Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Matemáticas

Códigos localmente recuperables

Autor: Jorge Hernández Sanz

Tutor: Diego Ruano Benito

Índice general

Introducción

Los códigos lineales constituyen una importante herramienta aplicada en numerosos campos en la actualidad, los datos que se almacenan diariamente no hacen más que aumentar y buscar un método eficiente y fiable se ha convertido en una prioridad para diferentes empresas y servicios.
Actualmente, complejos sistemas de almacenamiento distribuido o en la nube fraccionan y reparten grandes cantidades de información a lo largo de diferentes nodos o servidores. El trabajo de los códigos correctores consiste en almacenar toda esta información de la forma más segura y eficiente posible, de acuerdo a los requisitos que deba cumplir cada sistema. Una forma de asegurar la fiabilidad, aunque muy costosa, consiste en replicar la información de cada uno de los servidores a lo largo del resto, pero esto es, claramente ineficiente si observamos la proporción entre información almacenada y espacio de memoria total utilizado, lo que llamaremos tasa de almacenamiento.
Los sistemas actuales de almacenamiento de la información pueden llegar a almacenar incluso Exabytes (10^6 TB) de datos, con los grandes costes que ello conlleva, asociados a mantenimiento, abastecimiento energético, hardware, software... Por ello resulta conveniente obtener una buena tasa de almacenamiento.
Un buen método para obtener una buena tasa sin poner en riesgo la fiabilidad del sistema consiste en la utilización de lo que llamaremos codificación de borrado, en lugar del método de replicación de la información antes mencionado. Supongamos que se quiere almacenar un archivo, el cual fraccionamos en k partes distintas. Una codificación de borrado almacena un total de n partes a lo largo de n servidores distintos, donde k de ellas contienen la información original y las $n-k$ restantes información redundante, la cual cumplirá una función de control. Si se pretende optimizar la tasa de transmisión de la información, n/k, los códigos MDS, como por ejemplo los Reed-Solomon, resultan ser los más eficientes. La utilización de estos códigos garantiza recuperar de forma correcta la información original con tan solo acceder a k del total de n servidores del sistema, es decir, es capaz de asumir el fallo de hasta $n-k$ servidores sin suponer una pérdida real de información. En concreto el uso de los códigos Reed-Solomon, y los distintos códigos derivados de ellos, es muy extendido en la práctica. Por ejemplo, el código utilizado en el sistema de almacenamiento f4 Bloop de Microsoft Azure, basado en un código Reed-Solomon.
Una segunda consideración a tener en cuenta es la capacidad del sistema para asumir el fallo de un nodo o servidor en concreto, algo relativamente común en la práctica, ya sea porque está estropeado, no funciona correctamente o simplemente está en mantenimiento. La motivación principal de este trabajo está en diseñar códigos de borrado que no solo nos den una buena tasa de almacenamiento, sino que también permitan una recuperación rápida y sencilla de la información en caso de fallo en uno de los nodos del sistema.
Existen dos familias de códigos correctores que centran su atención en la optimización

del sistema, una de ellas en términos de reducir el ancho de banda utilizado para la recuperación de la información perdida de un servidor, y la otra reduciendo el número de servidores diferentes a los que es necesario acceder para recuperar esta información. Respectivamente estos son los códigos de regeneración (RGCs de las siglas en inglés *Re-Generating Codes*) y los códigos localmente recuperables (LRCs de las siglas en inglés *Locally Recoverable Codes*).
El foco de estudio de este trabajo serán estos últimos, los códigos localmente recuperables, cuyo principal propósito es el de reducir el grado de recuperación del sistema. Es interesante remarcar que el hecho de minimizar el grado de recuperación implica también una reducción en el ancho de banda, aunque esta no sea la prioridad de esta familia de códigos.

1. En el primer capítulo se da una breve introducción a los códigos correctores lineales, sus principales propiedades, definiciones y resultados necesarios para los desarrollos posteriores del trabajo. En particular se definen los códigos correctores Reed-Solomon, fundamentales para las construcciones que haremos en los capítulos siguientes.

