theory of error correcting codes" [25], "Introduction to the theory of error-correcting codes" [32], "Coding Theory: A First Course" [22], "Coding and information theory" [36] y "Fundamentals of error-correcting codes" [19]. Recientemente, W. Cary Huffman, J.-L. Kim y P. Solé, editaron el libro "Concise Encyclopedia of Coding Theory" [18], en el que se recogen ideas desde los conceptos básicos hasta las fronteras de la investigación en el tema, en particular sus diversas aplicaciones en la ciencia actual.

En cada capítulo de este libro presentamos ejemplos de rutinas computacionales en `SageMath` con el objetivo de mostrar de manera clara y concreta (cuando esto es posible) los conceptos estudiados. Estas implementaciones están basadas en los conceptos teóricos que aquí estudiamos y sustentadas en las capacidades que brinda `SageMath`. Aunque el equipo de desarrolladores de `SageMath` viene haciendo un excelente trabajo en la implementación de muchos conceptos, consideramos que aún hay espacio

para desarrollar nuevas implementaciones. Con esta idea, en cada capítulo dejamos un listado de prácticas en SageMath para que el lector ejercite, y desarrolle el interés por conocer, no solamente los comandos que ya están disponibles en SageMath, sino también la capacidad de crear sus propias implementaciones.

Recomendamos al lector consultar el manual "Coding Theory Release 10.0" disponible en la página web https://doc.sagemath.org/pdf/en/reference/coding/coding.pdf; aunque le advertimos que el mismo está en constante actualización. La combinación de la teoría y la práctica computacional se destaca en el desarrollo de este documento.

Este libro se divide en cinco capítulos. En el Capítulo 1, presentamos de manera general el objetivo de la teoría de códigos, con ejemplos introductorios sobre cómo detectar y corregir errores con símbolos de redundancia. Adicionalmente, en él se da una breve descripción de los orígenes de la teoría de códigos. En el Capítulo 2, damos la definición formal de código de bloque, los parámetros de un código, su capacidad de detección y corrección y algunas propiedades, entre otros conceptos. En el Capítulo 3 se aborda una clase muy interesante de códigos, denominados códigos lineales, cuya estructura algebraica se utiliza para desarrollar los procesos de codificación y decodificación. Describimos sus propiedades y algunas otras nociones relacionados con ellos. El Capítulo 4, se dedica al estudio del problema fundamental de la teoría de códigos, se incluyen algunas de las cotas más conocidas sobre el tamaño de códigos. Finalmente, en el Capítulo 5 se presentan los códigos cíclicos, una clase muy importante de códigos lineales que abre la puerta al uso de estructuras algebraicas y técnicas más especializadas; en particular a la teoría de anillos.

Este trabajo es resultado del proyecto de investigación *Una mirada computacional a la teoría algebraica de códigos* financiado por la Vicerrectoría de Investigación e Interacción Social de la Universidad de Nariño.

San Juan de Pasto, Colombia
New Orleans, Louisiana, Estados Unidos
Noviembre de 2024

1. Comunicación y códigos

En este capítulo exponemos el objetivo de la teoría de códigos, sobre la detección y corrección de errores con símbolos de redundancia. Adicionalmente, aquí damos una breve descripción de los orígenes de la teoría de códigos.

1.1. Esquema de comunicación y ruido

Los medios de información tales como sistemas de comunicación y dispositivos de almacenamiento no son absolutamente confiables, en el sentido en que los mensajes transmitidos sean recibidos correctamente debido a que ruido u otro tipo de interferencia lo impide.

Existe una variedad de mecanismos por los cuales se genera ruido, siendo las formas más comunes: ruido externo o interferencia (producido por el medio de transmisión), ruido interno o inherente (se presenta en los equipos electrónicos, son producidos únicamente por el receptor), ruido térmico (asociado al movimiento rápido y aleatorio de los electrones en un conductor producido por la agitación térmica), ruido atmosférico (se escucha en los receptores de comunicación, aún cuando no hay señal presente. La principal fuente son las tormentas eléctricas) y ruido creado por el hombre (o ruido industrial, aparece en ese afán del hombre por los procesos productivos).

El ruido está presente en estos sistemas y el efecto que produce en los mensajes es la introducción de errores, que modifican los mensajes transmitidos. Ante este problema surge la teoría de códigos correctores de errores, cuyo objetivo principal es detectar y corregir los errores producidos por ruido.

La transmisión de datos desde el punto de vista de la teoría de códigos supone las siguientes reglas:

- Emisor y receptor conocen la codificación de los mensajes a transmitir.
- Si ocurren errores, el receptor es capaz de detectarlos.
- Por un canal de comunicación, sólo palabras del nuevo lenguaje son enviadas.
- Los mensajes recibidos sólo deben contener símbolos que se manejan en el canal, es decir, si se utilizan números en la codificación, no es posible que llegue un mensaje con algún símbolo distinto al aceptado por el emisor y receptor.

Analicemos inicialmente el proceso de comunicación bajo la suposición de que no se presentan errores en la transmisión. Llamaremos *mensaje fuente* al conjunto de mensajes que se desean enviar, los cuales pueden estar en el lenguaje usual. Estos mensajes se codifican; es decir, se les asigna una cadena de símbolos, los cuales son tomados de un conjunto llamado *alfabeto*. Esta asignación se denomina *codificación de la fuente*. Suponiendo que no se presentan errores en la transmisión, los mensajes codificados se envían al destino deseado a través de un *canal de comunicación*. Estos mensajes llegan al siguiente bloque del esquema de comunicación, denominado *decodificación de la fuente*. La función del decodificador es convertir los mensajes recibidos a mensajes fuente, ya que estos fueron codificados inicialmente. Una vez hecho esto, los mensajes son recibidos por el receptor tal y como se encontraban originalmente. El esquema de comunicación descrito anteriormente se muestra en la Figura 1.1, ver [38].

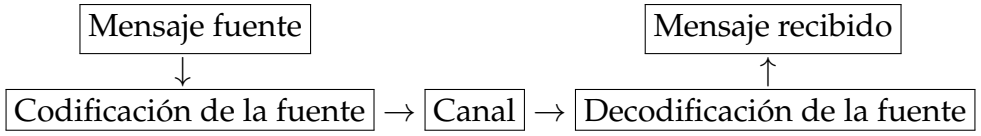


Figura 1.1: Esquema de comunicación.

Este esquema de comunicación presenta una falencia bastante importante originada por la suposición de que no se presentan errores en la transmisión de los mensajes. Es decir, si algún error por ruido se produce en la transmisión es muy probable que el receptor no lo perciba y reciba un mensaje equivocado que lo

lleve a una mala interpretación de la situación o que note el error pero no pueda corregirlo. Para una mayor comprensión de lo dicho anteriormente, analicemos el siguiente ejemplo.

Ejemplo 1.1.1 Supongamos que estamos transmitiendo remotamente las coordenadas de movimiento de un objeto. Las palabras a enviar son: norte, sur, oriente y occidente, las cuales constituyen los mensajes fuente en el esquema de comunicación. Ahora se realiza la codificación de los mensajes fuente mediante asignaciones de cadenas de longitud 2, cuyos símbolos son tomados del conjunto $\{0,1\}$ de la siguiente forma:

$$\text{Norte} \to 00, \quad \text{Sur} \to 01, \quad \text{Oriente} \to 10, \quad \text{Occidente} \to 11.$$

Figura 1.2: Codificación de los mensajes.

Supongamos que el mensaje norte, el cual está codificado como 00 se envía a través de un canal de comunicación con ruido que genera un error en la transmisión. Las posibles cadenas de longitud 2 que se pueden recibir teniendo en cuenta que ha ocurrido un error en el mensaje 00 son: 01 y 10. En cualquier caso, el decodificador de la fuente asignará a esta cadena bien sea sur u oriente, dependiendo del mensaje recibido; y vemos así que la comunicación falla. En la Figura 1.3, ilustramos el caso para el mensaje recibido 10.

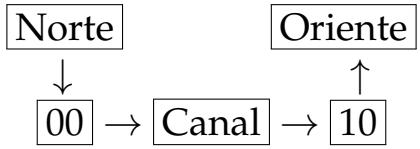


Figura 1.3: Ejemplo del proceso de comunicación. □

La situación en el Ejemplo 1.1.1 nos sugiere que debemos mejorar el esquema de comunicación. Para tal fin, se introducen dos nuevos bloques. El primero de ellos se denomina *codificación del canal*. Su función es realizar una segunda codificación a los mensajes codificados por la fuente, añadiendo unos símbolos de *redundancia* que permitan detectar algún número determinado de errores. El segundo bloque es la *decodificación del canal*, su función es detectar y corregir los errores. De aquí en adelante, el proceso continúa con el decodificador de la fuente el cual ya se describió anteriormente.

En la Figura 1.4 presentamos el nuevo *esquema de comunicación*, ver [38].

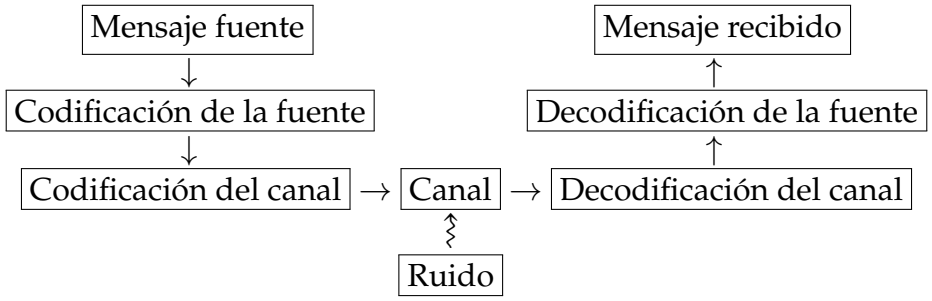


Figura 1.4: Esquema general de un sistema de comunicación.

Ejemplo 1.1.2 En el Ejemplo 1.1.1, podemos realizar la codificación del canal aumentando un símbolo de redundancia a cada cadena de longitud 2 de la siguiente forma:

$$00 \to 000, \quad 01 \to 011, \quad 10 \to 101, \quad 11 \to 110.$$

Figura 1.5: Aumento de un símbolo de redundancia.

Ahora, si se envía el mensaje norte codificado por el canal como 000 y ocurre un error en la transmisión entonces las posibles cadenas recibidas son 001, 010 y 100. En cualquier caso, como ninguna de estas cadenas hace parte de las cadenas codificadas, entonces el decodificador del canal detecta este error. Sin embargo, el decodificador no es capaz de corregir el error, porque por lo menos existen dos cadenas codificadas que difieren en exactamente una posición con cualquier cadena recibida. Por ejemplo, si se recibe la cadena 010, entonces el decodificador no es capaz de identificar la cadena enviada, ya que esta podría haber sido 000, 011 o 110, debido a que estas cadenas difieren en exactamente una posición con la cadena recibida. Ante esta situación procedemos a aumentar más símbolos de redundancia de la siguiente manera:

$$00 \to 00000, \quad 01 \to 01111, \quad 10 \to 10110, \quad 11 \to 11001.$$

Figura 1.6: Más símbolos de redundancia.

Supongamos ahora que se envía el mensaje 00000 y ocurre un error, entonces las posibles cadenas recibidas son 00001, 00010,

00100, 01000 y 10000. Sin pérdida de generalidad, supongamos que la cadena recibida es 01000. El decodificador de la fuente, por un lado detecta que hay errores, dado que la cadena recibida 01000 no hace parte de los mensajes codificados. Por otro lado, teniendo en cuenta la suposición inicial de que solamente un error ocurrió en la transmisión, decodifica la cadena recibida a 00000, ya que entre la cadena recibida y las cadenas del mensaje distintas de 00000 hay por lo menos 2 errores. Una vez realizado este proceso, el decodificador de la fuente la decodifica a 00 y el receptor recibe la palabra norte (Figura 1.7).

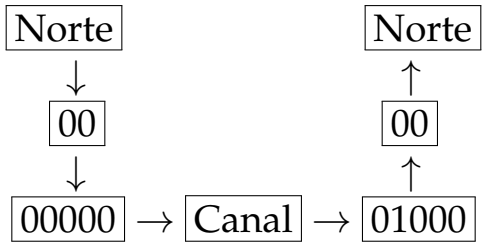


Figura 1.7: Ejemplo modificado. □

Ejemplo 1.1.3 Otra forma para detectar y corregir errores de cualquier longitud es la siguiente. Supongamos que las cadenas que se desean transmitir son de longitud fija k y que están compuestas de 0's y 1's. La codificación del canal se lleva a cabo tomando la cadena a ser enviada y repitiéndola $2r+1$ veces, donde r es un entero positivo fijo. Por ejemplo, si se desea enviar la cadena 01, esta se repite $5 = 2 \times 2 + 1$ veces:

$$01 \to 0101010101.$$

Figura 1.8: Otra forma de codificación.

Observemos que en este caso particular la decodificación de la primer entrada de la palabra enviada se realiza observando el bit que más se repita en la cadena recibida en las posiciones impares; es decir las entradas: 1, 3, 5, 7 y 9. Similarmente, se hace para decodificar la segunda entrada; esto es teniendo en cuenta las entradas pares: 2, 4, 6, 8 y 10.

En general este método garantiza que se pueden corregir hasta r errores; esto significa que con este procedimiento se pueden detectar y corregir errores de cualquier longitud. Sin embargo, el almacenamiento de estas cadenas puede ser muy grande. □

Cabe resaltar que tanto el emisor como el receptor conocen la codificación de los mensajes, por lo tanto si en la transmisión de algún mensaje codificado ocurren a lo sumo el máximo número de errores admitido, el receptor es capaz de detectar el error, aunque no necesariamente corregirlo.

Las consideraciones anteriores sugieren las siguientes preguntas:

- ¿Cómo se realizan los procesos de codificación y decodificación de los mensajes?
- ¿Cómo se detectan y corrigen los errores producidos en la transmisión?

1.2. Inicios y aplicaciones de la teoría de códigos

En las últimas décadas, se ha experimentado un gran aumento en la necesidad de transmisión y almacenamiento de información, por esta razón es cada vez más imprescindible contar con métodos que ayuden a mantener no solamente la seguridad de estos datos sino también su confiabilidad. Justamente por este último aspecto, resulta fundamental detectar y si es posible corregir errores que se puedan generar durante este tipo de procesos. Por ejemplo, al enviar una fotografía pueden surgir errores en la transmisión que distorsionen la imagen original. Tal vez esto puede parecer inconcebible para el usuario actual que envía y recibe fotos desde su celular sin mayores preocupaciones, pero en realidad la detección y corrección de errores trabaja silenciosamente para sustentar estas tareas; de hecho se utiliza en los dispositivos más modernos, entre otras cosas para la recepción de imágenes fiables de planetas tan lejanos como Marte. Es así como aparece a mitad del siglo pasado la teoría de códigos correctores de errores, o simplemente teoría de códigos, disciplina que desde entonces se ha visto influenciada por diferentes ramas como la informática, la computación y, por supuesto, las matemáticas.

El principal objetivo de la teoría de códigos consiste en construir códigos capaces de detectar y corregir errores durante la transmisión de información. La teoría de códigos está soportada especialmente en varias ramas del álgebra y de las matemáticas discretas, entre ellas se destacan: álgebra lineal, teoría de núme-

ros, teoría de anillos, álgebra abstracta, combinatoria, teoría de grafos y las ciencias de la computación.

La historia de la teoría de códigos comenzó como un tópico en Ingeniería Eléctrica con la publicación del artículo "A mathematical theory of communication" [38] de Claude Shannon. En la actualidad es una rama de las matemáticas en la que se aplican diversas técnicas principalmente algebraicas y combinatorias. Claude Shannon mostró que es posible transmitir información de manera confiable a través de una canal siempre y cuando se cumplan ciertas condiciones; sin embargo sus ideas no pudieron ponerse en práctica inmediatamente. Richard Hamming [16] y Marcel Golay [13] posiblemente fueron los primeros en dar construcciones explícitas de códigos. El lector podrá encontrar breves biografías de estos tres científicos en este libro en las secciones 1.5, 2.11 y 4.7, respectivamente. Por supuesto, como en muchas disciplinas matemáticas, son varios los científicos que han aportado para el crecimiento de esta rama. Otros científicos que han realizado valiosos aportes al desarrollo de la teoría de códigos son: David Slepian, Elwyn Berlekamp, Florence Jessie MacWilliams (ver Sección 5.8), Robert G. Gallager, Neil Sloane, Patrick Solé y Vera Pless (ver Sección 3.10) , entre muchos otros.

Una de las formas de aplicar la teoría de códigos para la transmisión de archivos digitales, como imágenes o sonidos, consiste en por ejemplo dividir una imagen o un sonido en pequeñas partes, y utilizar una cadena generalmente binaria para representar cada una de sus partes. De esta forma, se emplean cadenas binarias para comprimir la información o para corregir los errores que pueden ser causados por ruido cuando la información se envía a través de un canal posiblemente ruidoso. Por ruido se entiende cualquier situación que pueda dificultar la comunicación entre dos medios: ruido en el ambiente como por ejemplo el de una aspiradora, el generado por una obra de construcción, el causado por una muchedumbre en una calle, o las dificultades originadas por tormentas eléctricas, defecto en las componentes electrónicas de un sistema o incluso los errores de tipo humano generados al introducir la información.

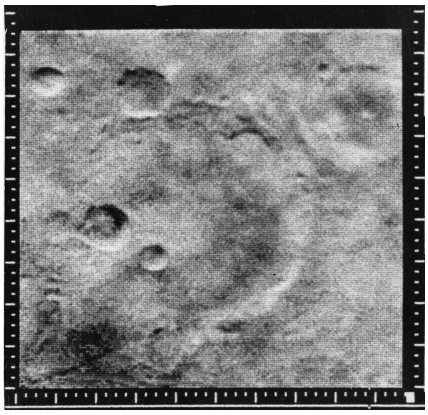
Entre las primeras industrias que se beneficiaron de las técnicas de corrección de errores, naturalmente se encuentran las de la computación y la telefonía; esto porque en ellas la información es transmitida principalmente por medio de computadores. Para

prevenir la aparición de errores en estas situaciones los sistemas computacionales utilizan entre otros los códigos de Hamming. Con el nacimiento del programa espacial, la NASA[1] pasó a ser uno de los principales usuarios y desarrolladores de estas tecnologías; dado que la falta de potencia en los cohetes que se utilizaron en los primeros proyectos, el material que se enviaba debía ser lo más liviano posible, de esta forma los equipos para transmitir la información no eran tan potentes como se deseaba. Para sobrepasar estos inconvenientes se utilizaron códigos correctores de errores para mejorar la fiabilidad y así reconstruir la información enviada. En la Figura 1.9, se presentan imágenes de Marte tomadas por la NASA con ayuda de varias áreas; una de ellas la teoría de códigos. Algunos ejemplos del empleo de la teoría de códigos en los proyectos de la NASA son:

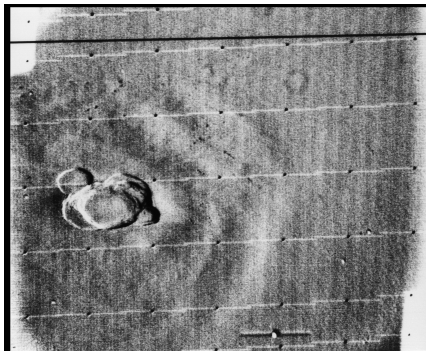
- El Programa Mariner 4 tomó las primeras fotos en blanco y negro de Marte en 1965. Las imágenes eran de tamaño 200×200 pixeles, donde a cada pixel se le asignaba un número entre 64 tonalidades de grises (utilizando 6 bits). La transmisión de una única imagen podría tomar hasta 8 horas (Figura 1.9a).
- En 1972, el Programa Mariner 9 transmitió mejores fotografías con la ayuda de códigos de Reed-Muller que tenían 6 bits de información y 26 bits adicionales. Aunque para este momento la velocidad de transmisión era mucho mejor, las imágenes eran de mayor tamaño, de tal forma que fue necesario almacenarlas (Figura 1.9b).
- El Programa Viking aterrizó en Marte en 1976 y consiguió enviar fotografías a color, tomando imágenes separadas del mismo lugar utilizando tres filtros de colores diferentes. Las imágenes a blanco y negro obtenidas por cada filtro fueron transmitidas y entonces una imagen a color fue reconstruida a partir de la información en las tres imágenes a blanco y negro (Figura 1.9c).

La Figura 1.9d evidencia la mejora de la calidad de las imágenes enviadas. Malkevitch [26] afirma que la NASA utilizó diferentes códigos correctores de errores. Por ejemplo, para las misiones entre 1969 y 1977, la nave Mariner utilizó códigos de Reed-Muller, mientras que para las misiones a Júpiter y Saturno, el

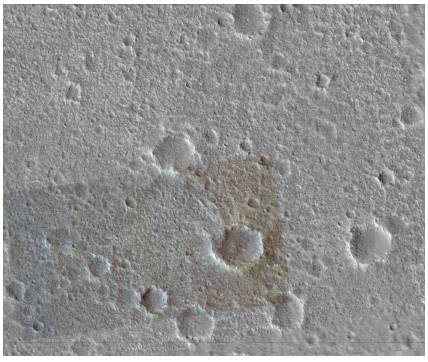
[1] Administración Nacional de Aeronáutica y el Espacio.



(a) Misión Mariner 4.



(b) Misión Mariner 9.



(c) Misión Viking.

(d) Misión Perseverance 2023.

Figura 1.9: Fotografías de Marte tomadas por la NASA https://photojournal.jpl.nasa.gov/.

Voyager 2, empleó un código de Golay. En otros proyectos, la NASA ha usado códigos convolucionales, códigos de Reed-Solomon, turbo códigos y códigos LDPC y combinaciones entre ellos [3].

Según Pinch [29], el valor de la teoría de códigos correctores de errores para la transmisión de informaciones, en la tierra y desde el espacio, fue inmediata y una amplia variedad de códigos fueron construidos, consiguiendo tanto economía en la transmisión, como capacidad de corrección de errores. Entre otras aplicaciones se destaca el uso de códigos para proteger información en los CD's (discos compactos) y CD-ROM's. Según el mismo autor, estos códigos pueden corregir hasta 4000 errores consecutivos y según [26], aunque estos dispositivos están protegidos por derechos de autor, se sabe que utilizan códigos de Reed-Solomon para proteger la información almacenada en ellos.

Entre otras tecnologías que también usan teoría de códigos se resaltan: los módems de los computadores, las comunicaciones satelitales, las transmisiones de radio y televisión de alta definición, entre otras. Vale destacar que cada día se encuentran nuevas aplicaciones. Por ejemplo, Robert J. McEliece [28] presentó un nuevo criptosistema basado en teoría de códigos. A partir de este trabajo surgieron muchas otras investigaciones en las que se están utilizando diferentes tipos de códigos, ver [10]. Más recientemente, Daniel J. Bernstein y sus colaboradores presentaron un criptosistema para computación post-cuántica de clave pública que se encuentra entre los escogidos por el Instituto Nacional de Estándares y Tecnología (NIST) de Estados Unidos como uno de los sistemas más prometedores para reemplazar los sistemas actuales de criptografía, ver [10, 42]. Por supuesto, existen aplicaciones de la teoría de códigos en otras ramas de las matemáticas puras, ver [30]. Otras aplicaciones de la teoría de códigos pueden encontrarse cada vez más en áreas con las que aparentemente no habría relación: sistemas biológicos [37, 47], biología molecular [8] y comunicaciones submarinas [6]. Naturalmente, esta lista cada día se amplia hacia nuevos horizontes. Aunque no somos conscientes los códigos están siempre trabajando todo el tiempo para nosotros, cada vez que escuchamos música, enviamos un mensaje, vemos una película o una transmisión deportiva en la televisión. En este libro estudiaremos los conceptos básicos de la teoría de códigos, esperamos que este trabajo sirva como un preámbulo para estudiantes e investigadores que deseen ahondar en esta emocionante área.

Este documento está apoyado por los cálculos que haremos con ayuda del computador. Para ello utilizaremos `SageMath` [44], el es un software de álgebra computacional de código abierto (open-source), licenciado bajo GPL que combinado con la potencia del lenguaje de programación `Python` se emplea para llevar a cabo cálculos algebraicos, simbólicos y numéricos. Las primeras letras de la sigla `SageMath` hacen referencia a "Software for Algebra and Geometry Experimentation", lo que define el nombre y el propósito original de este programa; aunque en los últimos años ha evolucionado para incluir más áreas de las matemáticas. `SageMath` es un entorno de cálculos matemáticos que introduce datos matemáticos en forma textual y los despliega en forma textual o tradicional. Mientras que la mayor parte de los entornos de cálculo matemático son entidades independientes, `SageMath`

provee algunos algoritmos por sí mismo y otros los toma de otros entornos de cálculo matemático. Esta estrategia le permite a SageMath el poder de múltiples entornos de cálculo matemáticos dentro de una arquitectura capaz de evolucionar para satisfacer futuras necesidades.

Otros programas que utilizan investigadores en matemáticas son: NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, Rand, Matlab, Mathematica y Maple. En el caso del sistema computacional SageMath vale la pena resaltar la gran ventaja que tiene por ser software libre, lo que hace que su distribución sea amplia y gratuita, posibilitando de esta manera que pueda usarse por toda la comunidad académica para investigación, exploración y especialmente para la docencia. Además, permite trabajar con implementaciones realizadas por otros investigadores e incluso, las implementaciones que se realizan pueden contribuir al desarrollo de otras investigaciones de ámbito nacional e internacional.

1.3. SageMath

A continuación presentamos ejemplos de cálculos que podemos realizar con ayuda de SageMath.

Los símbolos $+$, $-$, $*$ y $/$ se utilizan como lo esperado; es decir los cuatro representan inicialmente las operaciones entre números. De esta forma, por ejemplo

```
[In]: 1+1
[Out]: 2
[In]: 5-7
[Out]: -2
[In]: 3*2
[Out]: 6
[In]: 18/3
```

```
[Out]: 6
```

```
[In]: 18/7
```

```
[Out]: 18/7
```

Adicionalmente, el símbolo // representa la división entera. Esto es al introducir el comando $a//b$, SageMath retorna el cociente entero de la divisón entre a y b; esto es

```
[In]: 18//7
```

```
[Out]: 2
```

Podemos asignar en una letra, mayúscula o minúscula, un número y realizar operaciones con ella.

```
[In]: a=5
```

En consecuencia, en la línea anterior se guarda en a el número entero 5. Entonces al escribir

```
[In]: a^2
```

```
[Out]: 25
```

La operación anterior también se consigue con el comando **

```
[In]: a**2
```

```
[Out]: 25
```

Por supuesto en SageMath se pueden realizar más operaciones elementales entre números. Al lector interesado se le sugiere leer el tutorial que se encuentra en el enlace https://doc.sagemath.org/pdf/es/tutorial/SageTutorial.pdf, para conocer más sobre el tema.

Tipos de elementos y objetos en SageMath (Python)

Existen diferentes tipos de datos en SageMath, algunos son **integers**, **floats**, **boolean**, **strings**. Además, SageMath cuenta con estructuras de datos como **lists**, **sets**, **vectors**, entre otras.

A continuación ingresamos un entero, un real (float), una cadena (string), un booleano. Con el símbolo # ingresamos comentarios en una línea de código de SageMath.

```
[In]: 11 # Este es un entero
```

```
[Out]: 11
```

```
[In]: 2.14 # Este es un real (float)
```

```
[Out]: 2.14000000000000
```

```
[In]: "Hello Python 101" # Esta es una cadena
        (string)
```

```
[Out]: 'Hello Python 101'
```

Podemos verificar la clase de un objeto usando el comando `type()`.

```
[In]: type(11)
```

```
[Out]: <class 'sage.rings.integer.Integer'>
```

```
[In]: type(2.14)
```

```
[Out]: <class 'sage.rings.real_mpfr.RealLiteral'>
```

```
[In]: type("Hello Python 101")
```

```
[Out]: <class 'str'>
```

```
[In]: "Hola mundo 122133313"
```

```
[Out]: 'Hola mundo 122133313'
```

En SageMath podemos introducir cadenas con comillas dobles ("cadena") o con comillas sencilas ('cadena').

```
[In]: 'hola  mundo'
```

```
[Out]: 'hola  mundo'
```

```
[In]: float(2) # para convertir integer a float
```

```
[Out]: 2.0
```

Boolean es otra clase importante en SageMath; un Boolean puede tomar dos valores: `True` o `False`.

```
[In]: type(True)
```

```
[Out]: <class 'bool'>
```

```
[In]: type(False)
```

```
[Out]: <class 'bool'>
```

Con los comandos `str`, `bool`, `int` y `float`, podemos realizar transiciones entre clases de elementos diferentes.

```
[In]: str(122121) # Este comando transforma el 7
        número a una cadena
```

```
[Out]: '122121'
```

```
[In]: bool(1)
```

```
[Out]: True
```

```
[In]: bool(0)
```

```
[Out]: False
```

Observemos que una cadena debe estar contenida entre comillas.

```
[In]: "Michael Jackson" # Las comillas pueden
        ser dobles
```

```
[Out]: 'Michael Jackson'
```

```
[In]: 'Michael Jackson' # O simple
```

```
[Out]: 'Michael Jackson'
```

Podemos guardar una string en una variable

```
[In]: Name= "Michael Jackson"
      Name
```

```
[Out]: 'Michael Jackson'
```

Podemos recorrer los elementos de una cadena con indices; observe que **comienzan en 0**, letra a letra

```
[In]: Name[0]
```

```
[Out]: 'M'
```

```
[In]: Name[1]
```

```
[Out]: 'i'
```

```
[In]: Name[2]
```

```
[Out]: 'c'
```

O mediante bloques

```
[In]: Name[1:6] # recorremos de la segunda a la
        sexta letra
```

```
[Out]: 'ichae'
```

```
[In]: Name[0:3] # recorremos las tres primeras
        entradas
```

```
[Out]: 'Mic'
```

```
[In]: Name[5:9] # o cualquier bloque, por ejemplo
        en este desde la sexta a la novena
```

```
[Out]: 'el J'
```

Con el operador + podemos concatenar cadenas

```
[In]: Statement = Name + " is the best"
      Statement
```

```
[Out]: 'Michael Jackson is the best'
```

O repetir cadenas

```
[In]: 3*Name
```

```
[Out]: 'Michael JacksonMichael JacksonMichael
        Jackson'
```

```
[In]: 10*(Name + " ")
```

```
[Out]: 'Michael Jackson Michael Jackson Michael
       Jackson Michael Jackson Michael Jackson
       Michael Jackson Michael Jackson Michael
       Jackson Michael Jackson Michael Jackson'
```

Listas, conjuntos y vectores

Adicionalmente, necesitaremos realizar operaciones y trabajar con listas, conjuntos y vectores.

```
[In]: L=[1,2,3,["casa",2],3]
      L #Esta segunda línea permite que se imprima
        el valor que se guardó en L.
```

```
[Out]: [1, 2, 3, ['casa', 2], 3]
```

Con el comando `.append` podemos ingresar elementos al final de la lista.

```
[In]: L.append(-1); L
```

```
[Out]: [1, 2, 3, ['casa', 2], 3, -1]
```

```
[In]: L.append([0,1]); L
```

```
[Out]: [1, 2, 3, ['casa', 2], 3, -1, [0, 1]]
```

Observemos que en el último comando se agrega la lista $[0,1]$ a L. Esto entre otras cosas significa que una lista puede contener una lista como uno de sus elementos. Por otro lado, el comando `.extend` agrega los elementos de la lista $[9,8]$ a L.

```
[In]: L.extend([9,8]); L
```

```
[Out]: [1, 2, 3, ['casa', 2], 3, -1, [0, 1], 9, 8]
```

Para recorrer los elementos de una lista podemos utilizar un ciclo `for`. Por ejemplo, para imprimir los elementos que hay en las lista L que definimos previamente podemos usar el comando `print` y hacer lo siguiente:

```
[In]: for x in L:
          print(x)
```

```
[Out]: 1
       2
       3
       ['casa', 2]
       3
       -1
       [0, 1]
       9
       8
```

Para construir un conjunto, podemos usar el comando `set`.

```
[In]: C=set({}) # Construye el conjunto vacío.
```

```
[In]: C=set({1,2,3,"casa",3}) # Construye un
        conjunto no vacío.
      C
```

```
[Out]: {1, 2, 3, 'casa'}
```

Como sabemos en un conjunto no puede haber elementos repetidos. Los conjuntos numéricos de los naturales ($\mathbb{N}$), enteros ($\mathbb{Z}$), racionales ($\mathbb{Q}$), reales ($\mathbb{R}$) y complejos ($\mathbb{C}$) vienen cargados en SageMath con ayuda de los comandos `NN`, `ZZ`, `QQ`, `RR` y `CC`, respectivamente.

```
[In]: NN # Conjunto de los naturales
```

```
[Out]: Non negative integer semiring
```

```
[In]: ZZ # Conjunto de los enteros
```

```
[Out]: Integer Ring
```

```
[In]: QQ # Conjunto de los racionales
```

```
[Out]: Rational Field
```

```
[In]: RR # Conjunto de los reales
```

```
[Out]: Real Field with 53 bits of precision
```

```
[In]: CC # Conjunto de los complejos
```

```
[Out]: Complex Field with 53 bits of precision
```

Vectores

Los vectores merecen atención especial en nuestro trabajo, puesto que como veremos estos serán los elementos principales en un código. Podemos ingresar vectores con ayuda del comando vector(R,[a1,a2,...,an]), donde R representa el anillo en el que están las entradas y [a1,a2,...,an] identifican las componentes del vector.

```
[In]: vector(ZZ,[1,0,2])
```

```
[Out]: (1, 0, 2)
```

De esta forma, este último es un vector cuyas entradas están en los enteros. Por otro lado, si escribimos

```
[In]: vector(RR,[1,0,2])
```

```
[Out]: (1.00000000000000, 0.000000000000000,
        2.00000000000000)
```

El software identifica que este vector tiene sus entradas en los reales. Observe que si escribimos vector(ZZ,[1.1,0,2]) obtendremos un error, porque la primera entrada 1.1 no es un número entero.

Si se escribe vector([a1,a2,...,an]), el programa asume el anillo en el que están las componentes por contexto.

```
[In]: V1=vector([1.1,2,3,3])
      V1
```

```
[Out]: (1.10000000000000, 2.00000000000000,
        3.00000000000000, 3.00000000000000)
```

En el anterior ejemplo, la primera entrada es un número real, entonces SageMath identifica que todas las entradas son números reales. Observemos que el comando `.base_ring()` identifica el anillo al que pertenecen las entradas.

```
[In]: V1.base_ring()
```

```
[Out]: Real Field with 53 bits of precision
```

```
[In]: V1+V1
```

```
[Out]: (2.20000000000000, 4.00000000000000,
        6.00000000000000, 6.00000000000000)
```

Para ingresar las entradas [a1,a2,...,an] de un vector también podemos utilizar las corchetes [a1,a2,...,an], paréntesis (a1,a2,...,an) o llaves {a1,a2,...,an}.

```
[In]: V2=vector({1,2,3,3.2})
      V2
```

```
[Out]: (3.20000000000000, 1.00000000000000,
        2.00000000000000, 3.00000000000000)
```

```
[In]: V4=vector(IntegerRing(),{1,2,3})
      V4
```

```
[Out]: (1, 2, 3)
```

Cuerpos finitos en SageMath

En SageMath también se puede trabajar con cuerpos finitos. Estos conjuntos serán considerados como el alfabeto para construir elementos de un código.

```
[In]: GF(2) # Cuerpo Finito F_2
```

```
[Out]: Finite Field of size 2
```

```
[In]: K=GF(5); type(K)
```

```
[Out]: <class 'sage.rings.finite_rings.
         finite_field_prime_modn.
         FiniteField_prime_modn_w
         ith_category'>
```

```
[In]: K25= GF(5^2,'c'); type(K25)
```

```
[Out]: <class'sage.rings.finite_rings.
       finite_field_givaro.
       FiniteField_givaro_with_category'>
```

```
[In]: k = GF(9, 'a')
      k
```

En la línea anterior la letra a representa un generador del cuerpo finito.

```
[In]: for x in k:  #esta es una forma de recorrer
        los elementos de un cuerpo finito
          print(x)
```

```
[Out]: 0
       a
       a + 1
       2*a + 1
       2
       2*a
       2*a + 2
       a + 2
       1
```

```
[In]: len(k) # Este cuerpo tiene 9 elementos
```

```
[Out]: 9
```

```
[In]: set(k) # otra forma de ver todos los elementos
```

```
[Out]: {0, 1, 2, a, a + 1, a + 2, 2*a, 2*a + 1,
       2*a + 2}
```

Como mencionamos anteriormente, la letra a que aparece en el comando GF(3^2,'a') está pensada para representar la clase

lateral $[x]$ en el anillo cociente $\frac{\mathbb{F}_3[x]}{\langle f(x)\rangle}$ que es isomorfo al cuerpo finito $\mathbb{F}_9$. Sin embargo, para poder manipularlo hay que usar el siguiente comando.

```
[In]: K.<a>=GF(9)

[In]: a+a+a

[Out]: a

[In]: (a+1)^2

[Out]: a

[In]: parent(a)

[Out]: Finite Field in a of size 3^2

[In]: minimal_polynomial(a)

[Out]: x^2 + 2*x + 2
```

A continuación construimos el cuerpo finito $\mathbb{F}_{16} = GF(16)$. Dado que $x^4 + x + 1$ es irreducible sobre $\mathbb{F}_2$, sabemos que

$$GF(16) = \{ax^3 + bx^2 + cx + d + \langle x^4 + x + 1\rangle : a,b,c,d \in \mathbb{F}_3\}.$$

```
[In]: F16.<c>=GF(16)

[In]: type(F16)

[Out]: <class'sage.rings.finite_rings.
         finite_field_givaro.
         FiniteField_givaro_with_category'>

[In]: c.minimal_polynomial()

[Out]: x^4 + x + 1
```

La función anterior, calcula el polinomio irreducible que sirve para construir el cuerpo finito. Además, podemos realizar operaciones entre los elementos del cuerpo finito como mostramos a continuación.

```
[In]: (c^3+c^2+c+1)*(c^3+c)
```

```
[Out]: c^3 + c^2
```

```
[In]: (c^3+c^2+c+1)*(c^3+c)
```

```
[Out]: c^3 + c^2
```

```
[In]: c^10+ c^7
```

```
[Out]: c^3 + c^2
```

```
[In]: discrete_log(c^10+ c^7,c) # Este comando
        calcula el logaritmo discreto.
```

```
[Out]: 6
```

```
[In]: F16.primitive_element()
```

```
[Out]: c
```

```
[In]: S=[]
      for i in [1..15]:
          S.append(c^i)
      S.append(F16(0)); S
```

```
[Out]: [c,
        c^2,
        c^3,
        c + 1,
        c^2 + c,
        c^3 + c^2,
        c^3 + c + 1,
        c^2 + 1,
        c^3 + c,
        c^2 + c + 1,
        c^3 + c^2 + c,
        c^3 + c^2 + c + 1,
        c^3 + c^2 + 1,
        c^3 + 1,
        1,
        0]
```

```
[In]: list(F16)
```

```
[Out]: [0,
        c,
        c^2,
        c^3,
        c + 1,
        c^2 + c,
        c^3 + c^2,
        c^3 + c + 1,
        c^2 + 1,
        c^3 + c,
        c^2 + c + 1,
        c^3 + c^2 + c,
        c^3 + c^2 + c + 1,
        c^3 + c^2 + 1,
        c^3 + 1,
        1]
```

1.4. Ejercicios

1.4.1. Ejercicios teóricos

1. En el Ejemplo 1.1.1, realice el proceso de comunicación suponiendo que se envía la palabra Sur y se recibe la palabra Norte. ¿Cuántos errores sucedieron en la comunicación?
2. En el Ejemplo 1.1.1, realice el proceso de comunicación suponiendo que se envía la palabra Occidente y se recibe la palabra Sur. ¿Cuántos errores sucedieron en la comunicación?
3. En el Ejemplo 1.1.2, realice el proceso de comunicación suponiendo que se envía la palabra Sur y se recibe la palabra Norte.
4. En el Ejemplo 1.1.3, suponga se desea enviar la cadena 10, ¿cuál es la cadena que recibirá?
5. Suponga en el Ejemplo 1.1.3 que $r = 4$. ¿Cuál es la cadena recibida si se transmite la cadena 101?
6. Demuestre (en general) que si se realiza el procedimiento propuesto en el Ejemplo 1.1.3, se pueden detectar y corregir hasta r errores.

1.4.2. Prácticas en SageMath

Aunque tal vez existen funciones en `SageMath` (`Python`) que realicen las tareas propuestas a continuación, inicialmente trate de realizarlas por usted mismo. Por supuesto, puede usar funciones auxiliares que hayamos construido previamente o existentes en el software. Posteriormente, una vez haya trabajado (ojalá exitosamente) en los ejercicios compare su función con la que provee `SageMath`.

1. Construya en `SageMath` una función que intercambie el primero y el último elemento en una lista.
2. Construya en `SageMath` una función que intercambie dos elementos en una lista, sin importar las posiciones de los elementos de la lista.
3. Construya en `SageMath` una función que muestre en una lista o en una cadena (`string`) la expresión binaria para cualquier entero positivo n dado.
4. Escriba en `SageMath` una función que reciba una palabra y devuelva `True` si la palabra tiene más de tres vocales, de lo contrario que regrese `False`.
5. ¿Podría construir una función en `SageMath` que realice el procedimiento indicado en el Ejemplo 1.1.3?

1.5. Biografía

Claude Elwood Shannon (1916-2001)

Claude Shannon fue un matemático y científico de la computación estadounidense considerado como el "padre de la teoría de la información". Shannon realizó importantes contribuciones en el campo de la teoría de la información, la teoría de la comunicación y la electrónica. Su trabajo sentó las bases para el desarrollo de las comunicaciones digitales y la computación moderna. En 1948, Shannon publicó el artículo titulado "A Mathematical Theory of Communication" [38], en el que presentó la teoría de la información. En este artículo, Shannon estableció los fundamentos matemáticos para cuantificar la información y analizar cómo se puede transmitir de manera eficiente y segura a través de canales de comunicación.

En su tesis de maestría titulada "A Symbolic Analysis of Relay and Switching Circuits" [39], introdujo la noción de "bit" (dígito binario), que es la unidad básica de información. Shannon demostró que la información se puede medir en términos de bits, y desarrolló métodos para calcular la capacidad de transmisión de un canal y la tasa de compresión de datos. También trabajó en el diseño de circuitos digitales y es conocido por su trabajo en los circuitos de interruptores de relé utilizando álgebra booleana. Estos avances sentaron las bases para el desarrollo de la electrónica digital y los sistemas de computadoras modernas. Trabajó en los Laboratorios Bell (1941–1972), y fue profesor en el Instituto de Tecnología de Massachusetts (MIT) entre 1956 y 1978. Adicionalmente, hizo contribuciones en: criptografía (se resalta su artículo de 1949 "Communication theory of secrecy systems" [40]), inteligencia artificial (donde se destaca la creación de software para juegos de ajedrez [41]), y un ratón electrónico para resolver problemas de laberinto.

Claude Shannon recibió numerosos premios y reconocimientos a lo largo de su carrera, incluyendo la Medalla Nacional de Ciencia de Estados Unidos en 1966. Su trabajo ha tenido un impacto profundo en los campos de las comunicaciones, la criptografía, la computación y la teoría de la información, y su legado continúa siendo relevante en la actualidad.

Foto tomada de MacTutor.
https://mathshistory.st-andrews.ac.uk/Biographies/Shannon/

2. Códigos

En la primera sección de este capítulo se introducen las nociones fundamentales de los códigos de bloque que serán utilizadas a lo largo del presente trabajo. Posteriormente, se presenta la noción de distancia de Hamming, la regla de decodificación utilizando la distancia de Hamming y otros conceptos claves. Seguidamente, se tratarán las nociones de códigos equivalentes y códigos perfectos.

2.1. Códigos de bloque

En esta sección definimos los códigos de bloque junto con algunas de sus componentes más importantes.

Definición 2.1.1 Sea $A = \{a_1, a_2, \ldots, a_q\}$ un conjunto finito, llamado *alfabeto* y A^n el conjunto de todas las n-uplas sobre A. Cualquier subconjunto no vacío C de A^n se denomina *código de bloque q-ario*. Cada elemento en C se llama *palabra código* y si C contiene M elementos, entonces se dice que el código C tiene longitud n y tamaño M o simplemente que C es un (n, M)-código. Un subconjunto de un código C, es un *subcódigo* de C.

Si los elementos del conjunto C no tienen la misma longitud, se dice que C *es un código de longitud variable*. Un ejemplo de código de longitud variable es el *código Morse*. El código C sobre el alfabeto $A = \{0,1\}$ definido por $C = \{(0),(1,0),(1,1),(1,0,0),(1,0,1),(1,1,1)\}$ es también de esta clase. Sin embargo, en este libro no abordaremos esta clase de códigos, así que en adelante nos referiremos a códigos de bloque.

Normalmente el conjunto es un campo de Galois de orden q, es decir $A = \mathbb{F}_q$, donde q es una potencia prima. Cuando un código C tiene como alfabeto a $\mathbb{F}_2$, $\mathbb{F}_3$ o $\mathbb{F}_4$, entonces C es un *código binario*, *ternario* o *cuaternario* respectivamente, siendo los códigos binarios los más conocidos en la práctica.

Ejemplo 2.1.2 Los siguientes son códigos de bloque.

1. $C_1 = \{(0,0,0),(0,1,1),(1,1,0),(1,1,1)\}$ es un $(3,4)$-código binario.

2. $C_2 = \{(2,0,1,0),(1,0,2,0),(1,1,2,2),(2,2,1,1),(2,1,2,1)\}$ sobre $\mathbb{F}_3$ es un $(4,5)$-código ternario.

3. $C_3 = \{(1,\alpha,\alpha),(\alpha+1,0,1),(1,0,\alpha),(\alpha,\alpha+1,1)\}$ sobre $\mathbb{F}_4$ es un $(3,4)$-código cuaternario.

4. $C_4 = \{(a,a,\ldots,a) \in \mathbb{F}_q^n : a \in \mathbb{F}_q\}$ es un (n,q)-código q-ario conocido como *código de repetición*. □

Cuando no haya confusión, se denotará un vector de la forma $(v_1,v_2,\ldots,v_n)$ simplemente como $v_1v_2\cdots v_n$.

Una de las características que define que "tan bueno" es un código es su tasa de información.

Definición 2.1.3 La *tasa de información* de un (n,M)-código q-ario C, se define por

$$R(C) = \frac{\log_q |C|}{n}.$$

Esta tasa se interpreta como el número de coordenadas que guardan información del mensaje original sobre el total de las coordenadas trasmitidas. De ahí que uno de los objetivos de la teoría de códigos es buscar códigos con alta tasa de información, digamos $R > \frac{2}{3}$ o $R > \frac{3}{4}$.

Ejemplo 2.1.4 Para los códigos del Ejemplo 2.1.2 se tiene

$$R(C_1) = \frac{\log_2 4}{3} = \frac{2}{3},$$
$$R(C_2) = \frac{\log_3 5}{4} \approx 0.3662,$$
$$R(C_3) = \frac{\log_4 4}{3} = \frac{1}{3}.$$

Note que $|C_1| = |C_3|$; sin embargo C_1 tiene una mejor tasa de información. □

Al transmitir una palabra se puede recibir una palabra diferente a la enviada; esto es, pueden ocurrir cambios en las componentes de la palabra original. Cada uno de estos cambios se conoce como un error. Es decir, al enviar $\mathbf{x} \in \mathbb{F}_q^n$ se puede recibir $\mathbf{y} \in \mathbb{F}_q^n$ tal que $\mathbf{x} \neq \mathbf{y}$. De esta forma, $\mathbf{x} = \mathbf{y} + \mathbf{e}$, donde $\mathbf{e}$ se denomina vector de errores y el número de componentes no nulas de $\mathbf{e}$ es la cantidad de errores cometidos. En la siguiente sección se presenta el concepto de distancia de Hamming de un código, el cual tiene una relación directa con la capacidad de detección y corrección de errores.

2.2. Distancia de Hamming

La noción de distancia de Hamming se denomina así en honor al matemático Richard Hamming, ver Sección 2.11, científico estadounidense quien introdujo este concepto para el estudio de las capacidades de detección y corrección de errores en un código.

Definición 2.2.1 Sean $\mathbf{x}, \mathbf{y}$ palabras de longitud n sobre un alfabeto A. La *distancia de Hamming* entre $\mathbf{x}$ y $\mathbf{y}$, denotada por $d(\mathbf{x}, \mathbf{y})$, se define como el número de posiciones en las cuales $\mathbf{x}$ y $\mathbf{y}$ difieren, es decir: si $\mathbf{x} = x_1 x_2 \cdots x_n$ y $\mathbf{y} = y_1 y_2 \cdots y_n$, entonces

$$d(\mathbf{x}, \mathbf{y}) = |\{i : 1 \leq i \leq n, x_i \neq y_i\}|. \tag{2.1}$$

Ejemplo 2.2.2 Sean $A = \{0,1,2\}$, $\mathbf{x} = 10112$ y $\mathbf{y} = 20110$. Entonces

$$\begin{aligned} d(\mathbf{x},\mathbf{y}) &= |\{i : 1 \leq i \leq n, x_i \neq y_i\}| \\ &= |\{1,5\}| = 2. \end{aligned}$$

□

El siguiente resultado demuestra que la distancia de Hamming es una métrica.

Teorema 2.2.3 Sean $\mathbf{x}$, $\mathbf{y}$, y $\mathbf{z}$ palabras de longitud n sobre A. Entonces

1. $0 \leq d(\mathbf{x},\mathbf{y}) \leq n$ y $d(\mathbf{x},\mathbf{y}) = 0$ si y sólo si $\mathbf{x} = \mathbf{y}$.
2. $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$.
3. $d(\mathbf{x},\mathbf{z}) \leq d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z})$ (desigualdad triangular).

Demostración. Los dos primeros ítems son claros a partir de la definición de distancia de Hamming. Para probar el tercero, consideremos los siguientes conjuntos

$$A = \{i : x_i = z_i\} \quad \text{y} \quad B = \{i : x_i = y_i \ \wedge \ y_i = z_i\}.$$

Es claro que $B \subseteq A$ y en consecuencia $A^c \subseteq B^c$. Observe que

$$\begin{aligned} &d(\mathbf{x},\mathbf{z}) = |A^c|, \quad \text{y} \\ &\quad B^c = \{i : x_i \neq y_i \vee y_i \neq z_i\} = \{i : x_i \neq y_i\} \cup \{i : y_i \neq z_i\} = C \cup D, \end{aligned}$$

donde $C = \{i : x_i \neq y_i\}$ y $D = \{i : y_i \neq z_i\}$. De esta manera,

$$\begin{aligned} d(\mathbf{x},\mathbf{z}) = |A^c| \leq |B^c| &= |C \cup D| = |C| + |D| - |C \cap D| \\ &\leq |C| + |D| = d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z}). \end{aligned}$$

■

2.3. Decodificación por distancia mínima

El concepto de distancia de Hamming es de vital importancia debido a su utilidad en el proceso de decodificación. Sea C un código y supongamos que palabras código son enviadas sobre un canal de comunicación. Si $\mathbf{x}$ es una palabra recibida, la *decodificación por distancia mínima* decodifica $\mathbf{x}$ a $\mathbf{c}_\mathbf{x}$, si $d(\mathbf{x},\mathbf{c}_\mathbf{x})$ es mínima entre todas las palabras código, es decir si

$$d(\mathbf{x},\mathbf{c}_\mathbf{x}) = \text{mín}\{d(\mathbf{x},\mathbf{c}) : \ \mathbf{c} \in C\}. \tag{2.2}$$

Existen dos tipos de decodificación, *la decodificación por distancia mínima completa* y la *decodificación por distancia mínima incompleta*. Si una palabra **x** es recibida y 2 o más palabras código satisfacen la ecuación (2.2) se dice que hay un *empate*, entonces la decodificación por distancia mínima completa elige una de ellas arbitrariamente, mientras que la incompleta solicita retransmisión.

