INTRODUCCIÓN A LA TEORÍA DE CÓDIGOS CÍCLICOS

JIMMY OLMEDO DIAZ PORTILLO

FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE MATEMÁTICAS Y ESTADÍSTICA
UNIVERSIDAD DE NARIÑO
SAN JUAN DE PASTO
2016

INTRODUCCIÓN A LA TEORÍA DE CÓDIGOS CÍCLICOS

JIMMY OLMEDO DIAZ PORTILLO

Trabajo presentado como requisito parcial para optar al título de Licenciado en Matemáticas

Asesor
John Hermes Castillo Gómez
Doctor en Matemáticas

FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE MATEMÁTICAS Y ESTADÍSTICA
UNIVERSIDAD DE NARIÑO
SAN JUAN DE PASTO
2016

Nota de Responsabilidad

Todas las ideas y conclusiones aportadas en el siguiente trabajo son responsabilidad exclusiva de los autores.

Artículo 1^{ro} del Acuerdo No. 324 de octubre 11 de 1966 emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

Nota de Aceptación

John Hermes Castillo Gómez

Presidente de Tesis

Alexander Holguín Villa

Jurado

Oscar Fernando Soto Agreda

Jurado

San Juan de Pasto, 24 de noviembre de 2016

Este trabajo está dedicado a:

Mis padres, Olmedo Pepe Díaz Arciniegas y Luz del Carmen Portillo, y a mis hermanos, como muestra de gratitud por su esfuerzo incansable y el apoyo brindado a lo largo de mi etapa de estudios, ya que eso me ha permitido hoy alcanzar esta gran meta.

Agradecimientos

Al término de este trabajo de grado es ineludible expresar los sentimientos de agradecimiento, en primer lugar a Dios por iluminar y bendecir el camino hasta aquí recorrido. También, mis más sinceras gracias por todo el apoyo que me brindaron en cada etapa de este difícil camino a aquellas personas que de una u otra manera contribuyeron para que haya podido alcanzar esta meta, en especial a mis padres, hermanos y mi novia que se constituyeron en el motor que inspiraba cada nuevo logro conseguido.

A mis profesores, no solo de la universidad, a quienes debo el haberme formado como un excelente profesional, sino también a mis profesores del colegio, quienes establecieron los cimientos de ese edificio del conocimiento que aún estoy tratando de construir en mí, mi más profundo respeto y una completa gratitud para cada uno de ustedes.

Resumen

En consonancia con los nuevos campos de investigación ligados al desarrollo tecnológico actual, en este trabajo se ha intentado enfocar el estudio en la manera como se llevan a cabo procesos tan comunes como el envío y recepción de información, centrando la atención específicamente al hacerlo por medio de códigos cíclicos; para ello se hace un recorrido por aquellos conceptos claves para la comprensión de esta área de estudio. Se enfatiza sobre los procesos que permiten realizar la codificación y decodificación de información al usar códigos cíclicos. Por otra parte, un aporte importante de este trabajo es que incluye algunas instrucciones para el uso del software de matemáticas conocido como SAGE, que facilitan la comprensión y experimentación en torno a muchos de los conceptos aquí presentados, no sólo con códigos cíclicos, sino también con códigos cuasi-cíclicos.

Abstract

Coding theory is a research field related with the current techological development in our world. On this work we give an introduction, from the mathematical point of view, to the key concepts of this theory. Specially, we focus on the theory of cyclic codes. It is emphasized on the processes which allow the encoding and decoding information using cyclic codes. On the other hand, an important contribution of this work is that it provides some instructions for the use of the free open-source mathematics software system called SAGE, which facilitate the understanding and experimentation around many of the results presented here, not only with cyclic codes, but also with quasi-cyclic codes.

Índice general

Introducción

Actualmente la tecnología digital se encuentra presente en cada uno de los espacios donde cualquier persona se desenvuelve y en los que continuamente se enfrenta a la necesidad de mantenerse informado y de transmitir información. Los artefactos elaborados para dichos fines son muy variados, dado que las maneras de salvaguardar, procesar y compartir cualquier tipo de datos se han diversificado de muchas maneras, como audio, imagen o texto, entre otras; sin embargo, el llevar a cabo estos procesos de forma rápida, económica (en términos computacionales) y segura, ha requerido el desarrollo de una gran variedad de campos de estudio, entre los que se destacan la Teoría de la Información, la Teoría de Códigos y la Criptografía.

En relación a la Teoría de Códigos es de resaltar que el papel importante que juega radica en que se enfoca principalmente en la construcción de códigos que tengan la solvencia para manejar un alto volumen de información con la mayor verosimilitud posible a la vez, esta última condición se relaciona con la capacidad para detectar y corregir los errores que se puedan presentar en un mensaje que ha sido manipulado, particularmente al ser enviado a través de un canal de comunicación, ya que no está garantizado el hecho de que cualquier mensaje enviado sea exactamente el mensaje recibido.

Con esta monografía se ha querido dar una pequeña introducción a los objetos de estudio de la Teoría de Códigos, explicitando la manera en que las matemáticas entran a jugar un papel esencial en la sistematización de ciertos procesos al interior de ella; también se mencionan tres tipos de códigos importantes, los códigos lineales, los códigos cíclicos y los códigos cuasi-cíclicos, destacando principalmente al segundo de ellos y dando para los dos últimos algunas instrucciones útiles al momento de ser implementados en un entorno computacional, que en este caso ha sido el software SAGE.

En la sección relacionada con los temas preliminares de esta monografía, Capítulo 1, inicialmente se abordan varios conceptos y resultados de la teoría de campos finitos, orientados principalmente a la comprensión de las temáticas tratadas en los Capítulos 2 y 3, no sin antes mostrar el esquema del proceso de comunicación con los elementos que intervienen en él, describiendo brevemente la manera en que se traducen los mensajes al lenguaje de la máquina diseñada para tal fin. En la parte final de este capítulo se hace un completo recorrido por los códigos lineales, los cuales se destacan como la primera estructura que muestra un componente matemático más elaborado dentro de la Teoría de Códigos.

En el Capítulo 2 se da una mirada bien detallada de lo que son los códigos cíclicos, cuya es-

tructura matemática demuestra ser un poco más completa que la estructura de los códigos lineales, se presentan las definiciones y conceptos ligados a ellos, se dan demostraciones minuciosas de los resultados más sobresalientes para este tipo de códigos y se dan ejemplos ilustrativos que permitirán al lector tener una idea muy clara de la forma en que se codifica y se decodifica con estos códigos; adicionalmente, se indica la manera en que muchos de los procesos descritos en este capítulo se pueden facilitar haciendo uso del software SAGE, ver [1].

Ya en el Capítulo 3 se aborda un tipo de códigos estrechamente relacionado con los códigos cíclicos, los códigos cuasi-cíclicos, que son el resultado de una pequeña variación en la definición de los primeros. Siguiendo las ideas desarrolladas en [5], aquí se presentan algunos conceptos análogos a los dados para los códigos cíclicos, ayudados por una terminología auxiliar que permite establecer la conexión entre ellos. Además, en esta parte del texto se destaca una manera en que se puede trabajar con estos códigos en entornos computacionales, en este caso SAGE, orientada a la construcción de una matriz generadora que se componga solamente de filas linealmente independientes y algunas de sus rotaciones.

Capítulo 1

Preliminares

De manera general, cuando se desea almacenar, procesar, enviar, transportar y/o recibir información, inicialmente se requiere de un medio denominado *canal* capaz de manejar un determinado lenguaje. En un sentido amplio, el canal puede ser espacial o temporal: envío por línea telefónica u óptica, almacenamiento en un disco o dispositivo electrónico, etc., sin embargo, sin importar cual sea el medio utilizado para llevar a cabo estas acciones, pueden ocurrir fenómenos (*ruido en el canal*) que hacen que en dicho mensaje se produzcan alteraciones (*errores*), ocasionando que la información transmitida no se conserve igual a como se envió.

Teniendo en cuenta los aspectos mencionados en el párrafo anterior, se puede decir que para iniciar cualquiera de dichos procesos es necesario expresar el mensaje (*palabras fuente*) en unos términos acordes a los que el canal está capacitado para manejar (*palabras código*), lo cual se conoce como *codificación de la fuente*, para luego continuar con la *codificación del canal* cuyo fin es conformar un mensaje más resistente al ruido; en seguida, este es llevado a través del canal hasta el punto de recepción, en los mismos términos que este maneja; allí es necesario otro proceso que se conoce como *decodificación del canal*, que tiene como finalidad detectar y en lo posible corregir cualquier error al interior del mensaje producto del ruido en el canal, como ya se mencionó antes; finalmente, para lograr que el mensaje sea entendido por el receptor, se requiere ponerlo en los términos que él es capaz de interpretar, para ello se realiza la *decodificación de la fuente*. Una manera de ilustrar el proceso aquí descrito, se presenta en la Figura 1.1.

La *codificación del canal* permite realizar una segunda codificación a los mensajes codificados por la fuente, añadiendo siempre unos *símbolos de "redundancia"* que permitan detectar algún número determinado de errores dentro de cada palabra del código. La *decodificación del canal*, facilita el detectar y corregir los errores al momento de quitar los símbolos de redundancia agregados al mensaje.

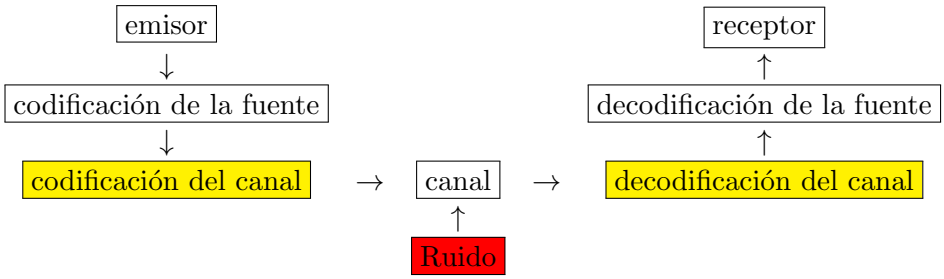

Figura 1.1: Esquema de Comunicación.

Vale la pena destacar que la Teoría de Códigos, en esencia se ocupa de la codificación y decodificación de mensajes, junto con el problema de detectar y corregir errores. Del problema más general que consiste en considerar todo el proceso, el diseño de canales a usar, el procesamiento de la información, la medición de la información, la capacidad de los sistemas para procesar y transmitir información, etc., se ocupa la Teoría de la Información, inaugurada por el trabajo pionero de Claude Shannon en 1948, ver [28]. Por otra parte, no se debe confundir a la Teoría de Códigos con la Criptografía, que también es parte de la Teoría de la Información; mientras que en la segunda, lo que importa es enviar un mensaje a un receptor amigo, que sea secreto e inviolable para los demás (el enemigo); en la primera, lo que interesa es enviar, almacenar y/o recibir un mensaje con la mayor eficiencia y verosimilitud posibles [25].

En ese orden de ideas, con el fin de proporcionar al lector unas bases suficientes para la comprensión de la temática abordada en este texto, a continuación se presentan una serie de definiciones y resultados relacionados.

1.1. Álgebra abstracta

Se empezará dando algunos conceptos y propiedades básicas de la teoría del álgebra abstracta y en seguida de la teoría de los campos finitos, puesto que muchos de estos se emplean en el estudio de los códigos en general, y particularmente en el estudio de los códigos cíclicos y los códigos cuasicíclicos. Las demostraciones en esta sección se darán únicamente cuando la relación existente con la temática de este texto así lo requieran, para otras demostraciones de interés el lector puede remitirse a las referencias dadas delante de cada teorema, lema o corolario mencionado.

Definición 1.1. Un *anillo* R, en términos generales, es un conjunto donde la adición, la sustracción y la multiplicación son posibles. Formalmente, R es un grupo aditivo, junto con una multiplicación que satisface que para todos los $a, b \in R$

- $a(b + c) = ab + ac$.

- $(ab)c = a(bc)$.

Cuando el anillo contiene un elemento 1 tal que $1 \cdot a = a \cdot 1 = a$ para todo $a \in R$, donde " $\cdot$ "hace referencia a la multiplicación definida en el anillo, entonces se dice que R es un anillo conmutativo con unidad o identidad.

Definición 1.2. Sea R un anillo. Un *R-módulo (izquierdo)* o *módulo sobre R*, consta de un grupo abeliano M junto con una operación de multiplicación externa $\nu : R \times M \to M$, tal que para todos los $m, n \in M$ y todos los $r, s \in R$, se satisfacen las siguientes condiciones:

1. $(rm) \in M$.
2. $r(m + n) = rm + rn$.
3. $(r + s)m = rm + sm$.
4. $(rs)m = r(sm)$.

Un R-módulo se parece mucho a un espacio vectorial, con la diferencia de que los escalares solo necesitan formar un anillo y no necesariamente un campo. Si R es un anillo con unidad 1, tal que $1\alpha = \alpha$ para toda $\alpha \in M$, entonces M es un *R-módulo unitario.* Por otra parte, si R es un campo, el concepto de R-módulo es idéntico al de R-espacio vectorial. Además, abusando un poco de la notación, cuando se haga referencia a un módulo se dará por entendido que se está trabajando con un R-módulo.

Definición 1.3. Sea M un R-módulo. Un subconjunto N de M es llamado un *submódulo* de M, si N mismo es un módulo al restringir la operación ν a N, i.e., si con $\nu : R \times N \to N$ es un módulo por sí mismo (con la operación inducida por R sobre M).

Definición 1.4. Sea M un módulo sobre un anillo R y sea S un subconjunto de M. Se dice que S es una *base* de M si S es no vacío, S genera a M y S es linealmente independiente. En este sentido, se entenderá como *módulo libre* a aquellos módulos que admiten una base, además del módulo cero.

Notación 1.1. En adelante, se emplearán los corchetes angulares para indicar que un elemento o un conjunto de elementos forman una base generadora de alguna estructura en particular. Por ejemplo, $K = \langle x_1, x_2, \ldots, x_n \rangle$, indica que la estructura K es generada por el conjunto $\{x_i : 1 \leq i \leq n\}$.

Para el siguiente teorema es preciso aclarar que el campo F considerado es un dominio entero de ideales principales; además, por simplicidad se dará la demostración cuando F es finitamente generado, es decir, tiene una base finita $\{x_i\}, i \in \{1, \ldots, n\}$. El lector interesado en el caso infinito puede ver [18]. Apéndice 2, S2. p. 880.

Teorema 1.1 ([18, Thm. 7.1, p. 146])**.** Sea F un módulo libre, y M un submódulo. Entonces, M es libre, y su dimensión es menor o igual que la dimensión de F.

Demostración. Sea M_r la intersección de M con $\langle x_1, x_2, \dots x_r\rangle$, es decir, con el módulo generado por $x_1, \dots, x_r$. Entonces, mediante inducción, se iniciará diciendo que $M_1 = M \cap \langle x_1\rangle$, y por lo tanto, es de la forma $(a_1 x_1)$ para algunos $a_1 \in R$. De ahí que M es el módulo cero o libre de dimensión 1. De esta manera, como hipótesis de inducción se puede asumir que M_r es libre de dimensión menor o igual que r.

Sea $\mathfrak{a}$ el conjunto que consta de todos los elementos $a \in R$ tales que, existe un elemento $x \in M$ que puede ser escrito como $x = b_1x_1 + \cdots + b_rx_r + ax_{r+1}$, es decir,

$$\mathfrak{a} = \{a \in R : \exists x \in M, x = b_1x_1 + \cdots + b_rx_r + ax_{r+1} \ \text{ con } \ b_i \in R;\ 1 \le i \le r\}.$$

Entonces, de la siguiente forma se puede probar que $\mathfrak{a}$ es un ideal. Para a y $a' \in \mathfrak{a}$, se demostrará que $a + a' \in \mathfrak{a}$. Ahora, sean $x,\ x'$ tales que $x = b_1x_1 + \cdots + b_rx_r + ax_{r+1}$ y $x' = b'_1x_1 + \cdots + b'_rx_r + a'x_{r+1}$, luego

$$\begin{aligned} x + x' &= b_1x_1 + \cdots + b_rx_r + ax_{r+1} + b'_1x_1 + \cdots + b'_rx_r + a'x_{r+1} \\ &= (b_1 + b'_1)x_1 + \cdots + (b_r + b'_r)x_r + (a + a')x_{r+1}. \end{aligned} \tag{1.1.1}$$

De esta manera, como $a,\ a' \in R$, entonces $a + a' \in R$ y por la igualdad 1.1.1, se tiene que $a + a' \in \mathfrak{a}$.

Por otra parte, para $h \in R$ y $x = b_1x_1 + \cdots + b_rx_r + ax_{r+1}$ tendremos que

$$hx = (hb_1)x_1 + \cdots + (hb_r)x_r + (ha)x_{r+1}. \tag{1.1.2}$$

Entonces, como h y a están en R, se tiene que, ha está en R también, además por la igualdad 1.1.2, se tiene que $ha \in \mathfrak{a}$. Lo que completa la demostración de que $\mathfrak{a}$ es un ideal.

Adicionalmente, dado que $\mathfrak{a}$ es principal por ser un ideal de F, este es generado por un elemento, supóngase a_{r+1}. Si $a_{r+1} = 0$, entonces,

$$M_{r+1} = M_r \tag{1.1.3}$$

y así el teorema está demostrado, puesto que por el paso inductivo se asumió que M_r es libre de dimensión menor o igual que r, lo cual también se aplica a M_{r+1} por la igualdad establecida en 1.1.3. Ahora bien, asumiendo que $a_{r+1} \neq 0$, se considera un $w \in M_{r+1}$, tal que el coeficiente de w correspondiente a x_{r+1} es a_{r+1}, entonces para cualquier $h \in M_{r+1}$, se tendrá que el coeficiente de h en la componente x_{r+1} es divisible por a_{r+1}, puesto que es un elemento de $\langle a_{r+1}\rangle$, de esta manera, existe $c \in R$ tal que $h - cw$ se halla en M_r. En efecto, sea $h = b'_1x_1 + \cdots + b'_rx_r + b'_{r+1}x_{r+1}$, donde

b'_{r+1} es un múltiplo de a_{r+1} y pertenece a $\mathfrak{a}$, dado que este es un ideal principal; sin pérdida de generalidad se asumirá $b'_{r+1} = ca_{r+1}$ y $w = b_1x_1 + \cdots + b_rx_r + a_{r+1}x_{r+1}$; por lo dicho anteriormente,

$$\begin{aligned} h - cw &= b'_1x_1 + \cdots + b'_rx_r + b'_{r+1}x_{r+1} - c(b_1x_1 + \cdots + b_rx_r + a_{r+1}x_{r+1}) \\ &= (b'_1 - cb_1)x_1 + \cdots + (b'_r - cb_r)x_r + (b'_{r+1} - ca_{r+1})x_{r+1} \\ &= (b'_1 - cb_1)x_1 + \cdots + (b'_r - cb_r)x_r. \end{aligned}$$

Por tanto

$$M_{r+1} = M_r + \langle w \rangle.$$

Por otro lado, es claro que $M_r \cap \langle w \rangle$ es cero, ya que si se asume que $z \in M_r \cap \langle w \rangle$ entonces, se tendrá por una parte

$$z = a_1x + \cdots + a_rx_r$$

y por otra

$$z = kw = kb_1x + \cdots + kb_rx_r + ka_{r+1}x_{r+1}.$$

Si los restamos, se sigue que

$$z - z = (a_1 - kb_1) + \cdots + (a_r - kb_r)x_r + ka_{r+1}x_{r+1} = 0.$$

Pero al ser $\{x_i\}$ un conjunto linealmente independiente, deberá cumplirse que $ka_{r+1} = 0$, de donde resulta $k = 0$, puesto que se asumió que $a_{r+1} \neq 0$. Luego, $z = kw = 0w = 0$; de ahí que esta suma sea directa. De este modo queda probado el teorema.

□

Si K es un subcampo del campo finito $\mathbb{F}_p$, con p primo, entonces K debe contener los elementos 0 y 1, por tanto, todos los otros elementos de $\mathbb{F}_p$ por la clausura de K bajo la adición. De ahí se sigue que $\mathbb{F}_p$ no contiene subcampos propios y resultan las siguientes definiciones.

Definición 1.5. Un campo que no contiene subcampos propios es llamado un *campo primo.*

Según la definición anterior, cualquier campo finito de orden p, con p primo, es un campo primo. Otro ejemplo de un campo primo es el campo $\mathbb{Q}$ de los números racionales, para la demostración de este hecho referirse a [2]. p. 12.

Definición 1.6 (Extensión de cuerpos)**.** Sea F un campo. Un submódulo K de F, que es por sí mismo un campo bajo las operaciones de F, es llamado un *subcampo* de F. En el otro sentido, F se conoce como una *extensión* (*campo de extensión*) de K. Si $K \neq F$, se dice que K es un *subcampo propio* de F.

Adicionalmente, Sea $K \subset F$ una extensión de cuerpos y sea M un subconjunto de elementos de F. Denotaremos por $K(M)$ el menor subcuerpo de F que contiene a K y a M.

Notemos que inicialmente nada garantiza que el cuerpo $K(M)$ exista. Consideremos la familia $\mathfrak{F} = \{E_i : E_i \text{ subcuerpo de } F, M \subset E_i, K \subset E_i\}$.
Claro está $\mathfrak{F}$ es no vacía dado que F pertenece a $\mathfrak{F}$. Ahora consideremos,

$$E = \bigcap_{E_i \in \mathfrak{F}} .$$

Claramente E es un cuerpo y como cada $E_i \in \mathfrak{F}$ contiene a M y K, entonces E contiene a M y K. Más aún, todo cuerpo en estas condiciones es un miembro de $\mathfrak{F}$ y por tal motivo contiene a la intersección E, i.e., E es el menor subcuerpo de F con la propiedades pedidas y así $E = K(M)$ efectivamente existe.

Definición 1.7. Sea $K \subset L$ una extensión de cuerpos. Si L, considerado como un espacio vectorial sobre K, es de dimensión finita, entonces L es llamado una *extensión finita* de K. La dimensión del espacio vectorial L sobre K es conocida como el *grado* de L sobre K y se simboliza $[L : K]$.

Para un subcampo K de F y un $\alpha \in F$, se tiene que si α satisface una ecuación polinomial no trivial, $a_n\alpha^n + \cdots + a_1\alpha + a_0 = 0$ con $a_i \in K$, no todos iguales a 0, entonces α se llama *algebraico* sobre K. De ahí que una extensión L de K es llamada *algebraica* sobre K (o una *extensión algebraica* de K) si todo elemento de L es algebraico sobre K. En cambio si α no satisface ninguna ecuación como la descrita, se dice que α es un elemento trascendente de L sobre K.

Definición 1.8 (Extensión simple y elemento definidor)**.** Para el caso finito, $M = \{\alpha_1, \ldots, \alpha_n\}$ se denota $K(M) = K(\alpha_1, \ldots, \alpha_n)$. Si M consta de un único elemento $\alpha \in F$, entonces $L = K(\alpha)$ se dice que es una *extensión simple* de K, denotada también como L/K y α es llamado un *elemento definidor* o *elemento primitivo* de L sobre K.

Teorema 1.2 ([19, Thm. 1.86, p. 33])**.** Sea $\alpha \in F$ algebraico de grado n sobre K y sea g el polinomio minimal de α sobre K.

a) $K(\alpha)$ es isomorfo a $K[x]/(g)$.

b) $[K(\alpha) : K] = n$ y $\{1, \alpha, \ldots, \alpha^{n-1}\}$ es una base de $K(\alpha)$ sobre K.

c) Todo $\alpha \in K(\alpha)$ es algebraico sobre K y su grado sobre K es un divisor de n.

Observación 1.1. Según la Definición 1.8, L está generado por un solo elemento. Dicho de otro modo, un *elemento primitivo* de una extensión de cuerpos L/K es un elemento α de L tal que

$$L = K(\alpha)$$

o en otras palabras, L está generado por α sobre K.

Teorema 1.3. Sea $K \subset L$ una extensión de cuerpos tales que $L = K(\alpha)$, entonces todo elemento de L puede ser escrito como cociente de dos polinomios en α con coeficientes en K.

Para el caso en que α es un elemento algebraico de L sobre K, se tiene que si la extensión L/K es simple, es decir, si admite un elemento primitivo, entonces L es una extensión finita de K.

Definición 1.9. Si $\alpha \in F$ es algebraico sobre K, entonces, el polinomio mónico y de menor grado univocamente determinado $g \in K[x]$ que genera al ideal $J = \{f \in K[x] : f(\alpha) = 0\}$ de $K[x]$ es llamado el *polinomio minimal (polinomio definidor o polinomio irreducible)* de α sobre K. Por el *grado* de α sobre K, se hará referencia al grado de g.