2. El segundo capítulo expone las principales definiciones y resultados a cerca de los códigos localmente recuperables, sobre los que se centra el trabajo. Se define la propiedad que debe cumplir un código para ser considerado óptimo, en virtud a la cota enunciada en la segunda sección, resultado principal del capítulo. Se hace una distinción entre los símbolos de control y los símbolos de información, para el caso de una codificación sistemática. La última sección del capítulo relaja las condiciones exigidas y propone una familia de códigos alternativa, donde la localidad es la misma para todos los símbolos del código.

3. El tercer capítulo trata sobre diferentes familias de códigos localmente recuperables óptimos, propiedad que adquieren al alcanzar la igualdad en la cota dada en el capítulo previo. veremos también algunas construcciones para estos códigos, partiendo de los conocidos códigos Reed-Solomon. También de dan ejemplos para estos códigos y sus construcciones y propiedades. En la sección final se dan diferentes generalizaciones para la construcción que da el capítulo.

4. El cuarto capítulo da una variedad de los códigos localmente recuperables, los cuales permiten además detectar errores en los conjuntos de coordenadas de recuperación, propiedad que llamamos detección de errores locales. Se ven las principales ventajas y desventajas que ello conlleva, en términos de códigos. Un ejemplo en este capítulo ilustra estas propiedades y los métodos para la codificación y descodificación.

5. En este capítulo final se incluyen programas hechos sobre el software *SageMath*, que muestran con más detalle diferentes ejemplos y métodos tratados durante los capítulos anteriores.

Las principales fuentes de este trabajo han sido los siguientes artículos:
El artículo [1], como introducción a los códigos localmente recuperables y sus propiedades y resultados principales. El artículo [2], que profundiza su estudio en aquellas familias a las que llamaremos óptimas. Por último el artículo [3], el cual proporciona una variante con una propiedad interesante, la localización de errores, utilizando como base las herramientas anteriormente presentadas. La fuente [6] ha sido de utilidad para la contextualización de todos estos conceptos en el marco común de los códigos correctores

aplicados al almacenamiento distribuido.
Este trabajo es una memoria autocontenida, en la que se recogen los resultados principales de diferentes artículos de forma más explicada, bajo una notación unificada e incluyendo los prerrequisitos necesarios para su estudio. Además, se incluye la implementación propia de un programa de álgebra computacional de algunos de los ejemplos incluídos en el trabajo.

Capítulo 1

Preliminares

1.1. Códigos lineales

En este apartado recordaremos algunas definiciones y propiedades básicas sobre los códigos correctores lineales. Para una información más detallada se recomienda consultar [5]. Antes de introducir dichos códigos introducimos cómo denotaremos de ahora en adelante los cuerpos finitos sobre los que trabajaremos:

Notación 1.1.1. Denotamos por $\mathbb{F}_q$ al cuerpo finito de q elementos, donde q es de la forma p^n, con p un número primo.

Definición 1.1.2. Un código lineal $\mathcal{C}$ de parámetros $[n, k, d]$ es un subespacio vectorial, sobre $\mathbb{F}_q$, de dimensión k del espacio vectorial $\mathbb{F}_q^n$. En estas condiciones diremos que $\mathcal{C}$ es un código de dimensión k.

Definición 1.1.3. La distancia mínima del código $\mathcal{C}$, a la que denotaremos por $d(\mathcal{C})$, o simplemente d si no hay lugar a confusión, se define como:

$$d = d(\mathcal{C}) = min\{d(x, y) : x, y \in \mathcal{C}, x \neq y\}, \text{ donde:}$$

$$d(x, y) = \#\{i : 1 \leq i \leq, x_i \neq y_i\} \text{ es la distancia de Hamming.}$$

Se puede probar de forma sencilla que $d(x, y)$ define una distancia sobre $\mathbb{F}_q^n$.

Definición 1.1.4. Llamaremos peso de Hamming al entero:

$$w(\mathbf{x}) = \#sop(\mathbf{x}), \text{ donde } sop(\mathbf{x}) = \{i : 1 \leq i \leq, x_i \neq 0\}.$$

Este concepto nos proporciona una definición equivalente para la distancia mínima de un código:

Definición 1.1.5. Definimos el peso mínimo del código $\mathcal{C}$ como:

$$w(\mathcal{C}) = min\{w(\mathbf{c}) : \mathbf{c} \ \in \mathcal{C}, \mathbf{c} \neq 0\}.$$

Notemos que $d(\mathcal{C}) = w(\mathcal{C})$.