Ejemplo 2.3.1 Para el código $C = \{01101, 00011, 10110, 11000\}$. Consideremos la decodificación por distancia mínima incompleta para decodificar las siguientes palabras recibidas.

a) 01111.
Calculamos las distancias

$$d(01111, 01101) = 1, \qquad d(01111, 00011) = 2,$$
$$d(01111, 10110) = 3, \qquad d(01111, 11000) = 4.$$

Se decodifica 01111 como 01101.

b) 11011.
Calculamos las distancias

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

Como hay dos distancias iguales a 2, entonces se solicita retransmisión. □

Definición 2.3.2 La *distancia mínima* de un código C, denotada por $d(C)$, se define por

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

Un código C de longitud n, tamaño M y distancia mínima d se dice que es un (n, M, d)-código. Calcular la distancia de un código consiste en determinar la menor distancia de Hamming entre todos los pares distintos de elementos de C. Sin embargo, si el tamaño del código es relativamente grande, esta comparación puede llevar mucho tiempo. De hecho, para calcular la distancia mínima de un (n, M)-código usando la definición requiere el cálculo de $\binom{M}{2}$ distancias.

2.4. Capacidades de corrección y detección

Veamos ahora cómo la distancia de Hamming de un código está relacionada con su capacidad de detección y corrección de errores.

Definición 2.4.1 Un código C *detecta* t*-errores* si cada vez que se envía una palabra código y ocurren entre 1 y t errores durante la transmisión, la palabra resultante no es una palabra código. Un código C detecta exactamente t-errores, si este detecta t errores, pero no detecta $t+1$ errores (es decir, existe al menos una palabra código para la cual podemos cambiar adecuadamente $t+1$ de sus coordenadas y obtener otra palabra código).

Ejemplo 2.4.2 El código ternario $C = \{000000, 000111, 111222\}$ es un detector de 2-errores ya que al enviar cualquier palabra código y cambiar dos o menos coordenadas de cada palabra, la resultante no es una palabra código. Por ejemplo,

$$\begin{aligned} 000000 &\to 000111 \text{ necesita cambiar 3 coordenadas,} \\ 000000 &\to 111222 \text{ necesita cambiar 6 coordenadas,} \\ 000111 &\to 111222 \text{ necesita cambiar 6 coordenadas.} \end{aligned}$$

De hecho, C detecta exactamente 2-errores, ya que al cambiar las tres últimas coordenadas de la palabra código 000000 por unos, se obtiene la palabra código 000111. □

En general el proceso usado en el ejemplo anterior para códigos de mayor tamaño emplea demasiado tiempo. El siguiente resultado nos permite simplificar los cálculos mediante el uso de la distancia mínima del código.

Teorema 2.4.3 Un código C detecta exactamente t-errores si y sólo si $d(C) = t+1$.

Demostración. Asumamos que $d(C) = t+1$. Supongamos que se envía $\mathbf{c} \in C$ y se recibe $\mathbf{x}$ de tal forma que

$$1 \leq d(\mathbf{c}, \mathbf{x}) \leq t < d(C).$$

Entonces $\mathbf{x} \notin C$, dado que la distancia entre $\mathbf{c}$ y $\mathbf{x}$ es menor que la distancia mínima de C; por lo tanto C es un código detector de t-errores.

Como $d(C) = t + 1$, entonces existen palabras código $\mathbf{c}_1, \mathbf{c}_2 \in C$, tal que $d(C) = d(\mathbf{c}_1, \mathbf{c}_2) = t + 1$. Cambiando en $\mathbf{c}_1$ las $t + 1$ coordenadas en las que difiere con $\mathbf{c}_2$ de tal manera que coincidan con las de este último; naturalmente se obtiene una palabra código. Luego, C no detecta $(t + 1)$-errores y por lo tanto C detecta exactamente t-errores.

Recíprocamente, supongamos que C detecta t-errores. Ahora cambiando t o menos coordenadas de cualquier palabra $\mathbf{c} \in C$, se forma una palabra $\mathbf{x}$ tal que $d(\mathbf{c}, \mathbf{x}) \leq t$, entonces por la hipótesis $\mathbf{x} \notin C$. Luego $d(C) \geq t + 1$. Además, como C no detecta $t + 1$ errores, por definición existen $\mathbf{c}_1, \mathbf{c_2} \in C$ tales que $d(\mathbf{c}_1, \mathbf{c}_2) = t + 1$. Pero como $d(C) \leq d(\mathbf{c}_1, \mathbf{c}_2) = t + 1$. Así $d(C) = t + 1$. ■

El resultado anterior nos permite determinar el número de errores que un código es capaz de detectar. Para determinar la capacidad de corrección de errores de un código se introduce el siguiente concepto.

Definición 2.4.4 Asumiendo que los empates son considerados como errores, un código C *corrige* t-*errores* si la decodificación por distancia mínima corrige todos los errores de tamaño t o menos en cualquier palabra código. Un código C corrige exactamente t-errores, si este corrige t-errores, pero no corrige $(t + 1)$-errores. Dicho de otra forma, todos los errores de tamaño t son corregidos, pero al menos un error de tamaño $t + 1$ es decodificado incorrectamente.

Ejemplo 2.4.5 Considere el código binario de repetición $C = \{000, 111\}$. Usando la decodificación por distancia mínima tenemos:

- Si 000 es enviada y un error ocurre en la transmisión, entonces las posibles palabras recibidas son 001, 010 y 100. Aquí, primero hay una detección del error, ya que ninguna de estas palabras hace parte del código. En cualquier caso, la palabra recibida, será decodificada a 000, ya que la decodificación por distancia mínima decodifica la palabra recibida co-

mo aquella palabra código tal que su distancia de Hamming sea la menor. Similarmente, se tiene para la palabra código enviada 111, bajo la consideración de que se ha cometido un error en la transmisión.

- Si se envía 000 y ocurren 2 errores en la transmisión, entonces las posibles palabras recibidas son 011, 110 y 101. Aunque el código es capaz de detectar los errores (puesto que su distancia mínima es 3) no los corrige, ya que para cualquier palabra recibida 011, 110 y 101, la decodificación por distancia mínima la decodificará a 111. Por lo tanto, C es un código que corrige exactamente un error. □

A continuación presentamos el resultado que permite determinar el número de errores que es capaz de corregir un código en función de su distancia.

Teorema 2.4.6 Un código C corrige exactamente t-errores si y sólo si $d(C) = 2t+1$ o $d(C) = 2t+2$.

Demostración. Supongamos que $d(C) = 2t+1$ o $d(C) = 2t+2$. Asumamos que durante la transmisión de una palabra $\mathbf{c} \in C$ en la palabra recibida $\mathbf{x}$ se han presentado a lo sumo t errores; es decir $d(\mathbf{x},\mathbf{c}) \leq t$. Si existiera $\mathbf{y} \in C$ con $\mathbf{c} \neq \mathbf{y}$ tal que $d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{c})$. Por la desigualdad triangular tenemos

$$\begin{aligned} d(\mathbf{y},\mathbf{c}) &\leq d(\mathbf{y},\mathbf{x}) + d(\mathbf{x},\mathbf{c}) \\ &\leq t + t = 2t < d(C), \end{aligned}$$

lo cual es una contradicción. Por lo tanto, C corrige t-errores o menos.

Sean $\mathbf{c},\mathbf{c}' \in C$ tales que $d(\mathbf{c},\mathbf{c}') = d(C) = d$. Sin pérdida de generalidad, supongamos que la longitud de C es n y que $\mathbf{c}$ y $\mathbf{c}'$ difieren en las primeras d coordenadas. Si se envía $\mathbf{c}$ y se recibe la palabra $\mathbf{x}$ cuyas coordenadas están distribuidas de la siguiente manera

$$\mathbf{x} = \underbrace{x_1 \cdots\cdots x_{t+1}}_{\substack{\text{coinciden} \\ \text{con } \mathbf{c}'}} \underbrace{x_{t+2} \cdots\cdots x_d}_{\substack{\text{coinciden} \\ \text{con } \mathbf{c}}} \underbrace{x_{d+1} \cdots\cdots x_n}_{\substack{\text{coinciden} \\ \text{con } \mathbf{c} \text{ y } \mathbf{c}'}}$$

tenemos

$$d(\mathbf{x},\mathbf{c}') = d - (t+1) \leq t+1 = d(\mathbf{x},\mathbf{c}).$$

Si $d = 2t + 1$, entonces la decodificación por distancia mínima decodifica $\mathbf{x}$ incorrectamente como $\mathbf{c}'$. Si $d = 2t + 2$, entonces hay empate y por lo tanto error. En cualquier caso, C no corrige $(t + 1)$-errores.
Recíprocamente, para cualquier $\mathbf{c}_1, \mathbf{c}_2 \in C$ no es posible que $d(\mathbf{c}_1, \mathbf{c}_2) = w \leq 2t$, ya que de lo contrario al enviar $\mathbf{c}_1$ podríamos construir una palabra recibida $\mathbf{x}$ tal que $d(\mathbf{x}, \mathbf{c}_1) = t$ y $d(\mathbf{x}, \mathbf{c}_2) = w - t$. En consecuencia, la decodificación por distancia mínima decodificaría $\mathbf{x}$ como $\mathbf{c}_2$ o podría reportar empate. Luego $d(C) > 2t$.
Ahora, si $d(C) \geq 2t + 3 = 2(t + 1) + 1$, entonces por el argumento anterior se tiene que C es un corrector de $(t + 1)$-errores, lo cual contradice la hipótesis. Por lo tanto, $d(C) = 2t + 1$ o $d(C) = 2t + 2$. ■

Del resultado anterior obtenemos el siguiente corolario, cuya prueba se deja como ejercicio para el lector.

Corolario 2.4.7 Sea C un código. Entonces $d(C) = d$ si y sólo si C corrige exactamente $\left\lfloor \frac{d-1}{2} \right\rfloor$-errores y detecta $d - 1$ errores.

Finalizamos esta sección con el siguiente concepto que es útil para entender el anterior resultado.

Definición 2.4.8 Sea $\mathbf{x}$ una palabra en A^n, donde $|A| = q$ y $r \geq 0$. La *esfera de radio r centrada en* $\mathbf{x}$ es el conjunto

$$S_q(\mathbf{x}, r) = \{\mathbf{y} \in A^n : d(\mathbf{x}, \mathbf{y}) \leq r\}.$$

La interpretación geométrica de este resultado se ilustra en la Figura 2.1. Supongamos que la distancia del código es $d(C) = 2t + 1$. Las esferas de radio t centradas en cada palabra código no se intersectan debido a que la distancia entre cualquier par de palabras código es mayor o igual que $2t + 1$. Por medio de la decodificación por distancia mínima, cualquier n-upla $\mathbf{c}$ en una esfera de radio t centrada en la palabra código $\mathbf{c}_1$ se decodifica como $\mathbf{c}_1$.

Si a lo sumo $2t$ errores ocurren durante la trasmisión de una palabra código $\mathbf{c}_i$, la palabra recibida $\mathbf{x}$ no puede considerarse como una palabra código válida $\mathbf{c}_j$, ya que la distancia entre $\mathbf{c}_i$ y $\mathbf{c}_j$ es al menos $2t+1$.

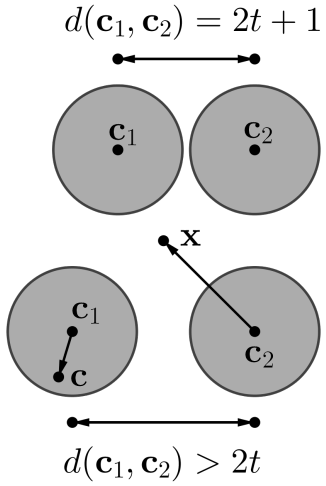


Figura 2.1: Distancia de un código.

2.5. Peso de Hamming

Otro de los conceptos básicos en la teoría de códigos es el de peso de Hamming de una palabra.

Definición 2.5.1 Sea $\mathbf{x}$ una palabra en $\mathbb{F}_q^n$. El *peso de Hamming* de $\mathbf{x}$ (o simplemente peso de $\mathbf{x}$), denotado por $wt(\mathbf{x})$, se define como el número de coordenadas no nulas en $\mathbf{x}$. Es decir

$$wt(\mathbf{x}) = d(\mathbf{x}, \mathbf{0}), \tag{2.3}$$

donde $\mathbf{0}$ es la palabra cero.

Definición 2.5.2 Sean $\mathbf{x} = x_1 x_2 \cdots x_n$, $\mathbf{y} = y_1 y_2 \cdots y_n \in \mathbb{F}_q^n$. La *intersección* de $\mathbf{x}$ y $\mathbf{y}$ se define por

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

El *producto interno* o *producto interior* entre $\mathbf{x}$ y $\mathbf{y}$ se define por

$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

Lema 2.5.3

1. Para todo $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$,

$$d(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x} - \mathbf{y}).$$

En particular, para $q = 2$,

$$d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} + \mathbf{y}).$$

2. Para todo $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$,

$$d(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x}) + wt(\mathbf{y}) - 2wt(\mathbf{x} \cap \mathbf{y}).$$

Demostración.

1. Se deja como ejercicio al lector.
2. Consideremos los siguientes conjuntos

$$\begin{aligned} A &= \{i : 1 \leq i \leq n, x_i \neq 0\}, \\ B &= \{i : 1 \leq i \leq n, y_i \neq 0\}, \\ A^* &= \{i : 1 \leq i \leq n, x_i \neq 0 \ \wedge \ y_i = 0\}, \\ B^* &= \{i : 1 \leq i \leq n, x_i = 0 \ \wedge \ y_i \neq 0\}. \end{aligned} \tag{2.4}$$

Es claro que $A \cap B \subset A$, $A \cap B \subset B$, $A^* = A \setminus (A \cap B)$, $B^* = B \setminus (A \cap B)$ y que $A^* \cap B^* = \emptyset$. Además, observe que

$$wt(\mathbf{x}) = |A|, \quad wt(\mathbf{y}) = |B| \quad \text{y} \quad wt(\mathbf{x} \cap \mathbf{y}) = |A \cap B|.$$

De esta manera

$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) &= wt(\mathbf{x} + \mathbf{y}) = d(\mathbf{x} + \mathbf{y}, \mathbf{0}) \\ &= |\{i : 1 \leq i \leq n, x_i + y_i \neq 0\}| \\ &= |A^* \cup B^*| \\ &= |A^*| + |B^*| \\ &= |A \setminus (A \cap B)| + |B \setminus (A \cap B)| \\ &= (|A| - |A \cap B|) + (|B| - |A \cap B|) \\ &= wt(\mathbf{x}) + wt(\mathbf{y}) - 2wt(\mathbf{x} \cap \mathbf{y}). \end{aligned}$$

■

Otras propiedades del concepto de peso se resumen en el siguiente lema.

Lema 2.5.4

1. Si $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$, entonces
$$wt(\mathbf{x}) - wt(\mathbf{y}) \leq wt(\mathbf{x}+\mathbf{y}) \leq wt(\mathbf{x}) + wt(\mathbf{y}).$$
2. Si $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, entonces $wt(\mathbf{x} \cap \mathbf{y}) \equiv \mathbf{x} \cdot \mathbf{y} \,(\text{mód}\, 2)$.
3. Si $\mathbf{x} \in \mathbb{F}_3^n$, entonces $wt(\mathbf{x}) \equiv \mathbf{x} \cdot \mathbf{x} \,(\text{mód}\, 3)$.

Demostración.

1. Para probar la primera desigualdad consideremos los conjuntos A, B, A^* y B^* definidos en (2.4). Es claro que A^*, B^* y $A \cap B$ son disjuntos dos a dos. Además, observe que
$$\{i : 1 \leq i \leq n, x_i + y_i \neq 0\} \subset A^* \cup B^* \cup (A \cap B).$$
De esta manera
$$\begin{aligned} wt(\mathbf{x}+\mathbf{y}) &= d(\mathbf{x}+\mathbf{y}, \mathbf{0}) \\ &= |\{i : 1 \leq i \leq n, x_i + y_i \neq 0\}| \\ &\leq |A^* \cup B^* \cup (A \cap B)| \\ &= |A^*| + |B^*| + |A \cap B| \\ &= |A \setminus (A \cap B)| + |B \setminus (A \cap B)| + |A \cap B| \\ &= (|A| - |A \cap B|) + (|B| - |A \cap B|) + |A \cap B| \\ &= wt(\mathbf{x}) + wt(\mathbf{y}) - wt(\mathbf{x} \cap \mathbf{y}) \\ &\leq wt(\mathbf{x}) + wt(\mathbf{y}). \end{aligned}$$
Ahora, para probar la segunda desigualdad considere $\mathbf{v} = \mathbf{x} + \mathbf{y}$ y $\mathbf{w} = -\mathbf{y}$. Entonces, de la primera desigualdad obtenemos que
$$wt(\mathbf{x}) = wt(\mathbf{v}+\mathbf{w}) \leq wt(\mathbf{v}) + wt(\mathbf{w}) = wt(\mathbf{x}+\mathbf{y}) + wt(-\mathbf{y}).$$
Finalmente, el resultado se sigue de la anterior desigualdad teniendo en cuenta que $wt(-\mathbf{y}) = wt(\mathbf{y})$.
2. Se deja como ejercicio al lector.
3. Por definición
$$wt(\mathbf{x}) = d(\mathbf{x}, \mathbf{0}) = |\{i : 1 \leq i \leq n, x_i \neq 0\}| = \sum_{x_i \neq 0} 1.$$

Por otro lado, módulo 3 tenemos que

$$x_i^2 \equiv \begin{cases} 1\,(\text{mód}3), & \text{si } x_i \neq 0, \\ 0\,(\text{mód}3), & \text{si } x_i = 0. \end{cases}$$

Por lo tanto,

$$wt(\mathbf{x}) \equiv \sum_{x_i \neq 0} 1\,(\text{mód}3) \equiv \sum_{i=1}^{n} x_i^2\,(\text{mód}3) \equiv \mathbf{x}\cdot\mathbf{x}\,(\text{mód}3). \quad \blacksquare$$

Definición 2.5.5 Sea C un código. El *peso mínimo* (de Hamming) de C, denotado por $wt(C)$ es el mínimo de los pesos de todas las palabras no nulas de C. Es decir,

$$wt(C) = \text{mín}\{wt(\mathbf{c}) : \mathbf{c} \in C,\ \mathbf{c} \neq \mathbf{0}\}.$$

Ejemplo 2.5.6 Para los códigos

$$C_1 = \{000, 101, 010, 111\},$$
$$C_2 = \{000000, 000111, 001110, 011100\},$$

tenemos que $wt(C_1) = 1 = d(C_1)$; mientras que $wt(C_2) = 3 \neq d(C_2) = 2$. Note que la distancia mínima de un código no siempre coincide con el peso del código, como se puede observar en el código C_2. □

Un (n, M, d, w)-código es un código de longitud n, tamaño M, distancia mínima d y peso w.

Definición 2.5.7 Un código C se llama *código de peso constante*, si todas las palabras de C tienen el mismo peso; es decir, $wt(\mathbf{c}) = k$, para algún $k \in \mathbb{N}$ y para toda $\mathbf{c} \in C$.

Una característica importante de un código es conocer la manera en cómo están distribuidos los pesos de Hamming de sus palabras. Una manera apropiada de estudiar esto es introduciendo los siguientes conceptos.

Definición 2.5.8 Sea C un código de longitud n. Para $0 \leq i \leq n$, denotamos con A_i el número de palabras código en C

de peso igual a i. Esto es,

$$A_i = |\{\mathbf{c} \in C : wt(\mathbf{c}) = i\}|.$$

La lista $A_0, A_1, \ldots, A_n$ se denomina la *distribución de pesos de* C y la suma formal

$$W_C(x) = \sum_{i=0}^{n} A_i x^i,$$

se denomina *polinomio enumerador de pesos de* C.
Adicionalmente, el polinomio en dos variables dado por

$$W_C(x,y) = y^n W_C\left(\frac{x}{y}\right) = \sum_{i=0}^{n} A_i x^i y^{n-i},$$

es el *polinomio enumerador de pesos homogéneo de* C.

Si bien la distribución de pesos en general no determina de manera única un código, sí proporciona información práctica y teórica. Sin embargo, calcular la distribución de pesos de un código de gran tamaño es un problema difícil.

Ejemplo 2.5.9 Calcular el polinomio enumerador de pesos para el código de control de paridad

$$C = \left\{ (x_1, x_2, x_3, x_4) \in \mathbb{F}_3^4 : x_4 = \sum_{k=1}^{3} x_k \right\}.$$

Tenemos que

$$\begin{aligned}
A_0 &= |\{0000\}| = 1,\\
A_1 &= |\emptyset| = 0,\\
A_2 &= |\{0012, 0021, 0102, 0120, 0201, 0210, 1002, 1020,\\
&\qquad 1200, 2001, 2010, 2100\}| = 12,\\
A_3 &= |\{0111, 0222, 1011, 1101, 1110, 2022, 2202, 2220\}| = 8,\\
A_4 &= |\{1122, 1212, 1221, 2112, 2121, 2211\}| = 6.
\end{aligned}$$

Por tanto, los polinomios enumerador y enumerador homogéneo de C son:

$$W_C(x) = \sum_{i=0}^{4} A_i x^i = 1 + 12x^2 + 8x^3 + 6x^4,$$

$$W_C(x,y) = \sum_{i=0}^{4} A_i x^i y^{4-i} = y^4 + 12x^2y^2 + 8x^3y + 6x^4. \qquad \square$$

2.6. Decodificación por máxima verosimilitud

En esta sección presentamos una definición más formal del concepto de canal de comunicación, la cual nos permitirá estudiar el proceso de decodificación desde un enfoque probabilístico.

Definición 2.6.1 Un *canal discreto sin memoria* consta de un alfabeto de entrada $A = \{a_1, a_2, \ldots, a_q\}$, un alfabeto de salida $B = \{b_1, b_2, \ldots, b_t\}$ con $A \subseteq B$ y un conjunto de *probabilidades del canal* o *probabilidades de transición* $P(b_j \text{ recibido} \mid a_i \text{ enviado})$, que satisfacen:

$$\sum_{j=1}^{t} P(b_j \text{ recibido} \mid a_i \text{ enviado}) = 1,$$

para todo $i = 1, 2, ..., t$.
Aquí $P(b_j \text{ recibido} \mid a_i \text{ enviado})$ es la probabilidad condicional de que b_j sea recibido dado que a_i es enviado.
Además, si $\mathbf{c} = c_1 c_2 \cdots c_n$ y $\mathbf{d} = d_1 d_2 \cdots d_n$ son palabras de longitud n sobre A y B respectivamente, entonces la probabilidad

$$P(\mathbf{d} \text{ recibido} \mid \mathbf{c} \text{ enviado}) = \prod_{i=1}^{n} P(d_i \text{ recibido} \mid c_i \text{ enviado}).$$

La razón por la cual en la Definición 2.6.1 aparece el término sin memoria, es debido a que el resultado de cualquier transmisión de una coordenada es independiente de los resultados de las transmisiones anteriores.

Definición 2.6.2 Un *canal simétrico q-ario* es un canal sin memoria con alfabetos de entrada y salida iguales, de tamaño q y que cumple las siguientes condiciones.

1. Cada símbolo transmitido tiene la misma probabilidad $p < 1/2$ de recibirse en error.
2. Si un símbolo es recibido en error, entonces cada uno de los $q-1$ posibles errores es igualmente probable.

Observación 2.6.3 Dado un canal simétrico q-ario y $\mathbf{x} = x_1x_2\cdots x_n$, $\mathbf{c} = c_1c_2\cdots c_n$ palabras de longitud n, por la segunda parte de la definición anterior se tiene

$$P(x_i \text{ recibido} \mid c_i \text{ enviado}) = \begin{cases} \dfrac{p}{q-1}, & \text{si } x_i \neq c_i, \\ 1-p, & \text{si } x_i = c_i. \end{cases}$$

Por la segunda parte de la Definición 2.6.1, para un canal simétrico q-ario tenemos que

$$\begin{aligned} P(\mathbf{x} \text{ recibido} \mid \mathbf{c} \text{ enviado}) &= \prod_{i=1}^{n} P(x_i \text{ recibido} \mid c_i \text{ enviado}) \\ &= \left(\frac{p}{q-1}\right)^{t} (1-p)^{n-t}, \end{aligned}$$

donde t es el número de coordenadas en las que $\mathbf{x}$ y $\mathbf{c}$ difieren.
Uno de los canales simétricos sin memoria más conocidos es el *canal simétrico binario* (Binary Symmetric Channel) BSC, el cual tiene como alfabeto a $\{0,1\}$ y probabilidades de transición

$$\begin{aligned} P(1 \mid 0) &= P(0 \mid 1) = p, \\ P(1 \mid 1) &= P(0 \mid 0) = 1-p. \end{aligned}$$

Por lo tanto, la probabilidad de recibir una coordenada en error es p, donde $p < 1/2$ (Figura 2.2).

Asumiendo que palabras de un código C son enviadas a través de un canal simétrico q-ario y que la menor cantidad de errores se han producido en la transmisión, podemos encontrar la pa-

labra probablemente enviada calculando las probabilidades del canal.

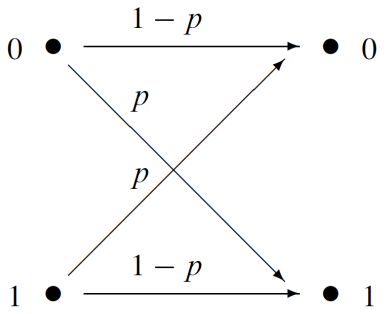


Figura 2.2: Probabilidad para el BSC.

Ejemplo 2.6.4 Supongamos que palabras del código $C = \{000, 111\}$ son enviadas sobre un BSC con probabilidad de error $p = 0.05$. Supongamos que la palabra 110 es recibida y deseamos determinar cual fue la palabra código probablemente enviada. Para este fin se calculan las siguientes probabilidades de transición

$$\begin{aligned} P(110 \mid 000) &= P(1 \mid 0)^2 \cdot P(0 \mid 0) \\ &= (0.05)^2 \cdot 0.95 \\ &= 0.002375, \\ P(110 \mid 111) &= P(1 \mid 1)^2 \cdot P(0 \mid 1) \\ &= (0.95)^2 \cdot 0.05 \\ &= 0.045125. \end{aligned}$$

Como la segunda probabilidad es mayor que la primera, se concluye que 111 es la palabra código probablemente enviada. □

Esta forma de decodificación recibe el nombre de *decodificación por máxima verosimilitud* (Maximum Likelihood Decoding) MLD. Es decir, si una palabra $\mathbf{x}$ es recibida, la decodificación por máxima verosimilitud concluirá que $\mathbf{c}_x \in C$ es la palabra probablemente enviada si $\mathbf{c}_x$ maximiza las probabilidades del canal; i.e.,

$$P(\mathbf{x}\text{recibido} \mid \mathbf{c}_x \text{ enviado}) = \max_{\mathbf{c} \in C} P(\mathbf{x} \text{ recibido } \mid \mathbf{c} \text{ enviado}).$$

Existen dos tipos de MLD: la *decodificación por máxima verosimilitud completa* CMLD y la *decodificación por máxima verosimilitud*

incompleta IMLD. La regla completa elige una palabra arbitraria entre dos o más palabras con la misma probabilidad máxima con la palabra recibida mientras que la regla incompleta solicita la retransmisión del mensaje enviado.

El siguiente teorema establece la relación entre las dos reglas de decodificación tratadas anteriormente en un canal simétrico binario con probabilidad de error $p < 1/2$.

Teorema 2.6.5 Para un canal simétrico binario con probabilidad de error $p < 1/2$, la decodificación por distancia mínima es equivalente a la decodificación por máxima verosimilitud.

Demostración. Sea C un (n, M)-código y $\mathbf{x}$ la palabra recibida durante la transmisión de un elemento de C. Para todo $\mathbf{c} \in C$ y todo $0 \leq i \leq n$,

$$d(\mathbf{x}, \mathbf{c}) = i \Leftrightarrow P(\mathbf{x}\,\text{recibido} \mid \mathbf{c}\,\text{enviado}) = p^i(1-p)^{n-i}.$$

Como $p < 1/2$ entonces

$$p^0(1-p)^n > p^1(1-p)^{n-1} > p^2(1-p)^{n-2} > \cdots > p^n(1-p)^0.$$

Por definición, la decodificación por máxima verosimilitud decodifica $\mathbf{x}$ a $\mathbf{c} \in C$ si $P(\mathbf{x}\,\text{recibido} \mid \mathbf{c}\,\text{enviado})$ es máxima, i.e., tal que $d(\mathbf{x}, \mathbf{c})$ sea mínima (o solicita retransmisión si la decodificación incompleta es usada y $\mathbf{c}$ no es única). ■

Bajo las hipótesis del teorema anterior, la decodificación por distancia mínima es de mayor utilidad en el sentido en que los cálculos que se realizan son más sencillos que los cálculos hechos con la decodificación por máxima verosimilitud.

2.7. Equivalencia de códigos

Un (n, M)-código q-ario permite codificar M mensajes fuente. Sin embargo, ciertos tipos de códigos con los mismos parámetros facilitan este proceso de codificación. Para conocer más al respecto, introducimos algunos conceptos básicos.

Definición 2.7.1 Un (n, q^k)-código q-ario se llama *sistemático* si existen k posiciones $i_1, i_2, \ldots, i_k$, tales que al restringir todas las palabras código en estas posiciones, se obtienen todas las q^k posibles palabras q-arias de longitud k. El conjunto $\{i_1, i_2, \ldots, i_k\}$ se denomina un *conjunto de información* y los símbolos de las palabras código en estas posiciones se llaman *símbolos de información*.

Ejemplo 2.7.2 El código binario

$$C = \{0000, 0110, 1001, 1010\},$$

es sistemático sobre la primera y la tercera posición. Por lo tanto, si la fuente es $\{00, 01, 10, 11\}$, se puede codificar como sigue

$$00 \longrightarrow \underline{0}0\underline{0}0,\ 01 \longrightarrow \underline{0}1\underline{1}0,\ 10 \longrightarrow \underline{1}0\underline{0}1,\ 11 \longrightarrow \underline{1}0\underline{1}0. \qquad \square$$

El tipo de codificación que se muestra en el Ejemplo 2.7.2 se denomina *codificación sistemática*. De manera similar, el proceso de codificación sistemática facilita el proceso de decodificación de los mensajes recibidos, ya que a partir de ellos se pueden leer directamente los mensajes fuente en el caso en que no hayan ocurrido errores en la transmisión.

Definición 2.7.3 Dos (n, M)-códigos q-arios C_1 y C_2 son *equivalentes*, lo cual se denota con $C_1 \simeq C_2$, si existe una permutación σ de las n coordenadas y permutaciones $\pi_1, \pi_2, \ldots, \pi_n$ del alfabeto tales que

$$c_1 c_2 \cdots c_n \in C_1 \text{ si y sólo si } \pi_1(c_{\sigma(1)})\pi_2(c_{\sigma(2)}) \cdots \pi_n(c_{\sigma(n)}) \in C_2.$$

En otras palabras, dos códigos son equivalentes si uno puede ser transformado en el otro por permutación de las coordenadas (mediante σ) y por permutación de los símbolos del código en cada coordenada de cada palabra código (mediante $\pi_1, \pi_2, \ldots, \pi_n$).

Ejemplo 2.7.4 Los códigos binarios $C = \{00100, 00011, 11111, 11000\}$ y $C' = \{00000, 01101, 11011, 10110\}$ son equivalentes. En efecto, al considerar $\sigma = (2,4)$ y

$\pi = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$, donde π_1, π_2, π_4 y π_5 son la permutación identidad y $\pi_3 = (0,1)$, se tiene que

$$C = \begin{Bmatrix} 00100 \\ 00011 \\ 11111 \\ 11000 \end{Bmatrix} \xrightarrow{\sigma} \begin{Bmatrix} 00100 \\ 01001 \\ 11111 \\ 10010 \end{Bmatrix} \xrightarrow{\pi} \begin{Bmatrix} 00000 \\ 01101 \\ 11011 \\ 10110 \end{Bmatrix} = C'. \qquad \square$$

Existe otra definición de equivalencia, pero primero veamos una definición preliminar.

Definición 2.7.5 Sea σ una permutación de tamaño n. Para $i = 1, \ldots, n$, sea $\pi_i : \mathbb{F}_q \longrightarrow \mathbb{F}_q$ la multiplicación por un escalar no cero α_i en $\mathbb{F}_q$; es decir, $\pi_i(s) = \alpha_i s$. Entonces la función $\mu : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$ definida por

$$\mu(c_1 c_2 \cdots c_n) = \pi_1(c_{\sigma(1)})\pi_2(c_{\sigma(2)}) \cdots \pi_n(c_{\sigma(n)}),$$

se llama *transformación monomial de grado n*. En palabras, una transformación monomial actúa primero sobre las n coordenadas con una permutación de éstas, seguida por la multiplicación de cada coordenada por un escalar no cero.

Cabe notar que en el caso $q = 2$ no hay operaciones de multiplicación no triviales. Por lo tanto, para este caso sólo se consideran las permutaciones de las coordenadas.

Definición 2.7.6 Dos (n, M)-códigos C_1 y C_2 sobre $\mathbb{F}_q$ son *múltiplo escalar equivalentes* si existe una transformación monomial μ de grado n para la cual $\mu(C_1) = C_2$, donde $\mu(C_1) = \{\mu(\mathbf{c}) : \mathbf{c} \in C_1\}$. Entonces, C_1 y C_2 son múltiplo escalar equivalentes si estos son equivalentes en el sentido de la definición anterior, donde cada permutación π_i es una multiplicación por un escalar no cero.

Ejemplo 2.7.7 Consideremos el código ternario $C = \{000, 111, 022, 021\}$. Permutando la primera y la segunda posición, seguida de la multiplicación de la tercera posición por 2 se obtiene el código $C' = \{000, 112, 201, 202\}$, el cual es equivalente a C. $\square$

Observación 2.7.8 La transformación μ de la Definición 2.7.5 admite una representación matricial de la siguiente manera. Asignando a la permutación σ una matriz de permutación de orden n y a la multiplicación por el escalar no cero una matriz diagonal D de orden n. El producto $DP = M$, resulta en una matriz cuadrada de orden n, la cual tiene exactamente una entrada no cero en cada fila y en cada columna. Esta matriz recibe el nombre de *matriz monomial* y por construcción realiza la misma labor que μ.

Para el Ejemplo 2.7.7 se tiene que la matriz monomial está dada por

$$DP = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} = M.$$

Observación 2.7.9 Dos códigos que son múltiplo escalar equivalentes son también equivalentes. En efecto, como $\mathbb{F}_q$ es un campo, entonces si C_1 y C_2 son códigos múltiplo escalar equivalentes sobre $\mathbb{F}_q$ entonces son equivalentes ya que la aplicación π_i definida en 2.7.5 es una biyección de $\mathbb{F}_q$. Sin embargo, el recíproco de esta afirmación no es verdadera, como lo muestra el siguiente ejemplo.

Ejemplo 2.7.10 El código $C = \{04213, 12034, 23401, 30142, 41320\}$ sobre $\mathbb{F}_5$ es equivalente al código de repetición $C' = \{00000, 11111, 22222, 33333, 44444\}$ sobre $\mathbb{F}_5$. En efecto, aplicando las permutaciones del alfabeto $\pi_2 = (40321)$, $\pi_3 = (20134)$, $\pi_4 = (10243)$ y $\pi_5 = (30412)$ se tiene

$$C = \begin{Bmatrix} 04213 \\ 12034 \\ 23401 \\ 30142 \\ 41320 \end{Bmatrix} \xrightarrow{\pi_2} \begin{Bmatrix} 00213 \\ 11034 \\ 22401 \\ 33142 \\ 44320 \end{Bmatrix} \xrightarrow{\pi_3} \begin{Bmatrix} 00013 \\ 11134 \\ 22201 \\ 33342 \\ 44420 \end{Bmatrix} \xrightarrow{\pi_4}$$
$$\begin{Bmatrix} 00003 \\ 11114 \\ 22221 \\ 33332 \\ 44440 \end{Bmatrix} \xrightarrow{\pi_5} \begin{Bmatrix} 00000 \\ 11111 \\ 22222 \\ 33333 \\ 44444 \end{Bmatrix} = C'.$$

Sin embargo, C y C' no son múltiplo escalar equivalentes, ya que a cada una de las permutaciones no se le puede asignar la multiplicación por un escalar no nulo ($\pi_i(0) \neq 0$). □

Lema 2.7.11 Si $0 \in A$, entonces cualquier (n, M)-código sobre el alfabeto A es equivalente (pero no necesariamente múltiplo escalar equivalente) a un código C' que contiene la palabra código cero.

Demostración. En efecto, si $\mathbf{0} \in C$, entonces permutando todas las coordenadas de las palabras de C se obtiene un código equivalente C' con $\mathbf{0} \in C'$. Si $\mathbf{0} \notin C$ sea $\mathbf{x} \in C$ con k coordenadas no nulas. Considerando la permutación π del alfabeto que conserve los ceros y transforme las restantes k coordenadas no nulas en cero, se obtiene que $\pi(\mathbf{x}) = \mathbf{0}$. Aplicando π al resto de palabras de C se obtiene el código C', el cual es equivalente a C y tal que $\mathbf{0} \in C'$. ■

Teorema 2.7.12 Si C_1 y C_2 son equivalentes, entonces $d(C_1) = d(C_2)$. Además, bajo la decodificación por distancia mínima y asumiendo un canal simétrico q-ario con $p < 1/2$, la probabilidad de decodificar un error es la misma para códigos equivalentes.

Demostración. Teniendo en cuenta que permutar coordenadas no afecta la distancia del código, podemos suponer que σ es la permutación identidad. Sean palabras $\mathbf{x}, \mathbf{y} \in C$ con $d(\mathbf{x}, \mathbf{y}) = d$. Supongamos que las coordenadas en las que $\mathbf{x}$ e $\mathbf{y}$ difieren son $i_1, \ldots i_d$. Probemos que la imagen de $\mathbf{x}$ e $\mathbf{y}$ denotadas por $\mathbf{x}', \mathbf{y}'$ respectivamente, difieren en exactamente esas posiciones. Para $1 \leq i \leq n$, sean π_i las permutaciones del alfabeto. Para cada i_j con $j \in \{1, \ldots, d\}$ se tiene que $\pi_i(x_{i_j}) \neq \pi_i(y_{i_j})$ y $\pi_i(x_{i_j}) = \pi_i(y_{i_j})$ si $j \notin \{1, \ldots, d\}$. Luego $d(\mathbf{x}', \mathbf{y}') = d$. Como $\mathbf{x}, \mathbf{y}$ se eligieron arbitrariamente, entonces $d(C_1) = d(C_2)$. La segunda proposición es directa dado que la distancia de dos códigos equivalentes es igual y por lo tanto corrigen la misma cantidad de errores. ■

Para cada código sobre $\mathbb{F}_q$ existe el grupo de automorfismos del código. Este grupo puede ser útil para estudiar la estructura del código y para el proceso de decodificación.

Definición 2.7.13 Sea C un (n, M)-código sobre $\mathbb{F}_q$. El *grupo de automorfismos de* C, denotado $\mathrm{Aut}(C)$ es el conjunto de todas las transformaciones monomiales μ de grado n tales que $\mu(C) \subseteq C$.

El nombre que recibe el conjunto $\mathrm{Aut}(C)$ no es casualidad, de hecho se puede demostrar el siguiente resultado, ver [36, Thm. 4.3.13].

Teorema 2.7.14 El conjunto $\mathrm{Aut}(C)$ es un grupo con la operación de composición de funciones.

Recordemos que para el caso $q = 2$, las transformaciones monomiales son sólo permutaciones π de las coordenadas, es decir

$$\pi(\mathbf{c}) = \pi(c_1 c_2 \cdots c_n) = c_{\pi(1)} c_{\pi(2)} \cdots c_{\pi(n)}.$$

En este caso, $\mathrm{Aut}(C)$ es un subgrupo del grupo simétrico S_n de todas las $n!$ permutaciones de n símbolos.

Ejemplo 2.7.15 El grupo de automorfismos del $(4,4)$-código binario

$$C = \{0000, 1100, 0011, 1111\}$$

es el siguiente subgrupo de S_4

$$\mathrm{Aut}(C) = \{(1), (12), (34), (12)(34), (13)(24), (14)(23), (1324), (1423)\}.$$

□

2.8. Códigos perfectos

Una familia interesante de códigos estrechamente relacionados con la distancia mínima son los códigos perfectos. Veamos primero una definición previa.

Definición 2.8.1 El *volumen* $V_q(n, r)$ de la esfera $S_q(\mathbf{x}, r)$ es el número de elementos de $S_q(\mathbf{x}, r)$.

El volumen de una esfera es independiente del centro, como demostramos a continuación.

Lema 2.8.2 El volumen $V_q(n,r)$ de la esfera $S_q(\mathbf{x},r)$ está dado por

$$V_q(n,r) = \sum_{k=0}^{r} \binom{n}{k} (q-1)^k. \tag{2.5}$$

Demostración. Sea $\mathbf{x}$ un elemento fijo en A^n. Determinemos el número de elementos $\mathbf{y} \in A^n$ que tienen distancia exactamente k con $\mathbf{x}$, donde $0 \leq k \leq n$. Las k posiciones en las cuales $\mathbf{x}$ y $\mathbf{y}$ difieren pueden ser obtenidas de $\binom{n}{k}$ maneras diferentes y en cada una de éstas posiciones las coordenadas de $\mathbf{y}$ pueden elegirse de $q-1$ maneras distintas de la correspondiente coordenada en $\mathbf{x}$. Por lo tanto, el número de vectores $\mathbf{y}$ tales que $d(\mathbf{x},\mathbf{y}) = k$ es $\binom{n}{k}(q-1)^k$ y de ahí que el número total de elementos en la esfera esté dado por la igualdad (2.5). ■

Ejemplo 2.8.3 Considere $\mathbf{x} = 111 \in \mathbb{F}_3^3$ y $r = 2$, entonces la esfera de radio 2 centrada en $\mathbf{x}$ es

$$\begin{aligned} S_3(\mathbf{x},2) = \{&111,100,010,001,110,101,011,211,121,112,221,212,\\ &122,210,201,120,102,021,012\}. \end{aligned}$$

y el volumen de $S_3(\mathbf{x},2)$ es

$$V_3(3,2) = \binom{3}{0}(3-1)^0 + \binom{3}{1}(3-1)^1 + \binom{3}{2}(3-1)^2 = 19. \quad \square$$

Definición 2.8.4 Sea $C \subseteq A^n$ un código. El *radio de empaquetamiento* de C es el mayor entero r para el cual las esferas $S_q(\mathbf{x},r)$ centradas en cada palabra código $\mathbf{x}$ son disjuntas. El *radio de cubrimiento* de C es el menor entero s para el cual las esferas $S_q(\mathbf{c},s)$ sobre cada palabra código c cubren a A^n; es decir, tal que la unión de las esferas $S_q(\mathbf{c},s)$ es A^n. El radio de empaquetamiento y el radio de cubrimiento de C se denotan por $pr(C)$ y $cr(C)$, respectivamente.

Ejemplo 2.8.5 Considere el código binario $C = \{1101, 0011\}$. Entonces $pr(C) = 1$ dado que las esferas

$$S_2(1101, 1) = \{1101, 0101, 1001, 1111, 1100\} \text{ y}$$
$$S_2(0011, 1) = \{0011, 1011, 0111, 0001, 0010\},$$

son disjuntas y $0001 \in S_2(0011, 2) \cap S_2(1101, 2)$. Adicionalmente, se puede verificar que $cr(C) = 2$. □

El siguiente resultado nos permite establecer la capacidad de corrección
(t-errores) de un código en términos de las esferas de radio fijo (t) sobre cada palabra código.

Teorema 2.8.6 Considerando que los empates son reportados como errores, un código C corrige t-errores si y sólo si las esferas $S_q(\mathbf{x}, t)$ centradas en las palabras código son disjuntas.

Demostración. Supongamos que C es un corrector de t-errores y $\mathbf{x}$ es una palabra recibida con t errores como máximo. Por contradicción, supongamos que $\mathbf{x} \in S_q(\mathbf{c}_1, t) \cap S_q(\mathbf{c}_2, t)$, con $\mathbf{c}_1, \mathbf{c}_2 \in C$ y $\mathbf{c}_1 \neq \mathbf{c}_2$. Sin pérdida de generalidad, asumamos que se envía $\mathbf{c}_2$ y $d(\mathbf{x}, \mathbf{c}_1) < d(\mathbf{x}, \mathbf{c}_2)$. Luego $\mathbf{x}$ se decodifica a $\mathbf{c}_1$, lo cual contradice el hecho de que C corrige t-errores. Incluso, si $d(\mathbf{x}, c_1) = d(\mathbf{x}, c_2)$, dado que los empates fueron asumidos como errores.

Recíprocamente, supongamos que $\mathbf{x}$ es una palabra recibida y que ocurrieron como máximo t errores en la transmisión. Entonces $\mathbf{x}$ pertenece a una única esfera $S_q(\mathbf{c}, t)$, para alguna $\mathbf{c} \in C$ y así la decodificación por distancia mínima decodificará $\mathbf{x}$ a $\mathbf{c}$, y por lo tanto C corrige t-errores. ■

El siguiente resultado es consecuencia inmediata del teorema anterior.

Corolario 2.8.7 Considerando que los empates son siempre reportados como errores, un código C corrige exactamente t-errores si y sólo si $pr(C) = t$.

Demostración. Por contradicción, supongamos que $pr(C) > t$. Luego las esferas $S_q(\mathbf{c}, pr(C))$ sobre cada $\mathbf{c} \in C$ son disjuntas. Por

el Teorema 2.8.6, C corrige $pr(C)$-errores, lo cual contradice el hecho de que C corrige exactamente t-errores. Luego, $pr(C) = t$.

Recíprocamente, por el Teorema 2.8.6, C corrige t-errores. Supongamos que una palabra $\mathbf{x}$ es recibida y han ocurrido $t+1$-errores en la transmisión. Entonces $\mathbf{x}$ pertenece por lo menos a dos esferas $S_q(\mathbf{c}, t+1)$ y $S_q(\mathbf{c}', t+1)$, con $\mathbf{c}, \mathbf{c}' \in C$, $\mathbf{c} \neq \mathbf{c}'$, ya que $pr(C) = t$. Luego $d(\mathbf{x}, \mathbf{c}) = t+1 = d(\mathbf{x}, \mathbf{c}')$ y como los empates se reportan como errores, entonces la decodificación por distancia mínima no puede decodificar la palabra recibida $\mathbf{x}$. ■

El siguiente concepto es de gran interés en la teoría de códigos.

Definición 2.8.8 Un código $C \subset A^n$ se dice *perfecto* si $pr(C) = cr(C)$. En palabras, un código C es perfecto, si existe un número r para el cual las esferas $S_q(\mathbf{c}, r)$ centradas sobre cada palabra código $\mathbf{c}$ son disjuntas y cubren a A^n, es decir

$$A^n = \bigcup_{\mathbf{c} \in C} S_q(\mathbf{c}, r).$$

Ejemplo 2.8.9 Consideremos el $(7, 16, 3)$-código binario

$$\mathrm{Ham}(2,3) = \left\{ \begin{array}{cccc} 0000000, & 0001101, & 1111111, & 1110010, \\ 1101000, & 1000110, & 0010111, & 0111001, \\ 0110100, & 0100011, & 1001011, & 1011100, \\ 0011010, & 1010001, & 1100101, & 0101110 \end{array} \right\}.$$

Este código corrige exactamente 1-error; es decir, las esferas $S_2(\mathbf{c}, 1)$ centradas en $\mathbf{c} \in \mathrm{Ham}(2,3)$ son disjuntas, así $pr(\mathrm{Ham}(2,3)) = 1$. Además,

$$|\mathrm{Ham}(2,3)| = 16, \quad |S_2(\mathbf{c}, 1)| = 1 + \binom{7}{1} = 8 \quad \text{y} \quad \left|\mathbb{F}_2^7\right| = 2^7 = 128.$$

Como $16 \cdot 8 = 128$, entonces las esferas $S_2(\mathbf{c}, 1)$ cubren a $\mathbb{F}_2^7$. Por lo tanto, $\mathrm{Ham}(2,3)$ es perfecto.

En realidad, el código $\mathrm{Ham}(2,3)$ hace parte de la familia de códigos de Hamming, ver Sección 5.5.1. □

Teorema 2.8.10 (La condición de empaquetamiento de esferas) Sea C un (n, M, d)-código q-ario. Entonces C es un código perfecto si y sólo si $d = 2t + 1$ y

$$M \cdot V_q(n,t) = q^n; \tag{2.6}$$

es decir,

$$M = q^n \Big/ \sum_{k=0}^{t} \binom{n}{k} (q-1)^k.$$

Demostración. Sean $\mathbf{x}, \mathbf{y} \in C$ tales que $d(C) = d(\mathbf{x}, \mathbf{y})$. Por contradicción, supongamos que $d(\mathbf{x}, \mathbf{y}) = 2t + 2$. Eligiendo $t + 1$ coordenadas en las cuales $\mathbf{x}$ y $\mathbf{y}$ difieren formamos la palabra $\mathbf{x}'$ cuyas $t + 1$ coordenadas citadas anteriormente coinciden con $\mathbf{x}$ y las demás coinciden con $\mathbf{y}$. Entonces $d(\mathbf{x}, \mathbf{x}') = t + 1 = d(\mathbf{x}', \mathbf{y})$. Como C es perfecto, entonces existe $\mathbf{c} \in C$ tal que $\mathbf{x}' \in S_q(\mathbf{c}, t)$. Luego $d(\mathbf{x}', c) \leq t$. Por la desigualdad triangular se tiene

$$\begin{aligned} d(\mathbf{x}, \mathbf{c}) &\leq d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}', \mathbf{c}) \\ &\leq t + 1 + t \leq 2t + 1. \end{aligned}$$

Pero $d(\mathbf{x}, \mathbf{c}) \geq 2t + 2$ ya que $\mathbf{x}, \mathbf{c} \in C$ y así $2t + 2 \leq 2t + 1$, lo cual es una contradicción. Luego $d(C) = 2t + 1$. Por otra parte, como $cr(C) = pr(C) = t$, entonces el conjunto de las M esferas de radio t sobre las palabras en C forman una partición de las q^n cadenas de longitud n y por lo tanto $M \cdot V_q(n,t) = q^n$.

Recíprocamente, si $d(C) = 2t + 1$ entonces C corrige exactamente t-errores y por el Corolario 2.8.7 se tiene $pr(C) = t$. Luego las esferas de radio t sobre cada palabra son disjuntas y por la condición de empaquetamiento de esferas, éstas cubren a A^n. Luego $pr(C) = cr(C) = t$. ■

J. H. van Lint en [45] afirma que las únicas soluciones para la condición de empaquetamiento de esferas para $n \leq 1000$, $d \leq 2001$, $(t \leq 1000)$, $q \leq 100$, son:

1. $(n, M, d) = (n, q^n, 1)$.
2. $(n, M, d) = (n, 1, 2n + 1)$.
3. $(n, M, d) = (2m + 1, 2, 2m + 1)$.

4. $(n, M, d) = (\frac{q^r-1}{q-1}, q^{n-r}, 3)$, $r \geq 2$.
5. $(n, M, d) = (23, 2^{11}, 7)$.
6. $(n, M, d) = (90, 2^{78}, 5)$.
7. $(n, M, d) = (11, 3^6, 5)$.

La primera solución corresponde a A^n, donde $|A| = q$. La segunda, al código de una sola palabra cuya distancia mínima es indefinida y se adopta convenir $d = 2n + 1$. La tercera solución corresponde al código binario de repetición. Éstos son conocidos como los *códigos perfectos triviales*.

Finalizamos esta sección con la siguiente definición.

Definición 2.8.11 Un código C se dice *cuasi-perfecto* si $cr(C) = pr(C) + 1$. En palabras, un código $C \subset A^n$ es quasi-perfecto si existe un número r para el cual las esferas $S_q(\mathbf{c}, r)$ de radio r son disjuntas y las esferas $S_q(\mathbf{c}, r+1)$ de radio $r+1$ cubren a A^n.