Ejemplo 1.1. Sea α una raíz del polinomio $1 + x + x^2 \in \mathbb{F}_2[x]$. Es claro que los dos polinomios x y $1 + x$ no son polinomios minimales de α. De ahí que, $1 + x + x^2$ es un polinomio minimal de α. Por otra parte, dado que $1 + (1 + \alpha) + (1 + \alpha)^2 = 1 + 1 + \alpha + 1 + \alpha^2 = 1 + \alpha + \alpha^2 = 0$ y como $1 + \alpha$ no es una raíz de x o de $1 + x$ en $\mathbb{F}_2[x]$, $1 + x + x^2$ es también un polinomio minimal de $1 + \alpha$.

Teorema 1.4 ([19, Thm. 1.82, p. 31])**.** Si $\alpha \in F$ es algebraico sobre K, entonces su polinomio minimal g sobre K tiene las siguientes propiedades

i) g es irreducible en $K[x]$.

ii) Para $f \in K[x]$ se tiene que $f(\alpha) = 0$, si y solo si, g divide a f.

iii) g es el polinomio mónico en $K[x]$ de grado mínimo teniendo a α como una raíz.

Teorema 1.5 ([19, Thm. 1.87, p. 34])**.** Sea $f \in K[x]$ irreducible sobre el campo K. Entonces, existe una extensión algebraica simple de K con una raíz de f como elemento definidor.

Teorema 1.6 ([26, Thm. 4.3.1, p. 144])**.** Si p es un número primo y n es un entero positivo, entonces existe (bajo isomorfismo) exactamente un campo de tamaño $q = p^n$, que es denotado por $\mathbb{F}_q$ o $GF(q)$. Además, todo campo finito tiene tamaño p^n, para algún número primo p y un entero positivo n, es decir, $q = p^n$.

Particularmente, de acuerdo a los objetivos perseguidos en este escrito, se considerará $\alpha \in \mathbb{F}_{q^m}$, entonces un polinomio minimal de α con respecto a $\mathbb{F}_q$, es un polinomio mónico $f(x)$ no cero de grado mínimo en $\mathbb{F}_q[x]$ tal que $f(\alpha) = 0$.

1.1.1. Campos finitos

Definición 1.10 (Campo)**.** Un *campo* es un conjunto no vacío F de elementos con dos operaciones '+' (llamada adición) y ' $\cdot$ ' (llamada multiplicación) que satisfacen los siguientes axiomas. Para todos los $a, b, c \in F$:

(i) F es cerrado bajo + y $\cdot$; es decir, $a + b$ y $a \cdot b$ están en F.

(ii) Leyes conmutativas: $a + b = b + a$, $a \cdot b = b \cdot a$.

(iii) Leyes asociativas: $(a + b) + c = a + (b + c)$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

(iv) Leyes distributivas: $a \cdot (b + c) = a \cdot b + a \cdot c$.

Además, deben existir en F dos elementos identidad diferentes 0 y 1 (llamados *identidades aditiva* y *multiplicativa*, respectivamente) que satisfacen lo siguiente:

(v) $a + 0 = a$ para todo $a \in F$.

(vi) $a \cdot 1 = a$ y $a \cdot 0 = 0$ para todo $a \in F$.

(vii) Para cualquier a en F, existe un elemento inverso aditivo $(-a)$ en F tal que $a + (-a) = 0$.

(viii) Para cualquier $a \neq 0$ en F, existe un elemento inverso multiplicativo a^{-1} en F tal que $a \cdot a^{-1} = 1$.

Más propiedades de un campo pueden ser deducidas de la definición.

Lema 1.1 ([27, Thm. 3.1.3, p. 21])**.** Sean a, b dos elementos cualquiera de un campo F. Entonces

(i) $(-1) \cdot a = -a$;

(ii) $ab = 0$ implica $a = 0$ o $b = 0$.

Definición 1.11. Un campo que contenga solamente un número finito de elementos es llamado un *campo finito* y a un conjunto F que satisface los axiomas (i) - (vii) en la Definición 1.10 es llamado un *anillo (conmutativo)* con $1 = 1_F$.

Teorema 1.7 ([27, Thm. 3.1.9, p. 21])**.** $\mathbb{Z}_m$ es un campo, si solo si, m es un primo.

Definición 1.12. (Característica) Sea R un anillo. Si existe $n \in \mathbb{N}$ tal que $na =$ para todo $a \in R$, el menor tal entero positivo es llamado la característica de R. Si ningún tal entero positivo, se dice que R tiene característica cero. La característica de R es denotada por $car(R)$.

Observación 1.2. Es posible demostrar que:

1. Sea R un anillo con 1. Si R tiene orden infinito respecto a la adición, entonces $car(R) = 0$. Si R tiene orden finito n con respecto a la suma, $car(R) = n$.

2. La característica de un dominio entero (o de integridad) D, en particular de un cuerpo F, es 0 o un primo p.

3. Sea F un cuerpo finito. Entonces:

   (i) $car(F) = p$, para algún primo p y,

   (ii) $\mathbb{Z}_p \sim F_p \subset F$, donde F_p es un subcuerpo de F.

Teorema 1.8 ([27, Thm. 3.1.12, p. 21])**.** La característica de un campo es o 0 o un número primo.

Teorema 1.9 ([27, Thm. 3.1.14, p. 22])**.** Un campo finito F de característica p contiene p^n elementos para algún entero $n \geq 1$.

La siguiente es una particularización de la definición 1.8.

Definición 1.13. Un elemento α en un campo finito $\mathbb{F}_q$ es llamado un *elemento primitivo* de $\mathbb{F}_q$ si $\mathbb{F}_q = \{0, \alpha, \alpha^2, \ldots, \alpha^{q-1}\}$.

1.2. La transmisión de los datos

Desde el punto de vista de la Teoría de Códigos, esta supone las siguientes reglas

- Emisor y receptor conocen la codificación de los mensajes a transmitir, es decir, conocen la codificación del mensaje y la codificación del canal. Por ejemplo, en toda comunicación hay un acuerdo en el tipo de idioma en que se envían y reciben los mensajes entre emisor y receptor.
- Si ocurren errores, el receptor es capaz de detectarlos. Por ejemplo, si es sabido que el idioma empleado es el español, entonces, el receptor podrá saber que si aparecen palabras como "my", "you", "for", etc., entonces, ocurrió un error en el mensaje.
- Por un canal de comunicación, solo palabras del nuevo lenguaje son enviadas. Por ejemplo, suponiendo que la codificación del canal se realizó con palabras de longitud ocho, si se recibe una palabra de longitud siete o nueve, es posible identificar directamente que ha ocurrido un error.
- Los mensajes recibidos solo deben contener símbolos que se manejan en el canal. Por ejemplo, si la codificación del canal emplea un alfabeto binario, entonces al recibirse los mensajes, los elementos que lo conformen deberán ser solo ceros y/o unos.

Definición 1.14 (Códigos de bloque)**.** Sean $A = \{a_1, a_2, \ldots, a_q\}$ 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 es llamado un *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 C no es de bloque, se dice que C es de *longitud variable*. Normalmente, el alfabeto considerado es un cuerpo finito de orden q, es decir, $A = \mathbb{F}_q$, donde q es una potencia de un primo y representa el número de elementos que este contiene.

Notación 1.2. En ocasiones las componentes de las palabras código estarán separadas por comas y entre paréntesis, pero cuando no haya lugar a confusión estas se escribirán sin símbolos de separación

ni paréntesis, es decir, se podrán expresar como $(x_1, x_2, \ldots, x_n)$ o simplemente como $x_1x_2 \ldots x_n$. En adelante, salvo que se aclare lo contrario, los códigos utilizados serán de bloque.

Ejemplo 1.2. Un par de ejemplos de códigos de longitud variable son

- El código Morse. Ver [10].
- El código $C = \{(0), (1,0), (1,1), (1,0,0), (1,0,1), (1,1,1)\}$ con alfabeto $A = \{0,1\}$.

Ejemplo 1.3. Algunos ejemplos de códigos de bloque son

- $C_1 = \{(0,0,0), (0,1,1), (1,1,0), (1,1,1)\}$ sobre $\mathbb{F}_2$ es un $(3,4)$-código binario.
- $C_2 = \{(2,0,1,0), (1,0,2,0), (1,1,2,2), (2,2,1,1), (2,1,2,1)\}$ sobre $\mathbb{F}_3$ es un $(4,5)$-código ternario.
- Sea α algebraico sobre $\mathbb{F}_4$, entonces, $C_3 = \{(1,\alpha,\alpha), (\alpha+1,0,1), (1,0,\alpha), (\alpha,\alpha+1,1)\}$ sobre $\mathbb{F}_4$ es un $(3,4)$-código cuaternario.

1.3. Distancia de Hamming

Definición 1.15 (Distancia de Hamming)**.** Sean $\mathbf{x}$ e $\mathbf{y}$ palabras de longitud n sobre un alfabeto A, la *distancia de Hamming* entre $\mathbf{x}$ e $\mathbf{y}$, denotada por $d(\mathbf{x}, \mathbf{y})$, se define como el número de coordenadas en las cuales $\mathbf{x}$ e $\mathbf{y}$ difieren, es decir, si $\mathbf{x} = x_1x_2 \ldots x_n$ e $\mathbf{y} = y_1y_2 \ldots y_n$, entonces

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

Ejemplo 1.4. Sea $A = \{0,1,2\}$, $\mathbf{x} = 1011210$ e $\mathbf{y} = 1021010$, entonces, $d(\mathbf{x}, \mathbf{y}) = 2$, puesto que estas difieren en 2 de las 7 coordenadas correspondientes.

Notación 1.3. Hasta el momento, las palabras se han denotado con letra en negrilla y sus componentes en letra sin negrilla, en adelante, si el contexto es claro, se emplearán letras sin negrilla para hacer referencia a alguna palabra en particular.

Teorema 1.10 ([26, Thm. 4.2.1, p. 129])**.** Sean x, y, z palabras de longitud n sobre A. Entonces

1. $0 \leq d(x,y) \leq n$ y $d(x,y) = 0$, si y solo si, $x = y$.
2. $d(x,y) = d(y,x)$.
3. $d(x,z) \leq d(x,y) + d(y,z)$. (*Desigualdad triangular*).

Es decir,

- Las distancias son no negativas y la única palabra a distancia cero de x es el mismo x.

- La distancia de Hamming es una función simétrica.
- La distancia de Hamming satisface la desigualdad triangular: la longitud de un lado de un triángulo es menor que la suma de las longitudes de los otros dos lados.

Demostración. Los dos primeros ítems resultan de la definición de distancia de Hamming, por tanto, únicamente se demostrará el tercero de ellos. Sean

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

Claramente

$$B \subseteq A \quad \text{entonces} \quad A^c \subseteq B^c.$$

De esta manera,

$$d(\mathbf{x}, \mathbf{z}) = |A^c| \leq |B^c|,$$

donde,

$$\begin{aligned} B^c &= \{i : x_i \neq y_i \vee y_i \neq z_i\} \\ &= \underbrace{\{i : x_i \neq y_i\}}_{C} \cup \underbrace{\{i : y_i \neq z_i\}}_{D}, \end{aligned}$$

luego,

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

□

Lo anterior demuestra que la distancia de Hamming permite definir una métrica sobre los conjuntos $\mathbb{F}_q^n$.

1.3.1. Decodificación por distancia mínima

Si x es una palabra recibida, esta se decodifica como c_x, si $d(x, c_x)$ es la mínima entre todas las palabras $c \in C$ y x, es decir

$$d(x, c_x) = \text{mín}\{d(x, c), c \in C\}.$$

En este momento se puede hablar de dos tipos de decodificación, *la decodificación por distancia mínima completa y la decodificación por distancia mínima incompleta.* En la primera, cuando hay un empate de distancias mínimas entre la palabra recibida y dos o más palabras del código, el código

se encargará de escoger una al azar entre aquellas palabras que haya empate, es decir, no hay algún criterio de decisión definido, mientras que en la segunda, en una situación similar, el código pedirá retransmisión del mensaje.

En lo que sigue de este trabajo, cuando no se especifique lo contrario, se asumirá que se está trabajando con el segundo tipo de decodificación.

Ejemplo 1.5. Para el código $C = \{01101, 00011, 10110, 11000\}$, se empleará la decodificación por distancia mínima incompleta para decodificar las siguientes palabras recibidas:

a) x= 01111.

b) y= 11011.

Obsérvese que ninguna de las dos palabras recibidas pertenecen al código, por tanto para su decodificación se procede a determinar las distancias entre estas y las palabras del código, lo que permite observar cual es la menor de ellas, así:

a) $d(01111, 01101) = 1$,
$d(01111, 00011) = 2$,
$d(01111, 10110) = 3$,
$d(01111, 11000) = 4$.
En este caso, la palabra recibida x se decodifica como 01101.

b) $d(11011, 01101) = 3$,
$d(11011, 00011) = 2$,
$d(11011, 10110) = 3$,
$d(11011, 11000) = 2$.
Aquí el código solicitará la retransmisión del mensaje.

Definición 1.16 (Distancia mínima)**.** La *distancia mínima* de un código C, denotada por $d(C)$, se define por

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

Vale recalcar aquí que si el código C contiene k elementos, entonces el proceso para determinar la distancia mínima de este código requiere el cálculo de $\binom{k}{2}$ distancias, lo cual puede resultar en un trabajo bastante dispendioso si el número de elementos es grande.

Notación 1.4. Un código C de longitud n, tamaño M y distancia mínima d se denotará como un $(n, M, d)-$código.

1.3.2. Detección de errores

Definición 1.17 (Código detector de t-errores)**.** 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 del código. Un código C detecta exactamente t-errores, si este detecta t errores, pero no detecta $t+1$ errores.

Lo anterior implica que existe al menos una palabra en C que al cambiarle $t+1$ coordenadas origina una nueva palabra del código, lo cual hace imposible que se pueda determinar si hubo o no errores en la transmisión de dicha palabra.

Teorema 1.11 ([26, Thm. 4.2.2, p. 131])**.** Un código C detecta exactamente t-errores, si y solo si, $d(C) = t+1$.

Definición 1.18 (Código corrector de t-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.

Lo anterior implica que ocurridos $t+1$ errores al interior de una palabra código, la palabra originada se halla a una distancia menor de una palabra distinta de C que de la palabra original, lo que hace imposible la corrección del o los errores que se presentaron.

Teorema 1.12 ([26, Thm. 4.2.3, p. 132])**.** Un código C corrige exactamente t-errores, si y solo si, $d(C) = 2t+1$ ó $d(C) = 2t+2$.

Corolario 1.1 ([26, Cor. 4.2.4, p. 132])**.** $d(C) = d$, si y solo si, C corrige exactamente $\left\lfloor \frac{d-1}{2} \right\rfloor$-errores.

Ejemplo 1.6. El código ternario $C = \{000000, 000111, 111222\}$ es un detector de exactamente 2 errores pero también es un corrector de un error, puesto que $d = 3$ y al enviar cualquier palabra código y cambiar cualquieras 2 o menos coordenadas en ella, la palabra resultante no pertenece al código, es decir,

- $000000 \to 000111$ necesita cambiar 3 coordenadas,
- $000000 \to 111222$ necesita cambiar 6 coordenadas,
- $000111 \to 111222$ necesita cambiar 6 coordenadas.

1.3.3. Peso de un código

Definición 1.19 (Peso)**.** Sea x una palabra en $\mathbb{F}_q^n$, *el peso* de x, denotado por $wt(x)$, se define como el número de coordenadas no nulas de x, es decir,

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

Donde $\mathbf{0}$ es la palabra cero de longitud n.

Definición 1.20 (Peso de un Código)**.** Sea C un código. *El peso mínimo (de Hamming)* de C, denotado por $wt(C)$, es el mínimo de los pesos de las palabras no nulas de C, es decir,

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

Ejemplo 1.7. Para los códigos $C_1 = \{000,\ 101,\ 010,\ 111\}$ y $C_2 = \{000000,\ 000111,\ 001110,\ 011100\}$ se tiene que

- $wt(C_1) = 1 = d(C_1)$ y
- $wt(C_2) = 3 \neq d(C_2) = 2$;

El ejemplo anterior revela que en un código cualquiera el peso y la distancia mínima no siempre coinciden.

Definición 1.21 (Código de peso constante)**.** Un código de *peso constante* C, es aquel en el que todas sus palabras tienen el mismo peso, es decir, $wt(c) = k$, para $k \in \mathbb{N}$ para todo $c \in C$.

Notación 1.5. Un (n, M, d, w)-código es un código de longitud n, tamaño M, distancia mínima d y peso constante w.

Definición 1.22 (Enumeradores de Peso)**.** Si C es un (n, M)-código, se denota con A_k el número de palabras de C de peso k, es decir,

$$A_k = \#\{c \in C :\ wt(c) = k\}.$$

Los números $A_0, \ldots, A_n$, se conocen como *la distribución de pesos de* C y la suma formal

$$W_C(s) = \sum_{k=0}^{n} A_k s^k,$$

es el *enumerador de peso de* C.

Ejemplo 1.8. Sea el código

$$C = \{000000, 000111, 001110, 011100,$$
$$100011, 110001, 111000, 111111\}.$$

Entonces, tendremos que $A_0 = A_6 = 1; A_1 = A_2 = A_4 = A_5 = 0$ y $A_3 = 6$. El enumerador de peso será

$$W_C(s) = 1 + 6s^3 + s^6.$$

1.3.4. Códigos sistemáticos

Dado que un (n, M)-código q-ario tiene M palabras código, este puede ser usado para codificar M mensajes fuente. Sin embargo, ciertos tipos de código son de mayor facilidad de uso para la codificación y decodificación de palabras, entre ellos están los códigos sistemáticos, que se definen a continuación.

Definición 1.23. Un (n, q^k)-código q-ario es llamado *sistemático* si existen k posiciones $i_1, i_2, \ldots, i_k$, no necesariamente consecutivas, con la propiedad de que, al restringir todas las palabras código a dichas posiciones se obtienen todas las q^k posibles palabras código de longitud k. De esta manera, al conjunto $\{i_1, i_2, \ldots, i_k\}$ se le conoce como *conjunto información* y al conjunto de símbolos de las palabras código en esas posiciones se los llama *símbolos de información*.

Si un mensaje fuente puede ser representado como el conjunto de todas las palabras q-arias de longitud k, entonces un código sistemático C q-ario de tamaño q^k puede ser usado para codificar cada palabra fuente a partir de la incrustación de estas sin ningún cambio dentro de cada palabra código de C, como se muestra en el siguiente ejemplo.

Ejemplo 1.9. El código binario

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

es sistemático en la primera y tercer posiciones o coordenadas, puesto que si el mensaje fuente es $\{00, 01, 10, 11\}$, es posible codificarlo como sigue

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

El tipo de codificación presentado en el ejemplo previo se conoce como *códificación sistemática*. Claramente, la codificación sistemática hace que el trabajo inverso sea muy fácil, puesto que es posible leer las palabras mensaje directamente de las palabras código.

1.4. Códigos lineales

Definición 1.24 (Código lineal)**.** Un *código lineal* L de longitud n sobre $\mathbb{F}_q$ es un subespacio vectorial de $\mathbb{F}_q^n$, donde $\mathbb{F}_q$ es el campo finito de dimensión q.

Notación 1.6. 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 1.10. Los siguientes conjuntos son ejemplos de subespacios vectoriales sobre $\mathbb{F}_q$ y por lo tanto, códigos lineales.

- $L_1 = \mathbb{F}_q^n$ y $L_2 = \{\mathbf{0}\}$, donde $\mathbf{0}$ es el vector nulo en $\mathbb{F}_q^n$.

- $L_3 = \{(\lambda, \ldots, \lambda) : \lambda \in \mathbb{F}_q\}$ es un $[n, 1, n]$-código lineal (código de repetición).
- $L_4 = \{(0,0,0,0), (1,0,1,0), (0,1,0,1), (1,1,1,1)\} \subset \mathbb{F}_2^4$ es un $[4, 2, 2]$-código lineal.
- $L_5 = \{(0,0,0,0), (1,1,0,0), (2,2,0,0)\} \subset \mathbb{F}_3^4$ es un $[4, 1, 2]$-código lineal.

Un código lineal definido sobre el campo $\mathbb{F}_2$, $\mathbb{F}_3$ ó $\mathbb{F}_4$ se denomina *código lineal binario*, *ternario* o *cuaternario*, respectivamente.

Teorema 1.13 ([26, Thm. 4.3.5, p. 146])**.** Sea L un código lineal sobre $\mathbb{F}_q$. Entonces, $d(L) = wt(L)$.

Lo anterior revela una ventaja de trabajar con códigos lineales, y es que ya no se requiere calcular todas las posibles distancias para determinar la distancia mínima del código. De esta manera, si el código tiene tamaño M bastará con observar los $M - 1$ pesos de las palabras no nulas del código, en lugar de las $\binom{M}{2}$ distancias.

Ejemplo 1.11. Considere el código lineal binario $L = \{0000, 1000, 0100, 1100\}$. Como $wt(1000) = 1$, $wt(0100) = 1$, $wt(1100) = 2$, entonces $d(L) = 1$.

Teorema 1.14 ([27, Thm. 4.1.15, p. 42])**.** Sea L un código lineal sobre $\mathbb{F}_q$. Si $\dim(L) = k$, entonces

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

Corolario 1.2. Sea L un código lineal de longitud n sobre $\mathbb{F}_q$. Entonces $\dim(L) = \log_q |L|$.

Definición 1.25 (Matriz generadora)**.** Sea L un $[n, k]$-código lineal. Una matriz G de tamaño $k \times n$ cuyas filas forman una base para L es llamada una *matriz generadora* para el código L.

Observación 1.3. Si L es un $[n, k]$-código lineal con matriz generadora G, entonces

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

Este principio es empleado para realizar la codificación de las palabras mensaje $x \in \mathbb{F}_q^k$ a partir de la matriz G, dando lugar a cadenas de mensajes codificados de longitud n; dicho método de codificación resulta en un tipo de *codificación no sistemática*.

1.4.1. Código dual

Antes de abordar este tema, es importante definir el tipo de producto que se considerará, es por eso que se presenta la siguiente definición.

Definición 1.26 (producto escalar)**.** Si $\mathbf{u}$ y $\mathbf{v}$ son vectores en un espacio bidimensional o tridimensional y θ el ángulo entre ellos, entonces el producto escalar, producto punto o producto euclidiano interior, denotado por $\mathbf{u} \cdot \mathbf{v}$, se define como

$$\mathbf{u} \cdot \mathbf{v} = \begin{cases} \|\mathbf{u}\| \, \|\mathbf{v}\| \cos\theta & \text{si } \mathbf{u} \neq 0 \text{ y } \mathbf{v} \neq 0. \\ 0 & \text{si } \mathbf{u} = 0 \text{ o } \mathbf{v} = 0. \end{cases}$$

Sin embargo, para el caso particular en que $\mathbf{u}$, $\mathbf{v}$ son vectores del espacio tridimensional, haciendo algunas sustituciones algebraicas, es posible definir el producto punto de la siguiente forma

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \, \|\mathbf{v}\| \, cos\theta = u_1 v_1 + u_2 v_2 + u_3 v_3.$$

Visto de otra manera, el producto punto se puede abordar como un producto matricial, es decir

$$\mathbf{u} \cdot \mathbf{v} = \begin{pmatrix} u_1 & u_2 & u_3 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = u_1 v_1 + u_2 v_2 + u_3 v_3.$$

Así para espacios n-dimensionales, es posible generalizar esta definición de la siguiente manera

$$\mathbf{u} \cdot \mathbf{v} = \begin{pmatrix} u_1 & u_2 & \cdots & u_n \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n.$$

Definición 1.27. Sea L un $[n, k]$-código lineal. El conjunto

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

es el *código dual* de L.

Teorema 1.15 ([26, Thm. 5.1.3, p. 199])**.** Sea L un $[n, k]$-código lineal sobre $\mathbb{F}_q$.

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. $L^{\perp}$ es un $[n, n-k]$-código lineal, por lo tanto $\dim(L) + \dim(L^{\perp}) = n$.

3. $(L^{\perp})^{\perp} = L$.

Ejemplo 1.12. Hallar el código dual $L^{\perp}$ del código lineal binario $L = \{000, 101, 010, 111\}$.
Según el Corolario 1.2, $\dim(L) = \log_2 4 = 2$ y por la parte 2 del Teorema 1.15, $\dim(L^{\perp}) = 1$. Luego, como se puede observar, dos elementos linealmente independientes de L son 101 y 010. Así una matriz generadora para L es

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

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

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

Entonces, los elementos en la base para el código dual $L^{\perp}$ deben satisfacer que $x_1 + x_3 = 0$ y $x_2 = 0$. De ahí que una base para $L^{\perp}$ es 101 y por lo tanto $L^{\perp} = \{000, 101\}$.