Definición 1.1.6. La redundancia de un código es la diferencia $n - k$. Se define también la tasa de transmisión de información como el cociente $\frac{k}{n}$.

De ahora en adelante se abreviará diciendo que $\mathcal{C}$ es un código de parámetros $[n, k, d]$, o simplemente $[n, k]$.
Por ser $\mathcal{C}$ un subespacio vectorial de $\mathbb{F}_q^n$, se puede ver como la imagen de una aplicación lineal e inyectiva:

$$f : \mathbb{F}_q^k \to \mathbb{F}_q^n, \text{ donde:}$$
$$Im(f) = \mathcal{C} \subset \mathbb{F}_q^n.$$

Llamaremos a f aplicación de codificación del código $\mathcal{C}$, la cual no es única. Esto introduce la siguiente definición:

Definición 1.1.7. llamaremos matriz generatriz de $\mathcal{C}$ a la matriz G asociada a la aplicación lineal e inyectiva f, aplicación de codificación de $\mathcal{C}$. Matriz de dimensiones $k \times n$ cuyas filas constituyen una base de $\mathcal{C} = Im(f)$.

De esta definición deducimos ciertas propiedades:
Puesto que $\mathcal{C}$ no admite una única base, la matriz generatriz G tampoco será única. Además, dadas dos matrices generatrices de $\mathcal{C}$, G_1 y G_2, existe otra matriz P inversible tal que $G_1 = P \cdot G_2 \cdot P^{-1}$, es decir, G_1y G_2 son matrices semejantes.
G nos proporciona una codificación del código $\mathcal{C}$. Un mensaje $\mathbf{a} \in \mathbb{F}_q^k$ codifica por G como $\mathbf{a} \cdot G \in \mathcal{C} \subset \mathbb{F}_q^n$. Podemos entonces dar una definición del código $\mathcal{C}$ a partir de una matriz generatriz asociada:

$$\mathcal{C} = \{\mathbf{a} \cdot G; \mathbf{a} \in \mathbb{F}_q^k\}$$

Veamos un primer ejemplo:

Ejemplo 1.1.8. Sea $\mathcal{C}$ el código binario (sobre el cuerpo $\mathbb{F}_2$) dado por la siguiente matriz generatriz:

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

De las dimensiones de G deducimos que $\mathcal{C}$ es un código de parámetros $[4, 3]$, longitud 4 y dimensión 3, luego $\#\mathcal{C} = q^k = 2^3 = 8$, es decir, $\mathcal{C}$ es un código de 8 palabras.
Utilizando la matriz generatriz G podemos obtener las 8 palaras de $\mathcal{C}$, codificando los 8 elementos de $\mathbb{F}_q^k$:

$$(100) \cdot G = (0101) \in C$$
$$(010) \cdot G = (1010) \in C$$
$$(001) \cdot G = (1100) \in C$$
$$(110) \cdot G = (1111) \in C$$
$$(101) \cdot G = (1001) \in C$$
$$(011) \cdot G = (0110) \in C$$
$$(111) \cdot G = (0011) \in C$$
$$(000) \cdot G = (0000) \in C$$

En este ejemplo no es difícil calcular la distancia mínima de $\mathcal{C}$, que resulta ser $d = 2$.

1.1.1. Codificación sistemática

Trataremos en esta sección un caso particular de codificación lineal, que verifica la siguiente propiedad:

Para cada $\mathbf{a} \in \mathbb{F}_q^k$ su codificación es de la forma $(\mathbf{a}, \mathbf{z}) \in \mathbb{F}_q^n$, donde $\mathbf{z} \in \mathbb{F}_q^{n-k}$.

En este tipo de codificación el mensaje codificado contiene en sus k primeras entradas el mensaje original $\mathbf{a}$, lo que hace automática la parte posterior de descodificación. Las $n-k$ últimas entradas, denotadas antes como vector $\mathbf{z}$, cumplen una función de control sobre la codificación.

