UNIVERSIDAD
DE GRANADA

E.T.S. de Ingenierías Informática y de Telecomunicación
Facultad de Ciencias

DOBLE GRADO EN INGENIERÍA
INFORMÁTICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados

Presentado por:
José María Martín Luque

Tutor:
Gabriel Navarro Garulo

Curso académico 2019–2020

Algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados

LICENCIA

1 Texto completo de la licencia disponible en https://creativecommons.org/licenses/by-sa/4.0/legalcode.

A mi abuela Nieves,
quien me acogió con cariño en Granada,
y que, tristemente, ya no me recuerda.

RESUMEN

Un código es una estructura algebraica con la que es posible transmitir información de forma que el receptor pueda corregir los errores que se hayan podido producir durante la transmisión. El objetivo principal de este trabajo es presentar, estudiar e implementar el algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados. Esta clase de códigos es relevante por la gran cantidad de códigos que pueden diseñarse, de forma que es posible encontrar algunos con propiedades interesantes. El algoritmo mencionado proporciona un método de decodificación eficiente para ellos. Para abordar nuestro estudio es necesaria una ingente cantidad de conocimiento previo, cuya exposición ocupa la mayor parte de este trabajo. Nos centramos primero en los fundamentos necesarios de álgebra —teoría de anillos y cuerpos finitos— y de teoría de códigos —códigos lineales y códigos cíclicos—. Estudiamos además la versión original del algoritmo PGZ para ayudarnos en la comprensión del algoritmo para códigos cíclicos sesgados. Posteriormente abordamos los anillos de polinomios de Ore, que constituyen la base de los códigos cíclicos sesgados. Una vez explicada esta clase de códigos, estaremos en disposición de estudiar e implementar el algoritmo prometido: Peterson-Gorenstein-Zierler para códigos cíclicos sesgados.

PALABRAS CLAVE teoría de códigos códigos cíclicos polinomios de Ore SageMath Peterson-Gorenstein-Zierler

SUMMARY

The main objetive of this project is to present, study and implement the Peterson-Gorenstein-Zierler algorithm for skew cyclic codes. But, in order to do so, we need a seizable amount of knowledge, and that is precisely what we will cover in the first chapters of this project.

CHAPTER 1 This chapter will set the foundations of the mathematic knowledge that is needed to understand the latter parts of the project. Skew cyclic codes require knowledge of two concepts to be understood: Ore polynomial rings and cyclic codes. Then, Ore polynomial rings are dependant on the theory of rings and field automorphisms, and cyclic codes are dependant on the theory of finite fields and rings. We will tackle in this chapter all the concepts and relevant results of these areas.

CHAPTER 2 In this chapter, we will introduce the concept of a code and the most studied class of codes: linear codes. We will also explain that linear codes over a finite field are a vector subspace, and therefore they are a mathematical structure that is well known and easy to study. We will also use this chapter to introduce some concepts that we will refer to throughout the project and that will be fundamental to the further development of coding theory. To that extent, we will present the definition of a generator matrix: a base of the vector subspace that a linear code ultimately is. As we shall see, this matrix is the basis of the most common procedure of message encoding. Another important concept we will introduce is the distance of a code, that is, the minimum number of coordinates that differentiate one codeword —an element of a code— from another. Finally, some simple families of codes will be presented, such as repetition codes or Hamming codes.

CHAPTER 3 In this chapter, we will shift our focus to the class of cyclic codes. As the name may suggest, these codes have the property that cyclic shifts of codewords in a code are also codewords of said code. We will see that codewords of these kinds of codes can be mapped to certain polynomials, and cyclic codes are simply the ideals of a polynomial ring quotient over the ideal generated by the polynomial $x^n - 1$ for some n. That is why we will dedicate the first part of this chapter to the study of the factorization of this polynomial on polynomial rings over finite fields. Once we know how to do this, we can properly describe all cyclic codes of any length. We will also

show two methods for encoding messages in cyclic codes. Subsequently, an alternative way of describing cyclic codes is explained, using what we will call generating idempotents: elements whose product by themselves results in the same element. Finally, we will explain the concept of zeroes of cyclic codes, a notion we will see is fundamental in the definition of BCH codes in next chapter.

CHAPTER 4 In this chapter, we will describe the family of BCH codes and the original version of the Peterson-Gorenstein-Zierler algorithm for BCH codes. We will also take the opportunity to briefly introduce the family of RS codes, as we will refer to them in subsequent chapters. To properly introduce BCH codes, we will first explain the BCH bound, a result that links the concept of zeroes of a cyclic code and the minimum distance of said code. We will see that BCH codes are defined to take advantage of the BCH bound. This means that it is possible to design BCH codes with any error correction capability, although their length will vary accordingly. To finish this chapter we will take on the Peterson-Gorenstein-Zierler algorithm for BCH codes: we will explain the decoding procedure and prove that it succesfully corrects errors.

CHAPTER 5 In this chapter, Ore polynomial rings will be discussed. Since in coding theory we will be working with finite fields we will limit our study to Ore polynomial rings over finite fields. We will introduce the main concepts as well as algorithms to calculate division from left or right, as well as an extented Euclid algorithm. We will also elaborate on the fact that factorization of this kind of polynomials is not unique in the common sense, but we will introduce a concept that is similar in practice.

CHAPTER 6 In this chapter, we will describe the class of skew cyclic codes. Throughout the chapter we will set the basis for the introduction of skew RS codes at the end. This is the family of codes that we will be using with the algorithm that will be explained in the following chapter.

CHAPTER 7 Finally, in this chapter, we will introduce the algorithm that is the main objective of this project: the Peterson-Gorenstein-Zierler algorithm for skew cyclic codes. As we did with the analogous algorithm for BCH codes, we will explain the decoding procedure as well as prove that it succesfully corrects errors.

As part of this project we have developed various classes for SageMath.

— A decoder class for the BCH codes implementation of SageMath that uses the Peterson-Gorenstein-Zierler algorithm described here.

— A skeleton class for skew cyclic codes. We have not developed methods for this class but rather left a framework for other classes to be inherited by this one.
— A skew RS code class that implements the definition we study in this project, as well as a simple encoder class for them.
— Finally, a decoder class for the skew RS codes that uses the Peterson-Gorenstein-Zierler algorithm for skew cyclic codes that is the main goal of this project.

These classes allow us to work with the structures in SageMath that are already implemented. The documentation for these classes can be found in annex A. Throughout the project we will present some examples for the concepts we explain, using SageMath either with the pre-existing classes or the classes we created. We have also used some other helpful functions that we have described in annex B.

KEYWORDS coding theory cyclic codes Ore polynomials SageMath Peterson-Gorenstein-Zierler

ÍNDICE GENERAL

1 PRELIMINARES 17
1.1 Anillos . . . 17
1.2 Cuerpos finitos . . . 20
1.2.1 Anillos de polinomios sobre cuerpos finitos . . . 21
1.2.2 Construcción de cuerpos finitos . . . 23
1.2.3 Elementos primitivos . . . 25
1.2.4 Clases ciclotómicas y polinomios minimales . . . 27
1.3 Automorfismos de cuerpos finitos . . . 30
2 FUNDAMENTOS DE TEORÍA DE CÓDIGOS 31
2.1 Códigos lineales . . . 31
2.1.1 Codificación y decodificación . . . 33
2.1.2 Distancias y pesos . . . 36
2.2 Ejemplos de códigos . . . 40
2.2.1 Códigos de repetición . . . 40
2.2.2 Códigos de control de paridad . . . 40
2.2.3 Códigos de Hamming . . . 40
3 CÓDIGOS CÍCLICOS 43
3.1 Factorización de $x^n - 1$ . . . 44
3.2 Construcción de códigos cíclicos . . . 47
3.3 Codificación de códigos cíclicos . . . 54
3.4 Idempotentes y multiplicadores . . . 55
3.5 Ceros y conjuntos característicos . . . 62
4 CÓDIGOS BCH 65
4.1 Construcción de códigos BCH . . . 65
4.2 Códigos Reed-Solomon . . . 67
4.3 Algoritmo de Peterson-Gorenstein-Zierler . . . 68
5 ANILLOS DE POLINOMIOS DE ORE 75
5.1 Anillos de polinomios de Ore sobre cuerpos finitos . . . 76
5.2 División . . . 77
5.3 Mínimo común múltiplo y máximo común divisor . . . 78
5.4 Descomposición en factores irreducibles . . . 79
5.5 Norma . . . 80
6 CÓDIGOS CÍCLICOS SESGADOS 83
7 ALGORITMO PGZ PARA CÓDIGOS CÍCLICOS SESGADOS 89
A IMPLEMENTACIÓN EN SAGEMATH DEL ALGORITMO PGZ 105
A.1 Decodificador basado en el algoritmo PGZ para códigos BCH . . . 105

A.2 Clase para códigos cíclicos sesgados 107
A.3 Codificadores para códigos cíclicos sesgados 109
A.4 Clase para códigos RS sesgados 113
A.5 Decodificador basado en el algoritmo PGZ para códigos RS sesgados 114
A.6 Funciones auxiliares 115
B FUNCIONES EN SAGEMATH USADAS EN LOS EJEMPLOS 117

INTRODUCCIÓN

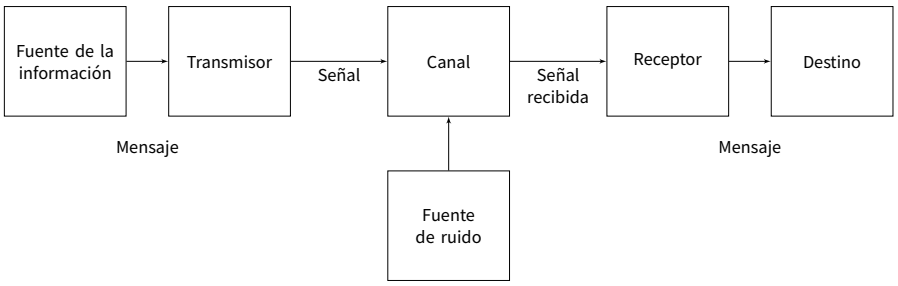
El artículo de Claude Shannon *Mathematical Theory of Cryptography* (Shannon, 1945) y su ampliación posterior, *Mathematical Theory of Communication* (Shannon, 1948) dieron a luz a dos disciplinas hoy plenamente establecidas: la teoría de información y la teoría de códigos. El objetivo principal de estas disciplinas es el de establecer mecanismos de comunicación que sean *eficientes* y *fiables* en ambientes posiblemente hostiles. La eficiencia requiere que la transmisión de la información no necesite de demasiados recursos, sean materiales o temporales. Por otro lado, la fiabilidad requiere que el mensaje recibido en una comunicación sea lo más parecido posible, dentro de unos márgenes de tolerancia, al mensaje original. La teoría de la información se encarga del estudio tanto de la representación de la información como de la capacidad que tienen los sistemas para transmitir y procesar la información. Por otra parte, la teoría de códigos se basa en los resultados de la teoría de la información para el diseño y desarrollo de modelos de transmisión de información mediante herramientas algebraicas.

A pesar de que digamos que Shannon fue el padre de estas disciplinas, el problema de codificar la información no surge, ni mucho menos, entonces. De hecho, el filósofo inglés Francis Bacon ya afirmó en el año 1623 que únicamente son necesarios dos símbolos para codificar toda la comunicación.

> La transposición de dos letras en cinco emplazamientos bastará para dar 32 diferencias [y] por este arte se abre un camino por el que un hombre puede expresar y señalar las intenciones de su mente, a un lugar situado a cualquier distancia, mediante objetos ... capaces solo de una doble diferencia. (Dyson, 2015, p. 30)

Bacon estaba en lo cierto, y así hoy en día la codificación binaria forma parte de prácticamente todos los aspectos nuestras vidas.

Sin embargo, el problema más relevante que trata la teoría de códigos —y el que nos ocupa— es el de codificar la información para que se produzca una correcta transmisión y recepción de los mensajes a través de un canal. Como parte de su teoría Shannon introdujo lo que posteriormente se conocería como *modelo de comunicación de Shannon*, un esquema simplificado de cómo se produce la transmisión de información entre emisor y receptor a través de un canal. Al enviar la información a través del canal es posible que, debido al ruido que pueda haber en el mismo, se produzcan interferencias que alteren el mensaje enviado. Por supuesto recibir un mensaje alterado no



Modelo de comunicación de Shannon

es deseable, pues es posible que la alteración sea tal que el mensaje recibido sea ilegible. La teoría de códigos trata por tanto de diseñar estructuras algebraicas que permitan la corrección de errores por parte del receptor del mensaje. Estas estructuras son los códigos, y son el objeto matemático sobre el que orbita todo este trabajo.

ENFOQUE

El título de este trabajo es «Algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados», y como tal, expresa el objetivo último del mismo: exponer y estudiar este algoritmo. En consecuencia, dirigiremos nuestra atención a los conceptos y herramientas necesarios para ello. En cuanto a los fundamentos matemáticos necesarios, hablaremos de cuerpos finitos, anillos de polinomios sobre cuerpos finitos y anillos de polinomios de Ore sobre cuerpos finitos. La teoría de códigos explicada se centrará en la exposición de los códigos lineales y de los códigos cíclicos, así como de las familias de códigos concretas sobre las que trabajarán los algoritmos que vamos a describir. Para comprender mejor el citado algoritmo estudiaremos también la versión original, de mitad del siglo pasado, para los conocidos como códigos BCH. A lo largo del trabajo ilustraremos algunos ejemplos con el sistema algebraico computacional SageMath (The Sage Developers, 2020).

MOTIVACIÓN

Los códigos cíclicos son muy utilizados porque son muy sencillos de implementar en sistemas digitales utilizando lo que se conoce como registros de desplazamiento, que permiten realizar desplazamientos de bits fácilmente. El algoritmo Peterson-Gorenstein-Zierler para códigos BCH es interesante porque es uno de los primeros algoritmos eficientes que se desarrollaron para códigos cíclicos. El uso de anillos de polinomios de Ore nos permite obtener una mayor cantidad de códigos cíclicos sesgados, simplemente variando el automorfismo utilizado. Esta gran variedad permite encontrar códigos

que tengan propiedades interesantes que mejoren a las familias de códigos cíclicos conocidas. Finalmente, el algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados que se expone permite decodificarlos con una eficiencia igual o superior a la que presenta el algoritmo clásico para códigos BCH.

OBJETIVOS

Los objetivos de este trabajo son los siguientes.

— Estudio de las nociones básicas sobre Teoría de Códigos lineales.
— Estudio de las extensiones de Ore y de sus cocientes.
— Exponer el algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados.
— Implementación de sistemas de decodificación en Python usando SageMath.

ESQUEMA

En los tres primeros capítulos introduciremos todos los fundamentos matemáticos necesarios, así como las definiciones y propiedades fundamentales de los códigos lineales y de los códigos cíclicos. El capítulo cuarto ahondaremos en la familia de los códigos BCH y presentaremos el algoritmo original de Peterson-Gorenstein-Zierler para esta familia de códigos. En el capítulo quinto explicamos las estructuras matemáticas que constituyen la base de los códigos cíclicos sesgados, los anillos de polinomios de Ore, para ya introducirlos propiamente en el capítulo sexto. Finalmente, en el capítulo séptimo describimos el funcionamiento del algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados.

1

PRELIMINARES

El objetivo de este capítulo es el de presentar todas las herramientas matemáticas que vamos a necesitar para el desarrollo de la teoría de códigos que requerimos en este trabajo. Dicha teoría se sustenta fundamentalmente en la teoría de cuerpos finitos, de anillos y de polinomios. Los resultados presentados son ampliamente conocidos por lo que en general no se aportarán demostraciones de los mismos, que pueden encontrarse con facilidad en libros especializados. Las fuentes principales de este capítulo son (Cohn, 1982, cap. 3 y 6), (Cohn, 1989, cap. 3) y (Lidl & Niederreiter, 1986, cap. 2).

I.I ANILLOS

Por anillo entendemos un conjunto R junto a dos operaciones binarias: $x+y$, llamada *suma*, y xy, llamada *producto*, tales que:

1. R es un grupo abeliano con la suma.
2. R es un monoide con el producto.
3. La suma y el producto están relacionadas mediante la propiedad distributiva:
$$(x+y)z = xz + yz, \qquad x(y+z) = xy + xz.$$

El elemento neutro para la suma se llama *cero* y se escribe 0, mientras que el elemento neutro para el producto se llama *uno* o *la unidad* y se escribe 1. El inverso de la suma para x se denota $-x$.

A continuación vamos a dar una definición completa de anillo, sin depender de remisiones a las definiciones de grupo y monoide.

DEFINICIÓN I.I.I. Un *anillo* es un conjunto junto a dos operaciones: la suma (+) y la multiplicación (·), que verifican las siguientes propiedades.

— Propiedad asociativa:
$$(x+y)+z = x+(y+z), \qquad (xy)z = x(yz).$$

— Propiedad conmutativa para la suma:
$$x+y = y+x.$$

— Existencia de elemento neutro:

$$x + 0 = x, \qquad x1 = x.$$

— Existencia de elemento inverso para la suma:

$$x + (-x) = 0.$$

— Propiedad distributiva para la multiplicación sobre la suma:

$$x(y + z) = xy + xz.$$

Si un anillo verifica la propiedad conmutativa para la multiplicación, es decir, $xy = yx$, se dice que es un *anillo conmutativo*.

Comprobamos que en cualquier anillo R se verifica que $0x = x0 = 0$ para todo $x \in R$, ya que $x0 = x(0 + 0) = x0 + x0$, de donde concluimos que $x0 = 0$ e igualmente, $0x = 0$. Cuando un anillo R tiene solo un elemento es necesario que $1 = 0$. Un anillo de este tipo se denomina *anillo trivial*. Podemos ver que este es el único caso en el que se da la igualdad $1 = 0$. Supongamos que en cualquier otro anillo se da que $1 = 0$. Entonces para cada elemento x del anillo se tiene que $x = x1 = x0 = 0$, luego tiene un único elemento.

Dos anillos son *isomorfos* si hay un *isomorfismo* entre ellos, es decir, existe una biyección que preserva todas las operaciones. Un elemento a de un anillo se dice que es *invertible* si existe un elemento a' en el anillo tal que $aa' = a'a = 1$. A este elemento, que es único, lo llamamos *elemento inverso* de a y lo denotamos por a^{-1}. El elemento 0 no puede tener inverso porque ya hemos visto que siempre que se multiplique por él se obtiene el 0. Los anillos en los que todo elemento distinto de 0 es invertible se llaman *anillos de división*.

DEFINICIÓN I.1.2. Un subconjunto S de un anillo R se denomina *subanillo* si S contiene a los elementos neutros de la suma y el producto de R, es cerrado bajo dichas operaciones y forma un anillo con ellas.

EJEMPLO I.1.3. Sea R el cuerpo $\mathbb{Q}$ de los racionales. Entonces, el subconjunto $\mathbb{Z}$ de los enteros es un subanillo, pues contiene a los elementos neutros de producto y suma —1 y 0, respectivamente—, la suma de dos enteros es un entero y el producto de dos enteros es, de nuevo, un entero.

DEFINICIÓN I.1.4. Sea J un subconjunto de un anillo R.

— Se dice que J es un *ideal por la izquierda* si es un subgrupo aditivo de R y verifica que para todo $j \in J$ y para todo $r \in R$, el elemento $rj \in J$.

— De igual forma, se dice que J es un *ideal por la derecha* si es un subgrupo aditivo de R y verifica que para todo $j \in J$ y para todo $r \in R$, el elemento $jr \in J$.
— Finalmente, se dice que J es un *ideal bilátero* si es ideal tanto por la izquierda como por la derecha.

Notamos que en un anillo conmutativo solo habrá ideales biláteros y por tanto nos referiremos a ellos como ideales a secas. Veamos a continuación un par de ejemplos.

EJEMPLO 1.1.5. Continuando el ejemplo 1.1.3, el conjunto $\mathbb{Z}$ de los enteros no es un ideal pues, por ejemplo, $1 \in \mathbb{Z}$, $1/2 \in \mathbb{Q}$, pero $1/2 \cdot 1 = 1/2 \notin \mathbb{Z}$.

EJEMPLO 1.1.6. Sea R el anillo de los enteros $\mathbb{Z}$ y consideremos el subconjunto de los números enteros $J = \{2n : n \in \mathbb{Z}\}$. Entonces J es un ideal (bilátero), pues todo producto de un número par es otro número par.

PROPOSICIÓN 1.1.7. *Sea J un ideal de un anillo R. Si $1 \in J$ entonces $J = R$.*

DEFINICIÓN 1.1.8. Sea R un anillo. Un ideal por la izquierda J de R se dice que es *principal* si existe un elemento $a \in R$ tal que $J = Ra = \{ra : r \in R\}$. De forma análoga, un ideal por la derecha J de R es *principal* si existe un elemento $a \in R$ tal que $J = aR = \{ar : r \in R\}$. Un ideal bilátero será principal si verifica cualquiera de las dos condiciones.

Dado un ideal bilátero I de un anillo R es posible definir una relación de equivalencia, dada por $a \sim b$ si y solo si $a - b \in I$. Esta relación nos permite definir el conjunto R/I, de las clases de equivalencia obtenidas a partir de ella. Para un $a \in R$ su clase de equivalencia viene dada por

$$[a] = a + I = \{a + x : x \in I\}.$$

Pero es más, R/I también tiene estructura de anillo y se conoce como *anillo cociente* de R por I, cuyas operaciones —bien definidas— vienen dadas por

$$(a + I) + (b + I) = (a + b) + I \quad \text{y} \quad (a + I)(b + I) = ab + I.$$

A continuación vamos a definir algunos otros conceptos relativos a anillos. Decimos que un ideal J es *minimal* si no existe ningún otro ideal entre $\{0\}$ y J. Podemos clasificar los elementos de un anillo distintos de cero en dos tipos: *divisores de cero* y *regulares*. Tomemos un elemento $a \neq 0$. Si existe $b \neq 0$ tal que ab o ba es cero, entonces a es un elemento *divisor de cero*, y en caso contrario, un elemento *regular*. A partir de esta clasificación podemos hablar de anillos *enteros* —aquellos que no son triviales y no tienen divisores de cero— y de *dominios de integridad* —anillos enteros conmutativos—.

Una propiedad importante de los elementos regulares —y que es tan natural que estamos plenamente acostumbrados a ella— es la llamada ley de cancelación.

PROPOSICIÓN I.1.9. *Si c es un elemento regular de un anillo R entonces para cada $a, b \in R$, tales que, o bien $ca = cb$, o bien $ac = bc$, se tiene que $a = b$.*

Para cerrar esta sección vamos a introducir dos definiciones: una será una propiedad de los anillos, mientras que la otra será una propiedad que verificarán algunos elementos de los anillos.

DEFINICIÓN I.1.10. Sea R un anillo. La *característica* del anillo es el menor natural n tal que $n1 = 0$. Si no existe tal número, la característica del anillo es 0.

DEFINICIÓN I.1.11. Un elemento e de un anillo tal que $e^2 = e$ se dice que es *idempotente*.

I.2 CUERPOS FINITOS

En esta sección vamos a introducir el concepto de cuerpo, junto a otros conceptos relevantes, para posteriormente centrarnos en los cuerpos finitos. Estudiaremos además los anillos de polinomios que podemos definir sobre ellos y cómo a partir del cociente de estos podemos construir cuerpos finitos.

DEFINICIÓN I.2.1. Un *cuerpo* es un anillo de división conmutativo. Se dice que un cuerpo es *finito*[1] si tiene un número finito de elementos, al que llamamos *orden* del cuerpo.

Sea F un cuerpo. Un subconjunto K de F que es por sí mismo un cuerpo bajo las operaciones de F se denomina *subcuerpo* de F. También podemos referirnos a ello en sentido inverso, diciendo que F es una *extensión* de K, lo que notaremos como F/K. Observamos que F es un espacio vectorial sobre K, esto es, los elementos de F pueden ser vistos como vectores sobre el cuerpo de escalares K, con las operaciones de suma $\alpha + \beta$, para $\alpha, \beta \in F$ y multiplicación por escalares $a\alpha$, para $a \in K$ y $\alpha \in F$, dadas por las propias operaciones de suma y multiplicación en K. Todas las nociones que hemos definido para anillos —como la característica— son válidas para los cuerpos, pues un cuerpo no deja de ser un anillo.

Habitualmente notaremos a los cuerpos finitos por $\mathbb{F}_q$, donde q denota el orden del cuerpo.

1 Los cuerpos finitos también se suelen conocer como *cuerpos de Galois* en honor a Évariste Galois, uno de los primeros matemáticos en trabajar con ellos.

PROPOSICIÓN 1.2.2. *Sea $\mathbb{F}_q$ un cuerpo de característica p. Entonces $(a+b)^p = a^p + b^p$ para todo $a, b \in \mathbb{F}_q$.*

TEOREMA 1.2.3. *Todo cuerpo F tiene al menos un subcuerpo P, el subcuerpo primo de F, que está contenido en cada subcuerpo de F. O bien F tiene característica 0 y $P \cong \mathbb{Q}$, o bien F tiene característica p, un número primo, y entonces $P \cong \mathbb{F}_p$.*

LEMA 1.2.4. *Un espacio vectorial n-dimensional sobre $\mathbb{F}_q$ tiene p^n elementos.*

Demostración. Sea V un espacio vectorial n-dimensional sobre $\mathbb{F}_q$. Si los elementos $u_1, \dots, u_n$ forman una base de V, entonces cada elemento de V se escribe de forma única en la forma $\sum \alpha_i u_i$, donde $\alpha_i \in \mathbb{F}_q$. Como cada coeficiente puede tener hasta p valores distintos, obtenemos un total de p^n elementos. □

Concluimos destacando que en virtud del teorema 1.2.3 todo cuerpo finito F tiene característica p (un primo) y su subcuerpo primo es $\mathbb{F}_p$.

1.2.1 *Anillos de polinomios sobre cuerpos finitos*

Para cualquier anillo R podemos definir un anillo de polinomios en x con coeficientes en R, que denotamos $R[x]$ y que está compuesto por el conjunto

$$\{a_0 + a_1 x + \cdots + a_n x^n : a_0, a_1, \dots, a_n \in R\}.$$

Dados $f(x) = a_0 + a_1 x + \cdots + a_n x^n$ y $g(x) = b_0 + b_1 x + \cdots + b_m x^m$ las operaciones de suma y multiplicación de $R[x]$ (suponiendo $m \leq n$) vienen dadas por:

$$f(x) + g(x) = a_0 + b_0 + (a_1 + b_1)x + \cdots + (a_m + b_m)x^m + \cdots + a_n x^n,$$

$$\begin{aligned} f(x)g(x) =& a_0 b_0 + (a_0 b_0 + a_1 b_0)x + (a_0 b_2 + a_1 b_1 + a_2 b_0)x^2 \\ &+ \cdots + a_n b_m x^{m+n}. \end{aligned}$$

El elemento neutro para la suma es el $0 \in R$, y el del producto, el $1 \in R$. Es fácil comprobar que efectivamente $R[x]$ así definido satisface todas las propiedades de los anillos. Veamos a continuación algunos conceptos básicos sobre polinomios. El *grado* de un polinomio es el mayor grado de cualquier término con coeficiente distinto de cero. El coeficiente del término de mayor grado se denomina *coeficiente líder*. Un polinomio es *mónico* si su coeficiente líder es 1. Sean $f(x)$ y $g(x)$ polinomios en $R[x]$. Decimos que $f(x)$ *divide a*

$g(x)$, denotado por $f(x)|g(x)$, si existe un polinomio $h(x) \in \mathbb{R}[x]$ tal que $g(x) = f(x)h(x)$. El polinomio $f(x)$ se llama *divisor* o *factor* de $g(x)$.

Como ya hemos anticipado vamos a centrarnos en el estudio de los anillos de polinomios en cuerpos finitos pues son los que necesitaremos para la teoría de códigos. Siguiendo la notación que hemos establecido, denotaremos el anillo de le los polinomios con coeficientes en $\mathbb{F}_q$ por $\mathbb{F}_q[x]$. Es un anillo conmutativo con las operaciones habituales de suma y multiplicación de polinomios que acabamos de describir. Es, de hecho, un dominio de integridad.

Un polinomio en $\mathbb{F}_q[x]$ viene dado por $f(x) = \sum_{i=0}^{n} a_i x^i$, donde a_i son los coeficientes del término de grado i y pertenecen a $\mathbb{F}_q$. Dados $f(x), g(x) \in \mathbb{F}_q[x]$ el *máximo común divisor* de $f(x)$ y $g(x)$, siendo al menos uno de ellos distinto de cero, es el polinomio mónico de $\mathbb{F}_q[x]$ de mayor grado que divida tanto a $f(x)$ como a $g(x)$. Lo denotamos por $\mathrm{mcd}(f(x), g(x))$. Decimos que dos polinomios son *primos relativos* si su máximo común divisor es 1.

El siguiente resultado nos proporciona la existencia y unicidad de divisores de un polinomio en $\mathbb{F}_q[x]$.

TEOREMA I.2.5. *Sean $f(x)$ y $g(x)$ polinomios de $\mathbb{F}_q[x]$, siendo $g(x)$ distinto de cero.*

1. *Existen polinomios únicos, $h(x)$, $r(x) \in \mathbb{F}_q[x]$ tales que*

$$f(x) = g(x)h(x) + r(x), \quad \textit{donde } \operatorname{gr} r(x) < \operatorname{gr} g(x) \textit{ o } r(x) = 0.$$

2. *Si $f(x) = g(x)h(x) + r(x)$, entonces*

$$\mathrm{mcd}(f(x), g(x)) = \mathrm{mcd}(g(x), r(x)).$$

Podemos utilizar este resultado para hallar el máximo común divisor de dos polinomios. Este procedimiento se conoce como *algoritmo de Euclides* y es muy parecido a su homólogo para números enteros.

TEOREMA I.2.6. (ALGORITMO DE EUCLIDES.) *Sean $f(x)$ y $g(x)$ dos polinomios en $\mathbb{F}_q[x]$ con $g(x)$ distinto de cero.*

1. *Realiza los siguientes pasos hasta que $r_n(x) = 0$ para algún n:*

$$\begin{aligned}
f(x) &= g(x)h_1(x) + r_1(x), && \textit{donde } \operatorname{gr} r_1(x) < \operatorname{gr} g(x),\\
g(x) &= r_1(x)h_2(x) + r_2(x), && \textit{donde } \operatorname{gr} r_2(x) < \operatorname{gr} r_1(x),\\
r_1(x) &= r_2(x)h_3(x) + r_3(x), && \textit{donde } \operatorname{gr} r_2(x) < \operatorname{gr} r_3(x),\\
&\vdots\\
r_{n-3}(x) &= r_{n-2}(x)h_{n-1}(x) + r_{n-1}(x), && \textit{donde } \operatorname{gr} r_{n-1}(x) < \operatorname{gr} r_{n-2}(x),\\
r_{n-2}(x) &= r_{n-1}(x)h_n(x) + r_n(x), && \textit{donde } r_n(x) = 0.
\end{aligned}$$

Entonces, $\mathrm{mcd}(f(x), g(x)) = cr_{n-1}(x)$*, donde* $c \in \mathbb{F}_q$ *se escoge para que* $cr_{n-1}(x)$ *sea mónico.*
2. *Existen polinomios* $a(x), b(x) \in \mathbb{F}_q[x]$ *tales que*

$$a(x)f(x) + b(x)g(x) = \mathrm{mcd}(f(x), g(x)).$$

La secuencia de pasos descrita termina porque en cada paso el grado del resto se reduce al menos en 1. A continuación vamos a comentar un par de resultados que nos serán útiles en el futuro.

PROPOSICIÓN 1.2.7. *Sean* $f(x)$ *y* $g(x)$ *polinomios en* $\mathbb{F}_q[x]$.

1. *Si* $k(x)$ *es un divisor de* $f(x)$ *y de* $g(x)$*, entonces* $k(x)$ *es divisor del polinomio* $a(x)f(x) + b(x)g(x)$ *para todo* $a(x), b(x) \in \mathbb{F}_q[x]$.
2. *Si* $k(x)$ *es un divisor de* $f(x)$ *y de* $g(x)$ *entonces* $k(x)$ *es divisor del polinomio* $\mathrm{mcd}(f(x), g(x))$.

PROPOSICIÓN 1.2.8. *Sea* $f(x)$ *un polinomio en* $\mathbb{F}_q[x]$ *de grado n. Entonces:*

1. *Si* $\alpha \in \mathbb{F}_q$ *es una raíz de* $f(x)$ *entonces* $x - \alpha$ *es un factor de* $f(x)$.
2. *El polinomio* $f(x)$ *tiene al menos n raíces en cualquier cuerpo que contenga a* $\mathbb{F}_q$.

Un polinomio no constante $f(x) \in \mathbb{F}_q[x]$ es *irreducible sobre* $\mathbb{F}_q$ si no es posible factorizarlo como producto de dos polinomios de $\mathbb{F}_q[x]$ de grado menor.

TEOREMA 1.2.9. *Sea* $f(x)$ *un polinomio no constante. Entonces,*

$$f(x) = p_1(x)^{a_1} p_2(x)^{a_2} \dots p_k(x)^{a_k},$$

donde cada $p_i(x)$ *es irreducible, los polinomios* $p_i(x)$ *son únicos salvo el orden en el que aparecen y producto por unidades, y los elementos* a_i *son únicos.*

Este resultado nos dice que $\mathbb{F}_q[x]$ es lo que se conoce como *dominio de factorización única*. Se puede comprobar que es, además, un dominio de ideales principales.

1.2.2 *Construcción de cuerpos finitos*

En esta subsección vamos a construir cuerpos finitos de característica arbitraria —siempre que sea un primo, por supuesto— a partir del cociente de anillos de polinomios por polinomios irreducibles.

PROPOSICIÓN I.2.10. *Sea p un número primo y $f(x) \in \mathbb{F}_p[x]$ un polinomio irreducible sobre $\mathbb{F}_p$ de grado m. Entonces, el anillo cociente dado por $\mathbb{F}_p[x]/(f(x))$ es un cuerpo de característica p con $q = p^m$ elementos.*

Puede probarse que existen polinomios irreducibles de cualquier grado, lo que nos lleva a afirmar el siguiente teorema.