Definición 1.28. Una matriz generadora para el código dual $L^{\perp}$, se denomina *matriz de control de paridad* H para el código lineal L.

Vale aclarar que *una matriz en la forma estándar* es aquella que a través de transformaciones lineales por filas ha sido llevada a la forma $G = [I_k \mid A]$, donde I_k es la matriz identidad de orden k. Una característica importante de esta matriz es que permite realizar una codificación sistemática de las palabras mensaje en $\mathbb{F}_q^k$.

Teorema 1.16 ([15, Thm. 1.2.1, p. 4])**.** Si $G = [I_k \mid A]$ es una matriz generadora para el (n, k)-código lineal L, entonces $H = [-A^{\top} \mid I_{n-k}]$ es una matriz de control de paridad para L.

Observación 1.4. Una matriz de control de paridad para un código lineal L en $\mathbb{F}_q^n$ de dimensión k es de tamaño $(n - k) \times n$.

Observación 1.5. La matriz H es llamada *la matriz de control de paridad* para un código lineal L, puesto que

$$L = \{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{x}H^{\top} = \mathbf{0}\};$$

de esta manera, siendo $H = (h_{ij})$ y $\mathbf{x} = (x_i)$, se tiene que $\mathbf{x}H^{\top} = 0$ tiene la forma

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

Así, las filas de H son los coeficienes de un sistema de ecuaciones lineales cuyas soluciones son precisamente las palabras código de L. A estas ecuaciones lineales se las conoce como *ecuaciones de control de paridad*.

1.4.2. Decodificación por síndrome

Definición 1.29 (Síndrome)**.** Sea L un $[n,k]$-código lineal 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 $s(\mathbf{x})$.

Definición 1.30 (Clases Laterales)**.** Sean L un código lineal de longitud n sobre $\mathbb{F}_q$ y $\mathbf{x} \in \mathbb{F}_q^n$ un vector de longitud n. Se define la *clase lateral* de L determinada por $\mathbf{x}$, como el conjunto

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

Dado que L es un grupo abeliano, se tiene que $L + \mathbf{x} = \mathbf{x} + L$. Es de resaltar que este conjunto también es un espacio vectorial sobre $\mathbb{F}_q$ con las operaciones

- $a(\mathbf{x} + L) = a\mathbf{x} + L$. Con $a \in \mathbb{F}_q$.
- $(\mathbf{x} + L) + (\mathbf{y} + L) = (\mathbf{x} + \mathbf{y}) + L$.

Por lo tanto, $\mathbf{x}+L = \mathbf{y}+L$, si y solo si, $\mathbf{x}-\mathbf{y} \in L$. A partir de esto se presenta el siguiente resultado.

Teorema 1.17 ([26, Thm. 5.1.6, p. 203])**.** Sea L un $[n,k]$-código, con matriz de control de paridad H. Entonces $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ tienen el mismo síndrome, si y solo si pertenecen a la misma clase lateral de L.

Teorema 1.18 ([26, Thm. 5.1.7, p. 204])**.** Sea L un código lineal con matriz de control de paridad H. Decodificar por distancia mínima es equivalente a 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 $\mathbf{x}$.

1.4.3. Arreglo estándar o arreglo Slepiano

Para comprender el proceso de decodificación descrito en el teorema anterior, se emplea una herramienta conocida como el arreglo estándar o arreglo Slepiano, que lleva su nombre gracias a David Slepian [1] (1960). Para L el arreglo es el siguiente.

$\mathbf{0}$	$\mathbf{c_1}$	$\mathbf{c_2}$	$\dots$	$\mathbf{c_m}$
a_1	$a_1 + c_1$	$a_1 + c_2$	$\dots$	$a_1 + c_m$
a_2	$a_2 + c_1$	$a_2 + c_2$	$\dots$	$a_2 + c_m$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
a_s	$a_s + c_1$	$a_s + c_2$	$\dots$	$a_s + c_m$.

[1] David S. Slepian. (Junio 30, 1923 a Noviembre 29, 2007). Matemático americano. Es reconocido por su trabajo con la teoría algebraica de códigos , la teoría de la probabilidad y el origen de la distribución de los códigos. Fue colega de Claude Shannon y Richard Hamming en los laboratorios Bell.

Donde $m = q^k - 1$ y $s = q^{n-k} - 1$. La primera fila 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 $a_1 \in \mathbb{F}_q$ de menor peso que no esté en L, es decir, la clase lateral $a_1 + L$. De manera general, la i-ésima fila del arreglo, se forma por la suma de cada palabra de la primera fila con una palabra a_i de menor peso que no está aún en el arreglo, esto es, la clase lateral $a_i + L$. Este proceso continúa hasta que el arreglo contenga todas las palabras de $\mathbb{F}_q^n$, o lo que es lo mismo, cuando se tenga $s + 1 = q^{n-k}$ filas. Las palabras a_i, denotadas en texto azul, se llaman líderes de las clases laterales.

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 solo si, están en la misma fila del arreglo, según el Teorema 1.18.

Dependiendo de la forma como los líderes de las clases laterales fueron seleccionados, una palabra recibida x está en la j-ésima columna del arreglo, es decir, $x = c_j + a_i$, para alguna palabra a_i de menor peso en la clase lateral $x + L$. A partir de esto, la decodificación por distancia mínima decodifica a x como c_j, es decir, la palabra código de la columna j. El ejemplo siguiente ilustra muy bien esta situación.

Ejemplo 1.13. Considérese el $[5, 2, 3]$-código binario $L = \{00000, 10110, 01011, 11101\}$.

1. Si se recibe la palabra $x = 11011$. El arreglo estándar sería el siguiente.

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.

   Como puede observarse en el arreglo, la palabra x se encuentra en la sexta fila, en la cual hay solo una palabra con peso mínimo que es 10000, por lo tanto, x se decodifica como 01011, con la certeza de haber corregido el error correctamente.

2. Si se recibe la palabra $x = 11010$. Se tendrá el siguiente arreglo estándar.

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.

En este caso, la palabra x se encuentra en la octava fila, pero el inconveniente que aparece a primera vista es que existen dos palabras con peso mínimo en esa misma fila, 01100 y 10001, por lo cual no se puede asegurar con certeza que la corrección del error sea la adecuada, puesto que el arreglo sería diferente si en lugar de tener la palabra líder 01100, se tuviese a 10001, dado que la elección de las palabras líderes se realiza de forma arbitraria. Esto ocurre porque el código no es capaz de corregir 2 errores, este nada más corrige un error, ver Corolario [1.1]. En esta situación, el código solicitará una retransmisión o hará una decodificación posiblemente errónea de la palabra, dependiendo del tipo de decodificación con el que se esté trabajando.

Observación 1.6. Para un código lineal L con distancia mínima d, todas las palabras de peso menor o igual a $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ son líderes de las clases laterales. En efecto, si se supone que dos palabras distintas x e y tienen peso menor o igual a t y que ambas están en la misma clase lateral. Por la desigualdad triangular se tiene que

$$d(x, y) \leq d(x, 0) + d(0, y) = wt(x) + wt(y) \leq 2t \leq d - 1;$$

esto es una contradicción, ya que esto no es posible debido a que $d(x, y) \geq d$.

Capítulo 2

Códigos cíclicos

En los apartes finales del capítulo anterior se logró develar un poco más de la estructura matemática que existe en el trabajo con códigos, sin embargo, el desarrollo de esos conceptos ha llevado consigo la búsqueda constante de la optimización de los procesos tanto de la codificación como de la decodificación de la información, sin perder de vista la detección y corrección de errores; siguiendo por esa misma línea de ideas, en este capítulo se tratarán una clase especial e importante de códigos lineales, los cíclicos, los cuales hacen uso de herramientas matemáticas que simplifican mucho las tareas ya mencionadas, en especial porque adhieren a esta rama de estudio una teoría muy desarrollada como lo es la de los anillos, en particular la de anillos de polinomios, que cuenta con herramientas muy útiles que se han podido adaptar de gran manera a este campo de investigación.

Dentro de la teoría de códigos cíclicos, se pueden destacar otros conjuntos de códigos que hacen parte de él, como son, ciertos códigos de Hamming (1949-1950), algunos códigos de Golay (1949), los códigos de Red-Muller (1969 apróx.), los códigos de Bose-Chaudhuri-Hocquenghem o códigos BCH (1959-1960), etc. Aunque en este trabajo no se estudiarán estos tipos particulares de códigos cíclicos, es importante que sean mencionados por su papel en la historia y en el presente; el lector interesado puede referirse a [14, 19, 26] para conocer más de ellos. A continuación se hace un pequeño paréntesis en el que se resalta el avance logrado en la transmisión de imágenes a partir de la implementación de los códigos cíclicos.

El código usado por el Mariner 9 para transmitir fotografías desde Marte era un $(32, 64, 16)$-código binario, llamado código Reed-Muller. Este código es muy adecuado para canales con mucho ruido y también tiene un algoritmo de rápida decodificación. La manera como se usó este código se describe en la siguiente breve historia de la transmisión de las fotografías desde las sondas espaciales de la NASA.

2.1. La transmisión de fotografías desde el espacio profundo

1965: El Mariner 4 fue la primera nave espacial en fotografiar otro planeta, tomando 22 fotografías completas de Marte. Cada imagen fue fraccionada en 200×200 elementos de imagen. Cada elemento fue asignado a una 6-upla binaria que representaba uno de las 64 niveles de brillo, desde el blanco (= 000000) hasta el negro (= 111111). Así, el número total de bits (es decir, de dígitos binarios) por cada fotografía fue de 240,000. Estos datos fueron transmitidos a una tasa de $8\frac{1}{3}$ de bits por segundo, por lo cual tomó 8 horas transmitir cada una de las fotografías, además de que tuvieron una muy mala resolución.

1969 - 1972: Muchas fotografías mejoradas enviadas desde Marte fueron tomadas por los Mariners 6, 7 y 9 (El mariner 8 se perdió durante el lanzamiento). Hay tres importantes razones para estas mejoras:

- Cada fotografía fue fraccionada en 700×832 elementos con su correspondiente mejora en la nitidez.
- El Mariner 9 fue la primera nave espacial en ser puesta en órbita al rededor de Marte.
- Se empleó la potencia del $(32, 64, 16)$-código de Reed-Muller para la corrección de errores. Así las 6-uplas binarias que representaban el brillo de un punto en la fotografía fue codificado como una palabra código binaria de longitud 32 (que contenía 26 bits de redundancias). La tasa de transmisión se incrementó de $8\frac{1}{3}$ a 16200 bits por segundo. Aún así, los bits de una imagen fueron producidos por las cámaras del Mariner por encima de los 100000 por segundo, y así los datos tuvieron que ser almacenados en cintas magnéticas antes de ser transmitidos.

1976: El Viking 1 aterrizó suavemente en Marte y regresó con fotografías a color de alta calidad. Sorpresivamente, la transmisión de fotografías a color en forma binaria resultó casi más fácil que la transmisión en blanco y negro. Esto se logró de forma sencilla, tomando simplemente algunas fotografías a blanco y negro varias veces, en cada ocasión con un filtro de diferente color. Las fotografías a blanco y negro fueron transmitidas como ya se dijo y el color de ellas se reconstruyó en la tierra.

5 de Marzo de 1979: Fueron enviadas las primeras fotografías en alta resolución de Júpiter y sus lunas por el Voyager 1.

12 de Noviembre de 1980: El Voyager 1 envía la primera fotografía en alta resolución de Saturno y sus lunas.

25 de Agosto de 1981: El Voyager 2 envía la más perfecta imagen de Saturno.

24 de Enero de 1986: El Voyager 2 se traslada a Urano.

24 de Agosto de 1989: El Voyager 2 se traslada a Neptuno.

2.2. Definiciones y apuntes preliminares

Para el propósito perseguido en esta sección, es preciso dar algunas definiciones previas relacionadas especialmente con los anillos de polinomios, que se constituye como uno de los pilares para el estudio de los códigos cíclicos.

2.2.1. El anillo de polinomios módulo $f(x)$

El anillo $\mathbb{F}[x]$ de polinomios sobre $\mathbb{F}$, donde $\mathbb{F}$ es un campo finito, según la Definición 1.11; es análogo en muchas maneras al anillo $\mathbb{Z}$ de los enteros. De igual manera a como se considera el conjunto de los enteros módulo algún entero m prefijado para obtener el anillo $\mathbb{Z}_m$, es posible considerar los polinomios en $\mathbb{F}[x]$ módulo algún polinomio $f(x)$ prefijado.

Sea $f(x)$ un polinomio fijo en $\mathbb{F}[x]$. Dos polinomios $g(x)$ y $h(x)$ en $\mathbb{F}[x]$ se dice que son *congruentes módulo* $f(x)$ y se simboliza como $g(x) \equiv h(x)(\text{mód} f(x))$, si $g(x) - h(x)$ es divisible por $f(x)$.

Por el algoritmo de la división, cualquier polinomio $a(x)$ en $\mathbb{F}[x]$ es congruente módulo $f(x)$ a un único polinomio $r(x)$ de grado menor que el grad$[f(x)]$; $r(x)$ es justamente el residuo principal cuando $a(x)$ es dividido por $f(x)$. Usualmente se denota como $\mathbb{F}[x]/\langle f(x)\rangle$ al conjunto de polinomios en $\mathbb{F}[x]$ de grado menor que el grad$[f(x)]$, con la adición y la multiplicación llevadas a cabo módulo $f(x)$ como sigue.

Suponga que $a(x)$ y $b(x)$ pertenecen a $\mathbb{F}[x]/\langle f(x)\rangle$. Entonces, la suma $a(x)+b(x)$ en $\mathbb{F}[x]/\langle f(x)\rangle$ es la misma suma que se lleva a cabo en $\mathbb{F}[x]$, porque el grad$[a(x) + b(x)] <$ grad$[f(x)]$. En el otro aspecto, el producto $a(x)b(x)$ en $\mathbb{F}[x]/\langle f(x)\rangle$ es el único polinomio de grado menor que el grad$[f(x)]$ con el cual $a(x)b(x)$ (como un producto en $\mathbb{F}[x]$) es congruente módulo $f(x)$.

Por ejemplo, para calcular $(x + 1)^2$ en $\mathbb{F}_2[x]/\langle x^2 + x + 1\rangle$, se procede de la siguiente manera, teniendo en cuenta que $x^2 + x + 1 \equiv 0$ mód $(x^2 + x + 1)$ implica $x^2 + 1 \equiv x$ mód $(x^2 + x + 1)$, por tanto

$$(x + 1)^2 = x^2 + 2x + 1 = x^2 + 1 \equiv x \text{ mód } (x^2 + x + 1).$$

De ahí que $(x + 1)^2 = x$ en $\mathbb{F}_2[x]/\langle x^2 + x + 1\rangle$.

De la misma manera como $\mathbb{Z}_m$ es un anillo, también lo es $\mathbb{F}[x]/\langle f(x)\rangle$, el cual es llamado el *anillo de polinomios (sobre* $\mathbb{F}$*) módulo* $f(x)$. Si $f(x)$ en $\mathbb{F}_q[x]$ tiene grado n, entonces el anillo $\mathbb{F}_q[x]/\langle f(x)\rangle$ consiste de todos los polinomios de grado menor o igual que $n - 1$. Cada uno de los n coeficientes de dicho polinomio pertenece a $\mathbb{F}_q$ y así

$$|\mathbb{F}_q[x]/\langle f(x)\rangle| = q^n.$$

Ejemplo 2.1. Las tablas de adición y multiplicación para $\mathbb{F}_2[x]/\langle x^2+x+1\rangle$ son fáciles de encontrar, teniendo en cuenta que $x^2+x+1=0$ en dicho anillo. Ellas se presentan en las tablas 2.1 y 2.2.

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

Cuadro 2.1: Tabla de la adición.

$\cdot$	0	1	x	$1+x$
0	0	0	0	0
1	0	1	x	$1+x$
x	0	x	$1+x$	1
$1+x$	0	$1+x$	1	x

Cuadro 2.2: Tabla de la multiplicación.

Por la estructura que presenta la tabla de la multiplicación, se puede ver que no es so lo un anillo. Dado que todo elemento diferente de cero tiene un inverso multiplicativo, se cumple que $\mathbb{F}_2[x]/\langle x^2+x+1\rangle$ es un campo, esto ocurre cuando el polinomio que cocienta el campo base es irreducible en él. En general, si R es un anillo e I es un ideal de R, se puede dar estructura de anillo al conjunto R/I de las clases laterales módulo I. En particular, tomando $R=\mathbb{F}[x]$ e $I=\langle x^n-1\rangle$, se define el anillo cociente $R_n=\mathbb{F}[x]/\langle x^n-1\rangle$ como sigue.

Definición 2.1. Sean n un entero positivo y $\mathbb{F}$ un cuerpo. La notación

$$\begin{aligned} R_n &= \mathbb{F}[x]/\langle x^n-1\rangle \\ &= \{r(x)+\langle x^n-1\rangle : r(x)\in\mathbb{F}_q[x]\}, \end{aligned}$$

hace referencia al anillo cociente (o anillo factor), que consta de las clases residuales de $\mathbb{F}[x]$ módulo x^n-1; en este se definen dos operaciones de la siguiente manera:

(i) $r(x)+\langle x^n-1\rangle + p(x)+\langle x^n-1\rangle = r(x)+p(x)+\langle x^n-1\rangle$.

(ii) $\big(r(x)+\langle x^n-1\rangle\big)\big(p(x)+\langle x^n-1\rangle\big) = r(x)p(x)+\langle x^n-1\rangle$.

Se puede probar que cada polinomio de grado menor o igual a $n-1$ en $\mathbb{F}[x]$ pertenece a una clase residual diferente, es por eso que se toman estos polinomios como representantes de las clases residuales en R_n. Adicionalmente, bajo la operación suma definida anteriormente y la multiplicación por escalar dada por

$$a(r(x)+\langle x^n-1\rangle) = ar(x)+\langle x^n-1\rangle,$$

para cada $a \in \mathbb{F}$, R_n es un espacio vectorial de dimensión n sobre $\mathbb{F}$.

Observe que aunque los elementos de R_n son de la forma $r(x) + \langle x^n - 1\rangle$, en lo restante de este trabajo simplemente se escribirá $r(x)$, esperando que no haya lugar a confusión dado que esta es la misma notación que se seguirá usando para los elementos de $\mathbb{F}[x]$.

Teorema 2.1. $R_n = \mathbb{F}_q[x]/\langle x^n - 1\rangle$ es un dominio de ideales principales.

Demostración. Sea I un ideal de R_n. Si $I = \{0\}$, es claro que $I = \langle 0\rangle$. Cuando I sea un ideal no nulo, considere $g(x)$ un polinomio de grado mínimo en I, i.e., si $h(x)$ es tal que $\text{grad}[h(x)] < \text{grad}[g(x)]$, entonces $h(x) \notin I$. Se demostrará que $I = \langle g(x)\rangle$. Ahora bien, sea $f(x) \in I$, por el algoritmo de la división se tiene que

$$f(x) = q(x)g(x) + r(x), \tag{2.2.1}$$

donde $r(x) = 0$ o $\text{grad}[r(x)] < \text{grad}[g(x)]$. Si se cumple la primera de esas posibilidades, entonces el teorema está demostrado. Asumiendo que $r(x) \neq 0$, entonces a partir de la ecuación 2.2.1 se sigue que

$$r(x) = q(x)g(x) - f(x).$$

De esta manera, dado que I es un ideal de R_n y $g(x) \in I$, se tiene que $q(x)g(x) \in I$, además, por hipótesis $f(x) \in I$, esto implica que $r(x) \in I$. Sin embargo, esto es una contradicción, puesto que en I no existen polinomios de grado menor al grado de $g(x)$. Con esto queda demostrado el teorema. □

Una última definición antes de abordar la temática principal de este capítulo y relacionada con la manera en que trabaja el software SAGE se presenta a continuación.

Definición 2.2 (mínimo común múltiplo)**.** El *mínimo común múltiplo*, denotado como

$$\text{mcm}(f_1(x), f_2(x))$$

entre dos polinomios no cero $f_1(x), f_2(x) \in \mathbb{F}_q[x]$ se define como el polinomio mónico de menor grado que es múltiplo tanto de $f_1(x)$ como de $f_2(x)$. Suponiendo que se tienen t polinomios no cero $f_1(x), f_2(x), \ldots, f_t(x) \in \mathbb{F}_q[x]$, el mínimo común múltiplo de $f_1(x), f_2(x), \ldots f_t(x)$ es el polinomio mónico de menor grado tal que es múltiplo de todos los f_i con $1 \leq t \leq t$, y se denota por

$$\text{mcm}(f_1(x), f_2(x), \ldots, f_t(x)).$$

Notación 2.1. Aunque se ha venido denotando las palabras en $\mathbb{F}_q^n$ como $c_1 \ldots c_n$, en este capítulo las mismas se indexarán desde 0 hasta $n-1$, es decir, $c_0 \ldots c_{n-1}$, lo que ayudará posteriormente en las operaciones y en la compresión del concepto de código cíclico que se presenta a continuación.

Definición 2.3 (Código cíclico)**.** Un código lineal $C \subset \mathbb{F}_q^n$ es cíclico si para cada palabra código $c_0c_1\cdots c_{n-2}c_{n-1} \in C$ se tiene que $c_{n-1}c_0c_1\cdots c_{n-2} \in C$.

De acuerdo a la anterior definición, un código lineal C es cíclico si es cerrado bajo el desplazamiento cíclico

$$c_0c_1\cdots c_{n-2}c_{n-1} \in C \longrightarrow c_{n-1}c_0c_1\cdots c_{n-2} \in C,$$

y como consecuencia, C es cerrado bajo todos los desplazamientos cíclicos, es decir,

$$c_0c_1\cdots c_{n-2}c_{n-1} \in C \longrightarrow c_k\cdots c_{n-1}c_0c_1\cdots c_{k-1} \in C,$$

para cualquier $1 \leq k \leq n-1$.

Ejemplo 2.2. Los siguientes son ejemplos de códigos cíclicos.

- $\{\mathbf{0}\}$,
- $C = \{\lambda \cdot \mathbf{1} \ : \lambda \in \mathbb{F}_q\}$,
- $\mathbb{F}_q^n$,
- El $[3, 2, 2]$-código lineal $\{000, 110, 101, 011\}$.

Ejemplo 2.3. El siguiente es un ejemplo de un código cíclico en el que se evidencia la principal característica de estos, la rotación.

$$\{0000000, 1011100, 0101110, 0010111, 1001011, 1100101, 1110010, 0111001\}.$$

Ejemplo 2.4. Observe que en la definición de código cíclico además de que este sea cerrado bajo el desplazamiento cíclico, también se exige que sea un código lineal. Es así que, por ejemplo, el código $C = \{110, 101, 011\}$ no es un código cíclico.

Si C es un código lineal sobre $\mathbb{F}_q^n$, en particular si es un código cíclico, se empleará con mucha frecuencia la representación de las palabras código en forma polinómica, es decir, a cada palabra código $c = c_0c_1\cdots c_{n-1}$, se le asocia una clase lateral de $R_n = \mathbb{F}_q[x]/\langle x^n - 1\rangle$ de la siguiente manera,

$$\begin{aligned} \phi : \mathbb{F}_q^n &\longrightarrow R_n \\ \phi(c_0c_1\cdots c_{n-1}) &= c_0 + c_1x + \cdots + c_{n-1}x^{n-1}, \end{aligned} \tag{2.2.2}$$

donde ϕ es una $\mathbb{F}_q$-transformación lineal de espacios vectoriales sobre $\mathbb{F}_q$. Efectivamente, si consideran $\alpha, \beta \in \mathbb{F}_q$ y $a, b \in \mathbb{F}_q^n$, donde $a = a_0a_1\ldots a_{n-1}$ y $b = b_0b_1\ldots b_{n-1}$, se tiene que

$$\begin{aligned} \phi(\alpha a + \beta b) &= \phi(\alpha a_0 + \beta b_0, \alpha a_1 + \beta b_1, \ldots, \alpha a_{n-1} + \beta b_{n-1}) \\ &= (\alpha a_0 + \beta b_0) + (\alpha a_1 + \beta b_1)x + \cdots + (\alpha a_{n-1} + \beta b_{n-1})x^{n-1} \\ &= \alpha a_0 + \alpha a_1x + \cdots + \alpha a_{n-1}x^{n-1} + \beta b_0 + \beta b_1x + \cdots + \beta b_{n-1}x^{n-1} \\ &= \alpha(a_0 + a_1x + \cdots + a_{n-1}x^{n-1}) + \beta(b_0 + b_1x + \cdots + b_{n-1}x^{n-1}) \\ &= \alpha\phi(a) + \beta\phi(b). \end{aligned}$$

Adicionalmente, dados $a = a_0a_1 \dots a_{n-1}$ y $b = b_0b_1 \dots b_{n-1}$ tales que $\phi(a) = \phi(b)$, se sigue que $a = b$, lo que prueba que ϕ es inyectiva. Además, se puede probar que los elementos de R_n son de la forma $c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$ con $c_i \in \mathbb{F}_q$. En efecto, sea $g(x) + \langle x^n - 1\rangle \in R_n$, entonces, por el algoritmo de la división se tiene que

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

Así, existen dos posibilidades $r(x) = 0$ o $\deg(r(x)) \leq \deg(g(x))$. Si se cumple la primera de ellas, entonces, $g(x) \equiv 0 (\text{mód} x^n - 1)$, lo cual indica que cualquier polinomio en R_n podrá ser expresado como una combinación lineal de polinomios cuyo grado no es mayor que $x^n - 1$. Por otra parte, cuando $\deg(r(x)) \leq \deg(g(x))$, resulta que

$$\begin{aligned} g(x) + \langle x^n - 1\rangle &= q(x)(x^n - 1) + r(x) + \langle x^n - 1\rangle \\ &\equiv r(x) + \langle x^n - 1\rangle \quad \text{en} \quad R_n. \end{aligned}$$

Es decir, $r(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$, resultado análogo a la primera de las posibilidades, lo que demuestra que ϕ es sobreyectiva, haciendo de ϕ una función biyectiva.