Proposición 1.1.9. *Una codificación es sistemática si y sólo si existe una matriz de codificación G asociada de la forma $G = (I_k, M)$, donde I_k denota la matriz cuadrada identidad de dimensión k, y M una matriz de dimensión $k \times (n-k)$. Esta forma de G se conoce como forma estándar o sistemática.*

Definición 1.1.10. Decimos que dos códigos $\mathcal{C}_1$ y $\mathcal{C}_2$ de la misma longitud n son equivalentes si existe una permutación σ del conjunto $\{1, \ldots, n\}$ de forma que $\mathcal{C}_2 = \{\sigma(c) : c \in \mathcal{C}_1\}$.

Es decir, dos códigos son equivalentes si sus palabras difieren en el orden de sus elementos según una permutación σ, o lo que es lo mismo, si reordenando las columnas de la matriz generatriz del primer código obtenemos una matriz generatriz del segundo.
Recíprocamente, dado un código $\mathcal{C}$ y una permutación σ del conjunto $\{1, \ldots, n\}$, el conjunto $\sigma(\mathcal{C}) = \{\sigma(c); c \in \mathcal{C}\}$ es un código equivalente a $\mathcal{C}$.

Proposición 1.1.11. *Todo código lineal es equivalente a un código sistemático.*

Demostración. Dado un código $\mathcal{C}$ de parámetros $[n, k]$, tomamos una matriz generatriz G, la cual sabemos que tiene rango k. Mediante una permutación σ de sus columnas, podemos colocar las k columnas linealmente independientes en las k primeras posiciones y, mediante una serie de operaciones elementales, obtener en la primera submatriz de dimensión $k \times k$ la matriz identidad, obteniendo así una matriz G' en forma estándar o sistemática. □

veamos con un ejemplo el procedimiento que plantea la demostración:

Ejemplo 1.1.12. Consideramos G, una matriz generatriz de cierto código binario $\mathcal{C}$:

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

Observamos que las 3 últimas columnas son linealmente independientes. Tomamos entonces la permutación $\sigma = (4, 3, 2, 1)$ y aplicada a G obtenemos la matriz:

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

Ahora, sumando a la primera fila la tercera, se obtiene la matriz G':

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

siendo esta última la matriz en forma estándar de un código lineal sistemático equivalente al primero, dado por G.

1.1.2. Matriz de control

Hemos definido un código $\mathcal{C}$, visto como subespacio vectorial de $\mathbb{F}_q^n$, dando un sistema de generadores, con lo que definíamos G. También podemos dar el código $\mathcal{C}$ en función a unas ecuaciones implícitas, lo que nos lleva a la siguiente definición:

Definición 1.1.13. H es una matriz de control (o de paridad) del código $\mathcal{C}$ si verifica:

$$\forall \mathbf{x} \in F_q^n, \mathbf{x} \in \mathcal{C} \Leftrightarrow H\mathbf{x}^t = \mathbf{0}$$

La dimensión de esta matriz H será $(n-k) \times n$, y su rango $n-k$.

Proposición 1.1.14. *Dado un código $\mathcal{C}$ con respectivas matrices G y H, generatrices y de control, se tiene que :*

$$G \cdot H^t = 0$$

Demostración. Resulta inmediato a partir de la definición de matriz de control. Puesto que para todo $\mathbf{x} \in \mathcal{C}$ se tiene que $H \cdot \mathbf{x}^t = \mathbf{0}$, esto implica que $\mathbf{x} \cdot H^t = \mathbf{0}^t$. Puesto que las filas de G constituyen una base de $\mathcal{C}$, estas son en particular elementos de $\mathcal{C}$, luego al hacer el producto G por H^t el resultado es 0. □

Se puede obtener H a partir de una matriz generatriz G del código en su forma estándar: Si $G = (I_k, M)$, entonces la matriz $H = (-M^t, I_{n-k})$ tiene el rango y las dimensiones adecuadas y verifica que $G \cdot H^t = 0$ como pedíamos.

