Universidad de Valladolid
Departamento de Álgebra, Análisis Matemático,
Geometría y Topología

Cálculo de Peso y Distancia Mínima y Descodificación de Códigos no Lineales.

Anastacia Londoño

Directores Prof.Dr. Edgar Martínez-Moro
Prof.Dr. Antonio Campillo

Trabajo Final de Máster
para obtener el grado de M.Sc. en Investigación en Matemáticas
2017

A mis padres, por su amor y apoyo.

Resumen

Un código corrector de errores es un proceso matemático que consiste en expresar una secuencia de elementos sobre un alfabeto añadiendo cierta redundancia con el objetivo de detectar y corregir tantos errores de transmisión como sean posibles. Los códigos más usados por sus propiedades para codificar y descodificar son los que tienen estructura lineal. En este trabajo trataremos códigos no lineales que en algunos casos aumentan en capacidad detectora y correctora código para una longitud establecida. La primera parte de la memoria trata sobre la representación de los códigos no lineales, en el caso lineal basta con presentar una matriz generadora o una matriz de paridad, para códigos no lineales la representación usual es mediante un núcleo y un conjunto de vectores representativos, aunque la idea de matriz de paridad se puede extender a un sistema de paridad. La segunda mitad del trabajo se concentra en presentar algoritmos para el cálculo de la distancia y el peso mínimos de un código no lineal y en generar un algoritmo eficiente para la descodificación desde un punto de vista de un proceso de reducción en bases de Gröbner.

Abstract

An error correcting code is a mathematical process in which one expresses a sequence of elements over an alphabet by adding some redundancy with the aim of detecting and correcting as many transmission errors as possible. The codes most used for their coding and decoding properties are those with a linear structure. In this paper, we will deal with nonlinear codes, which, although it loses those good properties, sometimes they increase in detection and correction capacity for an established length of code. The first part deals with the representation of the code, in the linear case it suffices to present a generating matrix or a parity matrix, for nonlinear codes the usual representation is through a kernel and a set of representative vectors, although the idea of parity matrix can be extended to a parity system. The second half of the document focuses on algorithms for calculating the minimum distance and weight of a nonlinear code and on generating an algorithm for decoding based on the reduction process in a Gröbner basis.

Agradecimientos

Quisiera expresar mis más sinceros agradecimientos a todas aquellas personas que hicieron posible la elaboración de esta monografía.

A los profesores Edgar Martínez-Moro y Antonio Campillo quienes fueron los responsables de mi formación para prepararme y salir adelante con este trabajo.

A la profesora Irene Marquez-Corbella por el apoyo sincero, permanente e incondicional.

Y muy especialmente al profesor Edgar Martínez-Moro (nuevamente) por brindarme la gran oportunidad que representa este trabajo. Espero que todo mi esfuerzo haya recompensado su especial atención.

Tambien agradezco a los miembros del Jurado por la revisión y corrección del texto, y a los responsables del proyecto de becas Iberoamerica+Asia del Banco Santander junto a la Universidad de Valladolid que me apoyaron para completar esta etapa en mi formación académica. Finalmente quiero agradecer a todas aquellas personas que me acompañaron y siempre estuvieron ahí para apoyarme. A todos ustedes, gracias.

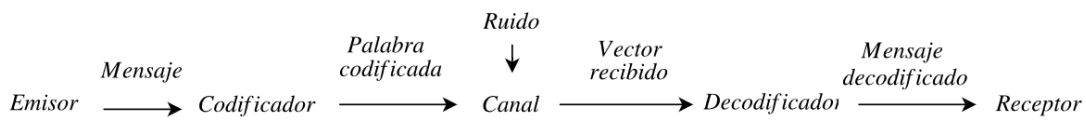
ÍNDICE GENERAL

Introducción

En la vida cotidiana, convivimos con muchos códigos aunque no nos demos cuenta. Ejemplos comunes de estos son el código de barras, el ISBN usado en los libros o el código ASCII usado en las computadoras. Los primeros ejemplos de códigos usados en la práctica son el código Morse, usado en telegrafía desde el siglo XIX, y el sistema Braille.

La teoría de códigos tiene sus origenes a finales de los años 40 con la publicación del artículo "A mathematical theory of communication" de Claude Shannon [14]. Posteriormente, durante la mitad del siglo pasado, la teoría de códigos se ha desarrollado como una disciplina que intersecta las matemáticas y la ingeniería con aplicaciones en casi cualquier área de comunicaciones como transmisiones entre satélites y teléfonos celulares, grabación en discos compactos o almacenamiento de datos.

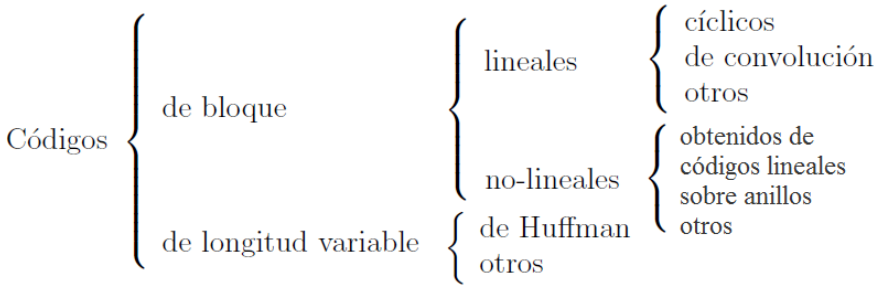
En general, al enviar un mensaje, deseamos que el receptor reciba lo mismo que fue enviado; o en el peor de los casos que sea capaz de notar que hubo errores en la transmisión. De hecho, los códigos correctores de errores, los cuales generan el contexto de este trabajo, darán al receptor la capacidad de como su nombre indica, corregir el mensaje recibido para recuperar el mensaje original. La siguiente figura ilustra este proceso:



Sin embargo un mecanismo tal puede generar muchas preguntas, una de ellas, pertinente para este trabajo es ¿acaso todos los códigos son iguales? Naturalmente cualquier persona con un entendimiento bastante básico obtendría la respuesta estricta, no son iguales, basta con alterar aritméticamente los parámetros que definan a un código para hallar uno diferente, pero nos referimos a una pregunta más profunda y en el afán de exactitud matemática podemos replanterla como ¿acaso todos los códigos tienen la misma estructura?

De nuevo esta pregunta se queda corta en nuestras intenciones, para probarlo puede el lector formurlarla a cualquier estudiante de grado que halla cursado una materia introductoria a la Teoría de Códigos y este le recitará una lista de estructuras de códigos *lineales*.

Haremos un último intento, nuestra reformulación final será más sencilla ¿Acaso todos los códigos son *lineales*? Pues bien, la respuesta a esta pregunta, quizá para sorpresa de nuestro anterior estudiante, es **"no"**. De hecho, la clasificación más básica de los códigos correctores teniendo en cuenta su estructura es



En la práctica, los códigos lineales son los más usados no sólo poseen buenas propiedades generales y algoritmos eficientes de codificación y decodificación, sino además porque pueden ser implementados en computadoras a través de ciertos circuitos lineales llamados shift-registers. Pues bien, uno de los objetivos centrales de la teoría de códigos correctores de errores es construir «bue-

nos» códigos. Esto es, códigos que permitan codificar muchos mensajes, que se puedan transmitir rápida y eficientemente y que detecten y corrijan simultáneamente la mayor cantidad de errores posibles. Desafortunadamente, en el caso de los códigos lineales estas metas son contradictorias entre sí.

Es aquí donde los códigos no lineales presentan su ventaja más significativa. Pues estos permiten mejorar la capacidad de detección y corrección de errores de los códigos lineales de iguales parámetros, haciendolos casos de interés.

En los sistemas informáticos modernos, los códigos correctores de 1 error y códigos de detección de doble error SEC-DED (por sus siglas en inglés), se utilizan como contramedida contra errores suaves para aumentar la fiabilidad del sistema. Estos códigos tienen una distancia de Hamming 4 y son capaces de corregir todos los errores de un solo bit y detectar todos los errores de dos bits.

Sin embargo, en el caso de errores múltiples, la fiabilidad de sistemas que utilizan esquemas de protección de errores basados en códigos lineales puede ser cuestionable. Para cualquier código lineal (n, k) SEC-DED, el número de errores múltiples indetectables es 2^k. Además de esto, un gran número de errores múltiples se corregirán como errores de un solo bit. es evidente entonces que estos códigos pueden no ser suficientes para proporcionar una alta fiabilidad.

La característica común de los códigos lineales SEC-DED es que concentran sus capacidades de detección y corrección de errores en un tipo específico de errores (es decir, errores con pequeñas multiplicidades o que pertenecen al mismo byte). En 2009, Wang, Karpovsky y Kulikowski en [17] propusieron códigos no lineales «robustos» como solución de estas fallas. Estos códigos no lineales están diseñados para proporcionar igual protección contra todos los errores eliminando así posibles áreas débiles en la protección que puedan ser

explotadas por un atacante.

Por otra parte, debido a la falta de linealidad, la representación de códigos no lineales suele ser difícil. Una primera solución sería determinar si tienen otra estructura o no. Por ejemplo, existen códigos binarios no lineales binarios que son imágenes (mediante la aplicación no lineal de Gray) de códigos $\mathbb{Z}_4$-lineales o $\mathbb{Z}_2\mathbb{Z}_4$-lineales y, por lo tanto, también se pueden representar compactamente usando una matriz generadora cuaternaria.

Sin embargo, en general, los códigos no lineales sin ninguna estructura conocida también se pueden representar compactamente. Específicamente, pueden ser vistos como una unión de algunas clases de un subcódigo lineal del código como veremos en el capítulo 1. Esto nos permite representar un código como un conjunto de palabras de código representativas en lugar de como un conjunto con todas las palabras de código. Además, estas palabras de código representativas pueden organizarse como una matriz, llamada sistema de paridad, que es una generalización de la matriz de comprobación de paridad para códigos lineales.

La segunda desventaja de los códigos no lineales es que el cálculo de la distancia mínima no se limita simplemente al cálculo del peso mínimo, ya que en el caso no lineal estos valores pueden ser diferentes.

El objetivo general de este trabajo es presentar nuevos resultados que permitan obtener el peso y la distancia mínima de los códigos no lineales generales así como un método de descodificación para estos, aprovechando un subconjunto lineal del código y extendiendo la idea de la decodificación por conjuntos de prueba. Los resultados presentados en este trabajo se acompañan de un algoritmo respectivo, y se presenta un pequeño análisis de complejidad de estos.

El resto de este texto se organiza como sigue:

- En el capítulo 1, se fija la notación y se desarrollan algunos aspectos básicos de la teoría de códigos. También se presentan los conjuntos de prueba de un código y se muestra un método de descodificación asociado a estos. Al final de este capítulo se introducen los códigos no lineales y se presentan los resultados obtenidos en [16, 19].
- En el capítulo 2, se introducen las bases de Gröbner y se muestra una construcción de un conjunto de prueba de un código lineal binario a partir de la base de Gröbner reducida de un ideal asociado al código.
- En el capítulo 3 se presentan los resultados obtenidos para el cálculo de distancia y peso mínimo de un código no lineal, así como el método de descodificación alcanzado. En la última sección se desarrollan los correspondientes algoritmos y ejemplos y se discute brevemente la complejidad de estos.

1 Códigos lineales y no lineales

El propósito fundamental del capítulo es fijar la notación, presentar algunos aspectos básicos fundamentales de la teoría de códigos y algunos resultados previos de los códigos no lineales. Asumimos que el lector está familiarizado con las principales definiciones y resultados del álgebra lineal y abstracta.

La transmisión de la información digital consiste en enviar mensajes, o cadenas de palabras que son, a su vez, cadenas de elementos de un alfabeto o conjunto de símbolos. Dichos mensajes se envían a través de un canal. La teoría de códigos tiene sus origenes a finales de los años 40 con la publicación del artículo "A mathematical theory of communication" de Claude Shannon [14]. Posteriormente, durante la mitad del siglo pasado, la teoría de códigos se ha desarrollado como una disciplina que interseca las matemáticas y la ingeniería con aplicaciones en casi cualquier área de comunicaciones como transmisiones entre satélites y teléfonos celulares, grabación en discos compactos o almacenamiento de datos. Así mismo, hoy en día constituye una disciplina matemática por si misma capaz de generar sus propios problemas y ayudar a otro tipo de disciplinas matemáticas puras o aplicadas.

En general, al enviar un mensaje, deseamos que el receptor reciba lo mismo que fue enviado; o en el peor de los casos que sea capaz de notar que hubo errores en la transmisión. De hecho, los códigos correctores de errores, los cuales generan el contexto de este trabajo, darán al receptor la capacidad de como su nombre indica, corregir el mensaje recibido para recuperar el mensaje original.

Un canal es cualquier medio de transmisión, como puede ser la red de internet, el servicio de correos, las ondas de radio, etc. En este trabajo sólo nos ocuparemos del estudio de los canales con interferencia o ruido. Además consi-

deraremos sólo canales sin pérdida, es decir, aquellos en los que si se transmite un símbolo se recibe un símbolo, también serán canales sin memoria, lo que implica que la transmisión no depende de los símbolos previamente enviados, y cada símbolo tiene una probabilidad p de ser erróneo, lo que se conoce habitualmente como probabilidad de error del canal. Finalmente, supondremos que el canal es simétrico es decir, la probabilidad de error en la transmisión de un símbolo es independiente del símbolo emitido. A los conjuntos finitos de símbolos los llamaremos **alfabetos** y los denotaremos por $\mathcal{A}$.

Definición 1. Llamaremos **palabra** a una cadena de símbolos de un alfabeto $\mathcal{A}$ y diremos que una palabra tiene **longitud** $n \in \mathbb{N}$ si está compuesta por exactamente n símbolos.

Definición 2. Se dice que un código dado es un **código en bloques** si existe un $n \in \mathbb{N}$, tal que toda palabra del mismo tiene longitud n. En ese caso se dice que la longitud del código es n.

Intuitivamente, codificar mensajes es reescribirlos (en el caso no trivial) en forma diferente, añadiendo la condición de que cada palabra tiene una y sólo una palabra correspondiente en el código, así podemos pensar en este proceso como una función inyectiva que se define como sigue

Definición 3. Sean $\mathcal{A}$ y $\mathcal{F}$ alfabetos y sea $Pal(\mathcal{F})$ el conjunto de todas las palabras de $\mathcal{F}$. Una **codificación del alfabeto** es una aplicación inyectiva

$$c : \mathcal{A} \to Pal(\mathcal{F}).$$

Llamamos **código** a la imagen de C y se llamará indistintamente **palabra** o **vector** a cada uno de sus elementos.

La descodificación es la parte más importante y delicada del proceso. Debemos tener en cuenta que, si en la transmisión no se cometen errores, entonces

la palabra recibida es la misma que la emitida, pero esto sólo ocurre en un canal sin ruido. En este caso el proceso de descodificación es sencillo y consiste simplemente en invertir la codificación. En el caso general de tener un canal con ruido, es necesario crear un sistema que nos permita recuperar la palabra emitida (con una cierta probabilidad) a partir de la palabra recibida.

1.1. Códigos lineales

Para obtener códigos en los que el proceso de codificar y descodificar sea más eficiente, podemos considerar cierta estructura algebraica en ellos, una extensa, eficaz y sencilla familia de estos son los códigos lineales.

Nótese que la definición (3) permite palabras de distina longitud, nosotros trabajaremos en códigos bloque, es decir, con $\mathcal{A}$ un alfabeto finito, si $|\mathcal{A}| = q$ y $\mathcal{C} \in \mathcal{A}^n$ llamamos **código q-ario en bloques** con longitud n. A los largo de este trabajo, el alfabeto considerados será siempre cuerpo finito que denotaremos por $\mathbb{F}_q$, donde $q = p^r$, con $p, r \in \mathbb{N}$ y p primo al cuerpo finito de q elementos, es más, la mayoría de los resultados se referirán al cuerpo finito con dos elementos $\mathbb{F}_2$.

Definición 4. Un **código lineal** de longitud n y rango k es un subespacio lineal $\mathcal{C}$ con dimensión k del espacio vectorial $\mathbb{F}_q^n$. Diremos entonces que $\mathcal{C}$ es un código lineal $[n, k]$ sobre $\mathbb{F}_q$. Llamaremos **tamaño** del código al número de palabras de este, que en este caso es igual a q^k.

Definición 5. El **soporte** de un vector de $\mathbb{F}_q^n$, es el conjunto de coordenadas en el que la palabra no se anula, es decir, $\forall v \in \mathbb{F}_q^n$, $supp(v) = \{i : v_i \neq 0\}$. Dado un código $\mathcal{C} \subset \mathbb{F}_q^n$ decimos que v tiene **soporte minimal** en $\mathcal{C}$ si v es minimal para la relación de inclusión de conjuntos.

Ejemplo 1. [Códigos de repetición] Supongamos que un comando militar quiere enviar los mensajes «Sí» y «No» a un equipo táctico para aprobar o no

una lista de instrucciones previamente compartidas, si se decide codificar dichas palabras simplemente por 1 y 0 respectivamente y se envían a través de un canal con ruido estas pueden sufrir modificaciones, que dada la naturaleza del mensaje pueden ser fatales (Suponga que la instrucción correspondiente al mensaje sea hacer estallar un edificio, el mensaje enviado sea 0 y el mensaje recibido sea 1). En este caso es clara la necesidad de utilizar un código detector y corrector de errores.

Supongamos además, que de alguna manera se sabe que el canal comete a lo más dos errores cada 5 emisiones.Consideremos como alfabeto $\mathbb{F}_2$, usamos entonces un código de 5 dígitos como sigue:

$$\begin{aligned} \text{Sí} &\rightarrow 11111 \\ \text{No} &\rightarrow 00000 \end{aligned}$$

Así el mensaje recibido podrá ser descodificado sin lugar a dudas, puesto que aún cometiendose la mayor tasa de errores (2 en una palabra) el valor más repetido en el bloque bastará para corregir el error. Este tipo de códigos se llaman **códigos de repetición**.