Ejemplo 2.5. Para el código cíclico $C = \{000, 110, 101, 011\}$, la asignación mediante ϕ es la siguiente

$$\phi(C) = \{0,\ 1 + x,\ 1 + x^2,\ x + x^2\}.$$

2.3. Polinomio generador y polinomio de control

Teorema 2.2. Sea ϕ la aplicación definida en la expresión (2.2.2). Entonces, un código C sobre $\mathbb{F}_q^n$ es un código cíclico, si y solo si, $\phi(C)$ es un ideal del anillo R_n.

Demostración. Sea ϕ la función definida en 2.2.2. Asumiendo que C es un código cíclico, se probará que $\phi(C)$ es un ideal de R_n.

Sean $p(x), q(x) \in \phi(C)$, dado que $\phi(C)$ es un subespacio vectorial de R_n, entonces $p(x) - q(x) \in \phi(C)$. Por otra parte, siendo

$$p(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1} = \phi(c_0c_1 \cdots c_{n-1}),$$

donde, $c_0c_1 \cdots c_{n-1} \in C$. Entonces, el polinomio

$$\begin{aligned} xp(x) = c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + c_{n-1}x^n &\equiv c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} + \langle x^n - 1\rangle \\ &= \phi(c_{n-1}c_0c_1 \cdots c_{n-2}), \end{aligned}$$

por tanto, es también un elemento de $\phi(C)$, dado que C es cíclico. Con un argumento análogo, se concluye que $x^2p(x)$ es también un elemento de $\phi(C)$. De manera inductiva se concluye que $x^ip(x)$

está en $\phi(C)$, para todo $i \geq 0$. Por lo tanto, para todo $q(x) = c'_o + c'_1 x + \cdots + c'_{n-1} x^{n-1} \in R_n$, el polinomio

$$q(x)p(x) = \sum_{i=0}^{n-1} c'_i (x^i p(x)),$$

es un elemento de $\phi(C)$. Así, $\phi(C)$ es un ideal de R_n.

Recíprocamente, si $\phi(C)$ es un ideal de R_n, entonces el polinomio nulo pertenece a $\phi(C)$, y por tanto el vector nulo pertenece a C. Además, para cualquier $\alpha, \beta \in \mathbb{F}_q$ y $a, b \in C$, por definición de ideal se tiene que $\alpha\phi(a) + \beta\phi(b) \in \phi(C)$ y dado que ϕ es un isomorfismo $\phi(\alpha a + \beta b) \in \phi(C)$, de donde, $\alpha a + \beta b$ es una palabra del código C. Luego, C es un código lineal. Ahora bien, si $c = c_0 c_1 \cdots c_{n-2} c_{n-1} \in C$, el polinomio

$$\phi(c) = c_0 + c_1 x + \cdots + c_{n-2} x^{n-2} + c_{n-1} x^{n-1},$$

es un elemento de $\phi(C)$. Como $\phi(C)$ es un ideal de R_n, el elemento

$$\begin{aligned} x\phi(c) &= c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1} + c_{n-1} x^n \\ &= c_{n-1} + c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1}, \end{aligned}$$

pertenece a $\phi(C)$, es decir, que $c_{n-1} c_0 c_1 \cdots c_{n-2}$ es una palabra del código C. De esta manera, se tendrá que C es cíclico. □

Notación 2.2. Dada la dualidad existente entre la representación de un código cíclico como un subconjunto de $\mathbb{F}_q^n$ y un ideal de R_n, en adelante, la utilización de las dos maneras de representación se hará de forma indistinta, dependiendo de qué sea lo más conveniente para algún caso determinado.

De esta manera, el resultado anterior establece una relación biunívoca entre códigos cíclicos de $\mathbb{F}_q^n$ e ideales de R_n; hecho que se consigna en el siguiente corolario.

Corolario 2.1. Tener un código cíclico es equivalente a tener un ideal de R_n.

2.3.1. Polinomio generador

En la sección anterior se mostró que estudiar códigos cíclicos es equivalente a estudiar ideales en R_n. Esto tiene una consecuencia mayor, dado que el anillo R_n es un anillo de ideales principales; esto es, cada ideal es generado por un elemento. Esta situación se traduce al lenguaje de los códigos cíclicos en el siguiente resultado.

Teorema 2.3. Existe un único polinomio mónico $g(x)$ de menor grado para cualquier ideal C de R_n. Además, este polinomio genera a C, es decir $C = \langle g(x) \rangle$. Además $g(x)$ divide a $x^n - 1$.

Demostración. Sea C un ideal de R_n. Supóngase que C tiene dos polinomios mónicos $g(x)$ y $k(x)$ de grado mínimo r. Entonces, se tiene que el polinomio no nulo $g(x)-k(x)$ pertenece a C. Dado que ambos polinomios son mónicos y del mismo grado, el grad$[g(x)-k(x)]$ es menor que r, contradiciendo de esta manera la hipótesis de que estos dos polinomios sean de grado mínimo en C. En seguida se probará que dicho polinomio genera a C, para lo que una vez mas se recurrirá el algoritmo de la división. 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

$$c(x) = q(x)g(x) + r(x),$$

donde $q(x), r(x) \in R_n$ y $r(x) = 0$ o grad$[r(x)] <$ grad$[g(x)] = r$. Así, $r(x) = c(x)-q(x)g(x) \in C$, que descarta la segunda posibilidad debido a la minimalidad del grado de $g(x)$. Por lo tanto, $r(x) = 0$ y así, $c(x) \in \langle g(x)\rangle$, o sea que $C \subseteq \langle g(x)\rangle$. Las dos contenencias mencionadas implican que $C = \langle g(x)\rangle$. De ahí que la última afirmación ahora es clara; por consiguiente, es posible establecer la siguiente expresión para algún $h(x) \in \mathbb{F}_q[x]$

$$h(x) = \frac{x^n - 1}{g(x)}. \tag{2.3.1}$$

□

La existencia encontrada en el resultado anterior permite establecer la siguiente definición.

Definición 2.4 (Polinomio generador)**.** El único polinomio mónico de menor grado de un ideal no cero I de R_n se denomina el *polinomio generador* de I. Para un código cíclico C sobre $\mathbb{F}_q^n$, el polinomio generador de $\phi(C)$ es llamado también el polinomio generador de C.

Observación 2.1. Un ideal (código cíclico) C puede ser generado por otros polinomios distintos del polinomio generador. En efecto, cualquier múltiplo escalar no nulo de él, genera al ideal, pero en este caso pierde la condición de ser mónico. En otros casos, el polinomio que genera al ideal puede no cumplir con la condición de ser minimal, como se observa en el ejemplo 2.6; por lo tanto, se adopta la notación $C = \langle\langle p(x)\rangle\rangle$ para denotar el hecho que C es el ideal generado por $p(x)$ y que $p(x)$ es el polinomio generador de C.

Ejemplo 2.6. Observe que para el código cíclico $C = \{000, 110, 101, 011\}$, se tiene que $\phi(C) = \{0, 1+x, 1+x^2, x+x^2\}$. De esta forma, un polinomio generador de este código es

$$\phi(C) = \langle 1 + x\rangle.$$

Así, los elementos de C se pueden expresar de la siguiente forma

$$0 = 0 \cdot (1+x) \qquad 1 + x = 1 \cdot (1+x)$$
$$x + x^2 = x \cdot (1+x) \qquad 1 + x^2 = x^2 \cdot (1+x).$$

Es de tener en cuenta que el código anterior también puede ser generado por el polinomio $1+x^2$, no obstante este no es un polinomio minimal en C.

Teorema 2.4. Sea $C = \langle\langle g(x)\rangle\rangle$ un ideal no cero de R_n, es decir, un código cíclico de longitud n, con $r = \text{grad}[g(x)]$.

1. Cualquier $c(x) \in C$ puede ser escrito de manera única como $c(x) = p(x)g(x)$ en $\mathbb{F}_q[x]$, donde $p(x) \in \mathbb{F}_q[x]$ es de grado menor que $n - r$.

2. La dimensión de C es $n - r$.

3. Si $g(x) = g_0 + g_1x + \cdots + g_rx^r$. Entonces, $g_0 \neq 0$ y C tiene matriz generadora

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ \vdots \\ x^{n-r-1}g(x) \end{pmatrix} = \begin{pmatrix} g_0 & g_1 & \cdots & & g_r & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & & g_r & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & & g_r & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & \cdots & & g_r \end{pmatrix},$$

   donde cada fila de G es un desplazamiento cíclico de la fila anterior. Obsérvese que un polinomio se está identificando con un vector.

Demostración.

1. Dado que $C = \langle\langle g(x)\rangle\rangle$, entonces

$$C = \langle\langle g(x)\rangle\rangle = \{f(x)g(x) : f(x) \in R_n\},$$

   con la reducción módulo $x^n - 1$, se demostrará que es suficiente con restringir $f(x)$ a los polinomios de grado menor que $n - r$. Dado que $g(x)$ divide a $x^n - 1$, se sigue que $x^n - 1 = p(x)g(x)$, para algún polinomio $p(x) \in \mathbb{F}_q[x]$ de grado $n - r$. Dividiendo $f(x)$ por $p(x)$ se tiene

$$f(x) = q(x)p(x) + r(x), \tag{2.3.2}$$

   con $q(x), r(x) \in \mathbb{F}_q[x]$ y $r(x) = 0$ o $\text{grad}[r(x)] < n - r$. Si ocurre lo primero, entonces la afirmación está demostrada, en caso de que $\text{grad}[r(x)] < n - r$, al multiplicar la ecuación (2.3.2) por $g(x)$ se sigue que

$$f(x)g(x) = q(x)p(x)g(x) + r(x)g(x) = q(x)(x^n - 1) + r(x)g(x),$$

   de donde $f(x)g(x) = r(x)g(x)$ en R_n, es decir, cualquier elemento en C puede ser expresado como el producto entre un polinomio $r(x)$ de grado menor que $n - r$ y el polinomio generador $g(x)$, que es lo que se quería demostrar.

2. Considérese aquí el ideal (código cíclico) $I = \langle g(x)\rangle = \{f(x)g(x) : f(x) \in \mathbb{F}_q[x]\}$; por el resultado anterior se tiene que $\text{grad}[f(x)] < n-r$, es decir, $f(x) = c_0+c_1x+\cdots+c_{n-r-1}x^{n-r-1}$, de esta manera, se tiene que $f(x)g(x) = c_0g(x) + c_1xg(x) + \cdots + c_{n-r-1}x^{n-r-1}g(x)$, en otras palabras, todo elemento $t(x) \in I$ se puede expresar como combinación lineal de términos de la forma $x^ig(x)$. Al hacer un conteo sencillo, se obtiene que el número de dichos términos es $n - r$, ahora hay que comprobar que este conjunto es linealmente independiente. Entonces, se supondrá que existen $c_0, c_1, \ldots, c_{n-r-1}$ escalares en $\mathbb{F}_q$, tales que

$$c_0g(x) + c_1xg(x) + \cdots + c_{n-r-1}x^{n-r-1}g(x) \equiv 0 \text{ mód } x^n - 1.$$

Luego

$$g(x)(c_0 + c_1x + \cdots + c_{n-r-1}x^{n-r-1}) \equiv 0 \text{ mód } x^n - 1.$$

Por tanto, para algún $d(x) \in \mathbb{F}[x]$, se satisface que

$$g(x)(c_0 + c_1x + \cdots + c_{n-r-1}x^{n-r-1}) = d(x) \cdot (x^n - 1).$$

Así, por la ecuación (2.3.1), se cumple que

$$c_0 + c_1x + \cdots + c_{n-r-1}x^{n-r-1} = d(x) \cdot h(x).$$

Como el grado de $h(x)$ es $n - r$, se sigue que $c_0 + c_1x + \cdots + c_{n-r-1}x^{n-r-1} = 0$, lo que implica $c_0 = c_1 = \cdots = c_{n-r-1} = 0$. De esta manera queda demostrado que el conjunto $\{g(x), xg(x), \ldots, x^{n-r-1}g(x)\}$, constituye una base para C y por tanto que la dimensión de C es $n - r$.

3. Sea $g(x) = g_0 + g_1x + \cdots + g_rx^r$ el polinomio generador de C. Si $g_0 = 0$, entonces, es posible factorizar x y expresar $g(x)$ de la siguiente manera, $g(x) = xf(x)$ y así $\text{grad}[f(x)] < r$. Luego,

$$\hat{g}(x) = 1 \cdot f(x) \equiv x^nf(x) = x^{n-1}xf(x) = x^{n-1}g(x).$$

de donde resulta que $f(x) \in C$, lo que establece una contradicción, ya que por el Teorema 2.3, no puede haber en C un polinomio de menor grado que $g(x)$. Luego, $g_0 \neq 0$. El hecho de que la matriz G es una matriz generadora para C, resulta de que el conjunto $\{g(x), xg(x), \ldots, x^{n-r-1}g(x)\}$ es linealmente independiente y genera a C, según el resultado en el ítem anterior y la Definición 1.25. □

Teorema 2.5. Un polinomio mónico $p(x)$ en R_n es el polinomio generador de algún código cíclico en $\mathbb{F}_q^n$, si y solo si, $p(x)$ divide a $x^n - 1$.

Demostración. Por la parte 1 del Teorema 2.4 se tiene una de las implicaciones. Para la segunda implicación, se consideran $p(x)$ que divide a $x^n - 1$, C como el ideal $\langle p(x)\rangle$ y $g(x)$ el polinomio

generador de C, entonces, si $p(x) \neq g(x)$, dado que $p(x)$ y $g(x)$ son ambos mónicos, por el Teorema 2.3, debe cumplirse que grad$[g(x)] <$ grad$[p(x)]$. Además, por hipótesis,

$$x^n - 1 = p(x)f(x), \tag{2.3.3}$$

para algún polinomio $f(x) \in \mathbb{F}_q[x]$. Adicionalmente, puesto que $g(x) \in \langle p(x) \rangle$, resulta que

$$g(x) = k(x)p(x),$$

para algún $k(x) \in R_n$. Si se multiplica la anterior igualdad por $f(x)$ y utilizando la igualdad (2.3.3), se tiene que

$$g(x)f(x) \equiv k(x)p(x)f(x) \equiv k(x)(x^n - 1) \equiv 0 + \langle x^n - 1 \rangle.$$

Sin embargo, grad$[g(x)f(x)] <$ grad$[p(x)f(x)] = n$. Por lo tanto, $g(x)f(x) = 0$ en $\mathbb{F}_q[x]$, lo cual implica que $f(x) = 0$, dado que $\mathbb{F}_q[x]$ es un dominio entero. Esto resulta en una contradicción a la expresión (2.3.3). Luego $p(x) = g(x)$. □

Mediante el uso del software SAGE es posible construir códigos cíclicos, teniendo como parámetros al anillo con el que se va a trabajar y al polinomio que lo cocienta, de la siguiente manera.

Ejemplo 2.7. La instrucción

```
sage.coding.code_constructions.CyclicCode(n, g, ignore=True)
```

le indica al programa que si g es un polinomio sobre $\mathbb{F}_q$ que divide al polinomio $x^n - 1$, construya el código "cíclico generado por g". Es decir, genera el código asociado al ideal principal $I = \langle g(x) \rangle$, en el anillo $R_n = \mathbb{F}_q[x]/\langle x^n - 1 \rangle$ en la forma usual.

La opción "ignore" indica que si se ignoran las condiciones, (a) que q y la longitud del código sean primos relativos, como se mencionará en el Corolario 2.2, y (b) que g debe dividir a $x^n - 1$; es decir, si ignore=True, en lugar de retornar un error, el código generado es el mcm$(x^n - 1, g)$, según la Definición 2.2.

Ejemplo 2.8. En el siguiente ejemplo, se especifica la indeterminada (x en este caso) en la que se define el anillo de polinomios y g como el polinomio que generará el código.

```
P.<x> = PolynomialRing(GF(3),"x")
g = x-1
C = codes.CyclicCodeFromGeneratingPolynomial(4,g); C
sage: Linear code of length 4, dimension 3 over Finite Field of size 3
```

En la última línea de código, el sofware SAGE presenta en pantalla el tipo de código cíclico creado.

Ejemplo 2.9. En el siguiente ejemplo, se indica la indeterminada "a"que corresponde a la raíz de un polinomio irreducible en $\mathbb{F}_2$ que genera al campo finito $\mathbb{F}_4$, mientras que "x" es la indeterminada en que se define dicho polinomio.

```
P.<x> = PolynomialRing(GF(4,"a"),"x")
g = x^3+1
C = codes.CyclicCodeFromGeneratingPolynomial(9,g); C
sage:
Linear code of length 9, dimension 6 over Finite Field in a of size 2^2
```

Finalmente, lo que presenta el sofware en pantalla, es el código generado a partir de los parámetros introducidos en el proceso.

Ejemplo 2.10. Según la cuarta parte del Teorema 2.4, un código cíclico también tiene una matriz que lo genera, de esta manera, en SAGE es posible determinar la matriz generadora de un código construido a partir del campo finito y el polinomio involucrado, según se indica con los siguientes comandos.

```
P.<x> = PolynomialRing(GF(2),"x")
g = x^3+x+1
C = codes.CyclicCodeFromGeneratingPolynomial(7,g); C
sage: Linear code of length 7, dimension 4 over Finite Field of size 2
C.generator_matrix()
sage:
[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]
```

Lema 2.1. 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. Sea π la función del conjunto D_n que envía a todos los divisores mónicos de $x^n - 1$ en el conjunto C_n de todos los códigos cíclicos en R_n definida como

$$\pi : D_n \longrightarrow C_n$$
$$\pi(g(x)) \longmapsto \langle\langle g(x)\rangle\rangle.$$

En otras palabras, π envía cada divisor mónico $g(x)$ de $x^n - 1$ al código cíclico $\langle\langle g(x)\rangle\rangle$. Así por los Teoremas 2.3 y 2.4, se tiene que la función π es biyectiva, dado que cada divisor mónico de $x^n - 1$ genera un único código cíclico de R_n, además, todo código cíclico de R_n debe ser generado por un divisor mónico de $x^n - 1$. □

Ejemplo 2.11. Los polinomios 1 y $x^n - 1$ corresponden a los códigos cíclicos $\mathbb{F}_q^n$ y $\{\mathbf{0}\}$, respectivamente.

En ese orden de ideas, el lema anterior revela la importancia que tiene la factorización del polinomio $x^n - 1$ sobre $\mathbb{F}_q^n$ cuando se investiga sobre los polinomios generadores de los códigos cíclicos que se pueden construir en R_n. En otros términos, esta factorización permite determinar el número de códigos cíclicos en R_n. El siguiente resultado va encaminado en esa misma dirección.

Teorema 2.6. Sea $x^n - 1 \in \mathbb{F}_q[x]$ cuya factorización es la siguiente

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

donde $p_1(x), p_2(x), \ldots, p_i(x)$ son polinomios 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. Según el Lema 2.1, determinar el número de códigos cíclicos de longitud n sobre $\mathbb{F}_q$ requiere contar el número de divisores mónicos de $x^n - 1$. De esta manera, si

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

donde los $p_1(x), p_2(x), \ldots, p_r(x)$ son polinomios mónicos irreducibles diferentes y $e_i \geq 1$ para todo $i \in \{1, 2, \ldots, r\}$, entonces, cada polinomio $p_i(x)$ tiene $e_i + 1$ divisores mónicos que son, $1, p_i, p_i^2, \ldots, p_i^{e_i}$, los cuales evidentemente dividen a $x^n - 1$ también. Así, si se consideran los r polinomios en la factorización de $x^n - 1$ con sus respectivos grados, el número de divisores mónicos es

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

□

Note que si n y q no son primos relativos, entonces, n se puede escribir como $n = mp^k$, donde $(m, q) = 1$ y p es la característica de $\mathbb{F}_q$. 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.

Corolario 2.2. En el caso que n y q sean 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 irreducíbles diferentes. En consecuencia, el número de códigos cíclicos de longitud n sobre $\mathbb{F}_q$ en este caso es 2^r.

Ejemplo 2.12. Para encontrar todos los códigos cíclicos de longitud 8, sobre $\mathbb{F}_3$ se factoriza el polinomio $x^8 - 1 \in \mathbb{F}_3[x]$ de la siguiente manera

$$x^8 - 1 = (x+2)(x+1)(x^2+1)(x^2+x+2)(x^2+2x+2).$$

Es evidente que los factores son irreducibles, puesto que ninguno de ellos tiene raíces en $\mathbb{F}_3$. Así, por el Corolario 2.2, se sabe que existen $2^5 = 32$ códigos cíclicos de longitud 8 sobre $\mathbb{F}_3$. Además, es posible construir seis $[8,4]$-códigos cíclicos sobre $\mathbb{F}_3$, ya que esas son todas las posibles combinaciones que podemos hacer de los factores de $x^8 - 1$, tales que el grado del producto de dichos polinomios sea cuatro. En este caso, los polinomios generadores de cada uno de estos códigos son los siguientes:

- $g_1(x) = (x+2)(x+1)(x^2+1)$,
- $g_2(x) = (x+2)(x+1)(x^2+x+2)$,
- $g_3(x) = (x+2)(x+1)(x^2+2x+2)$,
- $g_4(x) = (x^2+1)(x^2+x+2)$,
- $g_5(x) = (x^2+1)(x^2+2x+2)$,
- $g_6(x) = (x^2+x+2)(x^2+2x+2)$.

Por último, un $[8,2]$-código cíclico sobre $\mathbb{F}_3$ es $\langle\langle(x^2+1)(x^2+x+2)(x^2+2x+2)\rangle\rangle$, es decir:

$$\begin{aligned}\langle(x^2+1)(x^2+x+2)(x^2+2x+2)\rangle &= \{00000000, 10101010, 01010101, 20202020, 02020202,\\ &\quad 21212121, 12121212, 11111111, 22222222\}.\end{aligned}$$

2.3.2. Polinomio de Control

Definición 2.5 (Polinomio de control)**.** Dado que un código cíclico es un código lineal, a continuación se presenta el procedimiento para construir el polinomio de control de un código cíclico, el cual permitirá construir la matriz de control de paridad y posteriormente servir de ayuda para ciertos resultados sobre el proceso de decodificación.

Sea $g(x)$ el polinomio generador de un $[n, n-r]-$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 $n-r$. El polinomio $h(x)$ se denomina el polinomio de control de C.

Definición 2.6 (Polinomio recíproco)**.** Sea $h(x) = \sum_{i=0}^{k} a_i x^i$ un polinomio de grado k sobre $\mathbb{F}_q[x]$, se define el polinomio recíproco $h_R(x)$ por

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

Observación 2.2. Si $h(x)$ divide a $x^n - 1$, entonces, $h_R(x)$ también lo divide.

Demostración. Observe que el polinomio $h_R(x)$, según la Definición 2.6, se puede expresar como $x^k h(x^{-1})$, así, $h(x^{-1})g(x^{-1}) = (x^{-1})^n - 1 = x^{-n} - 1$, por consiguiente, $x^k h(x^{-1}) x^{n-k} g(x^{-1}) = x^n(x^{-n} - 1) = (1 - x^n)$. Y de esta manera, $h_R(x)$ es un factor de $x^n - 1$ o lo que es lo mismo, $h_R(x)$ divide a $x^n - 1$. □

Lema 2.2. Si C es un código cíclico, entonces el código dual $C^\perp$ es también un código cíclico.