Ejemplo 2.8.12 Sea $C = \{0000, 1111\}$ sobre $\mathbb{F}_2$. Dado que $d(C) = 4$, entonces C no es perfecto y $pr(C) = 1$. Por otro lado, como

$$\begin{aligned} S_2(0000, 2) = \{&0000, 0001, 0010, 0100, 1000, 0011, 0101, 1001, 0110, \\ &1010, 1100\}, \\ S_2(1111, 2) = \{&1111, 1110, 1101, 1011, 0111, 1100, 1010, 0110, 1001, \\ &0101, 0011\}, \end{aligned}$$

entonces $S_2(0000, 2) \bigcup S_2(1111, 2) = \mathbb{F}_2^4$ y así $cr(C) = pr(C) + 1$. Luego C es cuasi-perfecto. □

Algunos códigos de repetición de longitud par también son códigos cuasi-perfectos.

2.9. SageMath

A continuación mostramos la forma de introducir códigos en SageMath usando conjuntos y listas.

```
[In]: C={'0','10','11','100','101','111'} # Este es
        un código de longitud variable
      C # Sus elementos se ingresaron como cadenas

[Out]: {'0', '10', '100', '101', '11', '111'}

[In]: type(C) #Este objeto es de tipo conjunto

[Out]: <class 'set'>

[In]: for x in C:  # Recorremos sus elementos
          print(x)

[Out]: 101
       100
       11
       111
       0
       10

[In]: '11' in C # Podemos preguntar si una palabra
        está o no en el código

[Out]: True

[In]: '00' in C # una palabra que no está en el
        código

[Out]: False

[In]: T={'000','001','011','010','110','100','101',
      '111'} # Este es un código de bloque

[In]: T

[Out]: {'000', '001', '010', '011', '100', '101',
        '110','111'}
```

El ejemplo anterior también lo podemos construir con ayuda de una lista. La diferencia fundamental radica en los métodos que podamos usar para uno u otro tipo de objeto.

```
[In]: T=['000','001','011','010','110','100','101',
      '111']; T
```

```
[Out]: ['000', '001', '011', '010', '110', '100',
        '101', '111']
```

Incluso podemos construir sus elementos como vectores.

```
[In]: T=[vector([0,0,0]),vector([0,0,1]),
      vector([0,1,1]),vector([0,1,0]),
      vector([1,1,0]),vector([1,0,0]),
      vector([1,0,1]),vector([1,1,1])]
```

```
[In]: T
```

```
[Out]: [(0, 0, 0),
        (0, 0, 1),
        (0, 1, 1),
        (0, 1, 0),
        (1, 1, 0),
        (1, 0, 0),
        (1, 0, 1),
        (1, 1, 1)]
```

Códigos sobre $\mathbb{F}_q$

Con las ideas anteriores, presentamos la manera de construir códigos de bloque con vectores en cuerpos finitos.

```
[In]: C1=[vector(GF(2),[0,0,0]),
      vector(GF(2),[0,1,1]), vector(GF(2),[1,1,0]),
      vector(GF(2),[1,1,1])]
```

```
[In]: C1 # un (3,4)-código binario
```

```
[Out]: [(0, 0, 0), (0, 1, 1), (1, 1, 0), (1, 1, 1)]
```

La siguiente operación nos confirma que en en efecto el código `C1` es binario.

```
[In]: C1[1]+C1[1] # Piense por qué
```

```
[Out]: (0, 0, 0)
```

Para construir un código ternario podemos hacer lo siguiente:

```
[In]: C2=[(2,0,1,0),(1,0,2,0),(1,1,2,2),(2,2,1,1),
      (2,1,2,1)] # un (4,5)-código ternario
      C2

[Out]: [(2, 0, 1, 0), (1, 0, 2, 0), (1, 1, 2, 2),
       (2, 2, 1, 1), (2, 1, 2, 1)]
```

Sin embargo, lo siguiente muestra que en realidad 'C2' no es un código ternario

```
[In]: C2[0]+C2[1] # Aquí el símbolo + no es la suma
        vectorial

[Out]: (2, 0, 1, 0, 1, 0, 2, 0)
```

En realidad un código ternario lo podemos construir haciendo:

```
[In]: C3=[vector(GF(3),(2,0,1,0)),
      vector(GF(3),(1,0,2,0)),
      vector(GF(3),(1,1,2,2)),
      vector(GF(3),(2,2,1,1)),
      vector(GF(3),(2,1,2,1))]; # un (4,5)-código
        ternario
      C3

[Out]: [(2, 0, 1, 0), (1, 0, 2, 0), (1, 1, 2, 2),
       (2, 2, 1, 1), (2, 1, 2, 1)]

[In]: C3[0]+C3[1] # Verifique esta cuenta

[Out]: (0, 0, 0, 0)
```

Ahora un código sobre $\mathbb{F}_4$ (cuaternario)

Para esto necesitamos todos los elementos de $\mathbb{F}_4$

```
[In]: F4.<a>=GF(4) # El cuerpo F_4
      F4

[Out]: Finite Field in a of size 2^2
```

```
[In]: a+a # Con esto confirmamos que estamos
        trabajando en F_4
```

```
[Out]: 0
```

```
[In]: a.parent()
```

```
[Out]: Finite Field in a of size 2^2
```

```
[In]: set([a, a^2, a^3, a^4]) #El elemento a en
        realidad es un generador del grupo
        multiplicativo de F_4
```

```
[Out]: {1, a, a + 1}
```

```
[In]: for x in F4:
          print(x)
```

```
[Out]: 0
       a
       a + 1
       1
```

```
[In]: a+a+a
```

```
[Out]: a
```

```
[In]: C4=[vector((1,a,a)),vector((a+1,0,1)),
      vector((1,0,a)),vector((a,a+1,1))]# (3,4)
        -código cuaternario
```

```
[In]: C4
```

```
[Out]: [(1, a, a), (a + 1, 0, 1), (1, 0, a), (a, a +
         1, 1)]
```

Distancia de Hamming

SageMath trae implementada una función para calcular la distancia de Hamming, el peso y la distancia mínima para códigos lineales. Esto significa que para los códigos de bloque debemos construir funciones propias para calcular estos valores. Sin embargo, esta no es una tarea difícil si tenemos claros los conceptos.

```
[In]: def dis(a, b):
          """
          Esta función calcula la distancia entre
          dos palabras de la misma longitud.
          """
          if len(a)==len(b):
              l=len(a)
              d=0
              for i in  [0..l-1]:
                  if a[i]!=b[i]:
                      d=d+1
              return d
          else:
              return('Las palabras deben ser de la
                      misma longitud')
```

En la función anterior usamos la función `len` de SageMath para calcular la longitud de vectores, listas o cadenas. Veamos cómo funciona nuestra función `dis`.

```
[In]: dis('00','10') # Para cadenas
```

```
[Out]: 1
```

```
[In]: dis('00','11')
```

```
[Out]: 2
```

```
[In]: dis('10','10') # dos palabras iguales
```

```
[Out]: 0
```

```
[In]: dis('000','1111') # Aquí cometemos un error
      a propósito.
```

```
[Out]: 'Las palabras deben
       ser de la misma longitud'
```

```
[In]: dis('00',vector(GF(2),(0,0))) # los objetos
        pueden ser de diferentes clases
```

```
[Out]: 2
```

```
[In]: dis('00',[0,0])
```

```
[Out]: 2
```

```
[In]: x=vector(GF(2),(0,0))
      x
```

```
[Out]: (0, 0)
```

```
[In]: y=vector(GF(3),(0,0))
      y
```

```
[Out]: (0, 0)
```

```
[In]: dis(x,y)
```

```
[Out]: 2
```

Aunque los dos vectores anteriores "aparentemente son iguales", en esencia tienen sus entradas en cuerpos finitos diferentes. Esto hace que para SageMath sean diferentes. Aunque en algunas instancias la función `dis` pueda tratar dos objetos "diferentes", debemos tener el cuidado para que siempre los elementos que comparemos sean de la misma clase.

Distancia mínima de un código

```
[In]: def dis_min(C):
          """
          Esta función calcula la distancia mínima
          de un código de bloque C sin repetir
          cálculos y además entrega 2 palabras
          del código cuya distancia es la distancia
          mínima del código dado.
         """
          u=C[0]
          v=C[0]
          n=len(u) # la longitud en el código
          p=n+1
          m=len(C) # el tamaño del código
          for i in [0..m-2]:
              for j in [i+1..m-1]:
```

```
            if dis(C[i], C[j])<p:
                p=dis(C[i], C[j])
                u=C[i]
                v=C[j]
    return p, u, v
```

Una aplicación de nuestra función `dis_min` es la siguiente:

```
[In]: C =['000000','000111','111222']

[In]: dis_min(C)

[Out]: (3, '000000', '000111')
```

Esto significa que la palabras que están a menor distancia en el código son '000000' y '000111'. Aunque debemos ser conscientes de que no son necesariamente las únicas.

```
[In]: C=["0000","0101","0110","1011","1100"]

[In]: dis_min(C)

[Out]: (2, '0000', '0101')

[In]: dis_min(C1)

[Out]: (1, (0, 1, 1), (1, 1, 1))

[In]: dis_min(C2)

[Out]: (2, (2, 0, 1, 0), (1, 0, 2, 0))

[In]: dis_min(C3)

[Out]: (2, (2, 0, 1, 0), (1, 0, 2, 0))

[In]: dis_min(C4)

[Out]: (1, (1, a, a), (1, 0, a))
```

El peso de Hamming

```
[In]: def peso_min(C):
          """
          Esta función calcula el peso mínimo
          del código C y retorna una palabra del
          código que tiene este peso.
          """
          x=C[0]
          w=len(x)
          F=x.base_ring()
          vect_cero=vector(F,{w-1:0})
          for c in C:
              if dis(c,vect_cero)<w and
                                  c!=vect_cero:
                  w=dis(c,vect_cero)
                  x=c
          return([w,x])
```

```
[In]: peso_min(C1)
```

```
[Out]: [2, (0, 1, 1)]
```

```
[In]: peso_min(C3)
```

```
[Out]: [2, (2, 0, 1, 0)]
```

```
[In]: peso_min(C4)
```

```
[Out]: [2, (a + 1, 0, 1)]
```

```
[In]: C1 =[vector(GF(2),(0,0,0)),
      vector(GF(2),(1,0,1)),
      vector(GF(2),(0,1,0)), vector(GF(2),(1,1,1))]
```

```
[In]: C1
```

```
[Out]: [(0, 0, 0), (1, 0, 1), (0, 1, 0), (1, 1, 1)]
```

```
[In]: dis_min(C1)
```

```
[Out]: (1, (0, 0, 0), (0, 1, 0))
```

```
[In]: peso_min(C1)
```

```
[Out]: [1, (0, 1, 0)]
```

```
[In]: C2=[vector(GF(2),(0,0,0,0,0,0)),
      vector(GF(2),(0,0,0,1,1,1)),
      vector(GF(2),(0,0,1,1,1,0)),
      vector(GF(2),(0,1,1,1,0,0))]
```

```
[In]: C2
```

```
[Out]: [(0, 0, 0, 0, 0, 0),
        (0, 0, 0, 1, 1, 1),
        (0, 0, 1, 1, 1, 0),
        (0, 1, 1, 1, 0, 0)]
```

```
[In]: dis_min(C2)
```

```
[Out]: (2, (0, 0, 0, 1, 1, 1), (0, 0, 1, 1, 1, 0))
```

```
[In]: peso_min(C2)
```

```
[Out]: [3, (0, 0, 0, 1, 1, 1)]
```

Los ejemplos que mostramos aquí, pueden parecer pequeños para el lector; puede pensar que es muy fácil hacer los cálculos que hemos presentado con las funciones `dis`, `dis_min` y `peso_min` a mano (incluso algunos hasta mentalmente). Estamos de acuerdo, es más esperamos que los haya hecho (para corroborar los resultados). Sin embargo, que tal si nuestro código es `tan grande` como el siguiente:

```
[In]: C=[]
      longitud_maxima = len(bin(5999)[2:])
      for num in range(2,6001, 2):
              vec= bin(num)[2:].
                  zfill(longitud_maxima)
              C.append(vector([int(digito) for
                digito in vec]))
```

```
[In]: len(C)
```

```
[Out]: 3000
```

```
[In]: dis_min(C)
```

```
[Out]: (1, (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0),
        (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0))
```

```
[In]: peso_min(C)
```

```
[Out]: [1, (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)]
```

En este ejemplo, nuestro código `C` tiene 3000 palabras. Por supuesto, ¡muy pocos se animarían a hacer estas cuentas a mano!

2.10. Ejercicios

2.10.1. Ejercicios teóricos

1. Sea

$$C = \{000000, 101110, 001010, 110111, 100100, 011001, 111101, 010011\}$$

un código binario de longitud 6

a) ¿Cuál es la distancia mínima de C?

b) ¿Cuántos errores puede detectar C?

c) ¿Cuántos errores puede corregir C?

2. Considere el código binario C_2 del Ejemplo 2.1.2 y suponga que se están transmitiendo palabras código mediante un canal simétrico ternario con probabilidad de error $p = 0.4$.

a) Encuentre el conjunto de probabilidades del canal y construya un esquema similar al de la Figura 2.2

b) Use los métodos de decodificación por distancia mínima y decodificación por máxima verosimilitud para decodificar las siguientes palabras recibidas.

- $\mathbf{w}_1 = 1211$.
- $\mathbf{w}_2 = 0010$.
- $\mathbf{w}_3 = 0222$.

3. Para el código binario $C = \{01101, 00011, 10110, 11000\}$, use la decodificación por distancia mínima para decodificar las siguientes palabras recibidas:

   a) 00000, *c*) 10110, *e*) 11011,
   b) 01111, *d*) 10011, *f*) 11111.

4. Suponga que se transmite una palabra de longitud n sobre un canal simétrico binario (BSC) con probabilidad de error p.

   a) ¿Cuál es la probabilidad de que no haya errores en ninguna de las n posiciones?

   b) ¿Cuál es la probabilidad de que haya exactamente un error en total?

   c) ¿Cuál es la probabilidad de que hayan n errores?

   d) En general, ¿cuál es la probabilidad de que hayan k errores en total?

   e) ¿Cuál es la probabilidad de que haya un número par de errores en total?

5. Para el código ternario $C = \{00122, 12201, 20110, 22000\}$, use la decodificación por distancia mínima para decodificar las siguientes palabras recibidas:

   a) 01122, *b*) 10021, *c*) 22022, *d*) 20120.

6. Construya, si es posible, (n, M, d)-códigos binarios con los siguientes parámetros: $(6,2,6)$, $(3,8,1)$, $(4,8,2)$, $(5,3,4)$, $(8,30,3)$. Cuando no sea posible explique por qué.

7. Sea E_n el conjunto de todos los vectores en $\mathbb{F}_2^n$ de peso par. Muestre que E_n es el código que se obtiene al adicionarle una coordenada de paridad al código $\mathbb{F}_2^{n-1}$. Muestre que E_n es un $(n, 2^{n-1}, 2)$-código.

8. Muestre que si existe un (n, M, d)-código binario con d par, entonces existe un (n, M, d)-código binario en el cual todas las palabras tienen peso par.

9. Muestre que el número de $(n, 2)$-códigos binarios que no son equivalentes es n.

10. Muestre que cualquier (n,q,n)-código q-ario es equivalente a un código de repetición.

11. Calcule el grupo de automorfismos del $(3,4)$-código binario

$$C = \{000,011,101,110\}.$$

12. Calcule el grupo de automorfismos del código binario E_n.

13. Muestre que un $(q+1,M,3)$-código q-ario satisface que $M \leq q^{q-1}$.

14. *a*) Muestre que un $(3,M,2)$-código ternario debe tener $M \leq 9$.

    Sugerencia: Suponga que existe un $(3,M,2)$ con $M > 10$ y elimine la misma coordenada en cada palabra del código.

    b) Muestre que existe un $(3,9,2)$-código ternario.

    c) Generalice los resultados de los ítems anteriores para $(3,M,2)$-códigos q-arios, para cualquier entero $q \geq 2$.

15. Considere el código $C = \mathbb{F}_2^n$, el cual no es capaz de detectar ningún error. Suponga que con el fin de mejorar dicho código se realizan las siguientes modificaciones:

    - a cada palabra $\mathbf{w} \in C$ se le añade la misma palabra al final.
    - a cada palabra $\mathbf{w}$ se le añade un símbolo extra de tal manera que la suma de sus coordenadas sea un número par.

    a) ¿Es alguno de ellos mejor que el código inicial en cuanto a la capacidad de detección de errores?

    b) Calcular la tasa de información de dichos códigos y decidir que código es mejor en este sentido.

    c) ¿Es alguno de ellos mejor que el otro en cuanto a la capacidad de detección y/o corrección de errores?

16. Considere la función f producto interior de $\mathbb{F}_q^n \times \mathbb{F}_q^n$ en $\mathbb{F}_q$, definida por $f(\mathbf{x},\mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$. Demuestre que f es bilineal, esto es que satisface las siguientes propiedades:

    a) $f(\mathbf{u}_1 + \mathbf{u}_2, \mathbf{v}) = f(\mathbf{u}_1, \mathbf{v}) + f(\mathbf{u}_2, \mathbf{v})$,

b) $f(\mathbf{u}, \mathbf{v}_1 + \mathbf{v}_2) = f(\mathbf{u}, \mathbf{v}_1) + f(\mathbf{u}, \mathbf{v}_2)$,

c) $f(\alpha\mathbf{u}, \mathbf{v}) = \alpha f(\mathbf{u}, \mathbf{v})$,

d) $f(\mathbf{u}, \alpha\mathbf{v}) = \alpha f(\mathbf{u}, \mathbf{v})$,

para cualquier $\alpha \in \mathbb{F}_q$ y $\mathbf{u}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{v}, \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{F}_q^n$.

17. Demuestre que f es una forma bilineal no degenerada; es decir, demuestre que no existe $\mathbf{0} \neq \mathbf{x} \in \mathbb{F}_q^n$ tal que $f(\mathbf{x}, \mathbf{y}) = 0$ para todo $\mathbf{y} \in \mathbb{F}_q^n$.

18. *a*) En el Teorema 2.2.3 probamos la desigualdad triangular:

$$d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}).$$

Probar que la igualdad se cumple si y sólo si, para todo i se tiene que $x_i = y_i$ o $y_i = z_i$.

b) Probar que para todo $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_2^n$,

$$wt(\mathbf{x} + \mathbf{z}) + wt(\mathbf{y} + \mathbf{z}) + wt(\mathbf{x} + \mathbf{y} + \mathbf{z}) \geq 2wt(\mathbf{x} + \mathbf{y} + \mathbf{x} \cap \mathbf{y}) - 2wt(\mathbf{z}).$$

con igualdad si y sólo si nunca se tiene que x_i y y_i son 0, y z_i es 1.

19. Sean $\mathbf{x}, \mathbf{y}$ palabras de un código binario.

a) Si las palabras $\mathbf{x}$ y $\mathbf{y}$ tienen ambas peso par o ambas peso impar, pruebe que la palabra $\mathbf{x} + \mathbf{y}$ tiene peso par.

b) Si exactamente una de las palabras $\mathbf{x}$ o $\mathbf{y}$ tiene peso par y la otra tiene peso impar, pruebe que $\mathbf{x} + \mathbf{y}$ tiene peso impar.

Sugerencia: La suma entre palabras $\mathbf{x}, \mathbf{y}$ de un código binario se realiza considerándolas como vectores de $\mathbb{F}_2^n$.

20. Calcular el polinomio enumerador de pesos para el código de control de paridad

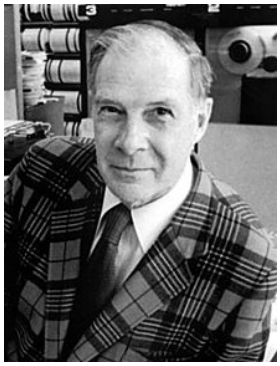
$$C = \left\{ (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n : x_n = \sum_{k=1}^{n-1} x_k \right\}.$$

2.10.2. Prácticas en SageMath

1. Con ayuda de la función `dis_min`, implemente en `SageMath` una función que calcule la capacidad de detección y la capacidad de corrección de un código de bloque. Experimente su función con los ejemplos dados en este capítulo.
2. Implemente una función que realice el proceso de decodificación por distancia mínima incompleta y otra el de decodificación por distancia mínima completa. Asegúrese de que en la completa la palabra escogida sea aleatoria.
3. Construya una función que realice el proceso de decodificación por máxima verosimilitud. Tenga en cuenta que su función debe recibir una probabilidad $0 < p < 1$. Verifique, mediante ejemplos, el Teorema 2.6.5. Encuentre un contraejemplo con el que compruebe que el Teorema 2.6.5 no es cierto si $p > 1/2$.
4. Implemente una función que calcule la intersección de dos vectores binarios.
5. Construya una función `esfera` que reciba un vector binario $\mathbf{x} \in \mathbb{F}_2^n$ y $r \in \mathbb{Z}^+$ y calcule $S_2(\mathbf{x}, r)$. Utilice la función `len` y la función `esfera` para encontrar el volumen de la esfera construida. Utilice la función que construyó para verificar que el código $\mathrm{Ham}(2,3)$ del Ejemplo 2.8.9 es perfecto.

2.11. Biografía

Richard Wesley Hamming **(1915-1998)**



Fue un destacado matemático y científico de la computación estadounidense. En los años 40, su investigación estuvo dedicada al nacimiento de los computadores y la teoría de códigos en los Laboratorios Bell. En 1945 fue reclutado para trabajar en el Laboratorio de los Álamos en el Proyecto Manhattan; dónde participó en el desarrollo de la primera computadora electromecánica, conocida como el Mark I. A partir de 1970 se dedicó a la docencia en diferentes instituciones: Universidad de Illinois, Universidad de Louisville, Universidad de Stanford, City College de Nueva York, Universidad de California, Universidad de Princeton y la Escuela de Posgrado Naval. En 1985, escribió el libro "Methods of mathematics applied to calculus, probability, and statistics" en el cual plasmó sus ideas acerca de la forma en la que creía debería enseñarse matemáticas. Otros de sus libros son: "Numerical Methods for Scientists and Engineers" (1962), "Introduction to applied numerical analysis" (1971), "Coding and information theory" (1980) y "The Art of Doing Science and Engineering : Learning to Learn" (1997). En los laboratorios Bell compartió por algún tiempo oficina con Claude Shannon; mientras este hacía teoría de la información, Hamming investigaba en teoría de códigos. Ante esto posteriormente comentó "Es sospechoso que los dos lo hiciéramos en el mismo lugar y al mismo tiempo, estaba en la atmósfera".

El artículo "Error detecting and error correcting codes", ver [16], fue una de sus contribuciones más destacadas en él estableció el concepto de código de Hamming, un algoritmo de detección y corrección de errores en la transmisión de datos digitales. Este código se utiliza ampliamente en la industria de las comunicaciones y ha sido fundamental para mejorar la fiabilidad de las transmisiones digita-

les. Adicionalmente, introdujo el concepto de distancia de Hamming y de métrica de Hamming, estas ideas se utilizan en otras áreas de las matemáticas.
Richard Hamming fue reconocido con numerosos premios y distinciones a lo largo de su carrera, incluyendo la Medalla Nacional de Ciencia de Estados Unidos en 1991.

Foto tomada de MacTutor.
https://mathshistory.st-andrews.ac.uk/Biographies/Hamming/

3. Códigos lineales

En los capítulos anteriores definimos los conceptos básicos de la teoría de códigos. En el presente capítulo presentaremos una clase importante de códigos, denominados códigos lineales. Como se verá en este capítulo, los códigos lineales por su estructura algebraica presentan ciertas ventajas con respecto a otros códigos, principalmente en los procesos de codificación y decodificación.

3.1. Definición y ejemplos

Definición 3.1.1 Un *código lineal* L de longitud n sobre $\mathbb{F}_q$ es un subespacio vectorial de $\mathbb{F}_q^n$.

La dimensión de un código lineal L se define como su dimensión como espacio vectorial sobre $\mathbb{F}_q$ y se denota $\dim(L)$. De esta manera, si L tiene dimensión k sobre $\mathbb{F}_q$, se dice que L es un $[n,k]$-código lineal. Además, si L tiene distancia mínima d, L se denomina un $[n,k,d]$-código lineal.

Ejemplo 3.1.2

- Los códigos $L_1 = \mathbb{F}_q^n$ y $L_2 = \{\mathbf{0}\}$ son lineales. El código L_1 se utiliza principalmente como conjunto de información.
- El código de repetición $L_3 = \{(\lambda,\ldots,\lambda) : \lambda \in \mathbb{F}_q\}$ es un $[n,1,n]$-código lineal. La capacidad de este código para detectar y corregir errores es la mejor posible entre los códigos de longitud n en vista de los teoremas 2.4.3 y 2.4.6, respectivamente. Sin embargo, su tamaño siempre es q, el cual es muy bajo comparado con q^n, que es el máximo número de palabras que puede tener un código con esta longitud.

- El código binario $L_4 = \{(0,0,0,0),(1,0,1,0),(0,1,0,1),(1,1,1,1)\}$ es lineal con longitud 4 y distancia mínima 2. Además, su dimensión es 2 dado que como espacio vectorial es generado por $(1,0,1,0)$ y $(0,1,0,1)$. Es decir, L_4 es un $[4,2,2]$-código lineal binario.
- El código $L_5 = \{(0,0,0,0),(1,1,0,0),(2,2,0,0)\}$ es un $[4,1,2]$-código lineal ternario. □

Para un (n,M)-código arbitrario en general, determinar su distancia mínima puede ser un proceso dispendioso, ya que su valor depende del cálculo de $\binom{M}{2}$ distancias de Hamming. Para códigos lineales este cálculo se reduce considerablemente debido al siguiente resultado.

Teorema 3.1.3 Sea L un código lineal sobre $\mathbb{F}_q$. Entonces $d(L) = wt(L)$.

Demostración. Sean $\mathbf{x},\mathbf{y} \in L$ tales que $d(L) = d(\mathbf{x},\mathbf{y})$. Por el Lema 2.5.3 se tiene $d(\mathbf{x},\mathbf{y}) = wt(\mathbf{x}-\mathbf{y})$. Como L es lineal entonces $\mathbf{x}-\mathbf{y} \in L$, luego $wt(\mathbf{x}-\mathbf{y}) \geq wt(L)$; es decir $d(L) \geq wt(L)$.

Por otro lado, sea $\mathbf{z} \in L$ una palabra código no nula tal que $wt(\mathbf{z}) = wt(L)$. De la Ecuación (2.3) sabemos que $wt(\mathbf{z}) = d(\mathbf{z},\mathbf{0})$. Además $d(\mathbf{z},\mathbf{0}) \geq d(L)$, luego $wt(L) \geq d(L)$. Por lo tanto, $d(L) = wt(L)$. ■

Ejemplo 3.1.4 Considere $L = \{0000,1000,0100,1100\}$ un código lineal binario. Para hallar su distancia mínima, en lugar de calcular $\binom{4}{2} = 6$ distancias, el anterior resultado nos dice que es suficiente con calcular el peso de las 3 palabras no nulas. Esto es

$$wt(1000) = 1,\ wt(0100) = 1,\ wt(1100) = 2,$$

de lo cual concluimos que $d(L) = 1$. □

A continuación mostramos una expresión para calcular el número de bases de un código lineal sobre $\mathbb{F}_q$.

Teorema 3.1.5 Sea L un código lineal sobre $\mathbb{F}_q$. Si $\dim(L) = k$, entonces

1. L tiene q^k elementos.

2. L tiene
$$\frac{1}{k!}\prod_{i=0}^{k-1}(q^k - q^i) \tag{3.1}$$
bases diferentes.

Demostración.

1. Sea $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ una base para L, entonces
$$L = \{\lambda_1\mathbf{v}_1 + \cdots + \lambda_k\mathbf{v}_k : \lambda_i \in \mathbb{F}_q\}.$$
Como $|\mathbb{F}_q| = q$, entonces existen q elecciones diferentes para cada uno de los coeficientes $\lambda_i \in \mathbb{F}_q$, por lo tanto L tiene q^k elementos.
2. Sea $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ una base para L. Dado que B es linealmente independiente entonces el vector nulo $\mathbf{0}$ no pertenece a B. Luego, por la primera parte, para $\mathbf{v}_1$ existen $q^k - 1$ elecciones posibles. Como B es una base, entonces se debe cumplir que $\mathbf{v}_2 \notin \langle \mathbf{v}_1 \rangle$. Es decir, existen $q^k - q$ posibilidades para $\mathbf{v}_2$. Similarmente, para cada i, con $2 \leq i \leq k$, se debe cumplir $\mathbf{v}_i \notin \langle \mathbf{v}_1, \ldots, \mathbf{v}_{i-1} \rangle$, de manera que existen $q^k - q^{i-1}$ elecciones posibles para $\mathbf{v}_i$. De esta manera, por el principio de la multiplicación, el número de k-uplas ordenadas distintas $(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ que conforman una base para L es igual a $\prod_{i=0}^{k-1}(q^k - q^i)$. Como el orden de los elementos $\mathbf{v}_1, \ldots, \mathbf{v}_k$ para una base es irrelevante, entonces el número de bases distintas es
$$\frac{1}{k!}\prod_{i=0}^{k-1}(q^k - q^i). \tag{3.2}$$
■

De la primera parte del Teorema 3.1.5 se tiene el siguiente resultado.

Corolario 3.1.6 Sea L un código lineal sobre $\mathbb{F}_q$. Entonces $\dim(L) = \log_q |L|$.

El número de códigos lineales de longitud n y dimensión k sobre $\mathbb{F}_q$ está dado en el siguiente resultado.

Teorema 3.1.7 Para $0 \leq k \leq n$, el número de $[n,k]$-códigos lineales sobre $\mathbb{F}_q$ es

$$\begin{bmatrix} n \\ k \end{bmatrix}_q = \frac{[n]_q!}{[k]_q![n-k]_q!}, \tag{3.3}$$

donde para $i \in \mathbb{Z}^+$, $[i]_q$ y $[n]_q!$ se definen como

$$[i]_q = \frac{q^i - 1}{q-1} = 1 + q + \cdots + q^{i-1}, \text{ con } [0]_q = 1.$$
$$[n]_q! = [1]_q[2]_q \cdots [n]_q.$$

Demostración. Notemos que con cada subconjunto linealmente de $\mathbb{F}_q^n$ con k elementos podemos generar un $[n,k]$-código lineal sobre $\mathbb{F}_q$. En seguida contamos el número de formas en las que podemos encontrar uno de estos subconjuntos. Igual que en la demostración del Teorema 3.1.5, tenemos que $\mathbf{0} \neq \mathbf{c}_1$ se puede escoger en $\mathbb{F}_q^n$ en $q^n - 1$ formas distintas, $\mathbf{c}_2 \notin \langle \mathbf{c}_1 \rangle$ se puede seleccionar en $q^n - q$ formas diferentes, $\mathbf{c}_3 \notin \langle \mathbf{c}_1, \mathbf{c}_2 \rangle$ en $q^n - q^2$ modos diferentes y así sucesivamente.

De esta forma, el número de maneras de tomar un subconjunto $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$ con k vectores linealmente independientes en $\mathbb{F}_q^n$ está dado por

$$\frac{1}{k!} \prod_{i=0}^{k-1} (q^n - q^i). \tag{3.4}$$

Ahora por el Teorema 3.1.5, tenemos que la expresión (3.2) representa el número de estos subconjuntos que generan el mismo subespacio. Por lo tanto, el cociente entre (3.4) y (3.2) es igual al número de subespacios de $\mathbb{F}_q^n$ de dimensión k que se busca; esto es

$$\prod_{i=0}^{k-1} \left(\frac{q^n - q^i}{q^k - q^i} \right) = \frac{(q^n-1)(q^n-q)(q^n-q^2)\cdots(q^n-q^{k-1})}{(q^k-1)(q^k-q)(q^k-q^2)\cdots(q^k-q^{k-1})}.$$

Al factorizar todas las potencias de q posibles en el numerador y el denominador en la expresión anterior tenemos que

$$\begin{aligned}
&\prod_{i=0}^{k-1}\left(\frac{q^n-q^i}{q^k-q^i}\right)\\
&\quad=\frac{(q^n-1)q(q^{n-1}-1)q^2(q^{n-2}-1)\cdots q^{k-1}(q^{n-k+1}-1)}{(q^k-1)q(q^{k-1}-1)q^2(q^{k-2}-1)\cdots q^{k-1}(q-1)}\\
&\quad=\frac{(q^n-1)(q^{n-1}-1)(q^{n-2}-1)\cdots(q^{n-k+1}-1)}{(q^k-1)(q^{k-1}-1)(q^{k-2}-1)\cdots(q-1)} \qquad (3.5)\\
&\quad=\frac{(q-1)^k[n]_q[n-1]_q[n-2]_q\cdots[n-k+1]_q}{(q-1)^k[k]_q[k-1]_q[k-2]_q\cdots[1]_q} \qquad (3.6)\\
&\quad=\frac{[n]_q!}{[k]_q![n-k]_q!}=\begin{bmatrix}n\\k\end{bmatrix}_q,
\end{aligned}$$

donde en (3.5) factorizamos en cada término $q-1$ para obtener (3.6). De esta forma, se sigue el resultado. ■

Observación 3.1.8 Los números $[n]_q!$ y $\begin{bmatrix}n\\k\end{bmatrix}_q$ se conocen como el *q-factorial* y el *coeficiente q-binomial Gaussiano*, respectivamente.

Ejemplo 3.1.9 El número de códigos lineales binarios de dimensión 2 en $\mathbb{F}_2^4$ es

$$\begin{aligned}
\begin{bmatrix}4\\2\end{bmatrix}_2&=\frac{[4]_2!}{[2]_2![2]_2!}=\frac{[3]_2[4]_2}{[1]_2[2]_2}\\
&=\frac{(2^3-1)(2^4-1)}{(2-1)(2^2-1)}=\frac{7\cdot 15}{3}=35.
\end{aligned}$$

Por su parte, el número de códigos lineales de dimensión 2 y dimensión 4 en $\mathbb{F}_5^4$ es:

$$\begin{aligned}
\begin{bmatrix}4\\2\end{bmatrix}_5&=\frac{[4]_5!}{[2]_5![2]_5!}=\frac{(5^3-1)(5^4-1)}{(5-1)(5^2-1)}=\frac{124\cdot 624}{4\cdot 24}=806,\\
\begin{bmatrix}4\\4\end{bmatrix}_5&=\frac{[4]_5!}{[4]_5![0]_2!}=1,
\end{aligned}$$

respectivamente. □

3.2. Matriz generadora

Como un código lineal es un subespacio vectorial, sus elementos quedan totalmente descritos por los elementos de una de sus bases. Una matriz cuyas filas están formadas por los elementos de una base recibe un nombre especial, pues como veremos en esta sección es de gran utilidad en los procesos de codificación y decodificación.

Definición 3.2.1 Sea L un $[n,k]$-código lineal. Una matriz G de tamaño $k \times n$, cuyas filas forman una base para L se llama *matriz generadora* para L.

Observación 3.2.2 Si L es un $[n,k]$-código lineal con matriz generadora G, entonces

$$L = \{\mathbf{x}G : \mathbf{x} \in \mathbb{F}_q^k\}.$$

En efecto, supongamos que $G = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_k \end{pmatrix}$. Notemos que si $\mathbf{c} = \mathbf{x}G$, donde $\mathbf{x} = (x_1, x_2, \ldots, x_k) \in \mathbb{F}_q^k$, entonces $\mathbf{c} = x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + \cdots + x_k\mathbf{v}_k$, lo cual es una combinación lineal de las filas de G y por lo tanto una palabra código.
Por otro lado, dado que toda palabra código $\mathbf{c} \in L$ se puede escribir como combinación lineal de las filas de G, se tiene que existen escalares $a_1, a_2, \ldots, a_k \in \mathbb{F}_q$ tales que

$$\mathbf{c} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k = (a_1, a_2, \ldots, a_k)G.$$

La matriz generadora G para un $[n,k]$-código lineal, permite entender de manera simple el proceso de codificación de los mensajes fuente a mensajes con símbolos de redundancia que pueden servir para detectar o corregir errores. Si la fuente se representa por medio de un conjunto de palabras sobre $\mathbb{F}_q$ de longitud k, entonces cada mensaje fuente $\mathbf{x}$ se codifica como la palabra código $\mathbf{x}G$, la cual contiene $n-k$ símbolos de redundancia.

Ejemplo 3.2.3 Considere el código lineal binario C con matriz generadora

$$G = \begin{pmatrix} 1100 \\ 0111 \\ 1010 \end{pmatrix}.$$

Esta matriz puede ser usada para codificar los mensajes fuente consistentes de palabras de $\mathbb{F}_2^3$. Por ejemplo, el mensaje $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{F}_2^3$, se codifica como la palabra código

$$(x_1, x_2, x_3) \begin{pmatrix} 1100 \\ 0111 \\ 1010 \end{pmatrix} = (x_1 + x_3, x_1 + x_2, x_2 + x_3, x_2). \qquad \square$$

El proceso de codificación para códigos lineales se puede llevar de manera más eficiente, cuando la matriz generadora tiene una forma especial.

Definición 3.2.4 Sea L un $[n,k]$-código lineal. Una matriz generadora G para L en la forma $G = (I_k \mid A)$, se denomina *matriz generadora en la forma estándar*.

Por el Teorema 3.1.5 para un $[n,k]$-código lineal L se tienen $\prod_{i=0}^{k-1}(q^k - q^i)$ matrices generadoras distintas; sin embargo, es posible que ninguna de estas matrices esté en la forma estándar como se ilustra en el siguiente ejemplo.

Ejemplo 3.2.5 Determinemos todas las matrices generadoras para el código lineal binario

$$L = \{000, 001, 100, 101\}.$$

Por el Corolario 3.1.6, $\dim(L) = \log_2 4 = 2$. Por el Teorema 3.1.5, el número de bases diferentes para L es

$$\frac{1}{2!}\prod_{i=0}^{2-1}(2^2 - 2^i) = \frac{1}{2!}(2^2 - 1)(2^2 - 2) = 3.$$

Puede verificarse que las bases de L son:

$$\{001, 100\},\ \{001, 101\},\ \{100, 101\}.$$

De esta manera, las matrices generadoras para L son:

$$\begin{pmatrix}001\\100\end{pmatrix}, \begin{pmatrix}100\\001\end{pmatrix}, \begin{pmatrix}001\\101\end{pmatrix}, \begin{pmatrix}101\\001\end{pmatrix}, \begin{pmatrix}100\\101\end{pmatrix}, \begin{pmatrix}101\\100\end{pmatrix}.$$

Como se puede observar, ninguna de estas matrices está en la forma estándar. □

Esta dificultad se puede resolver por medio del concepto de códigos equivalentes y haciendo uso del siguiente resultado.

Teorema 3.2.6 Cualquier código lineal L es equivalente a un código lineal L' con matriz generadora en la forma estándar.

Demostración. Supongamos que G es una matriz generadora para un $[n,k]$-código lineal L que no está la forma estándar. Por operaciones elementales de fila, G se puede transformar a su forma escalonada reducida por filas, digamos G'. Dado que las filas de G forman una base para L, entonces G' tiene k entradas principales (pivotes). Permutando las columnas de G' con pivote, es posible formar la matriz identidad de orden k en la parte izquierda de la matriz. El resto de columnas se ubican enseguida de la matriz identidad, obteniendo así una matriz generadora G'' en la forma estándar para un código L', el cual es por construcción equivalente al código L. ■

Ejemplo 3.2.7 Para el Ejemplo 3.2.5, elijamos la matriz generadora en la forma escalonada reducida por filas

$$G = \begin{pmatrix}100\\001\end{pmatrix}.$$

Permutando la segunda columna con la tercera se obtiene la matriz generadora en la forma estándar

$$G' = \begin{pmatrix}100\\010\end{pmatrix}.$$

Las filas de esta matriz generan el código lineal $L' = \{000,010,100,110\}$, el cual es equivalente al código L. □

Notemos que si G está en la forma estándar, entonces L es sistemático en las primeras k coordenadas. En este caso la codificación y la decodificación resultan bastante sencillas: si $\mathbf{x} \in \mathbb{F}_q^k$

entonces $\mathbf{x}$ se codifica como $\mathbf{x}G = \mathbf{x}(I_k \mid A) = (\mathbf{x} \mid \mathbf{x}A)$, siendo $\mathbf{x}A$, los símbolos de redundancia. Similarmente, si se recibe el vector $(\mathbf{x} \mid \mathbf{x}A)$, entonces esta palabra se decodifica a $\mathbf{x}$.

Ejemplo 3.2.8 La matriz G mostrada a continuación es una matriz generadora en la forma estándar para un código lineal binario L,

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

Veamos cómo es el proceso de codificación. Para cada $\mathbf{x} = (x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4$, tenemos

$$\begin{aligned} \mathbf{x}G &= (x_1, x_2, x_3, x_4) \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \\ &= (x_1, x_2, x_3, x_4, x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4). \end{aligned}$$

Note que las primeras cuatro coordenadas del vector $\mathbf{x}G$ son iguales a las de la palabra $\mathbf{x}$. Esto proporciona una ventaja a la hora de decodificar un mensaje recibido, dado que la información está precisamente en esas posiciones. Por ejemplo, la codificación del vector 1010 es la palabra código 1010101, y por tanto en caso de recibir esta palabra, su decodificación puede hacerse de manera inmediata tomando sus primeras cuatro componentes. □

De manera general, tenemos el siguiente teorema.

Teorema 3.2.9 Sea L un $[n,k]$-código lineal. Para cualquier k coordenadas dadas, existe un código equivalente a L que es sistemático en estas posiciones.

Demostración. Sea G una matriz generadora para L. Mediante operaciones elementales de fila, podemos transformar G en su forma escalonada reducida por filas: G'. Sean $\mathbf{c}_{i_j}$, para $j = 1, \ldots, k$, las columnas con pivote de G'. Tenemos que L es sistemático en estas posiciones. Sean $v_1, v_2, \ldots, v_k$ las coordenadas de las palabras del código L. Por permutación de las columnas $\mathbf{c}_{i_j}$, en las co-

lumnas $\mathbf{c}_{v_1}, \mathbf{c}_{v_2}, \ldots, \mathbf{c}_{v_k}$, se obtiene una matriz generadora G'' para un código L' equivalente al código L, que además, por construcción, es sistemático en estas k posiciones. ■

Ejemplo 3.2.10 El código lineal ternario L_1 con matriz generadora

$$G_1 = \begin{pmatrix} 1 & 0 & 2 & 0 & 0 & 2 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix},$$

es equivalente al código lineal L_2 con matriz generadora en forma estándar

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 1 & 2 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}.$$

De esta forma, observamos que la matriz $G_2 = (I_5|A)$, donde

$$A = \begin{pmatrix} 0 & 2 & 2 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}.$$ □

3.3. El código dual

Dado que un código lineal es un subespacio vectorial, su complemento ortogonal también es un código lineal. Este código y algunas de sus propiedades serán tratadas en la presente sección.

Definición 3.3.1 Sea L un $[n,k]$-código lineal. El conjunto

$$L^{\perp} = \{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{x} \cdot \mathbf{c} = \mathbf{0} \text{ para todo } \mathbf{c} \in L\}$$

se denomina el *código dual* de L.

Observación 3.3.2 Podemos demostrar que el conjunto $L^\perp$ es un un subespacio vectorial de $\mathbb{F}_q^n$, entonces tiene sentido que le llamemos código dual de L. En efecto, dado que $\mathbf{0} \cdot \mathbf{c} = 0$, para todo $\mathbf{c} \in L$, tenemos que $L^\perp \neq \emptyset$. Además, si $\mathbf{x}, \mathbf{y} \in L^\perp$ y $\alpha, \beta \in \mathbb{F}_q$, entonces $(\alpha\mathbf{x} + \beta\mathbf{y}) \cdot \mathbf{c} = \alpha(\mathbf{x} \cdot \mathbf{c}) + \beta(\mathbf{y} \cdot \mathbf{c}) = \alpha 0 + \beta 0 = 0$, entonces $\alpha\mathbf{x} + \beta\mathbf{y} \in L^\perp$.

En el siguiente teorema se establecen algunas propiedades básicas de los códigos duales.

Teorema 3.3.3 Sea L un código lineal sobre $\mathbb{F}_q$. Entonces

1. Si G es una matriz generadora para L, entonces
$$L^\perp = \{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{x}G^\top = \mathbf{0}\}.$$
2. Si L un $[n,k]$-código lineal, entonces $L^\perp$ es un $[n, n-k]$-código lineal. Es decir, $\dim(L) + \dim(L^\perp) = n$.
3. Además, $(L^\perp)^\perp = L$.

Demostración.

1. Se probará que el conjunto de la Definición 3.3.1, es igual al conjunto
$$M = \{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{x}G^\top = \mathbf{0}\}.$$
Sea $\mathbf{x} \in L^\perp$, por definición $\mathbf{x}$ es ortogonal a cada palabra código de L, en particular es ortogonal a las filas de la matriz generadora. Luego $L^\perp \subseteq M$. Para $\mathbf{x} \in M$, sea $\mathbf{c} \in L$, determinemos el valor de $\mathbf{x} \cdot \mathbf{c}$. Sean $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ las filas de G, por definición de matriz generadora, tenemos que $\mathbf{c} = \lambda_1\mathbf{v}_1 + \lambda_2\mathbf{v}_2 + \cdots + \lambda_k\mathbf{v}_k$ para algunos $\lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{F}_q$. Luego
$$\begin{aligned}\mathbf{x} \cdot \mathbf{c} &= \mathbf{x} \cdot (\lambda_1\mathbf{v}_1 + \lambda_2\mathbf{v}_2 + \cdots + \lambda_k\mathbf{v}_k) \\ &= \lambda_1(\mathbf{x} \cdot \mathbf{v}_1) + \lambda_2(\mathbf{x} \cdot \mathbf{v}_2) + \cdots + \lambda_k(\mathbf{x} \cdot \mathbf{v}_k) = \mathbf{0}.\end{aligned}$$
Luego $M \subseteq L^\perp$ y así $L^\perp = M$.
2. Si $k = 0$, entonces $L = \{\mathbf{0}\}$, luego $L^\perp = \mathbb{F}_q^n$ y obtenemos el resultado. Supongamos que $k \geq 1$, sea G una matriz de ta-

maño $k \times n$ generadora de L. Por el Teorema 3.2.6, L es equivalente a un código con matriz generadora en la forma estándar $(I_k \mid A)$. De esta forma, $L^\perp$ es equivalente a un código que es el espacio solución del sistema de k ecuaciones con n incógnitas, dado por

$$\begin{aligned} x_1 + a_{11}x_{k+1} + \cdots + a_{1,n-k}x_n &= 0 \\ x_2 + a_{21}x_{k+1} + \cdots + a_{2,n-k}x_n &= 0 \\ &\vdots \\ x_k + a_{k1}x_{k+1} + \cdots + a_{k,n-k}x_n &= 0. \end{aligned}$$

En consecuencia, despejando las incógnitas $x_1, \ldots, x_k$ en términos de las últimas $n-k$ incógnitas, se obtienen $n-k$ vectores que forman una base de $L^\perp$. Por lo tanto, $\dim\left(L^\perp\right) = n-k$.

3. Finalmente, notemos que de la definición de $L^\perp$ tenemos que $L \subset (L^\perp)^\perp$. De la segunda parte, obtenemos que

$$\dim((L^\perp)^\perp) = n - (n-k) = k = \dim(L),$$

por lo tanto $(L^\perp)^\perp = L$. ■

Ejemplo 3.3.4 Determinemos el código dual $L^\perp$ del código lineal binario $L = \{000, 101, 010, 111\}$. Como $\dim(L) = \log_2 4 = 2$ entonces una base para L está conformada por 101 y 010. Por lo tanto, una matriz generadora para L es

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

Ahora, determinemos una base para el espacio nulo del sistema $\mathbf{x}G^\top = \mathbf{0}$, donde $\mathbf{x} = x_1x_2x_3 \in \mathbb{F}_2^3$. Por definición, el espacio generado por estos elementos es $L^\perp$.

$$\mathbf{x}G^\top = (x_1, x_2, x_3)\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} = (x_1 + x_3, x_2) = (0 \quad 0).$$

De ahí que una base para $L^\perp$ es 101 y por lo tanto $L^\perp = \{000, 101\}$. □

3.4. Códigos auto-ortogonales y auto-duales

Note que en el Ejemplo 3.3.4 L y $L^{\perp}$ son subespacios ortogonales; sin embargo $L \cap L^{\perp} \neq \{000\}$. La razón de este hecho se debe a que los códigos lineales se definen sobre campos finitos como lo hemos visto. Esta idea conduce a las siguientes definiciones.

Definición 3.4.1 Sea L un código lineal.
1. Si $L \subseteq L^{\perp}$, se dice que L es *auto-ortogonal* .
2. Si $L = L^{\perp}$, diremos que L es *auto-dual* .

Ejemplo 3.4.2

1. Considere el código lineal generado por $S = \{01201\} \subset \mathbb{F}_3^5$. Este código es $L = \{00000, 01201, 02102\}$ y es auto-ortogonal, ya que

$$01201 \cdot 01201 = 0,\ 01201 \cdot 02102 = 0,\ 02102 \cdot 02102 = 0.$$

   Note que este código no es auto-dual, ya que la palabra $00111 \in L^{\perp}$, pero $00111 \notin L$.

2. El $[4,2]$-código lineal binario $L = \{0000, 1100, 0011, 1111\}$ es auto-dual. Para mostrarlo, sea G la matriz generadora para L formada por las filas 1100 y 0011. Determinemos su código dual, lo cual es equivalente a encontrar una base que lo genere. Sea $\mathbf{x} = (x_1, x_2, x_3, x_4) \in L^{\perp}$, entonces al resolver el sistema

$$\mathbf{x}G^{\top} = (x_1, x_2, x_3, x_4) \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} = (x_1 + x_2, x_3 + x_4) = (0,0),$$

   tenemos que $x_1 = x_2$ y $x_3 = x_4$, lo cual implica que el conjunto $\{1100, 0011\}$, es una base para el código dual. Como estos elementos coinciden con las filas de la matriz generadora G para L entonces generan el mismo espacio y así $L = L^{\perp}$. □

Un resultado que caracteriza la dimensión de los códigos auto-ortogonales y auto-duales se formula en el siguiente teorema.

Teorema 3.4.3 La dimensión de un código auto-ortogonal de longitud n, es menor o igual que $n/2$; mientras que la dimensión de un código auto-dual de longitud n es $n/2$, donde n es un número par.

Demostración. Si L es un $[n,k]$-código q-ario auto-ortogonal, entonces $L^{\perp}$ es un $[n,n-k]$-código lineal. Como L es auto-ortogonal, entonces $L \subseteq L^{\perp}$, lo cual implica que $\mid L \mid = q^k \leq q^{n-k} = \mid L^{\perp} \mid$; es decir $k \leq n-k$ y de ahí que $k \leq n/2$.
Si L es auto-dual, entonces $k = n-k$; esto es $k = n/2$, pero como k es entero, entonces n es par. ■

Otros resultados con respecto a esta clase de códigos lineales son los siguientes.

Teorema 3.4.4 Sea G una matriz generadora para un código lineal L sobre $\mathbb{F}_q$, donde $q = 2$ o $q = 3$. Entonces L es auto-ortogonal si y sólo si todas las filas distintas de G son ortogonales y tienen peso divisible por q.

Demostración. Por definición, las filas distintas de G son ortogonales. Probemos por separado para $q = 2$ y $q = 3$ que las filas de G tienen peso divisible por q.
Para $q = 2$, supongamos por contradicción, que una fila $\mathbf{x}$ de G tiene peso impar, es decir $wt(\mathbf{x}) = 2t+1$, para algún entero no negativo t. Luego, $\mathbf{x} \cdot \mathbf{x} = 2t+1 = 1 \neq 0$, pero esto es una contradicción, ya que L es auto-ortogonal.
Para $q = 3$, supongamos por contradicción que existe una fila $\mathbf{x}$ de G tal que $wt(\mathbf{x}) = 3t+1$ o $wt(\mathbf{x}) = 3t+2$. En el primer caso, sea x_i una coordenada no nula de $\mathbf{x}$. Entonces $x_i = 1$ o $x_i = 2$. En cualquier caso se tiene que $x_i^2 = 1$, ya que $x_i \in \mathbb{F}_3 \setminus \{0\}$. Por lo tanto, $\mathbf{x} \cdot \mathbf{x} = 3t+1 = 1 \neq 0$, lo cual contradice la hipótesis. Similarmente, si $wt(\mathbf{x}) = 3t+2$, entonces $\mathbf{x} \cdot \mathbf{x} = 3t+2 = 2 \neq 0$. Luego $wt(\mathbf{x}) = 3t$.