TEOREMA I.2.11. *Para cualquier primo p y cualquier entero positivo m, existe un cuerpo finito, único salvo isomorfismos, con $q = p^m$ elementos.*

PROPOSICIÓN I.2.12. *Sea $\mathbb{F}_q$ un cuerpo finito, con $q = p^m$. Entonces, para cada $r|m$ existe un único subcuerpo $\mathbb{F}_{p^r}$ de $\mathbb{F}_q$.*

A la hora de trabajar con el cuerpo $\mathbb{F}_p[x]/(f(x))$ escribiremos cada clase lateral $g(x) + (f(x))$ como un vector en $\mathbb{F}_p^m$, siguiendo la correspondencia $\mathfrak{v} : \mathbb{F}_p[x]/(f(x)) \to \mathbb{F}_q$,

$$g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_1x + g_0 + (f(x)) \mapsto (g_{m-1}, g_{m-2}, \ldots, g_1, g_0).$$

Esta notación vectorial nos permite realizar la suma en el cuerpo utilizando la suma habitual de los vectores. Multiplicar es una tarea a priori más complicada. Para multiplicar $g_1(x) + (f(x))$ por $g_2(x) + (f(x))$ primero utilizamos al algoritmo de división para escribir

$$g_1(x)g_2(x) = f(x)h(x) + r(x),$$

donde como sabemos o bien $\operatorname{gr} r(x) \leq m - 1$ o bien $r(x) = 0$. Puesto que estamos en el anillo cociente $\mathbb{F}_p[x]/(f(x))$ nos queda

$$(g_1(x) + (f(x)))(g_2(x) + (f(x))) = r(x) + (f(x)).$$

Notamos rápidamente que esta notación es engorrosa, por lo que vamos a tratar de encontrar otra mejor que nos permita multiplicar con más facilidad. Para ello vamos a tomar una raíz α de $f(x)$, es decir, un elemento tal que $f(\alpha) = 0$ y operaremos en α vez de en x. Así, $g_1(\alpha)g_2(\alpha) = r(\alpha)$ y podemos extender la correspondencia vectorial a

$$(g_{m-1}, g_{m-2}, \ldots, g_1, g_0) \iff g_{m-1}\alpha^{m-1} + g_{m-2}\alpha^{m-2} + \cdots + g_1\alpha + g_0.$$

En consecuencia, multiplicamos los polinomios en α de la forma habitual y utilizamos la ecuación $f(\alpha) = 0$ para reducir las potencias de α de grado mayor a $m-1$ a polinomios en α de grado menor que m. Veamos un ejemplo.

EJEMPLO I.2.13. El polinomio $f(x) = x^2 + 1$ es irreducible sobre $\mathbb{F}_3$. Vamos a construir el cuerpo finito $\mathbb{F}_9$ como $\mathbb{F}_3[x]/(x^2 + 1)$. Sea α una raíz

del polinomio $f(x)$. Utilizando la correspondencia clases laterales-vectores-polinomios en α obtenemos los elementos descritos en la tabla. Veamos un

Clases laterales	Vectores	Polinomio en α
$0 + (f(x))$	00	0
$1 + (f(x))$	01	1
$2 + (f(x))$	02	2
$x + (f(x))$	10	α
$x + 1 + (f(x))$	11	$\alpha + 1$
$x + 2 + (f(x))$	12	$\alpha + 2$
$2x + (f(x))$	20	2α
$2x + 1 + (f(x))$	21	$2\alpha + 1$
$2x + 2 + (f(x))$	22	$2\alpha + 2$

par de ejemplos de operaciones:

— Sumemos $(x + 2 + (f(x))) + (2x + 1 + (f(x)))$. Lo haremos con la notación vectorial, sumando $12 + 21 = 00 = 0$.
— Multipliquemos $(2x + 1 + (f(x)))(2x + 2 + (f(x)))$. Lo haremos con la expresión del polinomio en α:

$$(2\alpha + 1)(2\alpha + 2) = 4\alpha^2 + 6\alpha + 2 = \alpha^2 + 2,$$

y usando que $\alpha^2 = 2$,

$$\alpha^2 + 2 = 2 + 4 = 1.$$

Decimos $\mathbb{F}_q$ es la extensión generada por una una raíz α de $f(x)$ a partir de $\mathbb{F}_p$, lo que se nota como $\mathbb{F}_q = \mathbb{F}_p(\alpha)$. Esta raíz viene dada formalmente por $\alpha = x + (f(x))$ en el anillo cociente $\mathbb{F}_p[x]/(f(x))$. Por tanto, ya hemos visto antes que $g(x) + (f(x)) = g(\alpha)$ y $f(\alpha) = f(x + (f(x))) = f(x) + (f(x)) = 0 + (f(x))$.

En el siguiente apartado veremos que multiplicar elementos de un cuerpo finito generado por una raíz α es más sencillo cuando esta raíz es un tipo de elemento concreto.

1.2.3 *Elementos primitivos*

Vamos a buscar otra forma de expresar los elementos de un cuerpo $\mathbb{F}_q$ —con $q = p^m$ para p primo— de tal forma que podamos conectarla con nuestra notación como tuplas y que haga que la multiplicación sea más sencilla de expresar. Esta expresión vendrá dada por lo que vamos a denominar elementos primitivos.

Comenzamos viendo que el conjunto $\mathbb{F}_q^*$ —de los elementos de $\mathbb{F}_q$ distintos de cero— es un grupo.

TEOREMA I.2.14. *Se verifican las siguientes afirmaciones.*

1. *El grupo $\mathbb{F}_q^*$ es cíclico de orden $q-1$ con la multiplicación de $\mathbb{F}_q$.*
2. *Si γ es un generador de este grupo cíclico entonces*

$$\mathbb{F}_q = \{0, 1 = \gamma^0, \gamma, \gamma^2, \dots, \gamma^{q-2}\},$$

y se tiene que $\gamma^i = 1$ si y solo si $(q-1) \mid i$.

Cada generador γ de $\mathbb{F}_q^*$ se llama *elemento primitivo* de $\mathbb{F}_q$. Cuando los elementos distintos de cero de un cuerpo finito se expresan como potencias de γ podemos multiplicar de forma sencilla teniendo en cuenta que $\gamma^i\gamma^j = \gamma^{i+j} = \gamma^s$, donde $0 \le s \le q-2$, e $i + j \equiv s \bmod q-1$.

TEOREMA I.2.15. *Los elementos de $\mathbb{F}_q$ son las raíces del polinomio $x^q - x$.*

Demostración. Sea γ un elemento primitivo de $\mathbb{F}_q$. Entonces, $\gamma^{q-1} = 1$ por definición. Por tanto, $(\gamma^i)^{q-1} = 1$ para todo i tal que $0 \le i \le q-2$. En consecuencia, los elementos de $\mathbb{F}_q^*$ son las raíces de $x^{q-1} - 1 \in \mathbb{F}_p[x]$ y en consecuencia, de $x^q - x$. Como 0 es raíz de $x^q - x$, por la proposición 1.2.8 sabemos que los elementos de $\mathbb{F}_q$ son las raíces de $x^q - x$, como queríamos. □

Un elemento $\xi \in \mathbb{F}_q$ es una raíz enésima de la unidad si $\xi^n = 1$, y es una raíz enésima primitiva de la unidad si además $\xi^s \neq 1$ para todo s tal que $0 < s < n$. Un elemento primitivo γ de $\mathbb{F}_q$ es por tanto una raíz $(q-1)$-ésima de la unidad. Se deduce del teorema 1.2.14 que el cuerpo $\mathbb{F}_q$ contiene una raíz enésima primitiva de la unidad si y solo si $n \mid (q-1)$, en cuyo caso $\gamma^{(q-1)/n}$ es dicha raíz.

Un polinomio irreducible sobre $\mathbb{F}_p$ de grado m es *primitivo* si tiene una raíz que es un elemento primitivo de $\mathbb{F}_q = \mathbb{F}_{p^m}$. Cuando construimos un cuerpo finito con un polinomio primitivo, podemos expresar sus elementos como potencias de la raíz primitiva correspondiente, lo que simplificará las operaciones de multiplicación. Veamos un ejemplo.

EJEMPLO I.2.16. El polinomio $f(x) = x^2 + x + 2$ es irreducible sobre $\mathbb{F}_3$. Vamos a construir el cuerpo finito $\mathbb{F}_9$ como $\mathbb{F}_3[x]/(f(x))$. Sea α una raíz del polinomio $f(x)$. Utilizando de nuevo la correspondencia clases laterales-vectores-polinomios en α obtenemos la misma descripción de los elementos, pero usando la correspondencia $\alpha^2 = -\alpha - 2 = 2\alpha + 1$ podemos expresarlos en forma de potencias de α, como vemos en la tabla.

Así, si queremos multiplicar $(2x+1+(f(x)))(2x+2+(f(x)))$ podemos hacer simplemente $\alpha^2\alpha^3 = \alpha^5 = 2\alpha = 2x + (f(x))$.

Clases laterales	Vectores	Polinomio en α	Potencia de α
$0 + (f(x))$	00	0	0
$1 + (f(x))$	01	1	$1 = \alpha^0$
$2 + (f(x))$	02	2	α^4
$x + (f(x))$	10	α	α
$x + 1 + (f(x))$	11	$\alpha + 1$	α^7
$x + 2 + (f(x))$	12	$\alpha + 2$	α^6
$2x + (f(x))$	20	2α	α^5
$2x + 1 + (f(x))$	21	$2\alpha + 1$	α^2
$2x + 2 + (f(x))$	22	$2\alpha + 2$	α^3

1.2.4 *Clases ciclotómicas y polinomios minimales*

Si consideramos la extensión de cuerpos $\mathbb{F}_{q^t}/\mathbb{F}_q$ sabemos por el teorema 1.2.15 que cada elemento de $\mathbb{F}_{q^t}$ es raíz del polinomio $x^{q^t} - x$. Existe por tanto un polinomio mónico M_α en $\mathbb{F}_q[x]$ de grado mínimo que tiene a α como raíz. Este polinomio se conoce como *polinomio minimal* de α sobre $\mathbb{F}_q$. El siguiente teorema nos detalla algunas de las propiedades de los polinomios minimales.

TEOREMA 1.2.17. *Sea $\mathbb{F}_{q^t}/\mathbb{F}$ una extensión de cuerpos y sea α un elemento de $\mathbb{F}_{q^t}$ cuyo polinomio minimal es $M_\alpha \in \mathbb{F}_q[x]$. Se verifica:*

1. *El polinomio $M_\alpha(x)$ es irreducible sobre $\mathbb{F}_q$.*
2. *Si $g(x)$ es cualquier polinomio en $\mathbb{F}_q[x]$ tal que $g(\alpha) = 0$ entonces $M_\alpha(x) \mid g(x)$.*
3. *El polinomio $M_\alpha(x)$ es único.*

Si partimos de $f(x)$, un polinomio irreducible sobre $\mathbb{F}_q$ de grado r, podemos considerar la extensión generada por una de las raíces de $f(x)$ y obtendremos el cuerpo $\mathbb{F}_{q^r}$. De hecho, el siguiente teorema afirma que en ese caso todas las raíces de $f(x)$ estarán en $\mathbb{F}_{q^r}$.

TEOREMA 1.2.18. *Sea $f(x)$ un polinomio irreducible mónico sobre $\mathbb{F}_q$ de grado r. Entonces:*

1. *Todas las raíces de $f(x)$ están en $\mathbb{F}_{q^r}$ y en cualquier extensión de cuerpos de $\mathbb{F}_q$ generada por una de sus raíces.*
2. *Podemos expresar $f(x)$ como $f(x) = \prod_{i=1}^{r}(x - \alpha_i)$, donde $\alpha_i \in \mathbb{F}_{q^r}$ para $1 \leq i \leq r$.*
3. *El polinomio $f(x)$ divide a $x^{q^r} - x$.*

Demostración. Veamos la demostración por partes.

1. Sea α una raíz de $f(x)$ y consideramos el cuerpo $\mathbb{F}_{q^r} = \mathbb{F}_q(\alpha)$, la extensión de $\mathbb{F}_q$ generada por α. Sea β otra raíz de $f(x)$ y supongamos que no está en $\mathbb{F}_q(\alpha)$. Entonces, será raíz de algún factor irreducible de $f(x)$ sobre $\mathbb{F}_q(\alpha)$. Consideramos ahora la extensión $\mathbb{F}_q(\alpha, \beta)$, la extensión de $\mathbb{F}_q(\alpha)$ generada por β. Dentro de $\mathbb{F}_q(\alpha, \beta)$ hay un subcuerpo $\mathbb{F}_q(\beta)$, que es la extensión de $\mathbb{F}_q$ generada por β. Además, $\mathbb{F}_q(\beta)$ ha de tener q^r elementos, pues $f(x)$ es un irreducible de grado r sobre $\mathbb{F}_q$. Como tanto $\mathbb{F}_q(\alpha)$ como $\mathbb{F}_q(\beta)$ son subcuerpos del mismo tamaño, han de ser iguales por la proposición 1.2.12. Por tanto, todas las raíces de $f(x)$ están en $\mathbb{F}_{q^r}$. Finalmente, todo cuerpo conteniendo a $\mathbb{F}_q$ y una raíz de $f(x)$ contiene a $\mathbb{F}_{q^r}$.
2. Se deduce de lo anterior y de la proposición 1.2.8.
3. Se deduce del apartado anterior y del hecho de que por el teorema 1.2.15, $x^{q^r} - x = \prod_{\alpha \in \mathbb{F}_{q^r}} (x - \alpha)$. □

En particular este teorema se verifica para los polinomios minimales $M_\alpha(x)$ sobre $\mathbb{F}_q$, pues son mónicos irreducibles. Para la demostración del siguiente teorema necesitamos el lema que enunciamos a continuación.

LEMA 1.2.19. *Sea $s = p^r$ y $q = p^m$. Entonces $(x^s - x) \mid (x^q - x)$ si y solo si $r \mid m$.*

TEOREMA 1.2.20. *Sea $\mathbb{F}_{q^t} / \mathbb{F}_q$ una extensión de cuerpos y sea α un elemento de $\mathbb{F}_{q^t}$ con polinomio minimal M_α en $\mathbb{F}_q[x]$. Se verifican las siguientes afirmaciones.*

1. *El polinomio $M_\alpha(x)$ divide a $x^{q^t} - x$.*
2. *El polinomio $M_\alpha(x)$ tiene raíces distintas dos a dos, todas en $\mathbb{F}_{q^t}$.*
3. *El grado de $M_\alpha(x)$ divide a t.*
4. *Podemos expresar $x^{q^t} - x = \prod_\alpha M_\alpha(x)$, donde α varía entre los elementos de un subconjunto de $\mathbb{F}_{q^t}$ de forma que enumera los polinomios minimales de todos los elementos de $\mathbb{F}_{q^t}$ una sola vez.*
5. *Podemos expresar $x^{q^t} - x = \prod_f f(x)$, donde f varía entre todos los mónicos irreducibles sobre $\mathbb{F}_q$ cuyo grado divide a t.*

Demostración. Veamos la demostración por apartados.

1. Se deduce del teorema 1.2.17(2), pues por el teorema 1.2.15, $\alpha^{q^t} - \alpha = 0$.
2. Las raíces del polinomio $x^{q^t} - x$ son los q^t elementos de $\mathbb{F}_{q^t}$, luego este polinomio tiene raíces distintas dos a dos en $\mathbb{F}_{q^t}$ por el teorema 1.2.18(1). Ahora, como por 1 $M_\alpha(x)$ divide a $x^{q^t} - x$, todas sus raíces también son distintas dos a dos y están en $\mathbb{F}_{q^t}$.
3. Si el grado de $M_\alpha(x)$ es r, la extensión $\mathbb{F}_q(\alpha)$ genera el subcuerpo $\mathbb{F}_{q^r} = \mathbb{F}_{p^{mr}}$ de $\mathbb{F}_{q^t} = \mathbb{F}_{p^{mt}}$, de forma que $mr \mid mt$ por el teorema 1.2.12 y por tanto el grado de $M_\alpha(x)$ divide a t.

4. Como $x^{q^t} - x$ tiene raíces distintas dos a dos, sus factores $p_i(x)$ también lo son, y puesto que es mónico, podemos asumir que también lo son. Por tanto, $p_i(x) = M_\alpha(x)$ para cualquier $\alpha \in \mathbb{F}_{q^t}$ tal que $p_i(\alpha) = 0$, obteniendo así los factores del producto que buscamos.
5. Este apartado se deduce del anterior, si demostramos que todo polinomio mónico irreducible de grado $r|t$ sobre $\mathbb{F}_q$ es un factor de $x^{q^t} - x$. Pero $f(x)|(x^{q^r} - x)$ por el teorema 1.2.18(3). Como $mr|mt$, $(x^{q^r} - x)|(x^{q^t} - x)$ por el lema 1.2.19. □

Dos elementos de $\mathbb{F}_{q^t}$ que tienen el mismo polinomio minimal en $\mathbb{F}_q[x]$ se llaman *conjugados sobre* $\mathbb{F}_q$. Es importante encontrar todos los conjugados de $\alpha \in \mathbb{F}_q$, es decir, todas las raíces de $M_\alpha(x)$. Sabemos por el teorema 1.2.20 que las raíces de $M_\alpha(x)$ son todas distintas dos a dos y que se encuentran en el cuerpo $\mathbb{F}_{q^t}$. Podemos encontrar estas raíces con ayuda del siguiente teorema.

TEOREMA 1.2.21. *Sea $f(x)$ un polinomio en $\mathbb{F}_q[x]$ y sea α una raíz de $f(x)$ en una extensión $\mathbb{F}_{q^t}/\mathbb{F}_q$. Entonces se verifican las siguientes afirmaciones.*

1. *Evaluando el polinomio obtenemos que $f(x^q) = f(x)^q$.*
2. *El elemento α^q es también una raíz de $f(x)$ en $\mathbb{F}_q$.*

Demostración. Veamos la demostración por apartados.

1. Sea $f(x) = \sum_{i=0}^{n} a_i x^i$. Como $q = p^m$, donde p es la característica de $\mathbb{F}_q$, el polinomio $f(x)^q = \sum_{i=0}^{n} a_i^q x^{iq}$ aplicando la proposición 1.2.2 repetidamente. Sin embargo, $a_i^q = a_i$, porque $a_i \in \mathbb{F}_q$ y los elementos de $\mathbb{F}_q$ son la raíces de $x^q - x$ por el teorema 1.2.15.
2. Se deduce del apartado anterior, pues $f(\alpha^q) = f(\alpha)^q = 0$. □

Si aplicamos este teorema de forma consecutiva podremos obtener todas las raíces de $M_\alpha(x)$, que serán de la forma $\alpha, \alpha^q, \alpha^{q^2}$, etc.; secuencia que terminará tras r términos, cuando $\alpha^{q^r} = \alpha$.

Supongamos ahora que γ es un elemento primitivo de $\mathbb{F}_{q^t}$. Entonces sabemos que $\alpha = \gamma^s$ para algún s. Por tanto, $\alpha^{q^r} = \alpha$ si y solo si $\gamma^{sq^r - s} = 1$. Por el teorema 1.2.14 se tiene que $sq^r \equiv s$ mód $q^t - 1$. Basándonos en esta idea podemos definir la *clase q-ciclotómica de s módulo* $q^t - 1$ como el conjunto

$$C_s = \{s, sq, \dots, sq^{r-1}\} \text{ mód } q^t - 1,$$

donde r es el menor entero positivo tal que $sq^r \equiv s$ mód $q^t - 1$. Los conjuntos C_s dividen el conjunto de enteros $\{0, 1, 2, \dots, q^t - 2\}$ en conjuntos disjuntos.

TEOREMA I.2.22. *Si γ es un elemento primitivo de $\mathbb{F}_{q^t}$ entonces el polinomio minimal de γ^s sobre $\mathbb{F}_q$ es*

$$M_{\gamma^s}(x) = \prod_{i \in C_s} (x - \gamma^i).$$

I.3 AUTOMORFISMOS DE CUERPOS FINITOS

En esta sección vamos a describir someramente los automorfismos de cuerpos finitos, que nos serán necesarios cuando trabajemos con anillos de polinomios de Ore.

Dado un cuerpo finito $\mathbb{F}_q$ un automorfismo σ de $\mathbb{F}_q$ es una aplicación biyectiva $\sigma : \mathbb{F}_q \to \mathbb{F}_q$ tal que $\sigma(\alpha+\beta) = \sigma(\alpha)+\sigma(\beta)$ y $\sigma(\alpha\beta) = \sigma(\alpha)\sigma(\beta)$ para todo $\alpha, \beta \in \mathbb{F}_q$. Vamos a describir en el siguiente ejemplo un automorfismo al que nos referiremos en el futuro.

EJEMPLO I.3.1. Sea $\mathbb{F}_q$ un cuerpo finito, donde $q = p^m$ con p primo. La aplicación biyectiva dada por $\sigma_p : \mathbb{F}_q \to \mathbb{F}_q$ tal que $\sigma_p(\alpha) = \alpha^p$ para todo $\alpha \in \mathbb{F}_q$ es un automorfismo, conocido como *automorfismo de Frobenius*.

Al grupo G que forman los automorfismos de un cuerpo finito $\mathbb{F}_q$ lo llamamos *grupo de Galois* de $\mathbb{F}_q$. Así, definimos el *orden* de un automorfismo como el menor n tal que $\sigma^n(\alpha) = \alpha$ para todo $\alpha \in \mathbb{F}_q$.

TEOREMA I.3.2. *El grupo de Galois de un cuerpo finito $\mathbb{F}_q$, con $q = p^m$ y p primo es cíclico de orden m y está generado por el automorfismo de Frobenius σ_p.*

Decimos que un elemento $\alpha \in \mathbb{F}_q$ queda *fijo* por un automorfismo σ si $\sigma(\alpha) = \alpha$. El conjunto de los elementos de $\mathbb{F}_q$ que quedan fijos por un automorfismo σ forma un subcuerpo de $\mathbb{F}_q$ y se denomina *subcuerpo fijo de* $\mathbb{F}_q$ por σ. Lo denotamos por $\mathbb{F}_q^\sigma$.

DEFINICIÓN I.3.3. Sea $\mathbb{F}_q/\mathbb{F}_p$, con $q = p^m$, una extensión de cuerpos finitos. Una *base normal* de $\mathbb{F}_q$ sobre $\mathbb{F}_p$ es una base de la forma $\{\alpha, \alpha^p, \dots, \alpha^{q-1}\}$ para algún $\alpha \in \mathbb{F}_q$.

En general las bases normales de los cuerpos finitos están formadas por las imágenes de un elemento por los distintos automorfismos de su grupo de Galois. Pero como en este caso el grupo de Galois está generado de forma cíclica por el automorfismo de Frobenius, podemos expresarlo como en la definición 1.3.3. El siguiente resultado se conoce como *teorema de la base normal*.

TEOREMA I.3.4. *Existe una base normal para toda extensión de cuerpos.*

Así, podremos obtener una base normal de un cuerpo $\mathbb{F}_q$ como espacio vectorial de un subcuerpo fijo $\mathbb{F}_q^\sigma$.

2

FUNDAMENTOS DE TEORÍA DE CÓDIGOS

En la introducción ya hemos visto cuáles son los objetivos de la teoría de códigos, así como el medio principal del que se sirve: el álgebra. Esta sección vamos a comentar algunos de los conceptos y resultados fundamentales de la teoría de códigos. Comenzaremos viendo la definición más sencilla de código, para posteriormente introducir la clase de códigos lineales, de la que estudiaremos los principales conceptos y cómo codificar y decodificar. Finalmente veremos algunas familias de códigos lineales importantes.

Las definiciones y los resultados comentados en esta sección seguirán lo descrito en (Huffman & Pless, 2003, cap. 1, 3-5) y (Podestá, 2006).

2.1 CÓDIGOS LINEALES

Vamos a comenzar nuestro estudio con los códigos lineales, pues son los más sencillos de comprender. Consideremos el espacio vectorial de todas las n-tuplas sobre el cuerpo finito $\mathbb{F}_q$, al que denotaremos en lo que sigue como $\mathbb{F}_q^n$. A los elementos $(a_1, \dots, a_n)$ de $\mathbb{F}_q^n$ los notaremos usualmente como $a_1 \cdots a_n$.

DEFINICIÓN 2.1.1. Un (n, M) *código* $\mathcal{C}$ sobre el cuerpo $\mathbb{F}_q$ es un subconjunto de $\mathbb{F}_q^n$ de tamaño M. Si no hay riesgo de confusión lo denotaremos simplemente por $\mathcal{C}$. A los elementos de $\mathcal{C}$ los llamaremos *palabras código*. A n se le llama *longitud* del código.

Por ejemplo, un $(5, 4)$ código sobre $\mathbb{F}_2$ puede ser el formado por los siguientes elementos:

$$10101, \qquad 10010, \qquad 01110, \qquad 11111.$$

Como se puede ver realmente un código es un objeto muy sencillo. Concluimos que es necesario añadir más estructura a los códigos para que puedan ser de utilidad. Esto motiva la siguiente definición.

DEFINICIÓN 2.1.2. Decimos que un código $\mathcal{C}$ es un código *lineal de longitud n y dimensión k* —abreviado como $[n, k]$-*lineal*, o como $[n, k]_q$-*lineal* en caso de querer informar del cuerpo base— si dicho código es un subespacio vectorial de $\mathbb{F}_q^n$ de dimensión k.

NOTA 2.1.3. *Un código lineal $\mathcal{C}$ tiene q^k palabras código.*

Así, hemos pasado de trabajar con un objeto que no tiene estructura alguna a trabajar con espacios vectoriales, cuyas propiedades son ampliamente conocidas y disponemos de numerosas herramientas para tratarlos. Por ejemplo, en ocasiones hablaremos de *subcódigos* de un código $\mathcal{C}$. Si $\mathcal{C}$ es un código lineal entonces el subcódigo será un subespacio vectorial del mismo. En caso de que sea no lineal un subcódigo será simplemente un subconjunto de $\mathcal{C}$.

Veamos en la siguiente definición otro ejemplo de las herramientas que nos proporciona trabajar con espacios vectoriales.

DEFINICIÓN 2.1.4. Una *matriz generadora* para un $[n, k]$ código $\mathcal{C}$ es una matriz $k \times n$ cuyas filas conforman una base de $\mathcal{C}$[1].

DEFINICIÓN 2.1.5. Para cada conjunto k de columnas independientes de una matriz generadora G el conjunto de coordenadas correspondiente se denomina *conjunto de información* para un código $\mathcal{C}$. Las $r = n - k$ coordenadas restantes se llaman *conjunto redundante*, y el número r, la *redundancia* de $\mathcal{C}$.

Si las primeras k coordenadas de una matriz generadora G forman un conjunto de información entonces el código tiene una única matriz generadora de la forma $(I_k \mid A)$, donde I_k es la matriz identidad $k \times k$ y A es una matriz $k \times r$. Esta matriz generadora se dice que está en *forma estándar*. A partir de cualquier matriz generadora siempre es posible obtener una matriz en forma estándar realizando una permutación adecuada de las coordenadas.

Como un código lineal $\mathcal{C}$ es un subespacio de un espacio vectorial, podemos calcular el ortogonal a dicho subespacio, obteniendo lo que llamaremos el *código dual* (*euclídeo*, si usamos el producto escalar usual) y que denotaremos por $\mathcal{C}^\perp$.

DEFINICIÓN 2.1.6. El *código dual* $\mathcal{C}^\perp$ de un código $\mathcal{C}$ viene dado por

$$\mathcal{C}^\perp = \left\{ x \in \mathbb{F}_q^n : x \cdot c = 0 \quad \text{para todo } c \in \mathcal{C} \right\},$$

donde $(\cdot)$ representa el producto escalar usual.

DEFINICIÓN 2.1.7. Sea $\mathcal{C}$ un $[n, k]$ código lineal. Una matriz H se dice que es *matriz de paridad* si es una matriz generadora de $\mathcal{C}^\perp$.

1 Efectivamente la matriz generadora no es única: basta tomar la correspondiente a cualquier otra base del código —que no deja de ser un espacio vectorial— para obtener una distinta. Pero es más, podemos simplemente reordenar las filas de una matriz generadora y en esencia estaremos obteniendo otra distinta.

PROPOSICIÓN 2.1.8. *Sea H la matriz de paridad de un $[n, k]$ código lineal $\mathcal{C}$. Entonces,*

$$\mathcal{C} = \left\{x \in \mathbb{F}_q^n : xH^T = 0\right\} = \left\{x \in F_q^n : Hx^T = 0\right\}.$$

Demostración. Sea $c \in \mathcal{C}$ una palabra código. Sabemos que la podemos expresar como $c = uG$, donde $u \in \mathbb{F}_q^k$ y G es una matriz generadora de $\mathcal{C}$. Tenemos entonces que $cH^T = uGH^T$ y como $GH^T = 0$ —por ser H matriz generadora del subespacio ortogonal $\mathcal{C}$— se tiene que

$$\mathcal{C} \subset S_H = \left\{x \in \mathbb{F}_q^n : Hx^T = 0\right\},$$

que es el espacio solución de un sistema de $n - k$ ecuaciones con n incógnitas y de rango $n - k$. Como $\dim(S_H) = n - (n - k) = k = \dim L$, concluimos que

$$L = S_H = \left\{x \in \mathbb{F}_q^n : Hx^T = 0\right\}. \qquad \square$$

Este último resultado, junto a la definición previa, nos conducen al siguiente teorema.

TEOREMA 2.1.9. *Si $G = (I_k \mid A)$ es una matriz generadora para un $[n, k]$ código $\mathcal{C}$ en forma estándar entonces $H = (-A \mid I_{n-k})$ es una matriz de paridad para $\mathcal{C}$.*

Como nota final sobre nomenclatura de códigos duales, apuntamos que un código se dice *autoortogonal* cuando $\mathcal{C} \subseteq \mathcal{C}^\perp$, y *autodual* cuando $\mathcal{C} = \mathcal{C}^\perp$.

2.1.1 *Codificación y decodificación*

Codificar un mensaje consiste en escribirlo como palabra código de un código. La forma estándar de codificar mensajes con códigos lineales es utilizando una matriz generadora. Dado un mensaje $\mathbf{m} \in \mathbb{F}_q^k$ podemos obtener la palabra código $\mathbf{c}$ en $\mathcal{C}$ realizando la operación $\mathbf{c} = \mathbf{m}G$. Vamos a verlo mejor con un ejemplo.

EJEMPLO 2.1.10. Sea $\mathcal{C}$ $[3, 2]$ un código binario lineal y G la matriz generadora dada por

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \in \mathcal{M}_{2\times 3}(\mathbb{F}_2).$$

Dado un mensaje $\mathbf{m} = (x_1, x_2)$, se tiene que

$$(x_1, x_2)\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = (x_1, x_1 + x_2, x_2),$$

y por tanto esta matriz codifica de la forma

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

Observamos que una matriz generador G define una aplicación lineal de $\mathbb{F}_q^k$ en $\mathbb{F}_q^n$, de forma que el código obtenido es la imagen de dicha aplicación. Podemos comprobar también que es posible codificar en los mismos códigos lineales utilizando distintas matrices generadoras, lo que resultará en distintas palabras código para el mismo mensaje. Veamos un ejemplo con el mismo código binario lineal que en el ejemplo anterior pero con distinta matriz generadora.

EJEMPLO 2.1.11. Sea $\mathcal{C}$ un $[3, 2]$ código binario lineal y G la matriz generadora dada por

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \in \mathcal{M}_{2\times 3}(\mathbb{F}_2).$$

Dado un mensaje $\mathbf{m} = (x_1, x_2)$, se tiene que

$$(x_1, x_2)\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = (x_1, x_2, x_1 + x_2),$$

y por tanto esta matriz codifica de la forma

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

Observamos en este ejemplo que las primeras 2 coordenadas de cada palabra código son iguales a las del mensaje que las genera. Pero en el código anterior también podemos encontrar el mensaje, lo que hay que fijarse en la primera y última coordenada. Cuando un mensaje se encuentra incrustado íntegramente en la palabra código —aunque puede que desordenado— se dice que la codificación seguida es *sistemática*. En caso contrario, se dice que es *no-sistemática*. Veamos un ejemplo de codificación no-sistemática con el mismo código binario lineal de antes.

EJEMPLO 2.1.12. Sea $\mathcal{C}$ un $[3, 2]$ código binario lineal y G la matriz generadora dada por

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \in \mathcal{M}_{2\times 3}(\mathbb{F}_2).$$

Dado un mensaje $\mathbf{m} = (x_1, x_2)$, se tiene que

$$(x_1, x_2)\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} = (x_1 + x_2, x_2, x_1 + x_2),$$

y por tanto esta matriz codifica de la forma

$$00 \to 000, \quad 01 \to 111, \quad 10 \to 101, \quad 11 \to 010.$$

Comprobamos que los mensajes 01 y 11 no están contenidos en las palabras código correspondientes, 111 y 010, respectivamente, luego la codificación es no-sistemática.

Dada una palabra código $\mathbf{c}$ si se desea obtener el mensaje $\mathbf{m}$ a partir del que se obtuvo podemos realizar el procedimiento inverso a la codificación. Para ello tenemos en cuenta que al codificar mediante una matriz generadora G de tamaño $n \times k$ establecemos una correspondencia biyectiva entre mensajes y palabras código. Existe por tanto una matriz K de tamaño $k \times n$ llamada *inversa por la derecha* tal que $GK = I_k$. Así, puesto que $\mathbf{c} = \mathbf{m}G$ podemos obtener el mensaje original calculando $\mathbf{c}K = \mathbf{m}GK = \mathbf{m}$. Veamos un ejemplo de este proceso.

EJEMPLO 2.1.13. Sea C un $[7, 3]$ código binario lineal y G la matriz generadora dada por

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \in \mathcal{M}_{3\times 7}(\mathbb{F}_2).$$

Esta matriz codifica el mensaje $\mathbf{m} = (1, 0, 1)$ como:

$$\mathbf{c} = (1, 0, 1) \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} = (1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1)\,.$$

Para realizar el procedimiento inverso buscamos una matriz K tal que $GK = I_7$. Esta matriz viene dada por

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

y por tanto el mensaje original era

$$\mathbf{m} = (1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1)\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = (1, 0, 1).$$

El proceso de decodificación de los mensajes consiste en obtener una palabra código válida a partir de un mensaje recibido[2]. Es una tarea mucho más complicada que los procesos comentados antes, pues como ya se ha mencionado hay que tener en cuenta las posibles interferencias que se hayan podido producir en la comunicación. Existen numerosos métodos de decodificación, y en general, cada familia de códigos tendrá un sistema que se aproveche de sus propiedades para ofrecer mejores prestaciones. Destacamos de entre todos ellos un sistema aplicable a los códigos lineales, el conocido como *decodificación por síndromes*, pues en él se basará el algoritmo cuya descripción es el objetivo de este trabajo. Este método se basa en la propiedad de la matriz de paridad de que para toda palabra código $\mathbf{c} \in \mathcal{C}$ se tiene que $H\mathbf{c} = 0$. Someramente el método consiste en computar y almacenar previamente los resultados del producto de todos los posibles vectores error y cuando se recibe un mensaje $\mathbf{y} = \mathbf{c} + \mathbf{e}$, donde $\mathbf{e}$ representa el error que se ha producido en la transmisión, se calcula lo que se conoce como *síndrome*, que es el producto

$$H\mathbf{y} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}.$$

Este síndrome obtenido se compara con los productos previamente calculados para determinar qué error $\mathbf{e}$ se ha producido y el mensaje codificado se obtiene como $\mathbf{c} = \mathbf{y} - \mathbf{e}$.