Definición 1.1.15. Una matriz de control H está en forma estándar o sistemática si es de la forma:

$$H = (B, I_{n-k})$$

Ejemplo 1.1.16. Para el código lineal visto en el ejemplo 1.1.12, una matriz de control en forma estándar es:

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

A continuación damos dos resultados que relacionan la distancia mínima con la matriz de control:

Proposición 1.1.17. *Sea $\mathcal{C}$ un código lineal con matriz de control H y distancia mínima d. Se tiene que:*

$$d \geq r+1 \Leftrightarrow \textit{elegidas cualquier } r \textit{ columnas de } H \textit{ son linealmente independientes.}$$

Demostración. Para la implicación hacia la derecha probamos el contrarrecíproco. Supongamos H tiene r columnas linealmente dependientes, entonces $H \cdot \mathbf{x}^t = 0$ para un $\mathbf{x}$, cuyas coordenadas son precisamente los coeficientes de tal combinación lineal. Por lo tanto, $\mathbf{x} \in C$ con $w(\mathbf{x}) \leq r$, luego $d \leq r$ como queríamos ver.
Recíprocamente, si cualesquiera r columnas son linealmente independientes en H, entonces ningún vector $\mathbf{x}$ con peso $w(x) \leq r$ puede pertenecer a $\mathcal{C}$, y puesto que la distancia mínima es el menor de estos pesos, $d \geq r+1$. □

Corolario 1.1.18. *La distancia mínima de un código $\mathcal{C}$ con matriz de control H coincide con el menor cardinal de un conjunto de columnas linealmente dependientes de H.*

Vamos ahora con un ejemplo de código lineal $\mathcal{C}$ definido por una matriz de control H, y una aplicación de estos resultados:

Ejemplo 1.1.19. Sea $\mathcal{C}$ el código binario de matriz de control:

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

Se obtiene un código $\mathcal{C}$ asociado de parámetros $[7, 4]$. Se observa que las 7 columnas se corresponden con los 7 elementos no nulos de $\mathbb{F}_2^3$, luego todas las columnas son linealmente independientes dos a dos, y por el lema previo tenemos que $d \geq 3$. Basta entonces encontrar un elemento $\mathbf{x} \in \mathcal{C}$ de peso 3 para ver que se da la igualdad. Por ejemplo, el elemento $\mathbf{x} = (1101000)$ pertenece a $\mathcal{C}$ (pues $H \cdot \mathbf{x}^t = 0$) y $w(\mathbf{x}) = 3$.

1.1.3. Dualidad

Dado $\mathcal{C}$ un código lineal de matriz de control H asociada. Puesto que H es de rango máximo, $n - k$, podemos interpretar H como matriz generatriz de otro código lineal sobre el mismo cuerpo finito $\mathbb{F}_q$. A este código se le conoce como código dual de $\mathcal{C}$ y se denota por $\mathcal{C}^\perp$.
De la definición se sigue que si $\mathcal{C}$ tiene dimensión k, entonces $\mathcal{C}^\perp$ tendrá dimensión $n-k$. Si G es matriz generatriz de $\mathcal{C}$, entonces también será matriz de control para $\mathcal{C}^\perp$, pues $G \cdot H^t = 0 \Leftrightarrow H \cdot G^t = 0$. La notación para mentar al código dual coincide con la notación usual, para espacios vectoriales, del subespacio vectorial ortogonal. Recordamos la definición de subespacio vectorial ortogonal:

Definición 1.1.20. Dado S subespacio vectorial contenido en $\mathbb{F}_q^n$, se define el subespacio ortogonal a S como:

$$S^\perp = \{\mathbf{u} \in \mathbb{F}_q^n; \mathbf{u} \cdot \mathbf{v} = 0, \forall\ \mathbf{v} \in S\}$$

Además, si S es de dimensión k sobre $\mathbb{F}_q^n$, entonces $S^\perp$ tiene dimensión $n - k$. Es más, $(S^\perp)^\perp = S$, por ser el producto escalar una forma bilineal simétrica no degenerada.

Proposición 1.1.21. *Si $\mathcal{C}$ es un código lineal, entonces el código dual $\mathcal{C}^\perp$ es el subespacio ortogonal a $\mathcal{C}$.*

Demostración. Resulta inmediato de las definiciones conocidas. Dadas G y H matrices generatriz y de control de $\mathcal{C}$ respectivamente, de la igualdad $G \cdot H^t = 0$ se deduce de forma inmediata el resultado. Además, $rango(G) + rango(H) = dim(\mathcal{C}) + dim(\mathcal{C}^\perp) = n + (n - k) = n$. □

Cabe destacar la posibilidad de que $\mathcal{C} \cap \mathcal{C}^\perp \neq \varnothing$. Es más, existe un caso particular en el que $\mathcal{C} = \mathcal{C}^\perp$. A un código con esta propiedad lo llamaremos código autodual.