Recíprocamente, sean $\mathbf{x}, \mathbf{c} \in L$, $\dim(L) = k$ y $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k$ las filas de la matriz G. Entonces $\mathbf{x}$ y $\mathbf{c}$ se pueden escribir en la forma

$$\mathbf{x} = \lambda_1\mathbf{g}_1 + \lambda_2\mathbf{g}_2 + \cdots + \lambda_k\mathbf{g}_k,$$
$$\mathbf{c} = \beta_1\mathbf{g}_1 + \beta_2\mathbf{g}_2 + \cdots + \beta_k\mathbf{g}_k,$$

para algunos $\lambda_1, \ldots, \lambda_k, \beta_1, \ldots, \beta_k \in \mathbb{F}_q$.
Luego

$$\begin{aligned}\mathbf{x}\cdot\mathbf{c} &= (\lambda_1\mathbf{g}_1 + \lambda_2\mathbf{g}_2 + \cdots + \lambda_k\mathbf{g}_k)\cdot(\beta_1\mathbf{g}_1 + \beta_2\mathbf{g}_2 + \cdots + \beta_k\mathbf{g}_k)\\ &= \sum_{i=1}^{k}\lambda_1\beta_i(\mathbf{g}_1\cdot\mathbf{g}_i) + \sum_{i=1}^{k}\lambda_2\beta_i(\mathbf{g}_2\cdot\mathbf{g}_i) + \cdots + \sum_{i=1}^{k}\lambda_k\beta_i(\mathbf{g}_k\cdot\mathbf{g}_i)\\ &= \sum_{i=1}^{k}\sum_{j=1}^{k}\lambda_i\beta_j(\mathbf{g}_i\cdot\mathbf{g}_j) = \sum_{i=1}^{k}\lambda_i\beta_i(\mathbf{g}_i\cdot\mathbf{g}_i) = 0,\end{aligned}$$

dado que las filas distintas de G son ortogonales ($\mathbf{g}_i\cdot\mathbf{g}_j = 0$ para $i \neq j$) y tienen peso divisible por q ($\mathbf{g}_i\cdot\mathbf{g}_i = \beta q = 0$, con $\beta \in \mathbb{F}_q$). ■

Teorema 3.4.5 Si las filas distintas de una matriz generadora para un código lineal binario L son ortogonales y tienen peso divisible por 4, entonces L es auto-ortogonal y todos los pesos en L son divisibles por 4.

Demostración. Si las filas de la matriz generadora tienen peso divisible por 4, entonces también son divisibles por 2; así por el Teorema 3.4.4 el código L es auto-ortogonal. Para la segunda parte, sean $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k$, las filas de la matriz generadora G para L. Probemos por inducción que todas las combinaciones lineales no nulas de G tienen peso divisible por 4.

Claramente, para $\mathbf{x} = \mathbf{g}_i$ con $i = 1, \ldots, k$ se tiene

$$wt(\mathbf{x}) = wt(\mathbf{g}_i),$$

es divisible por 4, ya que todos los pesos $wt(\mathbf{g}_i)$ son divisibles por 4.

Para $\mathbf{g}_i + \mathbf{g}_j$ con $i, j = 1, \ldots, k$, $i \neq j$, por el Lema 2.5.3 se tiene

$$\begin{aligned}wt(\mathbf{x}) &= wt(\mathbf{g}_i + \mathbf{g}_j)\\ &= wt(\mathbf{g}_i) + wt(\mathbf{g}_j) - 2wt(\mathbf{g}_i \cap \mathbf{g}_j).\end{aligned} \tag{3.7}$$

Pero $wt(\mathbf{g}_i \cap \mathbf{g}_j)$ es siempre par, ya que de lo contrario $\mathbf{w} = \mathbf{g}_i + \mathbf{g}_j \in L$ tendría peso impar y por lo tanto $\mathbf{w} \cdot \mathbf{w} = 1 \neq 0$, lo cual contradice la definición de código auto-ortogonal. Por hipótesis, sabemos que $wt(\mathbf{g}_i)$ y $wt(\mathbf{g}_j)$ son divisibles por 4, por lo tanto, de (3.7), $wt(\mathbf{x})$ es divisible por 4. Siguiendo un razonamiento similar, para cualquier combinación no nula de las filas de G el peso es divisible por 4. ■

Ejemplo 3.4.6 El $[4,2,3]$-código ternario L con matriz generadora

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

es un código auto-dual. En efecto, primero notemos que las filas de la matriz son ortogonales entre sí. □

El siguiente teorema establece algunos criterios para garantizar la existencia de códigos auto-duales.

Teorema 3.4.7 Un $[n, n/2]$ código auto-dual q-ario existe si y sólo si una de las siguientes condiciones se satisface

1. q y n son pares.
2. $q \equiv 1 (\text{mód} 4)$ y n es par.
3. $q \equiv 3 (\text{mód} 4)$ y n es divisible por 4.

Demostración. La demostración se omite dado que requiere de nuevos aspectos teóricos. El lector interesado puede consultarla en [25, p. 633]. ■

Los códigos auto-duales son importantes en la teoría de códigos, porque muchos de los mejores códigos conocidos pertenecen a esta clase, como por ejemplo, el código extendido de Hamming, el código extendido de Golay y el código binario extendido de un código de residuos cuadráticos (cuando $p \equiv -1 \pmod 8$). Además, la distancia mínima de estos códigos satisface una cota superior menor que la cota de Singleton, ver [15] y [19, Thm. 9.3.5].

En general, es difícil determinar la distribución de pesos de un código dado. Una de las herramientas más importantes para este propósito es la *Identidad de MacWilliams*, la cual relaciona el polinomio enumerador de pesos de un código lineal con el polinomio enumerador de pesos de su dual. A continuación, presen-

tamos este resultado, cuya demostración puede ser consultada en [36, Cor. 5.2.9].

Teorema 3.4.8 (Identidad de MacWilliams) Sean L un código lineal q-ario de longitud n y

$$W_L(x) = \sum_{k=0}^{n} A_k x^k,$$
$$W_{L^\perp}(x) = \sum_{k=0}^{n} A_k^\perp x^k,$$

los polinomios enumeradores de pesos de L y $L^\perp$, respectivamente. Entonces

$$W_{L^\perp}(x) = \frac{1}{|L|}(1+(q-1)x)^n W_L\left(\frac{1-x}{1+(q-1)x}\right).$$

3.5. Matriz de control de paridad

A continuación presentamos una matriz que es útil en el proceso de decodificación.

Definición 3.5.1 Una *matriz de control de paridad* H para un código lineal L, es una matriz generadora para el código dual $L^\perp$.

De esta manera, tenemos la siguiente caracterización de un código lineal L.

Teorema 3.5.2 Sea L un código lineal con matriz de control de paridad H. Entonces

$$\mathbf{x} \in L \text{ si y sólo si } \mathbf{x}H^\top = \mathbf{0}. \tag{3.8}$$

Demostración. Dado que $L^\perp$ es un código lineal con matriz generadora H, del Teorema 3.3.3 tenemos que

$$L = (L^\perp)^\perp = \{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{x}H^\top = \mathbf{0}\}.$$ ■

Además, si $H = (h_{ij})$, $r = n - k$ y $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, entonces el sistema $\mathbf{x}H^\top = \mathbf{0}$, es decir

$$\begin{aligned} h_{11}x_1 + h_{12}x_2 + \cdots + h_{1n}x_n &= 0 \\ h_{21}x_1 + h_{22}x_2 + \cdots + h_{2n}x_n &= 0 \\ &\vdots \\ h_{r,1}x_1 + h_{r,2}x_2 + \cdots + h_{r,n}x_n &= 0, \end{aligned} \tag{3.9}$$

muestra que las filas de H son los coeficientes de un sistema de ecuaciones lineales cuyas soluciones son los elementos de L. Las ecuaciones en el sistema (3.9) se denominan *ecuaciones de control de paridad*.

Proposición 3.5.3 Sea L un $[n,k]$-código lineal sobre $\mathbb{F}_q$, con matriz generadora G. Entonces

1. El vector $\mathbf{v} \in \mathbb{F}_q^n$ pertenece a $L^\perp$ si y sólo si $\mathbf{v}$ es ortogonal a toda fila de G, o equivalentemente $\mathbf{v} \in L^\perp$ si y sólo si $\mathbf{v}G^\top = \mathbf{0}$.
2. Sea H una matriz de tamaño $(n-k) \times n$. Entonces H es una matriz de control de paridad para L si y sólo si las filas de H son linealmente independientes y $HG^\top = \mathbf{O}$.

Demostración.

1. Dado que G es una matriz generadora de $L = (L^\perp)^\perp$, entonces G es una matriz de control de paridad de $L^\perp$. De esta forma, del Teorema 3.5.2 se sigue el resultado.
2. Supongamos que H es una matriz de control de paridad para L. Por definición las filas de H son linealmente independientes y además como las filas de H son elementos del dual $L^\perp$, por la primera parte se sigue que $HG^\top = \mathbf{O}$. Recíprocamente, si $HG^\top = \mathbf{O}$, entonces nuevamente por la primera parte las filas de H pertenecen a $L^\perp$. Esto implica que el espacio generado por las filas de H está contenido en $L^\perp$. Dado que por hipótesis las filas de H son linealmente independientes, el espacio generado por las filas de H tiene dimensión $n - k$; es decir, las filas de H generan a $L^\perp$. En conclusión, H es una matriz de control de paridad de L. ■

Observación 3.5.4 Sea L un código lineal con matriz generadora en la forma estándar $G = (I_k \mid A)$ de tamaño $k \times n$. Consideremos la matriz H de la forma $H = (-A^\top \mid I_{n-k})$, donde $A^\top$ es la transpuesta de A. Entonces

$$GH^\top = (I_k \mid A)\begin{pmatrix} -A \\ I_{n-k} \end{pmatrix} = -A + A = 0.$$

Es decir, las filas de H son ortogonales a las filas de G y como el rango de H es igual a $n - k = \dim(L^\perp)$, entonces H es una matriz generadora para el código dual $L^\perp$. Una matriz generadora para el código dual en la forma $H = (-A^\top \mid I_{n-k})$ se llama *matriz de control de paridad en la forma estándar*.

Ejemplo 3.5.5 Determinemos explícitamente los elementos de un código lineal binario L, cuya matriz de control de paridad es

$$H = \begin{pmatrix} 1101000 \\ 1010100 \\ 0110010 \\ 1100001 \end{pmatrix}.$$

Sea $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ un elemento de L. De la Observación 3.5.4 tenemos que $\mathbf{x} \in L$ si y sólo si $\mathbf{x}H^\top = \mathbf{0}$, es decir

$$\begin{aligned} \mathbf{x}H^\top &= (x_1, x_2, x_3, x_4, x_5, x_6, x_7)\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= (x_1 + x_2 + x_4, x_1 + x_3 + x_5, x_2 + x_3 + x_6, x_1 + x_2 + x_7) \\ &= (0,0,0,0). \end{aligned}$$

Las ecuaciones de control de paridad son

$$\begin{aligned} x_1 + x_2 + x_4 &= 0 \\ x_1 + x_3 + x_5 &= 0 \\ x_2 + x_3 + x_6 &= 0 \\ x_1 + x_2 + x_7 &= 0. \end{aligned}$$

Por lo tanto, $B = \{1001101, 0101011, 0010110\}$ es una base para L. □

Aunque la matriz generadora de un código lineal nos permite determinar inmediatamente su longitud y su dimensión, usualmente no sirve para calcular su distancia mínima. Sin embargo, esto si se puede realizar a partir de la matriz de control de paridad, como se muestra en el siguiente resultado.

Proposición 3.5.6 Sea L un $[n,k]$-código lineal con matriz de control de paridad H. Entonces

1. L tiene distancia mayor o igual a e si y sólo si cualquier $e-1$ columnas de H son linealmente independientes.
2. L tiene distancia menor o igual a e si y sólo si H tiene e columnas que son linealmente dependientes.

Demostración. Para $1 \leq i \leq n$, sea $\mathbf{c}_i$ la i-ésima columna de H.

1. Inicialmente, supongamos que cualquier conjunto con $e-1$ columnas de H es linealmente independiente. Sea $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ una palabra cualquiera no nula de L. Si $1 \leq wt(\mathbf{v}) = t \leq e-1$, por el Teorema 3.3.3 se tiene que

$$\mathbf{0} = \mathbf{v}H^\top = v_1\mathbf{c}_1^\top + v_2\mathbf{c}_2^\top + \cdots + v_n\mathbf{c}_n^\top. \tag{3.10}$$

   Dado que $wt(\mathbf{v}) = t$, entonces de la igualdad (3.10), existen t columnas de H linealmente dependientes, lo que contradice la hipótesis inicial. Luego, $e \leq wt(\mathbf{v})$ para cualquier $\mathbf{v} \in L$ y así $e \leq d(L)$.

   Recíprocamente, supongamos que $wt(L) \geq e$. Si H tuviera un conjunto con $e-1$ columnas linealmente dependientes, digamos $\mathbf{c}_{i_1}, \mathbf{c}_{i_2}, \ldots, \mathbf{c}_{i_{e-1}}$, existirían escalares no todos nulos $v_{i_1}, v_{i_2}, \ldots, v_{i_{e-1}} \in \mathbb{F}_q$ tales que

$$v_{i_1}\mathbf{c}_{i_1}^\top + v_{i_2}\mathbf{c}_{i_2}^\top + \cdots + v_{i_{e-1}}\mathbf{c}_{i_{e-1}}^\top = \mathbf{0}. \tag{3.11}$$

   Sea $\mathbf{x} \in \mathbb{F}_q^n$ tal que su i_k-ésima coordenada es igual a v_{i_k}, para $1 \leq k \leq e-1$ y sus demás coordenadas son iguales a 0. Entonces, $\mathbf{x}H^\top = \mathbf{0}$; y así $\mathbf{x} \in L$, ver Teorema 3.3.3. Luego $wt(L) \leq wt(\mathbf{x}) \leq e-1$, lo que es absurdo.

2. Observe que la segunda parte es en realidad una consecuencia de la primera. ■

El siguiente resultado es consecuencia inmediata del teorema anterior.

Teorema 3.5.7 Sean L un código lineal y H una matriz de control de paridad para L. Las siguientes proposiciones son equivalentes:

1. L tiene distancia mínima d.
2. Cualquier $d-1$ columnas de H son linealmente independientes y H tiene d columnas que son linealmente dependientes.

Ejemplo 3.5.8 Para la matriz de control H del Ejemplo 3.5.5 cualquier par de columnas son linealmente independientes, pero H tiene 3 columnas que son linealmente dependientes, como por ejemplo 1101, 1011, 0110. Del corolario anterior concluimos que $d(L) = 3$. □

3.6. Decodificación por síndrome

Veamos ahora cómo las matrices de control de paridad para un código lineal se emplean para diseñar un proceso de decodificación. Primero definimos algunos conceptos importantes.

Definición 3.6.1 Sea L un código lineal de longitud n sobre $\mathbb{F}_q$ con matriz de control de paridad H. Para cualquier $\mathbf{x} \in \mathbb{F}_q^n$, la palabra $\mathbf{x}H^\top$ se llama el *síndrome* de $\mathbf{x}$ y se denota por $s(\mathbf{x})$.

Claramente, para un $[n,k]$-código lineal el síndrome de un vector de $\mathbb{F}_q^n$ es un vector de longitud $n-k$. Además, del Teorema 3.5.2, $\mathbf{x} \in L$ si y sólo si $s(\mathbf{x}) = \mathbf{0}$.

Definición 3.6.2 Sean L un código lineal de longitud n sobre $\mathbb{F}_q$ e $\mathbf{x} \in \mathbb{F}_q^n$. La *clase lateral de L determinada por* $\mathbf{x}$ se define como el conjunto

$$\mathbf{x} + L = \{\mathbf{x} + \mathbf{c} : \mathbf{c} \in L\}.$$

Teniendo en cuenta que L es un grupo Abeliano con la suma se tiene que $L + \mathbf{x} = \mathbf{x} + L$. Además, el conjunto de las clases laterales es un espacio vectorial sobre $\mathbb{F}_q$ con las operaciones

$$(\mathbf{x} + L) + (\mathbf{y} + L) = (\mathbf{x} + \mathbf{y}) + L \quad \text{y} \quad a(\mathbf{x} + L) = a\mathbf{x} + L.$$

Adicionalmente, se puede probar que $\mathbf{x} + L = \mathbf{y} + L$ si y sólo si $\mathbf{x} - \mathbf{y} \in L$. A partir de esto, se tiene el siguiente resultado.

Teorema 3.6.3 Sea L un código lineal de longitud n sobre $\mathbb{F}_q$ con matriz de control de paridad H. Entonces $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ tienen el mismo síndrome si y sólo si $\mathbf{x}$ e $\mathbf{y}$ determinan la misma clase lateral de L.

Demostración. Supongamos que $\mathbf{x} + L = \mathbf{y} + L$. Entonces $\mathbf{x} - \mathbf{y} \in L$, lo que implica que $(\mathbf{x} - \mathbf{y})H^\top = \mathbf{0}$, ver Teorema 3.5.2. De esta manera, $s(\mathbf{x}) = \mathbf{x}H^\top = \mathbf{y}H^\top = s(\mathbf{y})$. El recíproco se sigue de regresarnos en los argumentos anteriores. ■

Veamos como utilizar el teorema anterior para decodificar con códigos lineales.

Teorema 3.6.4 Sea L un código lineal con matriz de control de paridad H. La decodificación por distancia mínima es equivalente al proceso que consiste en decodificar la palabra recibida $\mathbf{x}$ como la palabra código $\mathbf{c} = \mathbf{x} - \mathbf{a}$, donde $\mathbf{a}$ es una palabra de menor peso en la clase lateral $\mathbf{x} + L$, o equivalentemente, $\mathbf{a}$ es una palabra de menor peso con igual síndrome que $\mathbf{x}$.

Demostración. Supongamos que se envía una palabra código de L y que se recibe el vector $\mathbf{x}$. La decodificación por distancia mínima decodifica $\mathbf{x}$ como la palabra código $\mathbf{c}$ si

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

lo cual es equivalente a que $d(\mathbf{x} - \mathbf{c}, \mathbf{0})$ sea mínima; es decir que el peso de $\mathbf{a} = \mathbf{x} - \mathbf{c}$ sea mínimo. Por lo tanto, cuando $\mathbf{c}$ varía sobre L, $\mathbf{a}$ varía sobre la clase lateral $\mathbf{x} + L$ y de ahí que la decodificación por distancia mínima requiere encontrar la palabra de

peso mínimo entre todas las palabras de una clase lateral, o equivalentemente, la palabra de menor peso entre todas las palabras con el mismo síndrome de $\mathbf{x}$. ■

El proceso de decodificación por síndrome puede describirse en términos de lo que se llama *arreglo estándar* o *arreglo Slepiano*, en honor a David Slepian[1], para L.

$\mathbf{0}$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\dots$	$\mathbf{c}_m$
$\mathbf{a}_1$	$\mathbf{c}_1 + \mathbf{a}_1$	$\mathbf{c}_2 + \mathbf{a}_1$	$\dots$	$\mathbf{c}_m + \mathbf{a}_1$
$\mathbf{a}_2$	$\mathbf{c}_1 + \mathbf{a}_2$	$\mathbf{c}_2 + \mathbf{a}_2$	$\dots$	$\mathbf{c}_m + \mathbf{a}_2$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\mathbf{a}_s$	$\mathbf{c}_1 + \mathbf{a}_s$	$\mathbf{c}_2 + \mathbf{a}_s$	$\dots$	$\mathbf{c}_m + \mathbf{a}_s$

Tabla 3.1: Arreglo estándar o arreglo Slepiano.

La primera fila del arreglo consta de todas las palabras código de L. La segunda fila consiste de la suma de cada palabra de la primera fila con una palabra $\mathbf{a}_1 \in \mathbb{F}_q$ de menor peso que no esté en L; es decir la clase lateral $\mathbf{a}_1 + L$. De manera general, la i-ésima fila del arreglo, se forma por la suma de cada palabra de la primera fila con una palabra $\mathbf{a}_i$ de menor peso que no está aún en el arreglo; es decir, contiene los elementos de la clase lateral $\mathbf{a}_i + L$. Este proceso continúa hasta que el arreglo contenga todas las palabras de $\mathbb{F}_q^n$; esto es, cuando se tengan $s = q^{n-k} - 1$ filas abajo de la primera en el arreglo. Cada elemento $\mathbf{a}_i$ se llama *líder de la clase lateral*. Como cada fila del arreglo estándar es una clase lateral de L, entonces dos palabras en $\mathbb{F}_q^n$, tienen el mismo síndrome si y sólo si están en la misma fila del arreglo, ver Teorema 3.6.3.

Teniendo en cuenta la forma en la que los líderes de la clase lateral fueron seleccionados, entonces una palabra recibida $\mathbf{x}$ que está en la j-ésima columna del arreglo, se pude escribir como $\mathbf{x} = \mathbf{a}_i + \mathbf{c}_j$, para alguna palabra $\mathbf{a}_i$ de menor peso en la clase lateral $\mathbf{x} + L$. La decodificación por distancia mínima decodifica a $\mathbf{x}$ como $\mathbf{c}_j$; es decir, la palabra código en la parte superior de la j-ésima columna.

[1](1923-2007) Matemático estadounidense, reconocido por sus aportes en teoría algebraica de códigos, probabilidad y codificación de fuentes distribuidas.

Ejemplo 3.6.5 Consideremos el $[5,2,3]$-código binario

$$L = \{00000, 10110, 01011, 11101\}.$$

Las clases laterales de L son:

$$\begin{aligned}
00000 + L &= \{00000, 10110, 01011, 11101\},\\
00001 + L &= \{00001, 10111, 01010, 11100\},\\
00010 + L &= \{00010, 10100, 01001, 11111\},\\
00100 + L &= \{00100, 10010, 01111, 11001\},\\
01000 + L &= \{01000, 11110, 00011, 10101\},\\
10000 + L &= \{10000, 00110, 11011, 01101\},\\
00101 + L &= \{00101, 10011, 01110, 11000\},\\
01100 + L &= \{00000, 11010, 00111, 10001\}.
\end{aligned}$$

Por lo tanto, el arreglo estándar de L es:

00000	10110	01011	11101
00001	10111	01010	11100
00010	10100	01001	11111
00100	10010	01111	11001
01000	11110	00011	10101
10000	00110	11011	01101
00101	10011	01110	11000
01100	11010	00111	10001

Tabla 3.2: Arreglo estándar del código L.

Asumamos que se recibe la palabra $\mathbf{x} = 11011$. Como esta palabra no está en la primera fila esto significa que hubo errores durante su transmisión. Dado que $\mathbf{x}$ está en la sexta fila y en esta fila la única palabra de menor peso es 10000, entonces $\mathbf{x}$ se decodifica como $\mathbf{c} = 01011$, la palabra que está en la primera fila en la misma columna que $\mathbf{x}$.

Por otra parte, si se recibe la palabra $\mathbf{x} = 11010$, como el líder de su clase es de peso dos se asume que se cometieron dos errores. Además, como en la octava fila hay dos palabras de peso mínimo

(01100 y 10001, con 01100 como líder de la clase lateral), según el arreglo estándar anterior $\mathbf{x}$ se decodifica como $\mathbf{c} = 10110$. Sin embargo, si hubiésemos considerado a 10001 como líder de la clase lateral, tendríamos que la octava fila del arreglo estándar sería

$$10001 \quad 00111 \quad 11010 \quad 01100$$

se tiene que $\mathbf{x}$ se decodificaría como $\mathbf{c}' = 01011$, obteniendo una decodificación diferente para la palabra $\mathbf{x}$ recibida. La razón de esta inconsistencia en la decodificación se debe al hecho de que L corrige exactamente 1-error, lo que implica que no corrige 2 errores. □

En general, en el caso de que la clase lateral de la palabra recibida tenga dos o más posibles líderes, la decodificación puede incurrir en errores dependiendo del líder que se escoja. Es decir, si $\mathbf{x} = \mathbf{c} + \mathbf{e}$ es una palabra recibida, donde $\mathbf{e}$ es el error y es un líder de alguna clase lateral, entonces $\mathbf{x} \in \mathbf{e} + L$ y por lo tanto al decodificar $\mathbf{x}$ se obtiene la palabra código correcta $\mathbf{c} = \mathbf{x} - \mathbf{e}$. Por otra parte, si $\mathbf{e}$ no es un líder de una clase lateral y se encuentra en la j-ésima fila, entonces $\mathbf{x}$ también se encuentra en la j-ésima fila y $\mathbf{x}$ se decodifica incorrectamente por $\mathbf{x} - \mathbf{a}_j \neq \mathbf{x} - \mathbf{e} = \mathbf{c}$.

La construcción del arreglo estándar es bastante dispendiosa para códigos de longitud relativamente grande. Sin embargo, no es necesario completar todo el arreglo, basta con construir una tabla de líderes de las clases laterales $\mathbf{a}_i$ con sus correspondientes síndromes, ya que cada fila está determinada por estos, puesto que cada fila del arreglo estándar tiene síndrome diferente. Este nuevo arreglo se conoce como *tabla líder-síndrome*. Por lo tanto, si $\mathbf{x}$ es una palabra recibida, se calcula su síndrome $s(\mathbf{x})$ y se busca en la tabla de consulta de síndromes el líder de la clase lateral $\mathbf{a}_i$ con igual síndrome que $\mathbf{x}$. Luego, $\mathbf{x}$ se decodifica como $\mathbf{c} = \mathbf{x} - \mathbf{a}_i$. Este proceso se conoce como *decodificación por síndrome*.

Observación 3.6.6 Para un código lineal L con distancia mínima d, todos los vectores de $\mathbb{F}_q^n$ de peso menor o igual a $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ son líderes únicos de sus clases laterales. En efecto, supongamos por contradicción que dos palabras distintas $\mathbf{x}$ e $\mathbf{y}$ tienen peso menor o igual a t y además que ambas están en la misma clase lateral de L.

Por la desigualdad triangular, tenemos que

$$\begin{aligned} wt(\mathbf{x}-\mathbf{y}) = d(\mathbf{x},\mathbf{y}) &\leq d(\mathbf{x},\mathbf{0}) + d(\mathbf{0},\mathbf{y}) \\ &= wt(\mathbf{x}) + wt(\mathbf{y}) \leq 2t \leq d-1, \end{aligned}$$

lo cual no es posible, ya que $\mathbf{x}-\mathbf{y} \in L$.

De la observación anterior, una forma rápida de construir una tabla de consulta de síndromes, para un código lineal de distancia mínima d con matriz de control de paridad, consiste en generar todos los patrones de error $\mathbf{e}$ con $wt(\mathbf{e}) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ como líderes de clase y se calculan el síndrome $s(\mathbf{e})$ para cada uno de ellos. A continuación, presentamos un ejemplo de esta situación.

Ejemplo 3.6.7 Para el código del Ejemplo 3.6.5, tenemos que su tabla de consulta de líderes es la siguiente.

Líder	Síndrome
00000	000
00001	111
00010	101
00100	001
01000	010
10000	100
00101	110
01100	011

Tabla 3.3: Tabla líder-síndrome.

De esta forma, para la palabra recibida $\mathbf{x} = 11011$, tenemos que su síndrome $s(\mathbf{x}) = 100$. Luego, como $s(10000) = 100$, entonces $\mathbf{x}$ se decodifica como $\mathbf{c} = 11011 - 10000 = 01011$. Ahora, si la palabra recibida es $\mathbf{x} = 11010$ y dado que por la Tabla 3.3 tenemos que $s(\mathbf{x}) = s(01100) = 011$, entonces $\mathbf{x}$ se decodifica como $\mathbf{x} - 01100 = 10110$. □

Ejemplo 3.6.8 Sea L el $[16,4,8]$-código ternario con matriz generadora

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

Observe que en este caso calcular el arreglo estándar o la Tabla-Síndrome de L es una tarea casi imposible, puesto que ambos tienen $3^{16-4} = 3^{12}$ filas. Sin embargo, podemos decodificar una palabra recibida $\mathbf{x} = 2001221120111221$ calculando un líder de su clase lateral. En efecto, el único líder de su clase lateral $\mathbf{x} + L$ es el vector $\mathbf{a} = 0020102000000000$. Así $\mathbf{x}$ se decodifica como

$$\mathbf{c} = \mathbf{x} - \mathbf{a} = 2011122120111221. \qquad \square$$

3.7. Operaciones sobre códigos

En esta sección mostraremos algunas técnicas usadas para obtener nuevos códigos a partir de códigos conocidos. Las definiciones tratadas aquí son en general para cualquier código, sin embargo prestaremos mayor atención a los códigos lineales.

Perforar un código

El proceso que consiste en remover una o más coordenadas de las palabras de un código se denomina *perforar un código*.

Definición 3.7.1 Sea C un código sobre $\mathbb{F}_q$ de longitud n. El *código perforado* C^* es el código que se obtiene a partir de la eliminación de la misma coordenada en todas las palabras de C. Para $1 \leq i \leq n$, con C^{*i} denotaremos al código que se obtiene al perforar el código C en la i-ésima coordenada.

Si C es un (n,M,d)-código sobre $\mathbb{F}_q$ tal que $d \geq 2$, entonces el código perforado C^* tiene parámetros

$$n^* = n-1,\ M^* = M,\ d^* = d \ \text{ o } \ d^* = d-1.$$

Ejemplo 3.7.2 Considere el código binario $C = \{000,011,101,110,111\}$, con parámetros $(3,5,1)$. Entonces el código perforado en la tercera coordenada es $C^{*3} = \{00,01,10,11\}$ y tiene parámetros $(2,4,1)$. Observemos que para el $[3,2,2]$-código lineal binario $L = \{000,011,101,110,\}$, tenemos que $L^{*3} = \{00,01,10,11\}$ es un $[2,2,1]$-código. □

Ahora, si C es un código lineal entonces el código perforado C^* también es lineal. En efecto, si $\mathbf{x}^*, \mathbf{y}^* \in C^*$ y $\lambda, \beta \in \mathbb{F}_q$, entonces existen $\mathbf{x}, \mathbf{y} \in C$ para las cuales eliminando la misma coordenada dieron origen a $\mathbf{x}^*$ e $\mathbf{y}^*$. Como $\lambda\mathbf{x} + \beta\mathbf{y} \in C$ entonces $\lambda\mathbf{x}^* + \beta\mathbf{y}^* \in C^*$, dado que esta palabra resulta de la eliminación de la misma coordenada de la palabra $\lambda\mathbf{x} + \beta\mathbf{y} \in C$. Además, claramente $C^* \neq \phi$, ya que la palabra cero está en C^*.

Algunas características de los códigos lineales perforados se resumen en el siguiente resultado. Recordemos que con $\mathbf{e}_i$ denotamos al vector coordenado de $\mathbb{F}_q^n$; esto es, el vector que tiene 1 en la i-ésima coordenada y 0 en las demás.

Teorema 3.7.3 Sean L un $[n,k,d]$-código lineal sobre $\mathbb{F}_q$ e $1 \leq i \leq n$. Entonces

1. Si $d > 1$, entonces L^{*i} es un $[n-1,k,d^*]$-código lineal donde:
   - *a*) $d^* = d - 1$, si L tiene una palabra de peso mínimo cuya i-ésima coordenada no es cero.
   - *b*) $d^* = d$, en otro caso.
2. Si $d = 1$, entonces L^{*i} es un $[n-1,k^*,d^*]$-código lineal donde:
   - *a*) $k^* = k$ y $d^* = 1$, si $\mathbf{e}_i \notin L$.
   - *b*) $k^* = k - 1$ y $d^* \geq 1$, si $k > 1$ y $\mathbf{e}_i \in L$.

Demostración. Por lo dicho anteriormente, L^{*i} es lineal y su longitud es $n - 1$.

1. Supongamos que $d > 1$. Dado que la distancia de Hamming entre cualquier par de palabras de L es mayor o igual que 2, se tiene que $|L^{*i}| = |L| = q^k$ o equivalentemente que $\dim(L^{*i}) = k$. Ahora observemos que $L = L_1 \cup L_2$, donde L_1 es el conjunto de todas las palabras de L cuya i-ésima coordenada es distinta de cero; mientras que L_2 es su complemento; es decir todas las palabras con i-ésima coorde-

nada igual a cero. Entonces, tenemos que $L^* = L_1^* \cup L_2^*$. Ahora, si existe $\mathbf{c} \in L_1$ tal que $d = wt(\mathbf{c})$, entonces como $wt(\mathbf{c}^*) = wt(\mathbf{c}) - 1 \leq wt(\mathbf{x}) - 1 \leq wt(\mathbf{x}^*)$, para cualquier $\mathbf{x} \in L$. Entonces, $d(L^{*i}) = wt(\mathbf{c}^*) = d - 1$. Ahora, si no existe tal $\mathbf{c} \in L_1$, entonces el peso mínimo de L se alcanza únicamente en palabras que pertenecen a L_2; es decir si $d = wt(\mathbf{b})$ con $\mathbf{b} \in L$, entonces $\mathbf{b} \in L_2$. Así $wt(\mathbf{b}) \leq wt(\mathbf{x})$ para todo $\mathbf{x} \in L$. En el caso en que $wt(\mathbf{b}) = wt(\mathbf{y})$ para alguna $\mathbf{y} \in L$, entonces tenemos que $\mathbf{y} \in L_2$ y así $d = wt(\mathbf{b}^*) = wt(\mathbf{y}^*)$. Por otro lado, si $wt(\mathbf{b}) < wt(\mathbf{y})$ para alguna $\mathbf{y} \in L$, entonces $wt(\mathbf{b}^*) \leq wt(\mathbf{y}) - 1 \leq wt(\mathbf{y}^*)$. De esta forma, $d = wt(\mathbf{b}^*) \leq wt(\mathbf{x}^*)$ para toda $\mathbf{x} \in L$. En conclusión, $d^* = wt(\mathbf{b}^*) = d$.

2. Supongamos que $d = 1$, nuevamente debemos estudiar dos casos. Si $\mathbf{e}_i \notin L$, esto implica que existe $\mathbf{e}_t \in L$ para algún $t \neq i$. Dado que $\mathbf{0} \neq \mathbf{e}_t^* \in \mathbb{F}_q^{n-1}$, entonces $d(L^{*i}) = 1$. Además, también se tiene que $|L^{*i}| = |L|$. De hecho, si existieran dos palabras $\mathbf{w}, \mathbf{v} \in L$ tales que $\mathbf{w}^* = \mathbf{v}^*$, entonces $\mathbf{w} - \mathbf{v} = \beta \mathbf{e}_i$, para algún $\beta \in \mathbb{F}_q$; lo que sería una contradicción.

   Ahora asumamos que $\mathbf{e}_i \in L$. Como $\mathbf{e}_i^* = \mathbf{0} \in \mathbb{F}_q^{n-1}$, esto demuestra que $|L^{*i}| < |L|$. Ahora como existe una base con k elementos de L que contiene a $\mathbf{e}_i$ digamos: $B = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k-1}, \mathbf{e}_i\}$, donde $\mathbf{v}_s = (v_{s1}, v_{s2}, \ldots, v_{sn})$ para $s \in \{1, \ldots, k-1\}$. Observe que se puede demostrar que para $s \neq m$, se debe tener que $\mathbf{v}_s^* \neq \mathbf{v}_m^*$. En efecto, puesto que de lo contrario tendríamos que $v_{si} \neq v_{mi}$ y así $\mathbf{v}_s - \mathbf{v}_m$ sería un múltiplo de $\mathbf{e}_i$; lo que no se puede dar.

   Entonces al perforar B en la i-ésima coordenada quedan $k - 1$ vectores diferentes, los que conforman una base para L^{*i}. En efecto, si existieran escalares $\lambda_1, \ldots, \lambda_{k-1} \in \mathbb{F}_q$ tales que

$$\lambda_1 \mathbf{v}_1^* + \cdots + \lambda_{k-1} \mathbf{v}_{k-1}^* = \mathbf{0} \in \mathbb{F}_q^{n-1}.$$

   Entonces esta expresión se podría extender nuevamente a $\mathbb{F}_q^n$ para obtener una expresión del tipo

$$\lambda_1 \mathbf{v}_1 + \cdots + \lambda_{k-1} \mathbf{v}_{k-1} + \alpha \mathbf{e}_i = \mathbf{0} \in \mathbb{F}_q^n, \tag{3.12}$$

   donde $\alpha = -\sum_{s=1}^{k-1} \lambda_s v_{si}$.

Dado que B es linealmente independiente, de (3.12) tenemos que $\lambda_i = 0$ para todo i. Así, la dimensión de L^{*i} es $k-1$. Finalmente, es claro que $d^* \geq 1$. ■

Ejemplo 3.7.4 Sea L el $[6,3,2]$-código lineal binario con matriz generadora

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

Sean L^{*1} y L^{*5}, los códigos obtenidos por la eliminación de la primera y la quinta coordenada de L, respectivamente. Las matrices generadoras para L^{*1} y L^{*5} son

$$G_1^* = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ y } G_5^* = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

respectivamente. Además, podemos verificar que L^{*1} es un $[5,3,1]$-código lineal, mientras que L^{*5} es un $[5,3,2]$-código lineal. □

De manera general, un código L puede ser perforado en un conjunto T de coordenadas, por eliminación de las componentes etiquetadas por T en todas las palabras de L. Si T tiene tamaño t, el código perforado denotado con L^{T^*}, es un $[n-t, k^*, d^*]$-código lineal, con $k^* \geq k-t$ y $d^* \geq d-t$. La prueba se realiza aplicando inductivamente el Teorema 3.7.3.

Extender un código

A continuación, presentamos otra operación que podríamos decir es contraria a la perforación; esto en el sentido que aumentamos la longitud del código.

Definición 3.7.5 El proceso que consiste en añadir una o más coordenadas adicionales a las palabras de un código se denomina extender el código. Una manera de extender el código consiste en agregar una o más coordenadas de control de paridad.

Si C es un (n,M,d)-código sobre $\mathbb{F}_q$, el *código extendido* $\widehat{C}$ se define por

$$\widehat{C} = \left\{ c_1c_2\cdots c_nc_{n+1} : c_1c_2\cdots c_n \in C \text{ y } \sum_{k=1}^{n+1} c_k = 0 \right\}.$$

Entonces si C es un (n,M,d)-código, entonces podemos demostrar que $\widehat{C}$ es un $(\widehat{n},\widehat{M},\widehat{d})$-código, donde

$$\widehat{n} = n+1,\ \widehat{M} = M,\ \widehat{d} = d \text{ o } \widehat{d} = d+1.$$

En el siguiente resultado, mostramos esta afirmación para el caso de códigos lineales.

Teorema 3.7.6 Si L es $[n,k,d]$-código lineal sobre $\mathbb{F}_q$, entonces $\widehat{L}$ es un $[n+1,k,\widehat{d}]$-código lineal sobre $\mathbb{F}_q$, donde $\widehat{d} = d$ o $\widehat{d} = d+1$. Adicionalmente, se satisfacen las siguientes propiedades

1. Si $G = \begin{pmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_k \end{pmatrix}$ es una matriz generadora de L, entonces $\widehat{G} = \begin{pmatrix} \widehat{\mathbf{g}_1} \\ \vdots \\ \widehat{\mathbf{g}_k} \end{pmatrix}$ es una matriz generadora de $\widehat{L}$.
2. Si H es una matriz de control de paridad de L, entonces

$$\widehat{H} = \left(\begin{array}{ccc|c} & H & & 0 \\ & & & \vdots \\ & & & 0 \\ \hline 1 & \cdots & 1 & 1 \end{array}\right)$$

es una matriz de control de paridad de $\widehat{L}$.

Demostración. Dado que al extender las q^k palabras de L obtenemos q^k palabras diferentes, entonces es claro que $\dim(\widehat{L}) = k+1$.

Sean $\mathbf{x},\mathbf{y} \in L$ y $\lambda,\beta \in \mathbb{F}_q$. Luego si $\mathbf{z} = \lambda\mathbf{x} + \beta\mathbf{y}$, entonces $\widehat{\mathbf{z}} = (\lambda x_1 + \beta y_1, \ldots, \lambda x_{n+1} + \beta y_{n+1}) \in \widehat{L}$, puesto que

$$\sum_{i=1}^{n+1} (\lambda x_i + \beta y_i) = \lambda \sum_{i=1}^{n+1} x_i + \beta \sum_{i=1}^{n+1} y_i = \lambda 0 + \beta 0 = 0.$$

Usando esta idea, podemos probar que si $\mathbf{g}_1, \ldots, \mathbf{g}_k$ es una base de L, entonces $\widehat{\mathbf{g}_1}, \ldots, \widehat{\mathbf{g}_k}$ es una base para $\widehat{L}$.

Ahora, supongamos que H es una matriz de control de paridad para L. Observemos que para $\mathbf{x} = (x_1, \ldots, x_n, x_{n+1}) \in \mathbb{F}_q^{n+1}$ se cumple que

$$\begin{aligned} \mathbf{x}\widehat{H}^\top &= \mathbf{x} \left(\begin{array}{ccc|c} & H^\top & & 1 \\ & & & \vdots \\ & & & 1 \\ \hline 0 & \ldots & 0 & 1 \end{array}\right) \\ &= \left((x_1, \ldots, x_n) H^\top, \sum_{i=1}^{n+1} x_i \right). \end{aligned}$$

Así para cualquier fila $\mathbf{g}_i$ de G tenemos que $\widehat{\mathbf{g}_i}\widehat{H} = (\mathbf{0}, 0)$. Esto demuestra que $\widehat{H}\widehat{G}^\top = \mathbf{O}$. Además, como las filas de H son linealmente independientes, entonces las filas de $\widehat{H}$ también lo son. Así de la Proposición 3.5.3, tenemos que $\widehat{H}$ es una matriz de control de paridad para $\widehat{L}$. ■

Note que en el caso binario, el código extendido contiene sólo palabras de peso par. Por lo tanto, si la distancia de un código binario L es impar, entonces la distancia del código extendido es par. Si la distancia del código L es par, entonces la distancia del código extendido también es par e igual a la distancia de L. Sin embargo, esto no es cierto si el código inicial no es binario.

Al extender un código, pueden suceder dos situaciones respecto a su distancia. Si la distancia del código extendido es igual a la del código inicial, entonces no es conveniente extender el código, ya que la capacidad de corrección de errores permanece igual. Por otra parte, si la distancia del código extendido se incrementa, entonces su capacidad de detección aumenta, aunque es posible que su capacidad de corrección sea igual que la del código inicial.

Para códigos lineales binarios, los procesos de perforación y extensión de un código pueden utilizarse para probar el siguiente resultado.

Teorema 3.7.7 Existe un $(n, M, 2t+1)$-código binario si y sólo si existe un $(n+1, M, 2t+2)$-código binario.

Demostración. Supongamos que C es un $(n, M, 2t+1)$-código binario y que $\widehat{C}$ es el código obtenido de C por adición de una coordenada de control de paridad. Por lo mencionado anteriormente cada $\widehat{\mathbf{x}} \in \widehat{C}$ tiene peso par. Por la segunda parte del Lema 2.5.3, para cada $\widehat{\mathbf{x}}, \widehat{\mathbf{y}} \in \widehat{C}$ se tiene que

$$d(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}) = wt(\widehat{\mathbf{x}}) + wt(\widehat{\mathbf{y}}) - 2wt(\widehat{\mathbf{x}} \cap \widehat{\mathbf{y}}).$$

Luego, $d(\widehat{C})$ es par y del Teorema 3.7.6 concluimos que $d(\widehat{C}) = 2t+2$.

Recíprocamente, si C es un $(n+1, M, 2t+2)$-código binario, entonces existen $\mathbf{x}, \mathbf{y} \in C$ tales que $d(\mathbf{x}, \mathbf{y}) = 2t+2$. Sea i una de las $2t+1$ coordenadas en las que $\mathbf{x}$ e $\mathbf{y}$ no coinciden y consideremos el código C^{*i}. Sean $\mathbf{x}^*, \mathbf{y}^* \in C^{*i}$, que se obtienen de $\mathbf{x}$ e $\mathbf{y}$, respectivamente. Entonces $d(\mathbf{x}^*, \mathbf{y}^*) = 2t+1 = d(C^{*i})$. Por lo tanto, C^{*i} es un $(n, M, 2t+1)$-código binario. ■

Expurgar y aumentar un código

El proceso que consiste en reducir el tamaño de un código mediante la eliminación de algunas palabras código se denomina *expurgar el código*. Por otro lado, el proceso opuesto a expurgar un código se denomina *aumentar el código*. A continuación, presentamos algunos ejemplos de estos dos métodos.

Ejemplo 3.7.8 Suponga que L es un (n, M, d)-código lineal binario. Si L contiene al menos una palabra de peso impar, entonces exactamente la mitad de las palabras de L tienen peso impar. En efecto, sean L_1 y L_2, los subconjuntos de L formados por las palabras de L de peso impar y peso par, respectivamente. Si $\mathbf{x} \in L_1$, entonces dado que L es un código lineal y el Lema 2.5.3, se tiene que $\mathbf{x} + \mathbf{c} \in L_2$ para toda $\mathbf{c} \in L_1$. Por lo tanto, $|\mathbf{x} + L_1| \leq |L_2|$. Similarmente, se tiene que $|\mathbf{x} + L_2| \leq |L_1|$. Dado que $|L_1| = |\mathbf{x} + L_1|$ y $|L_2| = |\mathbf{x} + L_2|$, por lo tanto $|L_1| = |L_2| = M/2$.

En consecuencia, si no tenemos en cuenta las palabras de peso impar, se tiene un $(n, M/2, d')$-código, donde $d' \geq d$. Todas las palabras de este código son de peso par y por supuesto tiene distancia mínima par; adicionalmente si d es impar, entonces $d' > d$. □

Para un código binario C, una manera común de aumentar el código C consiste en añadir el *complemento* $\mathbf{x}^c$ de cada palabra binaria $\mathbf{x} \in C$. Aquí, $\mathbf{x}^c$ es la palabra que se obtiene de $\mathbf{x}$ permutando sus coordenadas que son iguales a ceros por unos y los unos por ceros. El conjunto formado por los complementos de todos los elementos de C se denota por C^c.

Ejemplo 3.7.9 El conjunto

$$C = \{00000, 00101, 10001, 11011, 10100, 11110, 01010, 01111\},$$

es un $[5,3,2]$-código lineal binario. Entonces

$$C^c = \{11111, 11010, 01110, 00100, 01011, 00001, 10101, 10000\},$$

es un $(5,8,2)$-código. Note que C^c no es un código lineal, dado que no contiene la palabra código $\mathbf{0}$. □

Lema 3.7.10 Si $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, entonces $d(\mathbf{x}, \mathbf{y}^c) = n - d(\mathbf{x}, \mathbf{y})$.

Demostración. Sean $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$. Dado que las coordenadas de $\mathbf{x}$ e $\mathbf{y}$ son binarios, tenemos que $d(\mathbf{x}, \mathbf{y}^c)$ es igual al número de posiciones en las cuales $\mathbf{x}$ y $\mathbf{y}$ coinciden; es decir, a $d(\mathbf{x}, \mathbf{y}^c) = n - d(\mathbf{x}, \mathbf{y})$, entonces $d(\mathbf{x}, \mathbf{y}^c) = n - d(\mathbf{x}, \mathbf{y})$. ■

Usando el resultado anterior no es difícil ver que para $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, se tiene que $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{x}^c, \mathbf{y}^c)$. En consecuencia si C es un código lineal binario se tiene que, $d(C) = d(C^c)$. Además, se puede probar que C y C^c son códigos equivalentes.

El siguiente resultado permite determinar la distancia para el código binario $C \cup C^c$.

Teorema 3.7.11 Sea C un (n, M, d)-código binario. Entonces

$$d(C \cup C^c) = \min\{d, n - d_{\text{máx}}\},$$

donde $d_{\text{máx}}$ es la máxima distancia entre las palabras de C.

Demostración. Como $d(C) = d(C^c)$, entonces

$$d(C \cup C^c) = \min\left\{ d(C), \min_{\mathbf{x}\in C, \mathbf{y}\in C^c} d(\mathbf{x},\mathbf{y}) \right\}.$$

Por el Lema 3.7.10

$$\begin{aligned} \min_{\mathbf{x}\in C, \mathbf{y}\in C^c} d(\mathbf{x},\mathbf{y}) &= \min_{\mathbf{x}\in C, \mathbf{y}\in C} d(\mathbf{x},\mathbf{y}^c) \\ &= \min_{\mathbf{x}\in C, \mathbf{y}\in C} \{n - d(\mathbf{x},\mathbf{y})\} = n - \max_{\mathbf{x}\in C, \mathbf{y}\in C} d(\mathbf{x},\mathbf{y}). \blacksquare \end{aligned}$$

Para el caso de códigos lineales binarios se tiene el siguiente resultado.

Teorema 3.7.12 Sea L un $[n,k,d]$-código lineal binario que no contiene la palabra código $\mathbf{1} = 11\cdots1 \in \mathbb{F}_2^n$. Entonces el código $L \cup L^c$ es un $[n,k+1,d']$-código lineal binario, donde

$$d' = \min\{d, n - w_{\text{máx}}\}$$

y $w_{\text{máx}}$ es el máximo peso entre todos los pesos de las palabras de L.

Demostración. Dado que $\mathbf{0} \in L \subset (L \cup L^c)$, tenemos que $L \cup L^c \neq \phi$. Para probar que $L \cup L^c$ es lineal es suficiente mostrar que $\mathbf{x} + \mathbf{y} \in L \cup L^c$ para toda $\mathbf{x},\mathbf{y} \in L \cup L^c$. Consideremos los siguientes casos:

- Si $\mathbf{x},\mathbf{y} \in L$, entonces por la linealidad de L se tiene que $\mathbf{x} + \mathbf{y} \in L \subset (L \cup L^c)$.
- Si $\mathbf{x},\mathbf{y} \in L^c$, entonces $\mathbf{x}+\mathbf{1}, \mathbf{y}+\mathbf{1} \in L$, y así $\mathbf{x}+\mathbf{y} = (\mathbf{x}+\mathbf{1}) + (\mathbf{y}+\mathbf{1}) \in L \subset (L \cup L^c)$.
- Si $\mathbf{x} \in L$ e $\mathbf{y} \in L^c$, entonces $\mathbf{y} + 1 \in L$ y por la linealidad de L se tiene que $\mathbf{z} = \mathbf{x} + \mathbf{y} + \mathbf{1} \in L$. Por lo tanto, $\mathbf{x} + \mathbf{y} = (\mathbf{x} + \mathbf{y} + \mathbf{1}) + \mathbf{1} = \mathbf{z} + \mathbf{1} \in L^c \subset (L \cup L^c)$.
- La prueba para $\mathbf{x} \in L^c$, $\mathbf{y} \in L$ se realiza de manera similar al ítem anterior.

Luego $L \cup L^c$ es lineal. Su longitud es claramente n. Por otro lado, mostremos que $L \cap L^c = \phi$. Supongamos que existe $\mathbf{x} \in L \cap L^c$, entonces $\mathbf{x} \in L$ y $\mathbf{x} \in L^c$. Entonces $\mathbf{x} + \mathbf{1} \in L$ y así $\mathbf{x} + (\mathbf{x} + \mathbf{1}) =$

$\mathbf{1} \in L$, lo que contradice la hipótesis. Como $2^k = |L| = |L^c|$ y $L \cap L^c = \phi$, entonces $|L \cup L^c| = 2^{k+1}$.
Finalmente, el Teorema 3.7.11 afirma que $d(L \cup L^c) = \text{mín}\{d, n - d_{\text{máx}}\}$. Sin embargo, como $L \cup L^c$ es lineal, entonces $d_{\text{máx}} = w_{\text{máx}}$. ■

A continuación, presentamos una aplicación del resultado anterior.