2.1.2 *Distancias y pesos*

Códigos distintos poseen distintas propiedades, lo que implica que sus capacidades de corrección difieran. En este apartado vamos a estudiar dos propiedades de los códigos muy relacionadas con esta idea.

2 Es importante llamar la atención sobre el hecho de que *decodificar* no es el proceso inverso a *codificar*. Codificar consiste en escribir un mensaje como palabra código y decodificar, en corregir los errores que se hayan podido producir en la transmisión de dicha palabra.

DEFINICIÓN 2.1.14. La *distancia de Hamming* $\mathrm{d}(\mathbf{x}, \mathbf{y})$ entre dos vectores $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ se define como el número de coordenadas en las que difieren $\mathbf{x}$ e $\mathbf{y}$.

TEOREMA 2.1.15. *La función de distancia* $\mathrm{d}(\mathbf{x}, \mathbf{y})$ *verifica las siguientes propiedades.*

1. *No negatividad:* $\mathrm{d}(\mathbf{x}, \mathbf{y}) \geq 0$ *para todo* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$.
2. *La distancia* $\mathrm{d}(\mathbf{x}, \mathbf{y}) = 0$ *si y solo si* $\mathbf{x} = \mathbf{y}$.
3. *Simetría:* $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \mathrm{d}(\mathbf{y}, \mathbf{x})$ *para todo* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$.
4. *Desigualdad triangular:* $\mathrm{d}(\mathbf{x}, \mathbf{z}) \leq \mathrm{d}(\mathbf{x}, \mathbf{y}) + \mathrm{d}(\mathbf{y}, \mathbf{z})$ *para todo elemento* $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_q^n$.

Demostración. Las tres primeras propiedades son evidentes. La cuarta se comprueba fácilmente. Si $\mathbf{x} = \mathbf{z}$ la desigualdad se da de forma trivial, pues $\mathrm{d}(\mathbf{x}, \mathbf{z}) = 0$. En caso de que $\mathbf{x} \neq \mathbf{z}$ se tiene que $\mathbf{x} \neq \mathbf{y}$ o $\mathbf{y} \neq \mathbf{z}$, y en consecuencia, por la no negatividad, la desigualdad se verifica. □

La *distancia* (*mínima*) de un código $\mathcal{C}$ es la menor distancia posible entre dos palabras código distintas. Si la distancia mínima d de un $[n, k]$ código es conocida, nos referiremos a él como un $[n, k, d]$ código. Este valor es importante pues nos ayuda a determinar la capacidad de corrección de errores del código $\mathcal{C}$, como ilustra el siguiente teorema.

TEOREMA 2.1.16. *Sea* $\mathcal{C}$ *un* $[n, k, d]$ *código. Entonces* $\mathcal{C}$ *puede detectar hasta* $d - 1$ *errores y puede corregir hasta*

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

errores.

Demostración. Por definición de distancia mínima, al añadir menos de d errores no se obtiene una palabra código, y por tanto se pueden detectar hasta $d - 1$ errores. Sea ahora $\mathbf{y}$ un mensaje recibido con, como mucho, t errores. Entonces existe una palabra código $\mathbf{c} \in \mathcal{C}$ tal que $\mathrm{d}(\mathbf{y}, \mathbf{c}) \leq t$. Supongamos que existe otra palabra $\mathbf{c}' \in \mathcal{C}$ tal que $\mathrm{d}(\mathbf{y}, \mathbf{c}') \leq t$. Entonces, por la desigualdad triangular,

$$\mathrm{d}(\mathbf{c}, \mathbf{c}') \leq \mathrm{d}(\mathbf{c}, \mathbf{y}) + \mathrm{d}(\mathbf{y}, \mathbf{c}') \leq 2t \leq d - 1,$$

lo que es claramente imposible. Por tanto $\mathcal{C}$ puede corregir hasta t errores. □

Efectivamente, a mayor distancia mínima, mayor número de errores en el código se pueden corregir. Otra medida interesante es el *peso de Hamming*.

DEFINICIÓN 2.1.17. El *peso de Hamming* $\mathrm{wt}(\mathbf{x})$ de un vector $\mathbf{x}$ es el número de coordenadas distintas de cero de $\mathbf{x}$.

El siguiente teorema nos ilustra la relación existente entre los conceptos de peso y distancia.

TEOREMA 2.1.18. *Si* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$, *entonces* $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \mathrm{wt}(\mathbf{x}-\mathbf{y})$. *Si* C *es un código lineal, la distancia mínima es igual al peso mínimo de las palabras código de* C *distintas de cero.*

Demostración. Se tiene que $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \mathrm{d}(\mathbf{0}, \mathbf{y} - \mathbf{x}) = \mathrm{wt}(\mathbf{y} - \mathbf{x})$. Tenemos entonces que $\mathrm{wt}(\mathbf{x}) = \mathrm{d}(\mathbf{0}, \mathbf{x})$ y que $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \mathrm{wt}(\mathbf{x} - \mathbf{y})$. Sea $\mathbf{c}$ una palabra código de peso mínimo. Entonces $\mathrm{wt}(\mathbf{c}) = \mathrm{d}(\mathbf{0}, \mathbf{c})$ y por tanto tenemos que $d_{min} \leq w_{min}$. Por otro lado, si $\mathbf{c_1}$ y $\mathbf{c_2}$ son palabras entre las que hay distancia mínima, $\mathrm{d}(\mathbf{c_1}, \mathbf{c_2}) = \mathrm{wt}(\mathbf{c_1} - \mathbf{c_2})$ y como $\mathbf{c_1} - \mathbf{c_2}$ es también una palabra código, $w_{min} \leq d_{min}$. En consecuencia, $w_{min} = d_{min}$. □

Como consecuencia de este teorema —para códigos lineales— la distancia mínima también se llama *peso mínimo* del código.

DEFINICIÓN 2.1.19. Sea $A_i(C)$ —que abreviaremos A_i— el número de palabras código de peso i en C. Para cada $0 \leq i \leq n$, la lista A_i se denomina *distribución de peso* o *espectro de peso* de C.

EJEMPLO 2.1.20. Sea C el código binario con matriz generadora

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

Dado (x_1, x_2, x_3), se tiene que

$$(x_1, x_2, x_3) \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} = (x_1, x_1, x_2, x_2, x_3, x_3),$$

y por tanto podemos obtener las palabras código de la forma

$$000 \to 000000, \quad 001 \to 000011, \quad 010 \to 001100, \quad 011 \to 001111,$$

$$100 \to 110000, \quad 101 \to 110011, \quad 110 \to 111100, \quad 111 \to 111111.$$

Luego la distribución de peso de C es $A_0 = A_6 = 1$ y $A_2 = A_4 = 3$. Usualmente solo se listan los A_i que son distintos de cero.

PROPOSICIÓN 2.1.21. *Sea* C *un* $[n, k, d]$ *código sobre* $\mathbb{F}_q$. *Entonces,*

1. $A_0(\mathcal{C}) + A_1(\mathcal{C}) + \cdots + A_n(\mathcal{C}) = q^k$.
2. $A_0(\mathcal{C}) = 1$ *y* $A_1(\mathcal{C}) = A_2(\mathcal{C}) = \cdots = A_{d-1}(\mathcal{C}) = 0$.

Demostración. La primera afirmación es trivial, pues estamos sumando todas las palabras del código, que sabemos que son q^k en total. La segunda afirmación también lo es. Es evidente que solo hay una palabra código con peso 0, la $\mathbf{0}$. Por otro lado, si la distancia mínima del código es d toda palabra código se diferenciará de $\mathbf{0}$ en al menos d coordenadas, y en consecuencia, $A_1(\mathcal{C}) = A_2(\mathcal{C}) = \cdots = A_{d-1}(\mathcal{C}) = 0$. □

TEOREMA 2.1.22. *Sea $\mathcal{C}$ un código lineal con matriz de paridad H. Si $\mathbf{c} \in \mathcal{C}$, las columnas de H que se corresponden con coordenadas no nulas de $\mathbf{c}$ son linealmente dependientes. Recíprocamente, si entre w columnas de H existe una relación de dependencia lineal con coeficientes no nulos, entonces hay una palabra código en $\mathcal{C}$ de peso w cuyas coordenadas no nulas se corresponden con dichas columnas.*

Demostración. Si $\mathbf{c} \in \mathcal{C}$ por la definición de matriz de paridad tenemos que $\mathbf{c}^T H = \mathbf{0}$. Si $\mathbf{c} = (c_1, \ldots, c_n)$ podemos expresar esta relación como $\sum_{i=0}^{n} c_i \mathbf{h}_i = \mathbf{0}$, donde $\mathbf{h}_1, \ldots, \mathbf{h}_n$ son las columnas de H. Por tanto, si $\mathbf{c} \neq \mathbf{0}$ se define una relación de dependencia lineal entre las columnas correspondientes a las coordenadas no nulas de $\mathbf{c}$. Por otro lado, supongamos ahora que existe una relación de dependencia lineal entre w columnas de H. Tendremos entonces que $\sum_{i=1}^{w} a_i \mathbf{h}_i = \mathbf{0}$, pero sabemos que esto supone que los coeficientes a_i forman una palabra código de $\mathcal{C}$, que efectivamente tendrá peso w. □

COROLARIO 2.1.23. *Un código lineal tiene peso mínimo d si y solo si su matriz de paridad tiene un conjunto de d columnas linealmente dependientes pero no tiene un conjunto de $d-1$ columnas linealmente dependientes.*

Demostración. Comencemos suponiendo que un código lineal $\mathcal{C}$ tiene peso mínimo d. Así, habrá alguna palabra código $\mathbf{c}$ tal que $\mathrm{wt}(\mathbf{c}) = d$. En ese caso, por el teorema 2.1.2 la matriz de paridad de $\mathcal{C}$ tendrá d columnas linealmente dependientes. Si tuviese algún conjunto de $d-1$ columnas linealmente dependientes el mismo teorema nos asegura que existiría una palabra con peso $d-1$, lo que es imposible pues el peso mínimo es d. Recíprocamente, si la matriz de paridad tiene un conjunto de d columnas linealmente dependientes pero no tiene un conjunto de $d-1$ columnas linealmente dependientes toda palabra código tendrá entonces peso al menos d, y en consecuencia, d es el peso mínimo del código. □

2.2 EJEMPLOS DE CÓDIGOS

En esta sección vamos a describir someramente algunas familias de códigos relevantes: los códigos de repetición, los códigos de control de paridad y los códigos de Hamming.

2.2.1 *Códigos de repetición*

Los códigos de repetición son una de las familias de códigos más sencillas. Dado un mensaje $\mathbf{m} = (m_1, m_2, \dots, m_n) \in \mathbb{F}_q^n$ lo que hacemos para codificarlo es repetir cada elemento m_i de la tupla k veces:

$$\mathbf{c} = (m_{11}, m_{12}, \dots, m_{1k}, m_{21}, m_{22}, \dots, m_{2k}, \dots, m_{n1}, m_{n2}, \dots, m_{nk}).$$

A la hora de decodificar un mensaje cada bloque de k elementos se fija al valor del elemento que más se repita. Los más utilizados son los códigos de repetición binarios, es decir, los que se definen sobre $\mathbb{F}_2$. No son códigos lineales.

2.2.2 *Códigos de control de paridad*

Los $[n, n-1]$-códigos lineales que tienen matriz de paridad

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

se llaman *códigos de control de paridad* o *códigos de peso par*. Por la proposición 2.1.8 las palabras código $\mathbf{c}$ de este tipo de códigos han de cumplir que

$$\mathbf{c}H^T = c_1 + c_2 + \cdots + c_n = 0,$$

es decir, el número de 1 en la palabra código ha de ser par —de ahí el nombre—. La codificación de mensajes se realiza entonces añadiendo un *bit* de paridad al final del mensaje cuyo valor se fija para que el número de 1 en el mismo sea par. Estos códigos tienen distancia 2 y pueden corregir un solo error.

2.2.3 *Códigos de Hamming*

Consideremos una matriz $r \times (2^r - 1)$ cuyas columnas son los números $1, 2, 3, \dots, 2^{r-1}$ escritos en binario. Dicha matriz es la matriz de paridad de un $[n = 2^{r-1}, k = n - r]$ código binario. A los códigos de esta forma los llamaremos códigos de Hamming de longitud $n = 2^{r-1}$ y los denotamos por $\mathcal{H}_r$ o $\mathcal{H}_{2,r}$.

Como las columnas son distintas y no nulas, la distancia es al menos 3 por el corolario 2.1.23. Además, como las columnas correspondientes a los números $1, 2, 3$ son linealmente independientes, la distancia mínima es 3 por el mismo corolario. Podemos decir por tanto que los códigos de Hamming $\mathcal{H}_r$ son $[2^{r-1}, 2^{r-1-r}, 3]$ códigos binarios.

Podemos generalizar esta definición y definir los códigos de Hamming $\mathcal{H}_{q,r}$ sobre un cuerpo finito arbitrario $\mathbb{F}_q$. Para $r \geq 2$ un código $\mathcal{H}_{q,r}$ tiene matriz de paridad $H_{q,r}$, cuyas columnas están compuestas por un vector no nulo por cada uno de los subespacios de dimensión 1 de $\mathbb{F}_q^r$. Hay $(q^r - 1)/(q - 1)$ subespacios de dimensión 1, por lo que $\mathcal{H}_{q,r}$ tiene longitud $n = (q^r - 1)/(q - 1)$, dimensión $n - r$ y redundancia r. Como todas las columnas son independientes unas de otras, $\mathcal{H}_{q,r}$ tiene peso mínimo al menos 3. Si sumamos dos vectores no nulos de dos subespacios unidimensionales distintos obtenemos un vector no nulo de un tercer subespacio unidimensional, por lo que $\mathcal{H}_{q,r}$ tiene peso mínimo 3. Cuando $q = 2$, $\mathcal{H}_{2,r}$ es el código $\mathcal{H}_r$.

3

CÓDIGOS CÍCLICOS

En este capítulo vamos a estudiar los aspectos fundamentales de la clase de los códigos cíclicos, que representan la base del objeto de estudio de este trabajo. Los códigos cíclicos binarios fueron introducidos por primera vez en 1957 por Eugene Prange (Prange, 1957). Su importancia radica en la facilidad con la que pueden ser implementados en los circuitos digitales utilizando registros de desplazamiento. Las fuentes de este capítulo han sido (Huffman & Pless, 2003), (Kelbert & Suhov, 2013) y (MacWilliams & Sloane, 1977).

DEFINICIÓN 3.0.1. Un código lineal $\mathcal{C}$ de longitud n sobre $\mathbb{F}_q$ es *cíclico* si para cada vector $\mathbf{c} = c_0 \dots c_{n-2} c_{n-1}$ en $\mathcal{C}$, el vector $c_{n-1} c_0 \dots c_{n-2}$ —obtenido a partir de $\mathbf{c}$ desplazando cíclicamente las coordenadas, llevando $i \mapsto i+1$ mód n— también está en $\mathcal{C}$.

Al trabajar con códigos cíclicos pensamos en la posición de las coordenadas de forma cíclica, pues al llegar a $n-1$ se comienza de nuevo en 0. Al hablar de «coordenadas consecutivas» siempre tendremos en cuenta esta ciclicidad. Representaremos las palabras código de los códigos cíclicos como polinomios, pues podemos definir de forma natural una biyección entre el vector $\mathbf{c} = c_0 c_1 \dots c_{n-1}$ en $\mathbb{F}_q$ y los polinomios de la forma $c(x) = c_0 + c_1 x + \dots c_{n-1} x^{n-1}$ en $\mathbb{F}_q[x]$ de grado al menos $n-1$. Denotaremos a esta biyección por $\mathfrak{v}$, de forma análoga a como hicimos en el capítulo 1. Obtenemos así un isomorfismo entre $\mathbb{F}_q$-espacios vectoriales. Obsérvese que dado un polinomio $c(x)$ descrito como antes, el polinomio $xc(x) = c_{n-1} x^n + c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1}$ equivale a representar la palabra código $\mathbf{c}$ desplazada una posición a la derecha, siempre que x^n fuese igual a 1.

Formalmente, el hecho de que un código $\mathcal{C}$ sea invariante bajo un desplazamiento cíclico implica que si $c(x)$ está en $\mathcal{C}$, también ha de estar $xc(x)$, siempre que multipliquemos módulo $x^n - 1$. Esto nos sugiere que el contexto adecuado para estudiar códigos cíclicos es el anillo cociente $\mathcal{R}_n = \mathbb{F}_q[x]/(x^n - 1)$.

Por tanto, bajo la correspondencia vectores-polinomios que hemos descrito antes, los códigos cíclicos son ideales de $\mathcal{R}_n$, y los ideales de $\mathcal{R}_n$ son códigos cíclicos. En consecuencia, el estudio de los códigos cíclicos en $\mathbb{F}_q^n$ es equivalente al estudio de los ideales en $\mathcal{R}_n$, que va a depender de la factorización de $x^n - 1$ y por tanto lo vamos a abordar a continuación.

3.1 FACTORIZACIÓN DE $x^n - 1$

A la hora de factorizar $x^n - 1$ existen dos posibilidades, pues dicha factorización puede tener factores irreducibles repetidos o no. Vamos a asumir que q y n son primos relativos y por tanto $x^n - 1$ no tiene factores repetidos; en caso contrario el anillo cociente sería semisimple, lo que no nos aportaría nada: los ideales generados por los polinomios con factores repetidos no aumentan la distancia, pues es la misma que la del subcódigo sin factores repetidos.

Para factorizar $x^n - 1$ sobre $\mathbb{F}_q$ necesitamos considerar una extensión de cuerpos que contenga todas sus raíces, es decir, que sea lo que se conoce como *cuerpo de descomposición* del polinomio $x^n - 1$. Sabemos por el teorema 1.2.14 que una extensión $\mathbb{F}_{q^t}/\mathbb{F}_q$ contiene a una raíz enésima primitiva de la unidad cuando $n \mid (q^t - 1)$.

DEFINICIÓN 3.1.1. El *orden* $\mathrm{ord}_n(q)$ de q módulo n como el menor entero positivo a tal que $q^a \equiv 1 \bmod n$.

Si llamamos $t = \mathrm{ord}_n(q)$ entonces el cuerpo $\mathbb{F}_{q^t}$ contiene una enésima raíz primitiva de la unidad α, pero va a existir una extensión de cuerpos más pequeña de $\mathbb{F}_q$ que la contenga. Pero además, como todos los α^i son distintos dos a dos para $0 \leq i < n$ y $(\alpha^i)^n = 1$, el cuerpo $\mathbb{F}_{q^t}$ contiene de hecho a todas las raíces de $x^n - 1$ y es el cuerpo de descomposición que buscamos.

Los factores irreducibles de $x^n - 1$ sobre $\mathbb{F}_q$ deben ser el producto de los distintos polinomios minimales de las enésimas raíces de la unidad en $\mathbb{F}_{q^t}$. Sea γ un elemento primitivo de $\mathbb{F}_{q^t}$. Entonces $\alpha = \gamma^d$ es una enésima raíz primitiva de la unidad, donde $d = (q^t - 1)/n$. Y por tanto las raíces enésimas de la unidad vendrán dadas por $\alpha, \alpha^2, \ldots, \alpha^n = 1$. En definitiva, tenemos que buscar los polinomios minimales correspondientes a las raíces enésimas de la unidad y tomar el producto de todos ellos, evitando repetir aquellos que coincidan.

Las raíces del polinomio $M_{a^s}(x)$ son

$$\{\gamma^{ds}, \gamma^{dsq}, \gamma^{dsq^2}, \ldots, \gamma^{dsq^{r-1}}\} = \{\alpha^s, \alpha^{sq}, \alpha^{sq^2}, \ldots, \alpha^{sq^{r-1}}\},$$

donde r es el menor entero positivo tal que $dsq^r \equiv ds \bmod q^t - 1$ por el teorema 1.2.22. Pero $dsq^r \equiv ds \bmod q^t - 1$ si y solo si $sq^r \equiv s \bmod n$. Todo esto nos lleva a extender la definición de clases q-ciclotómicas que hemos introducido en la sección 1.2.4.

DEFINICIÓN 3.1.2. Sea s un entero tal que $0 \leq s < n$. La *clase q-ciclotómica de s módulo n* es el conjunto

$$C_s = \{s, sq, \ldots, sq^{r-1}\} \bmod n,$$

donde r es el menor entero positivo tal que $sq^r \equiv s$ mód n.

Se deduce entonces que C_s es la órbita de la permutación $i \mapsto iq$ mód n que contiene a s. Las distintas clases q-ciclotómicas módulo n dividen el conjunto de enteros $\{0, 1, 2, \ldots, n-1\}$. En la sección 1.2.4 estudiamos el caso particular en el que $n = q^t - 1$. Obsérvese que $\mathrm{ord}_n(q)$ es el tamaño de la clase q-ciclotómica C_1 módulo n.

Toda esta discusión nos conduce al siguiente teorema, cuya demostración ya hemos comentado parcialmente.

TEOREMA 3.1.3. *Sea n un entero positivo primo relativo con q y sea $t = \mathrm{ord}_n(q)$. Sea α una raíz enésima primitiva de la unidad en $\mathbb{F}_{q^t}$. Se verifican las siguientes afirmaciones.*

1. *Para cada entero s tal que $0 \le s < n$ el polinomio minimal de α^s sobre $\mathbb{F}_q$ es*

$$M_{\alpha^s}(x) = \prod_{i \in C_s} (x - \alpha^i),$$

   donde C_s es la clase q-ciclotómica de s módulo n.
2. *Los conjugados de α^s son los elementos α^i con $i \in C_s$.*
3. *Se tiene que*

$$x^n - 1 = \prod_s M_{\alpha^s}(x)$$

   es la factorización de $x^n - 1$ en factores irreducibles sobre $\mathbb{F}_q$, donde s varía en un conjunto de representantes de las clases q-ciclotómicas módulo n.

Demostración. Veamos la demostración por apartados.

1. Para la construcción de $M_\alpha(x)$ que hemos dado tenemos que comprobar tres cosas:
   - Que α^s es raíz. Esto es trivial, pues $s \in C_s$.
   - Que $M_\alpha(x) \in \mathbb{F}_q[x]$. Se tiene que

$$(M_\alpha(x))^q = \prod_{i \in C_s} (x - \alpha^i)^q = \prod_{i \in C_s} (x^q - \alpha^{iq}) = \prod_{i \in C_s} (x^q - \alpha^i) = M_\alpha(x^q).$$

   Por tanto, si $M_\alpha(x) = \sum_{i=0}^{r} a_i x^i$ entonces

$$(M_\alpha(x))^q = \left(\sum_{i=0}^{r} a_i x^i \right)^q = \sum_{i=0}^{r} a_i^q x^{iq}$$

   y

$$M_\alpha(x^q) = \sum_{i=0}^{r} a_i x^{iq}.$$

Así que $a_i^q = a_i$ para todo $0 \leq i \leq r$ y por tanto los coeficientes $a_i \in \mathbb{F}_q$, por lo que $M_\alpha(x) \in \mathbb{F}_q$.

— Que el $M_\alpha(x)$ no tiene raíces múltiples y que todo polinomio $f(x)$ que tenga a α^s como raíz es divisible por él. Como α es un elemento primitivo se tiene que $\alpha^j \neq \alpha^k$ para $j, k \in C_s$. Por tanto, no tiene raíces múltiples. Por otro lado, sea $f(x) = f_0 + f_1 x + \cdots + f_n x^n \in \mathbb{F}_q[x]$ tal que $f(\alpha^s) = 0$. Entonces para cada $j \in C_s$ existe un entero l tal que $j \equiv sq^l \mod n$. Así,

$$\begin{aligned} f(\alpha^j) = f(\alpha^{sq^l}) &= f_0 + f_1\alpha^{sq^l} + \cdots + f_n\alpha^{nsq^l} \\ &= f_0^{q^l} + f_1^{q^l}\alpha^{sq^l} + \cdots + f_n^{q^l}\alpha^{nsq^l} \\ &= (f_0 + f_1\alpha^i + \cdots + f_n\alpha^{ni})^{q^t} \\ &= f(\alpha^s)^{q^t} \\ &= 0. \end{aligned}$$

Por tanto $M_\alpha(x)$ así definido es un divisor de $f(x)$.

2. Queda probado con la demostración de la factorización de $M_\alpha(x)$ del apartado anterior.
3. Como ya hemos comentado antes, sabemos que en este caso $x^n - 1$ no tiene factores repetidos. Por tanto, su factorización en factores irreducibles ha de ser el producto de los distintos polinomios minimales correspondientes a las raíces enésimas de la unidad, pues, por definición, son irreducibles. □

EJEMPLO 3.1.4. Sean $q = 2$ y $n = 15$. Consideremos entonces el polinomio $x^{15} - 1$ sobre $\mathbb{F}_2$. Como $\mathrm{ord}_{15}(2) = 4$ el cuerpo de descomposición de $x^{15} - 1$ sobre $\mathbb{F}_2$ es $\mathbb{F}_{16}$. Vamos a utilizar SageMath para obtener la descomposición de $x^{15} - 1$ en factores irreducibles. Calculamos primero las clases 2-ciclotómicas módulo n.

```
sage: F = GF(2)
sage: x = polygen(F)
sage: Zmod(15).cyclotomic_cosets(2)
> [[0], [1, 2, 4, 8], [3, 6, 9, 12], [5, 10], [7, 11, 13,
    14]]
```

Una vez conocidas podemos obtener todos los polinomios minimales.

```
sage: (x-a^0)
> x + 1
sage: (x-a)*(x-a^2)*(x-a^4)*(x-a^8)
> x^4 + x + 1
```

```
sage: (x-a^3)*(x-a^6)*(x-a^9)*(x-a^12)
> x^4 + x^3 + x^2 + x + 1
sage: (x-a^5)*(x-a^10)
> x^2 + x + 1
sage: (x-a^7)*(x-a^11)*(x-a^13)*(x-a^14)
> x^4 + x^3 + 1
```

Podemos comprobar que efectivamente estos polinomios nos dan una descomposición de $x^{15} - 1$.

```
sage: (x-a^0)*(x-a)*(x-a^2)*(x-a^4)*(x-a^8)*(x-a^3)*(x-a
    ^6)*(x-a^9)*(x-a^12)*(x-a^5)*(x-a^10)*(x-a^7)*(x-a^11)
    *(x-a^13)*(x-a^14)
> x^15 + 1
```

3.2 CONSTRUCCIÓN DE CÓDIGOS CÍCLICOS

Una vez factorizado $x^n - 1$ vamos a ver que hay una correspondencia biyectiva entre sus polinomios divisores mónicos y los códigos cíclicos en $\mathcal{R}_n$. El siguiente teorema es el resultado fundamental de códigos cíclicos que nos va a permitir describirlos.

TEOREMA 3.2.1. *Sea C un ideal de $\mathcal{R}_n$, es decir, un código cíclico de longitud n. Entonces:*

1. *Existe un único polinomio mónico $g(x)$ de grado mínimo en C.*
2. *El polinomio descrito en (1) genera C, es decir, $C = \langle g(x) \rangle$.*
3. *El polinomio descrito en (1) verifica que $g(x) \mid x^n - 1$.*

Sea $k = n - \operatorname{gr} g(x)$ y sea $g(x) = \sum_{i_0}^{n-k} g_i x^i$, donde $g_{n-k} = 1$. Entonces:

4. *Se verifica que*

$$C = \langle g(x) \rangle = \{ f(x)g(x) : \operatorname{gr} f(x) < k \}.$$

5. *El conjunto $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ es una base de C y C tiene dimensión k.*
6. *La matriz G dada por*

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & \cdots & g_r & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k} \end{bmatrix},$$

donde cada fila es un desplazamiento cíclico de la fila previa, es una matriz generadora de $\mathcal{C}$.

7. *Si α es una enésima raíz primitiva de la unidad en alguna extensión de cuerpos de $\mathbb{F}_q$ entonces*

$$g(x) = \prod_s M_{\alpha^s}(x),$$

siendo dicho producto sobre un subconjunto de representantes de las clases q-ciclotómicas módulo n.

Demostración. Veamos la demostración apartado por apartado.

1. Supongamos que $\mathcal{C}$ contiene dos polinomios mónicos distintos, $g_1(x)$ y $g_2(x)$, ambos de grado mínimo r. Entonces, $g_1(x) - g_2(x)$ es un polinomio no nulo de grado menor que r, lo que es absurdo. Existe por tanto un único polinomio de grado mínimo r en $\mathcal{C}$, como queríamos.
2. Como $g(x) \in \mathcal{C}$ y $\mathcal{C}$ es un ideal, tenemos que $\langle g(x)\rangle \subset \mathcal{C}$. Por otra parte, dado $p(x) \in \mathcal{C}$ el algoritmo de división nos da elementos $q(x), r(x)$ tales que $p(x) = q(x)g(x) + r(x)$, de forma que o bien $r(x) = 0$ o bien $\operatorname{gr} r(x) < \operatorname{gr} g(x)$. Como podemos expresar $r(x)$ de la forma $r(x) = p(x) - q(x)g(x) \in \mathcal{C}$ y tiene grado menor que $\operatorname{gr} g(x)$, al ser este último de grado mínimo necesariamente ha de darse que $r(x) = 0$. Por tanto, $p(x) = q(x)g(x) \in \langle g(x)\rangle$ y $\mathcal{C} \subset \langle g(x)\rangle$. En consecuencia, $\langle g(x)\rangle = \mathcal{C}$.
3. Por el algoritmo de división, al dividir $x^n - 1$ por $g(x)$ tenemos que $x^n - 1 = q(x)g(x) + r(x)$. De nuevo, o bien $r(x) = 0$ o bien $\operatorname{gr} r(x) < \operatorname{gr} g(x)$. Como en $\mathcal{R}_n$ se tiene que $x^n - 1 = 0 \in \mathcal{C}$, necesariamente $r(x) \in \mathcal{C}$. Esto supone una contradicción, a menos que $r(x) = 0$. En consecuencia, $g(x) \mid x^n - 1$.
4. El ideal generado por $g(x)$ es $\langle g(x)\rangle = \{f(x)g(x) : f(x) \in \mathcal{R}_n\}$. Queremos ver que podemos restringir los polinomios $f(x)$ a aquellos que tengan grado menor que k. Por (3) sabemos que $x^n - 1 = h(x)g(x)$ para algún polinomio $h(x)$ que tenga grado $k = n - \operatorname{gr} g(x)$. Dividimos entonces $f(x)$ por este polinomio $h(x)$ y por el algoritmo de división obtenemos $f(x) = q(x)h(x) + r(x)$, donde $\operatorname{gr} r(x) < \operatorname{gr} h(x) = k$. Entonces, tenemos

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

luego $f(x)g(x) = r(x)g(x)$, y puesto que antes ya hemos visto que $\operatorname{gr} r(x) < k$, hemos obtenido lo que buscábamos.

5. A partir de (5) tenemos que el conjunto

$$\{g(x), xg(x), \dots, x^{k-1}g(x)\}$$

genera $\mathcal{C}$, y como es linealmente independiente, forma una base de $\mathcal{C}$. Esto demuestra también que la dimensión de $\mathcal{C}$ es k.
6. La matriz G es matriz generadora de $\mathcal{C}$ pues

$$\{g(x), xg(x), \dots, x^{k-1}g(x)\}$$

es una base de $\mathcal{C}$.
7. Se deduce del teorema 3.1.3 y de (3). □

Este teorema nos proporciona una forma de obtener los códigos cíclicos de longitud n a partir de los divisores del polinomio $x^n - 1$ así como describir una matriz generadora de dichos códigos a partir de ellos. Vamos a ver a continuación que el polinomio mónico divisor de $x^n - 1$ que genera a un código cíclico $\mathcal{C}$ es único.

COROLARIO 3.2.2. *Sea $\mathcal{C}$ un código cíclico en $\mathcal{R}_n$ distinto de cero. Son equivalentes:*

1. *El polinomio $g(x)$ es el polinomio mónico de menor grado en $\mathcal{C}$.*
2. *Podemos expresar $\mathcal{C}$ como $\mathcal{C} = \langle g(x) \rangle$, $g(x)$ es mónico y $g(x) \mid (x^n - 1)$.*

Demostración. Que (1) implica (2) ya lo hemos probado en el teorema 3.2.1. Veamos que partiendo de (2) obtenemos (1). Sea $g_1(x)$ el polinomio mónico de menor grado en $\mathcal{C}$. Por el teorema 3.2.1, $g_1(x) \mid g(x)$ en $\mathbb{F}_q[x]$ y $\mathcal{C} = \langle g_1(x) \rangle$. Como $g_1(x) \in \mathcal{C} = \langle g(x) \rangle$, podemos expresarlo como $g_1(x) = g(x)a(x)$ mód $x^n - 1$, luego tenemos que $g_1(x) = g(x)a(x) + (x^n - 1)b(x)$ en $\mathbb{F}_q[x]$. Por otro lado, como $g(x) \mid (x^n - 1)$, tenemos que $g(x) \mid g(x)a(x) + (x^n - 1)b(x)$, o lo que es lo mismo, que $g(x) \mid g_1(x)$. En consecuencia, como $g_1(x)$ y $g(x)$ son ambos mónicos y dividen el uno al otro en $\mathbb{F}_q[x]$, son necesariamente iguales. □

A este polinomio $g(x)$ lo llamamos *polinomio generador* del código cíclico $\mathcal{C}$. Por el corolario anterior, este polinomio es tanto el polinomio mónico en $\mathcal{C}$ de grado mínimo como el polinomio mónico que divide a $x^n - 1$ y genera a $\mathcal{C}$. Existe por tanto una correspondencia biunívoca entre los códigos cíclicos distintos de cero y los divisores de $x^n - 1$ distintos de él mismo. Para extender dicha correspondencia entre todos los códigos cíclicos en $\mathcal{R}_n$ y todos los divisores mónicos de $x^n - 1$ definimos como polinomio generador del código cíclico $\{\mathbf{0}\}$ el polinomio $x^n - 1$. Esta correspondencia biyectiva nos conduce al siguiente corolario.

COROLARIO 3.2.3. *El número de códigos cíclicos en $\mathcal{R}_n$ es 2^m, donde m es el número de clases q-ciclotómicas módulo n.*

Ahora mismo todo este desarrollo puede parecer demasiado abstracto. Vamos a ver un ejemplo exhaustivo para entender cómo podemos obtener los polinomios generadores de los códigos cíclicos de una longitud arbitraria y cómo éstos son generados a partir de ellos.