1.1.1. Capacidad de detección y corrección

La capacidad de detección de un código, es simplemente la cantidad máxima (por palabra) de errores que el código puede detectar. Equivalentemente la capacidad de corrección, es la cantidad máxima (por palabra) de errores que el código puede corregir. Las siguientes definiciones nos dan una herramienta fundamental para medir dichos parámetros;

Definición 6. La **distancia de Hamming** entre dos vectores v y w de un código $\mathcal{C}$, que denotaremos como $d_H(v,w)$, es el número de elementos en los que

difieren, es decir:

$$d_H(v,w) = |\{i : 1 \leq i \leq n \text{ y } v_i \neq w_i\}| = |supp(v-w)|. \tag{1.1}$$

Observación 1. La distancia de Hamming $d_H(x,y)$ satisface las siguientes propiedades:

1. (no-negatividad) $d_H(x,y) \geq 0 \ \forall x,y \in \mathbb{F}_q^n$
2. $d_H(x,y) = 0$ sii $x = y$
3. (simetría) $d_H(x,y) = d_H(y,x) \ \forall x,y \in \mathbb{F}_q^n$
4. (desigualdad triangular) $d_H(x,z) \leq d_H(x,y) + d_H(y,z) \ \forall x,y,z \in \mathbb{F}_q^n$

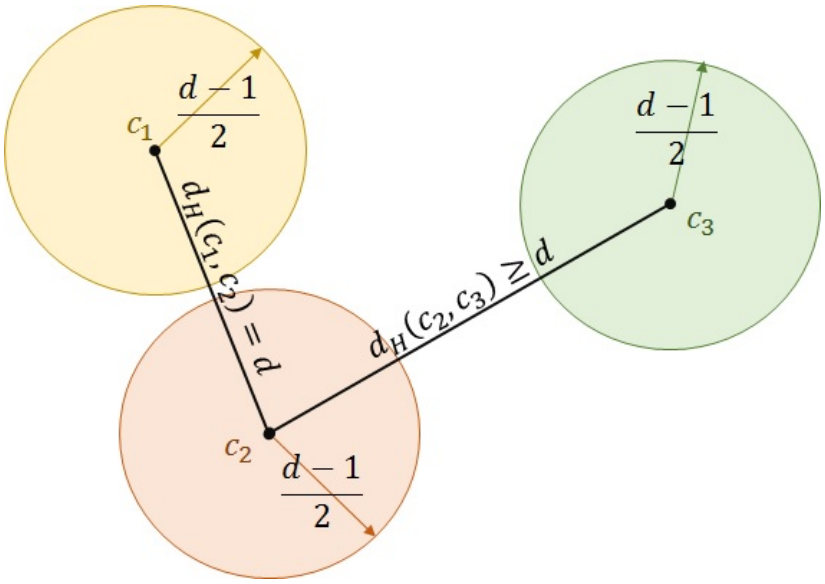
Es decir, la distancia de Hamming es una métrica para $\mathbb{F}_q^n$.

Definición 7. la **distancia mínima** de un código $\mathcal{C}$ - o simplemente distancia de $\mathcal{C}$ - que denotaremos por $d_H(\mathcal{C})$, es la distancia mínima entre dos palabras diferentes de código, es decir

$$d_H(\mathcal{C}) = \min\{d_H(v,w) : v,w \in \mathcal{C}, v \neq w\}. \tag{1.2}$$

De manera intuitiva, a mayor distancia mínima, mayor es la cantidad de errores que el código puede corregir. En efecto, si $d = d_H(\mathcal{C})$ Nótese que las bolas de radio $(d-1)/2$ centradas en las palabras del código son disjuntas, podemos ver esto con claridad en el siguiente ejemplo:

Ejemplo 2. Sea $\mathcal{C}$ un código de distancia mínima $d_H(\mathcal{C}) = d$, y sean $c_1, c_2, c_3 \in C$ tales que $d_H(c_1,c_2) = d$ y $d_H(c_2,c_3) \geq d$. Naturalmente las bolas cerradas centradas en c_1, c_2 y c_3 de radio $\frac{d-1}{2}$ serán disjuntas dos a dos como lo muestra la siguiente figura



Así cualquier palabra recibida, que este contenida en la bola centrada en c_1 (resp. c_2, c_3), es decir, que difiera de c_1 en a lo más $\lfloor \frac{d-1}{2} \rfloor$ términos está más próxima a c_1.

De esta manera, si el número de errores en la palabra recibida no supera $\lfloor (d-1)/2 \rfloor$, entonces la palabra corregida coincide con la realmente enviada. Más adelante, en el **Lema 1** veremos formalmente que el código permite detectar $d-1$ errores y corregir $\lfloor (d-1)/2 \rfloor$.

Antes de desarrollar la idea anterior, permitasenos dar dos definiciones que serán variantes útiles en los cálculos, de las definiciones anteriores.

Definición 8. El **peso de Hamming** de una palabra v de código $\mathcal{C}$, que denotaremos como $w_H(v)$, es el número de sus elementos que son distintos de cero, es decir

$$w_H(v) = |\{i : 1 \leq i \leq n \text{ y } v_i \neq 0\}|. \tag{1.3}$$

Definición 9. El **peso mínimo de un código** $\mathcal{C}$ -o simplemente **peso de** $\mathcal{C}$- que denotaremos por $w_H(\mathcal{C})$, es el peso mínimo entre palabras no nulas de código,

es decir:

$$w_H(\mathcal{C}) = \min\{w_H(v) : v \in C \setminus \{\mathbf{0}_C\}\} \tag{1.4}$$

El siguiente teorema justifica la relación mencionada anteriormente entre la distancia y el peso de Hamming:

Teorema 1. *Sea $\mathcal{C}$ un código lineal bloque de tamaño $n \in \mathbb{N}$ sobre el alfabeto $\mathbb{F}_q$, se verifica que $d_H(\mathcal{C}) = w_H(\mathcal{C})$*

Demostración.
Puesto que el código es lineal, se tiene que la palabra $(0,\ldots,0) \in \mathcal{C}$, además es claro que $w_H(v) = d_H(v,0)$ para todo elemento $v \in \mathcal{C}$, luego

$$d_H(\mathcal{C}) \leq w_H(\mathcal{C}). \tag{1.5}$$

Por otra parte, de nuevo como $\mathcal{C}$ es un código lineal, Nótese que $v - w \in \mathcal{C}$ $\forall v, w \in \mathcal{C}$ y $w_H(v-w) = d_H(v,w)$ $v, w \in \mathcal{C}$ de donde

$$d_H(\mathcal{C}) \geq w_H(\mathcal{C}). \tag{1.6}$$

Así, de **(1.5)** y **(1.6)** se sigue que $d_H(\mathcal{C}) = w_H(\mathcal{C})$. □

Retomemos ahora el tema de la capacidad de detección y correción de un código. Consideremos las bolas cerradas de algún radio con centro en las palabras del código; es decir, para $v \in \mathcal{C} \subset \mathbb{F}^n$ y $r \in \mathbb{N}$, (nótese que basta con tomar radios naturales porque la distancia de Hamming es discreta),

$$B_r(v) = \{w \in \mathbb{F}^n : d_H(v,w) \leq r\}. \tag{1.7}$$

Es importante tomar en cuenta que si se quiere construir un esquema de decisión, estas bolas deben ser disjuntas dos a dos. El siguiente lema dará un límite de radio para que esto ocurra, límite que se intuía de antemano.

Lema 1. *Sea $\mathcal{C}$ un código en $\mathbb{F}^n$ tal que $d_H(\mathcal{C}) = d$ y sea $r \in \mathbb{N}$. Tomando $t = \lfloor \frac{d-1}{2} \rfloor$, las bolas en el conjunto $\{B_r(v) : v \in C\}$ son disjuntas dos a dos sí, y sólo si, $r \leq t$.*

Demostración.
($\Rightarrow$) Supongamos que $d > r > t$, esto implica que $2 \cdot r > d - 1$, es decir, que $r + 1 > d - r$, o de otra forma, $r \geq d - r$.
Ahora bien, Sean $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_n)$ en C tales que $d_H(x, y) = d$, con $x_{i_j} \neq y_{i_j}$ para $j \in \{1, \ldots, d\}$. Consideremos un $z = (z_1, \ldots, z_n)$ tal que $z_{i_j} = x_{i_j}$ para $j \in \{1, \ldots, r\}$, $z_{i_j} = y_{i_j}$ para $j \in \{r + 1, \ldots, d\}$ y $z_k = x_k = y_k$ para todo $k \neq i_j$. Entonces por construcción, $d_H(y, z) = r$ y $d_H(x, z) = d - r$, de esta manera, se sigue que $z \in B_r(x) \cap B_r(y)$, lo cual es una *contradicción* a la hipótesis. Por lo tanto $r \leq t$.

($\Leftarrow$)Recíprocamente, sean de nuevo x e y palabras de $\mathcal{C}$ tales que $d_H(x, y) = d$. Dado $r \leq t$, supongamos que existe una palabra $z \in B_r(x) \cap B_r(y)$. Como la distancia es una métrica, se cumple que

$$d_H(x, y) \leq d_H(x, z) + d_H(z, y). \tag{1.8}$$

Pero como $z \in B_r(x) \cap B_r(y)$

$$d = d_H(x, y) \leq d_H(x, z) + d_H(z, y) \leq \lfloor \frac{d-1}{2} \rfloor + \lfloor \frac{d-1}{2} \rfloor \leq d - 1 < d \tag{1.9}$$

lo cual es una *contradicción*. Por lo tanto, $B_r(x) \cap B_r(y) = \emptyset$. □

De este lema se desprende la siguiene afirmación.

Observación 2. Para un código $\mathcal{C} \subset \mathbb{F}_q^n$ tal que $d_H(\mathcal{C}) = d$, se tiene que si $r \leq \lfloor \frac{d-1}{2} \rfloor$, el conjunto de bolas cerradas $\{B_r(c) : c \in C\}$ induce el siguiente esquema de decisión
$f : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$ tal que para cada $u \in \mathbb{F}_q^k$, $f(u) = c$ con $c \in C$, si, y sólo si, $d_H(u, c) \leq r$.

En otras palabras, podemos completar la idea del planteamiento primitivo con el siguiente teorema:

Teorema 2. *Dado $\mathcal{C}$ un código (no necesariamente lineal), con $d = d_H(\mathcal{C})$,*

1. *$\mathcal{C}$ puede detectar a lo sumo $d-1$ errores,*
2. *$\mathcal{C}$ puede corregir a lo sumo $\lfloor \frac{d-1}{2} \rfloor = t$ errores.*

Demostración.

1. Sea $c \in \mathcal{C}$ la palabra enviada y v la palabra recibida. Si v provine de haber cometido menos de $d-1$ errores al transmitir c entonces $d_H(c,v) \leq d_H(\mathcal{C})-1$ y por tanto v no es una de las palabras de $\mathcal{C}$.
2. De nuevo, sea $c \in \mathcal{C}$ la palabra enviada y v la recibida. Si se han cometido a lo más t errores, entonces $d_H(c,v) \leq t$ y por lo tanto, $v \in B_t(c)$. Así la palabra recibida se corregirá correctamente.

□

Observación 3. Sea $\mathcal{C}$ un $[n,k]$código y $d = d_H(\mathcal{C})$,

1. llamaremos capacidad de corrección de $\mathcal{C}$ al valor $\lfloor \frac{d-1}{2} \rfloor$,
2. describiremos a $\mathcal{C}$ como un $[n,k,d]$-código.

1.1.2. Matriz generadora y matriz de paridad

Las dos maneras más comunes de presentar un código lineal son mediante su matriz generadora o bien su matriz de paridad. Una **matriz generadora** de un código lineal $[n,k]$ es cualquier matriz G de tamaño $k \times n$ cuyas filas formen una base para $\mathcal{C}$. Nótese que la matriz generadora de un código lineal no es única.

Dado que existen k filas linealmente independientes en G, se tiene que el rango de G $rk(G) = k$ luego existe al menos un conjunto de k columnas independientes de una matriz generadora G. Cada uno de estos conjuntos de k coordenadas forma un grupo de información de $\mathcal{C}$ (Nótese que en general no existe un único grupo de información). Fijado un grupo de información las $r = n-k$ coordenadas restantes forman los términos de redundancia.

Si el grupo de información está formado por las primeras k columnas el código tiene una única matriz generadora de la forma $[I_k|A]$ donde I_k es la matriz identidad de tamaño k x k, a tal matriz la llamaremos matriz generadora en su forma estándar. Nótese que a partir de cualquier código lineal $\mathcal{C}$ mediante una permutación de las coordenadas, podemos obtener un código con matriz generadora en su forma estándar.
Como un [n,k] código lineal $\mathcal{C}$ es un subespacio de un espacio vectorial, entonces $\mathcal{C}$ es el núcleo de una transformación lineal. En efecto, tomemos k vectores linealmente independientes $\{c_1, c_2, \ldots, c_k\}$ en $\mathcal{C}$ y extendamos ese conjunto a una base para $\mathbb{F}_q^n$

$$\{c_1, c_2, \ldots, c_k, v_{k+1}, \ldots, v_n\}, \tag{1.10}$$

luego para cada vector $a \in \mathbb{F}_q^n$ exiten $\alpha_1, \ldots, \alpha_n$ tales que

$$a = \sum_{i=1}^{k} \alpha_i c_i + \sum_{i=k+1}^{n} \alpha_i v_i. \tag{1.11}$$

Así, podemos construir dicha transformación como sigue, $T : \mathbb{F}_q^k \to \mathbb{F}_q^n$ dada por

$$T(a) = \sum_{i=k+1}^{n} \alpha_i \tag{1.12}$$

Luego, existe una matriz H (correspondiente a T) de tamaño $(n-k)$ x n a la que llamaremos **matriz de paridad** de $\mathcal{C}$ definida por:

$$\mathcal{C} = \left\{ x \in \mathbb{F}_q^n : H \cdot x^T = 0 \right\} \tag{1.13}$$

Nótese que las filas de H son linealmente independientes puesto que $\mathcal{C} = ker(T)$, así, $rk(H) = n - k$. En general, la matriz de paridad de $\mathcal{C}$ no es única. El siguiente teorema, muestra como calcular una de ellas a partir de una matriz generadora.

Teorema 3. *Si $G = [I_k|A]$ es una matriz generadora para un código lineal $\mathcal{C}$ que*

admite una forma estándar, entonces $H = [-A^T | I_{n-k}]$ *es una matriz de paridad para* $\mathcal{C}$.

Demostración. Consideremos la transformación lineal $t : \mathbb{F}_q^n \to \mathbb{F}_q^{n-k}$ dado por $t(x) = (H \cdot x^T)^T$. Puesto que $H \cdot G^T = -A^T + A^T = 0$, Se sigue que $\mathcal{C}$ está contenido en el núcleo de t. Pero como H tiene rango $n-k$, la dimensión del núcleo de dicha transformación es k que coincide con la dimensión de $\mathcal{C}$. Luego $ker(t) = \mathcal{C}$. i.e. H es una matriz de paridad de $\mathcal{C}$. □

Más aún, Nótesese que las filas de una matriz de paridad H de un código $\mathcal{C}$ de parámetros $[n,k]$ son linealmente independientes; por lo tanto, H es la matriz generadora de algún código, al que llamaremos el dual de $\mathcal{C}$ y denotaremos por $\mathcal{C}^\perp$. Nótese que $\mathcal{C}^\perp$ es un código $[n, n-k]$.
Se quiere estudiar ahora la capacidad de corrección de los códigos lineales, para ello debemos hallar una manera de encontrar su peso mínimo, una opción (muy poco eficiente) sería examinar todos los vectores no nulos del mismo. El siguiente teorema, y más aún su corolario muestra un mecanismo ligeramente más eficiente.

Teorema 4. *Sea* $\mathcal{C}$ *un código lineal con matriz de paridad* H. *Si el vector* c *está en* $\mathcal{C}$, *entonces las columnas de* H *que correspondan a las coordenadas no nulas de* c *son linealmente dependientes. En el caso binario se cumple el recíproco.*

Demostración.
($\Rightarrow$) Sea $\mathcal{C}$ un código lineal con matriz de paridad H y $c = (c_1, \ldots, c_n) \in \mathcal{C}$, por definición se tiene que

$$H \cdot c^T = 0, \tag{1.14}$$

de donde,

$$\begin{aligned} h_{11}c_1 + \cdots + h_{1n}c_n &= 0 \\ \vdots &= \vdots \\ h_{(n-k)1}c_1 + \cdots + h_{(n-k)n}c_n &= 0 \end{aligned} \tag{1.15}$$

DeNótesemos $c_{i_1}, \ldots, c_{i_r}$, con $i_r \leq n$; a las entradas no nulas de c. Luego, **(1.15)** es equivalente a

$$\begin{aligned} h_{1i_1}c_{i_1} + \cdots + h_{1i_r}c_{i_r} &= 0 \\ \vdots &= \vdots \\ h_{(n-k)i_1}c_{i_1} + \cdots + h_{(n-k)i_r}c_{i_r} &= 0 \end{aligned} \tag{1.16}$$

Es decir, las columnas de H que corresponden a las coordenadas no nulas de c son linealmente dependientes.
El recíproco es trivial en el caso binario. □

Corolario 1. *un $[n,k]$–código lineal $\mathcal{C}$, tiene peso mínimo d si, y sólo si, su matriz de paridad tiene un conjunto de d columnas linealmente dependientes, pero cualquier conjunto de $d-1$ columnas es linealmente independientes.*

Observación 4. Nótese que en el caso lineal del cálculo del peso mínimo es equivalente a resolver el problema de la descodificación completa de un código lineal general, que es un problema NP-completo [2]. Por lo tanto no se puede esperar conseguir un algritmo eficiente para su cálculo.