Ejemplo 3.7.13 El conjunto $L = \{0000, 0101, 1100, 1001\}$ es un $[4,2,2]$-código lineal binario, entonces tenemos que $L^c = \{1111, 1010, 1100, 0110\}$. Por lo tanto, $L \cup L^c = \{0000, 0101, 1100, 1001, 1111, 1010, 1100, 0110\}$ es un $[4,3,2]$-código lineal binario. □

Definición 3.7.14 El proceso que conserva aquellas palabras de un código que tienen un símbolo común en una posición dada (por ejemplo, 0 en la primera posición) y luego perfora en esa coordenada se denomina *acortar un código*. De manera general, si C es un (n, M, d)-código sobre $\mathbb{F}_q$ y T es cualquier conjunto de t coordenadas, denotemos por $C(T)$ al conjunto de todas las palabras que tienen coordenada igual a cero en T. Perforando $C(T)$ sobre las t coordenadas se obtiene un código sobre $\mathbb{F}_q$ de longitud $n - t$ llamado el *código acortado* sobre T y el cual denotaremos con C_T. El código acortado obtenido de las palabras del código C con una s en la i-ésima coordenada, se denomina *sección transversal* $x_i = s$ (cross-section). Para este caso, el código acortado tiene longitud $n - 1$ y distancia mínima al menos d.

Ejemplo 3.7.15 Sea L el $[6,3,1]$-código lineal binario con matriz generadora

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

Sea $T = \{4,5\}$, entonces primero seleccionamos las palabras código de L que tienen su cuarta y quinta coordenadas iguales a 0; esto es $L(T) = \{(0,0,0,0,0,0), (1,1,0,0,0,0), (1,1,1,0,0,0), (0,0,1,0,0,0)\}$.

En consecuencia, el código acortado $L_T = \{(0,0,0,0),(1,1,0,0),(1,1,1,0),(0,0,1,0)\}$. □

Observación 3.7.16 No es difícil probar que para un código lineal L, el código acortado L_T sobre un conjunto de coordenadas T, también es un código lineal (ver Ejercicio 15).

Teorema 3.7.17 Si C es un (n,M,d)-código lineal binario, entonces la sección transversal $x_i = 0$ es un $(n-1,M/2,d')$-código lineal binario, donde $d' \geq d$.

Demostración. Sea L_1 el subcódigo de L conformados por las palabras de L cuya i-ésima componente es igual a cero y $L_2 = L \setminus L_1$. Sea $\mathbf{y}$ un elemento fijo de L_2, no es difícil ver que $\mathbf{y} + L_2 \subseteq L_1$. Probemos la otra inclusión, en efecto, sea $\mathbf{x} \in L_1$, entonces $\mathbf{x} + \mathbf{y} \in L_2$; es decir, $\mathbf{x} + \mathbf{y} = \mathbf{z}$, para algún $\mathbf{z} \in L_2$. De esta manera, $\mathbf{x} = \mathbf{y} + \mathbf{z} \in \mathbf{y} + L_2$.

En conclusión, $\mathbf{y} + L_2 = L_1$, y por lo tanto

$$|L_1| = |\mathbf{y} + L_2| = |L_2|.$$

Como L_1 y L_2 son disjuntos entonces $|L_1| + |L_2| = M$, lo cual implica que $|L_1| = \frac{M}{2}$. Luego al eliminar la posición x_i de todas las palabras de L_1 se obtiene que la sección transversal $x_i = 0$ es el código L_1^*, el cual es un $(n-1,M/2,d')$-código lineal binario. Ahora, dado que L_1^*, es lineal y L_1 es un subcódigo de L se tiene que

$$d = wt(L) \leq wt(L_1) = wt(L_1^*) = d'.$$ ■

Ejemplo 3.7.18 Para el $[5,2,2]$-código lineal binario

$$L = \{00000, 11000, 00111, 11111\},$$

tenemos que al considerar la sección transversal $x_2 = 0$ se obtiene el $[4,1,3]$-código lineal binario $L' = \{0000, 0111\}$. □

El siguiente resultado relaciona los códigos lineales perforados y acortados con sus duales.

Teorema 3.7.19 Sea L un $[n,k,d]$-código lineal sobre $\mathbb{F}_q$. Sea T un conjunto de t coordenadas. Entonces $(L^{\perp})_T = (L^{T^*})^{\perp}$ y $(L^{\perp})^{T^*} = (L_T)^{\perp}$.

Demostración. La linealidad de estos códigos está garantizada por el Teorema 3.7.3 y la Observación 3.7.16, de ahí que tenga sentido hablar de sus duales. Sean $\mathbf{c} \in L^{\perp}(T)$ y $\mathbf{c}^*$ la palabra que se obtiene de $\mathbf{c}$ cuyas coordenadas sobre T han sido removidas; es decir $\mathbf{c}^* \in (L^{\perp})_T$. Si $\mathbf{x} \in L$, entonces $0 = \mathbf{x} \cdot \mathbf{c} = \mathbf{x}^* \cdot \mathbf{c}^*$, donde $\mathbf{x}^*$ es la palabra $\mathbf{x}$ perforada sobre T. Luego, $(L^{\perp})_T \subseteq (L^{T^*})^{\perp}$. Por otro lado, cualquier palabra $\mathbf{c} \in (L^T)^{\perp}$ puede extenderse a una palabra $\widehat{\mathbf{c}}$ añadiendo ceros sobre T. Si $\mathbf{x} \in L$, perforando $\mathbf{x}$ sobre T se obtiene $\mathbf{x}^*$. Como $0 = \mathbf{x}^* \cdot \mathbf{c} = \mathbf{x} \cdot \widehat{\mathbf{c}}$, entonces $\mathbf{c} \in (L^{\perp})_T$. Luego $(L^{T^*})^{\perp} \subseteq (L^{\perp})_T$ y así $(L^{\perp})_T = (L^{T^*})^{\perp}$.

Para la otra parte, reemplazando L por $L^{\perp}$ en la anterior igualdad se obtiene

$$\begin{aligned} ((L^{\perp})^{\perp})_T &= ((L^{\perp})^{T^*})^{\perp} \\ L_T &= ((L^{\perp})^{T^*})^{\perp}. \end{aligned}$$

Como estos códigos son iguales, entonces sus duales también lo son; esto es

$$(L_T)^{\perp} = (((L^{\perp})^{T^*})^{\perp})^{\perp} = (L^{\perp})^{T^*}. \qquad \blacksquare$$

Ejemplo 3.7.20 Sea L el $[6,3,1]$-código lineal binario con matriz generadora

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

Tenemos que $L^{\perp}$ es un $[6,3,2]$-código lineal binario con matriz generadora

$$G^{\perp} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Si $T = \{4,5\}$, entonces matrices generadoras para el código acortado L_T y el código perforado L^{T^*} son

$$G_T = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \text{y} \quad G^{T^*} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

Acortando y perforando el código $L^\perp$ sobre T, se obtienen respectivamente, los códigos $(L^\perp)_T$ y $(L^\perp)^T$, los cuales tienen como matrices generadoras

$$(G^\perp)_T = (1 \;\; 1 \;\; 0 \;\; 1) \quad \text{y} \quad (G^\perp)^{T^*} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

respectivamente. De las matrices G_T y G^{T^*}, se tienen matrices generadoras para L_T y L^{T^*}, respectivamente, las cuales son

$$(G_T)^\perp = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{y} \quad (G^{T^*})^\perp = (1 \;\; 1 \;\; 0 \;\; 1).$$

Es decir, $(L^\perp)_T = (L^{T^*})^\perp$ y $(L^\perp)^{T^*} = (L_T)^\perp$. □

Suma directa

A continuación estudiamos dos operaciones con las que podemos combinar dos códigos para generar un nuevo código.

Definición 3.7.21 Sean C_1 un (n_1, M_1, d_1)-código y C_2 un (n_2, M_2, d_2)-código, ambos sobre $\mathbb{F}_q$, entonces la *suma directa* de C_1 y C_2 es el código

$$C_1 \oplus C_2 = \{(\mathbf{c}, \mathbf{d}) : \mathbf{c} \in C_1, \mathbf{d} \in C_2\}.$$

En general, se tiene que $C_1 \oplus C_2$ es un $(n_1 + n_2, M_1 M_2, d')$-código sobre $\mathbb{F}_q$, donde $d' = \min\{d_1, d_2\}$. Para el caso lineal, estas afirmaciones se enuncian como un teorema.

Teorema 3.7.22 Sea L_i un $[n_i, k_i, d_i]$-código lineal sobre $\mathbb{F}_q$, para $i = 1,2$. La suma directa de L_1 y L_2 definida por

$$L_1 \oplus L_2 = \{(\mathbf{c}_1, \mathbf{c}_2) : \mathbf{c}_1 \in L_1, \mathbf{c}_2 \in L_2\}.$$

es un $[n_1+n_2, k_1+k_2, \text{mín}\{d_1,d_2\}]$-código lineal sobre $\mathbb{F}_q$.

Demostración. Sean $\lambda,\beta \in \mathbb{F}_q$, $\mathbf{x}=(\mathbf{x}_1,\mathbf{x}_2)$, $\mathbf{y}=(\mathbf{y}_1,\mathbf{y}_2)$ elementos de $L_1 \oplus L_2$, con $\mathbf{x}_1,\mathbf{y}_1 \in L_1$ y $\mathbf{x}_2,\mathbf{y}_2 \in L_2$. Entonces

$$\lambda\mathbf{x}-\beta\mathbf{y}=\lambda(\mathbf{x}_1,\mathbf{x}_2)-\beta(\mathbf{y}_1,\mathbf{y}_2)=(\lambda\mathbf{x}_1-\beta\mathbf{y}_1,\lambda\mathbf{x}_2-\beta\mathbf{y}_2).$$

Como $\lambda\mathbf{x}_i-\beta\mathbf{y}_i \in L_i$, para $i=1,2$, entonces $\lambda\mathbf{x}-\beta\mathbf{y} \in L_1 \oplus L_2$. Además, $L_1 \oplus L_2 \neq \phi$, ya que $\mathbf{0} \in L_1 \oplus L_2$. Luego $L_1 \oplus L_2$ es un $[n,k,d]$-código lineal.
La longitud de $L_1 \oplus L_2$ es claramente $n=n_1+n_2$. Como el tamaño de $L_1 \oplus L_2$ es igual al producto de los tamaños de L_1 y L_2 se tiene que

$$k=\dim(L_1 \oplus L_2)=\log_q(|L_1 \oplus L_2|)=\log_q(|L_1|\cdot|L_2|)=k_1+k_2.$$

Para la distancia mínima, supongamos sin pérdida de generalidad que $d_1 \leq d_2$. Sea $\mathbf{u} \in L_1$ tal que $wt(\mathbf{u})=d_1$. Por definición $(\mathbf{u},\mathbf{0}) \in L_1 \oplus L_2$ y $wt((\mathbf{u},\mathbf{0}))=d_1$, por lo tanto

$$d(L_1 \oplus L_2) \leq wt((\mathbf{u},\mathbf{0}))=d_1.$$

Por otro lado, para cualquier palabra no cero $(\mathbf{c}_1,\mathbf{c}_2) \in L_1 \oplus L_2$, con $\mathbf{c}_1 \in L_1$ y $\mathbf{c}_2 \in L_2$ se tiene que $\mathbf{c}_1 \neq \mathbf{0}$ o $\mathbf{c}_2 \neq \mathbf{0}$. Por tanto, tenemos que

$$wt((\mathbf{c}_1,\mathbf{c}_2))=wt(\mathbf{c}_1)+wt(\mathbf{c}_2) \geq d_1.$$

En particular, para $(\mathbf{c}_1,\mathbf{c}_2) \in L_1 \oplus L_2$ tal que $d(L_1 \oplus L_2)=wt((\mathbf{c}_1,\mathbf{c}_2))$, se tiene $wt(\mathbf{c}_1,\mathbf{c}_2)=d(L_1 \oplus L_2) \geq d_1$. Por lo tanto, $d(L_1 \oplus L_2)=\text{mín}\{d_1,d_2\}$. ■

Observación 3.7.23 Si L_i para $i=1,2$ es un código lineal que tiene como matriz generadora a G_i y como matriz de control de paridad a H_i, entonces

$$G_1 \oplus G_2=\begin{pmatrix} G_1 & \mathbf{0} \\ \mathbf{0} & G_2 \end{pmatrix} \text{ y}$$

$$H_1 \oplus H_2=\begin{pmatrix} H_1 & \mathbf{0} \\ \mathbf{0} & H_2 \end{pmatrix},$$

son matrices generadora y de control para $L_1 \oplus L_2$, respectivamente.

Ejemplo 3.7.24 Tenemos que $L_1 = \{000,110,101,011\}$ es un $[3,2,2]$-código lineal binario y $L_2 = \{0000,1111\}$ es un $[4,1,4]$-código lineal binario. Entonces

$$L_1 \oplus L_2 = \{0000000,1100000,1010000,0110000,0001111,1101111,\\ 1011111,0111111\},$$

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

Dado que la distancia mínima de un código obtenido por suma directa no excede la distancia mínima de los códigos que lo forman, este código es generalmente de poco uso en aplicaciones, por lo que sólo tiene interés teórico.

La última construcción que veremos consiste en combinar dos códigos de la misma longitud para formar un tercer código cuya longitud es igual al doble de la longitud de los códigos dados.

Construcción $(\mathbf{u}+\mathbf{v})$

Definición 3.7.25 Sean C_1 un (n,M_1,d_1)-código y C_2 un (n,M_2,d_2)-código, ambos sobre $\mathbb{F}_q$. Entonces se define el código C por

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

En general, se puede demostrar que el código generado de esta forma es un $(2n, M_1M_2, \min\{2d_1, d_2\})$-código. Nuevamente, presentamos la demostración de este resultado para códigos lineales.

Teorema 3.7.26 Sea L_i un $[n,k_i,d_i]$-código lineal sobre $\mathbb{F}_q$, para $i = 1,2$. El código L definido por

$$L = \{(\mathbf{u},(\mathbf{u}+\mathbf{v})) : \mathbf{u} \in L_1, \mathbf{v} \in L_2\},$$

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

Demostración. Dado que $(\mathbf{0},\mathbf{0}+\mathbf{0}) \in L$, entonces $L \neq \phi$. Sean $(\mathbf{u}_1,\mathbf{u}_1+\mathbf{v}_1),(\mathbf{u}_2,\mathbf{u}_2+\mathbf{v}_2) \in L$ y $\alpha,\beta \in \mathbb{F}_q$, entonces

$$\begin{aligned}\alpha(\mathbf{u}_1,\mathbf{u}_1+\mathbf{v}_1)+\beta(\mathbf{u}_2,&\mathbf{u}_2+\mathbf{v}_2)\\ &=(\alpha\mathbf{u}_1,\alpha\mathbf{u}_1+\alpha\mathbf{v}_1)+(\beta\mathbf{u}_2,\beta\mathbf{u}_2+\beta\mathbf{v}_2)\\ &=(\alpha\mathbf{u}_1+\beta\mathbf{u}_2,\alpha\mathbf{u}_1+\alpha\mathbf{v}_1+\beta\mathbf{u}_2+\beta\mathbf{v}_2)\\ &=(\alpha\mathbf{u}_1+\beta\mathbf{u}_2,\alpha\mathbf{u}_1+\beta\mathbf{u}_2+\alpha\mathbf{v}_1+\beta\mathbf{v}_2)\\ &=(\mathbf{u},\mathbf{u}+\mathbf{v}),\end{aligned}$$

donde $\mathbf{u}=\alpha\mathbf{u}_1+\beta\mathbf{u}_2\in L_1$ y $\mathbf{v}=\alpha\mathbf{v}_1+\beta\mathbf{v}_2\in L_2$, ya que L_1 y L_2 son lineales. Esto prueba que L es lineal. Claramente la longitud de L es $2n$ y el tamaño es el producto del tamaño de L_1 y el tamaño de L_2; esto implica que, la dimensión de L es $k=k_1+k_2$. Por otro lado, para cualquier palabra no cero $(\mathbf{c}_1,\mathbf{c}_1+\mathbf{c}_2)\in L$, con $\mathbf{c}_1\in L_1$ y $\mathbf{c}_2\in L_2$, se tiene que $\mathbf{c}_1\neq\mathbf{0}$ o $\mathbf{c}_2\neq\mathbf{0}$.

- Caso 1. Si $\mathbf{c}_2=\mathbf{0}$, entonces $\mathbf{c}_1\neq\mathbf{0}$. Luego

$$\begin{aligned}wt((\mathbf{c}_1,\mathbf{c}_1+\mathbf{c}_2))&=wt((\mathbf{c}_1,\mathbf{c}_1))=2wt(\mathbf{c}_1)\\ &\geq 2d_1\geq \text{mín}\{2d_1,d_2\}.\end{aligned}$$

- Caso 2. Si $\mathbf{c}_2\neq\mathbf{0}$, entonces por el Lema 2.5.4

$$\begin{aligned}wt((\mathbf{c}_1,\mathbf{c}_1+\mathbf{c}_2))&=wt(\mathbf{c}_1)+wt(\mathbf{c}_1+\mathbf{c}_2)\\ &\geq wt(\mathbf{c}_1)+(wt(\mathbf{c}_2)-wt(\mathbf{c}_1))\\ &=wt(\mathbf{c}_2)\geq d_2\geq \text{mín}\{2d_1,d_2\}.\end{aligned}$$

De esta manera, el peso de cualquier palabra de L es mayor o igual que $\text{mín}\{2d_1,d_2\}$, entonces $d(L)\geq\text{mín}\{2d_1,d_2\}$. Del otro lado, sean $\mathbf{x}\in L_1$, $\mathbf{y}\in L_2$ tales que $wt(\mathbf{x})=d_1$ y $wt(\mathbf{y})=d_2$. Notemos que $(\mathbf{x},\mathbf{x}),(\mathbf{0},\mathbf{y})\in L$, entonces $d(L)\leq wt((\mathbf{x},\mathbf{x}))=2d_1$ y $d(L)\leq wt((\mathbf{0},\mathbf{y}))=d_2$, de donde $d(L)\leq\text{mín}\{2d_1,d_2\}$. Por lo tanto $d(L)=\text{mín}\{2d_1,d_2\}$. ■

Observación 3.7.27 Si L_i tiene matriz generadora G_i y matriz de control de paridad H_i, para $i=1,2$, entonces matrices generadora y de control de paridad para L son

$$\begin{pmatrix}G_1 & G_1\\ 0 & G_2\end{pmatrix}\ \text{y}\ \begin{pmatrix}H_1 & 0\\ -H_2 & H_2\end{pmatrix}.$$

Ejemplo 3.7.28 Sean $L_1 = \{000,110,101,011\}$ un $[3,2,2]$-código lineal binario y $L_2 = \{000,111\}$ un $[3,1,3]$-código lineal binario. Entonces

$$L = \{000000,110110,101101,011011,000111,110001,101010,\\ 011100\},$$

es un $[6,3,3]$-código lineal binario. □

El Teorema 3.7.26 puede utilizarse para demostrar el siguiente caso particular.

Corolario 3.7.29 Sea A un $[n,k,d]$-código lineal binario. Entonces el código L definido por

$$L = \{(\mathbf{c},\mathbf{c}) : \mathbf{c} \in A\} \cup \{(\mathbf{c},\mathbf{c}+\mathbf{1}) : \mathbf{c} \in A\},$$

donde $\mathbf{1} = 11\cdots1 \in \mathbb{F}_2^n$, es un $[2n,k+1,\text{mín}\{n,2d\}]$-código lineal binario.

Demostración. El resultado deseado se obtiene del Teorema 3.7.26, tomando $L_1 = A$ y $L_2 = \{\mathbf{0},\mathbf{1}\}$, donde $\mathbf{0} = 00\cdots0$ es el vector nulo de $\mathbb{F}_2^n$. ■

3.8. SageMath

En esta sección presentamos algunas maneras que existen en SageMath para construir códigos lineales. Para esto necesitamos construir matrices sobre cuerpos finitos. La forma general de hacer esto es con el comando `matrix(F,[v1,v2,..,vk])`, donde `F` es un cuerpo y `[v1,...,vk]` representa las filas de la matriz. Por ejemplo,

```
[In]: G=matrix(GF(2),[[1,0,0,0],[0,1,1,0],[0,0,1,0],
      [0,0,0,1]])
      G
```

```
[Out]: [1 0 0 0]
       [0 1 1 0]
       [0 0 1 0]
       [0 0 0 1]
```

Con este comando construimos una matriz de tamaño 4×4 con sus entradas en $\mathbb{F}_2$. Aún más, si queremos ver la matriz de una forma más clara podemos usar el comando `show(G)`.

```
[In]: show(G)
```

[Out]:

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

Una vez construida la matriz podemos usar el comando `LinearCode(G)` para implementar el subespacio vectorial (código lineal) generado por la filas de la matriz `G` dada.

```
[In]: L1=LinearCode(G1); L1
```

```
[Out]: [4, 4] linear code over GF(2)
```

Esto nos muestra que hemos construido un $[4,4]$ código lineal. Observemos que las filas de la matriz `G1` no son linealmente independientes. Cuando usemos una matriz con filas linealmente independientes, esta matriz es la que hemos definido como una matriz generadora del código. De esta forma, para construir el código del Ejemplo 3.5.5 usando SageMath hacemos lo siguiente.

```
[In]: G = Matrix(GF(2), [[1,0,0,1,1,0,1],
      [0,1,0,1,0,1,1],[0,0,1,0,1,1,0]])
      L = LinearCode(G); L
```

```
[Out]: [7, 3] linear code over GF(2)
```

Podemos listar los elementos de L como se muestra a continuación.

```
[In]: L.list()
```

```
[Out]: [(0, 0, 0, 0, 0, 0, 0), (1, 0, 0, 1, 1, 0, 1),
        (0, 1, 0, 1, 0, 1, 1), (1, 1, 0, 0, 1, 1, 0),
        (0, 0, 1, 0, 1, 1, 0), (1, 0, 1, 1, 0, 1, 1),
        (0, 1, 1, 1, 1, 0, 1), (1, 1, 1, 0, 0, 0, 0)]
```

Con los comandos `L.generator_matrix()` y `L.parity_check_matrix()` obtenemos una matriz generadora y una matriz de control de paridad para el código L.

```
[In]: L.generator_matrix()
```

```
[Out]: [1 0 0 1 1 0 1]
       [0 1 0 1 0 1 1]
       [0 0 1 0 1 1 0]
```

```
[In]: H = L.parity_check_matrix(); H
```

```
[Out]: [1 0 1 0 0 1 1]
       [0 1 1 0 0 1 0]
       [0 0 0 1 0 0 1]
       [0 0 0 0 1 1 1]
```

Estos cálculos nos confirman que la matriz G que usamos para definir el código es una matriz generadora de L. Es más podemos multiplicar estas dos matrices para verificar que obtenemos la matriz nula.

```
[In]: G*H.transpose()
```

```
[Out]: [0 0 0 0]
       [0 0 0 0]
       [0 0 0 0]
```

Codificación

Como vimos en este capítulo para codificar con códigos lineales, basta multiplicar el mensaje por la matriz generadora del código. Es decir, para codificar el mensaje $\mathbf{m} = 110$ hacemos lo siguiente

```
[In]: m=vector(GF(2),[1,1,0]); m*G
```

```
[Out]: (1, 1, 0, 0, 1, 1, 0)
```

Sin embargo, SageMath también trae la función implementada la función `L.encode(m)`, la cual nos da el mismo resultado.

```
[In]: L.encode(m)
```

```
[Out]: (1, 1, 0, 0, 1, 1, 0)
```

Decodificación

Para decodificar usamos el comando `L.decode_to_message()`. Así para decodificar la palabra $\mathbf{x} = 1100110$ escribimos

```
[In]: L.decode_to_message(vector(GF(2),[1, 1, 0, 0,
      1, 1, 0]))
```

```
[Out]: (1, 1, 0)
```

Podemos calcular la distancia mínima del código L con el comando `.minimum_distance()`

```
[In]: L.minimum_distance()
```

```
[Out]: 3
```

Es decir, sabemos que el código corrige hasta 1 error. Por ejemplo, supongamos que recibimos el mensaje $\mathbf{y} = 0100110$, en el cual hemos cometido 1 error con respecto al vector $\mathbf{x}$ anterior. Al decodificar obtenemos

```
[In]: L.decode_to_message(vector(GF(2),[0, 1, 0, 0,
      1, 1, 0])) # corrige el error cometido
```

```
[Out]: (1, 1, 0)
```

El método usado en realidad es el método del síndrome.

```
[In]: L.syndrome(vector(GF(2),[0, 1, 0, 0, 1, 1,
      0]))
```

```
[Out]: (1, 0, 0, 0)
```

Ahora con el comando `D = codes.decoders.LinearCodeSyndromeDecoder(L)` podemos construir la tabla de síndrome del código L

```
[In]: D = codes.decoders.
      LinearCodeSyndromeDecoder(L)
      D.syndrome_table()
```

```
[Out]: {(0, 0, 0, 0): (0, 0, 0, 0, 0, 0, 0),
        (1, 0, 0, 0): (1, 0, 0, 0, 0, 0, 0),
        (0, 1, 0, 0): (0, 1, 0, 0, 0, 0, 0),
```

```
(1, 1, 0, 0): (0, 0, 1, 0, 0, 0, 0),
(0, 0, 1, 0): (0, 0, 0, 1, 0, 0, 0),
(0, 0, 0, 1): (0, 0, 0, 0, 1, 0, 0),
(1, 1, 0, 1): (0, 0, 0, 0, 0, 1, 0),
(1, 0, 1, 1): (0, 0, 0, 0, 0, 0, 1),
(1, 0, 1, 0): (1, 0, 0, 1, 0, 0, 0),
(1, 0, 0, 1): (1, 0, 0, 0, 1, 0, 0),
(0, 1, 0, 1): (1, 0, 0, 0, 0, 1, 0),
(0, 0, 1, 1): (1, 0, 0, 0, 0, 0, 1),
(0, 1, 1, 0): (0, 1, 0, 1, 0, 0, 0),
(1, 1, 1, 1): (0, 1, 0, 0, 0, 0, 1),
(1, 1, 1, 0): (0, 0, 1, 1, 0, 0, 0),
(0, 1, 1, 1): (0, 0, 1, 0, 0, 0, 1)}
```

Si observamos detenidamente, el vector que tiene igual síndrome a la palabra enviada es `(1, 0, 0, 0, 0, 0, 0)`. Como sabemos esto significa que al sumar este vector a la palabra recibida, obtendremos la palabra enviada.

```
[In]: vector(GF(2),[0, 1, 0, 0, 1, 1, 0]) +
      vector(GF(2),[1, 0, 0, 0, 0, 0, 0])
```

```
[Out]: (1, 1, 0, 0, 1, 1, 0)
```

Código dual

Con el comando `.dual_code()` para construir el código dual de un código lineal. A continuación, mostramos la forma de construir el código dual de un código lineal.

```
[In]: G2 = matrix(GF(3),[[1,0,0,0,1,2,1],[0,1,0,0,2,
      1,0],[0, 0,1,2,2,2,2],[1, 0,1,2,0,1,0]])
      L2=LinearCode(G2); L2.dual_code()
```

```
[Out]: [7, 4] linear code over GF(3)
```

Recordemos que el código dual es el mismo generado por una matriz de control de paridad del código.

```
[In]: L2.dual_code()==LinearCode(L2.
      parity_check_matrix())
```

```
[Out]: True
```

Construcciones de códigos

Mediante el comando `.codes.ExtendedCod(C)` construimos el código extendido del código C. A continuación, presentamos un ejemplo para un código de Hamming.

```
[In]: C = codes.HammingCode(GF(2), 3);
      Ce=codes.ExtendedCode(C)
```

```
[In]: C.generator_matrix()
```

```
[Out]: [1 0 0 0 0 1 1]
       [0 1 0 0 1 0 1]
       [0 0 1 0 1 1 0]
       [0 0 0 1 1 1 1]
```

```
[In]: Ce.generator_matrix() # esta es la matriz
        generadora del código
```

```
[Out]: [1 0 0 0 0 1 1 1]
       [0 1 0 0 1 0 1 1]
       [0 0 1 0 1 1 0 1]
       [0 0 0 1 1 1 1 0]
```

Ahora para implementar el código perforado en la coordenada i usamos el comando `.shortened([0]`. Por ejemplo, para construir el código perforado en la primera coordenada del código lineal C que implementamos anteriormente, escribimos

```
[In]: C.shortened([0])
```

```
[Out]: [6, 3] linear code over GF(2)
```

```
[In]: Ce=C.extended_code()
      Ce
```

```
[Out]: Extension of [7, 4] Hamming Code over GF(2)
```

3.9. Ejercicios

3.9.1. Ejercicios teóricos

1. Sean $\mathbb{F}_q$ un campo finito y S un subconjunto de $\mathbb{F}_q^n$.

a) Probar que $S^{\perp}$ y $\langle S\rangle^{\perp}$ son subespacios de $\mathbb{F}_q^n$.

b) Probar que $S^{\perp} = \langle S\rangle^{\perp}$.

2. Para cada uno de los siguientes subconjuntos $S \subset \mathbb{F}_q^n$ con el correspondiente q, encuentre el espacio lineal $\langle S\rangle$ sobre $\mathbb{F}_q$ y su complemento ortogonal $S^{\perp}$.

   a) $S = \{101, 111, 010\}$, $q = 2$.

   b) $S = \{1020, 0201, 2001\}$, $q = 3$.

   c) $S = \{00101, 10001, 11011\}$, $q = 2$.

3. Determine cuales de los siguientes códigos son lineales sobre $\mathbb{F}_q$.

   a) $q = 2$ y $C = \{1101, 1110, 1011, 1111\}$

   b) $q = 3$ y $C = \{0000, 1001, 0110, 2002, 1111, 0220, 1221, 2112, 2222\}$

   c) $q = 2$ y $C = \{00000, 11110, 01111, 10001\}$.

4. Sean C y D códigos lineales de la misma longitud sobre $\mathbb{F}_q$. Defina

$$C + D = \{\mathbf{c} + \mathbf{d} : \mathbf{c} \in C, \mathbf{d} \in D\}$$

   Probar que $C + D$ es un código lineal y que $(C + D)^{\perp} = C^{\perp} \cap D^{\perp}$.

5. *a*) Sean $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$. Si $\mathbf{x}$ e $\mathbf{y}$ tienen ambos peso par o ambos tienen peso impar, mostrar que $\mathbf{x} + \mathbf{y}$ tiene peso par.

   b) Sean $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$. Si $\mathbf{x}$ tiene peso par e $\mathbf{y}$ tiene peso impar, mostrar que $\mathbf{x} + \mathbf{y}$ tiene peso impar.

   c) Probar que para cualquier código lineal binario C, todas las palabras tienen peso par o exactamente la mitad de las palabras tienen peso par.

6. Sea C un $[n, k, d]$ código lineal sobre $\mathbb{F}_q$. Suponga que para cada $1 \leq i \leq n$, existe al menos una palabra código cuya i-ésima componente no es cero.

   a) Muestre que la suma de los pesos de todas las palabras de C es igual a $n(q-1)q^{k-1}$.

b) Mostrar que $d \leq \frac{n(q-1)q^{k-1}}{q^k-1}$

c) Probar que no existe un código lineal binario de parámetros $[15,7,d]$ con $d \geq 8$.

7. Encuentre una matriz generadora y una matriz de control de paridad para el código lineal generado por cada uno de los siguientes subconjuntos. Determine los parámetros $[n,k,d]$ para cada uno de estos códigos.

a) $q = 2$ y $C = \{1000, 0110, 0010, 0001, 1001\}$.

b) $q = 3$ y $C = \{110000, 011000, 001100, 000110, 000111\}$.

c) $q = 2$ y $C = \{10101010, 11001100, 11110000, 01100110, 00111100\}$.

8. Asigne mensajes a las palabras en $\mathbb{F}_2^3$ como se muestra a continuación:

000	100	010	001	110	101	011	111
A	*C*	*D*	*E*	*G*	*I*	*N*	*O*

Sea C el código lineal con matriz generadora

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

Usando G codifique el mensaje ENCODING.

9. Construya un código lineal binario de longitud 6 como sigue: para cada $(x_1, x_2, x_3) \in \mathbb{F}_2^3$ construya una palabra $(x_1, x_2, x_3, x_4, x_5, x_6) \in C$, donde

$$\begin{aligned} x_4 &= x_1 + x_2 + x_3, \\ x_5 &= x_1 + x_3, \\ x_6 &= x_2 + x_3. \end{aligned}$$

a) Mostrar que C es un código lineal.

b) Encuentre una matriz generadora y de control de paridad para C.

10. Sea C un código lineal binario con matriz de control de paridad
$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$
Determine una matriz generadora para C y liste todas sus palabras.

Decodifique las siguientes palabras:

a) 110110,

b) 011011,

c) 101010.

11. Considere las palabras binarias $\mathbf{c}_1 = 11010000$, $\mathbf{c}_2 = 11100100$ y $\mathbf{c}_3 = 10101010$. Sea C el código formado por $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, todos los desplazamientos cíclicos de estas palabras y las palabras $\mathbf{0}$ y $\mathbf{1}$. Pruebe que C es un $(8,20,3)$-código.

12. Use la construcción $(\mathbf{u}, \mathbf{u}+\mathbf{v})$ y el código del ejercicio anterior para construir un $(16,2560,3)$-código.

13. Encuentre la distancia mínima para el código lineal con matriz generadora
$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

14. Suponga que se extiende un $[n,k]$ código lineal C sobre $\mathbb{F}_q$ al código $\widehat{C}$ donde
$$\widehat{C} = \left\{ x_1 x_2 \dots x_{n+1} \in \mathbb{F}_q^{n+1} : x_1 x_2 \dots x_n \in C; \sum_{i=1}^{n+1} x_i^2 = 0 \right\}.$$
¿Bajo qué condiciones $\widehat{C}$ es un código lineal?

15. Sea L un código lineal. Demuestre que el código acortado L_T sobre un conjunto de coordenadas T, también es un código lineal

16. Sea C el código lineal binario de repetición de longitud n. Describa los códigos $(C^{\perp})_T$ y $(C_T)^{\perp}$ para cualquier T.
17. Pruebe que la construcción $(\mathbf{u}|\mathbf{u}+\mathbf{v})$ usada para $[n,k_i,d_i]$-códigos lineales C_i produce un código de dimensión $k = k_1 + k_2$ y distancia $d = \min\{2d_1, d_2\}$
18. Sea L el $[7,4,3]$ código binario de Hamming. Calcule el polinomio enumerador de pesos de L y de $L^{\perp}$. Comprobar la identidad de MacWilliams para el código L.

3.9.2. Prácticas en SageMath

1. Implemente un programa para calcular la distancia de un código lineal a partir de la matriz de control de paridad (Teorema 3.5.7). Compruebe su funcionamiento resolviendo el Ejercicio 13.
2. Implemente un programa para determinar el arreglo estándar para un código lineal. Use el programa para encontrar el arreglo estándar para los siguientes códigos.
   a) $q = 3$ y $C = \{0000, 1010, 2020, 0101, 0202, 1111, 1212, 2121, 2222\}$.
   b) $q = 2$ y $C = \{00000, 10001, 01010, 11011, 00100, 10101, 01110, 11111\}$.
3. Sea C un (n, M, d) código binario. Implemente un programa en SageMath para calcular el código C^c.
4. Implemente un programa en SageMath para calcular el código recortado en t coordenadas para un código dado (Teorema 3.7.14).
5. Dados los $[n,k_i,d_i]$-códigos lineales sobre $\mathbb{F}_q$ para $i = 1,2$, implemente un programa en SageMath para construir el código $L = \{(\mathbf{u}, (\mathbf{u}+\mathbf{v})) : \mathbf{u} \in L_1, \mathbf{v} \in L_2\}$, del Teorema 3.7.26.

3.10. Biografía

Vera Stepen Pless **(1931-2020)**

Vera Pless fue una matemática estadounidense reconocida por sus contribuciones en el campo de la teoría de códigos y la geometría combinatoria. Nació el 17 de julio de 1931 en Riga, Letonia. Pless emigró a Estados Unidos en 1947 y se estableció allí. Obtuvo su doctorado en matemáticas en la Universidad de Nueva York en 1957. Durante su carrera, trabajó en diversas instituciones académicas, principalmente en la Universidad de Illinois en Chicago. Además, trabajó entre 1963 y 1972 en el Laboratorio de Investigación de Cambridge de la Fuerza Aérea.

Una de las contribuciones más importantes de Vera Pless fue su trabajo en la teoría de códigos. Pless desarrolló nuevos códigos y realizó importantes avances en el estudio de sus propiedades y rendimiento. Pless también hizo contribuciones significativas en el campo de la geometría combinatoria. Su investigación se centró en la teoría de diseño combinatorio, que trata de construir estructuras combinatorias con propiedades específicas. Sus investigaciones en este campo han tenido aplicaciones en la teoría de códigos, la teoría de grafos y la teoría de juegos. Fue autora de más de 120 publicaciones en revistas especializadas, varios libros y fue una firme defensora de los derechos de la mujer y otras causas contra la discriminación. Es recordada por su experiencia docente, sentido del humor y entusiasmo por el teatro y la música [2, 17]. Entre sus libros se destacan "Introduction to the Theory of Error-Correcting Codes" [32], en el cual se da una presentación amable de la teoría de códigos y "Handbook of Coding Theory" [33] y "Fundamentals of Error-Correcting Codes" [19] dos referencias obligatorias para interesados en el tema. Además, en 1981 publicó "The Cryptoclub: Using Mathematics to Make and Break Secret Codes" [4], una introducción a criptografía para estudiantes de secundaria.

Vera recibió varios reconocimientos por su trabajo, incluyendo el Premio Steele de Investigación Matemática en 2001 y el Premio Leroy P. Steele por trayectoria matemática en 2009, ambos otorgados por la Sociedad Matemática de América.

Foto tomada de MacTutor.
https://mathshistory.st-andrews.ac.uk/Biographies/Pless/

4. Cotas para el tamaño de un código

En la práctica se requiere que un (n,M,d)-código q-ario con n fijo tenga el mayor tamaño posible M y la mayor distancia posible d, debido a que el valor de M indica la cantidad de palabras código y d la capacidad del código para detectar y corregir errores. Este objetivo, sin embargo, no es tan fácil de alcanzar, razón por la cual se establecen diferentes cotas sobre el tamaño de un código. En este capítulo presentamos el problema fundamental de la teoría de códigos así como algunos de los resultados más importantes al respecto.

4.1. El problema fundamental

Con el propósito de codificar la máxima cantidad de información posible con una máxima capacidad para detectar y corregir errores que puedan presentarse en la trasmisión, surge un objetivo fundamental en la teoría de códigos. Sin embargo, estos dos objetivos son incompatibles, dado que entre más palabras tenga un código, su distancia mínima tiende a disminuir.

Definición 4.1.1 Sea A un alfabeto de orden q, la expresión $A_q(n,d)$ denota el mayor valor posible de un código q-ario de longitud n y distancia mínima d. Es decir

$$A_q(n,d) = \text{máx}\{M : \text{ existe un } (n,M,d)\text{-código sobre } A\}.$$

Un (n,M,d)-código q-ario para el cual $M = A_q(n,d)$ se denomina *código óptimo*.

El problema de determinar los valores de $A_q(n,d)$ se conoce como el *problema fundamental de la teoría de códigos*. Muchos esfuer-

zos se han dedicado a determinar estos valores; sin embargo, este problema es bastante complejo y por tanto varios de los resultados conocidos consisten en determinar los valores exactos de $A_q(n,d)$ para valores pequeños de q, n y d. Un enfoque más general consiste en determinar cotas inferiores y superiores para $A_q(n,d)$.

En lugar de considerar todos los códigos sobre un alfabeto dado, también podemos pensar en restringir este problema a los códigos lineales. En este caso tenemos la siguiente definición.

Definición 4.1.2 Sean q una potencia de un número primo, n y d enteros positivos con $1 \leq d \leq n$. El símbolo $B_q(n,d)$ denota el mayor valor de q^k para el cual existe un $[n,k,d]$-código lineal sobre $\mathbb{F}_q$; esto es

$$B_q(n,d) = \max\{q^k : \text{ existe un } [n,k,d]\text{-código lineal sobre } \mathbb{F}_q\}.$$

También podríamos pensar en buscar un código con la longitud más pequeña para valores dados del número de palabras M y la distancia mínima d. Esta longitud mínima se denota por $N_q(M,d)$. El problema de determinar $N_2(M,d)$ es equivalente al de encontrar $A_2(n,d)$, ver [25, p. 524, problema (1)]. Otro enfoque consiste en encontrar la distancia mínima más grande de un código para valores fijos de la longitud fija n y el tamaño M del código. Para códigos lineales este problema es abordado por A.E. Brouwer en [5], para encontrar algunos de estos valores se recomienda visitar la página web `http://www.codetables.de/` administrada por Markus Grassl, ver [14]. Sin embargo, estos problemas no serán abordados en este libro.

4.2. Resultados elementales

Algunos resultados elementales para los valores de $A_q(n,d)$ y $B_q(n,d)$ se resumen en el siguiente teorema.

Teorema 4.2.1 Sea q una potencia de un número primo, entonces

1. $B_q(n,d) \leq A_q(n,d) \leq q^n$.
2. $B_q(n,1) = A_q(n,1) = q^n$.

3. $B_q(n,n) = A_q(n,n) = q$.

Demostración.

1. Dado que $B_q(n,d)$ es el máximo tamaño posible para un código lineal, mientras que $A_q(n,d)$ lo es para un código arbitrario (en particular uno lineal) se cumple la primera desigualdad. La segunda desigualdad es inmediata, pues el máximo número de palabras de longitud n sobre un alfabeto q-ario es q^n.
2. Como $\mathbb{F}_q^n$ es un $[n,n,1]$-código lineal sobre $\mathbb{F}_q$ entonces $q^n \leq B_q(n,1) \leq A_q(n,1) \leq q^n$. Por lo tanto, $B_q(n,1) = A_q(n,1) = q^n$.
3. Sea C un (n,M,n)-código, dado que la longitud de C es n y la distancia entre cualquier par de palabras distintas de C es a lo menos n, obligatoriamente la distancia entre cualquier par de palabras distintas debe ser exactamente n. Por lo tanto, las palabras de C deben diferir en todas las coordenadas. Al considerar únicamente la primera posición de las M palabras de C, esta puede tomar a lo más q posibles valores, $0,1,\ldots,q-1$, es decir, $M \leq q$.

   Entonces,
   $$B_q(n,n) \leq A_q(n,n) \leq q.$$
   Por otro lado, el código de repetición de longitud n, es decir, $\{aa\cdots a : a \in \mathbb{F}_q\}$, es un $[n,1,n]$-código lineal sobre $\mathbb{F}_q$. Por lo tanto,
   $$B_q(n,n) = A_q(n,n) = q.$$ ■

Lema 4.2.2 Si $d < d^*$, entonces
$$A_q(n,d^*) \leq A_q(n,d).$$

Demostración. Se deja como ejercicio al lector (ver Ejercicio 1). ■

Teorema 4.2.3 Para todo $n \geq 2$,
$$A_q(n,d) \leq qA_q(n-1,d).$$

Demostración. Sea C un (n,M,d)-código q-ario óptimo, es decir $M = A_q(n,d)$. Considere los subcódigos

$$C_i = \{\mathbf{w} \in C : w_1 = i\},$$

para $i \in \mathbb{F}_q$. Afirmamos que existe $i_0 \in \mathbb{F}_q$ tal que $|C_{i_0}| \geq \frac{M}{q}$.

En efecto, supongamos que $|C_i| < \frac{M}{q}$ para todo $i = 0,1,\ldots,q-1$. Sea $C_j = \max\{|C_i| : 0 \leq i \leq q-1\}$, entonces

$$|C| = \sum_{i=0}^{q-1} |C_i| \leq q|C_j| < q\left(\frac{M}{q}\right) = M,$$

lo cual es una contradicción.

Ahora, note que eliminando la primera componente de cada palabra en C_{i_0} obtenemos un $(n-1,M',d')$-código $C_{i_0}^*$ con $d' \geq d$ y $M' = |C_{i_0}| \geq \frac{M}{q}$. Por lo tanto

$$A_q(n-1,d') \geq M' \geq \frac{M}{q} = \frac{A_q(n,d)}{q}.$$

Finalmente, el Lema 4.2.2 implica que $A_q(n-1,d') \leq A_q(n-1,d)$. Luego,

$$\frac{A_q(n,d)}{q} \leq A_q(n-1,d),$$
$$A_q(n,d) \leq qA_q(n-1,d).$$

■

Teorema 4.2.4 $B_2(n,2t+1) = B_2(n+1,2t+2)$.
En otras palabras, si d es par, entonces $B_2(n,d) = B_2(n-1,d-1)$.

Demostración. Se sigue del Teorema 3.7.7, según el cual un código binario $[n,M,2t+1]$ existe si y sólo si existe un código binario con parámetros $[n+1,M,2t+2]$. ■

Del anterior teorema se sigue que para el caso de códigos lineales binarios, sólo basta determinar el valor de $A_2(n,d)$ únicamente para valores impares de d (o sólo para los valores pares).

4.3. Cotas para $A_q(n,d)$

En esta sección presentamos algunos de los resultados más conocidos sobre cotas para $A_q(n,d)$. Iniciamos con la cota de Gilbert-Varshamov, seguida de la cota de Singleton, la cota de Hamming y por último la cota de Plotkin. Un estudio más amplio sobre este tema puede encontrarse en el Capítulo 2 de [19].

La cota de Gilbert-Varshamov es una cota inferior para $A_q(n,d)$, que se conoce desde la década de 1950. Su nombre viene de los trabajos de Edgar N. Gilbert [12] y Rom R. Varshamov [46].

Teorema 4.3.1 (Cota de Gilbert-Varshamov)

$$A_q(n,d) \geq \frac{q^n}{\sum_{i=0}^{d-1} \binom{n}{i}(q-1)^i}.$$

Demostración. Sea C un (n,M,d)-código óptimo sobre $\mathbb{F}_q$. Entonces no existe una palabra $\mathbf{w} \in \mathbb{F}_q^n$ tal que $d(\mathbf{w},\mathbf{c}) \geq d$, para todo $\mathbf{c} \in C$, dado que esto implica la existencia de un $(n,M+1,d)$-código sobre $\mathbb{F}_q$. En consecuencia

$$\bigcup_{\mathbf{c}\in C} S_q(\mathbf{c},d-1) = \mathbb{F}_q^n,$$

de donde

$$MV_q(n,d-1) \geq q^n$$
$$M \geq \frac{q^n}{V_q(n,d-1)}.$$

Luego,

$$A_q(n,d) \geq M \geq \frac{q^n}{\sum_{i=0}^{d-1} \binom{n}{i}(q-1)^i}.$$ ■

La siguiente es una cota superior para $A_q(n,d)$ que se debe al trabajo de Richard C. Singleton [43].

Teorema 4.3.2 (Cota de Singleton)

$$A_q(n,d) \leq q^{n-d+1}.$$

Demostración. Sea C un (n,M,d)-código q-ario. Si eliminamos las últimas $d-1$ coordenadas de cada palabra en C, las M palabras resultantes son aún diferentes. Entonces, dado que la longitud de estas palabras es $n-d+1$ se obtiene que,

$$M \leq q^{n-d+1},$$

y por tanto $A_q(n,d) \leq q^{n-d+1}$. ■

La cota de Singleton implica que cualquier $[n,k,d]$-código lineal satisface

$$\begin{aligned} q^k &\leq q^{n-d+1} \\ k &\leq n-d+1. \end{aligned}$$

Así obtenemos la siguiente desigualdad,

$$d \leq n-k+1. \tag{4.1}$$

Un código lineal para el cual se tiene la igualdad en la expresión (4.1) se denomina *código separable por distancia máxima*, o *código* MDS, dado que tiene la máxima distancia mínima posible para cualquier código con una longitud y dimensión dadas. Un código MDS tiene parámetros $[n,n-d+1,d]$ o $[n,k,n-k+1]$.

Por otro lado, si eliminamos cualquier conjunto de $n-k=d-1$ posiciones de las palabras en un código C que sea MDS, entonces obtenemos q^k palabras distintas de longitud k. Es decir, obtenemos $\mathbb{F}_q^k$. Por tanto, cualquier conjunto de k posiciones en C contiene todas las posibles k-uplas, lo cual implica que C es sistemático sobre cualquier conjunto de k posiciones.

Ahora, presentamos una adaptación de la Cota de Gilbert-Varshamov.

Teorema 4.3.3 (Cota de Gilbert-Varshamov para códigos lineales) Existe un $[n,k]$-código linear q-ario con distancia mínima al menos d si

$$q^k < \frac{q^n}{\sum_{i=0}^{d-2} \binom{n-1}{i}(q-1)^i}. \tag{4.2}$$

Además, si k es el mayor entero positivo tal que se satisface (4.2), entonces $A_q(n,d) \geq q^k$.

Demostración. Por el Teorema 3.5.7, debemos construir una matriz de control de paridad H de tamaño $(n-k) \times n$, con sus entradas en $\mathbb{F}_q$, tal que cualquier conjunto con $d-1$ de sus columnas sea linealmente independiente.

Para hacer esto, primero seleccionamos un vector columna no nulo de $\mathbb{F}_q^{n-k}$ para que sea la primera columna de H. Para la segunda columna, escogemos cualquier vector no nulo en $\mathbb{F}_q^{n-k}$ que no sea un múltiplo escalar de la primera columna. En general, la j-ésima columna de H debe ser un vector no nulo de $\mathbb{F}_q^{n-k}$ que no esté en el espacio generado por cualquier conjunto con $d-2$ columnas de las $j-1$ seleccionadas previamente. Ahora, observemos que el número de combinaciones lineales de $d-2$ o menos columnas, de las $j-1$ columnas ya seleccionadas, es igual a

$$N_j = \binom{j-1}{1}(q-1) + \cdots + \binom{j-1}{d-2}(q-1)^{d-2}.$$

Dado que el vector nulo $\mathbf{0}$ no puede ser una columna de H, la j-ésima columna se puede seleccionar en $q^{n-k} - N_j - 1$ formas diferentes. Ahora por (4.1), tenemos que $d-2 < n-k$ y luego

$$1 + N_n = \sum_{i=0}^{d-2} \binom{n-1}{i}(q-1)^i < \sum_{i=0}^{n-k} \binom{n-k}{i}(q-1)^i = q^{n-k}. \tag{4.3}$$

Además, es claro que $N_j < N_n$ para todo $j < n$, entonces $0 < q^{n-k} - N_j - 1$ para todo $j \leq n$, lo que garantiza que podemos completar la formación de la matriz H.

Por lo tanto, de (4.3), obtenemos que

$$q^k < \frac{q^n}{1+N_n},$$

que es la desigualdad buscada. ■

El hecho de que las esferas de radio $\left\lfloor \frac{d-1}{2} \right\rfloor$ centradas en las palabras de un código son disjuntas, proporciona la siguiente cota superior para $A_q(n,d)$, que se conoce como *cota de Hamming* o *cota de empaquetamiento de esferas*.

Teorema 4.3.4 (Cota de Hamming)

$$A_q(n,d) \leq \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i},$$

donde

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

Demostración. Sea C un (n,M,d)-código q-ario óptimo. Por el Corolario 2.4.7 y el Teorema 2.8.6, las esferas de radio $pr(C) = t = \left\lfloor \frac{d-1}{2} \right\rfloor$ y centro en cada elemento de C son disjuntas.

Luego

$$\begin{aligned} MV_q(n,t) &\leq q^n \\ M &\leq \frac{q^n}{V_q(n,t)} \\ A_q(n,d) = M &\leq \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i}. \end{aligned}$$

■

La siguiente es una cota superior para $A_q(n,d)$, su nombre es en honor a Morris Plotkin [34].

Teorema 4.3.5 (Cota de Plotkin)
Sea $\theta = \dfrac{q-1}{q}$, si $d > \theta n$, entonces

$$A_q(n,d) \leq \frac{d}{d-\theta n}.$$

Demostración. Sean C un (n,M,d)-código q-ario y $\theta = \frac{q-1}{q}$. Consideremos las sumas de las distancias entre las palabras código, la cual está dada por

$$S = \sum_{\mathbf{c}\in C} \sum_{\mathbf{b}\in C} d(\mathbf{c},\mathbf{b}).$$

Dado que la distancia mínima de C es d, se tiene que

$$S \geq M(M-1)d. \tag{4.4}$$

Por otro lado, sea B la matriz de tamaño $M \times n$, cuyas filas son las palabras de C. Para $1 \leq i \leq n$, sea K_{ij} el número de veces que $j \in \mathbb{F}_q$ aparece en la i-ésima columna de B. Entonces,

$$\sum_{j\in\mathbb{F}_q} K_{ij} = M,$$

para todo $1 \leq i \leq n$.