EJEMPLO 3.2.4. Vamos a describir todos los códigos cíclicos binarios de longitud 7. Para ello vamos a utilizar el código descrito en el anexo B tal y como mostramos en el listado siguiente.

```
1 sage: F = GF(2)
2 sage: x = polygen(F)
3 sage: (x^7 - 1).factor()
4 > (x + 1) * (x^3 + x + 1) * (x^3 + x^2 + 1)
5 sage: print(generators(x^7 - 1))
6 > [1, x + 1, x^3 + x + 1, x^3 + x^2 + 1, x^4 + x^3 + x^2 +
      1, x^4 + x^2 + x + 1, x^6 + x^5 + x^4 + x^3 + x^2 + x
     + 1, x^7 + 1]
```

Así sobre $\mathbb{F}_2$ podemos descomponer $x^7 - 1$ como

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

y los 8 polinomios generadores, todos los divisores de $x^7 - 1$, son:

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

Vamos a ver qué códigos generan estos polinomios:

1. La dimensión del código es $k = 7 - 0 = 7$, luego el código generado es un $[7, 7]$-código lineal, que es evidentemente $\mathbb{F}_2^7$. La matriz generadora que nos proporciona el teorema 3.2.1(6) es

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

2. La dimensión del código es $k = 7 - 1 = 6$, luego el código generado es un $[7, 6]$-código lineal. La matriz generadora que nos proporciona el teorema 3.2.1(6) es

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

   Comprobamos que la matriz de paridad es

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

   y por tanto el código obtenido es un código de control de paridad.
3. La dimensión del código es $k = 7 - 3 = 4$, luego el código generado es un $[7, 4]$-código lineal. La matriz generadora que nos proporciona el teorema 3.2.1(6) es

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

   La matriz de paridad en este caso es

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

y por tanto el código generado es un $\mathcal{H}_3$ código de Hamming.

4. La dimensión del código es $k = 7 - 3 = 4$, luego el código generado es un $[7, 4]$-código lineal. La matriz generadora que nos proporciona el teorema 3.2.1(6) es

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

La matriz de paridad en este caso es

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

y por tanto el código generado es un $\mathcal{H}_3$ código de Hamming.

5. La dimensión del código es $k = 7 - 4 = 3$, luego el código generado es un $[7, 3]$-código lineal. La matriz generadora que nos proporciona el teorema 3.2.1(6) es

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

La matriz de paridad en este caso es

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

y por tanto el código generado es un $\mathcal{H}_4$ código de Hamming.

6. La dimensión del código es $k = 7 - 4 = 3$, luego el código generado es un $[7, 3]$-código lineal. La matriz generadora que nos proporciona el teorema 3.2.1(6) es

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

La matriz de paridad en este caso es

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

y por tanto el código generado es un $\mathcal{H}_4$ código de Hamming.

7. La dimensión del código es $k = 7 - 6 = 1$, luego el código generado es un $[7, 1]$-código lineal. La matriz generadora que nos proporciona el teorema 3.2.1(6) es

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

por lo que concluimos que el código generado es el código de repetición de longitud 7.

8. La dimensión del código es $k = 7 - 7 = 0$, luego el código generado es $\{\mathbf{0}\}$.

Finalmente, el siguiente resultado nos muestra la relación entre dos polinomios generadores cuando un código es subcódigo de otro.

COROLARIO 3.2.5. *Sean $\mathcal{C}_1$ y $\mathcal{C}_2$ códigos cíclicos sobre $\mathbb{F}_q$ con polinomios generadores $g_1(x)$ y $g_2(x)$, respectivamente. Entonces, $\mathcal{C}_1 \subseteq \mathcal{C}_2$ si y solo si $g_2(x) \mid g_1(x)$.*

Demostración. Recordamos que por el teorema 3.2.1(4) todos los elementos $a(x)$ de los códigos cíclicos $\mathcal{C}_1$ y $\mathcal{C}_2 \in \mathcal{R}_n$ pueden expresarse como $a(x) = g_i(x)f_i(x)$ —donde, o bien $f_i(x) = 0$, o bien $\operatorname{gr}(f_i(x)) < k = n - \operatorname{gr}(g_i(x))$— para $i = 1, 2$ respectivamente. Veamos ambas implicaciones por separado.

1. Comenzamos con que si $g_2(x) | g_1(x)$ entonces $\mathcal{C}_1 \subseteq \mathcal{C}_2$. Por hipótesis podemos expresar $g_1(x) = r(x)g_2(x)$ para algún polinomio $r(x)$. Así, todo elemento $a(x)$ de $\mathcal{C}_1$ puede expresarse como $g_1(x)f_1(x) = r(x)g_2(x)f_1(x) = g_2(x)f_2(x)$ para algún $f_2(x)$, por lo que si $a(x) \in \mathcal{C}_1$, $a(x) \in \mathcal{C}_2$. Por tanto, $\mathcal{C}_1 \subseteq \mathcal{C}_2$.
2. Vemos a continuación que si $\mathcal{C}_1 \subseteq \mathcal{C}_2$ entonces $g_2(x) | g_1(x)$. Vamos a usar un argumento similar al anterior. Como $\mathcal{C}_1 \subseteq \mathcal{C}_2$ todo elemento de $\mathcal{C}_1$ puede expresarse como $g_1(x)f_1(x) = g_2(x)f_2(x)$ para ciertos $f_1(x)$, $f_2(x)$. Por tanto, $g_1(x) = g_2(x)f_2(x)/f_1(x)$ y en consecuencia, $g_2 | g_1(x)$, como queríamos. □

3.3 CODIFICACIÓN DE CÓDIGOS CÍCLICOS

Vamos a ver a continuación dos tipos de codificación de códigos cíclicos. Consideraremos un código cíclico $\mathcal{C}$ de longitud n sobre $\mathbb{F}_q$ con polinomio generador $g(x)$ de grado $n-k$, por lo que $\mathcal{C}$ tiene dimensión k.

CODIFICACIÓN NO-SISTEMÁTICA Esta forma de codificación está basada en la técnica natural de codificación que describimos en la sección 2.1.1. Sea G la matriz generadora obtenida a partir de los desplazamientos de $g(x)$ descrita en el teorema 3.2.1. Dado el mensaje $\mathbf{m} \in \mathbb{F}_q^k$, lo codificamos como la palabra código $\mathbf{c} = \mathbf{m}G$. De igual forma, si $m(x)$ y $c(x)$ son los polinomios en $\mathbb{F}_q[x]$ asociados a $\mathbf{m}$ y $\mathbf{c}$, entonces $c(x) = m(x)g(x)$.

CODIFICACIÓN SISTEMÁTICA El polinomio $m(x)$ asociado a un mensaje $\mathbf{m}$ tendrá como mucho grado $k-1$. Por tanto, el polinomio $n^{n-k}m(x)$ tendrá como mucho grado $n-1$ y sus primeros $n-k$ coeficientes son nulos. Por tanto, el mensaje está contenido en los coeficientes de $x^{n-k}, x^{n-k+1}, \dots, x^{n-1}$. Por el algoritmo de división tenemos que

$$x^{n-k}m(x) = g(x)a(x) + r(x), \qquad \text{donde } \operatorname{gr} r(x) < n-k \text{ o } r(x) = 0.$$

Sea $c(x) = x^{n-k}m(x) - r(x)$. Como $c(x)$ es múltiplo de $g(x)$, $c(x) \in \mathcal{C}$. El polinomio $c(x)$ difiere de $x^{n-k}m(x)$ en los coeficientes de $1, x, \dots, x^{n-k-1}$ ya que $\operatorname{gr} r(x) < n-k$. Por tanto, $c(x)$ contiene el mensaje $\mathbf{m}$ en los coeficientes de los términos de grado al menos $n-k$.

EJEMPLO 3.3.1. Sea $\mathcal{C}$ un código cíclico de longitud 15 con polinomio generador $g(x) = (1+x+x^4)(1+x+x^2+x^3+x^4)$. Supongamos que queremos codificar el mensaje $m(x) = 1 + x^2 + x^5$. Vamos a ver su codificación con los dos métodos descritos. Como la longitud de $\mathcal{C}$ es 15 y el grado de su polinomio generador es 8, la dimensión del código es $15-8 = 7$. Escribimos el mensaje $m(x)$ en forma de vector: $\mathbf{m} = (1,0,1,0,0,1,0)$. Una matriz generadora del código $\mathcal{C}$ es:

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

1. Codificación no-sistemática. Simplemente multiplicamos **m** por G, obteniendo:

$$\mathbf{c} = \mathbf{m}G = (1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0)\,.$$

2. Codificación sistemática. Calculamos el cociente de $x^{n-k}m(x)$ por $g(x)$ para obtener el resto $r(x) = x^6 + x + 1$. Entonces, la palabra código viene dada por $c(x) = x^{n-k}m(x) - r(x) = x^{13} + x^{10} + x^8 + x^6 + x + 1$, que en forma de vector resulta $\mathbf{c} = (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 0)$. Observamos que la codificación es efectivamente sistemática: nuestro mensaje m está contenido íntegramente en las últimas 7 coordenadas.

3.4 IDEMPOTENTES Y MULTIPLICADORES

En esta sección vamos a estudiar otra forma alternativa de generar los códigos cíclicos en $\mathcal{R}_n$. Se basará en encontrar unos elementos concretos de $\mathcal{R}_n$ que además podremos relacionar con los polinomios generadores que hemos definido hasta ahora.

Como ya vimos en la sección 1.1 un elemento e de un anillo es idempotente si $e^2 = e$. Partiendo de la suposición de que $\mathrm{mcd}(n, q) = 1$ afirmamos que el anillo $\mathcal{R}_n$ es semisimple. Esto implica, además de lo que ya comentamos, que cada ideal de $\mathcal{R}_n$ tiene un único elemento idempotente que lo genera. Este elemento se denomina *idempotente generador* del código cíclico. En el siguiente teorema probaremos este hecho y mostraremos además un método para determinar el idempotente generador de un código cíclico a partir de su polinomio generador.

TEOREMA 3.4.1. *Sea $\mathcal{C}$ un código cíclico en $\mathcal{R}_n$. Entonces:*

1. *Existe un único idempotente $e(x) \in \mathcal{C}$ tal que $\mathcal{C} = \langle e(x)\rangle$.*
2. *Si $e(x)$ es un idempotente no nulo en $\mathcal{C}$, entonces $\mathcal{C} = \langle e(x)\rangle$ si y solo si $e(x)$ es una unidad de $\mathcal{C}$.*

Demostración. Si $\mathcal{C}$ es el código cero, entonces el idempotente es el cero, con lo que (1) está claro y (2) no se aplica a este caso. Veamos entonces la demostración por apartados suponiendo que $\mathcal{C}$ es distinto de cero.

1. Supongamos primero que $e(x)$ es una unidad en $\mathcal{C}$. Entonces, $\langle e(x)\rangle \subset \mathcal{C}$, ya que $\mathcal{C}$ es un ideal. Si $c(x) \in \mathcal{C}$, entonces $c(x)e(x) = c(x)$ en $\mathcal{C}$. En consecuencia, $\langle e(x)\rangle = \mathcal{C}$. Por otro lado, supongamos que $e(x)$ es un idempotente distinto de cero y tal que $\mathcal{C} = \langle e(x)\rangle$. Entonces, cada elemento $c(x)$ lo podemos escribir como $c(x) = f(x)e(x)$. Pero se tiene que $c(x)e(x) = f(x)(e(x))^2 = f(x)e(x) = c(x)$, luego $e(x)$ es la unidad de $\mathcal{C}$.

2. Tenemos que probar la existencia y la unicidad. Comenzamos con la existencia. Sea $g(x)$ el polinomio generador dde $\mathcal{C}$. Entonces, sabemos que $g(x) \mid (x^n - 1)$ por el teorema 3.2.1. Tomemos $h(x) = (x^n - 1)/g(x)$. Sabemos que $\operatorname{mcd}(g(x), h(x)) = 1$ en $\mathbb{F}_q[x]$, ya que $x^n - 1$ tiene todas sus raíces distintas. En consecuencia, el algoritmo de Euclides nos proporciona los polinomios $a(x), b(x) \in \mathbb{F}_q[x]$ tales que $a(x)g(x) + b(x)h(x) = 1$. Llamemos $e(x) \equiv a(x)g(x) \bmod x^n - 1$, que será el representante de dicha clase de equivalencia en $\mathcal{R}_n$. Entonces, en $\mathcal{R}_n$,

$$\begin{aligned} e(x)^2 &\equiv (a(x)g(x))(1 - b(x)h(x)) \bmod x^n - 1 \\ &\equiv a(x)g(x) - a(x)g(x)b(x)h(x) \bmod x^n - 1 \\ &\equiv a(x)g(x) - a(x)b(x)(x^n - 1) \bmod x^n - 1 \\ &\equiv a(x)g(x) \bmod x^n - 1 \\ &\equiv e(x) \bmod x^n - 1. \end{aligned}$$

Por tanto, este elemento $e(x)$ es idempotente. Veamos ahora que si $c(x) \in \mathcal{C}$, entonces $c(x) = f(x)g(x)$, luego

$$\begin{aligned} c(x)e(x) &= f(x)g(x)(1 - b(x)h(x)) \\ &\equiv f(x)g(x) \bmod x^n - 1 \\ &\equiv c(x) \bmod x^n - 1, \end{aligned}$$

por lo que $e(x)$ es una unidad en $\mathcal{C}$. En consecuencia, podemos deducir la existencia a partir de (2). Veamos ahora la unicidad. Por (2), si tenemos dos elementos idempotentes $e_1(x)$ y $e_2(x)$ que generan $\mathcal{C}$, ambos han de ser unidades, y en consecuencia se tiene que $e_1(x) = e_1(x)e_2(x) = e_2(x)$, con lo que podemos deducir la unicidad. □

Deducimos por tanto que un método para encontrar el idempotente generador $e(x)$ de un código cíclico $\mathcal{C}$ a partir del polinomio generador $g(x)$ es resolver la ecuación

$$1 = a(x)g(x) + b(x)h(x)$$

para $a(x)$ utilizando el algoritmo de Euclides, donde $h(x) = (x^n - 1)/g(x)$. Entonces, reduciendo $a(x)g(x)$ módulo $x^n - 1$ obtenemos el idempotente $e(x)$ que buscamos. Pero vamos a ver además esta relación a la inversa, es decir, que podemos obtener el polinomio generador $g(x)$ a partir del idempotente $e(x)$.

TEOREMA 3.4.2. *Sea $\mathcal{C}$ un código cíclico sobre $\mathbb{F}_q$ con idempotente generador $e(x)$. Entonces, el polinomio generador de $\mathcal{C}$ es $g(x) = \operatorname{mcd}(e(x), x^n - 1)$, calculado en $\mathbb{F}_q[x]$.*

Demostración. Sea $d(x) = \text{mcd}(e(x), x^n - 1)$ en $\mathbb{F}_q[x]$ y sea $g(x)$ el polinomio generador de $\mathcal{C}$. Como $d(x) \mid e(x)$, podemos expresarlo como $e(x) = d(x)k(x)$ para algún $k(x) \in \mathbb{F}_q[x]$. Por tanto cada elemento de $\mathcal{C} = \langle e(x) \rangle$ es también múltiplo de $d(x)$, por lo que $\mathcal{C} \subset \langle d(x) \rangle$. Por el teorema 3.2.1 tenemos que en $\mathbb{F}_q[x]$, $g(x) \mid (x^n - 1)$ y que $g(x) \mid e(x)$, ya que $e(x) \in \mathcal{C}$. Luego, por la proposición 1.2.7 tenemos que $g(x) \mid d(x)$ y en consecuencia $d(x) \in \mathcal{C}$. Por tanto, $\langle d(x) \rangle \subseteq \mathcal{C}$ y deducimos entonces que $\mathcal{C} = \langle d(x) \rangle$. Como $d(x)$ es divisor mónico de $x^n - 1$ y genera a $\mathcal{C}$, necesariamente $d(x) = g(x)$ por el corolario 2.1.23. □

EJEMPLO 3.4.3. Continuando con el ejemplo 3.2.4 en el que describimos todos los códigos cíclicos binarios de longitud 7 vamos a indicar a continuación cuales son los idempotentes generadores de cada uno. Para ello vamos a utilizar el código descrito en el anexo B tal y como mostramos en el listado siguiente.

```
1  sage: F = GF(2)
2  sage: x = polygen(F)
3  sage: (x^7 - 1).factor()
4  > (x + 1) * (x^3 + x + 1) * (x^3 + x^2 + 1)
5  sage: print(generator_and_idempotents(x^7 - 1))
6  > [(1, 1),
7     (x + 1, x^6 + x^5 + x^4 + x^3 + x^2 + x),
8     (x^3 + x + 1, x^4 + x^2 + x),
9     (x^3 + x^2 + 1, x^6 + x^5 + x^3),
10    (x^4 + x^3 + x^2 + 1, x^6 + x^5 + x^3 + 1),
11    (x^4 + x^2 + x + 1, x^4 + x^2 + x + 1),
12    (x^6 + x^5 + x^4 + x^3 + x^2 + x + 1, x^6 + x^5 + x^4 +
          x^3 + x^2 + x + 1),
13    (x^7 + 1, 0)]
```

En la tabla 3.1 mostramos de forma más clara cuáles son los generadores y cuáles los idempotentes correspondientes.

Puesto que los idempotentes generadores producen códigos cíclicos es de rigor preguntarse si a partir de los idempotentes podemos obtener una base de los códigos generados, tal y como ocurre con los polinomios generadores. El siguiente teorema nos dice que sí, y además de la misma forma: a partir de los primeros $k - 1$ desplazamientos cíclicos del idempotente generador.

i	dimensión	generador $g_i(x)$	idempotente $e_i(x)$
0	0	x^7+1	0
1	1	$x^6+x^5+\cdots+x+1$	$x^6+x^5+\cdots+x+1$
2	3	$x^4+x^3+x^2+1$	$x^6+x^5+x^3+1$
3	3	x^4+x^2+x+1	x^4+x^2+x+1
4	4	$x^3+x+x+1$	x^4+x^2+x
5	4	x^3+x^2+1	$x^6+x^5+x^3$
6	6	$x+1$	$x^6+x^5+\cdots+x$
7	7	1	1

Tabla 3.1: Polinomios generadores e idempotentes para los códigos cíclicos de longitud 7

TEOREMA 3.4.4. *Sea $\mathcal{C}$ un $[n,k]$ código cíclico con idempotente generador $e(x)=\sum_{i=0}^{n-1} e_i x^i$. Entonces, la matriz $k\times n$*

$$\begin{pmatrix} e_0 & e_1 & e_2 & \cdots & e_{n-2} & e_{n-1} \\ e_{n-1} & e_0 & e_1 & \cdots & e_{n-3} & e_{n-2} \\ & & & \vdots & & \\ e_{n-k+1} & e_{n-k+2} & e_{n-k+3} & \cdots & e_{n-k-1} & e_{n-k} \end{pmatrix}$$

es una matriz generadora de $\mathcal{C}$.

Demostración. Probar este resultado equivale a probar que el conjunto $\{e(x), xe(x), \ldots, x^{k-1}e(x)\}$ es una base de $\mathcal{C}$. Entonces, solo hay que probar que si $a(x)\in\mathbb{F}_q[x]$ tiene grado menor que k, tal que $a(x)e(x)=0$, se tiene que $a(x)=0$. Sea $g(x)$ el polinomio generador de $\mathcal{C}$. Si $a(x)e(x)=0$, entonces $0=a(x)e(x)g(x)=a(x)g(x)$, tal que $e(x)$ es la unidad de $\mathcal{C}$ según el teorema 3.4.1, y por tanto, si $a(x)$ no es cero estaríamos contradiciendo el teorema 3.2.1. □

El siguiente resultado nos informa sobre los polinomios generadores e idempotentes generadores de sumas e intersecciones de códigos cíclicos de la misma longitud. Dados dos códigos cíclicos $\mathcal{C}_1$ y $\mathcal{C}_2$ de longitud n sobre $\mathbb{F}_q$ definimos su suma como

$$\mathcal{C}_1+\mathcal{C}_2=\{\mathbf{c}_1+c_2(x):\mathbf{c}_1\in\mathcal{C}_1 \text{ y } \mathbf{c}_2\}.$$

TEOREMA 3.4.5. *Sean $\mathcal{C}_1$ y $\mathcal{C}_2$ códigos cíclicos de longitud n sobre $\mathbb{F}_q$ con polinomios generadores $g_1(x)$ y $g_2(x)$ e idempotentes generadores $e_1(x)$ y $e_2(x)$, respectivamente. Entonces*

1. *La intersección $\mathcal{C}_1\cap\mathcal{C}_2$ es también un código cíclico, con polinomio generador* $\operatorname{mcm}(g_1(x),g_2(x))$ *e idempotente generador $e_1(x)e_2(x)$.*

2. *La suma $\mathcal{C}_1 + \mathcal{C}_2$ es también un código cíclico, con polinomio generador* $\mathrm{mcd}(g_1(x), g_2(x))$ *e idempotente generador* $e_1(x) + e_2(x) - e_1(x)e_2(x)$.

Demostración. Veamos la demostración por apartados.

1. La intersección $\mathcal{C}_1 \cap \mathcal{C}_2$ es un subcódigo y por tanto, por el corolario 3.2.5 es un código cíclico. Por el mismo corolario su polinomio generador debe ser divisible por $g_1(x)$ y $g_2(x)$, por lo que ha de ser divisible por el $g(x) = \mathrm{mcm}(g_1(x), g_2(x))$. Así, $g(x)$ es un polinomio generador de un código cíclico que está contenido tanto en $\mathcal{C}_1$ como en $\mathcal{C}_2$. Por tanto, $g(x)$ ha de ser el generador de $\mathcal{C}_1 \cap \mathcal{C}_2$, pues si no lo fuese, el código cíclico generador por $g(x)$ ha de ser mayor que la intersección, lo que contradice la propia definición de intersección. Veamos ahora que el idempotente generador es $e_1(x)e_2(x)$. Claramente $e_1(x)e_2(x) \in \mathcal{C}_1 \cap \mathcal{C}_2$ y es idempotente, pues $(e_1(x)e_2(x))^2 = e_1(x)^2 e_2(x)^2 = e_1(x)e_2(x)$. Si $c(x) \in \mathcal{C}_1 \cap \mathcal{C}_2$ entonces $e_1(x)e_2(x)c(x) = e_1(x)c(x) = c(x)$, pues por el teorema 3.4.1(2) e_1 y e_2 son unidades de $\mathcal{C}_1$ y $\mathcal{C}_2$, respectivamente. El mismo teorema nos asegura entonces que $e_1(x)e_2(x)$ es el idempotente generador que buscamos.
2. Veamos primero que $\mathcal{C}_1 + \mathcal{C}_2$ es un código cíclico. Sabemos que si $c_1(x) \in \mathcal{C}_1$ y $c_2(x) \in \mathcal{C}_2$ entonces $xc_1(x) \in \mathcal{C}_1$ y $xc_2(x) \in \mathcal{C}_2$. Dado un elemento $c_1(x) + c_2(x) \in \mathcal{C}_1 + \mathcal{C}_2$ tenemos que $x(c_1(x) + c_2(x)) = xc_1(x) + xc_2(x) \in \mathcal{C}_1 + \mathcal{C}_2$, por lo que $\mathcal{C}_1 + \mathcal{C}_2$ es cíclico. A continuación, sea $g(x) = \mathrm{mcd}(g_1(x), g_2(x))$. El algoritmo de Euclides nos proporciona $a(x)$ y $b(x) \in \mathbb{F}_q[x]$ tales que $g(x) = g_1(x)a(x) + g_2(x)b(x)$. Por tanto, $g(x) \in \mathcal{C}_1 + \mathcal{C}_2$. Como $\mathcal{C}_1 + \mathcal{C}_2$ es cíclico, $\langle g(x)\rangle \subseteq \mathcal{C}_1 + \mathcal{C}_2$. Por otro lado $g(x) | g_1(x)$ luego por el corolario 3.2.5 $\mathcal{C}_1 \subseteq \langle g(x)\rangle$. De la misma forma deducimos que $\mathcal{C}_2 \subseteq \langle g(x)\rangle$ y por tanto $\mathcal{C}_1 + \mathcal{C}_2 \subseteq \langle g(x)\rangle$. Así, $\mathcal{C}_1 + \mathcal{C}_2 = \langle g(x)\rangle$. Se tiene que $g(x) | (x^n - 1)$ puesto que $g(x) | g_1(x)$. Además, como $g(x)$ es mónico, se tiene por el corolario 3.2.2 que $g(x) = \mathrm{mcd}(g_1(x), g_2(x))$ es el polinomio generador de $\mathcal{C}_1 + \mathcal{C}_2$. Veamos finalmente que dado $c(x) = c_1(x) + c_2(x)$, con $c_1 \in \mathcal{C}_1$ y $c_2 \in \mathcal{C}_2$ se tiene que

$$\begin{aligned} &c(x)(e_1(x) + e_2(x) - e_1(x)e_2(x)) \\ &\quad = c_1(x) + c_1(x)e_2(x) - c_1(x)e_2(x) + c_2(x)e_1(x) + c_2(x) - c_2(x)e_1(x) \\ &\quad = c_1(x) + c_2(x) \\ &\quad = c(x). \end{aligned}$$

Por tanto, por el teorema 3.4.1 obtenemos que $e_1(x)+e_2(x)-e_1(x)e_2(x) \in \mathcal{C}_1 + \mathcal{C}_2$ es el idempotente generador, como queríamos demostrar. □

Estamos ya en disposición de describir los elementos que prometimos al comienzo de la sección. Nos permitirán obtener todos los idempotentes en $\mathcal{R}_n$, y en consecuencia, todos los códigos cíclicos en $\mathcal{R}_n$. Son los conocidos como *idempotentes primitivos*.

Consideremos la descomposición en factores $x^n - 1 = f_1(x) \cdots f_s(x)$, donde cada polinomio $f_i(x)$ es irreducible sobre $\mathbb{F}_q$ para $1 \leq i \leq s$. Sabemos que los factores $f_i(x)$ son distintos, pues estamos en el supuesto de que $x^n - 1$ tiene raíces distintas. Sea $\widehat{f}_i(x) = (x^n - 1)/f_i(x)$. En el teorema 3.4.6 a continuación vamos a ver que los ideales $\langle \widehat{f}_i(x) \rangle$ de $\mathcal{R}_n$ son los ideales minimales de $\mathcal{R}_n$ y cómo podemos obtener $\mathcal{R}_n$ a partir de ellos. Al idempotente generador de $\langle \widehat{f}_i(x) \rangle$ lo denotaremos por $\widehat{e}_i(x)$. Los elementos idempotentes $\widehat{e}_1(x), \ldots, \widehat{e}_s(x)$ son los *idempotentes primitivos* de $\mathcal{R}_n$. El teorema 3.4.6 que sigue nos muestra además la forma de obtener todos los idempotentes de $\mathcal{R}_n$ a partir de los idempotentes primitivos.

TEOREMA 3.4.6. *En $\mathcal{R}_n$ se verifican las siguientes afirmaciones.*

1. *Los ideales $\langle \widehat{f}_i(x) \rangle$ para cada $1 \leq i \leq s$ son todos los ideales minimales de $\mathcal{R}_n$.*
2. *$\mathcal{R}_n$ es el espacio vectorial suma directa de todos los $\langle \widehat{f}_i(x) \rangle$ para $1 \leq i \leq s$.*
3. *Si $i \neq j$ entonces $\widehat{e}_i(x)\widehat{e}_j(x) = 0$ en $\mathcal{R}_n$.*
4.
5. *La suma $\sum_{i=1}^{s} \widehat{e}_i(x) = 1$ en $\mathcal{R}_n$.*
6. *Los únicos idempotentes en $\langle \widehat{f}_i(x) \rangle$ son 0 y $\widehat{e}_i(x)$.*
7. *Si $e(x)$ es un idempotente no nulo en $\mathcal{R}_n$, entonces existe un subconjunto T de $\{1, 2, \ldots, s\}$ tal que $e(x) = \sum_{i \in T} \widehat{e}_i(x)$ y $\langle e(x) \rangle = \sum_{i \in T} \langle \widehat{f}_i(x) \rangle$.*

Demostración. Veamos la demostración por apartados.

1. Veamos por reducción al absurdo que cada $\langle \widehat{f}_i(x) \rangle$ es un ideal minimal de $\mathcal{R}_n$. Supongamos que no es un ideal minimal. Entonces, por el corolario 3.2.5 existiría un polinomio generador $g(x)$ de un ideal no trivial contenido en $\langle \widehat{f}_i(x) \rangle$ tal que $\widehat{f}_i(x) | g(x)$, con $g(x) \neq \widehat{f}_i(x)$. Pero como $f_i(x)$ es irreducible y $g(x) | (x^n - 1)$, es imposible. Por tanto cada $\langle \widehat{f}_i(x) \rangle$ es un ideal minimal de $\mathcal{R}_n$. Veamos que estos son todos los ideales minimales de $\mathcal{R}_n$. Sea $\mathcal{M} = \langle m(x) \rangle$ un ideal minimal de $\mathcal{R}_n$. Como el conjunto $\{\widehat{f}_i(x) : 1 \leq i \leq s\}$ no tiene factores irreducibles de $x^n - 1$ repetidos y cada uno de ellos divide a $x^n - 1$ el $\operatorname{mcd}(\widehat{f}_1(x), \ldots, \widehat{f}_s(x)) = 1$. Por tanto, aplicando el algoritmo de Euclides inductivamente obtenemos

$$1 = \sum_{i=1}^{s} a_i(x) \widehat{f}_i(x) \tag{3.1}$$

para ciertos $a_i(x) \in \mathbb{F}_q[x]$. Así, como

$$0 \neq m(x) = m(x) \cdot 1 = \sum_{i=1}^{s} m(x)a_i(x)\widehat{f_i}(x)$$

existe un i tal que $m(x)a_i(x)\widehat{f_i}(x) \neq 0$. Por tanto, $\mathcal{M} \cap \langle \widehat{f_i}(x) \rangle \neq \{0\}$, pues $m(x)a_i(x)\widehat{f_i}(x) \in \mathcal{M} \cap \langle \widehat{f_i}(x) \rangle$. Pero entonces $\mathcal{M} = \langle \widehat{f_i}(x) \rangle$ pues tanto $\mathcal{M}$ como $\langle \widehat{f_i}(x) \rangle$ son minimales. Por tanto todos los ideales minimales son de la forma $\langle \widehat{f_i}(x) \rangle$, como queríamos.

2. Por (3.1) concluimos que el 1 está en la suma de los ideales $\langle \widehat{f_i}(x) \rangle$, que es en sí mismo un ideal de $\mathcal{R}_n$. Por tanto, por la proposición 1.1.7, $\mathcal{R}_n$ es la suma de los ideales $\langle \widehat{f_i}(x) \rangle$. Para probar que es una suma directa tenemos que comprobar que los ideales son disjuntos, es decir, $\langle \widehat{f_i}(x) \rangle \cap \sum_{j \neq i} \langle \widehat{f_j}(x) \rangle = \{0\}$ para $1 \leq i \leq s$. Como $f_i(x) | \widehat{f_j}(x)$ para $j \neq i$, $f_j(x) | \widehat{f_j}(x)$ y los factores irreducibles de $x^n - 1$ son todos distintos, concluimos que

$$f_i(x) = \operatorname{mcd}\{\widehat{f_j}(x) : 1 \leq j \leq s, j \neq i\}.$$

Utilizando inducción sobre el teorema 3.4.5(2) concluimos que $\langle \widehat{f_i}(x) \rangle = \sum_{j \neq i} \langle \widehat{f_j}(x) \rangle$. Por tanto,

$$\begin{aligned} \langle \widehat{f_i}(x) \rangle \cap \sum_{j \neq i} \langle \widehat{f_j}(x) \rangle &= \langle \widehat{f_i}(x) \rangle \cap \langle f_i(x) \rangle \\ &= \langle \operatorname{mcm}(\widehat{f_i}(x), f_i(x)) \rangle \\ &= \langle x^n - 1 \rangle \\ &= \{0\}, \end{aligned}$$

por lo que los $\langle \widehat{f_i}(x) \rangle$ son disjuntos y la suma es directa, como queríamos ver.

3. Si $i \neq j$, $\widehat{e_i}(x)\widehat{e_j}(x) \in \langle \widehat{f_i}(x) \rangle \cap \langle \widehat{f_j}(x) \rangle = \{0\}$ por (2), luego $\widehat{e_i}(x)\widehat{e_j}(x) = 0$ como queríamos.
4. Usando (4) y aplicando inducción al teorema 3.4.5(2) obtenemos que $\sum_{i=1}^{s} \widehat{e_i}(x)$ es el idempotente generador de $\sum_{i=1}^{s} \langle \widehat{f_i}(x) \rangle = \mathcal{R}_n$, por (2). Luego el idempotente generador de $\mathcal{R}_n$ es 1.
5. Si $e(x)$ es un idempotente no nulo en $\langle \widehat{f_i}(x) \rangle$ entonces $\langle e(x) \rangle$ es un ideal contenido en $\langle \widehat{f_i}(x) \rangle$. Por minimalidad, dado que $e(x)$ es distinto de cero, $\langle \widehat{f_i}(x) \rangle = \langle e(x) \rangle$, lo que por el teorema 3.4.1 implica que $e(x) = \widehat{e_i}(x)$ ya que ambos son la unidad de $\langle \widehat{f_i}(x) \rangle$.
6. Notemos que $e(x)\widehat{e_i}(x)$ es un idempotente en $\langle \widehat{f_i}(x) \rangle$. Por tanto, por (6), $e(x)\widehat{e_i}(x)$ es, o bien 0 o bien $\widehat{e_i}(x)$. Sea $T = \{i : e(x)\widehat{e_i}(x) \neq 0\}$.

Entonces, por (5), $e(x) = e(x) \cdot 1 = e(x) \sum_{i=1}^{s} \hat{e}_i(x) = \sum_{i=1}^{s} e(x)\hat{e}_i(x) = \sum_{i \in T} \hat{e}_i(x)$. De hecho, $\langle e(x) \rangle = \langle \sum_{i \in T} \hat{e}_i(x) = \sum_{i \in T} \langle \hat{e}_i(x) \rangle$ aplicando por inducción el teorema 3.4.5(2). □

El siguiente teorema nos muestra que los ideales minimales descritos en el teorema 3.4.6 son extensiones de cuerpos de $\mathbb{F}_q$.