1.2. Conjuntos de prueba

Podemos pensar de manera intuitiva en un primer método de descodificación por distancia mínima, es decir, recibido un vector, este será descodificado por aquella palabra del código que tenga la menor distancia de Hamming al

vector recibido. De ahora en adelante, durante esta sección consideraremos solo el caso en el que $\mathcal{C}$ es un código lineal binario, es decir cuando $\mathcal{C} \subset \mathbb{F}_2^n$ para algún $n \in \mathbb{N}$. La descodificación por distancia mínima se puede formular como sigue:

Dado un código lineal binario $\mathcal{C} \subset \mathbb{F}_2^n$ con $n \in \mathbb{N}$ el objetivo es construir una aplicación $f : \mathbb{F}_2^n \to \mathcal{C}$ tal que $\forall x \in \mathbb{F}_2^n$

$$d_H(x, f(x)) = d_H(x, \mathcal{C}) \tag{1.17}$$

donde $d_H(x, \mathcal{C}) = min\{d_H(x, c) : c \in \mathcal{C}\}$.

Si para algún vector $x \in \mathbb{F}_2^n$ la condición se satisface para más de una palabra del código, el valor de $f(x)$ se tomará de manera arbitraria de entre ellas. Si llamamos **síndrome de** x (respecto de H) al vector $S(x) = Hx^T \in \mathbb{F}_2^{n-k}$ donde H es una matriz de paridad de $\mathcal{C}$, entonces tendremos que $Hx^T = 0 \Leftrightarrow x \in \mathcal{C}$ y si $y = c + e$ es un vector recibido con $c \in \mathcal{C}$, $S(c + e) = H(c + e)^T = \mathbf{0} + He^T$.

La función **(1.17)** da lugar a la definición de las regiones de Voronoi de las palabras de un código dada a continuación,

Definición 10. La **región de Voronoi** de una palabra c de un código lineal $\mathcal{C}$, denotada por $D(c)$ está definida como

$$D(c) := \{x \in \mathbb{F}_2^n : d_H(x, c) \leq d_H(x, c'),\ \forall c' \in \mathcal{C}\}. \tag{1.18}$$

Observación 5. El conjunto de regiones de Voronoi de todas las palabras de $\mathcal{C}$ cubre todo el espacio $\mathbb{F}_2^n$. Sin embargo, algunos vectores de $\mathbb{F}_2^n$ pueden estar contenidos en más de una región.

De la definición se sigue una propiedad que nos proporciona una primera herramienta de comparación entre las palabras de un código.

Lema 2. *Sean $c, c' \in \mathcal{C}$ y $x \in D(c)$, entonces $x + c' \in D(c + c')$.*

Demostración.
Como $x \in D(c)$, se tiene que $d_H(x,c) \leq d_H(x,\widehat{c})$ para todo $\widehat{c} \in \mathcal{C}$. Ahora bien, podemos trasladar linealmente estas distancias, de forma que $d_H(x+c', c+c') \leq d_H(x+c', \widehat{c}+c')$ para todo $\widehat{c}+c' \in \mathcal{C}$. Además como el código es lineal y binario se tiene que $\widehat{c}+c' = p \in \mathcal{C}$ y variando $\widehat{c} \in \mathcal{C}$ se describen todas las palabras de $\mathcal{C}$. Así, $d_H(x+c', c+c') \leq d_H(x+c', \widehat{c}+c')$ para todo $p \in \mathcal{C}$. Es decir, $x+c' \in D(c+c')$. □

Ahora bien, nótese que la descodificación por distancia mínima, compara el peso de tantas palabras como haya en el código, y aunque el **Teorema (1)** nos proporciona una potente herramienta para el cálculo de distancias mínimas, pues reduce el proceso de comparar todas las distancias dos a dos palabras del código a sólo calcular el peso de cada una de ellas, para un efecto práctico en que el código es suficientemente grande, rápidamente observamos que el proceso no será eficiente.
Con esto en mente, en [1] A. Barg presenta un método que consiste en comparar sólo un conjunto preseleccionado de palabras $\mathcal{T}_\mathcal{C} \subset \mathcal{C}$, tal que cada vector $y \in \mathbb{F}_2^n$ del código, bien está en $D(0)$ o existe $z \in \mathcal{T}_\mathcal{C}$ tal que

$$w_H(y+z) \leq w_H(y) \tag{1.19}$$

Al conjunto $\mathcal{T}_\mathcal{C}$ lo llamaremos **conjunto de prueba de** $\mathcal{C}$ Nótese que $D(0)$ es el conjunto de todos los representantes de clases de $\mathbb{F}_2^n/\mathcal{C}$ de peso mínimo (por el **lema (2)**).
La descodificación por descenso gradiente (o simplemente descodificación por gradiente) consiste en aprovechar las características de este conjunto, que sugiere que la descodificación puede lograrse inspeccionando recursivamente el conjunto de prueba a partir de la existencia de un vector $z \in \mathcal{T}_\mathcal{C}$ que cumple la propiedad **(1.19)** y sustrayéndolo del vector recibido.

Algoritmo 1 Descodificación por gradiente

Entrada: $\mathcal{C}$-Código lineal, $\mathcal{T}_{\mathcal{C}}$-Conjunto de Prueba para $\mathcal{C}$ e $y = c' + e \in \mathbb{F}_2^n \backslash \mathcal{C}$ -vector recibido con $c' \in \mathcal{C}$

Salida: $c \in \mathcal{C}$-La palabra del código con más probabilidad de ser la palabra enviada

$c \leftarrow 0$

mientras $y \notin D(0)$ **hacer**

Tomar un $z \in \mathcal{T}_{\mathcal{C}}$ tal que $w_H(y+z) < w_H(y)$

$c \leftarrow c + z$

$y \leftarrow y + z$

fin mientras

devolver c=y;

El siguiente teorema, demuesta que este algoritmo en efecto proporciona la palabra del código con mayor probabilidad de haber sido enviada.

Teorema 5. *Para cualquier conjunto de vectores que satisfagan la condición de 1.19 el algoritmo anterior realiza una descodificación completa por mínima distancia. Además este algoritmo tiene una complejidad de* $\mathcal{O}(n^2|\mathcal{T}_{\mathcal{C}}|)$.

Demostración.
Sea $y \in \mathbb{F}_2^n$ tal que $y \notin D(0)$, y sea $z_i \in \mathcal{T}_{\mathcal{C}}$ el vector del conjunto de prueba que le es añadido a y en el paso i-ésimo. Suponga que el algoritmo se detiene tras m pasos, entonces se ha encontrado un vector,

$$e = y + \sum_{i=1}^{m} z_i \tag{1.20}$$

tal que $e \in D(0)$ Luego, por el lema 2 $y \in D(\sum_{i=1}^{m} z_i)$.

Nótese que este algoritmo siempre para, puesto que en cada paso, se reduce y a un vector de menor peso, y el peso de todo vector está acotado inferiormente

por 0. Es decir, el peso de y no se puede reducir infinitamente (ya que además el peso de Hamming es discreto). La demostración del orden de complejidad de este método se puede encontrar en [1].

□

Para ultimar esta idea nos hace falta ver si existe una forma práctica de construir un conjunto de prueba. En [1] Barg demuestra que el conjunto de palabras de soporte minimal de un código lineal binario $\mathcal{C}$ es un conjunto de prueba, pero de nuevo este proceso requiere un cálculo costoso para códigos lineales en general, por ejemplo el abordado en [12]. Es evidente que ninguno de estos cálculos puede ser efectivo, pues el problema de descodificar por mínima distancia un código lineal general es un problema duro [2], observación que se utiliza en la propuesta de algunos criptosistemas basados en códigos correctores. En el capítulo 2 veremos cómo construir un conjunto de prueba más pequeño que el conjunto de palabras de soporte minimal a partir de un ideal asociado al código y el proceso de reducción asociado a base de Gröbner reducida del ideal.

1.3. Códigos no lineales

En esta sección describiremos los resultados obtenidos por Zeng F. en sus trabajos [16, 19] sobre códigos no lineales binarios.

1.3.1. Representación de códigos no lineales binarios

Un **código no lineal** binario $\mathcal{C}$ no es más que un subconjunto de $\mathbb{F}_2$ sin una estructura algebraica específica, lo que genera un costo mayor en el almacenamiento. Sin embargo, esa misma característica, permite que algunos códigos no lineales tengan más palabras que cualquier código lineal con los mismos parámetros. Lo que los hace un caso de estudio interesante.

En particular, estos códigos, se pueden almacenar como la unión de clases de un código lineal que llamaremos el **Kernel de** $\mathcal{C}$ que se define como

$$K_{\mathcal{C}} = \{x \in \mathcal{C} : x + \mathcal{C} = \mathcal{C}\}. \tag{1.21}$$

Siempre podemos considerar $0_{\mathbb{F}_2} \in K_{\mathcal{C}}$ módulo una traslación, en efecto, si $0_{\mathbb{F}_2} \notin K_{\mathcal{C}}$ podemos tomar $a \in K_{\mathcal{C}}$ y considerar el código lineal $K_{\mathcal{C}} + a = \{x + a : x \in K_{\mathcal{C}}\}$ que claramente tiene los mismos parámetros (número de palabras, distancia entre ellas y distancia mínima). De esta manera, asumiremos siempre que $K_{\mathcal{C}}$ contiene al vector nulo y por lo tanto $K_{\mathcal{C}}$ es un código lineal. La representación del código no lineal viene dada entonces por una expresión de la forma

$$\mathcal{C} = \bigcup_{i=0}^{t} (K_{\mathcal{C}} + v_i) \tag{1.22}$$

donde los v_i son algunos representantes de las clases de $\mathbb{F}_2^n / K_{\mathcal{C}}$ y $v_0 = 0$. Nótese además que t es una medida de linealidad del código, pues si $t = 0$ el código es lineal y $\mathcal{C} = K_{\mathcal{C}}$ mientras que si $t = |\mathcal{C}|$ entonces la representación es una lista exhaustiva del código y $K_{\mathcal{C}} = \{0\}$.

Ejemplo 3 (Código no lineal)**.** Sea $K_{\mathcal{C}}$ el código lineal binario de longitud $n = 31$, dimensión $k = 5$ y distancia mínima $d = 16$ construido como el dual del código binario de Hamming de longitud $n = 31$ y sean,

$$\begin{aligned}
v_0 &= (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\
v_1 &= (0,0,1,0,0,0,1,1,1,0,0,1,1,0,1,0,0,1,1,1,1,0,0,0,1,0,1,1,1,1,0) \\
v_2 &= (0,1,0,1,1,0,1,0,1,0,1,1,1,1,0,0,1,0,1,1,1,0,1,0,0,1,1,1,1,0,1) \\
v_3 &= (0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,1,1,0,1,0,0,0,1,1,1,1,0,1,0,1,1)
\end{aligned}$$

La unión $\mathcal{C}_{31} = \cup_{i=0}^{3} (K_{\mathcal{C}} + v_i)$ genera un código no lineal de longitud $n = 31$. Usando fuerza bruta, se calculó que $w_H(\mathcal{C}_{31}) = 10$ y $d_H(\mathcal{C}_{31}) = 8$ en $1{,}8 \times 10^{-4}$ y

$8{,}4 \times 10^{-3}$ segundos respectivamente.

La idea de matriz de paridad de un código lineal se extiende en códigos no lineales a un sistema de paridad como lo presenta Westerbäck en [18] generalizando la idea en [9], si consideramos el código no lineal $\mathcal{C} = \bigcup_{i=0}^{t} (K_{\mathcal{C}} + v_i)$ y H la matriz de paridad del código lineal $K_{\mathcal{C}}$ llamamos **sistema de paridad de** $\mathcal{C}$ a la matriz concatenada $(H|S)$ donde S es la matriz cuyas columnas vienen dadas por $col(S) = \left\{Hv_1^T, Hv_2^T, \ldots, Hv_t^T\right\}$.

Las clases de K obtenidas de esta forma de distintas columnas de la matriz S son distintas y por lo tanto disjuntas. Así,

$$c \in \mathcal{C} \Leftrightarrow Hc^T = 0^T \text{ o } Hc^T \in col(S). \tag{1.23}$$

De esta manera, así como la matriz de paridad es útil para la corrección de errores en códigos lineales, el sistema de paridad es útil para el mismo propósito en códigos no lineales. En efecto, para corregir un error en una palabra recibida v, considere el conjunto $Hv^T + col(S)$, una de las columnas debe ser igual a una de las columnas de la matriz H, si es igual a la columna i–ésima, entonces el error está en la posición i.

Ejemplo 4 (Sistema de paridad)**.** Veamos el sistema de paridad para un código no lineal, ya que necesitaremos la matriz de paridad del núcleo y puesto que el núcleo de $\mathcal{C}_{31}$ del ejemplo **(3)** tiene una matriz de paridad de tamaño 26×31, vamos a construir primero un ejemplo de código no lineal más pequeño. Consideremos el código lineal binario $K_{\mathcal{C}}$ generado a partir de la matriz,

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

y los vectores

$$\begin{aligned} v_0 &= (0,0,0,0,0,0,0) \\ v_1 &= (0,1,0,0,1,1,0) \\ v_2 &= (1,0,1,1,0,0,0). \end{aligned}$$

Luego $\mathcal{C} = \cup_{i=0}^{2}(K_{\mathcal{C}} + v_i)$. La matriz de paridad de $K_{\mathcal{C}}$ viene dada por

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Mientras que

$$S = [Hv_1^T, Hv_2^T] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Así, el sistema de paridad de $\mathcal{C}$ viene dado por

$$[H|S] = \left[\begin{array}{ccccccc|cc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{array}\right].$$

1.3.2. Cálculo de distancia y peso mínimo

Dado un código lineal K, y un vector $v \in \mathbb{F}_2^n \backslash K$ llamaremos a $K_v = K \cup (K+v)$ una **clase extendida** de K. Como K es lineal, K_v tambien lo será y $K_v = \langle K, v \rangle$ porque estamos trabajando en el cuerpo $\mathbb{F}_2$. Las siguientes dos proposiciones nos permitirán calcular el peso y distancia mínima de un código binario no

lineal a partir de su descomposición como unión de clases de su núcleo.

Proposición 1. *Sea $\mathcal{C}$ un código binario no lineal dado por $\mathcal{C} = \bigcup_{i=0}^{t}(K_{\mathcal{C}} + v_i)$, el peso mínimo de $\mathcal{C}$ viene dado por $w_H(\mathcal{C}) = \min\left\{w_H(K_{v_i}) : i \in \{0,1,\ldots,t\}\right\}$*

Demostración.
Nótese que los códigos K_{v_i} contienen exactamente todas las palabras de $K_{\mathcal{C}}$ y $K_{\mathcal{C}} + v_i$ para cada $i \in \{1,\ldots,t\}$. Por lo tanto, $w_H(\mathcal{C})$ se puede obtener buscando el mínimo entre todos los pesos mínimos de los códigos lineales K_{v_i}. □

Proposición 2. *Sea $\mathcal{C}$ un código binario no lineal dado por $\mathcal{C} = \bigcup_{i=0}^{t}(K_{\mathcal{C}} + v_i)$, la distancia mínima de $\mathcal{C}$ viene dada por*

$$d_H(\mathcal{C}) = \min\left\{w_H(K_{v_i+v_j}) : i \in \{0,1,\ldots,t-1\}, j \in \{i+1,\ldots,t\}\right\}. \tag{1.24}$$

Demostración.
En este caso tenemos que comparar la distancia entre cualquier par de palabras diferentes c_1 y c_2 en código, para esto consideremos 3 casos.

Caso 1: ($c_1, c_2 \in K_{\mathcal{C}}$) Como $K_{\mathcal{C}}$ es un código lineal, $d_{\mathcal{C}}(c_1,c_2) = d_H(c_1,c_2) = w_H(c_1+c_2) = w_H(k)$ para algún $k \in (K_{\mathcal{C}})$.

Caso 2: ($c_1 \in K_{\mathcal{C}}$ y $c_2 \in K_{\mathcal{C}} + v_i$) En este caso, $d_{\mathcal{C}}(c_1,c_2) = w_H(c_1+c_2) = w_H(k+v_i)$ para algún $k \in (K_{\mathcal{C}})$.

Caso 3: ($c_1 \in K_{\mathcal{C}} + v_i$ y $c_2 \in K_{\mathcal{C}} + v_j$ con $i \neq j$) En este caso, $d_{\mathcal{C}}(c_1,c_2) = w_H(k)$ para algún $k \in (K_{\mathcal{C}})$, pero $c_1 \in K_{\mathcal{C}} + v_1$ y $c_2 \in K_{\mathcal{C}} + v_2$ entonces $d_{\mathcal{C}}(c_1,c_2) = w_H(k+v_i+v_j)$. De donde tenemos que

$$d_H(\mathcal{C}) \geq \min\left\{w_H(K_{v_i+v_j}) : i \in \{0,1,\ldots,t-1\}, j \in \{i+1,\ldots,t\}\right\}. \tag{1.25}$$

La desigualdad contraria se sigue del hecho de que $K_{v_i+v_j} \subset \mathcal{C}$ para cualquier elección $i \in \{0,1,\ldots,t-1\}$, $j \in \{i+1,\ldots,t\}$. □

Es evidente que el costo computacional de estos cálculos depende de la cantidad de clases de $K_{\mathcal{C}}$ en la descomposición de $\mathcal{C}$. En efecto para el cálculo del peso mínimo se requiere obtener el peso mínimo de t códigos lineales, mientras que para la distancia mínima se requiere calcular el peso mínimo de $\binom{t+1}{2}$ códigos lineales.
En [19] se generan, a partir de estos resultados y del algoritmo de Brower-Zimmermann para el cálculo de peso mínimo de códigos lineales, los siguientes algoritmos.