Demostración. Sean C un código cíclico y $\mathbf{x} = x_0 x_1 \cdots x_{n-1} \in C^\perp$. Para cualquier $\mathbf{c} = c_0 c_1 \ldots c_{n-1} \in C$ se tiene que $\mathbf{c}' = c_1 c_2 \ldots c_{n-1} c_0$ también está en C, en consecuencia se tiene que

$$\begin{aligned} 0 = \mathbf{c}' \cdot \mathbf{x} &= c_1 x_0 + c_2 x_1 + \cdots + c_{n-1} x_{n-2} + c_0 x_{n-1} \\ &= c_0 x_{n-1} + c_1 x_0 + c_2 x_1 + \cdots + c_{n-1} x_{n-2}. \end{aligned}$$

De esta manera, si se define $\mathbf{x}' = x_{n-1} x_0 x_1 \cdots x_{n-2}$, se tiene $\mathbf{c} \cdot \mathbf{x}' = 0$. Por lo tanto, $\mathbf{x}' \in C^\perp$, así $C^\perp$ es también un código cíclico. □

Teorema 2.7. Sea $h(x) = h_0 + h_1 x + \cdots + h_{n-r} x^{n-r}$ el polinomio de control para un código cíclico C en R_n, entonces

1. El código C puede ser descrito por

$$C = \{p(x) \in R_n : p(x)h(x) \equiv 0 \text{ mód } (x^n - 1)\}.$$

2. $h_0^{-1} h_R(x)$ es el polinomio generador de $C^\perp$, donde h_0 es el término constante de $h(x)$. Además una matriz de control de paridad para C es

$$H = \begin{pmatrix} h_R(x) \\ x h_R(x) \\ x^2 h_R(x) \\ \vdots \\ x^{r-1} h_R(x) \end{pmatrix} = \begin{pmatrix} h_{n-r} & \cdots & & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_{n-r} & \cdots & & h_0 & 0 & \cdots & 0 \\ 0 & 0 & h_{n-r} & \ldots & & h_0 & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & & \ddots & 0 \\ 0 & 0 & 0 & 0 & h_{n-r} & \cdots & & h_0 \end{pmatrix}.$$

   Obsérvese que la dimensión del código $C^\perp$ es r, dado que $\text{grad}[h_R(x)] = n - r$.

Demostración.

1. Sea $g(x)$ el polinomio generador de C. Si $p(x) \in C$, entonces, $p(x) = k(x)g(x)$ para algún $k(x) \in R_n$. Por lo tanto,

$$p(x)h(x) = k(x)g(x)h(x) = k(x)(x^n - 1) \equiv 0 \text{ mód } (x^n - 1).$$

Por otra parte, si $p(x) \in R_n$ y $p(x)h(x) \equiv 0$ mód $(x^n - 1)$, entonces, por el algoritmo de la división se tiene que

$$p(x) = q(x)g(x) + r(x), \tag{2.3.4}$$

donde $r(x) = 0$ o grad$[r(x)] < r$. En cualquiera de estos casos, si se multiplica la ecuación 2.3.4 por $h(x)$ se sigue que

$$\begin{aligned} p(x)h(x) &= q(x)g(x)h(x) + r(x)h(x) \\ 0 &= q(x)(x^n - 1) + r(x)h(x), \end{aligned}$$

de donde, $r(x)h(x) \equiv 0$. Sin embargo, grad$[r(x)h(x)] < r + (n - r) = n$. Luego, $r(x)h(x) = 0$, en consecuencia $r(x) = 0$. De esta manera, $p(x) = q(x)g(x) \in C$.

2. Sean $g(x) = \sum_{i=0}^{n-1} g_i x^i$ el polinomio generador de algún código cíclico sobre R_n y $h(x) = \sum_{i=0}^{n-1} h_i x^i$ su polinomio de control, donde el grad$[h(x)] = n - r$. Entonces, considerando el producto

$$\begin{aligned} 0 &\equiv g(x)h(x) \\ &\equiv (g_0h_0 + g_1h_{n-1} + \cdots + g_{n-1}h_1) + (g_0h_1 + g_1h_0 + \cdots + g_{n-1}h_2)x \\ &\quad (g_0h_2 + g_1h_1 + \cdots + g_{n-1}h_3)x^2 + \cdots + (g_0h_{n-1} + g_1h_{n-2} \\ &\quad + \cdots + g_{n-1}h_0)x^{n-1} (\text{mód} x^n - 1). \end{aligned}$$

Así, se tendrá que los coeficientes de cada potencia de x en la última línea de la anterior expresión deben ser cero. Si se consideran los coeficientes de cada potencia de x, se observa que $g_i \cdot h_{n-1}h_{n-2}\cdots h_1h_0 = 0$, para todo $i = 0, 1, \ldots, n-1$, donde g_i es la secuencia obtenida de $g_0g_1\cdots g_{n-1}$ mediante un desplazamiento cíclico de i posiciones. Por lo tanto, $h_{n-1}h_{n-r-1}\cdots h_1h_0h_0$ es una palabra del código $C^\perp$, dado que por el Teorema 2.4, el conjunto $\{g_0, g_1, \ldots, g_{n-1}\}$ generara a C. De esta manera, mediante un desplazamiento cíclico de $n - r + 1$ posiciones de la secuencia $h_{n-1}h_{n-2}\cdots h_1h_0$ se obtiene la secuencia correspondiente a $h_R(x)$, es decir, $h_{n-r}h_{n-r-1}\cdots h_1h_0h_{n-1}\cdots h_{n-r+1}$. Lo anterior implica que $h_R(x)$ es una palabra código de $C^\perp$, dado que por el Lema 2.2, $C^\perp$ es también un código cíclico. Como grad$[h_R(x)] =$ grad$[h(x)] = n-r$, el conjunto $\{h_R(x), xh_R(x), \ldots, x^{r-1}h_R(x)\}$ es una base para $C^\perp$. Por lo tanto, el polinomio mónico $h_0^{-1}h_R(x)$ es el polinomio generador del código $C^\perp$.

Por otra parte, dado que $h_0^{-1}h_R(x)$ es el polinomio generador de $C^{\perp}$, por el Teorema 2.4, se sigue que la matriz H es una matriz generadora para $C^{\perp}$, es decir, una matriz de control para C.

□

Ejemplo 2.13. Al trabajar con el software SAGE es posible también determinar el código cíclico con el que se está trabajando a partir del polinomio de control, con los siguientes comandos.

```
P.<x> = PolynomialRing(GF(3),"x")
C = codes.CyclicCodeFromCheckPolynomial(4,x + 1); C
sage: Linear code of length 4, dimension 1 over Finite Field of size 3
```

Ejemplo 2.14. De forma análoga a como se trabajó con el polinomio generador, con el sofware SAGE es posible hallar la matriz generadora de un código determinado por su polinomio de control, de la siguiente manera.

```
   C = codes.CyclicCodeFromCheckPolynomial(4,x^3 + x^2 + x + 1); C
sage: Linear code of length 4, dimension 3 over Finite Field of size 3
C.generator_matrix()
sage:
[2 1 0 0]
[0 2 1 0]
[0 0 2 1]
```

2.4. Codificación con códigos cíclicos

Si $C = \langle\langle g(x)\rangle\rangle$ es un $[n, n-r]$-código cíclico, con $\text{grad}[g(x)] = r$, C puede codificar mensajes q-arios de longitud $n-r$ y requiere r símbolos de redundancia. Para esta clase de códigos, existen dos tipos de codificación que son: la *codificación no sistemática* y la *codificación sistemática*.

2.4.1. Codificación no sistemática

De forma análoga al trabajo realizado con los códigos lineales, un mensaje $m \in \mathbb{F}_q^{n-r}$, es decir, cuando se encuentra en su forma vectorial, se codifica como la palabra código $c = mG \in \mathbb{F}_q^n$, donde G es la matriz generadora para C, la cual se mencionó en la Observación 1.3.

2.4.2. Codificación sistemática

Apelando al carácter polinómico de los códigos cíclicos, si se quiere emplear la codificación sistemática con ellos, es necesario poner las cadenas mensaje en términos polinomiales, pero de una forma un tanto diferente a como se describió al inicio de este capítulo y que se denotó como la función

ϕ; dicho proceso se describe a continuación. Dado un mensaje fuente $a = a_0a_1 \cdots a_{n-r-1} \in \mathbb{F}_q^{n-r}$, se forma el polinomio mensaje

$$\overline{a}(x) = a_0x^{n-1} + a_1x^{n-2} + \cdots + a_{n-r-1}x^r. \tag{2.4.1}$$

De esta manera, se ha formado el polinomio $\overline{a}(x)$, el cual no tiene términos de grado menor que r. En seguida se divide a $\overline{a}(x)$ entre $g(x)$,

$$\overline{a}(x) = q(x)g(x) + s(x) \text{ con } s(x) = 0 \text{ o } \operatorname{grad}[s(x)] < r \tag{2.4.2}$$

y así, la codificación de $\overline{a}$ es

$$c(x) = \overline{a}(x) - s(x) = q(x)g(x). \tag{2.4.3}$$

Dado que $\overline{a}(x)$ y $s(x)$ no tienen términos del mismo grado, la codificación es sistemática, puesto que si se leen los términos de un polinomio del código desde el de mayor al de menor grado, se tendrá que los símbolos de información estarán ubicados en las primeras $n - r$ coordenadas, mientras que las r restantes son los símbolos de redundancia.

Ejemplo 2.15. Sea C el $[8, 2]$-código sobre $\mathbb{F}_3$, del Ejemplo 2.12, generado por $g(x) = (x^2+1)(x^2+x+2)(x^2+2x+2) = x^6+x^4+x^2+1$. Este código es capaz de codificar 9 mensajes de longitud 2, es decir, codifica el conjunto de información $\mathbb{F}_3^2 = \{00, 10, 01, 20, 02, 12, 21, 11, 22\}$. De esta manera, la información estará almacenada en las primeras $n - r = 8 - 6 = 2$ coordenadas, mientras que las 6 restantes son los símbolos de redundancia.

A continuación se presentan los polinomios mensaje con sus respectivos cocientes y residuos, obtenidos a partir del algoritmo de la división.

$00 = \overline{a}_1(x) = 0 = (x^6 + x^4 + x^2 + 1)(0) + 0,$
$10 = \overline{a}_2(x) = x^7 = (x^6 + x^4 + x^2 + 1)(x) + (2x^5 + 2x^3 + 2x),$
$01 = \overline{a}_3(x) = x^6 = (x^6 + x^4 + x^2 + 1)(1) + (2x^4 + 2x^2 + 2),$
$20 = \overline{a}_4(x) = 2x^7 = (x^6 + x^4 + x^2 + 1)(2x) + (x^5 + x^3 + x),$
$02 = \overline{a}_5(x) = 2x^6 = (x^6 + x^4 + x^2 + 1)(2) + (x^4 + x^2 + 1),$
$12 = \overline{a}_6(x) = x^7 + 2x^6 = (x^6 + x^4 + x^2 + 1)(x + 2) + (2x^5 + x^4 + 2x^3 + x^2 + 2x + 1),$
$21 = \overline{a}_7(x) = 2x^7 + x^6 = (x^6 + x^4 + x^2 + 1)(2x + 1) + (x^5 + 2x^4 + x^3 + 2x^2 + x + 2),$
$11 = \overline{a}_8(x) = x^7 + x^6 = (x^6 + x^4 + x^2 + 1)(x + 1) + (2x^5 + 2x^4 + 2x^3 + 2x^2 + 2x + 2),$
$22 = \overline{a}_9(x) = 2x^7 + 2x^6 = (x^6 + x^4 + x^2 + 1)(2x + 2) + (x^5 + x^4 + x^3 + x^2 + x + 1).$

Así, la codificación final $c_i(x) = \overline{a}_i - r_i(x)$ para cada $\overline{a}_i$ es la siguiente

$00 = \overline{a}_1(x) \to c_1(x) = 0,$
$10 = \overline{a}_2(x) \to c_2(x) = x^7 + x^5 + x^3 + x,$

$01 = \overline{a}_3(x) \to c_3(x) = x^6 + x^4 + x^2 + 1,$
$20 = \overline{a}_4(x) \to c_4(x) = 2x^7 + 2x^5 + 2x^3 + 2x,$
$02 = \overline{a}_5(x) \to c_5(x) = 2x^6 + 2x^4 + 2x^2 + 2,$
$12 = \overline{a}_6(x) \to c_6(x) = x^7 + 2x^6 + x^5 + 2x^4 + x^3 + 2x^2 + x + 2,$
$21 = \overline{a}_7(x) \to c_7(x) = 2x^7 + x^6 + 2x^5 + x^4 + 2x^3 + x^2 + 2x + 1,$
$11 = \overline{a}_8(x) \to c_8(x) = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1,$
$22 = \overline{a}_9(x) \to c_9(x) = 2x^7 + 2x^6 + 2x^5 + 2x^4 + 2x^3 + 2x^2 + 2x + 2.$

De lo anterior se sigue que la codificación de la información se realiza de la siguiente manera

$00 \to c_1(x) = 00000000$	$20 \to c_4(x) = 20202020$	$21 \to c_7(x) = 21212121$
$10 \to c_2(x) = 10101010$	$02 \to c_5(x) = 02020202$	$11 \to c_8(x) = 11111111$
$01 \to c_3(x) = 01010101$	$12 \to c_6(x) = 12121212$	$22 \to c_9(x) = 22222222$

Es claro que el trabajo de codificación realizado en el ejemplo anterior es casi imposible de realizar a mano cuando el tamaño del conjunto mensaje aumenta, dado el esfuerzo y el tiempo que esto puede conllevar. Con el fin de presentar una herramienta informática que además de facilitar la comprensión de este proceso, permita llevar a cabo de forma eficiente este trabajo, a continuación se describe la manera de realizar dicha actividad con ayuda del software SAGE, no sin antes recordar varios de los aspectos relacionados con los códigos lineales y los códigos cíclicos necesarios para iniciar con el proceso de codificación.

2.5. Codificación con códigos cíclicos usando SAGE

Según varios de los resultados obtenidos hasta el momento, en principio, la codificación en cadenas de longitud n de un conjunto de información M con t elementos, requiere que la longitud de cada palabra mensaje sea $n - r$, siendo $\mathbb{F}_q$ el alfabeto considerado para formar cada palabra, entonces, atendiendo a la parte 1 del Teorema 2.4, es necesario que a partir de la factorización del polinomio $x^n - 1$ en polinomios irreducibles sobre $\mathbb{F}_q[x]$, se considere el/los factores cuyo grado o el grado del producto de ellos sea r. También es para tener en cuenta, según la parte 1 del Teorema 1.14, que el número de elementos en un código lineal se puede determinar por $t = q^k$, este número representa la cantidad de palabras con las que se cuenta para enviar algún mensaje determinado; adicionalmente, la parte 3 del Teorema 2.4 indica que la dimensión del código k cumple con la igualdad $k = n - r$, lo que en últimas revela cual es la longitud de las palabras en el conjunto de información M.

2.5.1. Codificación no sistemática

En seguida se presenta la manera de emplear SAGE y los comandos que ya han sido implementados en él para realizar la codificación de una palabra en la forma *no sistemática*.

Entonces, suponiendo que la palabra que se va a codificar es $v = 21021$ de $\mathbb{F}_3^5$ que hace parte de un conjunto de información que contiene $3^5 = 243$ palabras mensaje, y que se desea realizar la codificación con un código cíclico de longitud 10, es decir, $n = 10$. Entonces, lo que se necesita, según el párrafo introductorio, es determinar la descomposición en factores primos del polinomio $x^{10} - 1$ sobre $\mathbb{F}_3[x]$, para ello se digita lo siguiente en SAGE.

```
  P.<x> = PolynomialRing(GF(3))
  t=x^10-1
G = t.factor(); G
sage:
(x + 1) * (x + 2) * (x^4 + x^3 + x^2 + x + 1) * (x^4 + 2*x^3 + x^2 + 2*x + 1)
```

Con esto se logra determinar los posibles polinomios generadores de los códigos cíclicos de longitud 10 en $\mathbb{F}_3[x]$. Ahora bien, como lo que se necesita es codificar una palabra de longitud 5, el polinomio generador debe ser de grado 5, se puede ver que hay varias posibilidades para que eso ocurra, en este ejemplo se tomará $g(x) = (x+1)(x^4+x^3+x^2+x+1) = x^5+2x^4+2x^3+2x^2+2x+1$. Ya con esto, lo siguiente es generar el código cíclico a partir de $g(x)$, introducir la palabra a codificar y ordenar al software que haga la codificación, no sin antes indicarle que considere a v como un vector y no como una lista, que es lo que inicialmente asume. Esto se logra como se muestra a continuación.

```
g=x^5 + 2*x^4 + 2*x^3 + 2*x^2 + 2*x + 1
C=codes.CyclicCodeFromGeneratingPolynomial(10,g)
v=[2,1,0,2,1]
h=vector((v))
C.encode(h)
sage:
  (2, 2, 0, 2, 2, 1, 1, 0, 1, 1)
```

Finalmente, lo que el software presenta es el resultado análogo a tomar el polinomio asignado al vector v por la función ϕ descrita en la expresión 2.2.2, y multiplicarlo por el polinomio generador $g(x)$. Así concluye la descripción de la codificación no sistemática en SAGE.

2.5.2. Codificación sistemática

En vista de que SAGE no tiene incorporado en su configuración interna un comando para realizar la codificación sistemática con códigos cíclicos, o al menos, no en la manera como se ha descrito en este texto, a continuación se muestra paso a paso la manera en que se ha ideado la manera de llevar a cabo ese trabajo ayudados por este software.

Supongamos que se desea enviar la palabra $s = 21201$ en $\mathbb{F}_3^5$, Si el objetivo es codificarla empleando un código de longitud 8, según lo realizado en el ejemplo 2.12, se empleará como polinomio generador al polinomio $g(x) = (x+2)(x^2+x+2) = x^3+x+1$. Como se mencionó, la dimensión del código cíclico generado es 5. Entonces los pasos a seguir son los siguientes.

1. De la misma manera como se trabajó en el ejemplo 2.10, lo primero es construir el anillo de polinomios con el que se va a trabajar, en este caso, $\mathbb{F}_3[x]$, designar el polinomio generador del código deseado, $g(x) = x^3+x+1$, y en seguida generar el código cíclico C con el cual se codificará la palabra mensaje.

```
P.<x> = PolynomialRing(GF(3),"x")
g= x^3 + x + 1
R=codes.CyclicCodeFromGeneratingPolynomial(8,g); R
sage:
Linear code of length 8, dimension 5 over Finite Field of size 3
```

2. Introducir la palabra mensaje que se va a codificar, digitada en forma de una lista en SAGE

```
s=[1,2,1,2,0]; s
sage:
[1, 2, 1, 2, 0]
```

3. Con el comando que se indica aquí se construye el polinomio imagen $f(x)$, según la asignación determinada por la función ϕ, descrita en la expresión 2.2.2.

```
 f=P(s); f
sage:
2*x^3 + x^2 + 2*x + 1
```

4. Dado que la codificación es sistemática, siguiendo lo descrito en la ecuación 2.4.1, la expresión polinómica de la palabra mensaje requiere ser puesta en una forma especial, para ello se procede así en el software

```
  f1=f.reverse(degree=7);f1
sage:
x^7 + 2*x^6 + x^5 + 2*x^4
```

donde "degree = 7" indica el grado desde el cual se va a empezar a hacer la asignación invertida, dicho grado siempre corresponde a $n-1$.

5. En seguida, lo que se necesita es calcular el residuo de dividir el polinomio obtenido en el paso anterior f entre el polinomio generador $g(x)$, para ello se digita

```
  c=f1.quo_rem(g);c
sage:
(x^4 + 2*x^3 + 2*x + 1, x^2 + 2)
```

lo que se obtiene aquí es un vector c de dos componentes, la primera de ellas corresponde al cociente y la segunda es el residuo buscado.

6. En este paso, se construye el polinomio h correspondiente a la palabra codificada final.

```
  h=f1-c[1];h
  sage:
x^7 + 2*x^6 + x^5 + 2*x^4 + 2*x^2 + 1
```

$c[1]$ indica que se está trabajando con la segunda componente del vector c. Y con esto concluye el procedimiento.

2.6. Decodificación con códigos cíclicos

Antes de abordar el estudio de la decodificación a partir de códigos cíclicos, es necesario dar algunos conceptos previos necesarios para la comprensión del tema. Se iniciará recalcando que por el hecho de que un código cíclico es también lineal, la decodificación por síndrome, mencionada en la Sección 1.4.2, se puede aplicar a este tipo de códigos, pero en su forma polinómica. Así, si $c(x) \in C$ es una palabra código enviada (en forma polinómica) y $u(x)$ es el polinomio recibido, entonces, $e(x) = u(x) - c(x)$ se conoce como el *polinomio error.* Además, el peso de un polinomio es el número de coeficientes no nulos que este posee.

Definición 2.7 (Síndrome de un Polinomio)**.** Sea $C = \langle\langle g(x)\rangle\rangle$ un $[n, n-r]$-código cíclico. El *síndrome de un polinomio* $u(x)$, denotado por $s(u(x))$, es el residuo de dividir $u(x)$ por $g(x)$, es decir,

$$u(x) = q(x)g(x) + s(u(x)) \text{ con } s(u(x)) = 0 \text{ o } \operatorname{grad}[s(u(x)] < r.$$

Dado que un código cíclico en primera instancia es un código lineal, la definición anterior debería coincidir con aquella dada para códigos lineales, precisamente esto es lo que se formula a continuación.

Proposición 2.1. Las definiciones 1.29 y 2.7 son equivalentes.

Demostración. La primera parte de la demostración consiste en dar una base para el código $C = \langle g(x) \rangle$, usar esta para construir una matriz de control de paridad y luego calcular el síndrome de una palabra en $\mathbb{F}_q^n$. La segunda parte se enfocará en encontrar una expresión para el síndrome de una palabra en $\mathbb{F}_q^n$ a partir de su representación polinómica. En ese sentido, primero se demuestra que el conjunto $B = \{x^{r+i} - s_i(x) : i = 0, 1, \ldots, n-r-1\}$ es una base para C, la cual resulta del hecho de que

$$x^{r+i} = a_i(x)g(x) + s_i(x), \tag{2.6.1}$$

donde $s_i(x) = s_{i,0} + s_{i,1}x + \cdots + s_{i,r-1}x^{r-1}$ es el residuo de dividir x^{r+1} entre $g(x)$. Se empezará tomando un $f(x) \in \langle B \rangle$, entonces

$$f(x) = \sum_{i=0}^{n-r-1} b_i(x^{r+i} - s_i(x)) = \sum_{i=0}^{n-r-1} b_i a_i(x)g(x),$$

donde $b_i \in \mathbb{F}_q$. Así, $f(x)$ es un elemento de C también, luego $\langle B \rangle \subseteq C$. Para probar la independencia lineal de B se consideran $\alpha_0, \alpha_1, \ldots, \alpha_{n-r-1} \in \mathbb{F}_q$ tales que

$$\alpha_0(x^r - s_0(x)) + \cdots + \alpha_{n-r-1}(x^{n-1} - s_{n-r-1}(x)) = 0,$$

esta ecuación se puede expresar de la siguiente manera, $\alpha_0 x^r + \cdots + \alpha_{n-r-1}x^{n-1} - k(x) = 0$, donde $-k(x) = -(\alpha_0 s_0(x) + \cdots + \alpha_{n-r-1}s_{n-r-1}(x))$ es un polinomio de grado menor que r, puesto que está formado por polinomios cuyo grado cumple con esa condición, lo cual no deja otra alternativa que $\alpha_0 = \alpha_1 = \cdots = \alpha_{n-r-1} = 0$, es decir, B es linealmente independiente. Adicionalmente, como $\dim(C) = n - r = \dim(\langle B \rangle)$, resulta que $C = \langle B \rangle$ y así queda demostrado que B es una base para C.