TEOREMA 3.4.7. *Sea $\mathcal{M}$ un ideal minimal de $\mathcal{R}_n$. Entonces $\mathcal{M}$ es una extensión de cuerpos de $\mathbb{F}_q$.*

Demostración. Basta con probar que cada elemento distinto de cero en $\mathcal{M}$ tiene inverso para el producto. Sea $a(x) \in \mathcal{M}$ distinto de cero. Entonces $\langle a(x) \rangle$ es un ideal de $\mathcal{R}_n$ distinto de cero contenido en $\mathcal{M}$, y por tanto, $\langle a(x) \rangle = \mathcal{M}$. Por tanto, si $e(x)$ es la unidad de $\mathcal{M}$ existe un elemento $b(x) \in \mathcal{R}_n$ tal que $a(x)b(x) = e(x)$. Sea ahora $c(x) = b(x)e(x) \in \mathcal{M}$, pues $e(x) \in \mathcal{M}$. Por tanto, $a(x)c(x) = e(x)^2 = e(x)$, con lo que $a(x)$ tiene inverso, como queríamos. □

A continuación vamos a describir una permutación que lleva idempotentes de $\mathcal{R}_n$ en idempotentes de $\mathcal{R}_n$. Sea a un entero tal que $\text{mcd}(a, n) = 1$. La función μ_a definida sobre $\{0, 1, \ldots, n-1\}$ por $i\mu_a \equiv ia \pmod{n}$ es una permutación de las posiciones de coordenadas $\{0, 1, \ldots, n-1\}$ de un código cíclico de longitud n y se denomina *multiplicador*. Dado que los códigos cíclicos de longitud n se representan como ideales de $\mathcal{R}_n$, para $a > 0$ es conveniente interpretar que μ_a actúa sobre $\mathcal{R}_n$ como

$$f(x)\mu_a \equiv f(x^a) \pmod{x^n - 1}. \tag{3.2}$$

Esta ecuación es consistente con la definición original de μ_a pues $x^i\mu_a = x^{ia} = x^{ia+jn}$ en $\mathcal{R}_n$ para un entero j tal que $0 \leq ia + jn$, pues $x^n = 1$ en $\mathcal{R}_n$. En otras palabras, $x^i\mu_a = x^{ia \bmod n}$. Si $a < 0$ podemos dar significado a $f(x^a)$ en $\mathcal{R}_n$ definiendo $x^i\mu_a = x^{ia \bmod n}$, donde $0 \leq ia \bmod n < n$. Con esta interpretación la ecuación (3.2) es consistente con la definición original de μ_a cuando $a < 0$.

3.5 CEROS Y CONJUNTOS CARACTERÍSTICOS

En esta sección vamos a ver que podemos caracterizar los códigos cíclicos $\mathcal{R}_n$ de otra forma: a partir de los ceros del polinomio $x^n - 1$, es decir, a partir de ciertas raíces enésimas de la unidad. Esta caracterización cobrará especial relevancia cuando estudiemos los conocidos como códigos BCH.

Como vimos en la sección 3.1, si $t = \text{ord}_n(q)$ entonces $\mathbb{F}_{q^t}$ es un cuerpo de descomposición de $x^n - 1$. Por tanto, $\mathbb{F}_{q^t}$ contiene una enésima raíz primitiva de la unidad α, y $x^n - 1 = \prod_{i=0}^{n-1}(x - \alpha^i)$ es la factorización de $x^n - 1$ sobre

$\mathbb{F}_{q^t}$. De hecho, $x^n - 1 = \prod_s M_{\alpha^s}(x)$ es la factorización de $x^n - 1$ en factores irreducibles sobre $\mathbb{F}_q$, donde s varía en un conjunto de representantes de las clases q-ciclotómicas módulo n.

Sea $\mathcal{C}$ un código cíclico en $\mathcal{R}_n$ con polinomio generador $g(x)$. Por los teoremas 3.1.3(1) y 3.2.1(7), podemos expresar el polinomio generador como $g(x) = \prod_s M_{\alpha^s}(x) = \prod_s \prod_{i \in C_s}(x - \alpha^i)$, donde s de nuevo varía en un conjunto C_s de representantes de las clases q-ciclotómicas módulo n. Sea $T = \bigcup_s C_s$ la unión de estas clases q-ciclotómicas. Las raíces de la unidad $\mathcal{Z} = \{\alpha^i : i \in T\}$ se denominan los *ceros* del código cíclico $\mathcal{C}$ y los elementos $\{\alpha^i : i \notin T\}$, los *elementos no nulos* de $\mathcal{C}$. El conjunto T se denomina *conjunto característico* de $\mathcal{C}$.

TEOREMA 3.5.1. *Sea α una raíz primitiva de la unidad en una extensión de cuerpos de $\mathbb{F}_q$ y sea $\mathcal{C}$ un código cíclico de longitud n sobre $\mathbb{F}_q$ con conjunto característico T y polinomio generador $g(x)$. Se verifica que:*

1. *Una palabra código $c(x) \in \mathcal{R}_n$ está en $\mathcal{C}$ si y solo si $c(\alpha^i) = 0$ para todo $i \in T$.*
2. *La dimensión de $\mathcal{C}$ es $n - |T|$.*

Demostración. Veamos la demostración por apartados.

1. Se deduce directamente del teorema 3.2.1, pues $c(x)$ será un múltiplo del polinomio generador $g(x)$ de $\mathcal{C}$, que por 3.2.1(7) verifica que $g(\alpha^i) = 0$ para todo $i \in T$.
2. Se de deduce del teorema 3.2.1, pues $|T|$ es el grado de $g(x)$. □

Es importante observar que T, y por ello tanto el conjunto de ceros como el de elementos distintos de cero, determinan por completo el polinomio generador $g(x)$.

EJEMPLO 3.5.2. Continuando con el ejemplo 3.2.4 en el que describimos todos los códigos cíclicos binarios de longitud 7 vamos a indicar a continuación cuales son los conjuntos característicos, tomando como $\alpha = \zeta_7^3$. Para ello vamos a utilizar el código descrito en el anexo B tal y como mostramos en el listado siguiente.

```
1 sage: F = GF(2)
2 sage: x = polygen(F)
3 sage: defining_sets(x^7 - 1)
4 > [(1, [], z3),
5    (x + 1, [0], z3),
6    (x^3 + x + 1, [1, 2, 4], z3),
7    (x^3 + x^2 + 1, [3, 5, 6], z3),
```

```
8     (x^4 + x^3 + x^2 + 1, [0, 1, 2, 4], z3),
9     (x^4 + x^2 + x + 1, [0, 3, 5, 6], z3),
10    (x^6 + x^5 + x^4 + x^3 + x^2 + x + 1, [1, 2, 3, 4, 5,
          6], z3),
11    (x^7 + 1, [0, 1, 2, 3, 4, 5, 6], z3)]
```

En la tabla siguiente mostramos de forma más clara cuáles son los generadores e idempotentes correspondientes para cada código.

i	dimensión	generador $g_i(x)$	conjunto T
0	0	$x^7 + 1$	$\{0, 1, 2, 3, 4, 5, 6\}$
1	1	$x^6 + x^5 + \cdots + x + 1$	$\{1, 2, 3, 4, 5, 6\}$
2	3	$x^4 + x^3 + x^2 + 1$	$\{0, 3, 5, 6\}$
3	3	$x^4 + x^2 + x + 1$	$\{0, 1, 2, 4\}$
4	4	$x^3 + x + 1$	$\{3, 5, 6\}$
5	4	$x^3 + x^2 + 1$	$\{1, 2, 4\}$
6	6	$x + 1$	$\{0\}$
7	7	1	$\emptyset$

Tabla 3.2: Polinomios generadores y conjuntos característicos para los códigos cíclicos de longitud 7

4

CÓDIGOS BCH

En este capítulo vamos a estudiar los códigos BCH, un tipo de códigos cíclicos que permiten ser diseñados con una capacidad de corrección concreta. Como ya sabemos, para cualquier tipo de código es importante determinar la distancia mínima si queremos determinar su capacidad de corrección de errores. A este respecto es útil disponer de cotas en la distancia mínima, especialmente cotas inferiores, pues son las que maximizan la capacidad de corrección. Existen varias cotas conocidas para la distancia mínima de un código cíclico, pero nos vamos a centrar en la llamada *cota de Bose-Ray-Chaudhuri-Hocquenghem*, usualmente abreviada como *cota BCH*. Esta cota es esencial para comprender la definición de los códigos BCH que estudiamos en este capítulo. La cota BCH va a depender de los ceros del código, concretamente en la posibilidad de encontrar cadenas de ceros «consecutivos». La fuente principal de este capítulo ha sido (Huffman & Pless, 2003).

4.1 CONSTRUCCIÓN DE CÓDIGOS BCH

En lo que sigue vamos a considerar un código cíclico $\mathcal{C}$ de longitud n sobre $\mathbb{F}_q$ y α una enésima raíz primitiva de la unidad en $\mathbb{F}_{q^t}$, donde $t = \operatorname{ord}_n(q)$. Recordemos que T es un conjunto característico de $\mathcal{C}$ siempre y cuando los ceros de $\mathcal{C}$ sean $\{\alpha^i : i \in T\}$. Por tanto T ha de ser una unión de clases q-ciclotómicas módulo n. Decimos que T contiene un conjunto de s *elementos consecutivos* si existe un conjunto $\{b, b+1, \dots, b+s-1\}$ de s enteros consecutivos tal que

$$\{b, b+1, \dots, b+s-1\} \bmod n = S \subseteq T.$$

Antes de proceder con la cota BCH vamos a enunciar un lema —que será utilizado en la demostración de dicha cota— sobre el determinante de una matriz de Vandermonde. Sean $\alpha_1, \dots, \alpha_s$ elementos de un cuerpo $\mathbb{F}$. La matriz de tamaño $s \times s$ dada por $V = (v_{i,j})$, donde $v_{i,j} = \alpha_j^{i-1}$ se denomina *matriz de Vandermonde*. Observamos que la transpuesta de una matriz de Vandermonde es otra matriz de Vandermonde.

LEMA 4.1.1. *El determinante de una matriz de Vandermonde V viene dado por* $\det V = \prod_{1 \le i < j \le s} (\alpha_j - \alpha_i)$. *En particular, V es no singular si los elementos $\alpha_1, \dots, \alpha_s$ son todos diferentes dos a dos.*

Estamos ya en condiciones de presentar y demostrar el teorema de la cota BCH.

TEOREMA 4.1.2. (COTA BCH.) *Sea $\mathcal{C}$ un código cíclico de longitud n sobre $\mathbb{F}_q$ con conjunto característico T. Supongamos que $\mathcal{C}$ tiene peso mínimo d. Asumamos que T contiene $\delta - 1$ elementos consecutivos para algún entero δ. Entonces, $d \geq \delta$.*

Demostración. Asumimos que el código $\mathcal{C}$ tiene ceros que incluyen

$$\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+\delta-2}.$$

Sea $c(x)$ una palabra código de $\mathcal{C}$ de peso w de la forma

$$c(c) = \sum_{j=1}^{w} c_{i_j} x^{i_j}.$$

Vamos a proceder por reducción al absurdo. Supongamos que $w \leq \delta$. Como $c(\alpha^i) = 0$ para $b \leq i \leq b + \delta - 2$, $M\mathbf{u}^T = \mathbf{0}$, donde

$$M = \begin{pmatrix} \alpha^{i_1 b} & \alpha^{i_2 b} & \ldots & \alpha^{i_w b} \\ \alpha^{i_1 (b+1)} & \alpha^{i_2 (b+1)} & \ldots & \alpha^{i_w (b+1)} \\ & & \vdots & \\ \alpha^{i_1 (b+w-1)} & \alpha^{i_2 (b+w-1)} & \ldots & \alpha^{i_w (b+w-1)} \end{pmatrix}$$

y $\mathbf{u} = c_{i_1} c_{i_2} \ldots_{i_w}$. Como $\mathbf{u} \neq \mathbf{0}$ la matriz M es singular, y por tanto $\det M = 0$. Pero $\det M = \alpha^{(i_1 + i_2 + \cdots + i_w) b} \det V$, donde V es la matriz de Vandermonde

$$V = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \ldots & \alpha^{i_w} \\ & & \vdots & \\ \alpha^{i_1 (w-1)} & \alpha^{i_2 (w-1)} & \ldots & \alpha^{i_w (w-1)} \end{pmatrix}.$$

Como los α^{i_j} son todos distintos dos a dos, $\det V \neq 0$ por el lema 4.1.1, lo que contradice que $\det M = 0$. □

Los códigos BCH son códigos cíclicos diseñados para aprovechar la cota BCH. Idealmente, a la hora de diseñar un código cíclico $\mathcal{C}$ de longitud n sobre $\mathbb{F}_q$ nos gustaría poder construirlo teniendo a la vez un peso mínimo grande y una dimensión grande. Tener un peso mínimo grande, basándonos en la cota BCH, se puede conseguir escogiendo un conjunto característico para $\mathcal{C}$ que tenga un gran número de elementos consecutivos.

Como la dimensión de $\mathcal{C}$ es $n - |T|$ por el teorema 3.5.1, nos gustaría que $|T|$ fuese tan pequeño como sea posible. Por tanto, si quisiésemos que

$\mathcal{C}$ tenga distancia mínima de al menos δ, podemos escoger un conjunto característico tan pequeño como sea posible que sea una unión de clases q-ciclotómicas con $\delta - 1$ elementos consecutivos.

Sea δ un entero tal que $2 \leq \delta \leq n$. Un *código BCH* $\mathcal{C}$ sobre $\mathbb{F}_q$ de longitud n y *distancia mínima prevista* δ es un código cíclico con conjunto característico

$$T = C_b \cup C_{b+1} \cup \cdots \cup C_{b+\delta-2}, \tag{4.1}$$

donde C_i es la clase q-ciclotómica módulo n que contiene a i. Por la cota BCH este código tiene distancia mínima prevista al menos δ.

TEOREMA 4.1.3. *Un código BCH de distancia mínima prevista δ tiene peso mínimo de al menos δ.*

Demostración. El conjunto característico 4.1 tiene al menos $\delta - 1$ elementos. El resultado se deduce de la cota BCH. □

Al variar el valor de b obtenemos distintos códigos con distancias mínimas y dimensiones diferentes. Cuando $b = 1$ el código $\mathcal{C}$ se dice que es un código BCH *en sentido estricto*. Como con cualquier código cíclico, si $n = q^t - 1$ entonces $\mathcal{C}$ es un código BCH *primitivo*. En la sección siguiente vamos a estudiar un algoritmo de decodificación que permite aprovechar las ventajas de los códigos BCH.

4.2 CÓDIGOS REED-SOLOMON

Vamos a describir brevemente los códigos Reed-Solomon, que abreviaremos como códigos RS, pues aludiremos a ellos cuando hablemos de códigos cíclicos sesgados. Son una subfamilia de los códigos BCH que acabamos de definir.

DEFINICIÓN 4.2.1. Un código RS sobre $\mathbb{F}_q$ es un código BCH de longitud $n = q - 1$.

El siguiente teorema nos presenta un par de propiedades importantes de los códigos RS.

TEOREMA 4.2.2. *Sea $\mathcal{C}$ un código RS sobre $\mathbb{F}_q$ de longitud $n = q-1$ y distancia mínima prevista δ. Entonces:*

1. *El código $\mathcal{C}$ tiene conjunto característico $T = \{b, b+1, \ldots, b+\delta-2\}$ para algún entero b.*
2. *El código $\mathcal{C}$ tiene distancia mínima $d = \delta$ y dimensión $k = n - d + 1$.*

Demostración. Veamos la demostración por apartados.

1. Como es un código BCH de distancia mínima prevista δ su conjutno característico T tiene que tener tamaño $\delta - 1$ y en consecuencia es $T = \{b, b+1, \dots, b+\delta-2\}$.
2. La distancia es obvia, pues es un código BCH y la dimensión proviene del teorema 3.5.1, pues $n - |T| = n - \delta + 1$.

□

4.3 ALGORITMO DE PETERSON-GORENSTEIN-ZIERLER

El algoritmo de Peterson-Gorenstein-Zierler —de ahora en adelante, algoritmo PGZ— es un algoritmo de decodificación de códigos BCH que permite corregir hasta $t = \lfloor (\delta - 1)/2 \rfloor$ errores. Fue desarrollado originalmente en 1960 por Peterson (Peterson, 1960) para decodificar códigos BCH binarios, y generalizado poco después por Gorenstein y Zierler para códigos no binarios (Gorenstein & Zierler, 1961).

Como en cualquier otro método de decodificación el objetivo es obtener el mensaje original $c(x)$ a partir de un mensaje recibido $y(x)$, para lo que hay que hallar primero los errores $e(x)$ que se han producido en la transmisión, de forma que $c(x) = y(x) - e(x)$. El vector de errores ha de tener peso $v \leq t$, ya que no podemos corregir más errores de los que el código permite. Vamos a considerar que los errores se han producido en coordenadas desconocidas $k_1, k_2, \dots, k_v$, de forma que el vector de errores lo podemos expresar como

$$e(x) = e_{k_1} x^{k_1} + e_{k_2} x^{k_2} + \cdots + e_{k_v} x^{k_v}.$$

Como nuestro objetivo es determinar $e(x)$ tenemos que hallar:

— las *coordenadas de error* k_j;
— las *magnitudes de error* e_{k_j}.

Vamos a estudiar a continuación el desarrollo teórico y la justificación del funcionamiento del algoritmo para después dar una versión del mismo esquematizada en pseudocódigo. El comienzo de este método es similar al del descrito en la sección 2.1.1, solo que en lugar de utilizar la propiedad de la matriz de paridad allí descrita utilizaremos la propiedad análoga de que, por el teorema 3.5.1, un elemento $c(x) \in \mathcal{C}$ si y solo si $c(\alpha^i) = 0$ para todo $i \in T$. En nuestro caso particular, dado que $t = \lfloor (\delta - 1)/2 \rfloor$ y T contiene a $\{1, 2, \dots, \delta - 1\}$, se tiene que

$$y(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i)$$

para todo $1 \leq i \leq 2t$. Estas ecuaciones van a ser fundamentales para encontrar el error $e(x)$. En este caso llamaremos *síndrome* s_i de $y(x)$ al elemento

de $\mathbb{F}_q^m$ dado por $s_i = y(\alpha^i)$. El primer paso del algoritmo es encontrar los síndromes para todo $1 \leq i \leq 2t$. Estos síndromes nos conducen a un sistema de ecuaciones en el que se encuentran las coordenadas de error k_j y las magnitudes de error e_{k_j}. Desarrollando lo anterior podemos expresar los síndromes como

$$s_i = y(\alpha^i) = \sum_{j=1}^{v} e_{k_j}(\alpha^i)^{k_j} = \sum_{j=1}^{v} e_{k_j}(\alpha^{k_j})^i \tag{4.2}$$

para todo $1 \leq i \leq 2t$. A fin de simplificar la notación, para $1 \leq j \leq v$ definimos:

— $E_j = e_{k_j}$, que llamaremos *magnitud de error en la coordenada* k_j, y
— $X_j = \alpha^{k_j}$, que llamaremos *número de coordenada de error correspondiente a la coordenada* k_j.

Observamos que al conocer X_j conocemos de forma unívoca la coordenada de error k_j, ya que si $\alpha^i = \alpha^k$ para i y k entre 0 y $n-1$, entonces $i = k$. Con la notación que hemos descrito la igualdad (4.2) la podemos escribir como

$$S_i = \sum_{j=1}^{v} E_j X_j^i, \quad \text{para } 1 \leq i \leq 2t, \tag{4.3}$$

lo que nos conduce al sistema de ecuaciones:

$$\begin{cases} S_1 = E_1X_1 + E_2X_2 + \cdots + E_vX_v, \\ S_2 = E_1X_1^2 + E_2X_2^2 + \cdots + E_vX_v^2, \\ S_3 = E_1X_1^3 + E_2X_2^3 + \cdots + E_vX_v^3, \\ \quad \vdots \\ S_{2t} = E_1X_1^{2t} + E_2X_2^{2t} + \cdots + E_vX_v^{2t}. \end{cases} \tag{4.4}$$

De este sistema desconocemos tanto los valores de los X_j como los de los E_j, pero es que además no es lineal para los X_j. Como no podemos resolverlo directamente vamos a tratar de encontrar otra forma con la que calcular los valores X_j y utilizarlos para resolver el sistema lineal que forman los E_j. Para ello vamos a buscar un sistema lineal que dependa de otras variables $\sigma_1, \ldots, \sigma_v$ que nos conduzca a los valores X_j. Definimos el *polinomio localizador de errores* $\sigma(x)$ como

$$\sigma(x) = (1 - xX_1)(1 - xX_2)\ldots(1 - xX_v) = 1 + \sum_{i=1}^{v} \sigma_i x^i.$$

Como vemos inmediatamente por su definición, las raíces de $\sigma(x)$ son los inversos de los números de coordenadas de error. Por tanto,

$$\sigma(X_j^{-1}) = 1 + \sigma_1 X_j^{-1} + \sigma_2 X_j^{-2} + \cdots + \sigma_v X_j^{-v} = 0$$

para $1 \leq j \leq v$. Si multiplicamos a ambos lados de la expresión por $E_j X_j^{i+v}$ obtenemos

$$E_j X_j^{i+v} + \sigma_1 E_j X_j^{i+v-1} + \cdots + \sigma_v E_j X_j^{i} = 0$$

para todo i. Si sumamos para todo j en $1 \leq j \leq v$ tenemos

$$\sum_{j=1}^{v} E_j X_j^{i+v} + \sigma_1 \sum_{j=1}^{v} E_j X_j^{i+v-1} + \cdots + \sigma_v \sum_{j=1}^{v} E_j X_j^{i} = 0.$$

Lo que hemos obtenido en estas sumas son los síndromes descritos en (4.3), ya que $1 \leq i$ y $i + v \leq 2t$. Como $v \leq t$ la expresión anterior se convierte en

$$S_{i+v} + \sigma_1 S_{i+v-1} + \sigma_2 S_{i+v-2} + \cdots + \sigma_v S_i = 0,$$

que equivale a

$$\sigma_1 S_{i+v-1} + \sigma_2 S_{i+v-2} + \cdots + \sigma_v S_i = -S_{i+v},$$

para todo $1 \leq i \leq v$. Por tanto podemos encontrar los σ_k si resolvemos el sistema de ecuaciones dado por:

$$\begin{pmatrix} S_1 & S_2 & S_3 & \dots & S_{v-1} & S_v \\ S_1 & S_2 & S_3 & \dots & S_{v-1} & S_v \\ S_1 & S_2 & S_3 & \dots & S_{v-1} & S_v \\ & & & \vdots & & \\ S_1 & S_2 & S_3 & \dots & S_{v-1} & S_v \end{pmatrix} \begin{pmatrix} \sigma_v \\ \sigma_{v-1} \\ \sigma_{v-2} \\ \vdots \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} -S_{v+1} \\ -S_{v+2} \\ -S_{v+3} \\ \vdots \\ -S_{2v} \end{pmatrix}.$$

La dificultad de este paso es que desconocemos el valor de v (el número de errores), por lo que vamos a realizar un procedimiento iterativo. Suponemos que nuestro número de errores es $\mu = t$, que es el máximo que podemos corregir. Tenemos que quedarnos con el menor valor de v que sea posible. Para ello tenemos en cuenta que la matriz

$$M_\mu = \begin{pmatrix} S_1 & S_2 & \dots & S_\mu \\ S_2 & S_3 & \dots & S_{\mu+1} \\ & & \vdots & \\ S_\mu & S_{\mu+1} & \dots & S_{2\mu-1} \end{pmatrix}$$

será no singular si $\mu = v$ y singular si $\mu > v$ (Huffman & Pless, 2003, Lema 5.4.2). Así, si M_μ es singular reducimos el valor de μ en 1, $\mu = \mu - 1$ y probamos de nuevo si M_μ es singular. Repetimos hasta encontrar una matriz que no sea singular. Ese valor μ será el número de errores v. Conocido el tamaño podemos resolver el sistema y obtener los valores σ_k. Ahora solo tenemos que deshacer el camino que hemos recorrido hasta ahora. Conocidos los σ_k podemos determinar $\sigma(x)$ y con él, sus raíces, utilizando el procedimiento que queramos, usualmente, calculando reiteradamente $\sigma(\alpha^i)$ para $0 \leq i < n$ hasta encontrarlas. Como ya dijimos, si las invertimos hallaremos los valores de X_j, y con ellos ya podemos resolver el sistema (4.4), obteniendo así los valores de los E_j. Conocidos todos los valores de X_j y E_j podemos obtener los de k_j y e_{k_j}, con los que podemos determinar el vector de error $e(x)$. Ya solo queda restar $y(x) - e(x)$ para obtener el mensaje original.

En resumen, el algoritmo consiste en:

1. Determinar los síndromes del mensaje recibido.
2. Encontrar el polinomio localizador.
3. Hallar las raíces del polinomio localizador e invertirlas para obtener las coordenadas de error k_j.
4. Utilizar estos inversos para resolver el sistema de ecuaciones formado por los síndromes, obteniendo así las magnitudes de error e_{k_j}.
5. Hallar el vector de error $e(x)$ y restárselo al mensaje $y(x)$.

Hemos expresado en el algoritmo 1 el algoritmo PGZ en pseudocódigo siguiendo este esquema. A partir de él se ha realizado una implementación en el sistema SageMath que puede consultarse en los archivos enlazadas en el anexo A. Veamos un par de ejemplos que utilizan esta implementación.

EJEMPLO 4.3.1. Sea $\mathcal{C}$ un $[15, 7]$ código BCH en sentido estricto de distancia designada $\delta = 5$. Supongamos que recibimos el mensaje $y(x) = x^{10} + x^9 + x^6 + x^5 + x + 1$. Vamos a corregir los errores que puedan haberse producido en la transmisión.

```
1  sage: F = GF(2)
2  sage: C = codes.BCHCode(F, 15, 5, offset=1)
3  sage: x = polygen(F)
4  sage: y = 1 + x +x^5 + x^6 + x^9 + x^10
5  sage: D = BCHPGZDecoder(C)
6  sage: DEBUG = true
7  sage: D.decode_to_code(y)
8  > polinomio generador: x^8 + x^7 + x^6 + x^4 + 1
9    raíz primitiva: z4
10   síndromes: [z4^2, z4 + 1, z4^3 + z4^2 + z4, z4^2 + 1]
```

```
11   tamaño de m_mu: 2
12   matriz m_mu:
13   [            z4^2           z4 + 1]
14   [          z4 + 1 z4^3 + z4^2 + z4]
15   vector b_mu: (z4^3 + z4^2 + z4, z4^2 + 1)
16   matriz de soluciones de m_mu*S = b_mu: (z4^3 + 1, z4^2)
17   polinomio localizador sigma(x): (z4^3 + 1)*x^2 + z4^2*x
         + 1
18   raíces de sigma(x): [(z4^2 + z4, 1), (z4^3 + z4^2 + z4,
         1)]
19   X_j: [z4^2 + z4 + 1, z4 + 1]
20   k_j: [10, 4]
21   magnitudes de error E: (1, 1)
22   error e: x^10 + x^4
23 > (1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0)
```

El mensaje corregido es por tanto $m(x) = x^9 + x^6 + x^5 + x^4 + x + 1$.

EJEMPLO 4.3.2. Sea $\mathcal{C}$ un $[15, 5]$ código BCH en sentido estricto de distancia designada $\delta = 7$. Supongamos que recibimos el mensaje $y(x) = x^{12} + x^9 + x^7 + x^5 + x^4 + x + 1$. Vamos a corregir los errores que puedan haberse producido en la transmisión.

```
1  sage: F = GF(2)
2  sage: C = codes.BCHCode(F, 15, 7, offset=1)
3  sage: x = polygen(F)
4  sage: y = 1 + x +x^4 + x^5 + x^7 + x^9 + x^12
5  sage: D = BCHPGZDecoder(C)
6  sage: DEBUG = true
7  sage: D.decode_to_code(y)
8  > polinomio generador: x^10 + x^8 + x^5 + x^4 + x^2 + x + 1
9    raíz primitiva: z4
10   síndromes: [z4^3, z4^3 + z4^2, z4^3, z4^3 + z4^2 + z4 + 1, 0, z4^3
11     + z4^2]
12   tamaño de m_mu: 3
13   matriz m_mu:
14   [               z4^3          z4^3 + z4^2                 z4^3]
15   [         z4^3 + z4^2                z4^3 z4^3 + z4^2 + z4 + 1]
16   [               z4^3 z4^3 + z4^2 + z4 + 1                   0]
17   vector b_mu: (z4^3 + z4^2 + z4 + 1, 0, z4^3 + z4^2)
18   matriz de soluciones de m_mu*S = b_mu: (z4^3 + z4^2, z4^2 + 1, z4^3)
19   polinomio localizador sigma(x): (z4^3 + z4^2)*x^3 + (z4^2 + 1)*x^2
20     + z4^3*x + 1
21   raíces de sigma(x): [(1, 1), (z4 + 1, 1), (z4^2 + z4, 1)]
```

```
22    X_j: [1, z4^3 + z4^2 + z4, z4^2 + z4 + 1]
23    k_j: [0, 11, 10]
24    magnitudes de error E: (1, 1, 1)
25    error e: x^11 + x^10 + 1
26  > (0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0)
```

El mensaje corregido es por tanto $m(x) = x^{12}+x^{11}+x^{10}+x^9+x^7+x^5+x^4+x$.

1 Aquí estamos describiendo una matriz por sus entradas, las filas varían en i, y las columnas, en j.

Entrada: el código $\mathcal{C}$, el mensaje recibido $y(x)$
Salida: el mensaje decodificado $c(x)$

$\delta \longleftarrow$ distancia designada de $\mathcal{C}$
$t \longleftarrow \lfloor (\delta - 1)/2 \rfloor$
$g \longleftarrow$ polinomio generador de $\mathcal{C}$
$\alpha \longleftarrow$ raíz primitiva del cuerpo de descomposición usado para generar el conjunto característico de $\mathcal{C}$
// Paso 1: calcular síndromes
para $1 \leq i \leq 2t$ **hacer**
  $S_i \longleftarrow y(\alpha^i)$
fin
// Paso 2: hallar polinomio localizador
$\mu \longleftarrow t$
$M_\mu \longleftarrow (S_{i+j-1})_{i,j}$ [1] $, 1 \leq i, j \leq \mu$
mientras M_μ *es no singular* **hacer**
  $\mu \longleftarrow \mu - 1$
  $M_\mu \longleftarrow (S_{i+j-1})_{i,j}, 1 \leq i, j \leq \mu$
fin
$v \longleftarrow \mu$
$\sigma \longleftarrow (\sigma_{v-i+1})_i, 1 \leq i \leq v$
$b_\mu \longleftarrow (-S_{v+i})_i, 1 \leq i \leq v$
$\sigma_k \longleftarrow$ soluciones del sistema $M_\mu \sigma = b_\mu$
$\sigma(x) \longleftarrow 1 + \sum_{i=1}^{v} \sigma_i x^i$
// Paso 3: obtener las coordenadas de error
$r_k \longleftarrow$ raíces de $\sigma(x)$
$X_j \longleftarrow r_j^{-1}$
$k_j \longleftarrow \log_\alpha(X_j)$
// Paso 4: obtener las magnitudes de error
$M_S \longleftarrow (X_j^i)_{i,j}, 1 \leq i, j \leq v$
$E \longleftarrow (E_i)_i, 1 \leq i \leq v$
$b_S \longleftarrow (S_i)_i, 1 \leq i \leq v$
$E_k \longleftarrow$ soluciones del sistema $M_S E = b_S$
// Paso 5: calcular el mensaje original
$e(x) \longleftarrow \sum_{j=1}^{v} E_i x^{k_i}$
$c(x) = y(x) - e(x)$

Algoritmo 1: Peterson-Gorenstein-Zierler para códigos cíclicos.

5

ANILLOS DE POLINOMIOS DE ORE

En este capítulo vamos a hablar sobre los anillos de polinomios de Ore, que serán la base de los códigos cíclicos sesgados. Fueron descritos por Oystein Ore en 1933 (Ore, 1933). En esencia, se trata de anillos de polinomios no conmutativos en los que el producto no está definido de la forma usual sino que sigue una regla concreta que vamos a describir a continuación. Primero daremos la definición general, sin detenernos a justificar su construcción, pues acto seguido vamos a centrarnos en el caso que nos va a ocupar cuando trabajemos con códigos cíclicos sesgados. Las definiciones y el desarrollo teórico seguido en este capítulo proceden de (Jacobson, 1996), (Ore, 1933) y (Gómez-Torrecillas, Lobillo & Navarro, 2020).

DEFINICIÓN 5.0.1. Sea R un anillo, σ un endomorfismo de R y δ una σ-*derivación* de R, es decir, δ es un homomorfismo de grupos abelianos tal que para $a, b \in R$ se verifica que

$$\delta(ab) = \sigma(a)\delta(b) + \delta(a)b.$$

Entonces, el anillo $R[t; \sigma, \delta]$ de los polinomios en $R[t]$ de la forma

$$a_0 + a_1 t + \cdots + a_n t^n,$$

donde $a_i \in R$, con la igualdad y suma usuales, y en el que la multiplicación verifica la relación

$$ta = \sigma(a)t + \delta a, \qquad a \in R,$$

se conoce como *anillo de polinomios de Ore* o *anillos de polinomios sesgados*.

Para comprobar que $R[t; \sigma, \delta]$ es un anillo tendríamos que ver que efectivamente con las operaciones que hemos dado se verifican todas las propiedades de los anillos. Puesto que hemos usado la suma usual de los polinomios, bastaría probar que se verifica la propiedad asociativa para la multiplicación que hemos definido. No vamos a entrar en detalle pues, como ya hemos comentado, no es el objetivo de este trabajo el estudio de los anillos de polinomios de Ore en general. En cualquier caso, si se desea consultar una demostración, puede encontrarse en (Jacobson, 1996, p. 2-3) una que utiliza cierta representación matricial de los elementos.

5.1 ANILLOS DE POLINOMIOS DE ORE SOBRE CUERPOS FINITOS

Trabajar con códigos cíclicos sesgados requiere del estudio del anillo $\mathbb{F}_q[x;\sigma]$, con σ un automorfismo. Por tanto, nos vamos a centrar los anillos de polinomios de Ore en los que $R = \mathbb{F}_q$ —cuerpo finito de q elementos—, hemos llamado x a t, σ es un automorfismo y $\delta = 0$. En estos anillos la multiplicación verifica la relación

$$xa = \sigma(a)x, \qquad a \in R.$$

Es este caso particular que nos va a ocupar de ahora en adelante.

Vamos a ver por inducción que, como podemos intuir, $x^n a = \sigma^n(a)x^n$. Visto el caso base xa y supuesto que se verifica la igualdad para $n-1$, para n tenemos que

$$x^n a = xx^{n-1}a = x\sigma^{n-1}(a)x^{n-1} = \sigma(\sigma^{n-1}(a))x^{n-1}x = \sigma^n(a)x^n,$$

como habíamos afirmado. Ahora definimos

$$(ax^n)(bx^m) = a\sigma^n(b)x^{n+m},$$

con lo que, junto a la propiedad distributiva podemos definir el producto de polinomios en x como cabría esperar,

$$(\sum a_n x^n)(\sum b_m x^m) = \sum (a_n x^n)(b_m x^m).$$

Para comprobar que $\mathbb{F}_q[x;\sigma]$ es un anillo es necesario comprobar que se verifica la propiedad asociativa para la multiplicación, lo que efectivamente se cumple, como ya hemos comentado en el caso general.