Algoritmo 2 Calcular el peso mínimo de un código no lineal

Entrada: $K_{\mathcal{C}}$-Kernel de $\mathcal{C}$ y $L = \{v_1, \ldots, v_t\}$-Clases de $K_{\mathcal{C}}$
Salida: $w_H(\mathcal{C})$-Peso mínimo de $\mathcal{C}$

```
w_H(C) ← longitud(C)
para i ∈ [1,...,t] hacer
    K_{v_i} ← ⟨K_C, v_i⟩
    w_H(C) ← mín(w_H(K_{v_i}, w_H(C)))
fin para
```

El costo computacional del algoritmo anterior depende directamente del costo de calcular el peso mínimo de un código lineal binario $\mathcal{B}$ de parámetros $[n,k,d]$. Con el objetivo de presentar un costo computacional de sus algoritmos en [19] enuncian un algoritmmo que calcula dicho peso, y demuestran que el costo computacional del mismo es

$$(n-k)\lceil n/k \rceil \sum_{r=1}^{\widehat{r}} \binom{k}{r} \tag{1.26}$$

donde $\widehat{r}$ es el menor entero tal que

$$\lfloor n/k(\widehat{r}+1) + \text{máx}(0, \widehat{r}+1-(k-n \mod (k))) \geq w_H(\mathcal{B})$$

La demostración se puede encontrar en [19, Proposition 2.7.5.]. Ahora podemos

ver el orden de complejidad del algoritmo **(1.3.2)**.

Proposición 3. *Para el cálculo del peso mínimo de un código no lineal* $\mathcal{C} = \bigcup_{i=0}^{t}(K_{\mathcal{C}} + v_i)$ *de longitud n tal que la dimensión de* $K_{\mathcal{C}}$ *como subespacio vectorial de* $\mathbb{F}_q^n$ *es* κ*. El costo computacional del algoritmo anterior es del orden de*

$$\sum_{i=0}^{t}\left((n-\kappa-1)\lceil n/(\kappa+1)\rceil \sum_{r=1}^{r_i}\binom{\kappa+1}{r}\right) \tag{1.27}$$

donde κ *es la dimensión del código lineal* $K_{\mathcal{C}}$ *y* r_i *es el menor entero tal que*

$$\lfloor n/(\kappa+1\rfloor)(r_i+1) + \max(0, r_i + 1 - (\kappa + 1 - n \mod (\kappa+1))) \geq w_H(K_{v_i})$$

Demostración.
Observe que el costo computacional de este algoritmo es el costo computacional de calcular $w_H(K_{v_i})$ para cada $i \in \{1,\ldots,t\}$. Y puesto que la dimensión de cada uno de esos códigos es $\kappa+1$, entonces la proposición se sigue de **(1.27)**. □

Calculemos el peso del código del ejemplo **(4)** usando este algoritmo.

Ejemplo 5 (Peso Mínimo)**.** Recordemos que la matriz generadora de $K_{\mathcal{C}}$ es

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

y los representantes de las clases que forman $\mathcal{C}$ son:

$$\begin{aligned} v_0 &= (0,0,0,0,0,0,0) \\ v_1 &= (0,1,0,0,1,1,0) \\ v_2 &= (1,0,1,1,0,0,0). \end{aligned}$$

El primer código lineal K_{v_1} viene dado por $\langle K_{\mathcal{C}}, v_1 \rangle$ es decir, por

$$\langle (1,0,0,1,1,1,1),(0,1,0,0,1,0,1),(0,0,1,1,1,0,0),(1,0,1,1,0,0,0) \rangle$$

y su matriz de paridad es

$$[H_2] = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

que da lugar al código K_{v_2} de longitud 7, dimensión 4 y peso mínimo 2. Luego, K_{v_1} es un código de longitud 7, dimensión 4 y peso mínimo 2, análogamente, K_{v_2} viene dado por $\langle K_{\mathcal{C}}, v_1 \rangle$ es decir, por

$$\langle (1,0,0,1,1,1,1),(0,1,0,0,1,0,1),(0,0,1,1,1,0,0),(1,0,1,1,0,0,0) \rangle.$$

Luego, su matriz de paridad es

$$[H_2] = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

que da lugar al código K_{v_2} de longitud 7, dimensión 4 y peso mínimo 2. Así, el código no lineal $\mathcal{C} = \cup_{i=0}^{2}(K_{\mathcal{C}} + v_i)$ tiene peso mínimo 2.

Observación 6. Usando este algoritmo en [16, 19] obtienen el peso mínimo del código no lineal $\mathcal{C}_{31}$ del ejemplo **(3)** en un tiempo de 6×10^{-4} segundos.

Algoritmo 3 Calcular la distancia mínima de un código no lineal

Entrada: $\mathcal{C}$-Código no lineal, $K_{\mathcal{C}}$-Kernel de $\mathcal{C}$ y $L = \{v_1, \ldots, v_t\}$-Clases de $K_{\mathcal{C}}$

Salida: $d_H(\mathcal{C})$-Distancia mínima de $\mathcal{C}$

```
d_H(C) ← n
para i ∈ [1,...,t−1] hacer
  para j ∈ [i+1,...,t] hacer
    K_{v_j−v_i} ← ⟨K_C, v_j − v_i⟩
    d_H(C) ← mín(w_H(K_{v_j+v_i}, d_H(C)))
  fin para
fin para
```

Para un código no lineal $\mathcal{C} = \bigcup_{i=0}^{t}(K_{\mathcal{C}} + v_i)$ de longitud n tal que la dimensión de $K_{\mathcal{C}}$ como subespacio vectorial de $\mathbb{F}_q^n$ es κ el costo computacional del algoritmo anterior es

$$\sum_{i=0}^{t-1}\left(\sum_{j=i+1}^{t}\left((n-\kappa-1)\lceil n/(\kappa+1)\rceil\sum_{r=1}^{r_{i,j}}\binom{\kappa+1}{r}\right)\right) \tag{1.28}$$

donde $r_{i,j}$ es el menor entero tal que

$$\lfloor n/(\kappa+1\rfloor)(r_{i,j}+1) + \max(0, r_{i,j}+1-(\kappa+1-n \mod (\kappa+1))) \geq w_H(K_{v_j-v_i})$$

Demostración.
Este algoritmo consiste en comparar el peso de todos los códigos $K_{v_1+v_j}$, así, el costo computacional del algoritmo anterior es la suma del costo computacional del cálculo de $w_H(K_{v_1+v_2})$ para cada $i, j \in \{1, \ldots, t\}$ tales que $i \neq j$, y tomando como costo computacional del cálculo del peso mínimo de un código lineal binario el enunciado en **(1.27)**, se sigue el orden de complejidad propuesto. □

Ejemplo 6 (Distancia Mínima)**.** Continuando con el código $\mathcal{C}$ del ejemplo **(4)**, como tenemos sólo dos clases distintas del núcleo que forman el código, enton-

ces según este algoritmo, la distancia mínima de $\mathcal{C}$ vendrá dada por

$$d_H(\mathcal{C}) = min\Big\{d_H(K_{v_1}), d_H(K_{v_2}), d_H(K_{v_1+v_2})\Big\}. \tag{1.29}$$

Pero , K_{v_1} y K_{v_2} son códigos lineales y por el ejemplo**(5)** ya sabemos que el peso mínimo de ambos es 2, por lo tanto, $d_H(K_{v_1}) = d_H(K_{v_2}) = 2$. Por otra parte, $v_1 + v_2 = (0,1,1,1,0,1,0)$, Así, $K_{v_1+v_2} = \langle K_{\mathcal{C}}, v_1 + v_2\rangle$ está definido por $\langle(1,0,0,1,1,1,1),(0,1,0,0,1,0,1),(0,0,1,1,1,0,0),(0,1,1,1,0,1,0)\rangle$ es decir, es un código lineal binario de longitud 7, dimensión 4 y peso mínimo 2. Luego $d_H(\mathcal{C}) = 2$.

Ejemplo 7. El código $\mathcal{C}_{31}$ de **(3)** nos proporciona un ejemplo más rico del cálculo de distancia mínima usando este algoritmo. Recordemos que el núcleo de $\mathcal{C}_{31}$ estaba dado por el código lineal $K_{\mathcal{C}}$ de longitud 31, dimensión 5 y distancia mínima 16 generado como el código dual del código de Hamming de longitud 31 y los vectores representantes de las clases venian dados por

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

Queremos calcular $w_H(K_{v_i+v_j}) = w_H(\langle K_{\mathcal{C}}, v_i + v_j\rangle)$ para cada par v_i, v_j distintos incluido v_0 el vector nulo. Las dimensiones de la matriz genaradora de $K_{\mathcal{C}}$ hacen poco práctico el mostrarla aquí, en su lugar veamos la siguiente tabla;

Usando el sotware matemático SageMath [13] obtuvimos las distancias mínimas de los códigos lineales binarios que aparecen a continuación.

Código lineal	distancia mínima
K_{v_1}	10
K_{v_2}	9
K_{v_3}	10
$K_{v_1+v_2}$	9
$K_{v_1+v_3}$	8
$K_{v_2+v_3}$	9

Así, $d_H(\mathcal{C}) = d_H(K_{v_1+v_3}) = 8$.

Observación 7. Usando este algoritmo en [19] obtienen la distancia mínima del código no lineal $\mathcal{C}_{31}$ del ejemplo **(3)** en un tiempo de $1{,}26 \times 10^{-3}$ segundos.

1.3.3. Descodificación

Extendiendo la idea de descodificación por síndrome de códigos lineales, obtenemos un primer método para descodificar nuestro código no lineal $\mathcal{C}$. La idea, como en el caso de códigos lineales, es que recibido un vector $u = c + e$ donde $c \in \mathcal{C}$, se quiere encontrar una palabra c' del código, que esté a mínima distancia de u, o equivalentemente, encontrar un vector $e' = u - c'$ de peso mínimo en el código no lineal $\mathcal{C} + u$. Nótese que si $\mathcal{C} = \bigcup_{i=0}^{t}(K_{\mathcal{C}} + v_i)$ entonces, podremos escribir $\mathcal{C}+u$ como $\mathcal{C}+u = \bigcup_{i=0}^{t}(K_{\mathcal{C}}+v_i+u)$, y si consideramos $K_{v_i+u} = K_{\mathcal{C}} \cup (K_{\mathcal{C}} + v_i + u)$ para cada $i \in \{0,\ldots,t\}$, basta aplicar la siguiente proposición.

Proposición 4. *Sea $\mathcal{C}$ un código binario no lineal de núcleo $K_{\mathcal{C}}$ con $d_H(K_{\mathcal{C}}) = d$ y clases representativas $\{v_1,\ldots,v_t\}$. Para un vector $u = c + e \notin \mathcal{C}$ donde $c \in \mathcal{C}$, sea $\mathcal{C}_u = \bigcup_{i=0}^{t}(K_{v_i+u})$, si $w_H(e) < d$ entonces u puede ser descodificado como $c' = u - e'$ donde e' es un vector de peso mínimo de $\mathcal{C}_u$, así, $w_H(e) = w_H(e')$. Nótese que si $w_H(e) < \lfloor \frac{d-1}{2} \rfloor$ entonces $e = e'$ y por lo tanto $c = c'$.*

Ahora bien, Nótese que antes de encontrar una palabra de peso mínimo, el peso mínimo de K_u es desconocido. Así, para estimar el costo computacional de la decodificación usando la proposición anterior, podemos usar el peso

mínimo $d = w_H(K_{\mathcal{C}})$ en lugar de $w_H(K_u)$ y obtener una cota superior del costo computacional dada por

$$\sum_{i=0}^{t}\left(n\lceil n/(\kappa+1)\rceil\sum_{r=1}^{r_i}\binom{\kappa+1}{r}\right) \tag{1.30}$$

donde κ es la dimensión del código lineal $K_{\mathcal{C}}$ y r_i es el menor entero tal que

$$\lfloor n/(\kappa+1\rfloor)(r_i+1)+\text{máx}(0,r_i+1-n \mod(\kappa+1))) \geq w_H(K_{v_i+u}). \tag{1.31}$$

Observación 8. Nótese que para decodificar una palabra recibida usando este resultado, se requiere calcular el peso mínimo de tantos códigos lineales como clases que formen el código no lineal. Además estos códigos lineales dependen directamente del vector recibido, así, al recibir una palabra diferente el proceso se tiene que iniciar de nuevo desde cero, esto hace que este proceso de descodificación sea poco eficiente.

2 Conjuntos de prueba minimales

El objetivo de este capítulo es presentar una construcción para un conjunto de prueba de tamaño mínimo para códigos binarios lineales utilizando bases de Gröbner. En este capítulo seguiremos las ideas expuestas en [3] por Borges-Quintana, Borges-Trenard, Fitzpatrick y Martínez-Moro para códigos lineales binarios.

2.1. Introducción a las bases de Gröbner

En esta sección fijaremos notación y daremos nociones básicas sobre las bases de Gröbner para posteriormente construir un conjunto de prueba de un código lineal (en principio binario) a partir de la base de Gröbner reducida de un ideal asociado. Durante este capítulo denotaremos por $\mathbb{K}[X]$ al anillo de polinomios donde $\mathbb{K}$ es un cuerpo arbitrario y $X = \{x_1, \ldots, x_n\}$ es un conjunto de n indeterminadas. Al monomio $x_1^{\alpha_1} x_1^{\alpha_2} \cdots x_1^{\alpha_n}$ lo denotaremos por $\underline{x}^{\underline{\alpha}}$ donde $\underline{\alpha} = (\alpha_1, \ldots, \alpha_n)$.

Definición 11. Un **orden monomial** $\prec$ es una relación de orden en el conjunto de los monomios que cumple lo siguiente:

1. Es total (cualesquiera par de monomios es comparable).
2. Es compatible con el producto (si $\underline{x}^{\underline{\alpha}} \prec \underline{x}^{\underline{\beta}}$ entonces $\underline{x}^{\underline{\alpha}} \underline{x}^{\underline{\gamma}} \prec \underline{x}^{\underline{\beta}} \underline{x}^{\underline{\gamma}}$).
3. Es un buen orden (toda colección no vacía de monomios tiene un elemento mínimo).

Para anillos de polinomios en varias variables existen muchas elecciones de ordenes monomiales. Implicitamente establecemos un orden en las variables de $\mathbb{K}[X]$ donde $x_1 \succ x_2 \succ \cdots \succ x_n$. Ejemplos clásicos de ordenes monomiales

son, el orden lexicográfico, el orden lexicográfico graduado y el orden lexicográfico graduado inverso [7, Ch.2,S.2].

Sea $f \in \mathbb{K}[X]$, el **término inicial** de f respecto del orden $\prec$, será el monomio mayor respecto a $\prec$ de f y se denotará por $in_{\prec}(f)$, mientras que indeg(f) será el **grado total** de $in_{\prec}(f)$, es decir la suma de los grados de todas las variables del monomio. Si $F = \{f_1, \ldots, f_r\} \subset A$ denotamos como $in_{\prec}\{F\}$ al conjunto de **todos los términos iniciales de los polinomios de** F respecto del orden $\prec$ y como $In_{\prec}(F)$ al **ideal generado por** $in_{\prec}\{F\}$. Finalmente, $\langle F \rangle$ es el **ideal polinomial generado por** F.

Definición 12. Sean F, un subconjunto finito y $\langle F \rangle$, $in_{\prec}\{F\}$ y $In_{\prec}(F)$ como arriba, un subconjunto finito $G_{\prec} = \{g_1, \ldots, g_s\}$ de $\langle F \rangle$ es una **base Gröbner** de $\langle F \rangle$ con respecto del orden monomial $\prec$ si genera el ideal $\langle F \rangle$ y

$$In_{\prec}(F) = \langle in_{\prec}(g_1), \ldots, in_{\prec}(g_s) \rangle = In_{\prec}(G_{\prec}). \tag{2.1}$$

Es decir, el ideal monomial generado por los términos iniciales de los elementos de F es igual al ideal generado por por términos iniciales de los elementos de $G_{\prec}$.

Definición 13. Una base de Gröbner $G_{\prec} = \{g_1, \ldots, g_s\}$ de $\langle F \rangle$ con respecto al orden monomial $\prec$ se dice **minimal** si cumple

1. El coeficiente de $in_{\prec}(g_i)$ es 1 para todo $1 \leq i \leq s$.

2. Para cada $g_i \in G_{\prec}$, $in_{\prec}(g_i) \notin in_{\prec}\{G_{\prec} \setminus \{g_i\}\}$.

Sean $f \in \mathbb{K}[X]$ y $F = \{f_1, \ldots, f_r\} \subset \mathbb{K}[X]$ como anteriormente. Se puede probar que existe un algoritmo de división con resto y cociente único de cualquier polinomio en $\mathbb{K}[X]$ por la base de Gröbner [7]. Llamaremos $Red_{\prec}(f, F)$ al resto de la división de f por la lista de polinomios F con respecto del orden monomial

$\prec$. Si $G_\prec$ es una base de Gröbner del ideal $\langle F\rangle$, entonces $Red_\prec(f, G_\prec)$ és llamada la forma normal de f en $\langle F\rangle$ con respecto del orden monomial $\prec$.

Una base de Gröbner para un ideal I puede ser calculada a partir de cualquier conjunto de generadores finito de I (nótese que el teorema de la base de Hilbert enuncia que todo ideal de $\mathbb{K}[X]$ tiene un sistema finito de generadores, ver por ejemplo [7]) por un método que fue introducido en [5]. Buchberger fue el primero en dar un algoritmo para el cálculo de las bases de Gröbner. Antes de presentar su algoritmo, daremos una definición introducida también en [5] que es la idea central del algoritmo.

Definición 14. Sean $f, g \in \mathbb{K}[X]$ polinomios no nulos y sea $\prec$ cualquier orden monomial, el **S-polinomio** de f y g con respecto a $\prec$ es la combinación:

$$SPol(f,g) = \frac{\underline{x}^\gamma}{in_\prec(f)} f - \frac{\underline{x}^\gamma}{in_\prec(g)} g \tag{2.2}$$

donde $\underline{x}^\gamma$ es el mínimo común multiplo de $in_\prec(f)$ y $in_\prec(g)$.