1.1.4. Descodificación de los Códigos Lineales

Sea $\mathcal{C}$ un código de parámetros $[n, k, d]$ sobre $\mathbb{F}_q$. Para el proceso de descodificación la distancia de Hamming definida en nuestro espacio vectorial tendrá un papel fundamental.

Definición 1.1.22. La capacidad correctora de un código $\mathcal{C}$ es el entero t:

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

donde $d = d(\mathcal{C})$ es la distancia mínima del código y $\lfloor\ \rfloor$ denota la parte entera.

Este concepto se basa en la distancia de Hamming y la separación mínima entre elementos del código. Veamos cómo funciona la descodificación:
Supongamos enviada una palabra $\mathbf{c} \in \mathcal{C}$ y recibido un vector $\mathbf{y} \in \mathbb{F}_q^n$. Sean $\mathbf{e} = \mathbf{y} - \mathbf{c}$ el error cometido y $w(\mathbf{e})$ el número de errores cometidos, es decir, el número de coordenadas en las que difieren $\mathbf{c}$ e $\mathbf{y}$.
En esencia, calcularemos la distancia del mensaje recibido $\mathbf{y}$ con respecto a todas las palabras $\mathbf{c}$ del código y descodificaremos por la más próxima, si es que esta existe. Existen entonces tres casos posibles:

1. Si $w(\mathbf{e}) \leq t$, es decir, si se han cometido a lo sumo t errores, entonces existe un único $\mathbf{c} \in \mathcal{C}$ con $d(\mathbf{c}, \mathbf{y}) = w(\mathbf{e})$, y la descodificación de $\mathbf{y}$ como $\mathbf{c}$ será correcta.
2. Si $t < w(\mathbf{e}) < d$, podemos notar que hay errores, pero no corregirlos en general.
3. Si $d \leq w(\mathbf{e})$ la descodificación será errónea, de forma general.

Debido a la estructura lineal de nuestro código este proceso resulta relativamente sencillo y económico computacionalmente.

1.2. Códigos Reed-Solomon

Antes de definir estos códigos daremos una cota superior para la distancia mínima de cualquier código.

Teorema 1.2.1. *Cota de Singleton:*
Sea $\mathcal{C}$ un código de parámetros $[n, k]$ con distancia mínima $d = d(\mathcal{C})$. Entonces:

$$d \leq n - k + 1$$

Demostración. Sea H una matriz de control de nuestro código $\mathcal{C}$, la cual sabemos tiene rango $n - k$, luego se tienen $n - k + 1$ columnas linealmente dependientes. Por tanto, el mínimo número de columnas linealmente dependientes (y en consecuencia la distancia mínima d) debe ser, a lo sumo, $n - k + 1$. □

Definición 1.2.2. Sean $x_1, \ldots, x_n$ n elementos distintos sobre el cuerpo finito $\mathbb{F}_q$. Para cierto $k \leq n$ consideramos $\mathbb{P}_k$, el conjunto de polinomios con coeficientes en $\mathbb{F}_q[x]$ de grado $\leq k - 1$. Las palabras de un código Reed-Solomon se definen como:

$$(f(x_1), f(x_2), \ldots, f(x_n)), \text{ donde } f \in \mathbb{P}_k.$$

Es claro que la longitud del código es $n \leq q$. Veamos que el código es efectivamente lineal. Sean $\mathbf{c}_1 = (f_1(x_1), \ldots, f_1(x_n)), \mathbf{c}_2 = (f_2(x_1), \ldots, f_2(x_n))$ dos palabras distintas del código, con $f_1, f_2 \in \mathbb{P}_k$. Entonces, para cualquier par $a, b \in \mathbb{F}_q$, se tiene que $a\mathbf{c}_1 + b\mathbf{c}_2 = (af_1(x_1) + bf_2(x_1), \ldots, af_1(x_n) + bf_2(x_n)) = ((af_1 + bf_2)(x_1), \ldots, (af_1 + bf_2)(x_n))$, donde $(af_1 + bf_2) \in \mathbb{P}_k$, luego $a\mathbf{c}_1 + b\mathbf{c}_2$ pertenece al código. Sabemos además que $\mathbb{P}_k$ tiene dimensión k como espacio vectorial, luego k será también la dimensión del código RS asociado.