A continuación vemos que, partiendo de que $\mathbb{F}_q$ es en particular un anillo de división, el anillo de polinomios de Ore $\mathbb{F}_q[x;\sigma]$ es un dominio de integridad no conmutativo. Dado un polinomio $f(x) = a_0 + a_1 x + \cdots + a_n x^n$ con $a_n \neq 0$ definimos el grado de dicho polinomio de la forma habitual, es decir, $\mathrm{gr}(f(x)) = n$ y establecemos el convenio de que $\mathrm{gr}(0) = -\infty$. Si consideramos otro polinomio $g(x) = b_0 + b_1 x + \cdots + b_m x^m$ con $b_m \neq 0$ entonces $f(x)g(x) = \cdots + a_n\sigma^n(b_m)x^{n+m}$ y $a_n\sigma^n(b_m) \neq 0$ y así,

$$\mathrm{gr}(f(x)g(x)) = \mathrm{gr}(f(x)) + \mathrm{gr}(g(x)).$$

Observamos entonces que $\mathrm{gr}(f(x)) + \mathrm{gr}(g(x)) = -\infty$ si y solo si $f(x)$ o $g(x)$ son el polinomio cero. Por tanto, $\mathbb{F}_q[x;\sigma]$ no tiene divisores de cero distintos del cero, por lo que es un dominio de integridad no conmutativo, como habíamos afirmado.

5.2 DIVISIÓN

Podemos definir algoritmos de división en $\mathbb{F}_q[x;\sigma]$ tanto a la izquierda como a la derecha —descritos en los algoritmos 2 y 3—, de forma que para cada $f(x), g(x) \in \mathbb{F}_q[x;\sigma]$ —con $g(x) \neq 0$— existen elementos $q(x), r(x)$ únicos, con $\operatorname{gr}(r) < \operatorname{gr}(g)$ tales que al dividir por la izquierda obtenemos

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

y al dividir por la derecha,

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

Entrada: polinomios $f, g \in \mathbb{F}_q[x;\sigma]$ con $g \neq 0$
Salida: polinomios $q, r \in \mathbb{F}_q[x;\sigma]$ tales que $f = qg + r$, y $\operatorname{gr}(r) < \operatorname{gr}(g)$

1 $q \longleftarrow 0$
2 $r \longleftarrow f$
3 **mientras** $\operatorname{gr}(g) \leq \operatorname{gr}(r)$ **hacer**
4 $\quad a \longleftarrow \operatorname{cl}(r)\sigma^{\operatorname{gr}(r)-\operatorname{gr}(g)}(\operatorname{cl}(g)^{-1})$
5 $\quad q \longleftarrow q + ax^{\operatorname{gr}(r)-\operatorname{gr}(g)}$
6 $\quad r \longleftarrow r - ax^{\operatorname{gr}(r)-\operatorname{gr}(g)}g$
7 **fin**

Algoritmo 2: División por la izquierda en $\mathbb{F}_q[x;\sigma]$

Entrada: polinomios $f, g \in \mathbb{F}_q[x;\sigma]$ con $g \neq 0$
Salida: polinomios $q, r \in \mathbb{F}_q[x;\sigma]$ tales que $f = gq + r$, y $\operatorname{gr}(r) < \operatorname{gr}(g)$

1 $q \longleftarrow 0$
2 $r \longleftarrow f$
3 **mientras** $\operatorname{gr}(g) \leq \operatorname{gr}(r)$ **hacer**
4 $\quad a \longleftarrow \sigma^{-\operatorname{gr}(g)}(\operatorname{cl}(g)^{-1}\operatorname{cl}(r))$
5 $\quad q \longleftarrow q + ax^{\operatorname{gr}(r)-\operatorname{gr}(g)}$
6 $\quad r \longleftarrow r - gax^{\operatorname{gr}(r)-\operatorname{gr}(g)}$
7 **fin**

Algoritmo 3: División por la derecha en $\mathbb{F}_q[x;\sigma]$

Cuando dividimos por la izquierda (respectivamente por la derecha) el polinomio $g(x)$ se le llama *cociente por la izquierda* (*derecha*) y a $r(x)$, *resto*

por la izquierda (*derecha*). Los denotaremos por $g(x) = \mathrm{coi}(f(x), g(x))$ o $\mathrm{cod}(f(x), g(x))$ y $r(x) = \mathrm{rei}(f(x), g(x))$ o $\mathrm{red}(f(x), g(x))$.

La existencia de algoritmos de algoritmos de división a izquierda y a derecha implica que $\mathbb{F}_q[x;\sigma]$ es un dominio de ideales principales a izquierda y a derecha, es decir, que es lo que llamamos simplemente dominio de ideales principales.

De ahora en adelante, para ser más concisos con la notación, vamos a llamar $R = \mathbb{F}_q[x;\sigma]$ y cuando no sea necesario hacer referencia a la variable x, a un polinomio $f(x)$ lo denotaremos simplemente por f. Los ideales biláteros de R serán de la forma $I = Rf = f^*R$ y para todo $g \in R$ existirán $g', \tilde{g} \in R$ tales que $fg = g'f$ y $gf^* = f^*\tilde{g}$. Los elementos f tales que para todo $g \in R$ existen g' y $\tilde{g}$ tales que $fg = g'f$ y $gf = f\tilde{g}$ se llaman elementos *biláteros* y además $Rf = fR$ es un ideal.

TEOREMA 5.2.1. *Sea $R = F_q[x,\sigma]$. Se verifican las siguientes afirmaciones.*

1. *Los elementos biláteros de R son de la forma $ac(t)x^n$, donde $a \in \mathbb{F}_q$, $n = 0, 1, \ldots$ y $c(t) \in \mathrm{Cent}(R)$, el centro de R.*
2. *Supongamos ahora que σ tiene orden n, de forma que $\sigma^n = \mathrm{Id}$. El centro de R es el conjunto de los polinomios de la forma*

$$\gamma_0 + \gamma_1 x^n + \gamma_2 x^{2n} + \cdots + \gamma_s x^{sn},$$

donde $\gamma_i \in \mathbb{F}_q$.

Dados $g, f \in R$ supongamos que $Rg \subseteq Rf$ con $Rg \neq 0$. Entonces $g = hf$, por lo que decimos que f es un *divisor por la derecha* de g y lo notaremos por $f \mid_d g$. Equivalentemente, podemos decir que g es un *múltiplo por la izquierda* de f. Observemos que de igual forma, si $f \mid_d g$ entonces $Rg \subseteq Rf$.

5.3 MÍNIMO COMÚN MÚLTIPLO Y MÁXIMO COMÚN DIVISOR

Tenemos que $Rf = Rg \neq 0$ si y solo si $f \mid_d g$ y $g \mid_d f$. Así, $g = hf$ y $f = lg$, por lo que $g = hlg$. Por tanto, $hl = lh = 1$ por lo que h y l son unidades de R. Se dice entonces que f y g son *asociados por la izquierda* en el sentido de que $g = uf$, siendo u una unidad de R.

Se tiene que $Rf + Rg = Rh$. Entonces $h \mid_d f$ y $h \mid_d g$. De hecho si $l \mid_d f$ y $l \mid_d f$ entonces $Rf \subset Rl$ y $Rg \subset Rl$, por lo que $Rh \subset Rl$ y $l \mid_d h$. Por tanto h es un *máximo común divisor por la derecha* de f y g y lo notamos como $h = (f,g)_d$. Dos máximo común divisor por la derecha de los mismos dos elementos son asociados por la izquierda.

Se puede comprobar que R satisface la condición de Ore por la izquierda (ver Jacobson, 1996, p. 4), por lo que si $f \neq 0$ y $g \neq 0$ se tiene que $Rf \cap Rg \neq$

0. Tenemos por tanto que $Rf \cap RgRh$ para algún h por lo que $m = g'f = f'g$. De hecho si $f \mid_d l$ y $g \mid_d l$ entonces $Rl \subset Rf \cap Rg = Rh$, por lo que $h \mid_d l$. Por tanto h es un *mínimo común múltiplo por la izquierda* y lo notamos por $h = [f, g]_i$. De nuevo, dos mínimo común múltiplo por la izquierda de los mismos dos elementos son asociados por la izquierda.

Puede definirse una versión del algoritmo extendido de Euclides en este contexto (ver el algoritmo 4), que nos permite calcular tanto el máximo común divisor como el mínimo común múltiplo.

Entrada: polinomios $f, g \in \mathbb{F}_q[x; \sigma]$ con $f \neq 0, g \neq 0$
Salida: un número $n \in \mathbb{N}$, polinomios $u_i, v_i, q_i, f_i \in \mathbb{F}_q[x; \sigma]$ tales que $f_i = u_i f + v_i g$, $q_i = \operatorname{coi}(f_{i-1}, f_i)$, para $1 \leq i \leq n+1$ y $f_n = (f, g)_d$, $u_n f = -v_n g = [f, g]_i$.

$u_0 \longleftarrow v_1 = 1$
$u_1 \longleftarrow v_0 = 1$
$f_0 \longleftarrow f$
$f_1 \longleftarrow g$
$i \longleftarrow 1$
mientras $f_i \neq 0$ **hacer**
  $q_i \longleftarrow \operatorname{coi}(f_{i-1}, f_i)$
  $u_{i+1} \longleftarrow u_{i-1} - q_i u_i$
  $v_{i+1} \longleftarrow v_{i-1} - q_i v_i$
  $f_{i+1} \longleftarrow f_{i-1} - q_i f_i$
  $n \longleftarrow i$
fin

Algoritmo 4: Algoritmo extendido de Euclides por la izquierda en $\mathbb{F}_q[x; \sigma]$

5.4 DESCOMPOSICIÓN EN FACTORES IRREDUCIBLES

Como R es un dominio de ideales principales es posible descomponer cada polinomio $f \in R$ en un producto de factores irreducibles. Pero vamos a ver que esta factorización no es única.

Decimos que dos polinomios $f, g \in R$ distintos de cero son *similares por la izquierda*, que notamos $f \sim_i g$ si existe un polinomio $h \in R$ tal que

$$(h, g)_d = 1 \quad \text{y} \quad f = [g, h]_i h^{-1}.$$

La condición $(h, g)_d = 1$ equivale a que existan a y $b \in R$ tales que

$$1 = ah + bg$$

y $f = [g,h]_i h^{-1}$ equivale a que

$$l = h'g = fh,$$

donde $(h', f)_i = 1$. Por tanto tenemos un h' tal que $(h', f)_i = 1$ y $g = h'^{-1}[h', f]_d$. Por tanto si f es similar por la izquierda a g entonces g es similar por la derecha a f, por lo que escribiremos simplemente que $f \sim g$. Es posible comprobar que la *similitud* es una relación de equivalencia (ver Jacobson, 1996, p. 11).

TEOREMA 5.4.1. *Si $f = p_1 \dots p_r$ y $f = q_1 \dots q_t$ son factorizaciones de $f \in R$ como producto de irreducibles entonces $r = t$ y salvo una posible reordenación, $q_i \sim p_i$.*

Demostración. Puede consultarse una generalización de la demostración en (Jacobson, 1996, Teorema 1.2.9). □

Comprobar si dos polinomios $f, g \in R$ verifican que $f \sim g$ supone un problema complicado para el que no hay un procedimiento aplicable en la práctica.

5.5 NORMA

A continuación vamos a introducir un concepto que es muy útil a la hora de calcular los restos obtenidos al dividir un polinomio.

Definimos la *norma i-ésima* de un elemento $\gamma \in \mathbb{F}_q$ como

$$N_i(\gamma) = \sigma(N_{i-1}(\gamma))(\gamma) = \sigma^{i-1}(\gamma) \dots \sigma(\gamma)\gamma \quad \text{para } i > 0 \quad \text{y } N_0(\gamma) = 1.$$

PROPOSICIÓN 5.5.1. *Si $f(x) = \sum_0^n a_i x^{n-i} \in \mathbb{F}_q[x;\sigma]$ y $\gamma \in \mathbb{F}_q$ entonces $(x-\gamma) \mid_d f(x)$ si y solo si $\sum_0^n a_i N_i(\gamma) = 0$.*

Demostración. Observamos primero lo siguiente, que es de hecho un caso especial de la proposición:

$$\text{Para todo } n \geq 0, \qquad x^n - N_n(\gamma) \in R(x-\gamma). \tag{5.1}$$

Se demuestra por inducción. El caso base $n = 0$ es trivial, pues $x^0 - N_0(\gamma) = 0 \in R(x-\gamma)$. Supuesto cierto para n, para $n+1$ se tiene

$$\begin{aligned} x^{n+1} - N_{n+1}(\gamma) &= x^{n+1} - \sigma(N_n(\gamma))(\gamma) \\ &= x^{n+1} - \sigma(N_n(\gamma))(\gamma) + \sigma(N_n(\gamma))x - \sigma(N_n(\gamma))x \\ &= x^{n+1} + \sigma(N_n(\gamma))(x-\gamma) - \sigma(N_n(\gamma))x \\ &= \sigma(N_n(\gamma))(x-\gamma) + xx^n - xN_n(\gamma) \\ &= \sigma(N_n(\gamma))(x-\gamma) + x(x^n - N_n(\gamma)) \in R(x-\gamma). \end{aligned}$$

Usando ahora (5.1) tenemos que

$$f(x) - \sum a_i N_i(\gamma) = \sum a_i (x^i - N_i(\gamma)) \in R(x-\gamma)$$

y por tanto $r = \sum a_i N_i(\gamma)$. □

También se dan las siguientes identidades, que nos serán útiles cuando estudiemos los códigos cíclicos sesgados en el capítulo siguiente. Dados $\alpha, \beta, \gamma \in \mathbb{F}_q$ tales que $\beta = \alpha^{-1}\sigma(\alpha)$ se tiene que

$$\begin{aligned} N_i(\sigma^k(\gamma)) &= \sigma^k(N_i(\gamma)), \\ N_i(\sigma^k(\beta)) &= \sigma^k(\alpha)^{-1}\sigma^{k+1}(\alpha). \end{aligned} \tag{5.2}$$

6

CÓDIGOS CÍCLICOS SESGADOS

En este capítulo definiremos los códigos cíclicos sesgados sobre un cuerpo finito $\mathbb{F}_q$. Las principales fuentes de este capítulo son (Gomez-Torrecillas, Lobillo & Navarro, 2016), (Gómez-Torrecillas, Lobillo & Navarro, 2018), (Shi, Alahmadi & Sole, 2017) y (Lam & Leroy, 1988).

Consideremos el anillo de polinomios de Ore $R = \mathbb{F}_q[x;\sigma]$. Supondremos que el orden del automorfismo σ es n. Entonces por el teorema 5.2.1 el polinomio $x^n - 1$ es central en R y por tanto el ideal $(x^n - 1)$ es bilátero, por lo que podemos definir el anillo cociente $\mathcal{R} = \mathbb{F}_q[x;\sigma]/(x^n - 1)$. Este cociente $\mathcal{R}$ es isomorfo a $\mathbb{F}_q^n$ mediante la aplicación de coordenadas $\mathfrak{v} : \mathcal{R} \to \mathbb{F}_q^n$.

DEFINICIÓN 6.0.1. Un *código cíclico sesgado* sobre $\mathbb{F}_q$ es un subespacio vectorial $\mathcal{C} \subseteq \mathbb{F}_q^n$ tal que $\mathfrak{v}^{-1}(\mathcal{C})$ es un ideal por la izquierda de $\mathcal{R}$. Equivalentemente, es un subespacio vectorial $\mathcal{C} \subseteq \mathbb{F}_q^n$ tal que si $(a_0, \dots, a_{n-2}, a_{n-1}) \in \mathcal{C}$ entonces $(\sigma(a_{n-1}), \sigma(a_0), \dots, \sigma(a_{n-2})) \in \mathcal{C}$.

Sabemos que todo ideal por la izquierda de $\mathcal{R}$ es principal, por lo que todo código cíclico sesgado está generado por un polinomio en R. De forma análoga a como ocurría con los códigos cíclicos, este generador es un divisor por la derecha de $x^n - 1$, por lo que nos interesa conocer de nuevo la descomposición de $x^n - 1$ en factores, esta vez sobre R.

No existe un algoritmo de factorización completo para los polinomios de Ore, por lo que vamos a utilizar a continuación un método concreto para $x^n - 1$, descrito en (Gomez-Torrecillas y col., 2016). Por el teorema de la base normal podemos tomar un elemento $\alpha \in \mathbb{F}_q$ tal que $\{\alpha, \sigma(\alpha), \dots, \sigma^{n-1}(\alpha)\}$ sea una base de $\mathbb{F}_q$ como $\mathbb{F}_q^\sigma$-espacio vectorial. Fijamos en lo que sigue $\beta = \alpha^{-1}\sigma(\alpha)$.

LEMA 6.0.2. *Para cada subconjunto* $\{t_1, t_2, \dots, t_m\} \subseteq \{0, 1, \dots, n-1\}$ *el polinomio*

$$g = \left[x - \sigma^{t_1}(\beta), x - \sigma^{t_2}(\beta), \dots, x - \sigma^{t_m}(\beta)\right]_i$$

tiene grado m. Por tanto, si $x - \sigma^s(\beta) \mid_d g$ *entonces* $s \in T$.

Demostración. Vamos a proceder por reducción al absurdo, suponiendo que $\mathrm{gr}(g) < m$, de forma que $g = \sum_{i=0}^{m-1} g_i x^i$. Como g es un múltiplo por la izquierda de $x - \sigma^{t_1}(\beta)$ para todo $1 \le j \le m$ por la proposición 5.5.1 se tiene que

$$\sum_{i=0}^{m-1} g_i N_i(\sigma^{t_j}(\beta)) = 0 \quad \text{para todo } 1 \le j \le m.$$

Estas ecuaciones definen un sistema homogéneo, cuya matriz de coeficientes es la transpuesta de

$$M = \begin{pmatrix} N_0(\sigma^{t_1}(\beta)) & N_0(\sigma^{t_2}(\beta)) & \dots & N_0(\sigma^{t_m}(\beta)) \\ N_1(\sigma^{t_1}(\beta)) & N_1(\sigma^{t_2}(\beta)) & \dots & N_1(\sigma^{t_m}(\beta)) \\ N_2(\sigma^{t_1}(\beta)) & N_2(\sigma^{t_2}(\beta)) & \dots & N_2(\sigma^{t_m}(\beta)) \\ \vdots & \vdots & \ddots & \vdots \\ N_{m-1}(\sigma^{t_1}(\beta)) & N_{m-1}(\sigma^{t_2}(\beta)) & \dots & N_{m-1}(\sigma^{t_m}(\beta)) \end{pmatrix}$$

Pero por (5.2) $|M| = 0$ si y solo si el determinante de la matriz

$$\begin{aligned} M' &= \begin{pmatrix} \sigma^{t_1}(\alpha) & \sigma^{t_2}(\alpha) & \dots & \sigma^{t_m}(\alpha) \\ \sigma^{t_1+1}(\alpha) & \sigma^{t_2+1}(\alpha) & \dots & \sigma^{t_m+1}(\alpha) \\ \sigma^{t_1+2}(\alpha) & \sigma^{t_2+2}(\alpha) & \dots & \sigma^{t_m+2}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{t_1+m-1}(\alpha) & \sigma^{t_2+m-1}(\alpha) & \dots & \sigma^{t_m+m-1}(\alpha) \end{pmatrix} \\ &= \begin{pmatrix} \sigma^{t_1}(\alpha) & \sigma^{t_2}(\alpha) & \dots & \sigma^{t_m}(\alpha) \\ \sigma(\sigma^{t_1}(\alpha)) & \sigma(\sigma^{t_2}(\alpha)) & \dots & \sigma(\sigma^{t_m}(\alpha)) \\ \sigma^2(\sigma^{t_1}(\alpha)) & \sigma^2(\sigma^{t_2}(\alpha)) & \dots & \sigma^2(\sigma^{t_m}(\alpha)) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{m-1}(\sigma^{t_1}(\alpha)) & \sigma^{m-1}(\sigma^{t_2}(\alpha)) & \dots & \sigma^{m-1}(\sigma^{t_m}(\alpha)) \end{pmatrix} \end{aligned}$$

es cero. Pero por (Gómez-Torrecillas y col., 2018, Lema 2.1) $|M'| \neq 0$, por lo que la única solución del sistema anterior es $g_0 = \cdots = g_{m-1} = 0$, lo que es una contradicción. Por tanto, $\mathrm{gr}(g) = m$. □

COROLARIO 6.0.3. *Se tiene que*

$$x^n - 1 = \left[x - \beta, x - \sigma(\beta), \dots, x - \sigma^{n-1}(\beta)\right]_i$$

Demostración. Como consecuencia del lema 6.0.2 se tiene que

$$\left[x - \beta, x - \sigma(\beta), \dots, x - \sigma^{n-1}(\beta)\right]_i$$

tiene grado n. Pero además, por (5.2) se tiene que

$$N_n(\sigma^k(\beta)) = \sigma^k(\alpha)^{-1}\sigma^{k+n}(\alpha) = \sigma^k(\alpha^{-1}\alpha) = 1$$

y por tanto

$$-1N_0(\sigma^k(\beta)) + 1N_n(\sigma^k(\beta)) = -1 + 1 = 0,$$

por lo que por la proposición 5.5.1 cada $x - \sigma^k(\beta)$ divide a $x^n - 1$ por la derecha para todo $0 \leq k \leq n-1$. Estas dos afirmaciones nos conducen a que

$$x^n - 1 = \left[x - \beta, x - \sigma(\beta), \dots, x - \sigma^{n-1}(\beta)\right]_i,$$

como queríamos. □

Este corolario nos permite afirmar que dados $\{t_1, \dots, t_k\} \subset \{0, 1, \dots, n-1\}$ el polinomio $g = [x - \sigma^{t_1}(\beta), \dots, x - \sigma^{t_k}(\beta)]_i$ genera un ideal por la izquierda $\mathcal{R}g$ tal que $\mathfrak{v}(\mathcal{R}g)$ es un código cíclico sesgado de dimensión $n - k$.

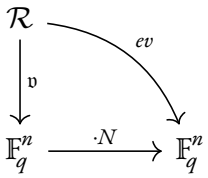
Antes de proceder a definir el tipo de códigos cíclicos sesgados que vamos a utilizar es necesario que comentemos algunos resultados que necesitaremos más adelante. Comenzamos indicando que llamaremos β-raíces a los elementos del conjunto $\{\beta, \sigma(\beta), \dots, \sigma^{n-1}(\beta)\}$. Por la proposición 5.5.1 y (5.2) se tiene que, dado un polinomio $f = \sum_{i=0}^{n-1} a_i x^i \in \mathcal{R}$,

$$x - \sigma^j(\beta) \mid_d f \iff \sum_{i=0}^{n-1} a_i N_i(\sigma^j(\beta)) = 0 \iff \sum_{i=0}^{n-1} a_i \sigma^{i+j}(\alpha) = 0. \quad (6.1)$$

Sea N la matriz formada por las normas de las β-raíces:

$$N = \begin{pmatrix} N_0(\beta) & N_0(\sigma(\beta)) & \cdots & N_0(\sigma^{n-1}(\beta)) \\ N_1(\beta) & N_1(\sigma(\beta)) & \cdots & N_1(\sigma^{n-1}(\beta)) \\ \vdots & \vdots & \ddots & \vdots \\ N_{n-1}(\beta) & N_{n-1}(\sigma(\beta)) & \cdots & N_{n-1}(\sigma^{n-1}(\beta)) \end{pmatrix}.$$

Las componentes de $\mathfrak{v}(f)N = (a_1, \dots, a_{n-1})N$ son las evaluaciones por la derecha de f en el conjunto de las β-raíces, es decir, el vector compuesto por los restos por la izquierda obtenidos al dividir f por los polinomios $x - \sigma^i(\beta)$ para $i = 0, \dots, n-1$. Por tanto, tenemos que el diagrama es conmutativo



en el que ev representa una aplicación que lleva cada polinomio f en la n-tupla formada por los restos por la izquierda de dividir f por $x - \sigma^i(\beta)$, para $i = 0, \dots, n-1$. Es posible probar que la matriz N es no singular (ver Gómez-Torrecillas y col., 2018, Lema 2.1), por lo que es un cambio de base de $\mathbb{F}_q^n$.

Dado un polinomio f llamamos *conjunto de β-raíces* del polinomio f al conjunto formado por las β-raíces γ que cumplen $x - \gamma \mid_d f$, es decir, a

aquellas correspondientes a las coordenadas nulas de $(a_0, \dots, a_{n-1})N$. Decimos que un divisor por la derecha no constante $f \mid_d x^n - 1$ *β-descompone totalmente* si existen $\{t_1, \dots, t_m\} \subseteq \{0, 1, \dots, n-1\}$ tales que

$$f = \left[x - \sigma^{t_1}(\beta), \dots, x - \sigma^{t_m}(\beta)\right]_i .$$

Sabemos por el lema 6.0.2 que $\operatorname{gr} f = m$, el cardinal del conjunto de las β-raíces de f.

LEMA 6.0.4. *Sea* $f = \sum_{i=0}^{m} a_i x^i \in \mathcal{R}$ *con* $a_m \neq 0$ *y*

$$M_f = \begin{pmatrix} a_0 & a_1 & \cdots & a_m & 0 & \cdots & 0 \\ 0 & \sigma(a_0) & \cdots & \sigma(a_{m-1}) & \sigma(a_m) & \cdots & 0 \\ & & \ddots & & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma^{n-m-1}(a_0) & \cdots & \cdots & \sigma^{n-m-1}(a_m) \end{pmatrix}_{(n-m)\times n} .$$

Entonces las filas de M_f *son la base de* $\mathfrak{v}(\mathcal{R}f)$ *como un* $\mathbb{F}_q$*-espacio vectorial. Es más,* f *β-descompone totalmente si y solo si*

$$\operatorname{mepf}(M_f N) = \begin{pmatrix} \varepsilon_{i_1} \\ \hline \vdots \\ \hline \varepsilon_{i_{m-n}} \end{pmatrix}$$

para algunos $0 \leq i_1 < \dots < i_{n-m} \leq n-1$*, donde* mepf *denota una matriz escalonada por filas y* ε_i *es un vector canónico de longitud* n[1].

Demostración. Una $\mathbb{F}_q$-base de $\mathcal{R}f$ es $\{f, xf, \dots, x^{n-m-1}f\}$ cuyas coordenadas corresponden precisamente a las filas de M_f. Teniendo esto en cuenta sabemos que $f = \left[x - \sigma^{t_1}(\beta), \dots, x - \sigma^{t_m}(\beta)\right]_i$ si y solo si cada múltiplo por la izquierda de f es también múltiplo por la izquierda de $x - \sigma^{t_i}(\beta)$ para $1 \leq i \leq m$. Ahora bien, recordemos que el producto matricial $M_f N$ representa en cada fila los restos por la izquierda obtenidos al dividir el polinomio correspondiente a los coeficientes de M_f en esa fila por los polinomios $x - \sigma^i(\beta)$ para $i = 0, \dots, n-1$. Es evidente entonces que $f = \left[x - \sigma^{t_1}(\beta), \dots, x - \sigma^{t_m}(\beta)\right]_i$ si y solo si las t_i-ésimas columnas de $M_f N$ son cero para $i = 1, \dots, m$. Como $M_f N$ tiene $n-m$ filas, rango $n-m$ y $n-m$ columnas distintas de cero, la forma escalonada por filas de la matriz $M_f N$ será justamente la que hemos descrito. □

En otras palabras, la proposición 6.0.4 nos dice que matriz M_f ahí definida es una matriz generadora del código $\mathfrak{v}(\mathcal{R}f)$.

1 Recordemos que un vector canónico ε_i de longitud n es de la forma $(0, \dots, 1, \dots, 0)$, una tupla de n elementos formada por 0 en todas las posiciones salvo en la i-ésima, que es 1

LEMA 6.0.5. *Sean $f, g \in \mathcal{R}$ polinomios que pueden β-descomponerse totalmente. Entonces $(f,g)_d$ y $[f,g]_i$ también pueden β-descomponerse totalmente.*

Demostración. Como f y g pueden β-descomponerse totalmente existen subconjuntos $T_1, T_2 \subseteq \{0, \dots, n-1\}$ tales que

$$f = \left[\{x - \sigma^i(\beta)\}_{i \in T_1}\right]_i \quad \text{y} \quad g = \left[\{x - \sigma^i(\beta)\}_{i \in T_2}\right]_i.$$

Se deduce rápidamente entonces que

$$[f,g]_i = \left[\{x - \sigma^i(\beta)\}_{i \in T_1 \cup T_2}\right]_i.$$

Por otro lado es evidente que

$$\left[\{x - \sigma^i(\beta)\}_{i \in T_1 \cap T_2}\right]_i \mid_d (f,g)_d,$$

pero como $\mathrm{gr}(f) + \mathrm{gr}(g) = \mathrm{gr}((f,g)_d) + \mathrm{gr}([f,g]_i)$ por el lema 6.0.2 se tiene la igualdad. □

Con todas estas herramientas ya podemos definir la clase de códigos que vamos a utilizar en el siguiente capítulo.

DEFINICIÓN 6.0.6. Bajo las condiciones y notación de este capítulo, un *código RS (Reed-Solomon) sesgado* de distancia mínima prevista δ es un código cíclico $\mathcal{C}$ tal que $\mathfrak{v}^{-1}(\mathcal{C})$ está generado por

$$\left[x - \sigma^r(\beta), x - \sigma^{r+1}(\beta), \dots, x - \sigma^{r+\delta-2}(\beta)\right]_i$$

para algún $r \geq 0$.

TEOREMA 6.0.7. *Un código RS sesgado de distancia mínima prevista δ tiene distancia mínima δ.*

Demostración. Vamos a denotar por

$$g = \left[x - \sigma^r(\beta), x - \sigma^{r+1}(\beta), \dots, x - \sigma^{r+\delta-2}(\beta)\right]_i$$

al generador de $\mathcal{C}$. Una matriz de paridad de este código es

$$H = \begin{pmatrix} N_0(\sigma^r(\beta)) & N_0(\sigma^{r+1}(\beta)) & \dots & N_0(\sigma^{r+\delta-2}(\beta)) \\ N_1(\sigma^r(\beta)) & N_1(\sigma^{r+1}(\beta)) & \dots & N_1(\sigma^{r+\delta-2}(\beta)) \\ N_2(\sigma^r(\beta)) & N_2(\sigma^{r+1}(\beta)) & \dots & N_2(\sigma^{r+\delta-2}(\beta)) \\ \vdots & \vdots & \ddots & \vdots \\ N_{n-1}(\sigma^r(\beta)) & N_{n-1}(\sigma^{r+1}(\beta)) & \dots & N_{n-1}(\sigma^{r+\delta-2}(\beta)) \end{pmatrix}$$

puesto que las columnas son la evaluaciones por la izquierda en las raíces. Por el corolario 2.1.23 tenemos que comprobar que todo menor de H de

tamaño $\delta - 1$ es distinto de cero. Recordemos que $N_i(\sigma^k(\beta)) = \sigma^k(N_i(\beta)) = \sigma^k(\alpha^{-1})\sigma^{i+k}(\alpha)$ para enteros i y k. Por tanto, dada una submatriz de orden $\delta - 1$,

$$M = \begin{pmatrix} N_{k_1}(\sigma^r(\beta)) & N_{k_1}(\sigma^{r+1}(\beta)) & \dots & N_{k_1}(\sigma^{r+\delta-2}(\beta)) \\ N_{k_2}(\sigma^r(\beta)) & N_{k_2}(\sigma^{r+1}(\beta)) & \dots & N_{k_2}(\sigma^{r+\delta-2}(\beta)) \\ N_{k_3}(\sigma^r(\beta)) & N_{k_3}(\sigma^{r+1}(\beta)) & \dots & N_{k_3}(\sigma^{r+\delta-2}(\beta)) \\ \vdots & \vdots & \ddots & \vdots \\ N_{k_{\delta-1}}(\sigma^r(\beta)) & N_{k_{\delta-1}}(\sigma^{r+1}(\beta)) & \dots & N_{k_{\delta-1}}(\sigma^{r+\delta-2}(\beta)) \end{pmatrix}$$

con $\{k_1 < k_2 < \dots < k_{\delta-1}\} \subset \{0, 1, \dots, n-1\}$, el determinante $|M| = 0$ si y solo si $|M'| = 0$, siendo M' la matriz

$$\begin{pmatrix} \sigma^{k_1+r}(\alpha) & \sigma^{k_1+r+1}(\alpha) & \dots & \sigma^{k_1+r+\delta-2}(\alpha) \\ \sigma^{k_2+r}(\alpha) & \sigma^{k_2+r+1}(\alpha) & \dots & \sigma^{k_2+r+\delta-2}(\alpha) \\ \sigma^{k_3+r}(\alpha) & \sigma^{k_3+r+1}(\alpha) & \dots & \sigma^{k_3+r+\delta-2}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_{\delta-1}+r}(\alpha) & \sigma^{k_{\delta-1}+r+1}(\alpha) & \dots & \sigma^{k_{\delta-1}+r+\delta-2}(\alpha) \end{pmatrix}$$

$$= \begin{pmatrix} \sigma^{k_1+r}(\alpha) & \sigma(\sigma^{k_1+r}(\alpha)) & \dots & \sigma^{\delta-2}(\sigma^{k_1+r}(\alpha)) \\ \sigma^{k_2+r}(\alpha) & \sigma(\sigma^{k_2+r}(\alpha)) & \dots & \sigma^{\delta-2}(\sigma^{k_2+r}(\alpha)) \\ \sigma^{k_3+r}(\alpha) & \sigma(\sigma^{k_3+r}(\alpha)) & \dots & \sigma^{\delta-2}(\sigma^{k_3+r}(\alpha)) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_v+r}(\alpha) & \sigma(\sigma^{k_v+r}(\alpha)) & \dots & \sigma^{\delta-2}(\sigma^{k_v+r}(\alpha)) \end{pmatrix}.$$

Como $\{\alpha, \sigma(\alpha), \dots, \sigma^{n-1}(\alpha)\}$ es una base de $\mathbb{F}_q$ como espacio vectorial del subcuerpo fijo $\mathbb{F}^\sigma$, por (Gómez-Torrecillas y col., 2018, Lema 2.1) $|M'| \neq 0$. □

7

ALGORITMO DE PETERSON-GORENSTEIN-ZIERLER PARA CÓDIGOS CÍCLICOS SESGADOS

En este capítulo nos adentramos finalmente en el algoritmo que es el objeto de nuestro estudio, el algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados, y es una exposición de los resultados descritos en (Gómez-Torrecillas y col., 2018).