Note que el S-polinomio $SPol(f,g)$ está definido para producir la cancelación de los términos iniciales de f y g. El algoritmo se basa en el siguiente teorema.

Teorema 6 (Criterio de Buchberger)**.** *Sea I un ideal polinomial, $\mathcal{G}$ es una base de Gröbner de I con respecto del orden monomial $\prec$ si y solo si cada par $f, g \in \mathcal{G}$ con $f \neq g$ la forma reducción del correspondiente S-polinomio $S(f,g)$ por $\mathcal{G}$ es cero, es decir, si $Red_\prec(SPol(f,g), \mathcal{G}) = 0$, $\forall f, g \in \mathcal{G}$ tales que $f \neq g$.*

cuya demostración se puede encontrar en en [7, Ch.2,S.6,Th.6].

Algoritmo 4 Algoritmo de Buchberger

Entrada: $F = \{f_1, \ldots, f_n\}$ - Sistema de generadores de un ideal I y $\prec$ - Un orden monomial.

Salida: $\mathcal{G} = \{g_1, \ldots, g_s\}$ - Base de Gröbner de I.

```
G ← F
G' ← {0}
mientras G ≠ G' hacer
  G' ← G
  para p, q ∈ G' : p ≠ q hacer
    S = Red_≺(SPol(p, q), G')
    si S ≠ 0 entonces
      G ← G ∪ {S}
    fin si
  fin para
fin mientras
```

Además, a partir de cualquier base de Gröbner $\mathcal{G}$ se puede obtener una base de Gröbner minimal usando el siguiente resultado que elimina la información redundante de $\mathcal{G}$. En efecto,

Lema 3. *Sean $\prec$ un orden monomial sobre $\mathbb{K}[x]$ y $\mathcal{G}$ una base de Gröbner del ideal $I \subset \mathbb{K}[x]$ respecto de $\prec$. Si $p \in \mathcal{G}$ es un elemento de $\mathcal{G}$ tal que $in_\prec(p) \in \langle In_\prec(\mathcal{G} \setminus \{p\})\rangle$, entonces $\mathcal{G} \setminus \{p\}$ es también una base de Gröbner de I respecto de $\prec$.*

La demostración de este lema es una consecuencia directa de la definición de base de Gröbner.

Definición 15. Una base de Gröbner $G_\prec = \{g_1, \ldots, g_s\}$ de $\langle F\rangle$ con respecto al orden monomial $\prec$ se dice **reducida** si cumple:

1. El coeficiente de $in_\prec(g_i)$ es 1 para todo $1 \leq i \leq s$.
2. Para cada $g_i \in G_\prec$, ningun término de g_i está en $In_\prec(G_\prec \setminus \{g_i\})$.

Finalmente, podemos enunciar algunas propiedades fundamentales de las bases de Gröbner minimales que nos serán útiles en el siguiente capítulo.

Proposición 5. *1. Todo ideal $I \in \mathbb{K}[X]$ tiene una base de Gröbner con respecto de un orden monomial $\prec$.*

2. Toda base de Gröbner de un ideal $I \in \mathbb{K}[X]$ también es un sistema de generadores de I.

3. Todo ideal $I \in \mathbb{K}[X]$ no nulo, tiene una única base de Gröbner reducida.

Las demostraciones de 1 y 2 forman parte del corolario 6 en [7, Ch.2,S.5] mientras que 3 se puede ver en [7, Ch.2,S.7]. El apartado 1 de la proposición anterior nos garantiza la existencia de una base de Gröbner reducida para cualquier ideal polinomial.

2.1.1. Bases de Gröbner sobre módulos

Definición 16. Un módulo M sobre un anillo A (conmutativo con elemento neutro) es un conjunto dotado de una operación + y una multiplicación escalar que verifican las siguiente propiedades

1. $(M,+)$ es un grupo abeliano.
2. $a(f+g) = af + ag$
3. $(a+b)f = af + bf$
4. $(ab)f = a(bf)$
5. $1f = f$

para todo $a,b \in R$, $f,g \in M$.

Ejemplo 8. El ejemplo más sencillo de módulo es A^m, siendo A un anillo.

Como en la sección anterior el anillo al que prestaremos atención será $A = \mathbb{K}[x_1, \ldots, x_n] = \mathbb{K}[X]$, el anillo de polinomios sobre el cuerpo $\mathbb{K}$. En esta sección trabajaremos siempre con módulos definidos sobre el anillo de polinomios A, concretamente con A^m, $m \geq 1$, y sus submódulos.

Un **submódulo** de un módulo M sobre A es un subconjunto de A^m que es cerrado para la suma y el producto por elementos de A, es decir, aquellos subconjuntos que son módulos en sí mismos. Denotamos por $\langle \mathbf{f}_1, \ldots, \mathbf{f}_s \rangle \subset M$ el submódulo generado por $\mathbf{f}_1, \ldots, \mathbf{f}_s$, es decir, el conjunto de combinaciones lineales de la forma $a_1\mathbf{f}_1 + \cdots + a_s\mathbf{f}_s$, con $a_1, \ldots, a_s \in A$.

Una base de un módulo es un conjunto de generadores linealemente independientes. Un espacio vectorial siempre tiene una base, en cambio, para módulos esta propiedad no se cumple en general. A los módulos que tienen base se los denomina **módulos libres**. Por ejemplo, A^m es un módulo libre, una base -denominada base estándar de A^m- es $\mathbf{e}_1 = (1, 0, \ldots, 0)$, $\mathbf{e}_2 = (0, 1, 0, \ldots, 0)$, $\ldots$, $\mathbf{e}_m = (0, , \ldots, 0, 1)$.

En esta sección introducimos los órdenes monomiales en los anillos libres A^m, $m \geq 1$, y las bases de Gröbner para submódulos $M \subset A^m$. El proceso para desarrollar bases de Gröbner en A^m será dar una definición "adecuada" de monomio y orden monomial (y sus conceptos asociados). Una vez que se tengan estos dos conceptos el resto de definiciones y resultados se obtendrá de forma inmediata por su analogía con las bases de Gröbner de ideales en anillos que vimos en la sección anterior.

Definición 17. Decimos que $\underline{m} \in A^m$ es un monomio si $\underline{m} = \underline{x}^{\underline{\alpha}}\mathbf{e}_i$, con $\underline{x}^{\underline{\alpha}}$, $i \in \{1, \ldots, m\}$.

De esta forma todo elemento $f \in A^m$ se escribe de forma única como combinación $\mathbb{K}$-lineal de monomios. Cada sumando de la combinación lineal, es

decir, el producto $c\mathbf{m}$ de un elemento de $\mathbb{K}$ por un monomio se denomina **término**.

Sean $\underline{m} = \underline{x}^{\underline{\alpha}}\underline{e}_i, \underline{n} = \underline{x}^{\underline{\beta}}\underline{e}_j$ dos monomios. Decimos que $\underline{n}$ **divide** a $\underline{m}$, y lo denotamos por $\underline{n}|\underline{m}$, si $i = j$ y $\underline{x}^{\underline{\beta}}$ divide a $\underline{x}^{\underline{\alpha}}$. Por tanto, el cociente $\underline{m}/\underline{n} = \underline{x}^{\neq\alpha-\neq\beta}$ es un elemento de A. En este caso, el máximo común divisior y el mínimo común múltiplo de $\underline{m}$ y $\underline{n}$, $\mathrm{mcd}(\underline{m},\underline{n})$ y $\mathrm{mcm}(\underline{m},\underline{n})$, son $\mathrm{mcd}(x^{\underline{a}}, x^{\underline{b}}) \cdot \underline{e}_i$ y $\mathrm{mcm}(x^{\underline{a}}, x^{\underline{b}}) \cdot \underline{e}_i$ (respectivamente). En caso contrario ($i \neq j$), se definen el máximo común divisor y el mínimo común múltiplo como $\mathbf{0}$.

Al igual que con anillos e ideales, para desarrollar la teoría de bases de Gröbner sobre módulos necesitaremos definir órdenes monomiales, construir el algoritmo de división y extender el algoritmo de Buchberger para módulos.

La definición de orden monomial en R^m es la extensión natural de la definción para anillos, para $m = 1$ ambas definiciones coinciden.

Definición 18. Un orden monomial sobre los monomios de A^m es un orden total $\prec$ que verifica:

1. $\underline{e}_i \prec \underline{x}^{\underline{\alpha}}\underline{e}_i$ para todo monomio $\underline{x}^{\underline{\alpha}}$ de A distinto de 1. Esta condición es equivalente a que $\prec$ sea un buen orden.

2. Si dos monomios de A^m verifican $\underline{m} \prec \underline{n}$ entonces $\underline{x}^{\underline{\gamma}}\underline{m} \prec \underline{x}^{\underline{\gamma}}\underline{n}$ para todo monomio $\underline{x}^{\underline{\gamma}}$ de A.

Los dos órdenes monomiales habitualmente utilizados en A^m son una extensión de los órdenes monomiales de A. Para ello debemos considerar un orden monomial en A y fijar un orden en la base estándar de A^m, nosotros consideraremos $\underline{e}_1 \succ \underline{e}_2 \succ \cdots \succ \underline{e}_m$.

Definición 19. (Orden TOP-*Term Over Position*, término sobre la posición). Sea $\prec$ un orden monomial en A y sean $\underline{x}^{\neq\alpha}\underline{e}_i$ y $\underline{x}^{\neq\beta}\underline{e}_j$ dos monomios de A^m. Decimos que $\underline{x}^{\neq\alpha}\underline{e}_i \succ_{\mathrm{TOP}} \underline{x}^{\neq\beta}\underline{e}_j$ si $\underline{x}^{\neq\alpha} \succ \underline{x}^{\neq\beta}$, o si $\underline{x}^{\neq\alpha} = \underline{x}^{\neq\beta}$ y $i < j$.

Definición 20. (Orden POT-*Position Over Term*, posición sobre el término). Sea $\prec$ un orden monomial en A y sean $\underline{x}^{\neq\alpha}\underline{e}_i$ y $\underline{x}^{\neq\beta}\underline{e}_j$ dos monomios de A^m. Decimos que $\underline{x}^{\neq\alpha}\underline{e}_i \succ_{\text{POT}} \underline{x}^{\neq\beta}\underline{e}_j$ si $i < j$, o si $i = j$ y $\underline{x}^{\neq\alpha} \succ \underline{x}^{\neq\beta}$.

Sea $\mathbf{f} \in A^m$. Podemos escribir $\mathbf{f}$ como $\mathbf{f} = \sum c_i \underline{m}_i$, con $0 \neq c_i \in \mathbb{K}$. Como en el caso de ideales, llamamos **término inicial** de $\mathbf{f}$ con respecto al orden monomial $\prec$ y lo denotamos por $\text{in}_{\succ}(\mathbf{f})$ al término $c_j\underline{m}_j$ donde $\underline{m}_j$ es el mayor monomio que aparece en $\mathbf{f}$ para el orden monomial $\prec$. Análogamente, definimos el **coeficiente inicial** de $\mathbf{f}$ como $\text{ic}_{\succ}(\mathbf{f}) = c_i$ y el **monomio inicial** de $\mathbf{f}$ como $\text{im}_{\succ}(\mathbf{f}) = \underline{m}_i$.

Ejemplo 9. En este ejemplo comprobamos cómo, para un mismo elemento, se tienen términos iniciales diferentes para una extensión POT y TOP. Consideramos A^2, con $A = \mathbb{F}_q[x, y]$. Sean

- $\mathbf{f}_1 = (y^2, x^2 - y) = y^2\underline{e}_1 + (x^2 - y)\underline{e}_2 \in \mathbb{F}_q[x, y]^2$
- $\mathbf{f}_2 = (x^2 + 1, y^3) = (x^2 + 1)\underline{e}_1 + y^3\underline{e}_2 \in \mathbb{F}_q[x, y]^2$

Consideramos $\underline{e}_1 \succ \underline{e}_2$, el orden lexicográfico en $\mathbb{F}_q[x, y]$ con $x \succ y$. Entonces se tiene que

- $\text{in}_{\succ_{\text{POT}}}(\mathbf{f}_1) = y^2\underline{e}_1$
- $\text{in}_{\succ_{\text{POT}}}(\mathbf{f}_2) = (x^2)\underline{e}_1$
- $\text{mcm}(y^2\underline{e}_1, (x^2 + 1)\underline{e}_1) = (x^2y^2 + y^2)\underline{e}_1$

En cambio, si consideramos la extensión TOP, tenemos

- $\text{in}_{\succ_{\text{TOP}}}(\mathbf{f}_1) = (x^2)\underline{e}_2$
- $\text{in}_{\succ_{\text{TOP}}}(\mathbf{f}_2) = (x^2)\underline{e}_1$
- $\text{mcm}((x^2 - y)\underline{e}_2, (x^2 + 1)\underline{e}_1) =\neq 0$

En A^m también se tiene un algoritmo de división que extiende el algoritmo de división multivariable.

Proposición 6. *Sea $\succ$ un orden monomial en A^m y sea $\mathbf{f}_1, \ldots, \mathbf{f}_s$ una s-upla ordenada de elementos de A^m. Entonces $f \in A^m$ puede expresarse de la forma siguiente*

$$\mathbf{f} = a_1\mathbf{f}_1 + \cdots + a_s\mathbf{f}_s + \mathbf{r}$$

donde $a_i \in A$ y el resto $\mathbf{r} \in A^m$ es o bien $\mathbf{0}$, o bien una combinación lineal de monomios no divisibles por los monomios $\{\mathrm{MonIn}(\mathbf{f}_i)\}_{i=1}^s$.

Al igual que en anillos, el resultado de la división depende del orden monomial elegido y de la ordenación de la s-upla $\mathbf{f}_1, \ldots, \mathbf{f}_s$. El algoritmo de división por sí solo tampoco resuelve el problema de pertenencia a un submódulo $M = \langle \mathbf{f}_1, \ldots, \mathbf{f}_s \rangle$, como muestra el ejemplo **(10)**. Si al dividir $\mathbf{f}$ por $\mathbf{f}_1, \ldots, \mathbf{f}_s$ obtenemos resto $\mathbf{r} = \mathbf{0}$ entonces $\mathbf{f} \in M$, pero el recípropo no es cierto en general. Para tener el resultado recíproco necesitaremos unos generadores especiales de M, la denominada base de Gröbner de un submódulo.

Ejemplo 10. Consideramos el módulo $\mathbb{F}_q[x,y]^2$ y $\mathbf{f}_1 = (y^2, x^2 - y)$, $\mathbf{f}_2 = (x^2 + 1, y^3)$ como en el ejemplo **(9)**. Sea $\mathbf{f}_3 = (0, x^4 - x^2y + x^2 - y^5 - y) \in \mathbb{F}_q[x,y]^2$.

Consideramos $\underline{e}_1 \succ \underline{e}_2$, el orden lexicográfico en $\mathbb{F}_q[x,y]$ con $x \succ y$, y la extensión POT. La división de $\mathbf{f}_3$ por $\mathbf{f}_1, \mathbf{f}_2$ produce

$$\mathbf{f}_3 = 0\mathbf{f}_1 + 0\mathbf{f}_2 + \mathbf{r}$$

donde $\mathbf{r} = \mathbf{f}_3$ puesto que el término inicial de $\mathbf{f}_3$ no es divisible por los términos iniciales de $\mathbf{f}_1$ y de $\mathbf{f}_2$.

En cambio, si consideramos el orden TOP y dividimos $\mathbf{f}_3$ por $\mathbf{f}_1, \mathbf{f}_2$ tenemos que

$$\mathbf{f}_3 = (x^2 + 1)\mathbf{f}_1 - y^2\mathbf{f}_2 + \mathbf{0}$$

y por tanto $\mathbf{f}_3$ pertenece al submódulo de $\mathbb{F}_q[x,y]^2$ generado por $\{\mathbf{f}_1, \mathbf{f}_2\}$.

Definición 21. Para un orden monomial $\prec$ y un submódulo $M \subset A^m$, decimos que que el conjunto $\{\mathbf{f}_1, \ldots, \mathbf{f}_s\} \subset A$ es una base de Gröbner si

$$\langle \mathrm{in}_{\succ}(\mathbf{f}_1), \ldots, \mathrm{in}_{\succ}(\mathbf{f}_s) \rangle = \langle \mathrm{in}_{\succ}(M) \rangle$$

donde $\langle \mathrm{in}_{\succ}(M) \rangle$ es el submódulo generado por los términos iniciales de los elementos de M con respecto a $\prec$.

Todas las propiedades de la sección anterior para bases de Gröbner son válidas también para submódulos de A^m con idénticas demostraciones. En particular, las bases de Gröbner permiten resolver el problema de pertenencia a un submódulo, proporcionando un sistema de generadores.

Proposición 7. *Sea $\prec$ un orden monomial, G una base de Gröbner de un submódulo $M \subset A^m$ y $\mathbf{f} \in A^m$. Entonces se tiene que*

1. *M es finitamente generado, equivalentemente, los submódulos de A^m verifican la condición de cadena ascendente.*

2. *$\mathbf{f} \in M$ si y sólo si el resto $\mathbf{r}$ de la división por G es cero.*

3. *G genera M como módulo.*

Aunque se utiliza la terminología de base de Gröbner para el sistema de generadores G anteriormente definido (por su analogía con el caso de anillos e ideales), cabe destacar que G no es una base del submódulo M en general, es decir, los elementos de una base de Gröbner de M son un sistema de generadores pero no tienen por qué ser linealmente independientes.

Finalmente, note que existen las versiones para módulos de S-polinomios, criterio de Buchberger y algorimo de Buchberger y son completamente análogos al de ideales.