Ahora, sea $j \in \mathbb{F}_q$. Si $K_{ij} = 0$ (es decir que j no aparece en la i-ésima columna), se tiene que esta columna no suma en S. Lo mismo en el caso $K_{ij} = M$, como j aparece en toda la i-ésima columna, esta columna no aporta a la suma S.
En general, supongamos que $K_{ij} = t$, es decir, j aparece para t filas de B en la i-ésima columna y que no aparece en las $M - t$ filas restantes. Entonces, cada una de las t veces en que j aparece en la i-ésima columna, dicha columna aporta $M - t$ unos a la suma S. En consecuencia,

$$S = \sum_{i=1}^{n} \left(\sum_{j=0}^{q-1} K_{ij}(M - K_{ij}) \right) = \sum_{i=1}^{n} \left(M^2 - \sum_{j=0}^{q-1} K_{ij}^2 \right).$$

Por la desigualdad de Cauchy-Schwarz para $\mathbf{x} = K_{i0}K_{i1}\cdots K_{iq-1}$ y $\mathbf{1} = 11\ldots1$ (donde $||\cdot||$ es la norma inducida por la distancia de Hamming), tenemos que $||\mathbf{x}\cdot\mathbf{1}|| \leq ||\mathbf{x}||\,||\mathbf{1}||$, esto es

$$\left(\sum_{j=0}^{q-1} K_{ij}\right)^2 \leq q\sum_{j=0}^{q-1} K_{ij}^2$$
$$\frac{1}{q}M^2 \leq \sum_{j=0}^{q-1} K_{ij}^2.$$

Luego,

$$\begin{aligned} S = \sum_{i=1}^{n}\left(M^2 - \sum_{j=0}^{q-1} K_{ij}^2\right) &\leq \sum_{i=1}^{n}\left(M^2 - \frac{M^2}{q}\right) \\ &= nM^2\left(1-\frac{1}{q}\right) = n\theta M^2. \end{aligned} \tag{4.5}$$

Por lo tanto, por las desigualdades (4.4) y (4.5) se tiene

$$\begin{aligned} M(M-1)d \leq S \leq \theta M^2 n \\ (M-1)d \leq \theta M n \\ Md - d \leq \theta M n \\ M(d-\theta n) \leq d \\ M \leq \frac{d}{d-\theta n}. \end{aligned}$$

■

Para códigos binarios, se tiene otra versión de la cota de Plotkin.

Lema 4.3.6 Sea $m > 0$ un número impar y supongamos que $m = x + y$, con x, y enteros positivos. Entonces

$$xy \leq \left(\frac{m-1}{2}\right)\left(\frac{m+1}{2}\right).$$

Demostración. Si x o y es igual a $\frac{m-1}{2}$, entonces $x = \frac{m-1}{2}$ y $y = \frac{m+1}{2}$ o recíprocamente. Por tanto, en este caso se tiene la igualdad.

En caso contrario, supongamos sin pérdida de generalidad que $x < \frac{m-1}{2}$, es decir $x = \frac{m-1}{2} - t$, con $t > 0$, entonces $y = \frac{m+1}{2} + t$ y por lo tanto

$$\begin{aligned} xy &= \left(\frac{m-1}{2} - t\right)\left(\frac{m+1}{2} + t\right) \\ &= \left(\frac{m-1}{2}\right)\left(\frac{m+1}{2}\right) + \frac{t(m-1)}{2} - \frac{t(m+1)}{2} - t^2 \\ &= \left(\frac{m-1}{2}\right)\left(\frac{m+1}{2}\right) - t - t^2 \\ &< \left(\frac{m-1}{2}\right)\left(\frac{m+1}{2}\right). \end{aligned}$$

■

Teorema 4.3.7 (Cota de Plotkin para códigos binarios)

1. Si d es par, entonces

$$A_2(n,d) \leq \begin{cases} 2\lfloor d/(2d-n)\rfloor, & \text{para } n < 2d, \\ 4d, & \text{para } n = 2d. \end{cases}$$

2. Si d es impar, entonces

$$A_2(n,d) \leq \begin{cases} 2\lfloor (d+1)/(2d+1-n)\rfloor, & \text{para } n < 2d+1, \\ 4d+4, & \text{para } n = 2d+1. \end{cases}$$

Demostración.

1. Sea C un (n,M,d)-código binario con d un número par. Si $n < 2d$, por el Lema 4.3.6 tenemos que $M \leq \frac{2d}{2d-n}$.

   Si M es par, digamos $M = 2t$, con $t \in \mathbb{Z}^+$, entonces

$$\begin{aligned} 2t &\leq 2\frac{d}{2d-n} \\ t &\leq \left\lfloor \frac{d}{2d-n} \right\rfloor \\ 2t &\leq 2\left\lfloor \frac{d}{2d-n} \right\rfloor \\ M &\leq 2\left\lfloor \frac{d}{2d-n} \right\rfloor. \end{aligned}$$

Por otro lado, si M es impar, en la demostración del teorema anterior obsérvese que

$$\begin{aligned} S &= \sum_{i=1}^{n} [k_{i,0}(M-k_{i,0}) + k_{i,1}(M-k_{i,1})] \\ &= \sum_{i=1}^{n} 2k_{i,0}k_{i,1} = 2\sum_{i=1}^{n} k_{i,0}k_{i,1}, \end{aligned}$$

dado que $k_{i,0} + k_{i,1} = M$. Sin embargo, por el Lema 4.3.6 $k_{i,0}k_{i,1} \leq \left(\frac{M+1}{2}\right)\left(\frac{M-1}{2}\right)$, en consecuencia se tiene

$$S \leq 2\left(\frac{n(M+1)(M-1)}{4}\right) = \frac{n(M+1)(M-1)}{2}.$$

Luego,

$$\begin{aligned} M(M-1)d \leq S &\leq \frac{n(M+1)(M-1)}{2} \\ Md &\leq \frac{n(M+1)}{2} \\ 2Md &\leq nM + n \\ M(2d-n) &\leq n \\ M &\leq \frac{n}{2d-n} \\ M &\leq \frac{2d}{2d-n} - 1 \end{aligned}$$

y dado que M es impar, por un razonamiento semejante al del caso par, obtenemos que $M \leq 2\left\lfloor \frac{d}{2d-n} \right\rfloor$. Para $n = 2d$, sea $d = 2k$, entonces por el Teorema 4.2.3 y por el caso anterior (cuando $n < 2d$), se tiene que

$$\begin{aligned} A_2(n,d) = A_2(4k,2k) &\leq 2A_2(4k-1,2k) \\ &\leq 4\left\lfloor \frac{2k}{4k-(4k-1)} \right\rfloor = 8k = 4d. \end{aligned}$$

2. Sea d un número impar, supongamos que $n < 2d+1$, entonces por el Teorema 4.2.4 y por el ítem 1, se tiene

$$A_2(n,d) = A_2(n+1,d+1) \leq 2\left\lfloor \frac{d+1}{2(d+1)-(n+1)} \right\rfloor = 2\left\lfloor \frac{d+1}{2d+1-n} \right\rfloor.$$

Para $n = 2d+1$, aplicando los teoremas 4.2.3, 4.2.4 y el Ítem 1, se tiene

$$A_2(2d+1,d) = A_2(2d+2,d+1) \leq 2A_2(2d+1,d+1) \leq 4\left\lfloor \frac{d+1}{2(d+1)-(2d+1)} \right\rfloor = 4(d+1). \quad \blacksquare$$

4.4. Cotas para $A(n,d,w)$

En esta sección nos centramos en el estudio de los valores de $A_q(n,d)$ para el caso de códigos binarios de peso constante. La notación $A(n,d,w)$ denota el máximo tamaño posible para un código binario de longitud n, distancia mínima al menos d y peso constante w. A continuación se presentan algunas propiedades básicas para los valores de $A(n,d,w)$.

Observación 4.4.1 La distancia entre dos palabras de igual peso es siempre par, ya que si fuese impar, una palabra debería tener al menos un uno más que la otra. Así, la distancia mínima en un código de peso constante debe ser par. Por tanto, el hecho que en la notación $A(n,d,w)$ se mencione que la distancia sea "al menos d"garantiza la existencia de códigos de este tipo cuando d es impar.

Lema 4.4.2
1. $A(n,2k,k) = \left\lfloor \frac{n}{k} \right\rfloor$.
2. $A(n,2k,w) = A(n,2k,n-w)$.
3. $A(n,2k-1,w) = A(n,2k,w)$.

Demostración.

1. Para construir un (n,M,d)-código binario C, con $d \geq 2k$ y $M = A(n,2k,k)$, todas las palabras en C deben tener k unos y distancia al menos $2k$, luego, ninguna palabra puede tener un 1 en la misma posición que otra. Así, el mayor número de palabras binarias, de longitud n que cumplen las anteriores condiciones es t, donde $n = tk + r$, con $0 \leq r < k$. Es decir
$$M = t = \left\lfloor \frac{n}{k} \right\rfloor.$$

2. Sean C un (n,M,d)-código binario con $d \geq 2k$, tal que $M = A(n,2k,k)$ y C' es el código obtenido por la permutación de los símbolos iguales cero y uno en las palabras de C. Entonces C y C' tienen la misma distancia por ser códigos equivalentes (Teorema 2.7.12). Además, todas las palabras en C' tienen peso $n - w$, por lo tanto
$$A(n,2k,w) \leq A(n,2k,n-w).$$
De forma similar se prueba la otra desigualdad.

3. Dado que la distancia entre palabras de igual peso debe ser par, el número máximo de palabras binarias con w unos que se pueden construir de tal manera que la distancia sea mayor o igual que $2k-1$, es el mismo que se requiere para que la distancia sea mayor o igual que $2k$. Luego,
$$A(n,2k-1,w) = A(n,2k,w).$$ ■

Teorema 4.4.3 $A(n,2k,w) \leq \left\lfloor \dfrac{kn}{w^2 - wn + kn} \right\rfloor$.

Demostración. Sean C un $(n,M,2k)$-código de peso constante w, para el cual $M = A(n,2k,w)$, B la matriz de tamaño $M \times n$ cuyas filas son las palabras en C y k_i el número de unos en la i-ésima columna de B. Calculemos de dos formas la suma S de los productos escalares $\mathbf{c}_i \cdot \mathbf{c}_j$ de todos los pares distintos de las filas de B.

Por un lado, si $i \neq j$, entonces

$$\begin{aligned}\mathbf{c}_i \cdot \mathbf{c}_j = w(\mathbf{c}_i \cap \mathbf{c}_j) &= \frac{1}{2}[w(\mathbf{c}_i) + w(\mathbf{c}_j) - d(\mathbf{c}_i, \mathbf{c}_j)] \\ &\leq \frac{1}{2}[2w - 2k] = w - k,\end{aligned}$$

y por lo tanto

$$S = \sum_{i=1}^{M} \sum_{\substack{j=1 \\ j \neq i}}^{M} \mathbf{c}_i \cdot \mathbf{c}_j \leq (w-k)M(M-1).$$

Por otro lado, viendo la sumatoria por columnas, la i-ésima columna aporta a la suma S con $k_i(k_i - 1)$, de donde se obtiene que

$$S = \sum_{i=1}^{n} k_i(k_i - 1) = \sum_{i=1}^{n} k_i^2 - \sum_{i=1}^{n} k_i = \sum_{i=1}^{n} k_i^2 - wM,$$

pero por la desigualdad de Cauchy-Schwarz, tenemos

$$\begin{aligned}\left(\sum_{i=1}^{n} k_i\right)^2 &\leq n \sum_{i=1}^{n} k_i^2 \\ \frac{1}{n}\left(\sum_{i=1}^{n} k_i\right)^2 &\leq \sum_{i=1}^{n} k_i^2 \\ \frac{1}{n}(wM)^2 &\leq \sum_{i=1}^{n} k_i^2,\end{aligned}$$

de donde

$$\frac{w^2M^2}{n} - wM \leq S.$$

Entonces,

$$\begin{aligned}
\frac{w^2M^2}{n} - wM &\leq S \leq (w-k)M(M-1) \\
\frac{w^2M^2 - wMn}{n} &\leq M(wM - w - Mk + k) \\
w^2M - wn &\leq wMn - wn - Mkn + kn \\
M(w^2 - wn + kn) &\leq kn \\
M &\leq \left\lfloor \frac{kn}{w^2 - wn + kn} \right\rfloor .
\end{aligned}$$

■

A continuación se presenta un resultado que proporciona una desigualdad recursiva para $A(n,2k,w)$.

Teorema 4.4.4 $A(n,2k,w) \leq \left\lfloor \frac{n}{w} A(n-1,2k,w-1) \right\rfloor$.

Demostración. Sea C un $(n,M,2k)$-código de peso constante w, con $M = A(n,2k,w)$. Por un lado, el número total de unos de todas las palabras en C es wM. Por otro lado, la sección transversal $x_1 = 1$ de C tiene longitud $n-1$, distancia mínima al menos $2k$ y peso constante $w-1$, de donde su tamaño es a lo más $A(n-1,2k,w-1)$. En consecuencia, hay a lo más $A(n-1,2k,w-1)$ palabras en C con un 1 en la primera posición. Aplicando este razonamiento a las $n-1$ posiciones restantes se obtiene que el número total de unos de todas las palabras en C es a lo más $nA(n-1,2k,w-1)$. Por lo tanto

$$\begin{aligned}
wM &\leq nA(n-1,2k,w-1), \\
M &\leq \left\lfloor \frac{n}{w} A(n-1,2k,w-1) \right\rfloor .
\end{aligned}$$

■

Corolario 4.4.5

$$\begin{aligned}
A(n,2k-1,w) &= A(n,2k,w) \\
&\leq \left\lfloor \frac{n}{k} \left\lfloor \frac{n-1}{k-1} \left\lfloor \cdots \left\lfloor \frac{n-w+k}{k} \right\rfloor \cdots \right\rfloor \right\rfloor \right\rfloor .
\end{aligned}$$

Demostración. La igualdad se sigue de la parte 3 del Lema 4.4.2. Para probar la desigualdad, primero aplicamos el Teorema 4.4.4 reiteradamente hasta obtener

$$\begin{aligned} A(n,2k,w) &\leq \left\lfloor \frac{n}{w} \left\lfloor \frac{n-1}{w-1} \left\lfloor \cdots \left\lfloor \frac{n-(w-k-1)}{w-(w-k-1)} \right. \right. \right. \right. \\ &\qquad \left. \left. \left. A(n-(w-k),2k,w-(w-k)) \right\rfloor \cdots \right\rfloor \right\rfloor \right\rfloor \\ &= \left\lfloor \frac{n}{w} \left\lfloor \frac{n-1}{w-1} \left\lfloor \cdots \left\lfloor \frac{n-w+k+1}{k+1} A(n-w+k,2k,k) \right\rfloor \cdots \right\rfloor \right\rfloor \right\rfloor. \end{aligned}$$

Entonces, por la parte 1 del Lema 4.4.2,

$$A(n-w+k,2k,k) = \left\lfloor \frac{n-w+k}{k} \right\rfloor.$$

Por lo tanto,

$$\begin{aligned} A(n,2k-1,w) &= A(n,2k,w) \\ &\leq \left\lfloor \frac{n}{w} \left\lfloor \frac{n-1}{w-1} \left\lfloor \cdots \left\lfloor \frac{n-w+k}{k} \right\rfloor \cdots \right\rfloor \right\rfloor \right\rfloor. \end{aligned}$$

■

Algunos de los resultados vistos anteriormente, permiten calcular valores exactos para $A_q(n,d)$.

Ejemplo 4.4.6 Para $A_2(6,4)$, por la cota de Plotkin se tiene

$$A_2(6,4) \leq 4. \tag{4.6}$$

Por otro lado, el código $C = \{00000, 01101, 10110, 11011\}$ es un código binario $(5,4,3)$. De donde $A_2(5,3) \geq 4$, pero por el Teorema 4.2.4 tenemos que $A_2(5,3) = A_2(6,4)$ y por la desigualdad (4.6)

$$4 \leq A_2(5,3) = A_2(6,4) \leq 4.$$

Por lo tanto, $A_2(6,4) = 4$, lo cual corresponde con el valor dado en la Tabla 4.2. □

En la Tabla 4.1 se encuentran los valores de $A_2(n,d)$ tomada de [36], mientras que en la Tabla 4.2 con los valores de $A_2(n,d)$ tomada de [19, p. 54, Tabla 2.1]. Una tabla con más valores de $B_2(n,d)$ se puede encontrar en [20, p. 34, Tabla 1].

n	$d=3$	$d=5$	$d=7$
5	4	2	-
6	8	2	-
7	16	2	2
8	20	4	2
9	40	6	2
10	72-79	12	2
11	144-158	24	4
12	256	32	4
13	512	64	8
14	1024	128	16
15	2048	256	32
16	2720-3276	256-340	36-37

Tabla 4.1: Algunos valores para $A_2(n,d)$.

Las cotas para $A_q(n,d,w)$ pueden ser usadas para derivar cotas superiores para $A_q(n,d)$. Esta cota recibe el nombre de Cota de Johnson y su demostración puede ser consultada en [19, Thm. 2.3.8].

n	$d=4$	$d=6$	$d=8$	$d=10$
6	4	2	1	1
7	8	2	1	1
8	16	2	2	1
9	20	4	2	1
10	40	6	2	2
11	72	12	2	2
12	144	24	4	2
13	256	32	4	2
14	512	64	8	2
15	1024	128	16	4
16	2048	256	32	4
17	2720-3276	256-340	36-37	6
18	5312-6552	512-680	64-72	10
19	10496-13104	1024-1288	128-144	20
20	20480-26208	2048-2372	256-279	40
21	36864-43689	2560-4096	512	42-48
22	73728-87378	4096-6941	1024	50-88
23	147456-173491	8192-13774	2048	76-150
24	294912-344308	16384-24106	4096	128-280

Tabla 4.2: Más valores para $A_2(n,d)$.

Teorema 4.4.7 (Cota de Johnson) Sea $t = \left\lfloor \frac{d-1}{2} \right\rfloor$.

1. Si d es un número impar, entonces

$$A_q(n,d) \leq \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i + \frac{\binom{n}{t+1}(q-1)^{t+1} - \binom{d}{t}A_q(n,d,d)}{A_q(n,d,t+1)}}.$$

2. Si d es un número par, entonces

$$A_q(n,d) \leq \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i + \frac{\binom{n}{t+1}(q-1)^{t+1}}{A_q(n,d,t+1)}}.$$

3. Si d es un número impar, entonces

$$A_2(n,d) \leq \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i} + \frac{\binom{n}{t+1} - \binom{d}{t}A_2(n,d,d)}{\lfloor \frac{n}{t+1} \rfloor}}.$$

4. Si d es un número par, entonces

$$A_2(n,d) \leq \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i} + \frac{\binom{n}{t+1}}{\lfloor \frac{n}{t+1} \rfloor}}.$$

5. Si d es un número impar, entonces

$$A_2(n,d) \leq \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i} + \frac{\binom{n}{t}\left(\frac{n-t}{t+1} - \lfloor \frac{n-t}{t+1} \rfloor\right)}{\lfloor \frac{n}{t+1} \rfloor}}.$$

4.5. SageMath

SageMath permite calcular algunas de las cotas más importantes para el tamaño de un código. Las cotas de Gilbert-Varshamov 4.3.1, Singleton 4.3.2, Hamming 4.3.4 y Plotkin 4.3.5, se pueden calcular en mediante las siguientes funciones: `codes.bounds.gilbert_lower_bound(n,q,d)`, `codes.bounds.singleton_upper_bound(n,q,d)`, `codes.bounds.hamming_upper_bound(n,q,d)` y `codes.bounds.plotkin_upper_bound(n,d,q)`, respectivamente.

Determinemos en SageMath cotas para $A_3(10,5)$.

```
[In]: codes.bounds.singleton_upper_bound(10,3,5)
```

```
[Out]: 729
```

```
[In]: codes.bounds.hamming_upper_bound(10,3,5)
```

```
[Out]: 293
```

```
[In]: floor(codes.bounds.
      gilbert_lower_bound(10,3,5))
```

```
[Out]: 13
```

Por la cota de Gilbert-Varshamov 4.3.1 se tiene que $A_3(10,5) \geq 13$, por la cota de Singleton 4.3.2 $A_3(10,5) \leq 729$ y por la cota de Hamming 4.3.4 $A_3(10,5) \leq 293$. Se puede observar que en este caso la cota Hamming da una mejor aproximación que la cota de Singleton. Sin embargo, con estos resultados aún existe una gran diferencia entre la cota inferior (13) y la cota superior (293).

En el ejemplo anterior, no se puede aplicar la cota de Plotkin 4.3.5, dado que no se cumple que $d > \theta n$, con $\theta = \frac{q-1}{q}$, pues en este caso $\theta = 2/3$ y $d = 5 < (2/3)10 = \theta n$.

Otra función de gran importancia en SageMath es `codes.bounds.codesize upper bound(n,d,q)`, la cual presenta la mejor cota superior que el programa posee para $A_q(n,d)$. Por ejemplo, para $A_3(10,5)$ se tiene

```
[In]: codes.bounds.codesize_upper_bound(10,5,3)
```

```
[Out]: 293
```

En este caso dicha cota coincide con la cota superior de Hamming, que es 293. Otro ejemplo para $A_3(14,10)$ se muestra a continuación

```
[In]: codes.bounds.singleton_upper_bound(14,3,10)
```

```
[Out]: 243
```

```
[In]: codes.bounds.hamming_upper_bound(14,3,10)
```

```
[Out]: 247
```

```
[In]: codes.bounds.plotkin_upper_bound(14,3,10)
```

```
[Out]: 15
```

```
[In]: codes.bounds.codesize_upper_bound(14,10,3)
```

```
[Out]: 15
```

Note que la mejor cota que SageMath tiene para $A_3(14,10)$ coincide con la cota de Plotkin. Podemos usar en este caso la cota de Plotkin para calcular una cota superior para $A_2(10,3)$.

```
[In]: codes.bounds.plotkin_upper_bound(10,2,3)
```

```
[Out]: 192
```

Sin embargo la mejor cota superior para $A_2(10,3)$ que el programa tiene es

```
[In]: codes.bounds.codesize_upper_bound(10,3,2)
```

```
[Out]: 93
```

Finalmente programemos el quinto caso de la cota de Johnson para códigos de peso constante.

```
[In]: def Johnson(n,w,l):
          if n>=w>l>=1:
```

```
        c=((n-l)/(w-l)).floor()
        for i in range(1,l):
            c=c*(n-(l-i))/(w-(l-i)).floor()
        d=(c/w).floor()
        print(d)
    else:
        print('Los valores de n, w y l deben
              cumplir n>=w>l>=1')
```

[In]:
```
Johnson(13,3,1)
```

[Out]: 2

[In]:
```
def CotaJ5(n,d):
    if (d-1)%2==0:
        t=(d-1)/2
        S=1
        for i in [1..t]:
            S=S+binomial(n,i)
        L=S
        R=(binomial(n,t)*((n-1)/(t+1)
         -floor((n-1)/(t+1))))/floor(n/(t+1))
        Valor=2^n/(L+R)
        print('La cota de Johnson
              para A_2(n,d) es ',Valor)
    else:
        print('El valor de d debe ser impar')
```

Probando la cota para $A_2(4,5)$ obtenemos

[In]:
```
CotaJ5(4,5)
```

```
La cota de Johnson para A_2(n,d) es 16/11
```

4.6. Ejercicios

4.6.1. Ejercicios teóricos

1. Demostrar el Lema 4.2.2.
2. Sin hacer uso de las cotas discutidas en este capítulo, mostrar que $A_2(6,5) = 2$ y $A_2(7,5) = 2$.

Sugerencia: para la primera igualdad, mostrar que $A_2(6,5) \leq 2$ encontrando un código explícitamente, luego intente mostrar que $A_2(6,5) \geq 2$ usando un argumento combinatorio.

3. Encontrar un código binario óptimo con $n = 3$ y $d = 2$.

4. Sean $q \geq 2$ y $n \geq 2$ enteros. Probar que $A_q(n,2) = q^{n-1}$.

5. Determine cuales de los siguientes códigos existen. Justifique su respuesta.

   a) Un $(8,29,3)$-código lineal binario.

   b) Un $(8,8,5)$-código lineal binario.

   c) Un $(8,5,5)$-código lineal binario.

   d) Un $(24,2^{12},8)$-código lineal binario.

   e) Un $(63,2^{57},3)$-código lineal binario.

6. Sea C el código sobre $\mathbb{F}_4 = \{0,1,\alpha,\alpha^2\}$ con matriz generadora

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & \alpha & \alpha^2 \end{pmatrix}$$

   a) Pruebe que C es un código MDS.

   b) Encuentre una matriz generadora para el código dual.

   c) Pruebe que $C^{\perp}$ es un código MDS.

7. Denote por $A_2(n,d,w)$ el máximo valor posible de M palabras código en un $(n,M,d;w)$ código binario de peso constante. Mostrar que

   a) $1 \leq A_2(n,d,w) \leq \binom{n}{w}$.

   b) $A_2(n,2,w) = \binom{n}{w}$.

   c) $A_2(n,d,w) = 1$ para $d > 2w$.

   d) $A_2(n,d,w) = A_2(n,d,n-w)$.

8. Sea C un $(n, M, d = 2t+1, w = 2t+1)$-código de peso constante sobre $\mathbb{F}_q$. Mostrar que

$$M \leq \frac{\binom{n}{t+1}(q-1)^{t+1}}{\binom{2t+1}{t}}.$$

   Sugerencia: Para cada palabra código $\mathbf{c}$ hay $\binom{2t+1}{t}$ palabras $\mathbf{y}$ de peso de Hamming igual a $t+1$ en in $\mathbb{F}_q^n$ tal que $d(\mathbf{y}, \mathbf{c}) = t$. Dada una palabra $\mathbf{y}$ de peso de Hamming igual a $t+1$ en $\mathbb{F}_q^n$, ¿cuántas palabras código $\mathbf{c} \in C$ hay tal $d(\mathbf{y}, \mathbf{c}) = t$?

9. Use los códigos de repetición, así como los códigos obtenidos de estos para verificar las cotas inferiores para $A_2(3,3)$, $A_2(4,3)$, $A_2(5,5)$, $A_2(6,5)$ y $A_2(7,5)$ mostrados en la Tabla 4.1.

4.6.2. Prácticas en SageMath

1. Construya una matriz de control de paridad H para un código Hamming binario de longitud 15, donde la j-ésima columna de H es igual representación binaria de j. Use H para construir una tabla de síndromes y utilícela para decodificar las siguientes palabras:
   a) 01010 01010 01000,
   b) 11100 01110 00111,
   c) 11001 11001 11000.
2. Implemente un programa que calcule la cota del Teorema 4.4.4.
3. Implemente un programa que calcule la Cota de Johnson (Teorema 4.4.7).
4. Use el programa anterior para calcular cotas para $A_2(9,4)$, $A_2(8,3)$, $A_2(13,6)$ y $A_2(15,5)$ y compárelas con la Tabla 4.2

4.7. Biografía

Marcel Jules Edouard Golay **(1902-1989)**

Fue un matemático, físico e inventor suizo, estudió ingeniería eléctrica en el Instituto Tecnológico Federal (ETH) en Zúrich. En 1924, empezó a trabajar en los Laboratorios Bell y cuatro años después ingresó a la Universidad de Chicago, dónde obtuvo un doctorado en física en 1931. Luego, colaboró en el Departamento de Señales de la Armada Americana. Posteriormente, se vinculó a la compañía Perkin-Elmer como investigador senior. Parte de su investigación la dedicó al estudio de cromatografía de gases y la espectrografía óptica.

Entre sus principales aportes a la teoría de códigos se destaca el descubrimiento de los códigos binarios y ternarios de Golay en el artículo “Notes on digital coding"[13]. Este es un manuscrito de una sola página, pero en él Golay también dio la generalización no binaria de los códigos perfectos de Hamming y presentó por primera vez una matriz de control de paridad. Además, Golay en este documento también afirmaba que él creía que con su artículo todos los códigos perfectos habían sido encontrados, resultado que fue confirmado 20 años después después del trabajo de varios investigadores al rededor del tema, ver [21, 45] y sus referencias.

Golay publicó al rededor de 90 artículos y fue un inventor prolífico que poseía más de 41 patentes sobre sus conceptos. Las comunidades de cromatografía y la teoría de códigos lamentaron su fallecimiento, reconociendo sus aportes en cada una de las tareas, incluso al leer estos obituarios parecería dos documentos dedicados a personas diferentes; esto sin duda muestra la versatilidad de Golay, ver [1, 7, 27].

En 1951, recibió el Premio IEEE Harry Diamond Memorial que en la actualidad lo entrega la IEEE en Estados Unidos. Recibió numerosos premios en muchas otras áreas de la

ciencia, es particularmente llamativo que aunque no era químico, fue el destinatario de dos premios de la Sociedad Química Americana.

Foto tomada de ETHW.
https://ethw.org/Marcel_J._E._Golay.

5. Códigos cíclicos

En este capítulo exponemos una clase muy importante de códigos lineales, estos códigos son interesantes debido a su estructura y a sus propiedades. En especial se demuestra que un código cíclico es un ideal de anillo.

5.1. Definición y ejemplos

En el Capítulo 3 se vio la importancia de los códigos lineales debido a su estructura algebraica. En esta sección se estudiarán una clase especial de códigos lineales: los códigos cíclicos. Estos códigos tienen una estructura matemática más rica que facilita los procesos de codificación y decodificación. Además, como se verá a continuación, un código cíclico de longitud n es determinado totalmente por un polinomio de grado menor que n, denominado polinomio generador.

Comenzaremos con una definición de código cíclico basada en su estructura combinatoria, y más adelante veremos que dicha estructura admite una interpretación algebraica sencilla de asociar.

Definición 5.1.1 Un código lineal $C \subseteq \mathbb{F}_q^n$ es cíclico si

$$c_0c_1\cdots c_{n-2}c_{n-1} \in C \text{ implica que } c_{n-1}c_0c_1\cdots c_{n-2} \in C.$$

De esta forma, un código lineal C es cíclico si es cerrado bajo el desplazamiento cíclico, el cual envía la última componente a la primera en una palabra código; es decir

$$c_0c_1\cdots c_{n-2}c_{n-1} \in C \longrightarrow c_{n-1}c_0c_1\cdots c_{n-2} \in C.$$

En consecuencia, C es cerrado bajo cualquier desplazamiento cíclico, más específicamente, se tiene el siguiente resultado.

Proposición 5.1.2 Sean $C \subseteq \mathbb{F}_q^n$ un código cíclico y $c_0c_1\cdots c_{n-1} \in C$, entonces $c_k\cdots c_{n-1}c_0c_1\cdots c_{k-1} \in C$, para cualquier $1 \leq k \leq n-1$.

En particular, si C es un código cíclico, realizando $n-1$ desplazamientos cíclicos de la última coordenada hacia la primera, se observa que C también es cerrado con respecto a desplazamientos cíclicos de la primera componente hacia la última; esto es

$$c_0c_1\cdots c_{n-2}c_{n-1} \in C \longrightarrow c_1\cdots c_{n-2}c_{n-1}c_0 \in C.$$

Ejemplo 5.1.3

1. El código binario $C_1 = \{000, 001, 010, 100, 011, 110, 101, 111\}$ es un código cíclico. Observemos que por ejemplo

$$001 \in C_1 \longrightarrow 100 \in C_1,$$

$$011 \in C_1 \longrightarrow 101 \in C_1,$$

$$110 \in C_1 \longrightarrow 011 \in C_1.$$

2. El código

$$C_2 = \{00000000, 10101010, 01010101, 20202020, 02020202, 21212121, 12121212, 11111111, 22222222\}$$

es un código cíclico sobre $\mathbb{F}_3$. Observe que en particular tenemos que

$$10101010 \longleftrightarrow 01010101,$$

$$20202020 \longleftrightarrow 02020202,$$

$$21212121 \longleftrightarrow 12121212.$$

3. Por su parte el código ternario

$$C_3 = \{000, 220, 022, 202, 110, 101, 011, 212, 221, 122\},$$

aunque cumple la condición de ser cerrado para los desplazamientos cíclicos, no es un código cíclico porque no es un código lineal, puesto que por ejemplo $220 + 101 = 021 \notin C_3$. □

Enseguida presentaremos otra forma de caracterizar los códigos lineales, la cual permite identificar la estructura algebraica que los códigos cíclicos poseen, como se verá en el Teorema 5.1.5. Si C es un código lineal en $\mathbb{F}_q^n$, entonces a cada palabra código $\mathbf{c} = c_0c_1\cdots c_{n-1}$, se le asocia un elemento en el anillo cociente $R_n = \frac{\mathbb{F}_q[x]}{\langle x^n-1\rangle}$ de la siguiente manera:

$$\begin{aligned} \phi: \quad \mathbb{F}_q^n &\longrightarrow R_n = \frac{\mathbb{F}_q[x]}{\langle x^n - 1\rangle} \\ c_0c_1\cdots c_{n-1} &\longmapsto c_0 + c_1x + \cdots + c_{n-1}x^{n-1}. \end{aligned} \tag{5.1}$$

esto es $\phi(c_0c_1\cdots c_{n-1}) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$. No es difícil ver que ϕ es un isomorfismo de espacios vectoriales de C sobre el subespacio $\phi(C)$ de R_n. Esto demuestra que una palabra de un código lineal puede ser vista como un polinomio y viceversa. Esta identificación, representa una ventaja una vez que R_n además de ser un subespacio es un anillo; es decir es una $\mathbb{F}_q$-álgebra.

Ejemplo 5.1.4

1. Para las siguientes palabras del código C_1 en el Ejemplo 5.1.3, tenemos que su imagen respectiva bajo ϕ es:

$$\begin{aligned} \phi(001) &= 0 + 0x + x^2 = x^2, \\ \phi(100) &= 1 + 0x + 0x^2 = 1, \\ \phi(110) &= 1 + x + 0x^2 = 1 + x, \\ \phi(111) &= 1 + x + x^2. \end{aligned}$$

2. La imagen bajo ϕ del código C_2 en el Ejemplo 5.1.3 es

$$\begin{aligned}\phi(C_2) = \{&0,1+x^2+x^4+x^6,x+x^3+x^5+x^7,2+2x^2+2x^4+\\ &2x^6,2x+2x^3+2x^5+2x^7,2+x+2x^2+x^3+2x^4+\\ &x^5+2x^6+x^7,1+2x+x^2+2x^3+x^4+2x^5+x^6+\\ &2x^7,1+x+x^2+x^3+x^4+x^5+x^6+x^7,2+2x+\\ &2x^2+2x^3+2x^4+2x^5+2x^6+2x^7\}.\end{aligned}$$

□

Teorema 5.1.5 Sea ϕ la función dada por la expresión (5.1). Entonces, un código C en $\mathbb{F}_q^n$ es un código cíclico si y sólo si $\phi(C)$ es un ideal del anillo R_n.

Demostración. Supongamos que C es un código cíclico. Sean $p(x) = \phi(c_0c_1\ldots c_{n-1})$ y $q(x) = \phi(a_0a_1\ldots a_{n-1})$ elementos en $\phi(C)$, entonces $p(x) - q(x) = \phi(c_0c_1\ldots c_{n-1}) - \phi(a_0a_1\ldots a_{n-1}) = \phi(c_0c_1\ldots c_{n-1} - a_0a_1\ldots a_{n-1}) \in \phi(C)$, dado que C es un código lineal. Ahora, como $p(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1} = \phi(c_0c_1\cdots c_{n-1}) \in \phi(C)$, entonces

$$\begin{aligned}xp(x) &= c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + c_{n-1}x^n\\ &= c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1}\\ &= \phi(c_{n-1}c_0c_1\ldots c_{n-2}),\end{aligned}$$

es un elemento de $\phi(C)$, dado que C es cíclico. Similarmente, aplicando la idea anterior, se tiene que $x^2p(x) = x(xp(x))$ es también un elemento de $\phi(C)$. Por inducción, tenemos que $\alpha x^i p(x)$ pertenece a $\phi(C)$, para todo $i \geq 0$ y para cualquier $\alpha \in \mathbb{F}_q$. Por lo tanto, para $q(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} \in R_n$, el polinomio $q(x)p(x) = \sum_{i=0}^{n-1} a_i(x^ip(x))$, es un elemento de $\phi(C)$. Por lo tanto, $\phi(C)$ es un ideal de R_n.

Recíprocamente, supongamos que $\phi(C)$ es un ideal de R_n. Entonces, para cualquier $\alpha,\beta \in \mathbb{F}_q \subset R_n$ y $\mathbf{a},\mathbf{b} \in C$, se tiene que $\alpha\phi(\mathbf{a}) + \beta\phi(\mathbf{b}) \in \phi(C)$. Es decir, $\phi(\alpha\mathbf{a} + \beta\mathbf{b}) \in \phi(C)$, de donde, $\alpha\mathbf{a} + \beta\mathbf{b}$ es una palabra del código C. Luego, C es un código lineal.

Además, si $\mathbf{c} = c_0c_1\cdots c_{n-2}c_{n-1} \in C$, el polinomio

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

es un elemento de $\phi(C)$. Por tanto, como $\phi(C)$ es un ideal de R_n, el elemento

$$\begin{aligned} x(\phi(\mathbf{c})) &= c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + c_{n-1}x^n \\ &= c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1}, \end{aligned}$$

está en $\phi(C)$; es decir, que $c_{n-1}c_0c_1\cdots c_{n-2}$ es una palabra del código C. Por lo tanto, C es un código cíclico. ■

Ejemplo 5.1.6 Del resultado anterior, para el código C_2 del Ejemplo 5.1.4, se tiene que $\phi(C_2)$ es un ideal del anillo R_8. En este caso, como el código C_2 es de longitud 8, entonces $x^8 = 1$ en R_8, y por consiguiente $x^9 = x$, $x^{10} = x^2$, $x^{11} = x^3$, y así sucesivamente. Estas igualdades las usamos para verificar que al multiplicar por la derecha un elemento de $\phi(C_2)$ por un elemento de R_8, obtenemos nuevamente un elemento de $\phi(C_2)$. En efecto

$$\begin{aligned} x(1 + x^2 + x^4 + x^6) &= x + x^3 + x^5 + x^7 \in \phi(C_2), \\ (1 + x^2)(x + x^3 + x^5 + x^7) &= x + x^3 + x^5 + x^7 + x^3 + \\ &\qquad x^5 + x^7 + x^9 \\ &= x + 2x^3 + 2x^5 + 2x^7 + x^9 \\ &= 2x + 2x^3 + 2x^5 + 2x^7 \in \phi(C_2), \\ (x + x^2)(1 + x^2 + x^4 + x^6) &= x + x^3 + x^5 + x^7 + x^2 + \\ &\qquad x^4 + x^6 + x^8 \\ &= 1 + x + x^2 + x^3 + x^4 + \\ &\qquad x^5 + x^6 + x^7 \in \phi(C_2), \\ (1 + 2x^4)(2x + 2x^3 + 2x^5 + 2x^7) &= 2x + 2x^3 + 2x^5 + 2x^7 + \\ &\qquad x^5 + x^7 + x^9 + x^{11} \\ &= 3x + 3x^3 + 3x^5 + 3x^7 \\ &= 0 \in \phi(C_2). \end{aligned}$$

□

Dado que un código cíclico puede ser visto como subconjunto de $\mathbb{F}_q^n$ o como un ideal de R_n, a partir de este momento utilizaremos estas dos formas de representación para estudiar sus propiedades.

5.2. Polinomio generador y polinomio de control

Una de las ventajas de los códigos cíclicos radica en que sus elementos pueden describirse por completo mediante un polinomio de grado menor que n, conocido como polinomio generador, donde n es la longitud del código. A continuación, presentaremos las nociones de polinomio generador y polinomio de control, los cuales están interrelacionados. También examinaremos algunas propiedades que muestran las ventajas que ofrecen los códigos cíclicos.

Dado un anillo R, recordemos que en el anillo de polinomios $R[x]$, un polinomio $f(x) = a_0 + a_1x + \cdots + a_nx^n$ se denomina *mónico* si $a_n = 1$.

Teorema 5.2.1 En cualquier ideal no nulo C de R_n existe un único polinomio mónico $g(x)$ de grado minimal[a]. Además, este polinomio genera a C, es decir $C = \langle g(x) \rangle$.

[a] Es decir, es el polinomio mónico de menor grado en el ideal.

Demostración. Sea C un ideal de R_n, supongamos que C tiene dos polinomios mónicos $g_1(x)$ y $g_2(x)$ de grado minimal r. Entonces el polinomio no nulo $g_1(x) - g_2(x)$ pertenece a C. Además, como ambos polinomios son mónicos y de grado r, el grado de $g_1(x) - g_2(x)$ es menor que r, lo cual contradice el hecho de que estos dos polinomios son de grado minimal en C. Luego, existe un único polinomio $g(x)$ de grado mínimo r en C. Probemos ahora que dicho polinomio genera a C.

Como $g(x) \in C$ y C es un ideal de R_n, entonces $\langle g(x) \rangle \subseteq C$. Por otro lado, sea $c(x) \in C$, por el algoritmo de la división se tiene $c(x) = q(x)g(x) + r(x)$, con $q(x), r(x) \in R_n$, donde $r(x) = 0$ o $\text{grad}(r(x)) < \text{grad}(g(x)) = r$.

Supongamos que $r(x) \neq 0$. Luego $r(x) = c(x) - q(x)g(x) \in C$, lo cual contradice la minimalidad del grado de $g(x)$. En consecuencia, $r(x) = 0$ y así $p(x) \in \langle g(x) \rangle$, es decir $C \subseteq \langle g(x) \rangle$. Por lo tanto, se obtiene que $C = \langle g(x) \rangle$. ■

La propiedad que acabamos de probar significa que R_n es un anillo de ideales principales, ver [11, p. 318].

Definición 5.2.2 El único polinomio mónico de menor grado de un ideal no nulo I de R_n se denomina el *polinomio generador* de I. Para un código cíclico C en $\mathbb{F}_q^n$, el polinomio generador de $\phi(C)$ se llama también el polinomio generador de C.

Observación 5.2.3 Un código cíclico C puede ser generado por otros polinomios distintos del polinomio generador. Por lo tanto, se adopta la notación $C = \langle\langle g(x) \rangle\rangle$ para denotar el hecho de que C es el ideal generado por $g(x)$ y que este es el polinomio generador de C.

Teorema 5.2.4 Sea $C = \langle\langle g(x) \rangle\rangle$ un ideal no nulo de R_n, es decir un código cíclico de longitud n, con $l = \text{grad}(g(x))$.

1. El polinomio $g(x)$ divide a $x^n - 1$.
2. Cualquier elemento $c(x) \in C$ puede escribirse de forma única como $c(x) = r(x)g(x)$ en $\mathbb{F}_q[x]$, donde $r(x) \in \mathbb{F}_q[x]$ tiene grado menor que $n - l$. Además, la dimensión de C como espacio vectorial sobe $\mathbb{F}_q$ es $n - l$.
3. Si $g(x) = g_0 + g_1 x + \cdots + g_r x^l$. Entonces, $g_0 \neq 0$ y C tiene matriz generadora

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{n-l-1}g(x) \end{pmatrix} = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & & g_l & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & & g_l & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & & g_l & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & & g_l \end{pmatrix}, \tag{5.2}$$

donde cada fila de G es un desplazamiento cíclico de la fila anterior[a].

[a] Observemos que en la notación dada para la matriz G, un polinomio se está identificando con un vector, mediante la función ϕ dada por (5.1).

Demostración. Sea $C = \langle\langle g(x) \rangle\rangle$ un ideal no nulo de R_n.

1. Por el Algoritmo de la división, ver [11, Thm. 16.2],

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

donde $p(x), r(x) \in \mathbb{F}_q[x]$, con $r(x) = 0$ o $\text{grad}(r(x)) < \text{grad}(g(x)) = l$. En R_n esto implica que $r(x) = -p(x)g(x) \in C$. Entonces, dado que $g(x)$ es el polinomio de menor grado entre los elementos de C, tenemos que $r(x) = 0$. Por lo tanto, $g(x)|(x^n - 1)$.

2. Por hipótesis $C = \langle\langle g(x) \rangle\rangle$, en particular

$$C = \langle g(x) \rangle = \{f(x)g(x) : f(x) \in R_n\}, \tag{5.3}$$

con la reducción módulo $x^n - 1$. Probemos que es suficiente restringir $f(x)$ en (5.3) a los polinomios de grado menor que $n - l$.

Dado que $g(x)|(x^n - 1)$, tenemos que $x^n - 1 = h(x)g(x)$, para algún polinomio $h(x) \in \mathbb{F}_q[x]$ de grado $n - l$. Luego, dividiendo $f(x)$ por $h(x)$, tenemos que $f(x) = q(x)h(x) + r(x)$, donde $q(x), r(x) \in \mathbb{F}_q[x]$, con $r(x) = 0$ o $\text{grad}(r(x)) < n - l$. Entonces

$$\begin{aligned} f(x)g(x) &= q(x)h(x)g(x) + r(x)g(x) \\ &= q(x)(x^n - 1) + r(x)g(x), \end{aligned}$$

es decir, $f(x)g(x) = r(x)g(x)$ en R_n, como se quería probar. Por lo tanto, cualquier $c(x) \in C$ se puede escribir de manera única como $c(x) = r(x)g(x)$ en $\mathbb{F}_q[x]$, donde $r(x) \in \mathbb{F}_q[x]$ tiene grado menor que $n - l$. A partir de esto también se sigue que el conjunto $B = \{g(x), xg(x), \ldots, x^{n-l-1}g(x)\}$, genera a C como espacio vectorial. Adicionalmente, supongamos que en R_n

$$c_0\left(g(x)\right) + c_1\left(xg(x)\right) + \cdots + c_{n-l-1}\left(x^{n-l-1}g(x)\right) = 0,$$

donde $c_i \in \mathbb{F}_q$. Esto es, $h(x)g(x) \equiv 0 \pmod{x^n - 1}$, donde $h(x) = c_0 + c_1x + \cdots + c_{n-l-1}x^{n-l-1} \in \mathbb{F}_q[x]$. Como $\text{grad}(h(x)) + \text{grad}(g(x)) = n - 1$, se sigue que $h(x) = 0$, y así B es también linealmente independiente; por lo tanto la dimensión de C es $n - l$.

3. Sea $g(x) = g_0 + g_1x + \cdots + g_lx^l$ el polinomio generador de C. Si $g_0 = 0$, entonces $g(x)$ se puede escribir como $g(x) = xg_1(x)$, donde $\text{grad}(g_1(x)) < l$. Luego,

$$g_1(x) = 1 \cdot g_1(x) \equiv x^n g_1(x) = x^{n-1}g(x) \pmod{x^n - 1};$$

por tanto $g_1(x) \in C$, lo cual no es posible dado que ningún polinomio en C puede tener grado menor que l. Por lo tanto, $g_0 \neq 0$.

El hecho de que la matriz G, dada en (5.2), es una matriz generadora para C, se sigue de que el conjunto $\{g(x), xg(x), \ldots, x^{n-l-1}g(x)\}$ es una base de C, lo cual se demostró en la prueba del ítem anterior. ■

Ejemplo 5.2.5

1. En el código cíclico C_1 del Ejemplo 5.1.3 tenemos que el polinomio mónico minimal es $g(x) = 1$, el cual claramente es un divisor de $x^3 - 1$ sobre $\mathbb{F}_2$. De esta forma, este código tiene dimensión 3 y su matriz generadora está dada por

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ x^2g(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

2. Dado que la factorización del polinomio $x^8 - 1$ en $\mathbb{F}_3[x]$ es

$$x^8 - 1 = (1 + x)(2 + x)(1 + x^2)(2 + x + x^2)(2 + 2x + x^2), \tag{5.4}$$

entonces el polinomio $g(x) = (1 + x)(2 + x)(1 + x^2) = 2 + x^4$ es un divisor de $x^8 - 1$. Así, el ideal $C_4 = \langle\langle g(x) \rangle\rangle$ es un código cíclico de dimensión 4 con matriz generadora

$$G_4 = \begin{pmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ x^3g(x) \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

3. Observemos que $g_2(x) = 1 + x^2 + x^4 + x^6 = (1 + x^2)(2 + x + x^2)(2 + 2x + x^2)$ es el polinomio mónico de menor grado del código C_2 del Ejemplo 5.1.4. Además, tenemos que $g_2(x)$

es un divisor de $x^8 - 1$. Así, $\phi(C_2) = \langle\langle g_2(x) \rangle\rangle$ es un código cíclico de dimensión 2 sobre $\mathbb{F}_3$. □

Teorema 5.2.6 Un polinomio mónico $p(x)$ en R_n es el polinomio generador para un código cíclico si y sólo si $p(x)|(x^n - 1)$.

Demostración. Una de las implicaciones ya se tiene por el Teorema 5.2.4. Para la otra implicación, supongamos que $p(x)$ es un polinomio mónico tal que $p(x)|(x^n - 1)$. Sea $g(x)$ el polinomio generador para el código $C = \langle p(x) \rangle$, probaremos que $p(x) = g(x)$. En efecto, asuma que $p(x) \neq g(x)$, dado que $p(x)$ y $g(x)$ son ambos mónicos, por el Teorema 5.2.1 debe cumplirse que $\text{grad}(g(x)) < \text{grad}(p(x))$. Por otro lado,

$$x^n - 1 = p(x)f(x), \tag{5.5}$$

para algún polinomio $f(x) \in \mathbb{F}_q[x]$. Además, dado que $g(x) \in \langle p(x) \rangle$, se tiene que

$$g(x) = a(x)p(x),$$

para algún $a(x) \in R_n$. Multiplicando ambos lados de la anterior igualdad por $f(x)$ y utilizando la igualdad (5.5) se obtiene

$$g(x)f(x) \equiv a(x)p(x)f(x) \equiv a(x)(x^n - 1) \equiv 0 \pmod{x^n - 1}.$$

Sin embargo, $\text{grad}(g(x)f(x)) < \text{grad}(p(x)f(x)) = n$, y en consecuencia, $g(x)f(x) = 0$ en $\mathbb{F}_q[x]$. Como $g(x) \neq 0$, esto implica que $f(x) = 0$, dado que $\mathbb{F}_q[x]$ es un domino entero. Pero esto no es posible según la expresión (5.5). Por lo tanto, $p(x) = g(x)$. ■

Corolario 5.2.7 Existe una correspondencia uno a uno entre los códigos cíclicos en $\mathbb{F}_q^n$ y los divisores mónicos de $x^n - 1 \in \mathbb{F}_q[x]$.

Demostración. Definimos una función π del conjunto D_n de todos los divisores mónicos de $x^n - 1$ en el conjunto C_n de todos los códigos cíclicos en R_n.

Para esto consideremos

$$\begin{aligned} \pi : D_n &\longrightarrow C_n \\ g(x) &\longmapsto \langle\langle g(x) \rangle\rangle, \end{aligned}$$

es decir que envía cada divisor mónico $g(x)$ de $x^n - 1$ al código cíclico $\langle\langle g(x) \rangle\rangle$. Del Teorema 5.2.6, se tiene que la función π es biyectiva, dado que cada divisor mónico de $x^n - 1$ genera un único código cíclico en R_n, además todo código cíclico en R_n debe ser generado por un divisor mónico de $x^n - 1$. ■

A partir del anterior resultado se nota la importancia de factorizar el polinomio $x^n - 1$ en $\mathbb{F}_q[x]$, con el fin de determinar los polinomios generadores de los códigos cíclicos que se pueden construir en R_n. El siguiente teorema nos permite determinar el número de códigos cíclicos de longitud n sobre $\mathbb{F}_q$.

Teorema 5.2.8 Suponga que $x^n - 1 \in \mathbb{F}_q[x]$ tiene la factorización

$$x^n - 1 = \prod_{i=1}^{r} p_i^{e_i}(x), \tag{5.6}$$

donde $p_1(x), p_2(x), \ldots, p_r(x)$ son polinomios mónicos irreducibles diferentes y $e_i \geq 1$ para todo $i = 1, 2, \ldots, r$. Entonces, existen $\prod_{i=1}^{r}(e_i + 1)$ códigos cíclicos de longitud n sobre $\mathbb{F}_q$.