El resultado anterior permite establecer que una matriz generadora para el código C es de la forma

$$G = \begin{pmatrix} x^r - s_0(x) \\ x^{r+1} - s_1(x) \\ \vdots \\ x^{n-1} - s_{n-r-1}(x) \end{pmatrix} = \begin{pmatrix} -s_{0,0} & -s_{0,1} & \ldots & -s_{0,r-1} & 1 & 0 & \ldots & 0 \\ -s_{1,0} & -s_{1,1} & \ldots & -s_{1,r-1} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -s_{n-r-1,0} & -s_{n-r-1,1} & \ldots & -s_{n-r-1,r-1} & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

Si se considera la matriz anterior como $G = (S \mid I_{n-r})$, se tiene que una matriz de control de paridad para C es $H = (I_r \mid -S^\top)$, donde $S^\top$ es la transpuesta de S, puesto que

$$GH^\top = (S \mid I_{n-r}) \cdot \begin{pmatrix} I_r \\ -S \end{pmatrix} = S - S = 0.$$

Es decir que la matriz de control de paridad H tendrá la siguiente estructura

$$H = \begin{pmatrix} 1 & 0 & \cdots & 0 & s_{0,0} & s_{1,0} & \cdots & s_{n-r-1,0} \\ 0 & 1 & \cdots & 0 & s_{0,1} & s_{1,1} & \cdots & s_{n-r-1,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & s_{0,r-1} & s_{1,r-1} & \cdots & s_{n-r-1,r-1} \end{pmatrix}.$$

Dada la composición de H, se sigue que el síndrome de u, siguiendo la Definición 1.29, es

$$s(u) = \begin{pmatrix} u \cdot H_1^\top \\ u \cdot H_2^\top \\ \vdots \\ u \cdot H_r^\top \end{pmatrix},$$

y así la j-ésima coordenada en el síndrome es

$$uH_j^\top = u_{j-1} + \sum_{i=0}^{n-r-1} u_{r+i} s_{i,j-1}, \tag{2.6.2}$$

donde $H_j^\top$ representa la j-ésima columna de $H^\top$. Además, observe que $u \cdot H_j^\top$ es el coeficiente de x^{j-1} en la representación polinómica correspondiente al síndrome $s(u)$, es decir

$$s(u)(x) = u \cdot H_1^\top + (u \cdot H_2^\top)x + \cdots + (u \cdot H_j^\top)x^{j-1} + \cdots + (u \cdot H_r^\top)x^{r-1}.$$

Por otra parte, si se considera u como un polinomio, se puede expresar como

$$u(x) = \sum_{j=0}^{r-1} u_j x^j + \sum_{i=0}^{n-r-1} u_{r+i} x^{r+i}.$$

De la ecuación (2.6.1), se sigue que $u(x)$ se representa como

$$\begin{aligned} u(x) &= \sum_{j=0}^{r-1} u_j x^j + \sum_{i=0}^{n-r-1} u_{r+i}(a_i(x)g(x) + s_i(x)) \\ &= g(x) \sum_{i=0}^{n-r-1} u_{r+i} a_i(x) + \Big(\sum_{j=0}^{r-1} u_j x^j + \sum_{i=0}^{n-r-1} u_{r+i} s_i(x) \Big) \\ &= g(x)q(x) + s(x), \end{aligned}$$

donde $q(x) = \sum_{i=0}^{n-r-1} u_{r+i} a_i(x)$ y el grado del polinomio $s(x) = \sum_{j=0}^{r-1} u_j x^j + \sum_{i=0}^{n-r-1} u_{r+i} s_i(x)$ es menor que r. De la Definición 2.7 resulta que $s(x)$ es el síndrome de $u(x)$. Además, el coeficiente de x^j, en $s(x)$ es $u_{j-1} + u_r s_{0,j-1} + u_{r+1} s_{1,j-1} + \cdots + u_{n-1} s_{n-r-1,j-1}$, lo que equivale a

$$u_{j-1} + \sum_{i=0}^{n-r-1} u_{r+i} s_{i,j-1}. \tag{2.6.3}$$

Finalmente, de las expresiones (2.6.2) y (2.6.3) se sigue el resultado deseado.

□

Observación 2.3. Un polinomio recibido $k(x)$ es una palabra del código, si y solo si, su síndrome es el polinomio nulo. Además, dos polinomios tienen el mismo síndrome, si y solo si, están en la misma clase lateral de C. Así, la forma polinómica de decodificación por síndrome es análoga a la vectorial.

Mediante el siguiente ejemplo se describe con detalle el proceso de decodificación para un código cíclico.

Ejemplo 2.16. Retomando el código cíclico ternario del Ejemplo 2.12, es decir,

$$C = \{00000000, 10101010, 01010101, 20202020, 02020202, 21212121, 12121212, 11111111, 22222222\},$$

se puede observar que este código es capaz de corregir solo un error, según el Teorema 1.13 y el Corolario 1.1, puesto que al ser un código cíclico es lineal también; en otras palabras, corrige todos los errores para los cuales el polinomio error es de peso uno. Los líderes de las clases laterales de peso uno con sus correspondientes síndromes se presentan en la Tabla 2.3.

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	$2x^5 + 2x^3 + 2x$			

Cuadro 2.3: Tabla de líderes con sus síndromes.

Ahora bien, supóngase que se ha recibido la palabra $u = 21112121$, cuando la palabra originalmente enviada fue 21212121, o sea que ocurrió un error en la coordenada seis, contada de derecha a izquierda, durante la transmisión, entonces en términos de polinomios se ha recibido $u(x) = x^7 + 2x^6 + x^5 + 2x^4 + x^3 + x^2 + x + 2$, lo siguiente es aplicar el algoritmo de la división y determinar el síndrome de u, así

$$u(x) = x^7 + 2x^6 + x^5 + 2x^4 + x^3 + x^2 + x + 2 = (x^6 + x^4 + x^2 + 1)(x + 2) + (2x^2).$$

Luego, $s(u(x)) = 2x^2$ y de la tabla 2.3 se tiene que el líder de esta clase lateral es $a(x) = 2x^2$,

por tanto, $u(x)$ se decodifica como

$$\begin{aligned} c(x) = u(x) - a(x) &= (x^7 + 2x^6 + x^5 + 2x^4 + x^3 + x^2 + x + 2) - (2x^2) \\ &= x^7 + 2x^6 + x^5 + 2x^4 + x^3 - x^2 + x + 2 \\ &= x^7 + 2x^6 + x^5 + 2x^4 + x^3 + 2x^2 + x + 2 \text{ en } \mathbb{F}_3[x], \end{aligned}$$

es decir que $u = 21112121$ se decodifica como $c = 21212121$, corrigiendo efectivamente el error.

Es de recalcar que la técnica aplicada para decodificar a partir de códigos cíclicos hasta el momento, no presenta mayores ventajas con respecto a los códigos lineales, es por ello que a continuación se describe como se puede sacar provecho de esta característica para mejorar el proceso de decodificación. Entonces, supóngase que es posible decodificar el coeficiente principal de cualquier palabra recibida $u(x)$, despreciando el hecho de que este sea nulo o no, se supondrá que dicho término es el coeficiente de x^{n-1}. De esta manera, luego de decodificar el coeficiente principal de $u(x)$, es posible realizar un desplazamiento cíclico módulo $x^n - 1$ y decodificar el coeficiente principal de la nueva palabra resultante de la rotación, pero que a su vez corresponde al coeficiente de x^{n-2} en $u(x)$. De forma análoga, se rota la palabra $u(x)$ hasta lograr decodificar todos los coeficientes que esta posee. La ventaja de este método radica en el hecho de que solo se requiere de las filas de la tabla de líderes-síndromes 2.3 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 las clases laterales de grado $n - 1$, consiste 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 lo tanto, se cambia dicho coeficiente por uno de los elementos restantes de $\mathbb{F}_q$, luego, se vuelve a calcular el síndrome de la palabra resultante de dicho cambio. Este proceso se repite hasta que el síndrome de la palabra obtenida por el cambio del coeficiente de x^{n-1} no aparezca en la tabla, ya que este nuevo coeficiente es el correcto.

Observación 2.4. 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 de los líderes de la clase lateral de grado $n - 1$, porque el líder de cada clase lateral corresponde al error en una determinada posición, en este caso en el coeficiente principal, de esta manera, si el síndrome de $u(x)$ no aparece en la tabla, significa que el error no se encuentra en el coeficiente del término de grado $n - 1$; este hecho se puede vislumbrar al analizar el arreglo Slepiano, dada la analogía ya demostrada entre las definiciones de síndrome.

Ejemplo 2.17. Volviendo al Ejemplo 2.16, en la tabla de líderes-síndrome 2.3, se observa que los líderes de peso uno y de grado $n - 1 = 7$ son x^7 y $2x^7$, por tanto solo se necesita la tabla 2.4.

En este caso, se supondrá nuevamente que se recibe la palabra $u = 21112121$, o equivalentemente $u(x) = x^7 + 2x^6 + x^5 + 2x^4 + x^3 + x^2 + x + 2$, cuando se ha enviado la palabra $u = 21212121$. Dado

Líder	Síndrome
x^7	$2x^5 + 2x^3 + 2x$
$2x^7$	$x^5 + x^3 + x$

Cuadro 2.4: Tabla de líderes y síndromes reducida.

que $s(u(x)) = 2x^2$ no está en la tabla 2.4, 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 $u(x)$

$$x(x^7 + 2x^6 + x^5 + 2x^4 + x^3 + x^2 + x + 2) \text{ mód } (x^8 - 1) = 2x^7 + x^6 + 2x^5 + x^4 + x^3 + x^2 + 2x + 1,$$

y se calcula su síndrome, el cual es $2x^3$. Nuevamente, este término no está en la Tabla 2.4, por tanto, el coeficiente de x^6 en $u(x)$ es correcto. Luego de haber repetido este mismo proceso 5 veces, se observará que los síndromes de las palabras resultantes no se encuentran en la tabla 2.4, no obstante en la iteración 6 se tiene que

$$u'(x) = x^7 + x^6 + 2x^5 + x^4 + 2x^3 + x^2 + 2x + 1, \tag{2.6.4}$$

cuyo síndrome es $x^5 + x^3 + x$, si se encuentra en la Tabla 2.4. Por ello se deduce que el coeficiente de x^2 en $u(x)$ es incorrecto; lo siguiente es cambiar el coeficiente de x^7 en la parte derecha de la expresión 2.6.4 ya sea por 0 o por 2; al cambiarlo por 0, la palabra obtenida y su síndrome son $x^6 + 2x^5 + x^4 + 2x^3 + x^2 + 2x + 1$, $2x^5 + 2x^3 + 2x$ respectivamente, este último si se encuentra en la tabla 2.4, luego no puede ser el coeficiente correcto de x^7, en seguida se comprueba con la segunda opción, así $u'(x) = 2x^7 + x^6 + 2x^5 + x^4 + 2x^3 + x^2 + 2x + 1$ y $s(u(x)) = 0$, valor que no se encuentra en la tabla 2.4, lo que indica que ese es el coeficiente de x^2 en la palabra recibida $u(x)$. Por último, se hacen dos rotaciones más de $u(x)$ y en cada una de ellas se comprueba también que los síndromes de las palabras obtenidas no se encuentran en la tabla de síndromes, lo que indica que no ha habido más errores en la transmisión.

De esta manera se decodifica $u(x)$ como $c(x) = 2x^7 + x^6 + 2x^5 + x^4 + 2x^3 + x^2 + 2x + 1$. Corrigiendo el error análogamente al ejemplo 2.16.

2.7. Decodificación con códigos cíclicos usando SAGE

Mediante las siguientes instrucciones, es posible realizar ejercicios exploratorios con SAGE que permiten corroborar los resultados relacionados con la decodificación con códigos cíclicos hasta el momento. Para ejemplificar esto, se considerará un caso particular, esperando que este permita vislumbrar la utilidad de este software en el desarrollo de esta actividad.

Entonces, se supondrá que el código con el que se trabajó es el mismo que en el Ejemplo 2.17, y que la palabra recibida es $u = 21112121$, mientras que la palabra enviada fue $u = 21212121$. Según lo descrito en el Ejemplo 2.16, los pasos a seguir para la decodificación de la palabra recibida son los siguientes.

1. Lo primero es definir el anillo de polinomios P en el que se trabajará en SAGE, el polinomio generador g y el código empleado C para la codificación, lo cual ya se ha hecho en ejemplos anteriores. En seguida se requiere tomar la palabra recibida y determinar su síndrome en forma polinómica, para lo cual se procede de la manera siguiente

```
v=[2,1,1,1,2,1,2,1]
P(v)
P(v).quo_rem(g)[1]
sage:
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + x^2 + x + 2
2*x^2
```

   El primero de los resultados que nos presenta el programa es $v(x)$ y el segundo es $s(v(x))$.

2. Lo que sigue es establece si el síndrome hallado se encuentra o no en la lista de síndromes reducida. Para generar dicha lista de síndromes en SAGE en este caso particular, se ha empleado el siguiente algoritmo y se la ha denotado como "`se`".

```
pe=[]
se=[]
m=0
s=0
z=zero_vector(GF(3),8)
for i in [1..2]:
    l= z+vector((0,0,0,0,0,0,0,i))
    pe.append(P(list(l)))
    s=pe[m].quo_rem(g)[1]
    se.append(s)
    print  pe[m], "  s=",se[m]
    m=m+1
sage:
   x^7   s= 2*x^5 + 2*x^3 + 2*x
  2*x^7   s= x^5 + x^3 + x
```

Aquí z es el vector cero de longitud 8 sobre $\mathbb{F}_3$, a este se le suma el vector $(0000000i)$, con i variando en el conjunto $\{1 \cdots q-1\}$, esto hace variar los coeficientes principales de los polinomios de peso 1 y de grado $n-1$ que luego se almacenan en el vector "`pe`". En el vector "`se`" se guardan los síndromes de estos polinomios y finalmente se imprimen los polinomios error de peso 1 con sus respectivos síndromes.

3. Cuando no haya error en el coeficiente principal del polinomio mensaje recibido, según la Observación 2.4, el síndrome de esta palabra no deberá aparecer en la lista generada en el paso anterior, luego, es necesario rotar una posición los coeficientes en $u(x)$ y repetir el procedimiento hasta aquí realizado. En caso de que los síndromes de las palabras obtenidas luego de cada una de las n rotación no aparezcan en la lista anterior, aquí concluye el proceso de decodificación y se asume que no ha ocurrido error alguno en la palabra $u(x)$. Para realizar las rotación de u en SAGE, se ha implementado el siguiente algoritmo.

```
def shift(L,l):
    k=len(L)-l
    return L[k:]+L[:k]
```

El parámetro l indica que se envían los últimos l elementos a las primeras l posiciones de la palabra L, dado que este es un algoritmo empleado en el capítulo siguiente de códigos cuasi-cíclicos, l es un parámetro variable, sin embargo, al trabajar con códigos cíclicos, siempre se tendrá que $l = 1$. Particularmente, para el ejemplo en cuestión, la rotación de los elementos de u en SAGE se logra de la siguiente manera.

```
v=[2,1,1,1,2,1,2,1]
l=1
shift(v,l)
sage:
[1, 2, 1, 1, 1, 2, 1, 2]
```

4. Por otra parte, en el caso en que haya ocurrido un error en el coeficiente principal del polinomio $u(x)$, el síndrome de $u(x)$ deberá aparecer en la tabla de líderes-síndromes del paso 2, así, por lo descrito en el ejemplo 2.17, es necesario cambiar el coeficiente líder por otro de los elementos en $\mathbb{F}_q$ y repetir el pasos 3. Para este caso específico, a continuación se muestra como generar una tabla en SAGE que imprime en pantalla una lista con $u(x)$ y sus n rotaciones acompañadas de sus respectivos síndromes, lo que permite observar si el coeficiente de x^n es correcto en cada rotación, dependiendo si el síndrome se encuentra o no en la tabla generada en el paso 2.

```
pe=[]
se=[]
s=0
m=0
v=(2,1,1,1,2,1,2,1)
n=len(v)
for i in [1..n]:
    l= vector(v)
    pe.append(P(list(l)))
    s=pe[m].quo_rem(g)[1]
    se.append(s)
    print pe[m], "síndrome", se[m]
    se[m]=shift(v,1)
    v=se[m]
    m=m+1
sage:
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + x^2 + x + 2 síndrome 2*x^2
2*x^7 + x^6 + 2*x^5 + x^4 + x^3 + x^2 + 2*x + 1 síndrome 2*x^3
x^7 + 2*x^6 + x^5 + x^4 + x^3 + 2*x^2 + x + 2 síndrome 2*x^4
2*x^7 + x^6 + x^5 + x^4 + 2*x^3 + x^2 + 2*x + 1 síndrome 2*x^5
x^7 + x^6 + x^5 + 2*x^4 + x^3 + 2*x^2 + x + 2 síndrome x^4 + x^2 + 1
x^7 + x^6 + 2*x^5 + x^4 + 2*x^3 + x^2 + 2*x + 1 síndrome x^5 + x^3 + x
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + 2*x^2 + x + 1 síndrome 2
2*x^7 + x^6 + 2*x^5 + x^4 + 2*x^3 + x^2 + x + 1 síndrome 2*x
```

Lo anterior, revela que hay un error en la coordenada de u en la posición i contada de derecha a izquierda, donde i representa el número de la iteración, de allí que es necesario cambiar dicho término en u y repetir el procedimiento. Para este ejemplo, se toma el caso en que $u = 21212121$, omitiendo el paso en que $u = 21012121$, dado que la conclusión es la misma a la que se llegó con $u = 21112121$, así entonces, se tiene que

```
pe=[]
se=[]
s=0
m=0
v=(2,1,2,1,2,1,2,1)
n=len(v)
z=zero_vector(GF(3),8)
```

```
for i in [1..n]:
    l= vector(v)
    pe.append(P(list(l)))
    s=pe[m].quo_rem(g)[1]
    se.append(s)
    print pe[m], "síndrome", se[m]
    se[m]=shift(v,1)
    v=se[m]
    m=m+1
sage:
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + 2*x^2 + x + 2 síndrome 0
2*x^7 + x^6 + 2*x^5 + x^4 + 2*x^3 + x^2 + 2*x + 1 síndrome 0
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + 2*x^2 + x + 2 síndrome 0
2*x^7 + x^6 + 2*x^5 + x^4 + 2*x^3 + x^2 + 2*x + 1 síndrome 0
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + 2*x^2 + x + 2 síndrome 0
2*x^7 + x^6 + 2*x^5 + x^4 + 2*x^3 + x^2 + 2*x + 1 síndrome 0
x^7 + 2*x^6 + x^5 + 2*x^4 + x^3 + 2*x^2 + x + 2 síndrome 0
2*x^7 + x^6 + 2*x^5 + x^4 + 2*x^3 + x^2 + 2*x + 1 síndrome 0
```

Teniendo que ninguno de los síndromes se encuentra en la tabla del paso 2, se concluye que $u = 21212121$ es la palabra originalmente enviada. Lo que efectivamente corrige el error en el mensaje.

Capítulo 3

Códigos cuasi-cíclicos

Desde el primer trabajo de Shannon en 1948, los teóricos en codificación han estado tratando de diseñar códigos capaces de transmitir una gran cantidad de información, con buenos índices de detección y corrección de errores. Una parte importante del trabajo sobre códigos correctores de errores de los últimos sesenta años ha estado enfocada en la construcción de diferentes tipos de códigos definidos sobre anillos conmutativos, por lo cual, en un principio las investigaciones se centraron en los códigos sobre campos finitos. Recientemente, la codificación sobre anillos se ha constituido en una área de constante exploración y muchos tipos de códigos con buenos parámetros se pueden construir en base a este campo [3, 4, 15, 29]. Esto ha llevado a que en la literatura se encuentren diferentes trabajos sobre diversos tipos de anillos como los anillos no conmutativos, los anillos de grupo, los skew-rings, entre otros [6–9, 12, 13, 16, 17, 24, 29]. En este sentido se pueden encontrar diferentes estudios, en los que se presentan generalizaciones del concepto de código cíclico. Así por ejemplo, Boucher, et. al, generalizan en [8] la noción de código cíclico mediante el uso de polinomios generadores en un anillo de polinomios no conmutativos (skew polynomial ring).

En este capítulo se estudiará el concepto de código cuasi-cíclico con base en el trabajo de Barbier et al. [5], en el cual se presenta este concepto como una generalización de aquel dado para código cíclico. Aquí se hace un breve recorrido por algunos aspectos que ligan a los códigos cuasi-cíclicos con los códigos cíclicos, definiendo algunos términos que llevan a explicitar la conexión existente entre estos códigos, destacando que los códigos cuasi-cíclicos son un claro ejemplo de estructuras basadas en anillos no conmutativos, para los cuales se presentarán algunos resultados encaminados a la obtención de matrices y polinomios generadores de estos.

3.1. Conceptos principales

Hasta este momento, las rotaciones cíclicas o *shift* de los elementos en una palabra $c = (c_1, c_2, \ldots, c_n) \in C$ se han venido considerando de la siguiente forma

$$\begin{aligned} T : \mathbb{F}_q^n &\longrightarrow \mathbb{F}_q^n \\ T(c_1, c_2, \ldots, c_n) &\longmapsto (c_n, c_1, ..., c_{n-1}). \end{aligned} \tag{3.1.1}$$

Observación 3.1. En algunos de los textos relacionados con códigos cíclicos y/o cuasi-cíclicos, como en [5], las rotaciones cíclicas o shifts, que aquí se denotarán como S, son consideradas de otra manera, como se muestra en seguida

$$\begin{aligned} S(c) &= S(c_1, c_2, \ldots, c_n) \\ &= (c_2, c_3, \ldots, c_n, c_1). \end{aligned}$$

No obstante, es posible establecer una conexión entre las dos definiciones de rotaciones cíclicas; así para cualquier $c \in C$

$$S(c) = T^{n-1}(c).$$

Por la dualidad existente en el uso de estos dos tipos de rotaciones, tratando de ser fieles a la terminología empleada hasta este momento, muchas de las demostraciones que se presentan en esta sección tomarán como sentido de rotación la que se indica por T; sin embargo habrán momentos en que sea conveniente el uso de la rotación designada por S, particularmente en la implementación de algunos de los algoritmos.

Notación 3.1. Sea m un entero positivo, sea $f : E \to F$ cualquier función de conjuntos. Se denotará por $f^{\times m}$ a la función de conjuntos $f^{\times m} : E^m \to F^m$ tal que

$$f^{\times m}(a_1, a_2, \ldots, a_m) = (f(a_1), f(a_2), \ldots, f(a_m)).$$

Definición 3.1 (Códigos Cuasi-cíclicos)**.** Sean n un entero positivo y ℓ un divisor de n. Se define un código ℓ-cuasi-cíclico sobre $\mathbb{F}_q$ de longitud n, a un código estable bajo T^ℓ, es decir, cerrado bajo ℓ rotaciones individuales de elementos de las palabras del código. Si el contexto es claro, simplemente la referencia se hace como código ℓ-cuasi-cíclico.

Observación 3.2. En términos de las rotaciones determinadas por la expresión 3.1.1, un código C es cuasi-cíclico, si y sólo sí, $T^\ell(C) = C$. En este sentido se sabe que un código C es cíclico, si y sólo si, $T(C) = C$, es decir, un código cíclico es un código 1-cuasi-cíclico, ver Definición 2.3. Por supuesto, esto es una evidencia de que el concepto de código cuasi-cíclico es una generalización del de código cíclico. En lo que sigue se mostrará que esta conexión va un poco más allá.

Definición 3.2 (Función plegado y desplegado, ver [5])**.** Sea ℓ un entero, y $\alpha \in \mathbb{F}_{q^\ell}$ tal que $\{1, \alpha, \ldots, \alpha^{\ell-1}\}$, ver la Definición 1.13, es una $\mathbb{F}_q$-base del espacio vectorial $\mathbb{F}_{q^\ell}$. Siguiendo la misma línea de ideas empleadas en [5], se define el *plegado* como la función $\mathbb{F}_q$-lineal

$$\begin{array}{rcl} \phi : \mathbb{F}_q^\ell & \longrightarrow & \mathbb{F}_{q^\ell} \equiv \mathbb{F}_q[\alpha] \\ (a_1, \ldots, a_\ell) & \longmapsto & a_1 + a_2\alpha + \cdots + a_\ell\alpha^{\ell-1}. \end{array}$$

Por otra parte, el *desplegado* es la función inversa $\mathbb{F}_q$-lineal siguiente

$$\begin{array}{rcl} \phi^{-1} : \mathbb{F}_{q^\ell} & \longrightarrow & \mathbb{F}_q^\ell \\ \overline{a} = a_1 + a_2\alpha + \cdots + a_\ell\alpha^{\ell-1} & \longmapsto & (a_1, a_2, \ldots, a_\ell). \end{array}$$

A partir de las funciones descritas en la Definición 3.2, es posible formular las siguientes definiciones.

Definición 3.3. (Códigos plegados y desplegados). Supóngase que $n = m\ell$. Se define el *código plegado* de C como $\phi^{\times m}(C)$. Sea C' un código en $\mathbb{F}_{q^\ell}^m$, se define el *código desplegado* de C' como $(\phi^{-1})^{\times m}(C')$.