Ejemplo 11. Consideramos el módulo $\mathbb{F}_q[x,y]^2$ y $\mathbf{f}_1 = (y^2, x^2 - y)$, $\mathbf{f}_2 = (x^2 + 1, y^3)$ como en los ejemplos **(9)** y **(10)**. También consideramos $\underline{e}_1 \succ \underline{e}_2$, el orden lexicográfico en $\mathbb{F}_q[x,y]$ con $x \succ y$, y el orden monomial en $\mathbb{F}_q[x,y]^2$ que da la extensión POT. Sea $M = \langle \mathbf{f}_1, \mathbf{f}_2 \rangle$. El S-Polinomio de $\mathbf{f}_1$ y $\mathbf{f}_2$ es

$$S(\mathbf{f}_1, \mathbf{f}_2) = (x^2)\mathbf{f}_1 - y^2\mathbf{f}_2 = (-y^2, x^4 - x^2y - y^5)$$

Luego, como vimos en el ejemplo (10), $\{\mathbf{f}_1, \mathbf{f}_2\}$ no es una base de Gröbner para $\prec_{\text{POT}}$ puesto que $Red_{\succ_{\text{POT}}}(SPol(\mathbf{f}_1, \mathbf{f}_2), G) \neq \mathbf{0}$. Siguiendo el algoritmo de Buchberger, consideramos como generadores de M $\{\mathbf{f}_1, \mathbf{f}_2, S(\mathbf{f}_1, \mathbf{f}_2)\}$. El lector puede comprobar fácilmente que la clase de cada S-polinomio de dos generadores es igual a $\mathbf{0}$, y por tanto $\{\mathbf{f}_1, \mathbf{f}_2, S(\mathbf{f}_1, \mathbf{f}_2)\}$ es una base de Gröbner de M para $\succ_{\text{POT}}$.

En cambio, si consideramos la extensión TOP del orden monomial en $\mathbb{F}_q[x,y]^2$, tenemos que

$$S(\mathbf{f}_1, \mathbf{f}_2) = \mathbf{0}$$

puesto que $\text{mcm}(\text{MonIn}_{\succ_{\text{TOP}}}(\mathbf{f}_1), \text{MonIn}_{\succ_{\text{TOP}}}(\mathbf{f}_1)) = \mathbf{0}$ (ver ejemplo 9). Por tanto, siguiendo el criterio de Buchberger, tenemos que $\{\mathbf{f}_1, \mathbf{f}_2\}$ es una base de Gröbner para $\succ_{\text{TOP}}$.

Fijado un orden monomial, se pueden definir bases de Gröbner minimales y reducidas de la misma forma que para ideales de anillos. Igualmente, se tiene que existe una única base de Gröbner reducida para cada orden monomial.

El siguiente resultado muestra cuándo A^m/M tiene dimensión finita como espacio vectorial sobre $\mathbb{K}$ para un submódulo $M \subset A^m$.

Proposición 8. *Sea M un submódulo de A^m y $\prec$ un orden monomial en M. El $\mathbb{K}$-espacio vectorial A^m/M tiene dimensión finita si y sólo si para todo $i \in \{1, \ldots, m\}$ y para todo $j \in \{1, \ldots, n\}$, existe $\mathbf{f} \in M$ tal que $\text{MonIn}_{\succ}(\mathbf{f}) = x_j^u \underline{e}_i$, para algún $a \geq 0$.*

Demostración. Sea G una base de Gröbner de M para el orden $\prec$. Los elementos de A^m/M son combinaciones lineales de monomios pertenecientes al comple-

mentario de $\langle \text{in}_{\succ}(M)\rangle$, al igual que para ideales en anillos. Y se tiene el resultado, puesto que el número de monomios en el complmentario de $\langle \text{in}_{\succ}(M)\rangle$ es finito si y sólo si se verifica la condición del enunciado. □

2.1.2. Cálculo de la base de Gröbner de ciertos ideales

En este apartado incluimos una forma de calcular una base de Gröbner de un ideal en $\mathbb{F}_2[X]$ que incluye al conjunto $\left\{x_i^2+1\right\}$ mediante técnicas del álgebra lineal. Este tipo de ideales será el que asociaremos a un código lineal binario. Primero daremos un enfoque general sobre la idea de calcular las bases de Gröbner vía los módulos de sizigias, este tipo de cálculos son incluso anteriores al planteamiento de las bases de Gröbner y fueron formalizados por Caboara y Traverso en [6] y forman el núcleo de las ideas el los algoritmos FGLM [8] para el cálculo de bases de Gröbner de ideales 0-dimensionales mediante álgebra lineal.

Plantearemos primero la idea general. Dado un conjunto $F = \{f_1, f_2, \ldots, f_r\}$ de polinomios en $\mathbb{K}[X] = \mathbb{K}[x_1, \ldots, x_n]$ que generan un ideal I calculemos una base para el módulo de sizigias M en $\mathbb{K}[X]^{r+1}$ del conjunto generador $F' = \{-1, f_1, f_2, \ldots, f_r\}$. cada una de las sizigias corresponde a la solución de

$$f = \sum_{i=1}^{r} b_i f_i \qquad b_i \in \mathbb{K}[X], i = 1, \ldots, r$$

La idea principal es que el conjunto

$$\begin{aligned} \mathbf{f}_1 &= (f_1, 1, 0, 0, \ldots, 0) \\ \mathbf{f}_2 &= (f_2, 0, 1, 0, \ldots, 0) \\ &\vdots \\ \mathbf{f}_r &= (f_r, 0, 0, 0, \ldots, 1) \end{aligned} \tag{2.3}$$

es una base del módulo de sizigias M y más aún es una base de Gröbner con

respecto al orden monomial posición sobre término (POT) $<_{\mathbf{w}}$ inducido por un orden monomial $<$ en $\mathbb{K}[X]$ y el vector $\mathbf{w} = (1, in_<(f_1), \ldots, in_<(f_r))$. Además, el término inicial de $\mathbf{f}_i$ es $\underline{e}_{(i+1)}$ con respecto al orden monomial $<_{\mathbf{w}}$ donde $\underline{e}_{(j)}$ denota el vector canónico de longitud $r+1$.

Si ahora, utilizando las ideas en [8] reordenamos los términos de $\mathbb{K}[X]^{r+1}$ en el nuevo orden determinado por $<$ y $\underline{e}_{(i)} < \underline{e}_{(j)}$ Si $i < j$, usando un orden de posición sobre posición (TOP), obtenemos una nueva base para el módulo. En cada paso (esencialmente eliminación gaussiana) la forma normal del término con respecto a la base original es 0 salvo la primera componente por lo que la determinación de las relaciones lineales tiene lugar en esa componente. Esto proporciona una representación conveniente para la forma normal con respecto a la base de Gröbner inicial como un espacio vectorial de $\mathbb{K}$, y cualquier relación lineal obtenida como consecuencia de la reducción de la primera componente en $\mathbb{K}[X]$ nos dará una relación correspondiente para los elementos del módulo. Mostremos la idea con un ejemplo.

Ejemplo 12. Sea $I = \left\langle x^2 + x + 1, xy + x + 1 \right\rangle$ en $\mathbb{F}_2[x, y]$ y tomemos $<$ el orden deglex con $x < y$. Mostrando solo las primeras componentes tenemos

	1	x	y	x^2	xy	y^2	x^3	x^2y	xy^2	y^3
$(1,0,0)$	1									
$(0,1,0)$	1	1		1						
$(0,0,1)$	1	1			1					
después de reducir										
$(1,0,0)$	1									
$(1,1,0)$		1		1						
$(0,1,1)$				1	1					
introducida x										
$(x,0,0)$		1								
$(x,x,0)$				1			1			
$(0,x,x)$							1	1		
después de reducir										
$(x+1,0,1)$					1					
$(1,x+1,0)$							1			
$(1,1,x)$								1		

	1	x	y	x^2	xy	y^2	x^3	x^2y	xy^2	y^3
introducida y										
$(y,0,0)$			1							
$(y,y,0)$					1			1		
$(0,y,y)$								1	1	
depués de reducir										
$(y,0,0)$			1							
$(y+x,y+1,x+1)$										
$(1,y+1,x+y)$									1	

Así $(y+x,y+1,x+1)$ es una sizigia y por lo tanto $y+x \in I$ y es el primer elemento en el orden deglex; ahora podemos omitir todos los multiplos de $y(1,1,0)$. Continuando el cálculo tenemos

	1	x	y	x^2	xy	y^2	x^3	x^2y	xy^2	y^3	x^4	x^3y
introducida x^2												
$(x^2+x,0,x)$								1				
$(x,x^2+x,0)$											1	
(x,x,x^2)												1
después de reducir												
$(x^2+x+1,1,0)$												
$(x,x^2+x,0)$											1	
(x,x,x^2)												1

Así, $(x^2+x+1,1,0)$ es una sizigia y x^2+x+1 es el segundo elemento en la base en I con respecto al orden deglex. Ahora podemos omitir los multiplos de $x^2(1,0,0)$. Por lo tanto, no se pueden calcular más sizigias con la primera componente distinta de cero y se sigue que $\{y+x, x^2+x+1\}$ debe ser un elemento de la base de Gröbner.

El procedimiento anterior es completamente general y puede utilizarse para cualquier cuerpo base. Aunque se utiliza sólo álgebra lineal, tiene un inconveniente importante en que para determinar que un polinomio f pertenece al ideal (en cuyo caso f será un elemento de la base de Gröbner). Es necesario calcular la representación mínima $f = \sum h_i f_i$ donde los f_i son los generadores iniciales. Se sabe que los grados de los cofactores h_i pueden ser doblemente exponenciales en n, el número de variables (Problema Nullstellensatz [4]).

Sin embargo si pretendemos calcular una base de Gröbner de un ideal en $\mathbb{F}_2[X]$ que incluye al conjunto $\left\{x_i^2+1\right\}$ (como en el caso de la base Gröbner asociada a un código binario presentada en la sección siguiente), el algoritmo es ventajoso por las siguientes razones.

1. No hay crecimiento de coeficientes.

2. La presencia del binomio $x_i^2 + 1$, y del resto de los generadores iniciales con términos en forma estándar, significa que los términos del cofactor correspondientes a cualquier polinomio en el ideal (con términos en forma estándar) también están en forma estándar.

3. La longitud máxima de un elemento $w \in [X]$ en forma estándar es n. Así, en nuestro caso, el grado máximo de los cofactores h_i puede crecer tanto como el número de variables (n).

Ejemplo 13. Tomemos ahora $I = \left\langle x_1x_3 + 1, x_2x_3 + 1, x_1^2 + 1, x_2^2 + 1, x_3^2 + 1 \right\rangle$ en el anillo de polinomios $\mathbb{F}_2[x_1, x_2, x_3]$ y tomemos $<$ el orden deglex con $x_1 < x_2 < x_3$.

En el cálculo asociado a la sizigia, las filas correspondientes a los binomios $x_i^2 - 1$ se consideran implícitas en los cálculos: véase, por ejemplo, en la tabla siguiente, cuando se obtiene la sizigia correspondiente a $x_3 - x_1$.

	1	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$	multiplos de x_i^2
$(1,0,0,0,0,0)$	1								
$(1,1,0,0,0,0)$						1			
$(1,0,1,0,0,0)$							1		
introducex_1									
$(x_1,0,0,0,0,0)$		1							
$(x_1,x_1,0,0,0,0)$									$x_1^2x_3$
$(x_1,0,x_1,0,0,0)$								1	
introducex_2									
$(x_2,0,0,0,0,0)$			1						
$(x_2,x_2,0,0,0,0)$								1	
$(x_2,0,x_2,0,0,0)$									$x_2^2x_3$
después de reducir									
$(x_2,0,0,0,0,0)$			1						
$(x_2-x_1,x_2,x_1,0,0,0)$									
$(x_2,0,x_2,0,0,0)$									$x_2^2x_3$

Así, $x_2 - x_1 = x_1f_2 - x_2f_1$ pertenece a la base de Gröbner y ahora podemos omitir todos los múltiplos de $x_2(1,1,0,0,0,0,0)$ de nuestro cálculo. Continuando encontramos

	1	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$	multiplos de x_i^2
introducex_3									
$(x_3, 0, 0, 0, 0, 0)$				1					
$(x_3, x_3, 0, 0, 0, 0)$									$x_1x_3^2$
$(x_3, 0, x_3, 0, 0, 0)$									$x_2x_3^2$
después de reducir									
$(x_3, 0, 0, 0, 0, 0)$				1					
$(x_3 - x_1, x_3, 0, 0, 0, x_1)$									
$(x_3 - x_2, 0, x_3, 0, 0, x_2)$									

De donde tenemos que la sizigia $x_3 - x_1 = x_1 f_5 - x_3 f_1$ pertenece a la base de Gröbner. El resultado del cálculo es la base de Gröbner $x_2 - x_1, x_3 - x_1, x_1^2 - 1$

2.2. Construcción de un conjunto de prueba para códigos lineales binarios

Durante este apartado nos referiremos a $\mathcal{C}$ siempre como un código lineal binario de longitud n con matriz generatiz G y matriz de paridad H. Sea $[X]$ el monoide libre conmutativo generado por n variables $X = \{x_1, x_2, \ldots, x_n\}$. Hay una correspondencia natural de X en la base canónica de $\mathbb{F}_2^n$

$$\begin{aligned} \psi : [X] &\rightarrow \mathbb{F}_2^n \\ x_i &\mapsto \underline{e}^{(i)} \text{ donde } \underline{e}_j^{(i)} = 0, \forall j \neq i \text{ y } \underline{e}_i^{(i)} = 1 \end{aligned} \tag{2.4}$$

Sea $\underline{a} = (a_i)_{i=1}^n$ un elemento de $\mathbb{N}^n$. Denotaremos como $\underline{x}^{\underline{a}}$ al monomio $\prod_{i=1}^n x_i^{a_i} \in [X]$. La aplicación que definimos antes se puede extender a un morfismo de $[X]$

en $\mathbb{F}_2^n$ sobreyectivo como sigue:

$$\psi(x^{\underline{a}}) = \psi\left(\prod_{i=1}^{n} x_i^{a_i}\right) = (a_1(mód\ 2), \ldots, a_n(mód\ 2)) \tag{2.5}$$

es decir,

$$x^{\underline{a}} = 1 \cdot x_{i_1} \cdots x_{i_m} \in [X] \mapsto \psi(x^{\underline{a}}) = 0 + e^{(i_1)} + \ldots + e^{(i_m)} \in \mathbb{F}_2^n \tag{2.6}$$

Por ejemplo, $x_1x_3x_5 \in [X]$ con $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ tiene como imagen al vector $(1,0,1,0,1,0) \in \mathbb{F}_2^6$.

Definimos una relación $\mathcal{R}_{\mathcal{C}}$ en $\mathbb{F}_2^n$ con respecto al código $\mathcal{C}$ como sigue, $\underline{y_1}, \underline{y_2} \in \mathcal{R}$ si, y sólo si, pertenecen a la misma clase de $\mathbb{F}_2^n/\mathcal{C}$

$$(\underline{y_1}, \underline{y_2}) \in \mathcal{R}_{\mathcal{C}} \Leftrightarrow \underline{y_1} - \underline{y_2} \in \mathcal{C} \tag{2.7}$$

Podemos transpasar dicha relación a nuestro monoide de la siguiente forma, tomemos $\xi(x^{\underline{a}}) = \psi(x^{\underline{a}})H^T$ cuando $x^{\underline{a}} \in [X]$, entonces

$$x^{\underline{a}} \equiv_{\mathcal{C}} x^{\underline{b}} \Leftrightarrow (\psi(x^{\underline{a}}), \psi(x^{\underline{b}})) \in \mathcal{R}_{\mathcal{C}} \Leftrightarrow \xi(x^{\underline{a}}) = \xi(x^{\underline{b}});\ x^{\underline{a}}, x^{\underline{b}} \in [X] \tag{2.8}$$

El morfismo ξ representa la transición de síndromes de $\mathbb{F}_2^n$ a $[X]$. Así, $\xi(x^{\underline{a}})$ es el síndrome de $x^{\underline{a}}$ que a su vez es el síndrome de $\psi(x^{\underline{a}})$. La conexión entre las dos estructuras se puede entender mejor a partir de **(2.6)** y el hecho de que

$$\forall u, v \in [X], (\psi(u) = \psi(v) \Leftrightarrow \exists t_1, t_2 \in [X] \text{ tales que } t_1^2 u = t_2^2 v). \tag{2.9}$$

Definición 22. El monomio $x^{\underline{a}} = \prod_{i=1}^{n} x_i^{a_i}$ se dice **monomio estándar** si $a_i < 2$ para cada $i \in \{1, \ldots, n\}$. Si $y \in \mathbb{F}_2^n$ decimos que $x^{\underline{a}}$ es la **representación estándar** de y si $x^{\underline{a}}$ es estandar y $\psi(x^{\underline{a}}) = y$.

2.2.1. El ideal asociado a un código binario

Consideremos el anillo de polinomios $A = \mathbb{K}[X]$. Fijemos un orden de términos $\prec$ cualquiera pero compatible con el grado total. Tomemos como $\mathbb{K}$ el cuerpo más pequeño en el cual hacer las operaciones, $\mathbb{F}_2$, pues los coeficientes no contienen ningún dato sobre el código ya que las palabras del mismo son exponentes.

Definición 23. Sea $\mathcal{C}$ un código lineal binario y $\mathcal{R}_\mathcal{C}$ la relación de equivalencia definida en **(2.7)**. Definimos el ideal asociado a $\mathcal{C}$ como

$$I(\mathcal{C}) = \langle \left\{ \underline{x^a} - \underline{x^b} : (\psi(\underline{x^a}), \psi(\underline{x^b})) \in \mathcal{R}_\mathcal{C} \right\} \rangle \subset A = \mathbb{F}_2[X]. \tag{2.10}$$

Es decir, el ideal binomial generado por las diferencias de los elementos en $[X]$ relacionados por $\mathcal{R}_\mathcal{C}$.