Podemos asumir sin pérdida de generalidad que $\mathcal{C}$ es un código RS sesgado en sentido estricto, es decir, que bajo la definición que dimos en el capítulo anterior, el valor $r = 0$. De no ser así bastaría tomar $\alpha' = \sigma^r(\alpha)$, que nos daría una base normal, y tendríamos $\sigma^r(\beta) = \beta' = (\alpha')^{-1}\sigma(\alpha')$ por lo que el polinomio $[x - \beta', \dots, x - \sigma^{\delta-2}(\beta')]_i$ sería un generador del código $\mathcal{C}$. Suponemos entonces que el ideal por la izquierda $\mathfrak{v}^{-1}(\mathcal{C})$ está generado por $[x - \beta, x - \sigma(\beta), \dots, x - \sigma^{\delta-2}(\beta)]_i$ para algún $2 \leq \delta \leq n$. La distancia del código $\mathcal{C}$ es δ por el teorema 6.0.7 y demostraremos más adelante que el algoritmo que vamos a describir permite corregir hasta $t = \lfloor(\delta - 1)/2\rfloor$ errores.

Nos encontramos en la misma situación que en casos anteriores. Se quiere enviar un mensaje $\mathbf{m} = (m_0, \dots, m_{n-\delta})$ a través de un canal. Con la identificación de $\mathcal{C}$ y $\mathfrak{v}^{-1}(\mathcal{C})$podemos escribir $\mathbf{m}$ como un polinomio, de forma que tenemos que $m = \sum_{i=0}^{n-\delta} m_i x^i$. Codificamos el mensaje realizando el producto por el generador, $c = mg$. Suponemos que recibimos el mensaje $y = c + e$ donde $e = e_1 x^{k_1} + \cdots + e_v x^{k_v}$ con $v \leq t$ es el error que se ha producido durante la transmisión. Para determinarlo es necesario por tanto conocer tanto las magnitudes de error $(e_1, \dots, e_v)$ como las coordenadas de error $(k_1, \dots, k_v)$.

Vamos a describir un algoritmo para decodificar códigos cíclicos sesgados similar al ya descrito para códigos BCH. Los pasos que dimos entonces fueron:

1. Determinar los síndromes del mensaje recibido.
2. Encontrar el polinomio localizador.
3. Obtener las coordenadas de error k_j y las magnitudes de error e_j.
4. Hallar el vector de error $e(x)$ y restárselo al mensaje $y(x)$.

Esta versión del algoritmo sigue un esquema similar. Así, el primer paso es el del cálculo de los síndromes, que será un procedimiento análogo al realizado

entonces pero utilizando la definición de norma i-ésima de un elemento. Por tanto para cada $0 \leq i \leq 2t-1$ el *síndrome i-ésimo* s_i del polinomio recibido y se define como el resto de dividir por la izquierda dicho polinomio y entre $x - \sigma^i(\beta)$. Como c es divisible por la derecha por cada $x - \sigma^i(\beta)$ para $i = 0, \dots, \delta - 2$ se tiene que

$$\begin{aligned} s_i &= \sum_{j=0}^{n-1} y_j N_j(\sigma^i(\beta)) = \sum_{j=1}^{v} e_j N_{k_j}(\sigma^i(\beta)) \\ &= \sum_{j=1}^{v} e_j \sigma^i(\alpha^{-1}) \sigma^{i+k_j}(\alpha) = \sigma^i(\alpha^{-1}) \sum_{j=1}^{v} e_j \sigma^{i+k_j}(\alpha). \end{aligned} \tag{7.1}$$

La siguiente proposición nos proporciona una forma de obtener las magnitudes de error a partir de las coordenadas de error y los síndromes, tal y como ocurría en el caso de códigos BCH.

PROPOSICIÓN 7.0.1. *Las magnitudes de error* $(e_1, \dots, e_v)$ *son las soluciones del sistema de ecuaciones lineales*

$$X \begin{pmatrix} \sigma^{k_1}(\alpha) & \sigma^{k_1+1}(\alpha) & \dots & \sigma^{k_1+v-1}(\alpha) \\ \sigma^{k_2}(\alpha) & \sigma^{k_2+1}(\alpha) & \dots & \sigma^{k_2+v-1}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_v}(\alpha) & \sigma^{k_v+1}(\alpha) & \dots & \sigma^{k_v+v-1}(\alpha) \end{pmatrix} = (\alpha s_0, \sigma(\alpha) s_1, \dots, \sigma^{v-1}(\alpha) s_{v-1}).$$

Demostración. Comenzamos viendo que

$$\begin{aligned} &\begin{pmatrix} \sigma^{k_1}(\alpha) & \sigma^{k_1+1}(\alpha) & \dots & \sigma^{k_1+v-1}(\alpha) \\ \sigma^{k_2}(\alpha) & \sigma^{k_2+1}(\alpha) & \dots & \sigma^{k_2+v-1}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_v}(\alpha) & \sigma^{k_v+1}(\alpha) & \dots & \sigma^{k_v+v-1}(\alpha) \end{pmatrix} \\ &= \begin{pmatrix} \sigma^{k_1}(\alpha) & \sigma(\sigma^{k_1}(\alpha)) & \dots & \sigma^{v-1}(\sigma^{k_1}(\alpha)) \\ \sigma^{k_2}(\alpha) & \sigma(\sigma^{k_2}(\alpha)) & \dots & \sigma^{v-1}(\sigma^{k_2}(\alpha)) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_v}(\alpha) & \sigma(\sigma^{k_v}(\alpha)) & \dots & \sigma^{v-1}(\sigma^{k_v}(\alpha)) \end{pmatrix}, \end{aligned}$$

que por (Gómez-Torrecillas y col., 2018, Lema 2.1) es no singular. Por (7.1) tenemos que

$$\sigma^i(\alpha) s_i = \sum_{j=1}^{v} e_j \sigma^{i+k_j}(\alpha),$$

por lo que es evidente que dado $X = (e_1, \dots, e_v)$ tenemos que

$$(e_1, \dots, e_v) \begin{pmatrix} \sigma^{k_1}(\alpha) & \sigma^{k_1+1}(\alpha) & \dots & \sigma^{k_1+v-1}(\alpha) \\ \sigma^{k_2}(\alpha) & \sigma^{k_2+1}(\alpha) & \dots & \sigma^{k_2+v-1}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_v}(\alpha) & \sigma^{k_v+1}(\alpha) & \dots & \sigma^{k_v+v-1}(\alpha) \end{pmatrix} = \left(\sigma^i(\alpha) s_i\right)_{1\times v},$$

como queríamos demostrar. □

Como consecuencia de la proposición 7.0.1 el proceso de decodificación se reduce a encontrar las coordenadas de error $\{k_1, \dots, k_v\}$. Para ello al igual que hicimos en el caso del algoritmo para códigos BCH vamos a definir un polinomio localizador de errores, que en este caso será el que verifique que

$$\lambda = \left[x - \sigma^{k_1}(\beta), x - \sigma^{k_2}(\beta), \dots, x - \sigma^{k_v}(\beta)\right]_i.$$

Por el lema 6.0.2 sabemos que este polinomio λ tiene grado v. Son sus raíces las que nos permitirán determinar las coordenadas de error. El problema radica por tanto en encontrar este polinomio a partir de la información de la que disponemos.

Observamos que, por definición, un polinomio

$$f = \sum_{k=0}^{n-1} f_k x^k \in \mathcal{R}\lambda \text{ si y solo si } x - \sigma^{k_j}(\beta) \mid_d f \quad \text{para todo } j = 1, \dots, v,$$

o bien, utilizando la norma, si y solo si

$$\sum_{k=0}^{n-1} f_k N_k(\sigma^{k_j}(\beta)) = 0 \quad \text{para todo } j = 1, \dots, v.$$

De esta forma podemos expresar la condición $(f_0, \dots, f_{n-1}) \in \mathfrak{v}(\mathcal{R}\lambda)$ en forma de ecuación matricial, pues si $(f_0, \dots, f_{n-1}) \in \mathfrak{v}(\mathcal{R}\lambda)$ se verifica que $(f_0, \dots, f_n)T = 0$, donde

$$T = \begin{pmatrix} N_0(\sigma^{k_1}(\beta)) & N_0(\sigma^{k_2}(\beta)) & \dots & N_0(\sigma^{k_v}(\beta)) \\ N_1(\sigma^{k_1}(\beta)) & N_1(\sigma^{k_2}(\beta)) & \dots & N_1(\sigma^{k_v}(\beta)) \\ \vdots & \vdots & & \vdots \\ N_{n-1}(\sigma^{k_1}(\beta)) & N_{n-1}(\sigma^{k_2}(\beta)) & \dots & N_{n-1}(\sigma^{k_v}(\beta)) \end{pmatrix}.$$

Esto, junto a que por (5.2) podemos expresar las normas anteriores como $N_k(\sigma^{k_j}(\beta)) = \sigma^{k_j}(\alpha)^{-1}\sigma^{k_j+k}(\alpha)$, nos da una forma de obtener todos los

elementos de $\mathcal{R}\lambda$, pues como deben verificar la ecuación antes comentada, forman el núcleo por la izquierda de la matriz

$$\Sigma = \begin{pmatrix} \sigma^{k_1}(\alpha) & \sigma^{k_2}(\alpha) & \dots & \sigma^{k_v}(\alpha) \\ \sigma^{k_1+1}(\alpha) & \sigma^{k_2+1}(\alpha) & \dots & \sigma^{k_v+1}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_1+n-1}(\alpha) & \sigma^{k_2+n-1}(\alpha) & \dots & \sigma^{k_v+n-1}(\alpha) \end{pmatrix} = \left(\frac{\Sigma_0}{\Sigma_1}\right),$$

donde Σ_0 se corresponde a las primeras $v+1$ filas de la matriz Σ anterior.

Consideramos ahora la matriz

$$E = \begin{pmatrix} e_1 & \sigma^{-1}(e_1) & \dots & \sigma^{-v+1}(e_1) \\ e_2 & \sigma^{-1}(e_2) & \dots & \sigma^{-v+1}(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ e_v & \sigma^{-1}(e_v) & \dots & \sigma^{-v+1}(e_v) \end{pmatrix}.$$

Así podemos considerar la siguiente matriz, que expresaremos por sus entradas para la fila k y la columna i,

$$S = \Sigma E = \left(\sum_{j=1}^{v} \sigma^{-i}(e_j)\sigma^{k_j+k}(\alpha) \right)_{0\le k<n,\ 0\le i<v}.$$

Por (7.1), cuando $k+i < 2t-1$, la componente (k,i)-ésima puede escribirse como $\sigma^{-i}(s_{k+i})\sigma^{k}(\alpha)$, de tal forma que podemos dividir la matriz S como

$$S = \left(\frac{S_0}{S_1}\right),$$

donde S_0 viene dada por

$$S_0 = \begin{pmatrix} s_0\alpha & \sigma^{-1}(s_1)\alpha & \dots & \sigma^{-v+1}(s_{v-1})\alpha \\ s_1\sigma(\alpha) & \sigma^{-1}(s_1)\sigma(\alpha) & \dots & \sigma^{-v+1}(s_v)\sigma(\alpha) \\ \vdots & \vdots & & \vdots \\ s_v\sigma^{v}(\alpha) & \sigma^{-1}(s_{v+1})\sigma^{v}(\alpha) & \dots & \sigma^{-v+1}(s_{2v-1})\sigma^{v}(\alpha) \end{pmatrix}_{(v+1)\times v}$$

y cuyos coeficientes pueden calcularse directamente a partir del polinomio y, pues ya no aparecen los e_j. Para calcular el parámetro v, que es el número de coordenadas de error, utilizaremos un procedimiento análogo al usado en el algoritmo PGZ para códigos BCH cuando buscábamos una matriz

de síndromes no singular (de rango máximo). Para cualquier $1 \leq r \leq t$ denotaremos por S^r a la matriz

$$S^r = \begin{pmatrix} s_0\alpha & \sigma^{-1}(s_1)\alpha & \dots & \sigma^{-r+1}(s_{r-1})\alpha \\ s_1\sigma(\alpha) & \sigma^{-1}(s_1)\sigma(\alpha) & \dots & \sigma^{-r+1}(s_r)\sigma(\alpha) \\ \vdots & \vdots & & \vdots \\ s_t\sigma^t(\alpha) & \sigma^{-1}(s_{t+1})\sigma^t(\alpha) & \dots & \sigma^{-r+1}(s_{t+r-1})\sigma^t(\alpha) \end{pmatrix}_{(t+1)\times r}.$$

Igual que antes, tenemos que para todo $r \leq t$ se tiene que $S^r = \Sigma^t E^r$, donde

$$E^r = \begin{pmatrix} e_1 & \sigma^{-1}(e_1) & \dots & \sigma^{-r+1}(e_1) \\ e_2 & \sigma^{-1}(e_2) & \dots & \sigma^{-r+1}(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ e_v & \sigma^{-1}(e_v) & \dots & \sigma^{-r+1}(e_v) \end{pmatrix}_{v\times r}.$$

y

$$\Sigma^t = \begin{pmatrix} \sigma^{k_1}(\alpha) & \sigma^{k_2}(\alpha) & \dots & \sigma^{k_v}(\alpha) \\ \sigma^{k_1+1}(\alpha) & \sigma^{k_2+1}(\alpha) & \dots & \sigma^{k_v+1}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{k_1+t}(\alpha) & \sigma^{k_2+t}(\alpha) & \dots & \sigma^{k_v+t}(\alpha) \end{pmatrix}_{(t+1)\times v}.$$

LEMA 7.0.2. *Para cada $r \leq t$ se tiene que* $\operatorname{rg}(S^r) = \operatorname{rg}(\Sigma E^r) = \operatorname{rg}(E^r)$.

Demostración. Por (Gómez-Torrecillas y col., 2018, Lema 2.1) tiene que $\operatorname{rg}(\Sigma) = \operatorname{rg}(\Sigma^t) = v$. Usando la desigualdad del rango de Sylvester se tiene que

$$\text{mín}\{\operatorname{rg}(\Sigma), \operatorname{rg}(E^r)\} \geq \operatorname{rg}(\Sigma E^r) \geq \operatorname{rg}(\Sigma) + \operatorname{rg} E^r - v = \operatorname{rg}(E^r).$$

Por tanto $\operatorname{rg}(\Sigma E^r) = \operatorname{rg}(E^r)$. Con un razonamiento análogo se puede comprobar que $\operatorname{rg}(S^r) = \operatorname{rg}(E^r)$. □

Hemos de que calcular el mayor valor de r para el que la matriz S^r tenga rango máximo. Por el lema 7.0.2 que acabamos de demostrar es también el mayor valor de $r \leq t$ tal que las matrices E^r y ΣE^r tienen rango máximo. Denotaremos por μ a tal máximo.

LEMA 7.0.3. *Para cada r tal que $\mu \leq r \leq t$ se tiene que* $\operatorname{rg}(E^r) = \operatorname{rg}(S^r) = \mu$. *Por tanto, $\mu \leq v$.*

Demostración. Por el lema 7.0.2 $\mu = \operatorname{rg}(E^\mu) = \operatorname{rg}(S^\mu)$, por lo que suponemos que $\mu < r$. Por la maximalidad de μ tenemos que la $(\mu+1)$-ésima columna de E^r es una combinación lineal de las μ columnas anteriores. Aplicando σ^{-1} obtenemos que la $(\mu+2)$-ésima columna es una combinación

lineal de las columnas segunda a $\mu+1$-ésima, y por tanto una combinación lineal de las primeras μ columnas. Si repetimos el proceso obtenemos que todas las columnas desde la $(\mu+1)$-ésima hasta la r-ésima son combinaciones lineales de las primeras μ columnas, y por tanto $\operatorname{rg}(E^r) = \mu$. Como E^r tiene v filas, $\mu \leq v$. Finalmente, $\operatorname{rg}(S^r) = \mu$ de nuevo por el lema 7.0.2. □

PROPOSICIÓN 7.0.4. *El núcleo por la izquierda V de la matriz ΣE^μ es un código cíclico sesgado. Por tanto se tiene que $\mathfrak{v}^{-1}(V) = \mathcal{R}\rho$ para algún polinomio $\rho \in \mathcal{R}$ de grado μ. Se tiene además que ρ es un divisor por la derecha de λ.*

Demostración. Demostraremos la primera afirmación comprobando que si el vector $(a_0, \dots, a_{n-2}, a_{n-1}) \in V \subseteq \mathbb{F}_q^n$ también se da que el desplazamiento cíclico del vector anterior $(\sigma(a_{n-1}), \sigma(a_0), \dots, \sigma(a_{n-2})) \in V$. Recordemos que el desplazamiento tiene esta forma porque $xa_ix^i = \sigma(a_i)x^{i+1}$ para cada $0 \leq i \leq n-1$. Supongamos entonces que $(a_0, a_1, \dots, a_{n-1})\Sigma E^\mu = 0$. La maximalidad de μ nos asegura que la última columna de $E^{\mu+1}$ es una combinación lineal de las μ columnas anteriores. Por tanto, $(a_0, a_1, \dots, a_{n-1})\Sigma E^{\mu+1} = 0$. Así,

$$\begin{aligned} 0 &= (a_0, a_1, \dots, a_{n-1})\Sigma E^{\mu+1} \\ &= (a_0, a_1, \dots, a_{n-1}) \left(\begin{array}{c|c} 0 & I_{n-1} \\ \hline 1 & 0 \end{array}\right) \left(\begin{array}{c|c} 0 & 1 \\ \hline I_{n-1} & 0 \end{array}\right) \Sigma E^{\mu+1} \\ &= (a_{n-1}, a_0, \dots, a_{n-2}) \left(\begin{array}{c|c} 0 & 1 \\ \hline I_{n-1} & 0 \end{array}\right) \Sigma E^{\mu+1}. \end{aligned}$$

Si aplicamos σ a esta ecuación matricial componente a componente obtenemos

$$(\sigma(a_{n-1}), \sigma(a_0), \dots, \sigma(a_{n-2})) \left(\begin{array}{c|c} 0 & 1 \\ \hline I_{n-1} & 0 \end{array}\right) \Sigma\sigma(E)\sigma(E^\mu) = 0.$$

Observamos que

$$\Sigma = \left(\begin{array}{c|c} 0 & 1 \\ \hline I_{n-1} & 0 \end{array}\right) \sigma(\Sigma) \quad \text{y} \quad \sigma(E^{\mu+1}) = \left(\begin{array}{c|c} \sigma(e_1) & \\ \vdots & E^\mu \\ \sigma(e_v) & \end{array}\right)$$

por lo que

$$(\sigma(a_{n-1}), \sigma(a_0), \dots, \sigma(a_{n-2}))\Sigma \left(\begin{array}{c|c} \sigma(e_1) & \\ \vdots & E^\mu \\ \sigma(e_v) & \end{array}\right) = 0.$$

En particular, $(\sigma(a_{n-1}), \sigma(a_0), \dots, \sigma(a_{n-2}))\Sigma E^{\mu} = 0$ por lo que el desplazamiento cíclico $(\sigma(a_{n-1}), \sigma(a_0), \dots, \sigma(a_{n-2})) \in V$, como queríamos. Además, dado que cualquier ideal de $\mathcal{R}$ es principal, $\mathfrak{v}^{-1}(V)$ lo es y estará generado por un polinomio $\rho \in \mathcal{R}$. Como $\mathfrak{v}(\mathcal{R}\lambda)$ es el núcleo por la izquierda de la matriz Σ por la definición de $\mathfrak{v}(\mathcal{R}\rho)$ como núcleo de ΣE^{μ} se tiene que $\mathcal{R}\lambda \subseteq \mathcal{R}\rho$, y por tanto ρ divide por la derecha a λ. Finalmente la dimensión de $\mathcal{R}\rho$ como un $\mathbb{F}_q$ espacio vectorial es $n - \mathrm{gr}(\rho)$. Por el lema 7.0.2 se tiene que $\mathrm{rg}(\Sigma E^{\mu}) = \mu$ y por tanto $\mathrm{gr}(\rho) = \mu$. □

El lema siguiente es el que nos va a proporcionar la forma de encontrar el polinomio ρ que estamos buscando.

LEMA 7.0.5. *La forma escalonada por columnas de la matriz S^t es*

$$\mathrm{mepc}(S^t) = \left(\begin{array}{c|c} \begin{array}{c} I_{\mu} \\ \hline a_0 \cdots a_{\mu-1} \\ \hline H' \end{array} & 0_{(t+1)\times(t-\mu)} \end{array}\right),$$

donde I_{μ} es la matriz identidad $\mu \times \mu$ y $a_0, \dots, a_{\mu-1} \in \mathbb{F}_q$ tales que $\rho = x^{\mu} - \sum_{i=0}^{\mu-1} a_i x^i$.

Demostración. Por el lema 7.0.3 el $\mathrm{rg}(S^t) = \mu = \mathrm{rg}(S^{\mu})$, por lo que

$$\mathrm{mrpc}(S^t) = \left(\mathrm{mrpc}(S^{\mu}) \mid 0_{(t+1)\times(t-\mu)}\right).$$

La matriz S^{μ} consiste en las primeras $t+1$ filas de ΣE^{μ} y ambas tienen el mismo rango μ, por lo que $\mathrm{mrpc}(S^{\mu})$ está formada por las primeras $t+1$ filas de $\mathrm{mrpc}(\Sigma E^{\mu})$. Por la proposición 7.0.4 $\mathfrak{v}(\mathcal{R}\rho)$ es el núcleo por la izquierda de la matriz $\mathrm{mrpc}(\Sigma E^{\mu})$. Una solución no nula del sistema homogéneo

$$X\left(\mathrm{mepc}(\Sigma E^{\mu}) \;\middle|\; \begin{array}{c} 0 \\ \hline I_{n-(\mu+1)} \end{array}\right) = 0 \tag{7.2}$$

es un elemento distinto de cero de $\mathfrak{v}(\mathcal{R}(\rho))$ cuyas últimas $n - (\mu + 1)$ coordenadas son cero. Como ρ tiene grado μ y su grado es mínimo en $\mathcal{R}\rho$ se deduce que $\mathfrak{v}(\rho)$ es la única solución, salvo producto por escalares de (7.2). Sea S_0^{μ} la matriz formada por las primeras $\mu + 1$ filas de S^{μ}. Entonces

$$\mathrm{mepc}(S^{\mu}) = \left(\begin{array}{c} \mathrm{mepc}(S_0^{\mu}) \\ \hline H' \end{array}\right).$$

Si realizamos más reducciones de columnas utilizando la matriz identidad en el bloque derecho de la matriz (7.2) podemos ver que ρ es también la solución no nula, salvo producto por escalares, del sistema homogéneo

$$X\left(\begin{array}{c|c} \operatorname{mepc}(S_0^{\mu}) & 0 \\ \hline 0 & I_{n-(\mu+1)} \end{array}\right) = 0. \tag{7.3}$$

El tamaño de $\operatorname{mepc}(S_0^{\mu})$ es $(\mu+1)\times\mu$. De hecho $\operatorname{rg}(\operatorname{mepc}(S_0^{\mu})) = \mu$ porque el espacio de soluciones de (7.3) tiene dimensión 1. Por tanto solo hay una fila de $\operatorname{mepc}(S_0^{\mu})$ sin pivote. Si no es la última entonces habría un polinomio no nulo de $\mathcal{R}\rho$ de grado estrictamente menor que μ, lo cual es imposible. Por tanto,

$$\operatorname{mepc}(S_0^{\mu}) = \left(\begin{array}{c} I_{\mu} \\ \hline a_0 \dots a_{\mu-1} \end{array}\right).$$

Finalmente $(-a_0, \dots, -a_{\mu-1}, 1, 0 \dots, 0)$ es una solución no nula de (7.3), por lo que $\rho = x^{\mu} - \sigma_{i=0}^{\mu-1} a_i x^i$. □

LEMA 7.0.6. *Si el ideal por la izquierda $\mathcal{R}\rho$ se corresponde, mediante $\mathfrak{v}$, con el núcleo por la izquierda de una matriz H entonces $H = \Sigma B$ para alguna matriz $B \in \mathcal{M}_{v\times\mu}(L)$ que no tiene ninguna fila nula.*

Demostración. Podemos deducir este resultado a partir del siguiente diagrama conmutativo de $\mathbb{F}$-espacios vectoriales.

$$\begin{array}{ccccccccc} 0 & \longrightarrow & \mathcal{R}\rho & \longrightarrow & \mathcal{R} & \xrightarrow{\cdot H} & \mathcal{R}/\mathcal{R}\rho & \longrightarrow & 0 \\ & & \uparrow & & \| & & \uparrow{\scriptstyle \cdot B} & & \\ 0 & \longrightarrow & \mathcal{R}\lambda & \longrightarrow & \mathcal{R} & \xrightarrow{\cdot \Sigma} & \mathcal{R}/\mathcal{R}\lambda & \longrightarrow & 0 \end{array}$$

Si $\mathcal{R}\rho$ se corresponde con el núcleo por la izquierda de una matriz H entonces existe una aplicación lineal sobreyectiva $\mathcal{R}/\mathcal{R}\lambda \to \mathcal{R}/\mathcal{R}\rho$ definida por la multiplicación por la izquierda por una matriz B de tamaño $v\times\mu$ tal que $\Sigma B = H$. Como ρ es también el núcleo por la izquierda de ΣE^{μ} existe una matriz no singular P de tamaño $\mu\times\mu$ tal que $\Sigma E^{\mu} P = \Sigma B$. Como Σ define una aplicación lineal sobreyectiva, $E^{\mu} P = B$. Finalmente, B se obtiene a partir de E^{μ} realizando operaciones elementales sobre las columnas. Como E^{μ} no tiene filas nulas, B tampoco. □

La siguiente proposición nos ilustra sobre la relación existente entre el polinomio ρ que hemos obtenido y el polinomio localizador de errores λ.

PROPOSICIÓN 7.0.7. *Sea $\lambda' \in \mathcal{R}$ un polinomio que β-descompone totalmente y es múltiplo de ρ. Entonces, $\lambda \mid_d \lambda'$.*

Demostración. Por la proposición 7.0.4 se tiene que $\rho \mid_d \lambda$ y, por hipótesis, $\rho \mid_d \lambda'$. Así, $\rho \mid_d (\lambda, \lambda')_d$. De hecho, por el lema 6.0.5 el polinomio $(\lambda, \lambda')_d$ también β-descompone totalmente. Denotemos por $\phi = (\lambda, \lambda')_d$. Vamos a demostrar que $\phi = \lambda$, lo que implica el hecho que queremos demostrar.

Por definición $\mathcal{R}\lambda \subseteq \mathcal{R}\phi$, por lo que $\mathcal{R}\phi$ se corresponde con el núcleo por la izquierda de una matriz ΣQ, donde Q es una matriz de rango máximo. De forma análoga se tiene que $\mathcal{R}\phi \subseteq \mathcal{R}\rho$, por lo que existe otra matriz Q' tal que $\mathcal{R}\rho$ es el núcleo por la izquierda de $\Sigma QQ'$. Por el lema 7.0.6 se tiene que $\Sigma QQ' = \Sigma B$, donde B es una matriz de rango máximo y sin ninguna fila nula. Por tanto $QQ' = B$, porque Σ define una aplicación lineal sobreyectiva y Q no tiene filas nulas.

Como $\phi \mid_d \lambda$ cualquier β-raíz de ϕ tiene que ser también β-raíz de λ por lo que pertenece al conjunto $\{\sigma^{k_1}(\beta), \dots, \sigma^{k_v}(\beta)\}$. Obsérvese que por (6.1) se tiene que $\sigma^{k_j}(\beta)$ es una β-raíz de ϕ si y solo si

$$\operatorname{rg}\left(\begin{array}{c|c} \Sigma Q & \begin{matrix} \sigma^{k_j}(\alpha) \\ \sigma^{k_j+1}(\alpha) \\ \vdots \\ \sigma^{k_j+n-1}(\alpha) \end{matrix} \end{array}\right) = \operatorname{rg}(\Sigma Q).$$

Por tanto, por el lema (Gómez-Torrecillas y col., 2018, Lema 2.3) que ϕ se pueda β-descomponer totalmente implica que $\{\sigma^{k_1}(\beta), \dots, \sigma^{k_v}(\beta)\}$ es el conjunto de β-raíces de ϕ. Por tanto, $\phi = \lambda$. □

Ya estamos en disposición de calcular el polinomio localizador de errores y por tanto hemos completado el diseño del algoritmo de Peterson-Gorenstein-Zierler para códigos cíclicos sesgados, que puede consultarse en el algoritmo 5.

TEOREMA 7.0.8. *Sea $\mathbb{F}_q$ un cuerpo finito, $\sigma \in \operatorname{Aut}(\mathbb{F}_q)$ de orden n y $\mathbb{F}_q^\sigma$ el subcuerpo invariante del generado por σ. Sea $\{\alpha, \sigma(\alpha), \dots, \sigma^{n-1}(\alpha)\}$ una base normal de $\mathbb{F}_q$ sobre $\mathbb{F}_q$ y $\beta = \alpha^{-1}\sigma(\alpha)$. Sean $\mathcal{R} = \mathbb{F}_q[x;\sigma]/(x^n - 1)$, $g = \left[x - \beta, \dots, x - \sigma^{\delta-2}(\beta)\right]_i$ y $\mathcal{C}$ el código RS sesgado tal que $\mathfrak{v}^{-1}(\mathcal{C}) = \mathcal{R}g$. Entonces el algoritmo descrito en el algoritmo 5 encuentra correctamente el error $e = (e_0, \dots, e_{n-1})$ de cualquier vector recibido si el número de coordenadas distintas de cero de e es $v \leq t = \lfloor(\delta - 1)/2\rfloor$.*

Demostración. Tras los ajustes iniciales la línea 9 calcula un polinomio $\rho = \sum_{i=0}^{\mu} \rho_i x^i$ como describe el lema 7.0.5 y que, por la proposición 7.0.4, es divisor por la izquierda del polinomio localizador de errores λ.

Por (6.1) la línea 10 calcula todas las β-raíces de ρ. Sabemos por el lema 6.0.2 que el número de β-raíces es $v = \mu$ si y solo si ρ —de grado μ— puede β-descomponerse totalmente. En ese caso, por la proposición 7.0.7 $\rho = \lambda$.

Entrada: el código $\mathcal{C}$, el mensaje recibido $y = (y_0, \dots, y_{n-1}) \in \mathbb{F}_q^n$ con no más de t errores

Salida: el error $e = (e_0, \dots, e_{n-1})$ tal que $y - e \in \mathcal{C}$

// Paso 1: calcular síndromes

para $0 \le i \le 2t-1$ **hacer**

  $s_i \longleftarrow \sum_{j=0}^{n-1} y_j N_j(\sigma^i(\beta))$

fin

si $s_i = 0$ *para todo* $0 \le i \le 2t-1$ **entonces**

  devolver 0

fin

// Paso 2: hallar polinomio localizador y las coordenadas de error

$S^t \longleftarrow \left(\sigma^{-j}(s_{i+j})\sigma^i(\alpha)\right)_{0\le i\le t, 0\le j\le t-1}$

Calcular

$$\operatorname{mepc}(S^t) = \left(\begin{array}{c|c} \dfrac{I_\mu}{a_0 \cdots a_{\mu-1}} & 0_{(t+1)\times(t-\mu)} \\ \hline H' & \end{array}\right)$$

$\rho = (\rho_0, \dots, \rho_\mu) \longleftarrow (-a_0, \dots, -a_{\mu-1}, 1)$ y $\rho N \longleftarrow (\rho_0, \dots, \rho_\mu, 0, \dots, 0)N$

$\{k_1, \dots, k_v\} \longleftarrow$ coordenadas igual a cero de ρN

si $\mu \ne v$ **entonces**

  Calcular

$$M_\rho \longleftarrow \begin{pmatrix} \rho_0 & \rho_1 & \cdots & \rho_\mu & 0 & \cdots & 0 \\ 0 & \sigma(\rho_0) & \cdots & \sigma(\rho_{\mu-1}) & \sigma(\rho_\mu) & \cdots & 0 \\ & & \ddots & & & \ddots & \\ 0 & \cdots & 0 & \sigma^{n-\mu-1}(\rho_0) & \cdots & \cdots & \sigma^{n-\mu-1}(\rho_\mu) \end{pmatrix}_{(n-\mu)\times n}$$

  $N_\rho \longleftarrow M_\rho N$

  $H_\rho \longleftarrow \operatorname{mepf}(N_\rho)$

  $H' \longleftarrow$ la matriz obtenida al eliminar las filas de H_ρ distintas de ε_i para algún i

  $\{k_1, \dots, k_v\} \longleftarrow$ las coordenadas de las columnas igual a cero de H'

fin

// Paso 3: resolver el sistema de los síndromes, obteniendo las magnitudes de error

Encontrar $(x_1, \dots, x_v)$ tal que $(x_1, \dots, x_v)(\Sigma^{v-1})^T = (\alpha s_0, \sigma(\alpha)s_1, \dots, \sigma^{v-1}(\alpha)s_{v-1})$

// Paso 4: construir el error y devolverlo

devolver $(e_0, \dots, e_{n-1})$ *con* $e_i = x_i$ *para* $i \in \{k_1, \dots, k_v\}$, *cero en otro caso*

Algoritmo 5: Peterson-Gorenstein-Zierler para códigos cíclicos sesgados.

Si por el contrario $v \neq \mu$ como el $\operatorname{gr}(\rho) = \mu$ las filas de M_ρ generan $\mathcal{R}\rho$ como un $\mathbb{F}_q$ espacio vectorial y las fila de N_ρ también generan $\mathcal{R}\rho$ bajo el cambio de base correspondiente a N. Como H_ρ es la forma reducida por columnas de M_ρ entonces sus filas también son una base de ρ como un $\mathbb{F}_q$-espacio vectorial. Por el lema 6.0.4 las filas de H' generan un $\mathbb{F}_q$-subespacio vectorial $\mathcal{R}\lambda'$ para algún polinomio λ' que puede β-descomponerse totalmente. Como H' se obtiene eliminando algunas filas de H_ρ el polinomio λ' tiene que tener mayor grado que ρ y se deduce que $\rho \mid_d \lambda'$.