Teorema 3.1. Un código C es ℓ-cuasi-cíclico, si y sólo si, su plegado $C' = \phi^{\times m}(C)$ es cíclico. Pero C' no es necesariamente $\mathbb{F}_{q^\ell}$-lineal.

Demostración. Supóngase que C es un código ℓ-cuasi-cíclico, entonces, sea $(\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_m) \in \phi^{\times m}(C)$. Se pretende demostrar que $(\overline{a}_m, \overline{a}_1, \ldots, \overline{a}_{m-1})$ también está en $\phi^{\times m}(C)$, donde $\overline{a}_i = a_{i0} + a_{i1}\alpha + \cdots + a_{i\ell-1}\alpha^{\ell-1}$, para ello, se debe comprobar la existencia de un elemento $\beta \in (\mathbb{F}_q^\ell)^m$, tal que

$$\phi^{\times m}(\beta) = (\overline{a}_m, \overline{a}_1, \ldots, \overline{a}_{m-1}).$$

Como $(\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_m) \in \phi^{\times m}(C)$, existe $(\gamma_1, \gamma_2, \ldots, \gamma_m)$ en C, con $\gamma_i = (a_{i0}, a_{i1}, \ldots, a_{i\ell-1})$ e $i \in \{1, \ldots, m\}$, donde

$$\phi^{\times m}(\gamma_1, \gamma_2, \ldots, \gamma_m) = (\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_m).$$

Además, si C es ℓ-cuaci cíclico, es estable bajo T^ℓ, luego se tiene

$$\begin{array}{rcl} T^\ell(a_{10}, \ldots, a_{1\ell-1}, a_{20}, \ldots, a_{2\ell-1}, \ldots, a_{m0}, \ldots, a_{m\ell-1}) & = & (a_{m0}, \ldots, a_{m\ell-1}, a_{10}, \ldots, a_{1\ell-1}, \\ & & \ldots, a_{m-1,0}, \ldots, a_{m-1,\ell-1}) \\ & = & (\gamma_m, \gamma_1, \ldots, \gamma_{m-1}). \end{array}$$

Así, $(\gamma_m, \gamma_1, \ldots, \gamma_{m-1})$ también pertenece a C. Luego, si se define $\beta = (\gamma_m, \gamma_1, \ldots, \gamma_{m-1})$, y se le aplica $\phi^{\times m}$, se sigue que

$$\begin{aligned} \phi^{\times m}(\beta) &= \phi^{\times m}(\gamma_m, \gamma_1, \ldots, \gamma_{m-1}) \\ &= (\phi^{\times m}(\gamma_m), \phi^{\times m}(\gamma_1), \ldots, \phi^{\times m}(\gamma_{m-1})) \\ &= (\overline{a}_m, \overline{a}_1, \ldots, \overline{a}_{m-1}). \end{aligned}$$

Que era lo que se quería demostrar.

Ahora, en el otro sentido, si C' es cíclico, para cualquier $a' = (\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_m)$ en C', entonces, $T(a') = (\overline{a}_m, \overline{a}_1, \ldots, \overline{a}_{m-1})$ también está en C'. Así, para cada rotación de $a' \in C'$, $(\phi^{-1})^{\times m}(a') \in C$, ya que según la definición de ϕ^{-1}, estas se corresponden con cadenas de elementos de C de longitud $m\ell$ que se han rotado ℓ posiciones. En otras palabras, al aplicar las rotaciones de las componentes de cada elemento de $a' \in \phi^{\times m}(C)$, el resultado de computar $(\phi^{-1})^{\times m}(a')$ es un elemento de C rotado ℓ componentes, de donde se sigue que $(\phi^{-1})^{\times m}(C') \subseteq C$. Si se asume que para algún $k \in C$, $(\phi^{-1})^{\times m}(c') \neq k$ para todo $c' \in C'$, entonces, $\phi^{\times m}(k) \notin C'$, lo cual es una contradicción a la definición de código plegado, esto implica que $(\phi^{-1})^{\times m}(C') = C$ y por tanto C es cerrado bajo T^{ℓ}, es decir, es un código ℓ-cuasi-cíclico. □

Lema 3.1. Sea R un anillo conmutativo de ideales principales, y M un módulo libre por izquierda de rango finito s sobre R. Entonces, todo submódulo N de M puede ser generado por a lo más s elementos.

Demostración. Esta es una fácil adaptación de la demostración del Teorema 1.1. □

Lema 3.2. Sea s un entero positivo y R un anillo conmutativo de ideales principales. Entonces, existe una correspondencia uno a uno entre los submódulos de R^s y los ideales a izquierda de $M_s(R)$.

Demostración. Para un submódulo $N \subseteq R^s$, es posible construir un ideal por izquierda de $M_s(R)$ cuyos elementos tienen filas en N. En efecto, sea $T \in M_s(R)$ una matriz de tamaño $s \times s$, entonces

$$T = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1s} \\ x_{21} & x_{22} & \cdots & x_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ x_{s1} & x_{s2} & \cdots & x_{ss} \end{pmatrix}.$$

Nótese que cada columna de T es también un elemento de R^s. En seguida, considérese un subconjunto I de $M_s(R)$ cuya característica principal es que las filas de todos sus elementos son elementos de N, entonces, para todo $P \in I$ y para todo $T \in M_s(R)$, el producto PT, definido como el producto usual de matrices, genera una matriz de tamaño $n \times n$ cuyas filas son elementos de N, puesto que N es un submódulo de R^s, es decir, $IM_s(R) \subseteq I$, así I es un ideal de $M_s(R)$. En el sentido inverso, para un ideal por izquierda $I \subseteq M_s(R)$, se puede asociar el submódulo de R^s generado por todas las filas de todos los elementos de I. Es directo comprobar que estos mapeos son inversos el uno del otro. □

Observación 3.3. Nótese que $M_\ell(\mathbb{F}_q)[X]/\langle X^m - 1\rangle$ y $M_\ell(\mathbb{F}_q[X]/\langle X^m - 1\rangle)$ son isomorfos como anillos, además $R = \mathbb{F}_q[X]/\langle X^m - 1\rangle$ es un anillo conmutativo de ideales principales.

Demostración. La función que determina el isomorfismo entre $M_\ell(\mathbb{F}_q)[X]/\langle X^m-1\rangle$ y $M_\ell(\mathbb{F}_q[X]/\langle X^m-1\rangle)$ puede ser definida de la siguiente manera. Sea $A = A_0 + A_1X + A_2X^2 + \cdots + A_{m-1}X^{m-1} \in M_\ell(\mathbb{F}_q)[X]/\langle X^m-1\rangle$, con $A_k \in M_\ell(\mathbb{F}_q)$ y $k \in \{0,\ldots,m-1\}$ se describen así: $A_0 = [a_{0,i,j}], A_1 = [a_{1,i,j}], \ldots, A_k = [a_{k,i,j}]$. Con esto en mente, entonces

$$\phi(A) = [a_{i,j}(x) + \langle x^m - 1\rangle],$$

donde

$$\begin{aligned} a_{i,j}(x) &= a_{0,i,j} + a_{1,i,j}x + \cdots + a_{m-1,i,j}x^{m-1} \\ &= \sum_{k=0}^{m-1} a_{k,i,j}x^k + \langle x^m - 1\rangle. \end{aligned}$$

Para $1 \leq i,j \leq \ell$. Por otra parte, entiéndase que en $M_\ell\mathbb{F}_q[X]$, el polinomio $X^m - 1$ puede ser expresado de la siguiente manera

$$X^m - 1 = I_\ell X^m - I_\ell,$$

cuya imagen bajo ϕ, expresada en forma matricial es la siguiente

$$\begin{pmatrix} x^m-1 & 0 & \cdots & 0 \\ 0 & x^m-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x^m-1 \end{pmatrix}$$

Ahora bien, para demostrar que ϕ es inyectiva, biyectiva e isomorfismo se procede de la siguiente manera. Sean

$$\begin{aligned} \phi(A(X) + \langle X^m - 1\rangle) &= \phi(B(X) + \langle X^m - 1\rangle) \\ [a_{i,j}(x) + \langle x^m - 1\rangle] &= [b_{i,j}(x) + \langle x^m - 1\rangle] \end{aligned}$$

por la igualdad de matrices se sigue que estas matrices son iguales componente a componente, es decir,

$$a_{i,j}(x) + \langle x^m - 1\rangle = b_{i,j}(x) + \langle x^m - 1\rangle.$$

Esto implica que $a_{i,j}(x) - b_{i,j}(x) \in \langle x^m - 1\rangle$, o equivalentemente, $a_{i,j}(x) - b_{i,j}(x) = q_{i,j}(x)(x^m - 1)$,

luego

$$[a_{i,j}(x)] - [b_{i,j}(x)] = \begin{pmatrix} (a_{1,1}(x) - b_{1,1}(x)) + \langle x^m - 1\rangle & \cdots & (a_{1,\ell}(x) - b_{1,\ell}(x)) + \langle x^m - 1\rangle \\ \vdots & \ddots & \vdots \\ (a_{\ell,1}(x) - b_{\ell,1}(x)) + \langle x^m - 1\rangle & \cdots & (a_{\ell,\ell}(x) - b_{\ell,\ell}(x)) + \langle x^m - 1\rangle \end{pmatrix}$$
$$= \begin{pmatrix} q_{1,1}(x)(x^m - 1) & \cdots & q_{1,\ell}(x)(x^m - 1) \\ \vdots & \ddots & \vdots \\ q_{\ell,1}(x)(x^m - 1) & \cdots & q_{\ell,\ell}(x)(x^m - 1) \end{pmatrix}$$
$$= \begin{pmatrix} q_{1,1}(x) & \cdots & q_{1,\ell}(x) \\ \vdots & \ddots & \vdots \\ q_{\ell,1}(x) & \cdots & q_{\ell,\ell}(x) \end{pmatrix} \begin{pmatrix} x^m - 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x^m - 1 \end{pmatrix}.$$

Esto indica que $A[X] - B[X] \in \langle X^m - 1\rangle$. De donde resulta que ϕ es inyectiva.
Por otra parte, sea $Y = [y_{i,j} + \langle x^m - 1\rangle]$ en $M_\ell(\mathbb{F}_q[x]/\langle x^m - 1\rangle)$, considérese $X = [y_{i,j}](X) + \langle X^m - 1\rangle \in M_\ell\mathbb{F}_q[X]/\langle X^m - 1\rangle)$, entonces, $\phi(X) = Y$. Y así, ϕ es biyectiva.

Para demostrar el isomorfismo, considérese $A[X] + \langle X^m - 1\rangle$ y $B[X] + \langle X^m - 1\rangle \in M_\ell(\mathbb{F}_q)[X]/\langle X^m - 1\rangle$, en seguida

$$\begin{aligned} \phi(A[X] + \langle X^m - 1\rangle + B[X] + \langle X^m - 1\rangle) &= \phi([a_{i,j}](X) + \langle X^m - 1\rangle + [b_{i,j}](X) + \langle X^m - 1\rangle) \\ &= \phi([a_{i,j}](X) + [b_{i,j}](X) + \langle X^m - 1\rangle) \\ &= \phi(([a_{i,j}] + [b_{i,j}])(X) + \langle X^m - 1\rangle) \\ &= [(a_{i,j}(x) + b_{i,j}(x)) + \langle x^m - 1\rangle] \\ &= [a_{i,j}(x) + \langle x^m - 1\rangle] + [b_{i,j}(x) + \langle x^m - 1\rangle] \\ &= \phi(A[x]) + \phi(B[x]). \end{aligned}$$

Y con esto concluye lo que se quería demostrar. □

En adición al resultado anterior, se tiene que por el Lema 3.1, cualquier submódulo de R^ℓ puede ser generado por a lo más ℓ elementos, de ahí que por el Lema 3.2, cualquier ideal por izquierda de $M_\ell(R) = M_\ell(F_q)[X]/\langle X^m - 1\rangle$ es principal.

Teorema 3.2. Existe una correspondencia uno a uno entre los códigos ℓ-cuasi-cíclicos sobre $\mathbb{F}_q$ de longitud $m\ell$ y los ideales por izquierda de $M_\ell(\mathbb{F}_q[X]/\langle X^m - 1\rangle)$.

Demostración. Sea $g = (g_{11}, \ldots, g_{1\ell}, g_{21}, \ldots, g_{2\ell}, \ldots, g_{m1}, \ldots, g_{m\ell}) \in \mathbb{F}_q^{m\ell}$. Asociamos a g el elemento $\varphi(g) \in (\mathbb{F}_q[X]/\langle X^m - 1\rangle)^\ell$, definido por

$$\begin{aligned} \varphi(g) = (&g_{11} + g_{21}X + \cdots + g_{m1}X^{m-1}, \\ &g_{12} + g_{22}X + \cdots + g_{m2}X^{m-1}, \ldots, \\ &g_{1\ell} + g_{2\ell}X + \cdots + g_{m\ell}X^{m-1}). \end{aligned}$$

Sea Q un código ℓ-cuasi-cíclico de longitud $m\ell$ sobre $\mathbb{F}_q$. Para probar que $\varphi(Q)$ es un submódulo de $\mathbb{F}_q[X]/\langle x^m-1\rangle$, se debe mostrar que este es un grupo Abeliano bajo la suma y que para $f(X)\in(\mathbb{F}_q[X]/\langle X^m-1\rangle)^\ell$ y $H\in\varphi(Q)$, se tiene que $f(X)H\in\varphi(Q)$.

Ahora bien, para demostrar que es un grupo abeliano, se consideran a Y y Z elementos en $\varphi(Q)$, tales que,

$$\begin{aligned} Y &= \varphi(g_{11},\ldots,g_{1\ell},g_{21},\ldots,g_{2\ell},\ldots,g_{m1},\ldots,g_{m\ell}) \\ &= (g_{11}+g_{21}X+\cdots+g_{m1}X^{m-1},\ldots,g_{1\ell}+g_{2\ell}X+\cdots+g_{m\ell}X^{m-1}) \end{aligned}$$

y

$$\begin{aligned} Z &= \varphi(f_{11},\ldots,f_{1\ell},f_{21},\ldots,f_{2\ell},\ldots,f_{m1},\ldots,f_{m\ell}) \\ &= (f_{11}+f_{21}X+\cdots+f_{m1}X^{m-1},\ldots,f_{1\ell}+f_{2\ell}X+\cdots+f_{m\ell}X^{m-1}), \end{aligned}$$

luego,

$$\begin{aligned} Y+Z &= (g_{11}+g_{21}X+\cdots+g_{m1}X^{m-1},\ldots,g_{1\ell}+g_{2\ell}X+\cdots+g_{m\ell}X^{m-1}) \\ &\quad +(f_{11}+f_{21}X+\cdots+f_{m1}X^{m-1},\ldots,f_{1\ell}+f_{2\ell}X+\cdots+f_{m\ell}X^{m-1}) \\ &= ((g_{11}+f_{11})+(g_{21}+f_{21})X+\cdots+(g_{m1}+f_{m1})X^{m-1}, \\ &\quad \ldots,(g_{1\ell}+f_{1\ell})+(g_{2\ell}+f_{2\ell})X+\cdots+(g_{m\ell}+f_{m\ell})X^{m-1}) \\ &= \varphi((g_{11}+f_{11}),(g_{21}+f_{21}),\ldots,(g_{m1}+f_{m1}),\ldots,(g_{1\ell}+f_{1\ell}),(g_{2\ell}+f_{2\ell}),\ldots,(g_{m\ell}+f_{m\ell})), \end{aligned}$$

dado que $\mathbb{F}_q^{m\ell}$ es un grupo Abeliano, se tendrá

$$\begin{aligned} &= \varphi((f_{11}+g_{11}),(f_{21}+g_{21}),\ldots,(f_{m1}+g_{m1}),\ldots,(f_{1\ell}+g_{1\ell}),(f_{2\ell}+g_{2\ell}),\ldots,(f_{m\ell}+g_{m\ell})) \\ &= (f_{11}+g_{11})+(f_{21}+g_{21})X+\cdots+(f_{m1}+g_{m1})X^{m-1}, \\ &\quad \ldots,(f_{1\ell}+g_{1\ell})+(f_{2\ell}+g_{2\ell})X+\cdots+(f_{m\ell}+g_{m\ell})X^{m-1}) \\ &= (f_{11}+f_{21}X+\cdots+f_{m1}X^{m-1},\ldots,f_{1\ell}+f_{2\ell}X+\cdots+f_{m\ell}X^{m-1}) \\ &\quad +(g_{11}+g_{21}X+\cdots+g_{m1}X^{m-1},\ldots,g_{1\ell}+g_{2\ell}X+\cdots+g_{m\ell}X^{m-1}) \\ &= Z+Y. \end{aligned}$$

Es decir, $\varphi(Q)$ es un grupo Abeliano.
Para probar la segunda condición obsérvese que basta con tomar $f(X)=X$ y que $X^m=1$ en $\mathbb{F}_q[X]/\langle X^m-1\rangle$. Entonces, sea $Y=(g_{11}+g_{21}X+\cdots+g_{m1}X^{m-1},\ldots,g_{1\ell}+g_{2\ell}X+\cdots+g_{m\ell}X^{m-1})\in$

$\varphi(Q)$, luego

$$\begin{aligned} XY &= X(g_{11} + g_{21}X + \cdots + g_{m-1,1}X^{m-2} + g_{m1}X^{m-1}, \ldots, g_{1\ell} + g_{2\ell}X + \cdots + g_{m-1,\ell}X^{m-2} \\ &\quad + g_{m\ell}X^{m-1}) \\ &= (g_{11}X + g_{21}X^2 + \cdots + g_{m-1,1}X^{m-1} + g_{m1}, \ldots, g_{1\ell}X + g_{2\ell}X^2 + \cdots + g_{m-1,\ell}X^{m-1} + g_{m\ell}) \\ &= \varphi(g_{m1}, \ldots, g_{m\ell}, g_{11}, \ldots, g_{1\ell}, \ldots, g_{m-1,1}, \ldots, g_{m-1\ell}) \\ &= \varphi(T^\ell(g_{11}, \ldots, g_{1\ell}, \ldots, g_{m-1,1}, \ldots, g_{m-1\ell}, g_{m1}, \ldots, g_{m\ell})). \end{aligned}$$

Pero al ser Q-cuasi-cíclico, se tiene que este es cerrado bajo los desplazamientos originados por T^ℓ, así, $\varphi(T^\ell(g_{11}, \ldots, g_{1\ell}, \ldots, g_{m-11}, \ldots, g_{m-1\ell}, g_{m1}, \ldots, g_{m\ell})) \in \varphi(Q)$. De la misma manera se puede demostrar que $X^iY \in \varphi(Q)$, en consecuencia, para $f(X) = a_0 + a_1X + \cdots + a_{m-1}X^{m-1} \in \mathbb{F}_q[X]/\langle X^m - 1\rangle$ se cumple que $f(X)Y \in \varphi(Q)$.

Finalmente, $\varphi(Q)$ es un submódulo sobre $(\mathbb{F}_q[X]/\langle X^m - 1\rangle)^\ell$. Entonces, φ induce una correspondencia uno a uno entre los códigos ℓ-cuasi-cíclicos de longitud $m\ell$ sobre $\mathbb{F}_q$ y los submódulos de $(\mathbb{F}_Q[X]/\langle X^m - 1\rangle)^\ell$ según lo descrito en el Lema 3.2. □

Definición 3.4. Sea $x = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$, se define la *proyección* de x, denotada por $\mathrm{pr}_{i,j}(x)$, sobre las coordenadas $i, i+1, \ldots, j$ así

$$\begin{aligned} \mathrm{pr}_{i,j} : \mathbb{F}_q^n &\longrightarrow \mathbb{F}_q^{j-i+1} \\ (x_1, \ldots, x_n) &\longmapsto (x_i, x_{i+1}, \ldots, x_{j-1}, x_j). \end{aligned}$$

Esto permite formular el siguiente lema.

Lema 3.3. Sea C un código ℓ-cuasi-cíclico sobre $\mathbb{F}_q$ de dimensión k y de longitud $m\ell$. Entonces existe un entero r tal que $0 < r \leq k$ para cualquier matriz generadora G de C, con $0 \leq i \leq m-1$, donde el rango de $i\ell + 1, i\ell + 2, \ldots, (i+1)\ell$ columnas de G es r.

Demostración. Sea $\mathrm{pr}_{i,j}$ la proyección de las palabras del código C sobre sus coordenadas $i, i+1, \ldots, j$ como se indica en seguida

$$\begin{aligned} \mathrm{pr}_{i,j} : \mathbb{F}_q^n &\longrightarrow \mathbb{F}_q^{j-i+1} \\ (x_1, \ldots, x_n) &\longmapsto (x_i, x_{i+1}, \ldots, x_{j-1}, x_j). \end{aligned}$$

Ahora, sea $r = \dim \mathrm{pr}_{1,\ell}(C)$, es decir, la dimensión de las proyecciones de todas las palabras de C sobre las coordenadas 1 a ℓ, y sea G una matriz generadora de C. Dado que la dimensión de G es k, entonces, G dispone de al menos una columna diferente de cero, así $r \geq 1$, según el Lema 3.3. Por otra parte, sean las filas de G, denotadas por $g_1, \ldots, g_k$ una base para C, nuevamente por el Lema 3.3, $r \leq \dim C = k$, entonces

$$\begin{aligned} r &= \dim \mathrm{pr}_{1,\ell}(C) \\ &= \dim\langle \mathrm{pr}_{1,\ell}(g_1), \ldots, \mathrm{pr}_{1,\ell}(g_k)\rangle. \end{aligned}$$

Por lo tanto, r representa el rango de la matriz formada por las primeras ℓ columnas de G.
Ahora, sea $0 \leq i \leq m-1$. Dado que $T^\ell \in \mathrm{Aut}(C)$, es decir, T^ℓ es un automorfismo de C, tenemos

$$\begin{aligned} r &= \dim \mathrm{pr}_{1,\ell}(C) \\ &= \dim\langle \mathrm{pr}_{1,\ell} T^{-i\ell}(g_1), \ldots, \mathrm{pr}_{1,\ell} T^{-i\ell}(g_k)\rangle \\ &= \dim\langle \mathrm{pr}_{i\ell+1,(i+1)\ell}(g_1), \ldots, \mathrm{pr}_{i\ell+1,(i+1)\ell}(g_k)\rangle. \end{aligned}$$

De esta manera, r es el rango de las $i\ell+1, i\ell+2, \ldots, (i+1)\ell$ columnas de G. Es decir, que sin importar el orden del bloque en el que se haga la proyección de todas las palabras del código C, el rango r seguirá siendo el mismo; que es lo que se quería demostrar. □

Definición 3.5 (Rango de Bloque)**.** Siguiendo la notación del Lema 3.3, se llamará al entero r el *rango de bloque* de C. Nótese que r depende sólo de C y no de alguna matriz generadora de C en particular.

El polinomio generador de un código ℓ-cuasi-cíclico

Para este apartado, se fija un código ℓ-cuasi-cíclico C sobre $\mathbb{F}_q$. Si $\ell = 1$, entonces C es un código cíclico de longitud n. El resultado obtenido aquí para códigos ℓ-cuasi-cíclicos es análogo al presentado en el Teorema 2.4, es decir, una matriz generadora de C puede estar dada por

$$\begin{pmatrix} g(X) & & & \\ & Xg(X) & & \\ & & \ddots & \\ & & & X^{n-deg\ g} g(X) \end{pmatrix},$$

donde $g(X) \in \mathbb{F}_q[X]$ es el polinomio generador de C. El rango de bloque de C es 1 ya que como se observa, se puede escribir una matriz generadora de C con sólo un vector y sus shifts (porque $T^\ell = T$.) La manera de generalizar este resultado a los códigos cuasi-cíclicos se logra usando el rango de bloque.

Sea r el rango de bloque de C, mediante el siguiente algoritmo, se computará una base de C a partir de r vectores de G y algunos de sus shifts, donde G es una de las matrices generadoras de C. Adicionalmente, se denominará como el *primer indice* de un vector no cero $x = (x_1, \ldots, x_{m\ell})$, al más pequeño entero entre $0 \leq i \leq m-1$ tal que $(x_{i\ell+1}, \ldots, x_{(i+1)\ell}) \neq 0$; este entero se denotará por $\mathcal{F}(\mathbf{x}) = \mathcal{F}(x_1, \ldots, x_{m\ell})$. Con esto en mente, es posible definir la proyección de la manera siguiente

$$\begin{aligned} p : \mathbb{F}_q^{m\ell} &\longrightarrow \mathbb{F}_q^{\ell} \\ \mathbf{x} = (x_1, \ldots, x_{m\ell}) &\longmapsto (x_{i\ell+1}, \ldots, x_{(i+1)\ell}) \end{aligned}$$

donde $i = \mathcal{F}(x_1, \ldots, x_n)$ si $\mathbf{x} \neq 0$ y $p(\mathbf{0}) = 0$.