Consideremos ahora $w_1, \ldots, w_k$ los vectores fila de una matriz generadora de $\mathcal{C}$ y el ideal

$$I = \langle \left\{ \underline{x^{w_1}} - 1, \ldots, \underline{x^{w_k}} - 1 \right\} \cup \left\{ \underline{x}_i^2 - 1 : i = 1, \ldots, n \right\} \rangle \subset \mathbb{F}_2[X] \tag{2.11}$$

Queremos ver que $I = I(\mathcal{C})$. Intuitivamente nótese que el primer conjunto genera a todos los elementos de $I(\mathcal{C})$ mientras el segundo conjunto reduce los grados a 0 o 1 para mantener los vectores asociados en $\mathbb{F}_2^n$. Para demostrar el resultado necesitamos primero el siguiente lema.

Lema 4. *Sea $\mathcal{C}$ un código lineal binario generado como subespacio de $\mathbb{F}_2^n$ por $\{w_1, \ldots, w_k\}$ y sean $u, v \in [X]$. entonces las siguientes afirmaciones son equivalentes:*

1. *$\xi(u) = \xi(v)$, esto es, $\psi(u)$ y $\psi(v)$ tienen el mismo síndrome.*
2. *$u - v \in I(\mathcal{C})$*
3. *$\psi(u) - \psi(v) = \psi(u) - \psi(v) \in \mathcal{C}$*

4. *Existen* $t_1, t_2 \in [X]$ *tales que*

$$uvt_1^2 = t_2^2 \prod_{j=1}^{s} x^{w_{i_j}} \tag{2.12}$$

Demostración. 1) $\Leftrightarrow$ 2) viene de la definición de $I(\mathcal{C})$ y de 2.8. La igualdad en 3) se da siempre porque $\psi(u), \psi(v) \in \mathbb{F}_2^n$ y 1) $\Leftarrow$ 3) es claro pues $\psi(u)$ y $\psi(v)$ tienen el mismo síndrome $\Leftrightarrow$ $\psi(u) - \psi(v)$ tiene síndrome 0 $\Leftrightarrow$ $\psi(u) - \psi(v) \in \mathcal{C}$.

Ahora veamos 3) $\Rightarrow$ 4);
Primero note que como el espacio ambiente es binario, dados dos vectores, su suma y resta coinciden. $\psi(u) - \psi(v) \in \mathcal{C}$ implica que para algunos $w_{i_1}, \ldots, w_{i_s}$,

$$\psi(u) - \psi(v) = \sum_{j=1}^{s} w_{i_j} \tag{2.13}$$

Por **(2.9)** existen $t_1, t_2 \in [X]$ tales que

$$uvt_1^2 = t_2^2 \prod_{j=1}^{s} x^{w_{i_j}} \tag{2.14}$$

Recíprocamente,para ver 4) $\Rightarrow$ 3);
Note que $\psi(uvt_1^2) \in \mathcal{C}$ puesto que $\{w_1, \ldots, w_k\}$ forman una base para $\mathcal{C}$. Por lo tanto, $\psi(u) - \psi(v) = \psi(u) + \psi(v) \in \mathcal{C}$. □

Ahora ya podemos demostrar nuestro resultado.

Teorema 7. $I = I(\mathcal{C})$

Demostración. Es claro que $I \subset I(\mathcal{C})$ puesto que todos los binimios que generan al primero pertenecen al segundo. Para probar la contenencia opuesta, $I(\mathcal{C}) \subset I$

hay que ver que cualquier binomio $u - v \in I(\mathcal{C})$ pertenece a I. por el apartado 4 de la proposición anterior, existen $t_1, t_2 \in [X]$ tales que $uvt_1^2 = t_2^2 \prod_{j=1}^{s} \underline{x}^{\underline{w_{i_j}}}$, además, si $z_1 - 1, z_2 - 1 \in I$ entonces $z_1 z_2 - 1 = (z_1 - 1)(z_2 - 1) + (z_1 - 1) + (z_2 - 1) \in I$, por lo tanto

$$t_2^2 \prod_{j=1}^{s} \underline{x}^{\underline{w_{i_j}}} - 1 \in I \tag{2.15}$$

es decir, $uvt_1^2 - 1 \in I$.

Pero también tenemos que

$$uv - 1 = (uvt_1^2 - 1) - uv(t_1^2 - 1) \in I \tag{2.16}$$

de donde

$$u - v = v(uv - 1) - u(v^2 - 1) \in I \tag{2.17}$$

□

2.2.2. Una base del ideal asociado como conjunto de prueba

Sea G_T la base de Gröbner minimal reducida del ideal $I(\mathcal{C})$ con respecto al orden $\prec$. Note que G_T puede ser calculada con la propuesta que se hace en la sección "Cálculo de la base de Gröbner de ciertos ideales" de este capítulo a partir de los generadores $\left\{\underline{x}^{\underline{w_1}} - 1, \ldots, \underline{x}^{\underline{w_k}} - 1\right\} \cup \left\{x_i^2 - 1 : i \in \{1, \ldots, n\}\right\}$. Como ya dijimos en dicha sección, existen algunas ventajas computacionales en este caso. Como los binomios que generan el ideal son de la forma $x^{\underline{a}} - x^{\underline{b}}$ con

$x^{\underline{b}} \prec x^{\underline{a}}$ y $x^{\underline{a}}, x^{\underline{b}} \in [X]$ los coeficientes siempre son 1, así no hay crecimiento de los coeficientes, además, la mayor longitud de una palabra es n pues los binomios $x_i^2 - 1$ preveen otro caso, así las dos principales desventajas computacionales del cálculo de una base de Gröbner en general no son válidas en este caso. (Crecimiento de coeficientes y grado). Además claramente $I(\mathcal{C})$ es un ideal 0−dimensional lo que permite el cálculo de bases de Gröbner de $I(\mathcal{C})$ mediante álgebra lineal.

Los siguientes resultados nos permitiran definir un conjunto de prueba como en **(1.2)** a partir de la base de Gröbner reducida de $I(\mathcal{C})$.

Proposición 9. *Si I es un ideal binomial de $A = \mathbb{K}[X]$ entonces:*

1. *Los elementos de una base de Gröbner de I tambien son binomios.*
2. *La forma normal de un monomio es tambien un monomio.*

La demostración de esta proposición se puede ver en [15].

A pesar de que la reducción de Buchberger puede ser realizada con el mismo resultado, dada la naturaleza de este ideal introduciremos un nuevo método de reducción más eficiente para este caso.

Definición 24. La **reducción en un paso** ($\longrightarrow$) de un elemento $x^{\underline{a}}$ de $[X]$ usando G_T está definida como sigue

1. reducir $x^{\underline{a}}$ a su forma estandar $\underline{x^{a'}}$ usando las relaciones $x_i^2 \to 1$ para todo $x_i \in X$.
2. reducir $\underline{x^{a'}}$ con respecto de G_T por la reducción de base de Gröbner usual.

De la proposición 9 y el hecho de que $I(\mathcal{C})$ es binomial, se sigue que G_T es binomial, más aún, dado un elemento $x^{\underline{a}} \in [X]$, tenemos el siguiente resultado.

Teorema 8. *Sea $\mathcal{C}$ un código lineal binario con capacidad correctora $t = \lfloor \frac{d_H(\mathcal{C})-1}{2} \rfloor$, $I(\mathcal{C})$ su ideal asociado, G_T la base de Gröbner reducida de $I(\mathcal{C})$ con respecto a un orden monomial $\prec$ como antes y $\psi : X \to \mathbb{F}_2^n$ también como antes. Sea $x^{\underline{a}} \in [X]$, si $w_H(\psi(Red_{\prec}(x^{\underline{a}}, G_T))) \leq t$ entonces $\psi(Red_{\prec}(x^{\underline{a}}, G_T))$ es el vector error correspondiente a la palabra recibida $\psi(x^{\underline{a}})$, es decir, $\psi(x^{\underline{a}}) - \psi(Red_{\prec}(x^{\underline{a}}, G_T))$ es la palabra del codigo más cercana a $\psi(x^{\underline{a}})$. De lo contrario, si $w_H(\psi(Red_{\prec}(x^{\underline{a}}, G_T))) > t$ entonces, $\psi(x^{\underline{a}})$ contiene más de t errores.*

Demostración. Sea $\psi(x^{\underline{a}})$ un vector de $\mathbb{F}_2^n$ recibido a partir de una palabra $c \in \mathcal{C}$ en cuya transmisión se han cometido a lo más t errores, y sea e el vector de error, es decir, $\psi(x^{\underline{a}}) = c + e$ y $w_H(e) \leq t$. Por una parte tenemos que $\psi(x^{\underline{a}})H = eH$ donde H es una matriz de paridad de $\mathcal{C}$.

Además, si llamamos $\underline{x^e}$ a la representación estandar de e, tenemos que $w_H(e) = deg(\underline{x^e})$, luego, $deg(\underline{x^e}) \leq t$. Supongamos ahora que existe otro $x^{\underline{b}} \in [X]$ tal que $Td(x^{\underline{b}}) \leq t$ y $\psi(x^{\underline{b}})H = \psi(x^{\underline{b}})H$, esto implica que hay dos soluciones para el sistema lineal de peso a lo más t. Lo cual es una contradicción. Por lo tanto, $\underline{x^e}$ es el elemento minimal con respecto a $\prec$ que tiene el mismo síndrome que $\psi(x^{\underline{a}})$.

Finalmente, si $w_H(\psi(Red_{\prec}(x^{\underline{a}}, G_T))) > t$, entonces el argumento anterior implica que el peso del representante de menor peso de la clase de equivalencia de $\mathcal{R}_{\mathcal{C}}$ que contiene a $\psi(x^{\underline{a}})$ es mayor que t, es decir, $\psi(x^{\underline{a}})$ tiene más de t errores. □

Es decir que podemos descodificar un código lineal binario a partir de G_T. Veamos un ejemplo de esto.

Ejemplo 14. Consideremos $\mathcal{C} \in \mathbb{F}_2^7$ el código lineal binario de longitud 7 y dimensión 2. Una matriz generadora de $\mathcal{C}$ viene dada por:

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{2.18}$$

Como las filas de esta matriz generan $\mathcal{C}$ podemos construir $I(\mathcal{C})$ como en ***(2.11)***, es decir

$$I(\mathcal{C}) = \langle \underline{x}^{(1,0,1,0,1,1,0)} + 1, \underline{x}^{(0,1,1,0,1,0,1)} + 1, x_1^2 - 1, x_2^2 - 1, \\ x_3^2 - 1, x_4^2 - 1, x_5^2 - 1, x_6^2 - 1, x_7^2 - 1\rangle. \tag{2.19}$$

Una base de Gröbner de $I(\mathcal{C})$ con respecto al orden monomial lexicográfico graduado es:

$$G_T = \Big\{x_5x_3x_2 + x_7, x_5x_3x_1 + x_6, x_7^2 + 1, x_7x_6 + x_2x_1, x_7x_5 + x_3x_2, \\ x_7x_3 + x_5x_2, x_7x_2 + x_5x_3, x_7x_1 + x_6x_2, x_6^2 + 1, x_6x_5 + x_3x_1, x_6x_3 \\ + x_5x_1, x_6x_1 + x_5x_3, x_5^2 + 1, x_4^2 + 1, x_3^2 + 1, x_2^2 + 1, x_1^2 + 1]\Big\}. \tag{2.20}$$

Por resultado anterior el número de errores de una palabra $y \in \mathbb{F}_2^7$ recibida será el peso de la palabra asociada a la reducción por G_T de la forma estandar de y. Supongamos que recibimos el vector $(1,1,1,0,0,1,1)$, el monomio asociado (su forma estandar) es $x_1x_2x_3x_6x_7$. Si este monomio lo reducimos con respecto de G_T obtendremos x_3 cuyo vector asociado es $(0,0,1,0,0,0,0)$ que tiene por peso de Hamming 1. el factor de corrección del código t, es igual 1. Entonces, por el teorema anterior podremos correguir el error, más aún la palabra con más probabilidad de haber sido enviada será $(1,1,1,0,0,1,1)-(0,0,1,0,0,0,0) = (1,1,0,0,0,1,1)$.

Teorema 9. *La representación vectorial de los elementos de G_T forma un conjunto de prueba para $\mathcal{C}$.*

Demostración. Sean $\mathcal{C}$ un código lineal, $I(\mathcal{C})$ el ideal asociado a $\mathcal{C}$ y

$$G_T = \left\{\underline{x}^{g_1^+} - \underline{x}^{g_1^-}, \ldots, \underline{x}^{g_s^+} - \underline{x}^{g_s^-}\right\}$$

su base de Gröbner reducida. Supongamos que $y \in \mathbb{F}_2^n$ es un vector recibido tal que $y \notin D(0)$, es decir, que existe algún $w \in \mathcal{C}$ tal que $d_H(y,0) > d_H(y,w)$, o equivalentemente, que $w_H(y) > w_H(y-w)$.

Como $w \in \mathcal{C}$, entonces $\underline{x}^w + 1$ se puede expresar como combinación lineal de elementos en G_T,

$$\underline{x}^w = \sum_{i=1}^{s} f_i(\underline{x}^{g_i^+} - \underline{x}^{g_i^-}). \tag{2.21}$$

Por lo tanto, existe al menos un $i \in \{1,\ldots,s\}$ tal que $supp(g_i^+) \subset supp(w)$, es decir, $w_H(y) > w_H(y-w) \geq w_H(y-g_i^+)$, es decir, existe un elemento (g_i^+) en la representación vectorial de G_T que obliga a y a satisfacer la condición de conjunto de prueba. □

3 Aplicación a códigos no lineales

3.1. Cálculo de distancia y peso mínimo

En este capítulo notaremos por $\mathcal{C}$ a un código no lineal binario de longitud n y por $K_{\mathcal{C}}$ el núcleo del código, cuya matriz de paridad es H y $\{v_1,\dots,v_t\}$ son los vectores representantes de aquellas clases de equivalencia de $\mathbb{F}_2^n/K_{\mathcal{C}}$ que componen el código. Al ser lineal se puede considerar el ideal asociado a al núcleo viene dado por

$$I(K_{\mathcal{C}}) = \left\langle \underline{x}^{k_1} - \underline{x}^{k_2} : k_1 - k_2 \in K_{\mathcal{C}} \right\rangle. \tag{3.1}$$

Llamemos $B_{K_{\mathcal{C}}}$ a la base de Gröbner reducida de este ideal con respecto del orden deglex. (en general, podríamos hacer la misma construcción con cualquier orden compatible con el grado). Las siguientes proposiciones nos permitirán calcular el peso y la distancia mínima de $\mathcal{C}$.

Proposición 10. *Sean $\mathcal{C}, K_{\mathcal{C}}, \{v_1,\dots,v_t\}$ y $B_{K_{\mathcal{C}}}$ como arriba, se tiene que*

$$w_H(\mathcal{C}) = \min\{w_H(K_{\mathcal{C}}), w_H(\widehat{v_i}) : 1 \le i \le t\} \tag{3.2}$$

donde $x^{\widehat{v_i}}$ es la forma normalizada de x^{v_i} con respecto a $B_{K_{\mathcal{C}}}$ para cada $1 \le i \le t$.

Demostración.
En la proposición **(1)** ya se tiene que $w_H(\mathcal{C}) = \min\left\{w_H(K_{v_i}) : i \in \{0,1,\dots,t\}\right\}$ recordemos que $K_{v_i} = \langle K_{\mathcal{C}}, v_i\rangle$. Usando esto, para probar nuestro resultado basta ver que $\widehat{v_i}$ es un representante de peso mínimo de K_{v_i}.
En efecto, supongamos que $B_{K_{\mathcal{C}}} = \{g_1,\dots,g_s\}$ como $x^{\widehat{v_i}}$ es la forma normalizada

de x^{v_i} con respecto a $B_{K_\mathcal{C}}$, existen $f_1, \ldots, f_s$ tales que

$$x^{v_i} = \sum_{i=0}^{s} f_i g_i + x^{\widehat{v_i}} \tag{3.3}$$

y además $x^{\widehat{v_i}}$ es el monomio de menor grado tal que al sumarle la representación estándar de una palabra de $K_\mathcal{C}$ se obtiene x^{v_i}.
Sea ahora, $w_i \in K_{v_i}$ tal que $w_H(K_{v_i}) = w_H(w_i)$, puesto que $\widehat{v_i} \in K_{v_i}$ se tiene que $w_H(w_i) \leq w_H(\widehat{v_i})$ pero $w_i \in K_{v_i}$ implica que $w_i = c + v_i$ para algún $c \in K_\mathcal{C}$.
Luego, de la observación que sigue a **(3.3)** y el hecho de que $deg(x^{\widehat{v_i}}) = w_H(\widehat{v_i})$ se tiene que $w_H(w_i) = w_H(\widehat{v_i})$, es decir, $w_H(K_{v_i}) = w_H(\widehat{v_i})$. □

Esta proposición nos permite calcular el peso mínimo de un código no lineal, observe que para esto necesitamos del cálculo del peso mínimo de solo un código lineal, para las clases de equivalencia solo calculamos formas normales. Naturalmente el mayor costo computacional de este proceso estará en el cálculo de la base de Gröbner de $K_\mathcal{C}$, sin embargo esta se reutilizará para obtener la distancia mínima del núcleo pues $w_H(K_\mathcal{C})$ es simplemente es el peso de la palabra asociado a la primera elemento que se obtiene en el algoritmo en la sección "Cálculo de la base de Gröbner de ciertos ideales" del capítulo anterior, y para calcular la forma normal. La próxima proposición nos permite calcular la distancia mínima de $\mathcal{C}$.

Proposición 11. *Sean $\mathcal{C}, K_\mathcal{C}, \{v_1, \ldots, v_t\}$ y $B_{K_\mathcal{C}}$ como arriba, se tiene que*

$$d_H(\mathcal{C}) = \min\left\{ w_H(K_\mathcal{C}), w_H(\widehat{w_l}) : 1 \leq l \leq \frac{t(t-1)}{2} \right\} \tag{3.4}$$

donde $x^{\widehat{w_l}}$ es la forma normalizada de $x^{v_i+v_j}$ con respecto a $B_{K_\mathcal{C}}$ para cada $1 \leq i, j \leq t$ con $i \neq j$.