Vamos a ver que el polinomio que hemos encontrado es el polinomio localizador de errores, $\lambda' = \lambda$. Como $\rho \mid_d \lambda'$ por la proposición 7.0.7 se tiene que $\lambda \mid_d \lambda'$. Procederemos por reducción al absurdo. Supongamos entonces que $\lambda \neq \lambda'$. Entonces la matriz $H_\lambda = \operatorname{mepf}(M_\lambda N)$ contiene al menos una fila adicional, que será un vector canónico ε_d, que no está en H'. Como $\rho \mid_d \lambda$ por definición se tiene que $\mathcal{R}\lambda \subseteq \mathcal{R}\rho$ y por tanto,

$$\operatorname{rg}\left(\frac{H_\rho}{\varepsilon_d}\right) = \operatorname{rg}(H_\rho).$$

Por (Gómez-Torrecillas y col., 2018, Lema 2.4) ε_d es una fila de H_ρ, por lo que la línea 15 no la elimina, y en consecuencia, ε_d pertenece a H', lo que se contradice con la afirmación anterior. Concluimos entonces que $\lambda = \lambda'$. Una vez obtenido el polinomio localizador de errores λ podemos calcular las coordenadas de error a partir de sus β-raíces. Finalmente, por la proposición 7.0.1, la línea 18 calcula las magnitudes de error. Con ellas ya podemos construir el polinomio de error en la línea 19 y obtener el mensaje. □

En cuanto a lo que a la eficiencia del algoritmo respecta, la complejidad del mismo está dominada por el cálculo de las formas reducidas por filas de las líneas 8 y 14. Dichas operaciones tienen un orden de complejidad de $\mathcal{O}(t^3)$ y $\mathcal{O}(n^3)$, respectivamente. Como en el peor de los casos $t \approx n/2$, el orden de complejidad de este algoritmo es $\mathcal{O}(n^3)$. Sin embargo, la mayoría de veces la condición de la línea 11 no se verifica, y por tanto solo se realiza uno de los dos cálculos de matrices escalonadas (ver Gómez-Torrecillas y col., 2018, Remark 1).

Veamos a continuación algunos ejemplos utilizando para ello la implementación en SageMath comentada en el anexo A.

EJEMPLO 7.0.9. Sea $\mathbb{F} = \mathbb{F}_2(a)$ un cuerpo con $2^{12} = 1024$ elementos, donde se verifica la relación $a^{12} + a^7 + a^6 + a^5 + a^3 + a + 1 = 0$. Consideremos el automorfismo $\sigma : \mathbb{F} \to \mathbb{F}$ dado por $\sigma = \sigma_2^{10}$, donde σ_2 es el automorfismo de Frobenius, de tal forma que $\sigma(a) = a^{1024}$. El orden de σ es 6 por lo que un código cíclico sobre $\mathbb{F}$ es un ideal por la izquierda del cociente $\mathcal{R} = \mathbb{F}[x;\sigma]/(x^6 - 1)$. Tomaremos $\alpha = a$, lo que nos proporciona una base

normal de $\mathbb{F}$, y $\beta = \sigma(a)a^{-1} = a^{1023}$. Las imágenes de β por las potencias de σ nos da el conjunto $\{a^{1023}, a^{3327}, a^{3903}, a^{4047}, a^{4083}, a^{4092}\}$. Consideremos el código RS sesgado generado por

$$\begin{aligned} g &= \left[x - a^{1023}, x - a^{3327}, x - a^{3903}, x - a^{4047}\right]_i \\ &= x^4 + a^{2103}x^3 + a^{687}x^2 + a^{1848} + a^{759}. \end{aligned}$$

Vamos a seguir este ejemplo utilizando SageMath.

```
sage: F.<a> = GF(2^12)
sage: a^12
> a^7 + a^6 + a^5 + a^3 + a + 1
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma); S
> Skew Polynomial Ring in x over Finite Field in a of size
    2^12 twisted by a |--> a^(2^10)
sage: b = a^-1*sigma(a)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a^3903, x
    - a^4047])
sage: C = SkewRSCode(generator_pol=g); C
> [6, 2] Skew RS Code on Skew Polynomial Ring in x over
    Finite Field in a of size 2^12 twisted by a |--> a
    ^(2^10)
```

Definimos el decodificador basado en el algoritmo PGZ.

```
sage: D = SkewRSPGZDecoder(C); D
> Peterson-Gorenstein-Zierler algorithm based decoder for
    [6, 2] Skew RS Code on Skew Polynomial Ring in x over
    Finite Field in a of size 2^12 twisted by a |--> a
    ^(2^10)
sage: D.correction_capability()
> 2
```

Supongamos que queremos enviar el mensaje $m = x + a$, por lo que el polinomio codificado se obtiene como $c = mg$.

```
sage: c = C.encode(x + a, "SkewCyclicPolynomialEncoder");
    c
> (a^9 + a^8 + a^6 + a^5 + a^2 + a + 1, a^9 + a^6 + a^5 +
    a^4 + a^3 + 1, a^11 + a^9 + a^8 + a^2 + a + 1, a^11 + a
```

```
^10 + a^8 + a^6 + a^4 + a^3 + a^2 + 1, a^11 + a^8 + a^7
 + a^6 + a^5 + a^4 + a^3 + a^2 + a, 1)
```

Este vector equivale a la expresión polinómica $c = x^5 + a^{3953}x^4 + a^{1333}x^3 + a^{2604}x^2 + a^{1596}x + a^{760}$. Supongamos que se ha recibido tras la transmisión el vector $y = x^5 + a^{3953}x^4 + a^{671}x^3 + a^{2604}x^2 + a^{1596}x + a^{3699}$.

```
sage: y = x^5 + a^3953*x^4 + a^671*x^3 + a^2604*x^2 + a
    ^1596*x + a^3699; y
> x^5 + (a^11 + a^8 + a^7 + a^6 + a^5 + a^4 + a^3 + a^2 +
    a)*x^4 + (a^11 + a^10 + a^8 + a^6 + a^4 + a^2 + 1)*x^3
    + (a^11 + a^9 + a^8 + a^2 + a + 1)*x^2 + (a^9 + a^6 + a
    ^5 + a^4 + a^3 + 1)*x + a^9 + a^8 + a^6 + a^5 + a + 1
```

Vamos a utilizar el decodificador definido antes.

```
sage: D.decode_to_code(y)
> Peterson-Gorenstein-Zierler algorithm based decoder for [6, 2]
     Skew RS Code on Skew Polynomial Ring in x over Finite Field
     in a of size 2^12 twisted by a |--> a^(2^10)
sage: D.correction_capability()
> 2
sage: D.decode_to_code(y)
DEBUG: s, syndromes vector: [a^10 + a^6 + a^5 + a^3, a^11 + a^10
     + a^7 + a^5 + a^4 + a^3 + a^2 + a, a^11 + a^9 + a^8 + a^5 +
     a^4 + a^2 + a, a^11 + a^10 + a^6 + a^3 + a^2 + a + 1]
  S_t:
  [                   a^11 + a^7 + a^6 + a^4 a^10 + a^7 + a^6 +
      a^5 + a^4 + a^2 + a + 1]
  [           a^11 + a^9 + a^4 + a^2 + a + 1           a^9 + a
      ^8 + a^7 + a^5 + a^2 + a]
  [            a^9 + a^6 + a^4 + a^3 + a + 1         a^11 + a
      ^10 + a^9 + a^4 + a^2 + 1]
  rcef_S_t:
  [                                          1
                                                0]
  [                                          0
                                                1]
  [a^11 + a^10 + a^9 + a^8 + a^7 + a^6 + a^5 + a^4
                           a^10 + a^8 + a^7 + a^2 + 1]
  rho: [a^11 + a^10 + a^9 + a^8 + a^7 + a^6 + a^5 + a^4, a^10 +
      a^8 + a^7 + a^2 + 1, 1]
  rho_N: (0, a^11 + a^8 + a^7 + a^6 + a^4 + a^3 + a^2 + a, a^10
      + a^9 + a^8 + a^7 + a^5 + a^3 + a^2 + 1, 0, a^9 + a^8 + a
```

```
        ^6 + a^3 + a^2 + a + 1, a^10 + a^8 + a^7 + a^4 + a^3 + a +
         1)
17  k: [0, 3]
18  v: 2
19  Note: solve for E, where E*Sigma.transpose() = b_syn
20  Sigma:
21  [                                a             a^8 + a^4 + a
        ^3 + a + 1]
22  [ a^11 + a^9 + a^8 + a^5 + a^4 + a^2 a^11 + a^10 + a^9 + a^7 +
         a^5 + a^4]
23  b_syn: [a^11 + a^7 + a^6 + a^4, a^11 + a^9 + a^4 + a^2 + a +
        1]
24  E: (a^2, a^3)
25  error: a^3*x^3 + a^2
26  m = y - e: x^5 + (a^11 + a^8 + a^7 + a^6 + a^5 + a^4 + a^3 + a
        ^2 + a)*x^4 + (a^11 + a^10 + a^8 + a^6 + a^4 + a^3 + a^2 +
         1)*x^3 + (a^11 + a^9 + a^8 + a^2 + a + 1)*x^2 + (a^9 + a
        ^6 + a^5 + a^4 + a^3 + 1)*x + a^9 + a^8 + a^6 + a^5 + a^2
        + a + 1
27  > (a^9 + a^8 + a^6 + a^5 + a^2 + a + 1, a^9 + a^6 + a^5 + a^4 +
       a^3 + 1, a^11 + a^9 + a^8 + a^2 + a + 1, a^11 + a^10 + a^8 +
        a^6 + a^4 + a^3 + a^2 + 1, a^11 + a^8 + a^7 + a^6 + a^5 + a
       ^4 + a^3 + a^2 + a, 1)
```

Y efectivamente podemos comparar ambos valores para comprobar que son el mismo.

```
1  sage: C.encode(x + a, "SkewCyclicPolynomialEncoder") == D.
       decode_to_code(y)
2  > True
```

Supongamos ahora que al mensaje $x + a$ codificado se le suma un error $e = a^2 + a^{1367} * x^3$, de forma que $y = c + e$.

```
1  sage: y_ = S(c.list()) + a^2 + a^1367*x^3; y_
2  > x^5 + (a^11 + a^8 + a^7 + a^6 + a^5 + a^4 + a^3 + a^2 +
       a)*x^4 + (a^11 + a^7 + a^6 + a^5 + a^3 + a^2)*x^3 + (a
       ^11 + a^9 + a^8 + a^2 + a + 1)*x^2 + (a^9 + a^6 + a^5 +
        a^4 + a^3 + 1)*x + a^9 + a^8 + a^6 + a^5 + a + 1
```

Al decodificar este mensaje nos encontraremos en el caso en el que $\mu \neq v$.

```
1  sage: D.decode_to_code(y_)
2  > (...)
3    rho: [a^10 + a^8 + a^7 + a^5 + a^3 + a^2, 1]
```

```
4    rho_N: (a^5 + a^4 + a^2 + a, a^10 + a^9 + a^6 + a^5 + a^3 + a
         ^2 + 1, a^7 + a^4 + a, a^9, a^11 + a^10 + a^9 + a^8 + a^5
         + a^4 + a^2 + a, a^11 + a^9 + a^8 + a^7 + a^6 + a^5 + a^4
         + a^3 + a^2 + a)
5    Case mu != v
6    (...)
7    error: (a^10 + a^8 + a^7 + a^5 + a^4 + 1)*x^3 + a^2
8  > (a^9 + a^8 + a^6 + a^5 + a^2 + a + 1, a^9 + a^6 + a^5 + a^4 +
       a^3 + 1, a^11 + a^9 + a^8 + a^2 + a + 1, a^11 + a^10 + a^8 +
        a^6 + a^4 + a^3 + a^2 + 1, a^11 + a^8 + a^7 + a^6 + a^5 + a
       ^4 + a^3 + a^2 + a, 1)
```

De nuevo, podemos comprobar que efectivamente el mensaje decodificado es igual al mensaje original.

```
1  sage: C.encode(a + x, "SkewCyclicPolynomialEncoder") == D.
       decode_to_code(y_)
2  > True
```

A

IMPLEMENTACIÓN EN SAGEMATH DEL ALGORITMO DE PETERSON-GORENSTEIN-ZIERLER

Se han desarrollado implementaciones en SageMath del algoritmo de Peterson-Gorenstein-Zierler, tanto en su versión para códigos BCH como para códigos RS sesgados.

Dichas implementaciones aprovechan la estructura de códigos que ya tiene implementada SageMath. Así, para la versión de códigos BCH se ha implementado un decodificador para códigos BCH, `BCHPGZDecoder`, que hereda de la clase `Decoder` de SageMath. Por otro lado, para la versión de códigos cíclicos ha sido necesario implementar primero la clase `SkewCyclicCode`, que hereda de la clase `AbstractLinearCode` de SageMath, y que implementa de forma sencilla los aspectos básicos de códigos cíclicos sesgados, utilizando para ello la implementación existente de anillos de polinomios de Ore de SageMath. Una vez diseñada esta clase que permite trabajar con códigos cíclicos sesgados se ha implementado una clase `SkewRSCode` para manejar códigos RS sesgados y un decodificador para este tipo de códigos, `SkewRSPGZDecoder`.

Su uso es muy sencillo. Con la orden `load()` de SageMath pueden cargarse los archivos proporcionados, que incluyen todas las clases descritas antes.

```
1 sage: load(pgz.sage)
2 sage: load(pgz-sesgados.sage)
```

En este anexo describimos la documentación de las clases y funciones desarrolladas. El código puede encontrarse en

https://github.com/jmml97/tfg/tree/master/code.

A.1 DECODIFICADOR BASADO EN EL ALGORITMO PGZ PARA CÓDIGOS BCH

`class BCHPGZDecoder(self, code)`

Hereda de: `Decoder`

Construye un decodificador para códigos BCH basado en el algoritmo de Peterson-Gorenstein-Zierler para códigos BCH.

ARGUMENTOS

code Código asociado a este decodificador

EJEMPLOS

```
sage: F = GF(2)
sage: C = codes.BCHCode(F, 15, 5, offset=1); C
> [15, 7] BCH Code over GF(2) with designed distance 5
sage: D = BCHPGZDecoder(C); D
> Peterson-Gorenstein-Zierler algorithm based decoder
    for [15, 7] BCH Code over GF(2) with designed
    distance 5
```

decode_to_code(self, word)

Corrige los errores de word y devuelve una palabra código del código asociado a self.

ARGUMENTOS

word Mensaje recibido que se quiere decodificar. Puede representarse en forma vectorial o polinómica.

EJEMPLOS

```
sage: F = GF(2)
sage: C = codes.BCHCode(F, 15, 5, offset=1)
sage: D = BCHPGZDecoder(C)
sage: x = polygen(F)
sage: y = 1 + x +x^5 + x^6 + x^9 + x^10
sage: D.decode_to_code(y)
> (1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0)
sage: y = vector(F, (1, 1, 0, 0, 0, 1, 1, 0, 0, 1,
    1, 0, 0, 0, 0))
sage: D.decode_to_code(y)
> (1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0)
```

correction_capability(self)

Devuelve la capacidad de corrección de errores del decodificador self.

EJEMPLOS

```
sage: F = GF(2)
sage: C = codes.BCHCode(F, 15, 5, offset=1)
sage: D = BCHPGZDecoder(C)
sage: D.correction_capability()
> 2
```

A.2 CLASE PARA CÓDIGOS CÍCLICOS SESGADOS

Esta clase *esqueleto* sirve como modelo para la implementación de los códigos RS sesgados.

`class SkewCyclicCode(self, generator_pol=None)`

Hereda de: `AbstractLinearCode`

Representación de un código cíclico sesgado como un código lineal.

ARGUMENTOS

`generator_pol` Polinomio generador utilizado para construir el código

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a^3903,
    x - a^4047])
sage: C = SkewCyclicCode(g); C
> [6, 2] Skew Cyclic Code on Skew Polynomial Ring in x
    over Finite Field in a of size 2^12 twisted by a
    |--> a^(2^10)
```

`generator_polynomial()`

Devuelve un polinomio generador del código.

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: C.generator_polynomial()
> x^4 + (a^11 + a^10 + a^9 + a^8 + a^7 + a^5 + a^4
    + a^2)*x^3 + (a^4 + a^2 + a)*x^2 + (a^11 + a^10
    + a^9 + a^8 + a^6 + a^3)*x + a^11 + a^8 + a^7 +
    a^6 + a^2 + a
```

`polynomial_ring()`

Devuelve el anillo de polinomios sobre el que está definido el código.

Ejemplos

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: C.polynomial_ring()
> Skew Polynomial Ring in x over Finite Field in a
    of size 2^12 twisted by a |--> a^(2^10)
```

primitive_root()

Devuelve una raíz primitiva del cuerpo sobre el que está definido el código.

Ejemplos

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: C.primitive_root()
> a
```

ring_automorphism()

Devuelve el automorfismo usado en el anillo de polinomios de Ore sobre el que está definido el código.

Ejemplos

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: C.ring_automorphism()
```

```
> Frobenius endomorphism a |--> a^(2^10) on Finite
    Field in a of size 2^12
```

A.3 CODIFICADORES PARA CÓDIGOS CÍCLICOS SESGADOS

Las siguientes clases son codificadores para los códigos cíclicos sesgados. Uno de ellos codifica vetores en palabras código y el otro, polinomios en palabras código. La clase `SkewCyclicVectorEncoder` está indicada como clase codificadora por defecto y por tanto puede utilizarse directamente con el método `encode()` del código.

`class SkewCyclicVectorEncoder(self, code)`

Hereda de: `Encoder`

Codificador que codifica vectores en palabras código. Sea $\mathcal{C}$ un código cíclico sesgado sobre un cuerpo finito $\mathbb{F}$ y g un polinomio generador suyo. Sea $m = (m_1, m_2, \dots, m_k)$ un vector en $\mathbb{F}^k$. Para codificar m este codificador realiza el producto $m\mathcal{M}(g)$, donde $\mathcal{M}(g)$ es una matriz generadora de g.

ARGUMENTOS

code Código asociado a este codificador

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a^3903,
    x - a^4047])
sage: C = SkewCyclicCode(g)
sage: E = SkewCyclicVectorEncoder(C); E
> Vector-style encoder for [6, 2] Skew Cyclic Code on
    Skew Polynomial Ring in x over Finite Field in a of
    size 2^12 twisted by a |--> a^(2^10)
sage: E.encode(vector(F, [a, 1]))
> (a^9 + a^8 + a^6 + a^5 + a^2 + a + 1, a^9 + a^6 + a^5
    + a^4 + a^3 + 1, a^11 + a^9 + a^8 + a^2 + a + 1, a
    ^11 + a^10 + a^8 + a^6 + a^4 + a^3 + a^2 + 1, a^11 +
     a^8 + a^7 + a^6 + a^5 + a^4 + a^3 + a^2 + a, 1)
```

El siguiente ejemplo usa el codificador directamente desde el código, ya que está fijado como codificador por defecto.

```
sage: C.encode(vector(F, [a, 1]))
> (a^9 + a^8 + a^6 + a^5 + a^2 + a + 1, a^9 + a^6 + a^5
   + a^4 + a^3 + 1, a^11 + a^9 + a^8 + a^2 + a + 1, a
   ^11 + a^10 + a^8 + a^6 + a^4 + a^3 + a^2 + 1, a^11 +
    a^8 + a^7 + a^6 + a^5 + a^4 + a^3 + a^2 + a, 1)
```

generator_matrix()

Devuelve una matriz generadora del código sobre el que está construido el codificador.

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: E = SkewCyclicVectorEncoder(C); E
sage: E.generator_matrix()
> [              a^11 + a^8 + a^7 + a^6 + a^2 + a
                a^11 + a^10 + a^9 + a^8 + a^6 + a^3
                                      a^4 + a^2 + a
      a^11 + a^10 + a^9 + a^8 + a^7 + a^5 + a^4 + a^2
                                                 1
                                                 0]
[                                             0
                                    a^11 + a^10 + a
                 a^11 + a^9 + a^8 + a^5 + a^3 + a + 1
                a^9 + a^7 + a^6 + a^4 + a^3 + a^2 + a
       a^11 + a^8 + a^7 + a^6 + a^5 + a^4 + a^3 + a^2
                                                 1]
```

`class SkewCyclicPolynomialEncoder(self, code)`

Hereda de: `Encoder`

Codificador que codifica polinomios en palabras código. Sea $\mathcal{C}$ un código cíclico sesgado sobre un cuerpo finito $\mathbb{F}$ y g un polinomio generador suyo. Dado cualquier polinomio $p \in \mathbb{F}[x]$ calculando el producto $c = p \cdot g$ y devolviendo el vector de coeficientes correspondiente.

ARGUMENTOS

code Código asociado a este codificador

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a^3903,
    x - a^4047])
sage: C = SkewCyclicCode(g)
sage: E = SkewCyclicPolynomialEncoder(C); E
> Polynomial-style encoder for [6, 2] Skew Cyclic Code
    on Skew Polynomial Ring in x over Finite Field in a
    of size 2^12 twisted by a |--> a^(2^10)
```

message_space(self)

Devuelve el espacio de mensajes del codificador, que es el anillo de polinomios sobre el que está definido el código asociado.

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: E = SkewCyclicPolynomialEncoder(C)
sage: E.message_space()
> Skew Polynomial Ring in x over Finite Field in a
    of size 2^12 twisted by a |--> a^(2^10)
```

encode(self, p)

Transforma `p` en un elemento del código asociado a `self`.

ARGUMENTOS

`p` Un polinomio del espacio de mensajes de `self`.

SALIDA

— Una palabra código del código asociado a `self`

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
```

```
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: E = SkewCyclicPolynomialEncoder(C)
sage: E.encode(x + a)
> (a^9 + a^8 + a^6 + a^5 + a^2 + a + 1, a^9 + a^6 +
     a^5 + a^4 + a^3 + 1, a^11 + a^9 + a^8 + a^2 + a
     + 1, a^11 + a^10 + a^8 + a^6 + a^4 + a^3 + a^2
    + 1, a^11 + a^8 + a^7 + a^6 + a^5 + a^4 + a^3 +
    a^2 + a, 1)
```

unencode_nocheck(self, c)

Devuelve el mensaje correspondiente a c. No comprueba si c pertenece al código asociado a `self`.

ARGUMENTOS

c Un vector de la misma longitud que el código asociado a `self`.

SALIDA

— Un polinomio del espacio de mensajes de `self`.

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
sage: C = SkewCyclicCode(g)
sage: E = SkewCyclicPolynomialEncoder(C)
sage: E.unencode_nocheck(vector(F, (a^9 + a^8 + a^6
     + a^5 + a^2 + a + 1, a^9 + a^6 + a^5 + a^4 + a
    ^3 + 1, a^11 + a^9 + a^8 + a^2 + a + 1, a^11 + a
    ^10 + a^8 + a^6 + a^4 + a^3 + a^2 + 1, a^11 + a
    ^8 + a^7 + a^6 + a^5 + a^4 + a^3 + a^2 + a, 1)))
> x + a
```

A.4 CLASE PARA CÓDIGOS RS SESGADOS

`class SkewRSCode(self, generator_pol=None, b_roots=None)`

Hereda de: `SkewCyclicCode`

Representación de un código RS sesgado. Puede construirse de dos formas equivalentes, o bien mediante un polinomio generador o bien mediante las raíces del mismo. En cualquier caso el polinomio generador ha de ser un divisor de $x^n - 1$, donde n es el orden del automorfismo del anillo de polinomios de Ore subyacente.

ARGUMENTOS

`generator_pol` Polinomio generador utilizado para construir el código

`b_roots` β-raíces utilizadas para construir un polinomio generador del código

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a^3903,
    x - a^4047])
sage: C = SkewRSCode(generator_pol=g); C
> [6, 2] Skew RS Code on Skew Polynomial Ring in x over
    Finite Field in a of size 2^12 twisted by a |--> a
    ^(2^10)
sage: C = SkewRSCode(b_roots=[x - a^1023, x - a^3327, x
    - a^3903, x - a^4047]); C
> [6, 2] Skew RS Code on Skew Polynomial Ring in x over
    Finite Field in a of size 2^12 twisted by a |--> a
    ^(2^10)
```

`designed_distance(self)`

Devuelve la distancia mínima prevista del código.

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
```

```
sage: C = SkewRSCode(generator_pol=g)
sage: C.designed_distance()
> 5
```

A.5 DECODIFICADOR BASADO EN EL ALGORITMO PGZ PARA CÓDIGOS RS SESGADOS

`class SkewRSPGZDecoder(self, code)`

Hereda de: `Decoder`

Construye un decodificador para códigos RS sesgados basado en el algoritmo de Peterson-Gorenstein-Zierler para códigos RS sesgados.

ARGUMENTOS

code Código asociado a este decodificador

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a^3903,
    x - a^4047])
sage: C = SkewRSCode(generator_pol=g)
sage: D = SkewRSPGZDecoder(C); D
> Peterson-Gorenstein-Zierler algorithm based decoder
    for [6, 2] Skew RS Code on Skew Polynomial Ring in x
     over Finite Field in a of size 2^12 twisted by a
    |--> a^(2^10)
```

`decode_to_code(self, word)`

Corrige los errores de `word` y devuelve una palabra código del código asociado a `self`.

EJEMPLOS

```
sage: y = x^5 + a^3953*x^4 + a^671*x^3 + a^2604*x^2
    + a^1596*x + a^3699
sage: D.decode_to_code(y)
> (a^9 + a^8 + a^6 + a^5 + a^2 + a + 1, a^9 + a^6 +
    a^5 + a^4 + a^3 + 1, a^11 + a^9 + a^8 + a^2 + a
    + 1, a^11 + a^10 + a^8 + a^6 + a^4 + a^3 + a^2
    + 1, a^11 + a^8 + a^7 + a^6 + a^5 + a^4 + a^3 +
    a^2 + a, 1)
```

```
4 sage: C.encode(x + a, "SkewCyclicPolynomialEncoder
      ") == D.decode_to_code(y)
5 > True
```

A.6 FUNCIONES AUXILIARES

Para el desarrollo de las clases anteriores fueron necesarias varias funciones auxiliares que realizan tareas que se repiten a lo largo de todo el código. Las describimos a continuación.

`_to_complete_list(poly, length)`

Devuelve una lista de longitud exactamente `length` correspondiente a los coeficientes del polinomio `poly`. Si es necesario, se completa con ceros.

ARGUMENTOS

`poly` Un polinomio

`length` Un entero

SALIDA

— La lista de los coeficientes

EJEMPLOS

```
1 sage: F.<a> = GF(2^12)
2 sage: Frob = F.frobenius_endomorphism()
3 sage: sigma = Frob^10
4 sage: S.<x> = SkewPolynomialRing(F, sigma)
5 sage: _to_complete_list(x + a, 15)
6 > [a, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

`left_lcm(pols)`

Calcula el mínimo común múltiplo por la izquierda de todos los polinomios de la lista `pols`.

ARGUMENTOS

`pols` Lista de polinomios

SALIDA

— El mínimo común múltiplo de todos los polinomios en `pols`

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: g = left_lcm([x - a^1023, x - a^3327, x - a
    ^3903, x - a^4047])
> x^4 + (a^11 + a^10 + a^9 + a^8 + a^7 + a^5 + a^4 +
    a^2)*x^3 + (a^4 + a^2 + a)*x^2 + (a^11 + a^10 + a
    ^9 + a^8 + a^6 + a^3)*x + a^11 + a^8 + a^7 + a^6
    + a^2 + a
```

norm(i, gamma, sigma)

Calcula la i-ésima norma de gamma con el automorfismo sigma.

Recordemos que definimos la *norma i-ésima* de un elemento $\gamma \in \mathbb{F}_q$ como $N_i(\gamma) = \sigma(N_{i-1}(\gamma))(\gamma) = \sigma^{i-1}(\gamma) \dots \sigma(\gamma)\gamma$ para $i > 0$ y $N_0(\gamma) = 1$.

ARGUMENTOS

i El orden de la norma

gamma El elemento al que se le quiere calcular la norma

sigma El automorfismo usado para calcular la norma

SALIDA

— La i-ésima norma de gamma con el automorfismo sigma

EJEMPLOS

```
sage: F.<a> = GF(2^12)
sage: Frob = F.frobenius_endomorphism()
sage: sigma = Frob^10
sage: S.<x> = SkewPolynomialRing(F, sigma)
sage: norm(3, F(1 + a), sigma)
> a^11 + a^10 + a^8 + a^3 + a^2 + a
```

B

FUNCIONES EN SAGEMATH USADAS EN LOS EJEMPLOS

En este anexo se describen algunas de las funciones utilizadas durante los ejemplos a lo largo del trabajo. De nuevo, el código puede encontrarse en

https://github.com/jmml97/tfg/tree/master/code.

`generators(poly)`

Devuelve una lista de polinomios generadores de códigos cíclicos de longitud el grado de `poly`.

ARGUMENTOS

`poly` Un polinomio de la forma $x^n - 1$

SALIDA

— La lista de tupla polinomio generador e idempotente generador

`defining_sets(poly)`

Devuelve una lista de tuplas consistentes en un polinomio generador de un código cíclico de longitud el grado de `poly`, un conjunto característico y una raíz primitiva.

ARGUMENTOS

`poly` Un polinomio de la forma $x^n - 1$

SALIDA

— La lista de polinomios generadores

`generator_and_idempotents(poly)`

Devuelve una lista de tuplas consistentes en un polinomio generador de un código cíclico de longitud el grado de `poly` y el idempotente generador correspondiente.

ARGUMENTOS

`poly` Un polinomio de la forma $x^n - 1$

SALIDA

— La lista de tupla polinomio generador e idempotente generador

mult(iterable)

Devuelve el producto de todos los elementos de `iterable`.

ARGUMENTOS

`iterable` Un objeto iterable

SALIDA

— El producto de todos los elementos de `iterable`

powerset(iterable)

Devuelve todas las posibles combinaciones de elementos de `iterable`.

ARGUMENTOS

`iterable` Un objeto iterable

SALIDA

— Todas las posibles combinaciones de elementos de `iterable`

EJEMPLOS

```
1 sage: powerset([1,2,3])
2 > () (1,) (2,) (3,) (1,2) (1,3) (2,3) (1,2,3)
```

BIBLIOGRAFÍA

Shannon, C. E. (1945, 1 de septiembre). *A mathematical theory of cryptography*. Bell Telephone Labs. Recuperado desde https://www.iacr.org/museum/shannon/shannon45.pdf

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379-423. doi:10.1002/j.1538-7305.1948.tb01338.x

Dyson, G. (2015). *La catedral de Turing: los orígenes del universo digital*. OCLC: 904326706. Barcelona: Debate.

The Sage Developers. (2020). SageMath, the Sage Mathematics Software System (Version 9.0).

Cohn, P. M. (1982). *Algebra* (2nd ed.). New York: Wiley.

Cohn, P. M. (1989). *Algebra Vol. 2* (2nd ed., reprint with corr.). OCLC: 832519027. New York: Wiley.

Lidl, R. & Niederreiter, H. (1986). *Introduction to finite fields and their applications*. Cambridge [Cambridgeshire] ; New York: Cambridge University Press.

Huffman, W. C. & Pless, V. (2003). *Fundamentals of error-correcting codes*. doi:10.1017/CBO9780511807077

Podestá, R. (2006). *Introducción a la Teoría de Códigos Autocorrectores*. Universidad Nacional de Córdoba. Recuperado desde https://www.famaf.unc.edu.ar/documents/940/CMat35-3.pdf

Prange, E. (1957). *Cyclic error-correcting codes in two symbols*. Air force Cambridge research center.

Kelbert, M. & Suhov, Y. M. (2013). *Information theory and coding by example*. OCLC: ocn858661041. Cambridge ; New York: Cambridge University Press.

MacWilliams, F. J. & Sloane, N. J. A. (1977). *The theory of error correcting codes*. North-Holland mathematical library ; v. 16. Amsterdam ; New York : New York: North-Holland Pub. Co. ; sole distributors for the U.S.A. y Canada, Elsevier/North-Holland.

Peterson, W. (1960). Encoding and error-correction procedures for the bose-chaudhuri codes. *IEEE Transactions on Information Theory*, *6*(4), 459-470. doi:10.1109/TIT.1960.1057586

Gorenstein, D. & Zierler, N. (1961). A class of error-correcting codes in p^m symbols. *J. SIAM*, *9*, 207-214.

Ore, O. (1933). Theory of Non-Commutative Polynomials. *The Annals of Mathematics*, *34*(3), 480. doi:10.2307/1968173

Jacobson, N. (1996). *Finite-dimensional division algebras over fields*. Berlin ; New York: Springer.

Gómez-Torrecillas, J., Lobillo, F. J. & Navarro, G. (2020). Factoring ore polynomials over fields i: Mathematical algorithms.

Gomez-Torrecillas, J., Lobillo, F. J. & Navarro, G. (2016). A New Perspective of Cyclicity in Convolutional Codes. *IEEE Transactions on Information Theory*, *62*(5), 2702-2706. doi:10.1109/TIT.2016.2538264

Gómez-Torrecillas, J., Lobillo, F. J. & Navarro, G. (2018). Peterson–gorenstein–zierler algorithm for skew RS codes. *Linear and Multilinear Algebra*, *66*(3), 469-487. doi:10.1080/03081087.2017.1301364

Shi, M., Alahmadi, A. & Sole, P. (2017). *Codes and rings: theory and practice*. Pure and applied mathematics. OCLC: ocn974698974. London, United Kingdom ; San Diego, CA, United States: Elsevier/AP, Academic Press, an imprint of Elsevier.

Lam, T. & Leroy, A. (1988). Vandermonde and wronskian matrices over division rings. *Journal of Algebra*, *119*(2), 308-336. doi:10.1016/0021-8693(88)90063-4

AGRADECIMIENTOS

Quiero agradecer a mi tutor, Gabriel Navarro, las pautas, los consejos y la ayuda que me ha proporcionado a lo largo de todo el desarrollo del trabajo.

Además, quiero agradecer a mi madre, Herminia Luque, a mi padre, Francisco Manuel Martín, y a mi hermano, Francisco Javier Martín, todo el cariño y el apoyo recibidos y la confianza depositada en mí durante todos estos años.

Quiero mencionar asimismo a mis amigos y compañeros de clase, a quienes les debo gran parte de los buenos momentos de estos años en Granada. He de hacer una mención especial a Sofía Almeida y a Pablo Baeyens, que han estado siempre ahí, para escucharme y apoyarme, y a quienes les tengo un cariño enorme.

Agradezco también a Ignacio Casares su trabajo de revisión en profundidad del texto en inglés.

Finalmente, he de agradecer igualmente a todos los desarrolladores de SageMath, pues parte de este trabajo depende del que han realizado ellos, así como a todas aquellas personas que dedican sus esfuerzos a promover el acceso libre y gratuito del conocimiento científico.

COLOFÓN

Este trabajo comenzó a escribirse en Granada en octubre de 2019 y fue terminado en Rincón de la Victoria en junio de 2020, mientras el mundo se encontraba inmerso en la pandemia de la covid-19.

Ha sido compuesto utilizando LaTeX con el estilo proporcionado por el paquete `classicthesis`, desarrollado por André Miede e Ivo Pletikosić e inspirado en el del libro de Robert Bringhurst, «*The Elements of Typographic Style*». Puede obtenerse una copia de dicho paquete en

https://bitbucket.org/amiede/classicthesis/.

Las tipografías utilizadas han sido *EBGaramond* para el cuerpo, *Garamond Math* para las matemáticas, *Open Sans* para las leyendas y *Go Mono* para el código.