Demostración. Por el Corolario 5.2.7, para determinar el número de códigos cíclicos de longitud n sobre $\mathbb{F}_q$, debemos contar el número de divisores mónicos de $x^n - 1$. Observemos que para un polinomio irreducible $p_i(x)$ dado, el polinomio $p_i^{e_i}(x)$ tiene $e_i + 1$ divisores mónicos:

$$1, p_i(x), p_i^2(x), \ldots, p_i^{e_i}(x);$$

los cuales, según (5.15), también dividen a $x^n - 1$. Por lo tanto, el resultado se sigue considerando los r polinomios irreducibles en la factorización (5.15), con sus respectivos grados. ■

Observación 5.2.9 Sea p la característica de $\mathbb{F}_q$, note que si n y q no son primos relativos, entonces n se puede escribir

como $n = mp^k$, donde $(m,q) = 1$. Entonces,

$$x^n - 1 = x^{mp^k} - 1 = (x^m - 1)^{p^k},$$

y por tanto $x^n - 1$ tiene factores repetidos en su descomposición en factores primos.
Por otro lado, en el caso que n y q son primos relativos, el polinomio $x^n - 1$ no tiene raíces múltiples en ninguna extensión de $\mathbb{F}_q$, dado que su derivada, es decir $D[x^n - 1] = nx^{n-1}$, no tiene factores comunes con $x^n - 1$. Por lo tanto, en este caso $x^n - 1$ se factoriza como

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

donde $p_1(x), p_2(x), \ldots, p_r(x)$, son polinomios mónicos irreducibles diferentes. En consecuencia, por el Teorema 5.2.8, el número de códigos cíclicos de longitud n sobre $\mathbb{F}_q$ en este caso es 2^r.

Ejemplo 5.2.10 Con el fin de determinar todos los códigos cíclicos de longitud 8 sobre $\mathbb{F}_3$, debemos tener en cuenta la factorización (5.4) del polinomio $x^8 - 1$ sobre $\mathbb{F}_3[x]$. Los factores en (5.4) son irreducibles, dado que ninguno tiene raíces en $\mathbb{F}_3$. Por el Teorema 5.2.8, existen $2^5 = 32$ códigos cíclicos de longitud 8 sobre $\mathbb{F}_3$.

También, mediante el ítem 2 del Teorema 5.2.4 se puede concluir que existen únicamente seis $[8,4]$-códigos cíclicos sobre $\mathbb{F}_3$, que son los generados por los polinomios

$$\begin{aligned}
f_1(x) &= (1+x)(2+x)(1+x^2),\\
f_2(x) &= (1+x)(2+x)(2+x+x^2),\\
f_3(x) &= (1+x)(2+x)(2+2x+x^2),\\
f_4(x) &= (1+x^2)(2+x+x^2),\\
f_5(x) &= (1+x^2)(2+2x+x^2),\\
f_6(x) &= (2+x+x^2)(2+2x+x^2).
\end{aligned} \tag{5.7}$$

Por último, uno de los cuatro $[8,2]$-códigos cíclicos ternarios, está generado por el polinomio

$$g_2(x) = (1+x^2)(2+x+x^2)(2+2x+x^2),$$

esto es el código C_2 del Ejemplo 5.1.3. De esta forma, podemos ver que $\phi(21212121) = (2+x)g(x)$. □

Definición 5.2.11 Sea $g(x)$ el polinomio generador de un $[n, n-l]$-código cíclico. Dado que $g(x)$ divide a $x^n - 1$ en R_n, se tiene que

$$x^n - 1 = g(x)h(x),$$

donde $h(x) \in \mathbb{F}_q[x]$ es un polinomio de grado $n-l$. El polinomio $h(x)$ se denomina el *polinomio de control* de C.

Antes de mencionar algunas propiedades del polinomio de control de un código cíclico, se presenta otra definición.

Definición 5.2.12 Sea $h(x) = \sum_{i=0}^{k} a_i x^i$ un polinomio de grado k sobre $\mathbb{F}_q$. El *polinomio recíproco* de $h(x)$ se define como el polinomio dado por

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

Ejemplo 5.2.13 El polinomio recíproco del polinomio $h(x) = 2 + x^2 + x^4 + x^6 \in \mathbb{F}_3[x]$ es

$$\begin{aligned} h_R(x) = x^6 h(1/x) &= x^6 \left(2 + \left(\frac{1}{x}\right)^2 + \left(\frac{1}{x}\right)^4 + \left(\frac{1}{x}\right)^6\right) \\ &= 1 + x^2 + x^4 + 2x^6. \end{aligned}$$

□

Lema 5.2.14 Si C es un código cíclico, entonces, el código dual $C^\perp$ es también un código cíclico.

Demostración. Sean C un código cíclico y $\mathbf{x} = x_0 x_1 \cdots x_{n-1} \in C^\perp$. Entonces, como para cualquier $\mathbf{c} = c_0 c_1 \cdots c_{n-1} \in C$ se tiene que

$\mathbf{c}' = c_1c_2\cdots c_{n-1}c_0$ también está en C, tenemos que

$$\begin{aligned} 0 = \mathbf{c}'\cdot\mathbf{x} &= c_1x_0 + c_2x_1 + \cdots + c_{n-1}x_{n-2} + c_0x_{n-1} \\ &= c_0x_{n-1} + c_1x_0 + c_2x_1 + \cdots + c_{n-1}x_{n-2}. \end{aligned}$$

Luego, si $\mathbf{x}' = x_{n-1}x_0x_1\cdots x_{n-2}$, entonces $\mathbf{c}\cdot\mathbf{x}' = 0$. De esta manera, $\mathbf{x}' \in C^\perp$ y por lo tanto $C^\perp$ es también un código cíclico. ■

Teorema 5.2.15 Sea $h(x)$, con $\operatorname{grad}(h(x)) = n - l$, el polinomio de control para un código cíclico C en R_n.

1. El código C puede ser descrito como
$$C = \{p(x) \in R_n : p(x)h(x) \equiv 0\}.$$
2. Si $h(x) = h_0 + h_1x + \cdots + h_{n-l}x^{n-l}$, entonces una matriz de control de paridad para C es
$$H = \begin{pmatrix} h_R(x) \\ xh_R(x) \\ x^2h_R(x) \\ \vdots \\ x^{n-l-1}h_R(x) \end{pmatrix} = \begin{pmatrix} h_{n-l} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_{n-l} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & 0 & h_{n-l} & \cdots & h_0 & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \cdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & h_{n-l} & \ldots & h_0 \end{pmatrix}.$$
3. Además, el código dual $C^\perp$ está dado por $C^\perp = \langle\langle h_0^{-1}h_R(x)\rangle\rangle$, y su dimensión es l.

Demostración.

1. Sea $g(x)$ el polinomio generador de C. Si $p(x) \in C$, entonces, $p(x) = f(x)g(x)$ para algún polinomio $f(x) \in R_n$. Por lo tanto, en R_n
$$p(x)h(x) = f(x)g(x)h(x) = f(x)(x^n - 1) = 0.$$
Recíprocamente, si $p(x) \in R_n$ y $p(x)h(x) = 0$, entonces, por el algoritmo de la división
$$p(x) = q(x)g(x) + r(x),$$

donde $r(x) = 0$ o $\text{grad}(r(x)) < \text{grad}(g(x)) = l$.

Multiplicando por $h(x)$ se obtiene

$$\begin{aligned} p(x)h(x) &= q(x)g(x)h(x) + r(x)h(x) \\ 0 &= q(x)(x^n - 1) + r(x)h(x), \end{aligned}$$

de donde, $r(x)h(x) \equiv 0$ mód $(x^n - 1)$. Sin embargo, $\text{grad}(r(x)h(x)) < l + (n - l) = n$. En consecuencia $r(x) = 0$, y por lo tanto $p(x) = q(x)g(x) \in C$.

2. Por el ítem 1, tenemos que $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$ es una palabra código si y sólo si $c(x)h(x) = 0$ en R_n. Para que esto suceda, en particular se debe tener que los coeficientes de $x^{n-l}, x^{n-l+1}, \ldots, x^{n-1}$ en el producto $c(x)h(x)$ deben ser todos iguales a 0; es decir

$$\begin{aligned} c_0h_{n-l} + c_1h_{n-l-1} + \cdots + c_{n-l}h_0 &= 0 \\ c_1h_{n-l} + c_2h_{n-l-1} + \cdots + c_{n-l+1}h_0 &= 0 \\ \vdots \qquad\qquad & \qquad \vdots \\ c_{l-1}h_{n-l} + c_lh_{n-l-1} + \cdots + c_{n-1}h_0 &= 0. \end{aligned}$$

Esto es equivalente a la igualdad $(c_0, c_1, \ldots, c_{n-1})H^\top = \mathbf{0}$. Así, H genera un código C' que es ortogonal a C; es decir $C' \subseteq C^\perp$. Sin embargo, dado que $h_{n-l} \neq 0$, se sigue que la dimensión de $C^\perp$ es l, y por lo tanto $C' = C^\perp$.

3. Dado que $h(x)g(x) = x^n - 1$, entonces

$$h\left(\frac{1}{x}\right) g\left(\frac{1}{x}\right) = \frac{1}{x^n} - 1.$$

Esto implica que

$$x^{n-l}h\left(\frac{1}{x}\right) x^l g\left(\frac{1}{x}\right) = 1 - x^n,$$

o equivalentemente, $h_R(x)$ divide a $x^n - 1$. Por los teoremas 5.2.6 y 5.2.4, $h_0^{-1}h_R(x)$ es el polinomio generador de un código cíclico con matriz generadora H. Así, de la demostración del ítem anterior tenemos que $C^\perp = \langle\langle h_0^{-1}h_R(x) \rangle\rangle$ y además $\dim(C^\perp) = l$. ■

Ejemplo 5.2.16

1. Dado que en $\mathbb{F}_3[x]$,

$$x^8 - 1 = (1+x)(2+x)(1+x^2)(2+x+x^2)(2+2x+x^2),$$

se sigue que el polinomio $h(x) = 2 + x^2 = (1+x)(2+x)$ es un polinomio de control del código $C_2 = \langle\langle (1+x^2)(2+x+x^2)(2+2x+x^2) \rangle\rangle$. En consecuencia, $h_R(x) = 1 + 2x^2$.

Así tenemos que $C_2^\perp = \langle\langle 2h_R(x) \rangle\rangle = \langle\langle 2 + x^2 \rangle\rangle$, y que una matriz de control de paridad de C_2 es

$$H_2 = \begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \end{pmatrix}.$$

2. Para los $[8,4]$-códigos ternarios generados por los polinomios en (5.7), se tiene que su polinomio de control y el polinomio generador del código dual respectivo deben estar en este mismo listado. Por ejemplo, para el código cíclico C generado por $f_3(x) = 1 + x + x^2 + 2x^3 + x^4$ se tiene que el polinomio de control es $h(x) = f_4(x) = 2 + x + x^3 + x^4$, mientras que $2h_R(x) = 2 + 2x + 2x^3 + x^4$ es el polinomio generador del código $C^\perp$. En consecuencia, una matriz generadora y una matriz de control de paridad de C están dadas por

$$G = \begin{pmatrix} 1 & 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 1 \end{pmatrix} \text{ y}$$

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

3. Para el polinomio $g_5(x) = (1+x)(1+x^2) = 1 + x + x^2 + x^3$ en $R_8 = \mathbb{F}_3[x]/\langle x^8 - 1 \rangle$, su polinomio de control es

$$h_5(x) = (2+x)(2+x+x^3)(2+2x+x^2) = 2 + x + 2x^4 + x^5.$$

Así, una matriz generadora y una matriz de control del código $C_5 = \langle\langle g_5(x) \rangle\rangle$, están dadas por

$$G_5 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \text{ y}$$

$$H_5 = \begin{pmatrix} 2 & 1 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 & 2 & 1 \end{pmatrix}. \qquad \square$$

5.3. Codificación con códigos cíclicos

Existen dos maneras de codificar mensajes utilizando códigos cíclicos: una *sistemática* y otra *no sistemática*. La primera corresponde a la codificación previamente estudiada para códigos lineales. Por otro lado, la segunda aprovecha la estructura específica de los códigos cíclicos.

Sea $C = \langle\langle g(x) \rangle\rangle \subset \mathbb{F}_q[x]/\langle x^n - 1 \rangle$, con $\operatorname{grad}(g(x)) = l$. De acuerdo al Teorema 5.2.4, C es un $[n, n-l]$-código cíclico. Entonces, de la Observación 3.2.2 sabemos que C permite codificar mensajes fuente de longitud $n - l$, agregando l símbolos de redundancia. Esto puede ser realizado en dos formas que presentamos a continuación.

La codificación no sistemática es la descrita en la Observación 3.2.2. Es decir, un mensaje fuente $\mathbf{m} \in \mathbb{F}_q^{n-l}$ se codifica como la palabra código $\mathbf{c} = \mathbf{m}G \in \mathbb{F}_q^n$, donde G es una matriz generadora para C, la cual está dada por el Teorema 5.2.4. Es posible que esta matriz esté en la forma estándar, obteniendo un código sistemático en las primeras $n - l$ posiciones. Sin embargo, en general este no es el caso y por tanto dicha codificación es llamada no sistemática.

Ejemplo 5.3.1

1. La codificación no sistemática para el código $C_2 = \langle\langle x^6 + x^4 + x^2 + 1 \rangle\rangle \subset \mathbb{F}_3[x]/\langle x^8 - 1 \rangle$, con matriz generadora

$$G_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

codifica el mensaje $\mathbf{m} = 10$, como $\mathbf{m}G_2 = 10101010$. En este caso observamos que esta matriz está en la forma estándar.

2. Por su parte para el código $C_5 = \langle\langle 1 + x + x^2 + x^3 \rangle\rangle \subset \mathbb{F}_3[x]/\langle x^8 - 1\rangle$, del Ejemplo 5.2.16, tenemos que la codificación de los mensajes $\mathbf{m}_1 = 11101$ y $\mathbf{m}_2 = 01001$ está dada por

$$\begin{aligned}\mathbf{m}_1 G_5 &= (1,1,1,0,1)\begin{pmatrix}1&1&1&1&0&0&0&0\\0&1&1&1&1&0&0&0\\0&0&1&1&1&1&0&0\\0&0&0&1&1&1&1&0\\0&0&0&0&1&1&1&1\end{pmatrix}\\ &= (1,2,0,0,0,2,1,1),\\ \mathbf{m}_2 G_5 &= (0,1,0,0,1)\begin{pmatrix}1&1&1&1&0&0&0&0\\0&1&1&1&1&0&0&0\\0&0&1&1&1&1&0&0\\0&0&0&1&1&1&1&0\\0&0&0&0&1&1&1&1\end{pmatrix}\\ &= (0,1,1,1,2,1,1,1).\end{aligned}$$

Observemos que la codificación es no sistemática, dado que los dígitos del mensaje fuente no corresponden a los primeros dígitos de la palabra código obtenida. □

Para obtener una codificación sistemática, dado un mensaje fuente $\mathbf{a} = a_0 a_1 \cdots a_{n-l-1} \in \mathbb{F}_q^{n-l}$, se forma el polinomio mensaje

$$\bar{a}(x) = a_0 x^{n-1} + a_1 x^{n-2} + \cdots + a_{n-l-1} x^l.$$

Observemos que $\bar{a}(x)$ no tiene términos de grado menor que r. Ahora, se divide $\bar{a}(x)$ por $g(x)$,

$$\bar{a}(x) = q(x)g(x) + s(x), \text{ con } \operatorname{grad}(s(x)) < l$$

y se codifica a $\bar{a}(x)$ como $c(x) = \bar{a}(x) - s(x) = q(x)g(x) \in C$. Dado que $\bar{a}(x)$ y $s(x)$ no tienen términos del mismo grado, la codificación es sistemática. En efecto, si se lee los términos de un polinomio del código, de grado mayor a menor, se observa que las primeras $n - l$ coordenadas son los símbolos de información y que las l restantes son símbolos de redundancia.

Ejemplo 5.3.2

1. Sea C_2 el $[8,2]$-código sobre $\mathbb{F}_3$ del Ejemplo 5.2.10, el cual es generado por $g_2(x) = (1+x^2)(2+x+x^2)(2+2x+x^2) = x^6+x^4+x^2+1$. Es decir

$$C_2 = \{00000000, 10101010, 01010101, 20202020, 02020202, 21212121, 12121212, 11111111, 22222222\}.$$

Como podemos ver, este código codifica los nueve mensajes de longitud 2 sobre $\mathbb{F}_3$, es decir $\{00,10,01,20,02,12,21,11,22\}$, añadiendo seis dígitos de redundancia, obteniendo así un código de longitud 8.

En este caso, para realizar una codificación sistemática, consideremos los polinomios mensaje $\bar{a}_1(x)$, $\bar{a}_2(x)$, $\bar{a}_3(x)$, $\bar{a}_4(x)$, $\bar{a}_5(x)$, $\bar{a}_6(x)$, $\bar{a}_7(x)$, $\bar{a}_8(x)$ y $\bar{a}_9(x)$, mostrados a continuación y dividimos cada uno entre $g_2(x)$. Mediante el algoritmo de la división se obtiene:

$$\begin{aligned}
\bar{a}_1(x) &= 0 = (x^6+x^4+x^2+1)(0) + 0,\\
\bar{a}_2(x) &= x^7 = (x^6+x^4+x^2+1)(x) + (2x^5+2x^3+2x),\\
\bar{a}_3(x) &= x^6 = (x^6+x^4+x^2+1)(1) + (2x^4+2x^2+2),\\
\bar{a}_4(x) &= 2x^7 = (x^6+x^4+x^2+1)(2x) + (x^5+x^3+x),\\
\bar{a}_5(x) &= 2x^6 = (x^6+x^4+x^2+1)(2) + (x^4+x^2+1),\\
\bar{a}_6(x) &= x^7+2x^6 = (x^6+x^4+x^2+1)(x+2)\\
&\qquad + (2x^5+x^4+2x^3+x^2+2x+1),\\
\bar{a}_7(x) &= 2x^7+x^6 = (x^6+x^4+x^2+1)(2x+1)\\
&\qquad + (x^5+2x^4+x^3+2x^2+x+2),\\
\bar{a}_8(x) &= x^7+x^6 = (x^6+x^4+x^2+1)(x+1)\\
&\qquad + (2x^5+2x^4+2x^3+2x^2+2x+2),\\
\bar{a}_9(x) &= 2x^7+2x^6 = (x^6+x^4+x^2+1)(2x+2)\\
&\qquad + (x^5+x^4+x^3+x^2+x+1).
\end{aligned}$$

Entonces $\bar{a}_i(x)$ se codifica como $c_i(x) = \bar{a}_i(x) - s_i(x)$, donde $s_i(x)$ es el residuo de dividir $\bar{a}_i(x)$ entre $g_2(x)$.

Por lo tanto, se tiene la codificación

$$\begin{aligned}
\bar{a}_1(x) &\to c_1(x) = 0\\
\bar{a}_2(x) &\to c_2(x) = x^7 + x^5 + x^3 + x\\
\bar{a}_3(x) &\to c_3(x) = x^6 + x^4 + x^2 + 1\\
\bar{a}_4(x) &\to c_4(x) = 2x^7 + 2x^5 + 2x^3 + 2x\\
\bar{a}_5(x) &\to c_5(x) = 2x^6 + 2x^4 + 2x^2 + 2\\
\bar{a}_6(x) &\to c_6(x) = x^7 + 2x^6 + x^5 + 2x^4 + x^3 + 2x^2 + x + 2\\
\bar{a}_7(x) &\to c_7(x) = 2x^7 + x^6 + 2x^5 + x^4 + 2x^3 + x^2 + 2x + 1\\
\bar{a}_8(x) &\to c_8(x) = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1\\
\bar{a}_9(x) &\to c_9(x) = 2x^7 + 2x^6 + 2x^5 + 2x^4 + 2x^3 + 2x^2 + 2x + 2.
\end{aligned}$$

En conclusión, el proceso de codificación es la siguiente:

$$\begin{aligned}
00 &\longrightarrow \bar{a}_1(x) \longrightarrow c_1(x) \longrightarrow 00000000\\
10 &\longrightarrow \bar{a}_2(x) \longrightarrow c_2(x) \longrightarrow 10101010\\
01 &\longrightarrow \bar{a}_3(x) \longrightarrow c_3(x) \longrightarrow 01010101\\
20 &\longrightarrow \bar{a}_4(x) \longrightarrow c_4(x) \longrightarrow 20202020\\
02 &\longrightarrow \bar{a}_5(x) \longrightarrow c_5(x) \longrightarrow 02020202\\
12 &\longrightarrow \bar{a}_6(x) \longrightarrow c_6(x) \longrightarrow 12121212\\
21 &\longrightarrow \bar{a}_7(x) \longrightarrow c_7(x) \longrightarrow 21212121\\
11 &\longrightarrow \bar{a}_8(x) \longrightarrow c_8(x) \longrightarrow 11111111\\
22 &\longrightarrow \bar{a}_9(x) \longrightarrow c_9(x) \longrightarrow 22222222.
\end{aligned}$$

En este ejemplo podemos verificar que la codificación es sistemática. En efecto, los dos primeros dígitos en cada palabra código corresponden a la información que se desea transmitir, es decir al mensaje fuente y los últimos seis son dígitos de redundancia, con el fin de detectar y corregir errores.

2. La codificación sistemática de los mensajes 11101 y 01001 en el código C_5 del ítem 2 del Ejemplo 5.3.1 está dada por

$$\begin{aligned}
11101 \longrightarrow \bar{a}_1(x) \longrightarrow c_1(x) &= x^7 + x^6 + x^5 + x^3 + x^2 + x + 2\\
&\longrightarrow 11101112,\\
01001 \longrightarrow \bar{a}_2(x) \longrightarrow c_2(x) &= x^6 + x^3 + x^2 + x + 2\\
&\longrightarrow 01001112.
\end{aligned}$$

□

5.4. Decodificación con códigos cíclicos

Dado que todo código cíclico es lineal, se puede utilizar la decodificación por síndrome estudiada en la Sección 3.6, pero en su versión polinomial. Si $c(x) \in C$ es una palabra código enviada (en forma polinómica) y $u(x)$ es el polinomio recibido, entonces, $e(x) = u(x) - c(x)$ es el *polinomio de error*. Además, anotemos que consideraremos como el *peso de un polinomio* al número de coeficientes no nulos que este posee.

Definición 5.4.1 Sea $C = \langle\langle g(x) \rangle\rangle$ un $[n, n-l]$-código cíclico. El *síndrome de un polinomio* $u(x)$, denotado con $\text{syn}(u(x))$, es el residuo de dividir $u(x)$ entre $g(x)$, es decir

$$u(x) = q(x)g(s) + \text{syn}(u(x)) \text{ con } \text{grad}(\text{syn}(x)) < l.$$

Observación 5.4.2 La definición de síndrome para un polinomio coincide con la dada para palabras código en códigos lineales. En efecto, sea $s_i(x)$ el residuo de dividir x^{l+i} entre $g(x)$, es decir

$$x^{l+i} = a_i(x)g(x) + s_i(x), \tag{5.8}$$

con $\text{grad}(s_i(x)) < l$. Probemos que el conjunto $B = \{x^{l+i} - s_i(x) : i = 0, 1, \ldots, n-l-1\}$ es una base para C.
Sea $f(x) \in \langle B \rangle$, entonces

$$f(x) = \sum_{i=0}^{n-l-1} b_i(x^{l+i} - s_i(x)) = \sum_{i=0}^{n-l-1} b_i a_i(x)g(x),$$

donde $b_i \in \mathbb{F}_q$ para todo i. Por tanto, $f(x) \in C$ y así $\langle B \rangle \subseteq C$.
Ahora probemos que el conjunto B es linealmente independiente. Sean $\alpha_0, \alpha_1, \ldots, \alpha_{n-l-1} \in \mathbb{F}_q$ tales que

$$\alpha_0(x^l - s_0(x)) + \cdots + \alpha_{n-l-1}(x^{n-1} - s_{n-l-1}(x)) = 0,$$

entonces $\alpha_0 x^l + \cdots + \alpha_{n-l-1}x^{n-1} - t(x) = 0$, donde $t(x) = -\sum_{i=0}^{n-l-1} \alpha_i s_i(x)$ es un polinomio de grado menor que l. Por lo tanto, $\alpha_i = 0$ para todo i.

Como $\dim(C) = n - l = \dim(\langle B \rangle)$ y $\langle B \rangle \subseteq C$, entonces $C = \langle B \rangle$; en consecuencia B es una base para C. A partir de la base anterior de C tenemos que una matriz generadora para este código es

$$G = \begin{pmatrix} x^l - s_0(x) \\ x^{l+1} - s_1(x) \\ \vdots \\ x^{n-1} - s_{n-l-1}(x) \end{pmatrix}$$
$$= \begin{pmatrix} -s_{0,0} & -s_{0,1} & \cdots & -s_{0,l-1} & 1 & 0 & \cdots & 0 \\ -s_{1,0} & -s_{1,1} & \cdots & -s_{1,l-1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -s_{n-l-1,0} & -s_{n-l-1,1} & \cdots & -s_{n-l-1,l-1} & 0 & 0 & \cdots & 1 \end{pmatrix},$$

donde $s_i(x) = \sum_{j=0}^{l-1} s_{i,j} x^j$.

Considerando la matriz anterior como $G = (S \mid I_{n-l})$, mediante un razonamiento similar al de la Observación 3.5.4, una matriz de control de paridad para C es $H = (I_l \mid -S^\top)$, donde $S^\top$ es la transpuesta de S, es decir

$$H = \begin{pmatrix} 1 & 0 & \cdots & 0 & s_{0,0} & s_{1,0} & \cdots & s_{n-l-1,0} \\ 0 & 1 & \cdots & 0 & s_{0,1} & s_{1,1} & \cdots & s_{n-l-1,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & s_{0,l-1} & s_{1,l-1} & \cdots & s_{n-l-1,l-1} \end{pmatrix}.$$

Entonces, para $\mathbf{u} = u_0 u_1 \cdots u_{n-1} \in \mathbb{F}_q^n$, la j-ésima coordenada de su síndrome es

$$\mathbf{u} H_j^\top = u_j + \sum_{i=0}^{n-l-1} u_{l+i} s_{i,j}, \tag{5.9}$$

donde $H_j^\top$ denota la j-ésima columna de $H_j^\top$.

Por otro lado, en términos de polinomios $\mathbf{u}$ se puede ver como

$$u(x) = \sum_{j=0}^{l-1} u_j x^j + \sum_{i=0}^{n-l-1} u_{l+i} x^{l+i}.$$

Sin embargo, de la expresión (5.8) se obtiene que

$$\begin{aligned} u(x) &= \sum_{j=0}^{l-1} u_j x^j + \sum_{i=0}^{n-l-1} u_{l+i}\left(a_i(x)g(x) + s_i(x)\right) \\ &= g(x)\sum_{i=0}^{n-l-1} u_{l+i}a_i(x) + \left(\sum_{j=0}^{l-1} u_j x^j + \sum_{i=0}^{n-l-1} u_{l+i}s_i(x)\right) \\ &= g(x)q(x) + t(x), \end{aligned}$$

donde $q(x) = \sum_{i=0}^{n-l-1} u_{l+i}a_i(x)$ y además el grado del polinomio $t(x) = \sum_{j=0}^{l-1} u_j x^j + \sum_{i=0}^{n-l-1} u_{l+i}s_i(x)$ es menor que l. Por lo tanto, $t(x) = \text{syn}(u(x))$; es decir es el síndrome del polinomio $u(x)$. Adicionalmente, el coeficiente de x^j es

$$u_j + \sum_{i=0}^{n-l-1} u_{l+i}s_{ij}. \tag{5.10}$$

A partir de las expresiones (5.9) y (5.10) se concluye que las dos definiciones de síndrome son equivalentes.

Ahora se presenta el proceso de decodificación para un código cíclico. La prueba del siguiente resultado se deja para el lector (ver Ejercicio 10).

Teorema 5.4.3 Sea C un código cíclico. Si se recibe $u(x)$ codificado con C, entonces

1. $u(x)$ es una palabra código si y sólo si $\text{syn}(u(x)) = 0$.
2. Dos polinomios tienen el mismo síndrome si y sólo si están en la misma clase lateral de C.

Luego, la forma polinomial de decodificación por síndrome es análoga a la vectorial, veamos esto con el siguiente ejemplo.

Ejemplo 5.4.4 Nuevamente, utilicemos el código C_2 del Ejemplo 5.2.10. Dado que este es un código lineal con $d(C_2) = wt(C_2) = 4$, entonces es capaz de corregir a lo sumo un error. Es decir que únicamente puede corregir errores donde el polinomio error tiene peso uno. Los líderes de las clases laterales y su correspondiente síndrome están en la Tabla 5.1.

Líder	Síndrome	Líder	Síndrome
0	0	2	2
1	1	$2x$	$2x$
x	x	$2x^2$	$2x^2$
x^2	x^2	$2x^3$	$2x^3$
x^3	x^3	$2x^4$	$2x^4$
x^4	x^4	$2x^5$	$2x^5$
x^5	x^5	$2x^6$	x^4+x^2+1
x^6	$2x^4+2x^2+2$	$2x^7$	x^5+x^3+x
x^7	$2x^5+2x^3+2x$		

Tabla 5.1: Tabla líder-síndrome código cíclico.

Supongamos que se envía la palabra código 10101010, sin embargo, en la transmisión ocurre un error en la sexta coordenada y se recibe la secuencia $\mathbf{u} = 10101110$. En términos de polinomios se recibe $\phi(\mathbf{u}) = u(x) = x^6 + x^5 + x^4 + x^2 + 1$ y como

$$u(x) = x^6 + x^5 + x^4 + x^2 + 1 = (x^6 + x^4 + x^2 + 1)(1) + x^5,$$

entonces, $\text{syn}(u(x)) = x^5$ y en la tabla anterior se observa que el líder de esta clase lateral es $a(x) = x^5$. Por lo tanto, $u(x)$ se decodifica como

$$\begin{aligned} c(x) &= u(x) - a(x) = (x^6 + x^5 + x^4 + x^2 + 1) - (x^5) \\ &= x^6 + x^4 + x^2 + 1, \end{aligned}$$

es decir que $\mathbf{u} = 10101110$ se decodifica como $\mathbf{c} = 10101010$, corrigiendo efectivamente el error ocurrido. □

Ejemplo 5.4.5 Considere el código binario cíclico $L = \langle\langle 1 + x^2 + x^3 \rangle\rangle$ de longitud 7 y distancia mínima 3. Supongamos que se se recibió la palabra 0011110. Entonces tenemos que el síndrome del polinomio $u(x) = x^2 + x^3 + x^4 + x^5$ es $\text{syn}(u(x)) = x^2 + 1$. De esta manera, como se tiene también que el síndrome de x^3 es igual a $x^2 + 1$, entonces la palabra enviada corresponde al polinomio $u(x) - x^3 = x^2 + x^4 + x^5$; es decir la palabra enviada fue 0011110. □

Hasta el momento no hemos evidenciado ninguna ventaja para la decodificación derivada del hecho que el código sea cíclico. La principal dificultad de esta forma de decodificación es que las tablas líder-síndrome son bastante grandes, como se observa en el Ejemplo 5.4.4. Justamente, es en esta situación donde mostraremos cómo podemos sacar provecho al trabajar con códigos cíclicos para superar esta dificultad. Supongamos que tenemos algún método para decodificar el coeficiente principal de cualquier palabra recibida $u(x)$; es decir el coeficiente de x^{n-1}, el cual puede o no ser distinto de cero. Luego, podemos decodificar el coeficiente principal de $u(x)$, realizar un desplazamiento cíclico módulo $x^n - 1$, y decodificar el nuevo coeficiente principal, que es el coeficiente de x^{n-2} en $u(x)$. Repitiendo este proceso se decodifica toda la palabra. Este método ahorra tiempo porque sólo se necesitan las filas de la tabla lidere-síndrome que contienen líderes de grado $n - 1$.

Un método para decodificar el coeficiente de x^{n-1}, empleando una tabla construida únicamente con los líderes de clase lateral de grado $n - 1$, consistiría en calcular el síndrome de la palabra recibida $u(x)$; si este no aparece en la tabla se concluye que el coeficiente de x^{n-1} es correcto. De lo contrario, si el síndrome de $u(x)$ aparece en la tabla se deduce que existe un error en el coeficiente de x^{n-1}, por tanto se cambia dicho coeficiente por uno de los elementos restantes en $\mathbb{F}_q$ y se vuelve a calcular el síndrome de la palabra resultante. El anterior proceso se repite hasta que al cambiar el coeficiente de x^{n-1} en la palabra recibida, el síndrome de la palabra obtenida no aparezca en la tabla, pues este nuevo coeficiente debe ser el correcto.

Observación 5.4.6 Se concluye que el coeficiente de x^{n-1} es correcto en una palabra recibida $u(x)$, cuando su síndrome no aparece en la tabla con los líderes de clase lateral de grado $n - 1$, porque el líder de clase lateral corresponde al error y si el síndrome de $u(x)$ no aparece en la tabla significa que el o los errores están en los coeficientes de los términos de grado menor que $n - 1$. A esta técnica se le conoce como *decodificación por captura de errores* (o error trapping decoding). Mayor información se puede encontrar en [22, Ch. 7].

Ejemplo 5.4.7 Para el caso del Ejemplo 5.4.4, los únicos líderes de clase lateral con peso 1 y grado $n-1=7$ son x^7 y $2x^7$, por lo tanto sólo se necesitan dos filas en la Tabla 5.2.

Líder	Síndrome
x^7	$2x^5+2x^3+2x$
$2x^7$	x^5+x^3+x

Tabla 5.2: Tabla líder-síndrome para términos de grado 7.

Nuevamente, si se recibe $u(x)=x^6+x^5+x^4+x^2+1$. Dado que $\text{syn}(u(x))=x^5$ no está en la Tabla 5.2, se asume que el coeficiente líder de $u(x)$, es decir el de x^7 es correcto. En seguida se realiza un desplazamiento cíclico en $u(x)$

$$x(x^6+x^5+x^4+x^2+1)\equiv x^7+x^6+x^5+x^3+x(\text{mód}\, x^8-1),$$

y se calcula su síndrome, el cual es x^6. Igualmente, este término no está en la Tabla 5.2, por tanto el coeficiente de x^6 en $u(x)$ es correcto. Nuevamente, se realiza otro desplazamiento cíclico y se calcula su síndrome,

$$x(x^7+x^6+x^5+x^3+x)\equiv x^7+x^6+x^4+x^2+1(\text{mód}\, x^8-1), \tag{5.11}$$

cuyo síndrome es $2x^5+2x^3+2x$ y si está en la Tabla 5.2, por tanto se deduce que el coeficiente de x^5 en $u(x)$ es incorrecto. Se tienen dos opciones, cambiar el coeficiente de x^7 en la parte derecha de (5.11) por 0 o por 2. Si se cambia por 2, se obtiene la palabra $2x^7+x^6+x^4+x^2+1$, cuyo síndrome es x^5+x^3+x y también aparece en la Tabla 5.2, por lo tanto el coeficiente adecuado debe ser 0, el cual lo reemplazamos en la palabra recibida por el coeficiente de x^5.

Continuando de esta manera, se decodifica $u(x)$ como $c(x)=x^6+x^4+x^2+1$. Lo anterior coincide con la decodificación que se encontró en el Ejemplo 5.4.4. □

Polinomio	Síndrome
$u(x)$	x^2+1
$xu(x)$	x^2+x+1
$x^2u(x)$	$x+1$
$x^3u(x)$	x^2+x
$x^4u(x)$	1
$x^5u(x)$	x
$x^6u(x)$	x^2

Tabla 5.3: Síndrome de desplazamientos cíclicos de $u(x)$.

Ejemplo 5.4.8 Para aplicar esta técnica al código $L = \langle\langle 1 + x^2 + x^3 \rangle\rangle$ del Ejemplo 5.4.5. Dado que el único polinomio de peso 1 es x^6, debemos calcular únicamente su síndrome; esto es $\text{syn}(x^6) = x^2 + x$. En consecuencia, si recibimos la palabra $u(x) = x^2 + x^3 + x^4 + x^5$, debemos calcular el síndrome de cada uno de sus desplazamiento cíclicos. Esta información la resumimos en la siguiente Tabla 5.3. En consecuencia vemos que la primera potencia con síndrome igual al de x^6 es $x^3u(x)$, esto significa que hay un error en el coeficientes de x^3 de $u(x)$, de esta manera cambiamos $u(x)$ por $w(x) = x^2 + x^4 + x^5$. Si repetimos este procedimiento para los polinomios $x^i w(x)$, vemos que todas tienen síndrome igual a 0; es decir pertenecen al código. Esto significa que la palabra enviada fue $w(x)$. □

5.5. Algunas familias de códigos cíclicos

En esta sección presentamos dos familias de códigos cíclicos: código de Golay binario y ternario y códigos de residuos cuadráticos. Existen otras familias de códigos cíclicos, de las cuales no hablaremos aquí, como son: los códigos BCH, códigos de Reed-Muller, códigos de Reed-Solomon, entre otros.

5.5.1. Códigos de Hamming

La familia de los códigos de Hamming es posiblemente la más famosa en la teoría de códigos correctores de errores. Fueron des-

cubiertos independientemente por Marcel Golay [13] en 1949 y Richard Hamming [16] en 1950. El propósito del código es que en la transmisión de un mensaje exista suficiente redundancia de modo que incluso si se produce un ruido, el receptor es capaz de corregir automáticamente el error.

Dado el cuerpo finito $\mathbb{F}_q$, vamos a construir una matriz $\mathcal{H}_{q,r}$ tal que cualquier par de sus columnas sean linealmente independientes, pero que al menos un conjunto de tres columnas sea linealmente dependiente. Recordemos que por el Teorema 3.1.7, el número de subespacios de dimensión 1 de $\mathbb{F}_q^r$ es igual a $n = [r]_q = \dfrac{q^r - 1}{q - 1}$. Sea $\mathcal{H}_{q,r}$ la matriz cuyas columnas están formadas por un vector no nulo de cada uno de los miembros de esta familia. La matriz $\mathcal{H}_{q,r}$ se denomina *matriz de Hamming de orden r*.

Definición 5.5.1 Sean $r \in \mathbb{Z}^+$ con $r \geq 2$ y q una potencia de un primo. El código $\mathrm{Ham}(q,r)$ con $\mathcal{H}_{q,r}$ como matriz de control de paridad se denomina *código de Hamming q-ario de orden r*. Es decir,

$$\mathrm{Ham}(q,r) = \left\{ \mathbf{x} \in \mathbb{F}_q^n : \mathbf{x}\mathcal{H}_{q,r}^\top = \mathbf{0} \right\},$$

donde $n = \dfrac{q^r - 1}{q - 1}$.

A continuación, encontramos los parámetros de $\mathrm{Ham}(q,r)$.

Teorema 5.5.2 El código de Hamming $\mathrm{Ham}(q,r)$ es un $[n, n-r, 3]$, donde $n = \dfrac{q^r - 1}{q - 1}$.

Demostración. Por lo anterior, la longitud de $\mathrm{Ham}(q,r)$ es igual a n. Ahora, observemos que en las columnas de $\mathcal{H}_{q,r}$ hay un representante de cada uno de los subespacios generados por los vectores canónicos de $\mathbb{F}_q^r$: $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_r$. De esta forma, la dimensión del espacio fila de $\mathcal{H}_{q,r}$ es r, lo que significa que la dimensión del código $\mathrm{Ham}(q,r)$ es igual a $n - r$.

Dado que dos vectores no nulos $\mathbf{u},\mathbf{v} \in \mathbb{F}_q^n$ son linealmente dependientes si y sólo si existe $0 \neq \alpha \in \mathbb{F}_q$ tal que $\mathbf{u} = \alpha\mathbf{v}$. De esta manera, cualquier par de columnas de $\mathcal{H}_{q,r}$ son linealmente independientes. Además,si $\mathbf{u}$ y $\mathbf{v}$ son dos columnas cualesquiera de $\mathcal{H}_{q,r}$, entonces el vector $\mathbf{u}+\mathbf{v}$ debe pertenecer a alguno de los subespacios de dimensión 1 generado por una de las columnas de $\mathcal{H}_{q,r}$; esto implica que estas tres columnas son linealmente dependientes. Así por el Teorema 3.5.7, se tiene que la distancia mínima es $d(\mathrm{Ham}(q,r)) = 3$. ■

Observación 5.5.3 Debemos anotar que la forma de seleccionar las columnas de la matriz de Hamming $\mathcal{H}_{q,r}$ no es única, y así existen tanto matrices de Hamming como códigos de Hamming diferentes, para un conjunto de parámetros dados; esto es q y $r \geq 2$. Sin embargo, fijados los parámetros q y r, cualquier matriz de Hamming se puede obtener a partir de cualquier otra permutando las columnas y multiplicando algunas columnas por escalares no nulos. En consecuencia, estas matriz forman códigos q-arios equivalentes.
El caso binario $\mathrm{Ham}(2,r)$, para $q = 2$, por el Teorema 5.5.2 es un $[2^r-1, 2^r-1-r, 3]$-código. Dado que cada espacio de dimensión 1 de $\mathbb{F}_2^r$ tienen únicamente dos vectores, se tiene que las columnas de la matriz de Hamming de orden r son todos los vectores de $\mathbb{F}_2^r$ en algún orden. Equivalentemente, tenemos que las columnas de $\mathrm{Ham}(2,r)$ son la representación binaria (en base 2) de los primeros 2^r-1 números positivos.

Ejemplo 5.5.4 Para construir el código ternario de orden 2, se puede tomar como matriz de control de paridad la matriz

$$\mathcal{H}_{3,2} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 2 & 2 & 0 & 1 \end{pmatrix}.$$

De esta forma, $\mathrm{Ham}(3,2)$ es un $[4,2,3]$-código y

$$\begin{aligned}\mathrm{Ham}(3,2) = \{&(0,0,0,0),(1,0,1,1),(2,0,2,2),(0,1,1,2),(1,1,2,0),\\ &(2,1,0,1),(0,2,2,1),(1,2,0,2),(2,2,1,0)\}.\end{aligned}$$ □

Teorema 5.5.5 El código $\mathrm{Ham}(q,r)$ es perfecto.

Demostración. Por los teoremas 2.8.10 y 5.5.2, es suficiente probar que $V_q(n,1) = q^r$, lo cual es cierto porque por el Lema 2.5 tenemos $V_q(n,1) = 1 + \binom{n}{1}(q-1) = 1 + (q^r - 1) = q^r$. ■

Ejemplo 5.5.6 El código binario $\mathrm{Ham}(2,3)$ tiene como matriz de control de paridad la matriz $\mathcal{H}_{2,3} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$. Por lo tanto, las ecuaciones de control de paridad de este código son

$$\begin{aligned} x_2 + x_3 + x_4 + x_5 &= 0 \\ x_1 + x_3 + x_4 + x_6 &= 0 \\ x_1 + x_2 + x_4 + x_7 &= 0. \end{aligned}$$

□

Se puede demostrar que el código binario de Hamming $\mathrm{Ham}(2,r)$ es equivalente a un código cíclico, ver [25, Ch. 7, Sect. 3, Thm. 2].

El código dual código de Hamming $\mathrm{Ham}(q,r)$, que se denota con $\mathrm{Sim}(q,r)$, se denomina *código Simplex*. Dado que $\mathcal{H}_2(r)$ es un $[(q^r-1)/(q-1), (q^r-1)/(q-1) - r]$-código, entonces el código $\mathrm{Sim}(q,r)$ tiene parámetros $[(q^r-1)/(q-1), r]$. Más aún, una matriz de control de paridad de $\mathcal{H}_2(r)$ es también una matriz generadora de $\mathrm{Sim}(q,r)$.

Teorema 5.5.7 Sea $\mathrm{Sim}(q,r)$ el código Simplex con parámetros $[(q^r-1)/(q-1), r]$, entonces se tienen las siguientes propiedades.

1. Cada palabra no nula en $\mathrm{Sim}(q,r)$ tiene peso q^{r-1}. Es decir, que $\mathrm{Sim}(q,r)$ es un $[(q^r-1)/(q-1), r, q^{r-1}]$-código.
2. El código $\mathrm{Sim}(2,r)$ es equivalente a un código cíclico.

Demostración. Dado que $\mathrm{Sim}(q,r) = \left\{ \mathbf{x}\mathcal{H}_{q,r} : \mathbf{x} \in \mathbb{F}_q^r \right\}$, entonces el peso de Hamming de una palabra $\mathbf{x}\mathcal{H}_{q,r}$ es igual a

$$(q^r - 1)/(q-1) - s,$$

donde s es el número de coordenadas nulas en $\mathbf{x}\mathcal{H}_{q,r}$, o lo que es lo mismo el número de columnas $\mathbf{y}$ de $\mathcal{H}_{q,r}$ tal que $\mathbf{x}\cdot\mathbf{y}^T = 0$. Estos últimos vectores pertenecen al dual del espacio generado por $\mathbf{x}$. Así, $s = |\{\mathbf{y} \in \mathbb{F}_q^r : \mathbf{y} \in \langle\mathbf{x}\rangle^\perp\}|$.

Ahora como la dimensión de $\langle\mathbf{x}\rangle^\perp$ es $r-1$, entonces basta contar el número de subespacios de dimensión $r-1$ en $\mathbb{F}_q^r$; esto es $s = [r-1]_q = \frac{q^{r-1}-1}{q-1}$. Por lo tanto, $wt(\mathbf{x}\mathcal{H}_{q,r}) = \frac{q^r-1}{q-1} - \frac{q^{r-1}-1}{q-1} = q^{r-1}$. ■

5.5.2. Códigos de Golay

Los códigos de Golay son probablemente los más importantes en el desarrollo de la teoría de códigos, debido a sus múltiples aplicaciones prácticas y sus notables propiedades teóricas. Fueron descubiertos por el matemático y teórico de la información Marcel J. E. Golay en 1949. En su investigación, Golay describió cuatro códigos lineales: dos binarios y dos ternarios. Aunque estos códigos son cíclicos, mostraremos cómo se definen en términos de matrices generadoras.

5.5.3. Códigos binarios de Golay

El código de Golay $\mathcal{G}_{24}$ es un $[24,12,8]$-código binario con matriz generadora $G = (I_{12}|A)$, donde

$$A = \left(\begin{array}{c|cccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right). \tag{5.12}$$

Note la estructura cíclica de la matriz A sin tener en cuenta la primera fila y la primera columna. Además, cada fila de la matriz

G tiene peso igual a 8 o a 12, esto implica que el producto interior de cualquier par de filas de G es un número par.

Teorema 5.5.8 El código $\mathcal{G}_{24}$ es auto-dual, es decir $\mathcal{G}_{24} = \mathcal{G}_{24}^{\perp}$.

Demostración. Si $\mathbf{x}$ y $\mathbf{y}$ son dos filas de G no necesariamente distintas, entonces $\mathbf{x} \cdot \mathbf{y} \equiv 0 \mod 2$; así $\mathcal{G}_{24} \subseteq \mathcal{G}_{24}^{\perp}$. Como $\dim(\mathcal{G}_{24}) = 12$, entonces $\dim(\mathcal{G}_{24}^{\perp}) = 24 - \dim(\mathcal{G}_{24}) = 12$ y por lo tanto $\mathcal{G}_{24} = \mathcal{G}_{24}^{\perp}$. ■

Como $A^{\top} = A$ y $\mathcal{G}_{24}$ es auto-dual, tenemos el siguiente resultado.

Corolario 5.5.9 El código de Golay $\mathcal{G}_{24}$ también tiene como matriz generadora a $G = (A|I_{12})$.

Del corolario anterior, tenemos que si $(\mathbf{a}, \mathbf{b}) \in \mathcal{G}_{24}$, donde $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^{12}$ entonces $(\mathbf{b}, \mathbf{a}) \in \mathcal{G}_{24}$.

Teorema 5.5.10 Si $\mathbf{c} \in \mathcal{G}_{24}$, entonces $wt(\mathbf{c})$ es divisible por 4.

Demostración. Sean $\mathbf{x}$ y $\mathbf{y}$ dos filas de G, entonces del Lema 2.5.3, tenemos

$$wt(\mathbf{x} + \mathbf{y}) = wt(\mathbf{x}) + wt(\mathbf{y}) - 2wt(\mathbf{x} \cap \mathbf{y}).$$

Ahora, como $wt(\mathbf{x} \cap \mathbf{y}) \equiv \mathbf{x} \cdot \mathbf{y} \equiv 0 \pmod 2$, entonces $wt(\mathbf{x} + \mathbf{y})$ es divisible por 4. De forma inductiva, se cumple que el peso de cualquier combinación lineal de las filas de G es divisible por 4. ■

Teorema 5.5.11 El código $\mathcal{G}_{24}$ no tiene palabras de peso 4.

Demostración. Por el Corolario 5.5.9, tenemos que $G_1 = (I_{12}|A)$ y $G_2 = (A|I_{12})$ son matrices generadoras para $\mathcal{G}_{24}$. Sea $\mathbf{c} = (\mathbf{a}, \mathbf{b}) \in \mathcal{G}_{24}$, donde $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^{12}$ y supongamos que $wt(\mathbf{c}) = 4$. En cualquier

combinación lineal no trivial de las filas de G_1 y G_2 tenemos que $wt(\mathbf{a}) \geq 1$ y $wt(\mathbf{b}) \geq 1$, respectivamente. Ahora, si $wt(\mathbf{a}) = 1$ o $wt(\mathbf{b}) = 1$, entonces $\mathbf{c}$ es una fila de G_1 o G_2, respectivamente, pero ninguna de estas filas tiene peso 4, por lo tanto, la única posibilidad es que $wt(\mathbf{a}) = 2 = wt(\mathbf{b})$. Lo anterior implica que $\mathbf{c}$ es la suma de dos filas de G_1, sin embargo se puede comprobar que ninguna de estas sumas tienen peso igual a 4. ■

Como resultado inmediato de los dos teoremas anteriores obtenemos el siguiente resultado.

Corolario 5.5.12 La distancia mínima del código $\mathcal{G}_{24}$ es 8.

Demostración. Como $wt(\mathcal{G}_{24}) \geq 8$ y la segunda fila de G tiene peso 8, entonces la distancia mínima de $\mathcal{G}_{24}$ es 8. ■

Por los teoremas 5.5.10 y 5.5.11, tenemos que los posibles pesos para las palabras de $\mathcal{G}_{24}$ pertenecen al conjunto $\{0,8,12,16,20,24\}$. Como la palabra cuyas componentes son todas iguales a 1 pertenece a $\mathcal{G}_{24}$ (se obtiene sumando las filas de la matriz generadora) entonces la distribución de pesos de $\mathcal{G}_{24}$ satisface

$$A_0 = A_{24} = 1,\ A_8 = A_{16},\ A_i = 0 \text{ para } i \notin \{0,8,12,16,24\}.$$

Para determinar el polinomio enumerador de pesos de $\mathcal{G}_{24}$ podemos usar la identidad de MacWilliams, ver Teorema 3.4.8, la cual toma la siguiente forma

$$W_{\mathcal{G}_{24}^{\perp}}(x) = \frac{1}{|\mathcal{G}_{24}|}(1+x)^{24} W_{\mathcal{G}_{24}}\left(\frac{1-x}{1+x}\right).$$

Teniendo en cuenta que $\mathcal{G}_{24}$ es auto-dual, la anterior ecuación se puede expresar así

$$\sum_{k=0}^{24} A_k (1-x)^k (1+x)^{24-k} = |\mathcal{G}_{24}|\, W_{\mathcal{G}_{24}}(x). \tag{5.13}$$

Teorema 5.5.13 El polinomio enumerador de pesos del código de Golay $\mathcal{G}_{24}$ es

$$W_{\mathcal{G}_{24}}(x) = 1 + 759x^8 + 2576x^{12} + 759x^{16} + x^{24}.$$

Demostración. Sabemos que $A_0 = A_{24}$, $A = A_8 = A_{16}$, $B = A_{12}$ y $A_i = 0$ para todo $i \notin \{0,8,12,16,24\}$, entonces $W_{\mathcal{G}_{24}}(x) = 1 + Ax^8 + Bx^{12} + Ax^{16} + x^{24}$. Como $W_{\mathcal{G}_{24}}(1) = |\mathcal{G}_{24}| = 2^{12} = 4096$ entonces $B = 4094 - 2A$. Al reemplazar este valor en la Ecuación 5.13 e igualar los coeficientes de x^{22} en ambos miembros de la ecuación resultante tenemos

$$2\binom{24}{2} + 2\left[\binom{8}{6} - \binom{8}{7}\binom{16}{15} + \binom{16}{14}\right]A - \binom{12}{11}(4094 - 2A) = 0,$$

lo cual se reduce a

$$552 + 40A - 12(4094 - 2A) = 0,$$

de donde obtenemos $A = 759$ y por tanto $B = 2576$. ■

El siguiente resultado fue probado en forma conjunta por V. Pless en 1968 [31] y por P. Delsarte y J. M. Goethals en 1975 [9]. Pless demostró que cualquier código binario lineal con los mismos parámetros que $\mathcal{G}_{24}$ es equivalente a $\mathcal{G}_{24}$, mientras que Delsarte y Goethals probaron que cualquier código con los mismos parámetros de $\mathcal{G}_{24}$ es necesariamente lineal.