3.1.1. Algoritmo matriz generadora de códigos cuasi-cíclicos.

Siguiendo el objetivo que se busca en [5], el Algoritmo 3.1 que se presenta aquí y que ha sido implementado en el software SAGE, determina e imprime en pantalla una matriz G_0, formada a partir de algunas de las filas de G que son linealmente independientes y a las cuales se le anexan algunos de sus shifts.

No obstante, la implementación de este algoritmo en la plataforma mencionada, requirió que se definiera algunas funciones más que permitieron lograr resultados análogos a los que se puede apreciar en [5]; estas funciones se presentan a continuación con su respectiva descripción.

Es de aclarar también, que un requisito esencial para que este algoritmo funcione de forma análoga al presentado en [5], fue el de considerar los shifts o rotaciones cíclicas determinadas por S, la cual se menciona en la Observación 3.1. Esta consideración permitió que al determinar la independencia lineal de las proyecciones de las filas de G y/o sus rotaciones, se trabaje sobre el mismo bloque en todas las filas.

Algoritmo 3.1 (Función F)**.**

```
def F(v,l):
    n=len(v)
    m=Integer(n/l)
    for k in [1..m]:
        p=proj(v,l,k)
        if p<>list(zero_vector(GF(2),l)):
    return k
```

En este algoritmo, las entradas son un vector (`v`) y la longitud de los bloques (`l`) en que se dividirá la palabra, para luego determinar cuál de dichos bloques es el de mayor índice y que es diferente del vector 0 de longitud ℓ. En otras palabras, siguiendo la definición dada anteriormente de $\mathcal{F}$, con esta función se determina el

$$\max\{\mathcal{F}(g_i); i \in \{1, \ldots, k\}\}.$$

Algoritmo 3.2 (Función proyección (proj))**.**

```
def proj(v,l,k):
    p=[v[i] for i in [(k-1)*l..(k*l-1)]]
    return p
```

De a cuerdo con la Definición 3.4, aquí las entradas son un vector (`v`) de longitud $n = m\ell$, el tamaño de los bloques (`l`) en que se dividirá cada vector, y `k` que corresponde al orden del bloque

en el cual se desea realizar la proyección, el cual se puede determinar a partir del algoritmo 3.1. La salida es un vector de longitud ℓ, que es el primer bloque de `v` diferente del vector $\mathbf{0}$ de longitud ℓ.

Algoritmo 3.3 (Realiza los shifts de las palabras)**.**

```
def shift2(v,l):
    return v[l:]+v[:l]
```

Este algoritmo permite realizar rotaciones de `l` elementos en un vector determinado `v`, donde `l` y `v` se constituyen como las entradas requeridas, enviando las primeras ℓ coordenadas a las últimas ℓ posiciones, es decir, realizando las rotaciones cíclicas según la definición de S dada en la Observación 3.1.

Algoritmo 3.4 (Presenta la matriz generadora de un código ℓ-cuasi-cíclico C)**.**

```
def quasi(G,l,Fq):
    n=G.ncols()
    f=G.nrows()
    C=Set([])
    for i in [0..(f-1)]:
        C=C+Set([F(G[i],l,Fq)])
    M=max(C)
    V=[]
    G0=[]
    for j in [0..M-1]:
        B=[i for i in [0..(f-1)] if F(G[i],l,Fq)==M-j]
        S=[]
        for v in V:
            S.append(proj(v,l,M-j))
        for x in B:
            S.append(proj(G[x],l,M-j))
            if (zero_vector(Fq,l) in matrix(S).rref())==False:
                V.append(G[x])
            else:
                S.remove(proj(G[x],l,M-j))
            G0.append(list(x))
        T=[shift2(list(x),l) for x in V]
        V=T
    return matrix(G0)
```

Las entradas aquí son, una matriz (`G`) generadora de un código ℓ-cuasi-cíclico, la longitud (`l`) de los m bloques en los que se desea dividir los vectores que forman esta matriz y también el cuerpo finito (`Fq`) en el cual se está trabajando. Lo que presenta finalmente el algoritmo es una matriz formada sólo por algunos de los vectores de la matriz original que son linealmente independientes y algunos de los shifts de éstos.

De manera general, el programa funciona de la siguiente manera: Luego de recibir la matriz generadora G de un código ℓ-cuasi-cíclico, y la longitud ℓ de los bloques en los que se dividirá cada fila o vector en la base del código, encuentra $M = \max\{\mathcal{F}(g_i)\}$ de entre todas las filas g_i de G, con $i \in \{0, \ldots, k-1\}$, para luego determinar y agrupar en B los índices de las filas que satisfacen $\mathcal{F}(g_i) = M - j$ con $j \in \{0, \ldots, M-1\}$. En seguida, el algoritmo considera las proyecciones de cada una de las palabras (filas) g_x, con $x \in B$, sobre el bloque $M - j$, para determinar si son o no linealmente independientes, si lo son, se agregan los vectores (filas) g_x al conjunto V, que a su vez se anexan a la matriz $G0$. Antes de repetir este proceso con los elementos g_x que satisfacen $\mathcal{F}(g_x) = M - j + 1$, se aplica la transformación S^ℓ a las palabras en V, que constituyen los elementos con los cuales se entrará a comprobar la independencia lineal de las proyecciones de las palabras siguientes. De esta manera, la matriz $G0$ sólo estará conformada por elementos de la matriz original G y algunos de los shifts de estas que sean linealmente independientes.

Corolario 3.1. Existen $g_1 \ldots, g_r$ vectores linealmente independientes de C tales que $g_1, \ldots, g_r, T^\ell(g_1)$, $\ldots, T^\ell(g_r), \ldots, T^{(m-1)\ell}(g_1), \ldots, T^{(m-1)\ell}(g_r)$ generan a C. Si se denota por $g_{i,j}$ la j-ésima coordenada de g_i y se hace

$$G_i = \begin{pmatrix} g_{1,i\ell+1} & \cdots & g_{1,(i+1)\ell} \\ g_{2,i\ell+1} & \cdots & g_{2,(i+1)\ell} \\ \vdots & & \vdots \\ g_{r,i\ell+1} & \cdots & g_{r,(i+1)\ell} \\ & 0 & \end{pmatrix} \in M_\ell(\mathbb{F}_q)$$

y

$$g(X) = \frac{1}{X^v} \sum_{i=0}^{m-1} G_i X^i \ \in M_\ell(\mathbb{F}_q)[X],$$

donde v es el menor entero tal que $G_i \neq 0$, entonces C corresponde al ideal a izquierda $\langle g(X) \rangle$ mencionado en el Teorema 3.2. La construcción del polinomio aquí descrita se ilustra de forma clara mediante el ejemplo 3.1.

Corolario 3.2. Tomando la notación de la prueba del Teorema 3.2, el submódulo

$$\varphi(C) \subseteq (\mathbb{F}_q[X]/\langle X^m - 1 \rangle)^\ell$$

es generado por r elementos como un $\mathbb{F}_q[X]/\langle X^m - 1\rangle$-módulo, pero no puede ser generado por menos de r elementos. Si C es un código cíclico, entonces se tendrá que $r = 1$ y se encuentra el resultado clásico relacionado con los códigos cíclicos.

Definición 3.6 (Polinomio generador)**.** El polinomio generador $g(X)$ del Corolario 3.1 es llamado un polinomio generador de C.

Ejemplo 3.1. Consideremos $\mathbb{F}_4 = \mathbb{F}_2[\omega]$ e $I = \langle P(X), Q(X)\rangle \subset M_3(\mathbb{F}_4)[X]/\langle X^5 - 1\rangle$ un ideal a izquierda, donde

$$P(X) = \begin{pmatrix} \omega & 0 & 1 \\ \omega & \omega & 0 \\ \omega^2 & \omega^2 & 0 \end{pmatrix} X^4 + \begin{pmatrix} \omega & \omega^2 & \omega^2 \\ 0 & \omega & 1 \\ \omega^2 & 0 & \omega \end{pmatrix} X^3 + \begin{pmatrix} 0 & \omega^2 & \omega \\ \omega & \omega^2 & \omega^2 \\ 0 & 1 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & 0 & \omega^2 \\ 0 & \omega & 1 \\ 0 & \omega & 1 \end{pmatrix} X + \begin{pmatrix} 1 & 0 & \omega^2 \\ 0 & 1 & \omega^2 \\ 0 & 0 & 0 \end{pmatrix}$$

y

$$Q(X) = \begin{pmatrix} \omega & 0 & 1 \\ \omega & \omega & 0 \\ \omega^2 & \omega^2 & 0 \end{pmatrix} X^4 + \begin{pmatrix} 0 & 1 & \omega^2 \\ 0 & 1 & \omega^2 \\ 0 & \omega & 1 \end{pmatrix} X^3 + \begin{pmatrix} \omega & \omega^2 & \omega^2 \\ 1 & \omega & \omega \\ \omega & \omega^2 & \omega^2 \end{pmatrix} X^2 + \begin{pmatrix} 1 & \omega^2 & 1 \\ 0 & \omega^2 & \omega \\ 0 & 1 & \omega^2 \end{pmatrix} X + \begin{pmatrix} 1 & 1 & 0 \\ \omega & \omega^2 & \omega^2 \\ \omega^2 & 1 & 1 \end{pmatrix}.$$

La forma escalonada de una matriz generadora del código 3-cuasi-cíclico C_1 asociado al ideal a izquierda I es

$$G = \left(\begin{array}{ccc|ccc|ccc|ccc|ccc} 1 & 0 & \omega^2 & 0 & 0 & 0 & 0 & \omega^2 & \omega & \omega & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & \omega^2 & 0 & 0 & 0 & 0 & 0 & 0 & \omega & \omega & 0 & 1 & 0 & \omega^2 \\ \hline 0 & 0 & 0 & 1 & 0 & \omega^2 & 0 & 0 & 0 & 0 & \omega^2 & \omega & \omega & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & \omega^2 & 0 & \omega^2 & \omega & \omega & 0 & 1 & \omega & \omega & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & \omega^2 & 0 & \omega & 0 & \omega^2 & \omega \end{array}\right).$$

En este punto hay que tener en cuenta que dependiendo del orden en que se considere los elementos que pertenecen a uno de los conjuntos `V` en el algorítmo 3.4, que corresponde a los vectores g_i para los cuales $\mathcal{F}(g_i)$ es el mismo, la matriz resultante de aplicar dicho algoritmo puede ser una u otra, esto se puede evidenciar al contrastar los resultados presentados en [5] con los resultados aquí obtenidos mediante el uso del software SAGE.

En ese orden de ideas, los resultados obtenidos en la plataforma SAGE, indicaron que los vectores (filas) g_i de G que son linealmente independientes y que se conservaron para formar la matriz $G0$, son g_3 y g_5. Luego, siguiendo las ideas planteadas en el Corolario 3.1 se tendrá que

$$G_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

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

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

$$G_3 = \begin{pmatrix} 0 & \omega^2 & \omega \\ \omega^2 & 0 & \omega \\ 0 & 0 & 0 \end{pmatrix},$$

$$G_4 = \begin{pmatrix} \omega & 0 & 1 \\ 0 & \omega^2 & \omega \\ 0 & 0 & 0 \end{pmatrix}.$$

Siguiendo lo descrito en el corolario 3.1, para formar el polinomio generador del código C, se debe tener en cuenta $v \in \mathbb{Z}$, tal que todos los G_i sean diferentes de 0, el cual es 1 en este caso, así, el resultado buscado es el siguiente

$$g(X) = \begin{pmatrix} 1 & 0 & \omega^2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} X + \begin{pmatrix} 0 & \omega^2 & \omega \\ \omega^2 & 0 & \omega \\ 0 & 0 & 0 \end{pmatrix} X^2 + \begin{pmatrix} \omega & 0 & 1 \\ 0 & \omega^2 & \omega \\ 0 & 0 & 0 \end{pmatrix} X^3.$$

que es un polinomio generador de C_I e $I = \langle P(X), Q(X) \rangle = \langle g(X) \rangle$.

Una manera de comprobar que el código 3-cuasi-cíclico generado por el polinomio obtenido a partir del algoritmo implementado en el software SAGE y el código 3-cuasi-cíclico que se obtiene por el polinomio descrito en [5], es realizando la diferencia simétrica entre los dos conjuntos en mención. Para ello, nuevamente se hizo uso de SAGE y se determinó que la diferencia simétrica entre éstos es $\emptyset$. A continuación se describe la manera de llevar a cabo dicho procedimiento con el programa en mención.

En principio es necesario definir la extensión finita en la que se trabajará, que para este caso es $GF(4)$. El comando empleado es el siguiente.

```
F4.<w>=GF(2^2).
```

Con el comando siguiente, se genera el espacio matricial sobre $GF(4)$, con matrices de dimensión 5×15.

```
G = MatrixSpace(F4,5,15).
```

A continuación se muestra cómo se define la matriz J, la cual es un elemento de G.

```
J=G([1, 0, w^2, 0,0,0,0, w^2,w,w,0,1,0,0,0,
0,1,w^2,0,0,0,0,0,0,w,w,0,1,0,w^2,
0,0,0,1,0,w^2,0,0,0,0,w^2,w,w,0,1,
0,0,0,0,1,w^2,0,w^2,w,w,0,1,w,w,0,
0,0,0,0,0,0,1,1,0,w^2,0,w,0,w^2,w]).
```

Para lograr que ésta se muestre en pantalla, simplemente se introduce el nombre de la matriz (`J`) en una nueva fila de trabajo dentro de SAGE.

Con el comando siguiente se genera el código C a partir de la matriz J.

```
C  = LinearCode(J).
```

Posteriormente, para generar el segundo código (`C1`), se introduce la matriz `J1` que será quien lo genere, definida esta sobre la misma extensión finita $GF(4)$. Ya con los dos códigos generados, C y $C1$, lo siguiente es determinar su diferencia simétrica, mediante el siguiente comando

```
Y = Set(C2).symmetric_difference(Set(C)).
```

El resultado en pantalla se obtiene luego de pedir al programa que muestre el conjunto `Y`, evidenciando que efectivamente $Y = \emptyset$.

Con este resultado, se pudo conjeturar que sin importar el orden en que se consideren los elementos dentro del conjutno `V` del algoritmo, éste es capaz de permitir formar polinomios diferentes, que pueden generar el mismo código ℓ-cuasi-cíclico.

3.1.2. Una propiedad de los polinomios generadores

La siguiente proposición generaliza el Lema 2.2.

Proposición 3.1. Si C es un código cuasi-cíclico, entonces, el código dual $C^{\perp}$ es también un código cuasi-cíclico.

Demostración. Sea C un código cuasi-cíclico y $\mathbf{x} = x_0x_1\cdots x_{n-1} \in C^\perp$. Para cualquier $\mathbf{c} = c_0c_1\cdots c_{n-1} \in C$, se tiene que $T^\ell(\mathbf{c}) = \mathbf{c}' = c_{n-\ell}c_{n-\ell+1}\cdots c_{n-\ell-2}c_{n-\ell-1}$ también está en C y en consecuencia

$$\begin{aligned} 0 = \mathbf{c}'\cdot\mathbf{x} &= c_{n-\ell}x_0 + c_{n-\ell+1}x_1 + \cdots + c_{n-1}x_\ell + c_0x_{\ell+1} + \cdots + c_{n-\ell-1}x_{n-1} \\ &= c_0x_{\ell+1} + \cdots + c_{n-\ell-1}x_{n-1} + c_{n-\ell}x_0 + \cdots + c_{n-1}x_\ell. \end{aligned}$$

Ahora, si se denota como $\mathbf{x}' = x_{\ell+1} + \cdots + x_{n-1} + x_0 + \cdots + x_\ell$, 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 cuasi-cíclico. □

Proposición 3.2. Sea C un código ℓ-cuasi-cíclico de longitud $m\ell$ sobre $\mathbb{F}_q$. Sea $P(X)$ un polinomio generador de C y $Q(X)$ un polinomio generador de su código dual, entonces

$$P(X)({}^tQ^\star(X)) \equiv 0 \text{ mód } (X^m - 1),$$

donde $Q^\star(X) = X^{\mathrm{grad}(Q)}Q(1/X)$ denota el polinomio recíproco de Q, y ${}^tQ$ es el polinomio cuyos coeficientes son las matrices transpuestas de los coeficientes de Q.

Demostración. Dado que $P(X) = \sum_{i=0}^{m-1} P_iX^i$ es un polinomio generador de C, las filas de la matriz

$$\begin{pmatrix} P_0 & P_1 & \cdots & P_{m-1} \end{pmatrix}$$

y sus shifts generan a C. De forma análoga, para $Q(X) = \sum_{i=0}^{m-1} Q_iX^i$ se tiene que las filas de

$$\begin{pmatrix} Q_0 & Q_1 & \cdots & Q_{m-1} \end{pmatrix}$$

y sus shifts generan a $C^\perp$. Por la definición de código dual, se sigue que

$$\begin{pmatrix} P_0 & P_1 & \cdots & P_{m-1} \end{pmatrix} \begin{pmatrix} {}^tQ_0 \\ {}^tQ_1 \\ \vdots \\ {}^tQ_{m-1} \end{pmatrix} = \sum_{i=0}^{m-1} P_i({}^tQ_i) = 0.$$

Como C y $C^\perp$ son códigos ℓ-cuasi-cíclicos, se tienen

$$\sum_{i=0}^{m-1} P_i({}^tQ_{i+j \text{ mód } m}) = 0$$

para todo $j \in \mathbb{Z}$. Por tanto

$$P(X)({}^tQ^\star(X)) = \sum_{j=0}^{m-1}\sum_{i=0}^{m-1} P_i({}^tQ_{i-jmodm})X^j = 0 \text{ mód } (X^m - 1).$$

Que es de donde resulta esta proposición. □

Conclusiones

- En este trabajo se describe de manera detallada el contenido teórico relacionado con los procesos de codificación y decodificación con códigos cíclicos, caracterizado por tener unas sólidas bases en el campo del álgebra y potenciado por la conexión existente con la teoría de los campos finitos y los anillos de polinomios construidos a partir de ellos.
- El manejo minucioso que se le ha dado a las demostraciones relacionadas con las temáticas principales aquí tratadas, dan la posibilidad de que el trabajo se convierta en un punto de apoyo para el lector que comienza el estudio de la teoría de códigos cíclicos.
- Las estructuras algebraicas que han ido surgiendo con el desarrollo de la teoría de códigos revelan contar con unos componentes teóricos más fuertes y con una complejidad creciente, producto de tratar de suplir muchas de las necesidades que han aparecido en el camino, entre ellas la de optimizar los procesos de codificación y decodificación, la de mejorar la capacidad de corrección de errores, la de establecer unas distancias mínimas adecuadas o simplemente por la variación de ciertos conceptos.
- El paquete informático SAGE, empleado como herramienta para la exploración y verificación de varios de los resultados que se han presentado en este trabajo, demostró ser un elemento con un gran potencial no solo en actividades investigativas, sino también en el ámbito pedagógico, lo que puede dar pie a la realización de trabajos de profundización posteriores apoyados por este software, ya que en la búsqueda de documentación relacionada con este tema se constató que la bibliografía existente es bastante reducida o no esta lo suficientemente organizada en ese sentido.
- En el capítulo 3 se presentan los aspectos iniciales estudiados en [5], permitiendo que interesados en este tema lo puedan tomar como un paso inicial hacia el estudio de los códigos cuasi-cíclicos como generalización de los códigos cíclicos. Cabe resaltar que en este trabajo no se abordaron todos los temas que aparecen en el citado artículo, por lo que futuros trabajos podrían dedicarse a esta labor. La lectura del artículo [5] muestra que los códigos cuasi-cíclicos son un campo de estudio que está en desarrollo. Adicionalmente, en la literatura se pueden encontrar otros trabajos donde se aborda este tema, ver [11, 20–23].

Referencias

[1] SAGE mathematics software. Versión 7.0. http://www.sagemath.org, 2016-10-16. Stein, William. et al. The Sage Development Team.

[2] I. T. Adamson. *Introduction to field theory*. Cambridge University Press, Cambridge-New York, second edition, 1981.

[3] N. Aydin and T. A. Gulliver. Some good cyclic and quasi-twisted $\mathbb{Z}_4$-linear codes. *Ars Comb.*, 99:503–518, 2011.

[4] N. Aydin and D. K. Ray-Chaudhuri. Quasi-cyclic codes over $\mathbb{Z}_4$ and some new binary codes. *IEEE Transactions on Information Theory*, 48(7):2065–2069, 2002.

[5] M. Barbier, C. Chabot, and G. Quintin. On quasi-cyclic codes as a generalization of cyclic codes. *Finite Fields Appl.*, 18(5):904–919, 2012.

[6] S. D. Berman. Semisimple cyclic and Abelian codes. II. *Cybernetics*, 3(3):17–23 (1970), 1967.

[7] S. D. Berman. On the theory of group codes. *Cybernetics*, 3(1):25–31 (1969), 1969.

[8] D. Boucher, W. Geiselmann, and F. Ulmer. Skew-cyclic codes. *Applicable Algebra in Engineering, Communication and Computing*, 18(4):379–389, 2007.

[9] D. Boucher and F. Ulmer. Coding with skew polynomial rings. *J. Symbolic Comput.*, 44(12):1644–1656, 2009.

[10] P. Carron. *Morse Code: The Essential Language*. Radio amateur's library, Estados Unidos, 2 edition, 1996.

[11] P.-L. Cayrel, C. Chabot, and A. Necer. Quasi-cyclic codes as codes over rings of matrices. *Finite Fields Appl.*, 16(2):100–115, 2010.

[12] V. Drensky and P. Lakatos. Monomial ideals, group algebras and error correcting codes. In *Applied algebra, algebraic algorithms and error-correcting codes (Rome, 1988)*, volume 357 of *Lecture Notes in Comput. Sci.*, pages 181–188. Springer, Berlin, 1989.

[13] R. A. Ferraz and C. Polcino Milies. Idempotents in group algebras and minimal abelian codes. *Finite Fields Appl.*, 13(2):382–393, 2007.

[14] C. M. Gómez. Códigos correctores de errores (o cuántas preguntas son necesarias para conocer un número). *Gaceta de la Real Sociedad Matemática Española*, 6(3):713–731, 2003.

[15] W. C. Huffman and V. Pless. *Fundamentals of error-correcting codes*. Cambridge University Press, Cambridge, 2003.

[16] S. Jitman, S. Ling, H. Liu, and X. Xie. Abelian codes in principal ideal group algebras. *IEEE Trans. Inform. Theory*, 59(5):3046–3058, 2013.

[17] P. Landrock and O. Manz. Classical codes as ideals in group algebras. *Des. Codes Cryptogr.*, 2(3):273–285, 1992.

[18] S. Lang. *Algebra*. Springer- Verlag, New York, 4 edition, 2000.

[19] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, Cambridge, 1986.

[20] S. Ling, H. Niederreiter, and P. Solé. On the algebraic structure of quasi-cyclic codes. IV. Repeated roots. *Des. Codes Cryptogr.*, 38(3):337–361, 2006.

[21] S. Ling and P. Solé. On the algebraic structure of quasi-cyclic codes. I. Finite fields. *IEEE Trans. Inform. Theory*, 47(7):2751–2760, 2001.

[22] S. Ling and P. Solé. On the algebraic structure of quasi-cyclic codes. II. Chain rings. *Des. Codes Cryptogr.*, 30(1):113–130, 2003.

[23] S. Ling and P. Solé. On the algebraic structure of quasi-cyclic codes. III. Generator theory. *IEEE Trans. Inform. Theory*, 51(7):2692–2700, 2005.

[24] J. MacWilliams. Codes and ideals in group algebras. In *Combinatorial Mathematics and its Applications (Proc. Conf., Univ. North Carolina, Chapel Hill, N.C., 1967)*, pages 317–328. Univ. North Carolina Press, Chapel Hill, N.C., 1969.

[25] R. Podestá. Introducción a la teoría de códigos autocorrectores. Notas del curso dado en el ENA III, Vaquerías, 8/2006. Trabajos de Matemática de FaMAF, Serie C 2006/35, 50 págs, Universidad Nacional de Córdoba, Argentina, 2006.

[26] S. Roman. *Coding and information theory*, volume 134 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992.

[27] C. X. San Ling. *Coding Theory: A First Course*. Cambridge University Press, 2004.

[28] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[29] P. Solé, editor. *Codes over rings*, volume 6 of *Series on Coding Theory and Cryptology*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2009.