Demostración. De nuevo, según la proposición **(1.24)** tenemos que

$$d_H(\mathcal{C}) = \min\left\{w_H(K_{v_i+v_j}) : i \in \{0,1,\ldots,t-1\}, j \in \{i+1,\ldots,t\}\right\} \tag{3.5}$$

donde $K_{v_i+v_j} = \langle K_{\mathcal{C}}, v_i, v_j\rangle$. Así que igual que antes basta ver que $\widehat{w_l}$ es un representante de peso mínimo de $K_{v_i+v_j}$. Además también tenemos que si $B_{K_{\mathcal{C}}} = \{g_1,\ldots,g_s\}$ es la base de Gröbner reducida del ideal asociado a $K_{\mathcal{C}}$, como $x^{\widehat{w_l}}$ es la forma normalizada de $x^{v_i+v_j}$ con respecto a $B_{K_{\mathcal{C}}}$ siempre que $i \neq j$, existen $f_1,\ldots,f_s$ tales que

$$x^{v_i+v_j} = \sum_{i=0}^{s} f_i g_i + x^{\widehat{w_l}} \tag{3.6}$$

y además $x^{\widehat{w_l}}$ es el monomio de menor grado tal que al sumarle la representación estándar de una palabra de $K_{\mathcal{C}}$ se obtiene $x^{v_i+v_j}$.
Tomemos nuevamente $w_{i,j} \in K_{v_i+v_j}$ tal que $w_H(K_{v_i+v_j}) = w_H(w_{i,j})$, puesto que $\widehat{w_l} \in K_{v_i+v_j}$ se tiene que $w_H(w_{i,j}) \leq w_H(\widehat{w_l})$ pero $w_{i,j} \in K_{v_i+v_j}$ implica que $w_{i,j} = c + v_i + v_j$ para algún $c \in K_{\mathcal{C}}$.
Luego, de la observación que sigue a **(3.6)** y el hecho de que $deg(x^{\widehat{w_l}}) = w_H(\widehat{w_l})$ se tiene que $w_H(w_{i,j}) = w_H(\widehat{w_l})$. □

3.2. Aplicación a la descodificación

Finalmente, queremos extender la idea de descodificación por conjunto de prueba a códigos no lineales, con esta idea en mente y del método para hallar distancias mínimas de la proposición anterior, se tiene la siguiente proposición que nos permitirá descodificar

Proposición 12. *Sean $\mathcal{C}, K_{\mathcal{C}}, \{v_1,\ldots,v_t\}$ y $B_{K_{\mathcal{C}}}$ como arriba, y sea $y = c + e \notin \mathcal{C}$ un vector recibido con $c \in \mathcal{C}$. Llamamos $\underline{x}^{e_i}$ a la forma normalizada de $\underline{x}^{y+v_i}$ para cada $0 \leq i \leq t$ y consideramos $E = \{e_1,\ldots,e_t\}$. Sea $e_l \in E$ tal que $w_H(e_l) =$*

mín $\{w_H(e_i) : e_i \in E\}$, *si* $w_H(e) < d_H(K_{\mathcal{C}})$ *entonces se descodifica* y *en el código por la palabra* $c' = y - e_l$.

Demostración. Por la demostración de la proposición anterior (del cálculo de distancia mínima de un código no lineal) tenemos que e_i es un representante de menor peso del código lineal K_{y+v_i} para cada $0 \leq i \leq t$. además e_l es un vector de peso mínimo entre todos los e_i y por lo tanto un representante de peso mínimo para $\mathcal{C}_y = \bigcup_{i=0}^{t}(K_{y+v_i})$. Como $w_H(e) < d_H(K_{\mathcal{C}})$, por la proposición **(4)**, y se descodifica como $c' = y - e_l$. □

3.3. Algoritmos y complejidad

Las proposiciones **(10)**,**(11)** y **(12)**de este capítulo dan lugar a los siguientes algoritmos.

Algoritmo 5 Calcular el peso mínimo de un código no lineal 2

Entrada: $\mathcal{C}$-Código no lineal, $K_{\mathcal{C}}$-Kernel de $\mathcal{C}$, $B_{K_{\mathcal{C}}}$-Base de Gröbner reducida de $I(K_{\mathcal{C}})$, $L = \{v_1, \ldots, v_t\}$-Clases de $K_{\mathcal{C}}$ y $\prec$ - Orden monomial graduado

Salida: $w_H(\mathcal{C})$-Peso mínimo de $\mathcal{C}$

$w_H(\mathcal{C}) \leftarrow w_H(K_{\mathcal{C}})$

para $i \in [1, \ldots, t]$ **hacer**

$\underline{x}^{\widehat{v_i}} \leftarrow Red_{\prec}(\underline{x}^{v_i}, B_{K_{\mathcal{C}}})$

$w_H(\mathcal{C}) \leftarrow$ mín$(\deg(\underline{x}^{\widehat{v_i}}), w_H(\mathcal{C})))$

fin para

Cuya complejidad es del orden $\mathcal{O}(t \cdot n^2 \cdot |B_{K_{\mathcal{C}}}|)$ pues el cálculo de la reducción por una base de Gröbner en este caso es equivalente a la eliminación gaussiana. Note que esta es una cota superior de la complejidad que se podría afinar sin embargo ya es de orden polinomial en n.

Ejemplo 15 (Peso mínimo)**.** Retomemos el código $\mathcal{C}$ del ejemplo **(4)** para calcular de nuevo su peso, esta vez con nuestro algoritmo. D nuevo, $K_{\mathcal{C}}$ era el código lineal de peso mínimo 3 definido por la matriz de paridad,

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

y los representantes de las clases que forman $\mathcal{C}$ son

$$\begin{aligned} v_0 &= (0,0,0,0,0,0,0) \\ v_1 &= (1,1,0,1,1,1,0) \\ v_2 &= (1,0,1,1,0,0,0). \end{aligned}$$

El ideal asociado al núcleo es

$$\begin{aligned} I = \langle x_7x_6x_5x_4x_1 &+ 1, x_7x_5x_2+1, x_5x_4x_3+1, x_1^2+1, x_2^2+1, \\ x_3^2 &+ 1, x_4^2+1, x_5^2+1, x_6^2+1, x_7^2+1\rangle \end{aligned} \tag{3.7}$$

cuya base de Gröbner reducida es

$$\begin{aligned} B_{K_{\mathcal{C}}} = [x_7x_6x_5x_4x_2x_7 &+ x_4x_3x_5x_2x_1 + x_6x_3, x_4x_2x_1 + x_6, \\ x_3x_2x_1 &+ x_6x_5, x_6^2+1, x_6x_4 + x_2x_1, x_6x_2 + x_4x_1, \\ x_6x_1 &+ x_4x_2, x_5^2+1, x_6x_5x_4x_2x_1 + x_3x_2, x_5x_3 + x_4, \\ x_4^2 &+ 1, x_4x_3 + x_5, x_3^2+1, x_2^2+1, x_1^2+1]. \end{aligned} \tag{3.8}$$

Pues bien, ahora basta reducir $\underline{x}^{v_1} = x_1x_2x_4x_5x_6$ y $\underline{x}^{v_2} = x_1x_3x_4$ con respecto de $BK_{\mathcal{C}}$. Tenemos que $Red_{\prec}(\underline{x}^{v_1}, B_{\ker}) = x_3x_2$ y $Red_{\prec}(\underline{x}^{v_2}, B_{\ker}) = x_5x_1$. Luego $w_H(\mathcal{C}) = \min w_H(K_{\mathcal{C}}), deg(x_5x_2), deg(x_5x_1) = 2$.

Algoritmo 6 Calcular la distancia mínima de un código no lineal 2

Entrada: $\mathcal{C}$-Código no lineal, $K_{\mathcal{C}}$-Kernel de $\mathcal{C}$, $B_{K_{\mathcal{C}}}$-Base de Gröbner reducida de $I(K_{\mathcal{C}})$, $L = \{v_1, \ldots, v_t\}$-Clases de $K_{\mathcal{C}}$ y $\prec$ - Orden monomial graduado

Salida: $d_H(\mathcal{C})$-Distancia mínima de $\mathcal{C}$

```
d_H(C) ← w_H(K_C)
para i ∈ [1,...,t−1] hacer
    para j ∈ [i+1,...,t] hacer
        x^{w_l} ← Red_≺(x^{v_i+v_j}, B_{K_C})
        d_H(C) ← mín(deg(x^{w_l}), d_H(C)))
    fin para
fin para
```

Cuya complejidad es análoga, específicamente es del orden de $\mathcal{O}(\binom{t}{2} \cdot n^2 \cdot |B_{K_{\mathcal{C}}}|)$.

Algoritmo 7 Descodificar una palabra recibida en un código no lineal

Entrada: $\mathcal{C}$-Código no lineal, $K_{\mathcal{C}}$-Kernel de $\mathcal{C}$, $B_{K_{\mathcal{C}}}$-Base de Gröbner reducida de $I(K_{\mathcal{C}})$, $L = \{v_1, \ldots, v_t\}$-Clases de $K_{\mathcal{C}}$ $y = c + e \in \mathbb{F}_2^n \backslash \mathcal{C}$ - Una palabra recibida tal que $c \in \mathcal{C}$ y $\prec$ - Orden monomial graduado

Salida: $c' \in \mathcal{C}$ - la palabra de $\mathcal{C}$ con menos errores respecto a y

```
e ← n
para i ∈ [0,...,t] hacer
    x^{e_i} ← Red_≺(x^{y+v_i}, B_{K_C})
    si deg(x^{e_i}) < w_H(e) entonces
        e ← e_i
    fin si
fin para
c' ← y − e
```

Cuya complejidad también es polinomial en n.

Ejemplo 16 (Distancia mínima)**.** Queremos calcular ahora la distancia mínima del código $\mathcal{C}$ que hemos estado trabajando. Reutilizaremos la base de Gröbner reducida del ideal asociado a $K_{\mathcal{C}}$ que construimos en el ejemplo anterior **(15)**. De nuevo como tenemos sólo dos clases distintas del núcleo que forman el código, entonces según este algoritmo, la distancia mínima de $\mathcal{C}$ vendrá dada por el mínimo entre $w_H(K_{\mathcal{C}}) = 3$ y $deg(Red_{\prec}(\underline{x}^{v_1+v_2}, B_{K_{\mathcal{C}}})) = deg(x_4x_1) = 2$ es decir, 2.

Ejemplo 17. Llamemos por última vez al código no lineal $\mathcal{C}$ que hemos venido utilizando para los ejemplos, cuyo ideal asociado al núcleo está dado por

$$\begin{aligned} I = \langle x_7x_6x_5x_4x_1 &+ 1, x_7x_5x_2 + 1, x_5x_4x_3 + 1, x_1^2 + 1, x_2^2 + 1, \\ x_3^2 &+ 1, x_4^2 + 1, x_5^2 + 1, x_6^2 + 1, x_7^2 + 1\rangle. \end{aligned} \tag{3.9}$$

Y su respeciva base de Gröbner reducida es

$$\begin{aligned} B_{K_{\mathcal{C}}} = [x_7x_6x_5x_4x_2x_7 &+ x_4x_3x_5x_2x_1 + x_6x_3, x_4x_2x_1 + x_6, \\ x_3x_2x_1 &+ x_6x_5, x_6^2 + 1, x_6x_4 + x_2x_1, x_6x_2 + x_4x_1, \\ x_6x_1 &+ x_4x_2, x_5^2 + 1, x_6x_5x_4x_2x_1 + x_3x_2, x_5x_3 + x_4, \\ x_4^2 &+ 1, x_4x_3 + x_5, x_3^2 + 1, x_2^2 + 1, x_1^2 + 1] \end{aligned} \tag{3.10}$$

y los representantes de las clases que forman $\mathcal{C}$ son

$$\begin{aligned} v_0 &= (0,0,0,0,0,0,0) \\ v_1 &= (1,1,0,1,1,1,0) \\ v_2 &= (1,0,1,1,0,0,0). \end{aligned}$$

Supongamos que se recibe el vector $y = (1,1,0,1,1,1,1)$ que de hecho es el vector $v_1 + (0,0,0,0,0,0,1)$ queremos descodificar y en nuestro código no lineal.
En primer lugar calcularemos las formas normales de los monomios $\underline{x}^{v_i}$ para $0 \leq i \leq 2$ note que es importante incluir el caso $i = 0$ pues este se asegura de comprobar que la palabra a descodificar no pertenezca al núcleo del código.

Dado que $\underline{x}^{y+v_0} = x_1x_2x_4x_5x_6x_7$, $\underline{x}^{y+v_1} = x_7$ y $\underline{x}^{y+v_2} = x_2x_3x_5x_6x_7$, se sigue que

$$\begin{aligned} Red_{\prec}(\underline{x}^{y+v_0}) &= x_4x_3 \\ Red_{\prec}(\underline{x}^{y+v_1}) &= x_7 \\ Red_{\prec}(\underline{x}^{y+v_2}) &= x_5x_2. \end{aligned}$$

Luego, $deg(Red_{\prec}(\underline{x}^{y+v_1}) = x_7) = 1$ es el monomio de menor grado y por lo tanto el vector error es $e = (0,0,0,0,0,0,1)$ y en efecto la palabra recibida se corrige a v_1 como esperabamos.

Observación 9. Es evidente que este algoritmo proporciona un método de descodificación para códigos no lineales significativamente más eficiente que el proceso de descodificación de **(4)** originalmente presentado en [16, 19].

Observación 10. Como el lector atento habrá notado, la notable mejora en la eficiencia a la que se refiere la observación anterior, es consecuencia de que en este algoritmo los datos de entrada son mayores, además de lo que pide Zeng en su proposición, aquí pedimos una base de Gröbner reducida de un cierto ideal. Somos concientes de que el cálculo de tal base de Gröbner añade tanto coste computacional como para quedar al menos a la par del costo del proceso propuesto por Zeng. Sin embargo, queremos llamar la atención de nuevo sobre la observación (8) en la cual explicamos que el proceso de Zeng se inicia de cero con cada palabra recibida, mientras que para nosotros basta calcular una vez dicha base de Gröbner y luego dada cuaquier palabra, el coste de descodificarla será el coste reducido (polinomial en n).

Conclusión

En este trabajo se presentó una extensión de los conjuntos de prueba de códigos binarios lineales a los códigos binarios no lineales. Esto nos permite, mediante un preprocesado para el conjunto de prueba (típicamente por bases de Gröbner) conseguir un cálculo de la distancia mínima y de la descodificación más rápida que los resultados de Zeng F. [16, 19], polinomial en n.

Sin duda los resultados no son comparables, pues Zeng en su tesis no utiliza preprocesado, pero en el caso de la descodificación, sólo se realiza un preprocesado, lo que aventaja a su algoritmo que se ha de correr completamente para cada palabra que recibe.

Bibliografía

[1] Ashikhmin, A., Barg. A. (1998). Minimal Vectors in Linear Codes, IEEE vol 44, nº5 2010-2017.

[2] Berlekamp, E., McEliece, R., and Van Tilborg, H. (1978). On the inherent intractability of certain coding problems (Corresp.). IEEE Transactions on Information Theory, 24(3), 384-386.

[3] Borges-Quintana, M., Borges-Trenard, M. A., Fitzpatrick, P., and Martínez-Moro, E. (2008). Gröbner bases and combinatorics for binary codes. Applicable Algebra in Engineering, Communication and Computing, 19(5), 393-411.

[4] Brownawell, W (1987) Bounds for the degrees in the Nullstellensatz. Ann. Math. (2) 126(3), 577– 591.

[5] Buchberger, (1965) An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. Journal of Symbolic Computation, 41(3-4):475–511, 2006.

[6] Caboara M. , Traverso C. (1998) Efficient algorithms for module operation. Proc. ISSAC'98, ACM, p. 147–157.

[7] Cox, D., Little, J., and O'Shea, D. Ideals, Varieties, and Algorithms, second edition, Springer-Verlag, New York, 1996.

[8] J. C. Faugère, P. Gianni, D. Lazard, and T. Mora (1993) Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Comput.*, 16(4), p. 329–345.

[9] O. Heden (2008) On perfect p-ary codes of length $p+1$, Des. Codes Cryptogr. 46(1) 45–56.

[10] MacWilliams F., Sloane, N. (1997). The theory of error-correcting codes. North-Holland Mathematical Library, Vol. 16. North-Holland Publishing Co.

[11] Márquez Corbella, I. (2013). A Combinatorial Commutative Algebra Approach to Complete Decoding (Doctoral dissertation, Universidad de Valladolid)

[12] Márquez-Corbella, I., and Martínez-Moro, E. (2011). Algebraic structure of the minimal support codewords set of some linear codes. Adv. Math. Commun., 5(2), 233-244.

[13] The Sage Developers (2017). SageMath, the Sage Mathematics Software System (Version 7.6).

[14] Shannon, C. (1948). A Mathematical Theory of Communication. Bell System Technical Journal 27 (3): 379-423.

[15] Sturmfels, B. (1996). Gröbner bases and convex polytopes (Vol. 8). American Mathematical Soc.

[16] Villanueva, M.; Zeng, F.; Pujol, J. (2015). Efficient representation of binary nonlinear codes: constructions and minimum distance computation. Des. Codes Cryptogr. 76, no. 1, 3–21.

[17] Wang, Z., Karpovsky, M. G., and Kulikowski, K. J. (2009). Replacing linear Hamming codes by robust nonlinear codes results in a reliability improvement of memories. In Dependable Systems and Networks, 2009. DSN09. IEEE/IFIP International Conference on (pp. 514-523). IEEE.

[18] Westerbäck, T. (2016). Parity check systems of nonlinear codes over finite commutative Frobenius rings. arXiv preprint arXiv:1604.06996.

[19] Zeng, F. (2014). Nonlinear codes: representation, constructions, minimum distance computation and decoding (Doctoral dissertation, Universitat Autònoma de Barcelona)