Teorema 5.5.14 Cualquier $(24,2^{12},8)$-código binario es equivalente al código de Golay $\mathcal{G}_{24}$.

El código de Golay $\mathcal{G}_{23}$ es el código que se obtiene al eliminar una componente de cada palabra código de $\mathcal{G}_{24}$, por lo tanto $\mathcal{G}_{23}$ es un $[23,12,7]$-código lineal binario. En [31] y [9] también probaron la unicidad de $\mathcal{G}_{23}$.

Teorema 5.5.15 Cualquier $(23,2^{12},7)$-código binario es equivalente al código de Golay $\mathcal{G}_{23}$.

Proposición 5.5.16 El código $\mathcal{G}_{23}$ es un código perfecto.

Demostración. Por el Teorema 2.8.10, es suficiente probar que $|\mathcal{G}_{23}| \cdot \sum_{k=0}^{3} \binom{23}{k} = 2^{23}$. En efecto,

$$|\mathcal{G}_{23}| \cdot \sum_{k=0}^{3} \binom{23}{k} = 2^{12}(1 + 23 + 23 \cdot 11 + 23 \cdot 11 \cdot 7)$$
$$= 2^{12} \cdot 2^{11} = 2^{23}.$$

■

En el siguiente resultado se encuentra el polinomio enumerador de pesos del código $\mathcal{G}_{23}$, cuya prueba puede ser consultada en [25, Ch. 2, Thm. 27].

Teorema 5.5.17 El polinomio enumerador de pesos de $\mathcal{G}_{23}$ es

$$W_{\mathcal{G}_{23}}(x) = 1 + 253x^7 + 506x^8 + 1288x^{11} + 1288x^{12} + 506x^{15} + 253x^{16} + x^{23}.$$

5.5.4. Códigos ternarios de Golay

El código de Golay $\mathcal{G}_{12}$ es un $[12,6,6]$-código ternario con matriz generadora $G = (I_6|A)$, donde

$$A = \left(\begin{array}{c|cccccc} 0 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{array}\right). \tag{5.14}$$

De manera similar, al caso binario, el código de Golay $\mathcal{G}_{11}$ es el código que se obtiene al eliminar una componente de cada palabra código de $\mathcal{G}_{12}$, por lo tanto $\mathcal{G}_{11}$ es un $[11,6,5]$-código lineal ternario.

Algunas de las propiedades de los códigos de Golay ternarios se resumen a continuación. Su prueba se deja para el lector en el Ejercicio 17.

Teorema 5.5.18 Los códigos $\mathcal{G}_{12}$ y $\mathcal{G}_{11}$ satisfacen las siguientes propiedades:

1. $\mathcal{G}_{12}$ es auto-dual.

2. La distancia mínima de $\mathcal{G}_{12}$ es 6.
3. $\mathcal{G}_{11}$ es un código perfecto.

Los polinomios enumeradores de pesos de los códigos de Golay ternarios se muestran en el siguiente teorema.

Teorema 5.5.19 El polinomio enumerador de pesos del código $\mathcal{G}_{12}$ es

$$W_{\mathcal{G}_{12}}(x) = 1 + 264x^6 + 440x^9 + 24x^{12}.$$

El polinomio enumerador de pesos del código $\mathcal{G}_{11}$ es

$$W_{\mathcal{G}_{11}}(x) = 1 + 132x^5 + 132x^6 + 330x^8 + 110x^9 + 24x^{11}.$$

Al igual que con los binarios, V. Pless [31] y P. Delsarte y J. M. Goethals [9] también probaron la unicidad de los códigos de Golay ternarios.

Teorema 5.5.20 Cualquier $(12, 2^6, 6)-$código ternario es equivalente al código de Golay $\mathcal{G}_{12}$, y cualquier $(11, 2^6, 5)-$código ternario es equivalente al código de Golay $\mathcal{G}_{11}$.

5.5.5. Códigos de residuos cuadráticos

Definición 5.5.21 Sean p un número primo impar y $a \in \mathbb{F}_p$ tal que $\gcd(a, p) = 1$. Si la congruencia cuadrática $x^2 \equiv a \pmod{p}$ tiene solución, entonces se dice que a es un *residuo cuadrático módulo* p. En caso contrario se dice que a es un *no residuo cuadrático módulo* p.

Ejemplo 5.5.22 Para encontrar cuáles de los elementos de $\mathbb{F}_{13}$ son residuos cuadráticos basta recorrer los elementos $a \in \mathbb{F}_{13}$ y ver cuales satisfacen la definición anterior, esto es tales que $x^2 \equiv a \pmod{p}$. Equivalentemente, para determinar los residuos cuadráticos módulo 13, podemos recorrer todos

los elementos de $\mathbb{F}_{13}$ y ver sus cuadrados módulo 13

$$\begin{aligned}
1^2 &\equiv 12^2 \equiv 1 \pmod{13},\\
2^2 &\equiv 11^2 \equiv 4 \pmod{13},\\
3^2 &\equiv 10^2 \equiv 9 \pmod{13},\\
4^2 &\equiv 9^2 \equiv 3 \pmod{13},\\
5^2 &\equiv 8^2 \equiv 12 \pmod{13},\\
6^2 &\equiv 7^2 \equiv 10 \pmod{13}.
\end{aligned}$$

En consecuencia, los residuos cuadráticos de 13 son: 1, 3, 4, 9, 10 y 12; mientras que los no residuos cuadráticos son: 2, 5, 6, 7, 8 y 11. □

Para un primo impar p con $\mathcal{Q}_p$ denotaremos el conjunto de residuos cuadráticos módulo p y con $\mathcal{N}_p$ el conjunto de residuos no cuadráticos módulo p.

Proposición 5.5.23 Sea p un primo impar. Entonces se satisfacen las siguientes propiedades.

1. El producto de dos residuos cuadráticos módulo p es un residuo cuadrático módulo p.
2. El producto de dos residuos no cuadráticos módulo p es un residuo cuadrático módulo p.
3. El producto de un residuo cuadrático módulo p con un residuo no cuadrático módulo p es un residuo no cuadrático módulo p.
4. Se tiene que $|\mathcal{Q}_p| = \frac{p-1}{2}$ y $|\mathcal{N}_p| = \frac{p-1}{2}$. Además, $\mathbb{F}_p = \{0\} \bigcup \mathcal{Q}_p \bigcup \mathcal{N}_p$.
5. Para $\alpha \in \mathcal{Q}_p$ y $\beta \in N_p$ se satisface que

$$\begin{aligned}
\alpha\mathcal{Q}_p &= \{\alpha r : r \in \mathcal{Q}_p\} = \mathcal{Q}_p,\\
\beta\mathcal{Q}_p &= \{\beta r : r \in \mathcal{Q}_p\} = \mathcal{N}_p,\\
\alpha\mathcal{Q}_p &= \{\alpha n : n \in \mathcal{N}_p\} = \mathcal{N}_p,\\
\beta\mathcal{Q}_p &= \{\beta n : n \in \mathcal{N}_p\} = \mathcal{Q}_p.
\end{aligned}$$

Sea q un primo tal que $q \neq p$ y q es un residuo cuadrático módulo p. Sea $m \in \mathbb{Z}^+$ tal que $q^m \equiv 1 \pmod{p}$. Sean θ un elemento primitivo de $\mathbb{F}_{q^m}$ y $\alpha = \theta^{(q^m-1)/p}$. Entonces el orden de α en $\mathbb{F}_{q^m}$ es p. Esto implica que los elementos $1 = \alpha^0 = \alpha^p, \alpha = \alpha^1, \alpha^2, \alpha^3, ..., \alpha^{p-1}$

son distintos dos a dos en $\mathbb{F}_{q^m}$; es decir son todas las raíces del polinomio $x^p - 1$ en $\mathbb{F}_{q^m}$. Entonces

$$x^p - 1 = \prod_{i=0}^{p-1} (x - \alpha^i). \tag{5.15}$$

Ahora considere los polinomios

$$g_{\mathcal{Q}}(x) = \prod_{r \in \mathcal{Q}_p} (x - \alpha^r), \tag{5.16}$$

$$g_{\mathcal{N}}(x) = \prod_{n \in \mathcal{N}_p} (x - \alpha^n). \tag{5.17}$$

Observemos que por definición tenemos que $g_{\mathcal{Q}}(x), g_{\mathcal{N}}(x) \in \mathbb{F}_{q^m}[x]$. En el siguiente resultado, en realidad demostramos que estos polinomios en realidad pertenecen al anillo de polinomios $\mathbb{F}_q[x]$.

Lema 5.5.24 Los polinomios $g_{\mathcal{Q}}(x)$ y $g_{\mathcal{N}}(x)$ pertenecen a $\mathbb{F}_q[x]$. Además,

$$x^p - 1 = (x-1) g_{\mathcal{Q}}(x) g_{\mathcal{N}}(x). \tag{5.18}$$

Demostración. Es suficiente demostrar que los coeficientes de $g_{\mathcal{Q}}(x)$ y $g_{\mathcal{N}}(x)$ pertenecen a $\mathbb{F}_q$. Supongamos que $g_{\mathcal{Q}}(x) = a_0 + a_1 x + \cdots + a_k x^k$, donde $a_i \in \mathbb{F}_{q^m}$ y $k = \frac{p-1}{2}$.

Dado que $\mathbb{F}_{q^m}$ es un cuerpo finito de característica q, por automorfismo de Frobenius, al elevar a la q-ésima potencia cada coeficiente de $g_{\mathcal{Q}}(x)$ obtenemos que

$$\begin{aligned} a_0^q + a_1^q x + \cdots + a_k^q x^k &= \prod_{r \in \mathcal{Q}_p} (x - \alpha^{qr}) \\ &= \prod_{j \in q\mathcal{Q}_p} (x - \alpha^j) = \prod_{j \in \mathcal{Q}_p} (x - \alpha^j) = g_{\mathcal{Q}}(x), \end{aligned}$$

dado que por el ítem 5 de la Proposición 5.5.23 $q\mathcal{Q}_p = \mathcal{Q}_p$.

En consecuencia, $a_i^q = a_i$, para todo $0 \leq i \leq k$, lo que implica que todos los coeficientes de $g_{\mathcal{Q}}(x)$ son elementos de $\mathbb{F}_q$,

ver [22, Cor. 3.3.2]. Argumentos similares se pueden usar para probar que $g_{\mathcal{N}}(x) \in \mathbb{F}_q[x]$. ■

Definición 5.5.25 Sean q y p primos distintos tales que q es un residuo cuadrático módulo p y $m \in \mathbb{Z}^+$ tal que $q^m \equiv 1 \pmod{p}$. Sea θ un elemento primitivo de $\mathbb{F}_{q^m}$ y $\alpha = \theta^{(q^m-1)/p}$. Los códigos cíclicos $C_{\mathcal{Q}} = \langle\langle g_{\mathcal{Q}}(x)\rangle\rangle$ y $C_{\mathcal{N}} = \langle\langle g_{\mathcal{N}}(x)\rangle\rangle$ se denominan *códigos q-arios de residuos cuadráticos*.

Observación 5.5.26 Los códigos $C_{\mathcal{Q}}$ y $C_{\mathcal{N}}$ son $[p, \frac{p+1}{2}]$-códigos q-arios, ver Teorema 5.2.4.

Ejemplo 5.5.27

1. Sea $p = 7$. Entonces se puede comprobar que $\mathcal{Q}_7 = \{1,2,4\}$ y que $\mathcal{N}_7 = \{3,5,6\}$. Sea α una raíz de $1 + x + x^3 \in \mathbb{F}_2[x]$. Así

$$\begin{aligned} g_{\mathcal{Q}}(x) &= \prod_{r \in \mathcal{Q}_7} (x - \alpha^r) = (x-\alpha)(x-\alpha^2)(x-\alpha^4) \\ &= 1 + x + x^3, \\ g_{\mathcal{N}}(x) &= \prod_{r \in \mathcal{N}_7} (x - \alpha^r) = (x-\alpha^3)(x-\alpha^5)(x-\alpha^6) \\ &= 1 + x^2 + x^3, \end{aligned}$$

   De esta manera, la matrices generadoras de los códigos $C_{\mathcal{Q}}$ y $C_{\mathcal{N}}$ están dadas por

$$G_{\mathcal{Q}} = \begin{pmatrix} g_{\mathcal{Q}}(x) \\ x g_{\mathcal{Q}}(x) \\ x^2 g_{\mathcal{Q}}(x) \\ x^3 g_{\mathcal{Q}}(x) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \text{ y}$$

$$G_{\mathcal{N}} = \begin{pmatrix} g_{\mathcal{N}}(x) \\ x g_{\mathcal{N}}(x) \\ x^2 g_{\mathcal{N}}(x) \\ x^3 g_{\mathcal{N}}(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix},$$

   respectivamente.

2. Los dos códigos de residuos cuadráticos ternarios de longitud 11 están dados por

$$C_{\mathcal{Q}} = \langle g_{\mathcal{Q}}(x) \rangle = \langle 2 + x^2 + 2x^3 + x^4 + x^5 \rangle,$$
$$C_{\mathcal{N}} = \langle g_{\mathcal{N}}(x) \rangle = \langle 2 + 2x + x^2 + 2x^3 + x^5 \rangle.$$

   Se puede probar que estos códigos son equivalentes al $[11,6,5]$-código ternario de Golay.

3. Considere el código de residuo cuadrático binario generado por el polinomio $g_{\mathcal{Q}}(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$. Este código es equivalente al $[23,12,7]$-código binario de Golay. □

Recordemos que la función ϕ dada por (5.1) tenemos que $\mathbb{F}_q^n$ y R_n son isomorfos como espacios vectoriales. En consecuencia, para demostrar que dos códigos cíclicos C y D son equivalentes, es suficiente encontrar una permutación φ de R_n, tal que $\varphi(C) = D$. Para finalizar esta sección se demuestra que los códigos q-arios de residuos cuadráticos $C_{\mathcal{Q}}$ y $C_{\mathcal{N}}$ son equivalentes. Para esto, primero demostraremos el siguiente lema.

Lema 5.5.28 Sean $m,n \in \mathbb{Z}^+$ tales que $\gcd(m,n) = 1$. Entonces la función

$$\begin{aligned} \chi_m : \quad R_n &\longrightarrow R_n \\ f(x) &\longmapsto f(x^m), \end{aligned}$$

es una permutación de R_n.

Demostración. Basta probar que χ_m es una permutación de R_n. En efecto, observe que si $f(x) = \sum_{i=0}^{n-1} f_i x^i$, entonces $\chi_m(f(x)) = f(x^m) \pmod{x^n - 1} = \sum_{i=0}^{n-1} f_i x^{(mi \bmod n)}$. El resultado se sigue dado que $mi \pmod n$ es una permutación del conjunto $\{1,2,\ldots,n\}$ cuando i recorre este conjunto, puesto que por hipótesis $\gcd(m,n) = 1$. ■

Proposición 5.5.29 Los códigos q-arios de residuos cuadráticos $C_{\mathcal{Q}}$ y $C_{\mathcal{N}}$ son equivalentes.

Demostración. Sea $m \in \mathcal{N}_p$. Consideremos $t \in \mathcal{N}_p$, entonces $tm \in \mathcal{Q}_p$. De ahí que

$$0 = g_{\mathcal{Q}}(\alpha^{tm}) = g_{\mathcal{Q}}((\alpha^t)^m) = \chi_m(g_{\mathcal{Q}}(\alpha^t)).$$

Esto implica que $g_{\mathcal{N}}(x)$ es un divisor de $\chi_m(g_{\mathcal{Q}}(x))$ dado que $g_{\mathcal{N}}(x)$ por definición no tiene raíces múltiples. En consecuencia, $\chi_m(C_{\mathcal{Q}}) \subseteq C_{\mathcal{N}}$. Ahora como $|\chi_m(C_{\mathcal{Q}})| = |C_{\mathcal{N}}|$, entonces $\chi_m(C_{\mathcal{Q}}) = C_{\mathcal{N}}$. ■

5.6. SageMath

Anillos de polinomios en SageMath

Como vimos para trabajar con códigos cíclicos debemos manipular polinomios. A continuación presentamos la forma en que se construyen estos objetos en SageMath. El comando que se utiliza para esto es `PolynomialRing(R)` donde R es el anillo del cual tomamos los coeficientes. De esta manera, si escribimos `R=PolynomialRing(QQ)` construimos el anillo de polinomios $\mathbb{Q}[t]$ con indeterminada t.

```
[In]: Q=PolynomialRing(QQ,'t')
```

Sin embargo, al escribir esto no podemos usar la indeterminada `t`. Para poder hacerlo debemos escribir

```
[In]: Q.<t>=PolynomialRing(QQ)
```

Así por ejemplo podemos calcular

```
[In]: t^3+t+5*t+1/2*t-1
```

```
[Out]: t^3 + 13/2*t - 1
```

```
[In]: (2*t+1)^4
```

```
[Out]: 16*t^4 + 32*t^3 + 24*t^2 + 8*t + 1
```

Observemos que podemos lograr esto con el comando `QQ['t']` o el comando `R.<t> = QQ[]`. Además, podemos construir el código con cualquier indeterminada.

Por ejemplo a continuación presentamos el mismo anillo de polinomios con indeterminada w.

```
[In]: R.<w> = PolynomialRing(QQ)
      (1+w)^3
```

```
[Out]: w^3 + 3*w^2 + 3*w + 1
```

```
[In]: (1+w)^3==1+w^3
```

```
[Out]: False
```

En consecuencia, para construir el anillo de polinomios $\mathbb{F}_3[x]$ hacemos

```
[In]: S.<x>=GF(3)[x]; S
```

```
[Out]: Univariate Polynomial Ring in x over Finite
       Field of size 3
```

```
[In]: (1+x)^3==1+x^3
```

```
[Out]: True
```

```
[In]: 3*x
```

```
[Out]: 0
```

```
[In]: K.<y>=PolynomialRing(GF(3)); K
```

```
[Out]: Univariate Polynomial Ring in y over Finite
       Field of size 3
```

```
[In]: (1+y)^3==1+y^3
```

```
[Out]: True
```

Creación de códigos cíclicos

Para construir un código de longitud n con polinomio generador $g(x)$ podemos usar el comando `codes.CyclicCode(length=3,generator_pol=g)`. A continuación montamos el código cíclico binario $C_1 =$

$\{000,001,010,100,011,110,101,111\}$ generado por el polinomio $g(x) = 1$ hacemos lo siguiente

```
[In]: F.<x>=GF(2)[] #anillo de polinomios con
        coeficientes sobre F2
      g=F(1) #Polinomio generador
      C1=codes.CyclicCode(length=3,generator_pol=g)
      C1
```

```
[Out]: [3, 3] Cyclic Code over GF(2)
```

De esta manera, una forma de obtener los elementos de C_1 se logra haciendo

```
[In]: for x in C1:
          print(x)
```

```
[Out]: (0, 0, 0)
       (1, 0, 0)
       (0, 1, 0)
       (1, 1, 0)
       (0, 0, 1)
       (1, 0, 1)
       (0, 1, 1)
       (1, 1, 1)
```

Por otro lado, para construir el código cíclico ternario

$$C_2 = \{00000000,10101010,01010101,20202020,02020202,\\ 21212121,12121212,11111111,22222222\}$$

escribimos

```
[In]: F.<x>=GF(3)["x"] # anillo de polinomios con
      coeficientes sobre F2
      g=F(x^6 + x^4 + x^2 + 1) # polinomio generador
      C2=codes.CyclicCode(length=8,generator_pol=g)
      C2
```

```
[Out]: [8, 2] Cyclic Code over GF(3)
```

Para obtener un listado de los elementos de C_2 hacemos

```
[In]: for x in C2:
          print(x)
```

```
[Out]: (0, 0, 0, 0, 0, 0, 0, 0)
       (1, 0, 1, 0, 1, 0, 1, 0)
       (2, 0, 2, 0, 2, 0, 2, 0)
       (0, 1, 0, 1, 0, 1, 0, 1)
       (1, 1, 1, 1, 1, 1, 1, 1)
       (2, 1, 2, 1, 2, 1, 2, 1)
       (0, 2, 0, 2, 0, 2, 0, 2)
       (1, 2, 1, 2, 1, 2, 1, 2)
       (2, 2, 2, 2, 2, 2, 2, 2)
```

Recordemos que por el Teorema 5.2.6 sabemos que los códigos cíclicos de longitud 8 se pueden obtener a partir de los divisores de $x^8 - 1$ en $\mathbb{F}_3[x]$. Esto lo podemos conseguir mediante el comando `divisors(f)`, ver Ejemplo 5.2.10.

```
[In]: f=F(x^8-1)
      divisors(f)
```

```
[Out]: [1,
        x + 1,
        x + 2,
        x^2 + 1,
        x^2 + 2,
        x^2 + x + 2,
        x^2 + 2*x + 2,
        x^3 + x + 1,
        x^3 + x + 2,
        x^3 + x^2 + 1,
        x^3 + x^2 + x + 1,
        x^3 + 2*x^2 + 2,
        x^3 + 2*x^2 + x + 2,
        x^4 + 1,
        x^4 + 2,
        x^4 + x^3 + x + 2,
        x^4 + x^3 + x^2 + 2*x + 1,
        x^4 + 2*x^3 + 2*x + 2,
        x^4 + 2*x^3 + x^2 + x + 1,
        x^5 + 2*x^3 + x^2 + x + 1,
        x^5 + 2*x^3 + 2*x^2 + x + 2,
        x^5 + x^4 + x + 1,
        x^5 + x^4 + x^3 + 2*x^2 + 1,
        x^5 + 2*x^4 + x + 2,
        x^5 + 2*x^4 + x^3 + x^2 + 2,
        x^6 + x^4 + x^2 + 1,
```

```
x^6 + 2*x^4 + x^2 + 2,
x^6 + x^5 + 2*x^4 + 2*x^2 + 2*x + 1,
x^6 + 2*x^5 + 2*x^4 + 2*x^2 + x + 1,
x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1,
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + 2*x^2 + x
+ 2, x^8 + 2]
```

De esta manera, podemos obtener todos los códigos cíclicos ternarios de longitud 8, con las siguientes instrucciones.

```
[In]: codigos_ciclicos_8=[]
      for g in divisors(f):
          codigos_ciclicos_8.append(codes.
           CyclicCode(length=8,generator_pol=g))
```

De esta manera, al escribir `len(codigos_ciclicos_8)` obtenemos el número de códigos cíclicos ternarios de longitud 8.

```
[In]: len(codigos_ciclicos_8)
```

```
[Out]: 32
```

Estas ideas las podemos implementar en una función que calcule en general todos los códigos cíclicos en $\mathbb{F}_q$ de longitud n, siempre y cuando n y q sean primos relativos, ver Observación 5.2.9.

```
[In]: def codigos_ciclicos(n,q):
          F.<x>=GF(q)["x"]
          f=F(x^n-1) # def. del polinomio x^n-1
          L=divisors(f)
          codigos_ciclic=[]
          for g in divisors(f):
              codigos_ciclic.append(codes.
               CyclicCode(length=n,generator_pol=g))
          return(codigos_ciclic)
```

```
[In]: codigos_ciclicos(8,3)
```

```
[Out]: [[8, 8] Cyclic Code over GF(3),
        [8, 7] Cyclic Code over GF(3),
        [8, 7] Cyclic Code over GF(3),
        [8, 6] Cyclic Code over GF(3),
        [8, 6] Cyclic Code over GF(3),
        [8, 6] Cyclic Code over GF(3),
```

```
[8, 6] Cyclic Code over GF(3),
[8, 5] Cyclic Code over GF(3),
[8, 5] Cyclic Code over GF(3),
[8, 5] Cyclic Code over GF(3),
[8, 5] Cyclic Code over GF(3),
[8, 5] Cyclic Code over GF(3),
[8, 5] Cyclic Code over GF(3),
[8, 4] Cyclic Code over GF(3),
[8, 4] Cyclic Code over GF(3),
[8, 4] Cyclic Code over GF(3),
[8, 4] Cyclic Code over GF(3),
[8, 4] Cyclic Code over GF(3),
[8, 4] Cyclic Code over GF(3),
[8, 3] Cyclic Code over GF(3),
[8, 3] Cyclic Code over GF(3),
[8, 3] Cyclic Code over GF(3),
[8, 3] Cyclic Code over GF(3),
[8, 3] Cyclic Code over GF(3),
[8, 3] Cyclic Code over GF(3),
[8, 2] Cyclic Code over GF(3),
[8, 2] Cyclic Code over GF(3),
[8, 2] Cyclic Code over GF(3),
[8, 2] Cyclic Code over GF(3),
[8, 1] Cyclic Code over GF(3),
[8, 1] Cyclic Code over GF(3),
[8, 0] Cyclic Code over GF(3)]
```

```
[In]: codigos_ciclicos(7,3) # Listado de polinomios
      generadores de códigos cíclicos de longitud 7
```

```
[Out]: [[7, 7] Cyclic Code over GF(3),
        [7, 6] Cyclic Code over GF(3),
        [7, 1] Cyclic Code over GF(3),
        [7, 0] Cyclic Code over GF(3)]
```

Si no conocemos el polinomio generador de un código cíclico, este lo podemos obtener con el comando `generator_polynomial()`.

```
[In]: C=codigos_ciclicos(8,3)[6]
      C.generator_polynomial()
```

```
[Out]: x^2 + 2*x + 2
```

Matrices generadoras y polinomios de control

Con los comandos `generator_matrix()` y `check_polynomial()` obtenemos la matriz generadora y el polinomio de control de un código cíclico dado.

```
[In]: C1.generator_matrix()

[Out]: [1 0 0]
       [0 1 0]
       [0 0 1]

[In]: C1.generator_polynomial()

[Out]: x + 1

[In]: C1.check_polynomial()

[Out]: x^3 + 1

[In]: C.generator_matrix()

[Out]: [2 2 1 0 0 0 0 0]
       [0 2 2 1 0 0 0 0]
       [0 0 2 2 1 0 0 0]
       [0 0 0 2 2 1 0 0]
       [0 0 0 0 2 2 1 0]
       [0 0 0 0 0 2 2 1]

[In]: C.check_polynomial()

[Out]: x^6 + x^5 + 2*x^4 + 2*x^2 + 2*x + 1
```

Adicionalmente, la matriz de control de paridad se puede obtener con el comando `parity_check_matrix()`.

```
[In]: C.parity_check_matrix()

[Out]: [1 1 2 0 2 2 1 0]
       [0 1 1 2 0 2 2 1]
```

Codificación

A continuación mostramos la forma de realizar la codificación con códigos cíclicos en SageMath. Retomamos el código cíclico C_2 del Ejemplo 5.3.1.

```
[In]: F.<x>=GF(3)["x"]
      g=F(x^6 + x^4 + x^2 + 1) # polinomio generador
      C2=codes.CyclicCode(length=8,generator_pol=g)
```

```
[In]: G=C2.generator_matrix() # Matriz generadora
        del código
```

Entonces para codificar el mensaje $\mathbf{m} = 10$ hacemos lo siguiente:

```
[In]: m=vector(GF(3),[1,0]); m*G
```

```
[Out]: (1, 0, 1, 0, 1, 0, 1, 0)
```

Lo que acabamos de hacer es lo mismo que trabajar con la codificación no sistemática usamos el comado. En general podemos usar el comando `C.encode(m,"Vector")`. Entonces la codificación del mensaje anterior se logra escribiendo simplemente

```
[In]: C2.encode(m,'Vector')
```

```
[Out]: (1, 0, 1, 0, 1, 0, 1, 0)
```

Para escoger la codificación sistemática usamos el comando `C.encode(m,"Systematic")` donde C es el código cíclico y $\mathbf{m}$ el mensaje que se desea transmitir.

```
[In]: F.<x>=GF(3)["x"]
      g=F(x^6 + x^4 + x^2 + 1) # polinomio generador
      C2=codes.CyclicCode(length=8,generator_pol=g)
```

```
[In]: m=vector(GF(3),[1,0])
      C2.encode(m,"Systematic")
```

```
[Out]: (1, 0, 1, 0, 1, 0, 1, 0)
```

```
[In]: m=vector(GF(3),[0,1]);
      C2.encode(m,"Systematic")
```

```
[Out]: (0, 1, 0, 1, 0, 1, 0, 1)
```

Observemos que en este caso obtenemos que la codificación es la misma. Esto es porque la matriz generadora generadora del código está en la forma estándar. Para ver las diferencias consideremos el siguiente código

```
[In]: F.<x>=GF(3)["x"]
      g=F(1+x+x^2+x^3) # polinomio generador
      C=codes.CyclicCode(length=8,generator_pol=g)
```

```
[In]: m=vector(GF(3),[1,1,1,0,1]) #Codificación
        sistemática
      C.encode(m,"Systematic")
```

```
[Out]: (1, 1, 1, 0, 1, 1, 1, 2)
```

```
[In]: C.encode(m,"Vector") #Codificación no
        sistemática
```

```
[Out]: (1, 2, 0, 0, 0, 2, 1, 1)
```

En este caso estas dos decodificaciones dan resultados distintos porque la matriz no está en la forma estándar, como podemos verificar a continuación.

```
[In]: C.generator_matrix()
```

```
[Out]: [1 1 1 1 0 0 0 0]
       [0 1 1 1 1 0 0 0]
       [0 0 1 1 1 1 0 0]
       [0 0 0 1 1 1 1 0]
       [0 0 0 0 1 1 1 1]
```

```
[In]: C.encode(vector(GF(3),[0,1,0,0,1]),'Vector')
```

```
[Out]: (0, 1, 1, 1, 2, 1, 1, 1)
```

La codificación sistemática para las palabras 11101 y 01001 son respectivamente:

```
[In]: C.encode(vector(GF(3),[1,1,1,0,1]),
      "Systematic")
```

```
[Out]: (1, 1, 1, 0, 1, 1, 1, 2)
```

```
[In]: C.encode(vector(GF(3),[0,1,0,0,1]),
      "Systematic")
```

```
[Out]: (0, 1, 0, 0, 1, 0, 1, 1)
```

Decodificación

Para realizar la decodificación podemos usar el método de calcular el síndrome de una palabra recibida. Con ayuda de la función `.quo_rem()` podemos calcular el síndrome de un polinomio. Dados dos polinomios $f(x), g(x) \in \mathbb{F}_q[x]$ con $g(x)$ no nulo el comando `f.quo_rem(g)` retorna el resultado de la aplicación del algoritmo de la división. Por ejemplo si deseamos aplicar el algoritmo de la división para los polinomios $f(x) = 3x^4 + x^3 + 2x^2 + 1$ y $g(x) = x^2 + 4x + 2$ sobre $\mathbb{F}_5$ hacemos:

```
[In]: R.<x>=PolynomialRing(GF(5))
      f=3*x^4+x^3+2*x^2+1; g=x^2+4*x+2
```

Enotonces al escribrir

```
[In]: f.quo_rem(g)
```

```
[Out]: (3*x^2 + 4*x, 2*x + 1)
```

Obtenemos una tupla con dos coordenadas. La primera corresponde con el cociente y la segunda con el residuo. Para verificar esto hacemos

```
[In]: q,r=f.quo_rem(g)
      f==q*g+r
```

```
[Out]: True
```

Naturalmente podemos usar el comando `.quo_rem()` para calcular el síndrome de un polinomio. En efecto, si queremos calcular el síndrome del polinomio $u(x) = x^2 + x^3 + x^4 + x^5$ en el código cíclico binario de longitud 7 generado por $g(x) = 1 + x^2 + x^3$, hacemos lo siguiente:

```
[In]: F.<x>=PolynomialRing(GF(2))
      u=x^2+x^3+x^4+x^5;g=1+x^2+x^3
```

Entonces el síndrome de $u(x)$ es igual a

```
[In]: u.quo_rem(g)[1]
```

```
[Out]: x^2 + 1
```

5.7. Ejercicios

5.7.1. Ejercicios teóricos

1. Factorice el polinomio $x^7 - 1$ sobre $\mathbb{F}_2$.
2. ¿Cuáles de los siguientes códigos son cíclicos?
   a) El código binario $C_1 = \{0000, 1100, 0110, 0011, 1001\}$.
   b) El código binario $C_2 = \{00000, 10110, 01101, 11011\}$.
   c) El código ternario $C_3 = \{0000, 1122, 2211\}$.
   d) El código binario E_n de las palabras de peso par de longitud n.
3. Demuestre que el conjunto $I = \{f(x) \in \mathbb{F}_q[x] : f(0) = f(1) = 0\}$ es un ideal de $\mathbb{F}_q[x]$ y encuentre un generador.
4. Encuentre todos los posibles generadores mónicos para cada uno de los siguientes ideales
   a) $I = \langle 1 + x + x^3 \rangle \subset \dfrac{\mathbb{F}_2[x]}{\langle x^7 - 1 \rangle}$,
   b) $I = \langle 1 + x^2 \rangle \subset \dfrac{\mathbb{F}_2[x]}{\langle x^4 - 1 \rangle}$.
5. Para cada uno de los siguientes códigos cíclicos, encuentre el correspondiente polinomio generador
   a) $\{\lambda(1, 1, \ldots, 1) : \lambda \in \mathbb{F}_q \subset \mathbb{F}_q^n$,
   b) $\{0000, 1010, 0101, 1111\} \subset \mathbb{F}_2^4$,
   c) $\{x_0 x_1 \ldots x_{n-1} \in \mathbb{F}_q^n : \sum_{i=0}^{n-1} x_i = 0\}$,
   d) $\{x_0 x_1 \ldots x_{n-1} \in \mathbb{F}_q^n : \sum_{i=0}^{n-1} x_i^3 = 0\}$.

6. Determine la menor longitud para un código binario cíclico tal que los cada uno de los siguientes polinomios es el polinomio generador:

   a) $g(x) = 1 + x^4 + x^5$,

   b) $g(x) = 1 + x + x^2 + x^4 + x^6$.

7. Sea $g(x) = (1+x)(1+x+x^3) \in \mathbb{F}_2[x]$ el polinomio generador de un $[7,3]$-código cíclico C. Encuentre una matriz generadora y una matriz de control de paridad para C. Encuentre una matriz en la forma estándar para el código C.

8. Sea $g(x) = x^8 + x^7 + x^6 + x^4 + 1 \in \mathbb{F}_2[x]$ el polinomio generador de un $[15,7]$-código binario C. Encuentre una matriz generadora y una matriz de control de paridad para C. Encuentre una matriz en la forma estándar para el código C.

9. Sea C un código lineal tal que toda fila de una matriz generadora tiene la propiedad de que el shift cíclico de toda fila es nuevamente una palabra del código. Demuestre que C es un código cíclico.

10. Sea C un código cíclico. Demuestre que si se recibe $u(x)$ codificado con C, entonces

    a) $u(x)$ es una palabra código si y sólo si $\text{syn}(u(x)) = 0$.

    b) Dos polinomios tienen el mismo síndrome si y sólo si están en la misma clase lateral de C

11. Sean $C_1 = \langle g_1(x) \rangle$ y $C_2 = \langle g_2(x) \rangle$ dos códigos cíclicos en $\mathbb{F}_q^n$.

    a) Demuestre que $C_1 \subseteq C_2$ si y sólo si $g_1(x)$ divide a $g_2(x)$.

    b) Demuestre que $C_1 \cap C_2$ y $C_1 + C_2$ son cíclicos.

    c) Determine los polinomios generadores de $C_1 \cap C_2$ y $C_1 + C_2$ en términos de $g_1(x)$ y $g_2(x)$.

12. Sea C un código binario cíclico con polinomio generador $g(x)$.

    a) Pruebe que si $g(x)$ es divisible por $x - 1$, entonces todas las palabras código de C tienen peso par.

    b) Suponga que la longitud C es impar. Muestre que $\mathbf{1} = 11\ldots1 \in C$ si y sólo si $g(x)$ no es divisible por $x - 1$.

c) Suponga que la longitud de C es impar. Muestre que C contiene una palabra código de peso impar si y sólo si $\mathbf{1} = 11\ldots1 \in C$.

13. Sea $g(x)$ el polinomio generador de un $[n,k]$-código q-ario tal que $\gcd(n,q) = 1$. Demuestre que $\mathbf{1} = 11\ldots1 \in C$ si y sólo si $g(x)$ no es divisible por $x-1$.

14. Sea $C = \langle 1 + x^4 + x^6 + x^7 + x^8 \rangle \subseteq \mathbb{F}_2^{15}$. Decodifique las siguientes palabras recibidas:

    a) 110111101110110,

    b) 111110100001000.

15. Sea $C = \langle 1 + x^2 + x^4 + x^5 \rangle \subseteq \mathbb{F}_2^{15}$.

    a) Encuentre la distancia mínima de C.

    b) Decodifique la palabra 010110000000010,

    c) Decodifique la palabra 110000111010011

    d) 111110100001000.

16. Una palabra código $e(x)$ de un código q-ario cíclico C de longitud n se denomina *idempotente* si $e^2(x) \equiv e(x) \pmod{x^n - 1}$. Si un elemento $e(x)$ es también generador de C, se denomina también *generador idempotente*. Sea $C = \langle g(x) \rangle$ y considere $h(x) = \dfrac{x^n - 1}{g(x)}$. Demuestre que, si $\gcd(g(x),h(x)) = 1$, entonces C tiene un único generador idempotente. En particular, demuestre que si $\gcd(n,q) = 1$, entonces siempre existe un único generador idempotente para un código q-ario cíclico de longitud n.

17. Demuestre el Teorema 5.5.18.

5.7.2. Prácticas en SageMath

1. Considere la siguiente función implementada en SageMath

```
def desplazamiento_ciclico(v):
        l=len(v)
        return v[l-1:]+v[:l-1]
```

Esta función recibe una lista (no un vector) y realiza un desplazamiento cíclico de la lista dada. Por ejemplo, obtenemos

```
desplazamiento_ciclico([1,2,3,4,5,6,7])
```

```
[7, 1, 2, 3, 4, 5, 6]
```

   a) Verifique con ayuda de la función `desplazamiento_ciclico` que los códigos *a*) al *c*) del Ejercicio 2 son cerrados bajo desplazamientos cíclicos.

   b) Compruebe con ayuda de la función `desplazamiento_ciclico` las cuentas hechas en el Ejemplo 5.1.3.

2. Para construir la función ϕ dada por (5.1) utilice los comandos `R.<x>= PolynomialRing(GF(q))` y `R(list(reversed([c0,c1,...,cn])))`. Utilice estos comandos para implementar una función que le permita verificar las cuentas del Ejemplo 5.1.4.

3. Sea $I = \langle\langle x^5 + 4x^4 + 4x^3 + x^2 + 3x + 4\rangle\rangle \subset \dfrac{\mathbb{F}_5[x]}{\langle x^{17} - 1\rangle}$. Con ayuda de SageMath encuentre para el código cíclico I:

   a) Una matriz generadora.

   b) Una matriz de control de paridad.

   c) Una matriz generadora en la forma estándar.

4. Usando SageMath factorice el polinomio $x^8 - 1$ sobre $\mathbb{F}_2$ y encuentre todos los generadores de todos los polinomios cíclicos de longitud 8 sobre $\mathbb{F}_5$.

5. Construya una función que realice la técnica de decodificación descrita en los ejemplos 5.4.7 y 5.4.8.

5.8. Biografía

Florence Jessie Collinson MacWilliams **(1917-1990)**

Fue una de las primeras mujeres en publicar en teoría de códigos. Matemática británica que realizó sus estudios de pregrado y maestría en matemáticas en la Universidad de Cambridge. En 1939, trabajó con Oscar Zariski, reconocido por sus aportes en geometría algebraica, con la intención de estudiar junto a él en la Universidad de Harvard. Sin embargo, hizo una para debido a su reciente matrimonio y a su dedicación a la crianza de su familia. En 1958, MacWilliams se unió a los Laboratorios Bell como programadora de computadores. Una charla de R. C. Bose, uno de los inventores de los códigos BCH, la inspiró para trabajar en teoría de códigos. Para aspirar a una posición de investigadora en su trabajo necesitaba tener un doctorado, por lo que en 1961, a la edad de 44 años, regresó a Harvard para obtener un doctorado en tan sólo un año.
Entre las principales contribuciones de su tesis se destaca la que hoy se conoce como la Identidad de MacWilliams, ver Teorema 3.4.8. Más precisamente, MacWilliams obtuvo una formula que relaciona la distribución de pesos de un código lineal con la de su dual [24]. Dado que se pueden obtener cotas para el número de errores a partir de la distribución de pesos de un código, este resultado es de interés para matemáticos e ingenieros. Este trabajo fue considerado por Vera Pless como uno de los más potentes en la teoría de códigos, ver `https://awm-math.org/awards/noether-lectures/noether-lectures-1980/`.
Entre 1962 y 1976, MacWilliams produjo importantes resultados sobre algebraica construcciones y propiedades combinatorias de códigos. Trabajó en códigos cíclicos, generalizándolos a códigos de grupo Abeliano. En 1977, junto con N. J. A. Sloane escribió el libro "The Theory of Error-Correcting Codes" [25]; en él se incluyen resultados tanto algebraicos como combinatorios en la teoría de códigos.

En 1980, MacWilliams fue la primera expositora en las Conferencias de Emmy Noether, realizada por la Asociación para Mujeres en Matemáticas, ver https://awm-math.org/awards/noether-lectures/.

Foto tomada de Biographies of Women Mathematicians. https://mathwomen.agnesscott.org/women/macwill.htm.

Referencias

[1] "In Memoriam Marcel J. E. Golay," *Journal of Liquid Chromatography*, vol. 12, no. 9, pp. 1541–1543, jul 1989.

[2] "The Life and Work of Vera Stepen Pless," *Notices of the AMS*, 2022. [Online]. Available: https://hal.science/hal-03787109

[3] M. Baldi, M. Bertinelli, F. Chiaraluce, P. Closas, P. Dhakal, R. Garello, N. Maturo, M. Navarro, J. M. Palomo, E. Paolini, S. Pfletschinger, P. F. Silva, L. Simone, and J. Vilà-Valls, "State-of-the-art space mission telecommand receivers," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 6, pp. 4–15, jun 2017.

[4] J. Beissinger and V. Pless, *The Cryptoclub: Using Mathematics to Make and Break Secret Codes*. Wellesley, MA: A K Peters, 2018.

[5] A. E. Brouwer, "The linear programming bound for binary linear codes," *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 677–680, 1993.

[6] M. L. Costa Vianna, J. P. Kappes Marques, and M. L. de Campos, "O uso de códigos corretores de erros em comunicações acústicas submarinas," in *XXXIV Simpósio Brasileiro de Telecomunicações*, Santarém, PA, pp. 244–245, 2016.

[7] K. Cramers, "Obituary Professor Dr. Marcel J. Golay (1902-1989)," *Journal of High Resolution Chromatography*, vol. 12, no. 5, pp. 269–269, may 1989.

[8] Z. Dawy, P. Hanus, J. Weindl, J. Dingel, and F. Morcos, "On genomic coding theory," *European Transactions on Telecommunications*, vol. 18, no. 8, pp. 873–879, 2007.

[9] P. Delsarte and J. Goethals, "Unrestricted codes with the golay parameters are unique," *Discrete Mathematics*, vol. 12, no. 3, pp. 211–224, 1975.

[10] D. Engelbert, R. Overbeck, and A. Schmidt, "A summary of McEliece-type cryptosystems and their security," *Journal of Mathematical Cryptology*, vol. 1, no. 2, pp. 151–199, 2007.

[11] J. Gallian, *Contemporary Abstract Algebra*. Boca Raton, USA: CRC Press, 2021.

[12] E. N. Gilbert, "A comparison of signalling alphabets," *Bell System Technical Journal*, vol. 31, no. 3, pp. 504–522, May 1952.

[13] M. Golay, "Notes on digital coding," *Proc. IRE*, vol. 37, no. 6, p. 657, Jun. 1949.

[14] M. Grassl, "Bounds on the minimum distance of linear codes," *http://www. codetables. de*, 2008.

[15] I. Gutiérrez García and D. Villar Salinas, "On authomorphisms of extremal type II codes," *Revista Integración*, vol. 31, pp. 107 – 120, 12 2013. [Online]. Available: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-419X2013000200001&nrm=iso

[16] R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[17] W. C. Huffman, "A tribute to vera pless (1931–2020)," *Finite Fields and Their Applications*, vol. 65, pp. 1–4, 2020.

[18] W. C. Huffman, J.-L. Kim, and P. Solé, Eds., *Concise Encyclopedia of Coding Theory*. Boca Raton, USA: CRC Press, 2021.

[19] W. C. Huffman and V. Pless, *Fundamentals of error-correcting codes*. Cambridge, UK: Cambridge University Press, 2010.

[20] S. J. K. Kenneth, "Construction of binary linear codes," Master's thesis, Dept. of Math., National University of Singapore, Singapore, 1999.

[21] K. Lindström, "All nearly perfect codes are known," *Information and Control*, vol. 35, no. 1, pp. 40–47, 1977.

[22] S. Ling and C. Xing, *Coding theory: a first course*. New York, USA: Cambridge University Press, 2004.

[23] J. J. López Santander and H. M. Ruiz, "La matemática de la teoría de códigos," Tesis Pregrado, Universidad de Nariño, 2013.

[24] F. MacWilliams, "Combinatorial problems of elementary group theory," Ph.D. dissertation, Dept. of Math., Harvard University, Cambridge, MA, 1962.

[25] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. New York, USA: North-Holland, 1977.

[26] J. Malkevitch, "Digital revolution (part III) - error correction codes," Feb. 2003, feature column. Monthly essays on mathematical topics. American Mathematical Society. [Online]. Available: http://www.ams.org/publicoutreach/feature-column/fcarc-errors1

[27] J. L. Massey, "Marcel J. E. Golay (1902-1989)," *IEEE Information Society Newsletter, June*, 1990.

[28] R. J. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[29] R. Pinch, "Coding theory: the first 50 years," Sep. 1997, plus Magazine. [Online]. Available: https://plus.maths.org/content/coding-theory-first-50-years

[30] R. Pinto, P. R. Malonek, and P. Vettori, Eds., *Coding Theory and Applications*. Switzerland: Springer International Publishing, 2015.

[31] V. Pless, "On the uniqueness of the golay codes," *Journal of Combinatorial Theory*, vol. 5, no. 3, pp. 215–228, Nov. 1968.

[32] ——, *Introduction to the theory of error-correcting codes*. New York, USA: John Wiley & Sons, 1998.

[33] V. Pless, R. A. Brualdi, and W. C. Huffman, Eds., *Handbook of coding theory*. Amsterdam: North-Holland, 1998.

[34] M. Plotkin, "Binary codes with specified minimum distance," *IRE Transactions on Information Theory*, vol. 6, no. 4, pp. 445–450, 1960.

[35] G. E. Ramos Zambrano, Z. E. Muñoz Burbano, and J. H. Castillo, *Teoría de fotones, estructura atómica de la materia y códigos lineales*. Pasto, COL: Editorial Universidad de Nariño, 2024.

[36] S. Roman, *Coding and information theory*. New York, USA: Springer Verlag, 1992.

[37] A. H. Salavati, "Applications of coding theory in biological systems," jun 2010, algorrithmics Laboratory (ALGO), I&C. [Online]. Available: https://wiki.epfl.ch/edicpublic/documents/Candidacy%20exam/salavati.pdf

[38] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[39] ——, "A symbolic analysis of relay and switching circuits," *Electrical Engineering*, vol. 57, no. 12, pp. 713–723, 1938.

[40] ——, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[41] ——, "Programming a computer for playing chess," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[42] H. Singh, "Code based cryptography: classic Mceliece," *arXiv preprint arXiv:1907.12754*, 2019.

[43] R. Singleton, "Maximum distance q-nary codes," *IEEE Transactions on Information Theory*, vol. 10, no. 2, pp. 116–118, 1964.

[44] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 9.3)*, 2023, https://www.sagemath.org.

[45] J. van Lint, "A survey of perfect codes," *Rocky Mountain Journal of Mathematics*, vol. 5, pp. 199–224, 1975.

[46] R. R. Varshamov, "Estimate of the number of signals in error correcting codes," *Docklady Akad. Nauk, SSSR*, vol. 117, pp. 739–741, 1957.

[47] H. P. Yockey, *Information theory, evolution, and the origin of life*. Cambridge: Cambridge University Press, 2005.

Índice alfabético

Sobre los autores

John Hermes Castillo Gómez: Matemático de la Universidad del Cauca (2003). Realizó estudios de Maestría en Matemáticas en la Universidad de Antioquia (2006) y de Doctorado en Matemáticas en el Instituto de Matemáticas y Estadística de la Universidad de Sao Paulo, Brasil (2012). Actualmente es Profesor Asociado del Departamento de Matemáticas y Estadística de la Universidad de Nariño. Miembro del grupo de investigación Álgebra, Teoría de Números y Aplicaciones: ERM. Sus áreas de investigación incluyen teoría de números, anillos de grupos y teoría de códigos.

John Jairo López Santander: Licenciado en Matemáticas de la Universidad de Nariño (2013) y Magíster en Ciencias Matemáticas de la Universidad del Cauca (2017). En la actualidad se encuentra adelantando estudios de Doctorado en Matemáticas en la Escuela de Ciencia e Ingeniería de la Universidad de Tulane, Estados Unidos. Sus áreas de investigación incluyen teoría de números, combinatoria y random matrix theory.

Hamilton Mauricio Ruiz: Licenciado en Matemáticas de la Universidad de Nariño (2013), Magíster en Ciencias Matemáticas de la Universidad del Cauca (2018) y Doctor en Ciencias Matemáticas de la Universidad del Cauca (2023). Miembro del grupo de investigación Álgebra, Teoría de Números y Aplicaciones: ERM. Sus áreas de investigación incluyen teoría de códigos, teoría de diseños y conjuntos de Sidon. Actualmente es profesor auxiliar en la Escuela Superior de Administración Pública (ESAP).

êditorial
Universidad de Nariño

Año de publicación: 2025
San Juan de Pasto-Nariño-Colombia

Este texto ofrece una introducción a los conceptos básicos de la Teoría de Códigos Correctores de Errores, también conocida como Teoría de Códigos. El principal objetivo de este manuscrito es incentivar el estudio de esta área entre los lectores de habla hispana, en especial en nuestro país. Aquí brindamos un acercamiento a las demostraciones de algunos de sus resultados fundamentales, con ejemplos y ejercicios para que el lector practique los conceptos estudiados. Este libro está dirigida a estudiantes de pregrado o posgrado en matemáticas e ingeniería, y a investigadores que quieran conocer los fundamentos sobre el tema. En cada capítulo presentamos ejemplos de rutinas computacionales en SageMath con el objetivo de ejemplificar, cuando esto sea posible, los conceptos estudiados. Estas implementaciones están basadas en los conceptos teóricos que aquí presentamos y sustentadas en las capacidades que brinda el software libre SageMath.

Este libro es resultado del proyecto de investigación Una mirada computacional a la Teoría Algebraica de Códigos financiado por la Vicerrectoría de Investigación e Interacción Social de la Universidad de Nariño.

9 786287 771208

Universidad de Nariño
FUNDADA EN 1904

ai Universidad de Nariño

120 Universidad de Nariño

êditorial Universidad de Nariño