Teorema 1.2.3. *La distancia mínima de un código $\mathcal{C}$ Reed-Solomon de parámetros $[n, k]$ es:*

$$d = n - k + 1.$$

Demostración. Razonemos por reducción al absurdo. Supongamos que $d < n - k + 1$. Teniendo en cuenta lo visto en la definición 1.1.15, existe $\mathbf{c} \in \mathcal{C}$, $\mathbf{c} \neq \mathbf{0}$, tal que $w(\mathbf{c}) < n-k+1$. Esto implica que $\mathbf{c}$ tiene como mucho $n-k$ entradas no nulas, o equivalentemente, un mínimo de k entradas nulas. Esto último añadido a la condición de $\mathbf{c} \neq \mathbf{0}$ implica necesariamente que un polinomio no nulo de grado menor o igual que $k-1$ tiene al menos k raíces, lo que entra en contradicción con el teorema fundamental del álgebra. □

Definición 1.2.4. Un código que verifique esta igualdad es lo que llamamos un código separable con distancia máxima, de forma abreviada como un código MDS, del inglés *Maximum Distance Separable*.

Veamos cómo funcionan estos códigos con un ejemplo:

Ejemplo 1.2.5. consideramos el código $\mathcal{C}$ Reed-Solomon de parámetros $[10, 5]$ sobre el cuerpo finito $\mathbb{F}_{11}$ (en general, cuando $n = q - 1$, tomaremos los n elementos de $\mathbb{F}_q^*$). Notemos que el 2 es un elemento primitivo de $\mathbb{F}_{11}$. Ordenamos entonces los 10 elementos de $\mathbb{F}_{11}^*$ como sigue:

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
1	2	4	8	16	32	64	128	256	512
1	2	4	8	5	10	9	7	3	6
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9

Dado un elemento $(a_0, a_1, a_2, a_3, a_4) \in \mathbb{F}_{11}^5$, este codifica como

$$(f_a(x_0), f_a(x_1), \ldots, f_a(x_9)) \in \mathcal{C},$$

donde $f_a(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \in \mathbb{P}_5$.
La codificación de una base de $\mathbb{F}_{11}^5$, por ejemplo la canónica, nos da una base de $\mathcal{C}$:

$$\begin{matrix}(1 & 0 & 0 & 0 & 0)\\ (0 & 1 & 0 & 0 & 0)\\ (0 & 0 & 1 & 0 & 0)\\ (0 & 0 & 0 & 1 & 0)\\ (0 & 0 & 0 & 0 & 1)\end{matrix}$$

codifica como:

$$\begin{matrix}(1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1)\\ (1 & 2 & 4 & 8 & 5 & 10 & 9 & 7 & 3 & 6)\\ (1 & 4 & 5 & 9 & 3 & 1 & 4 & 5 & 9 & 3)\\ (1 & 8 & 9 & 6 & 4 & 10 & 3 & 2 & 5 & 7)\\ (1 & 5 & 3 & 4 & 9 & 1 & 5 & 3 & 4 & 9)\end{matrix}$$

Estos cinco elementos de una base se pueden utilizar como las cinco filas de una matriz generatriz asociada a nuestro código:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 5 & 10 & 9 & 7 & 3 & 6 \\ 1 & 4 & 5 & 9 & 3 & 1 & 4 & 5 & 9 & 3 \\ 1 & 8 & 9 & 6 & 4 & 10 & 3 & 2 & 5 & 7 \\ 1 & 5 & 3 & 4 & 9 & 1 & 5 & 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 2^2 & \dots & 2^9 \\ 1 & 2^2 & 2^3 & \dots & 2^8 \\ 1 & 2^3 & 2^4 & \dots & 2^7 \\ 1 & 2^4 & 2^5 & \dots & 2^6 \end{pmatrix}$$

De forma general, la matriz de un código Reed-Solomon $[n, k]$ es de la forma:

$$G = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_0 & x_1 & \dots & x_{n-1} \\ \vdots & \vdots & \dots & \vdots \\ x_0^{k-1} & x_1^{k-1} & \dots & x_{n-1}^{k-1} \end{pmatrix}$$

Es más, si α es un elemento de orden n sobre $\mathbb{F}_q$, entonces podemos considerar:

$$G = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \alpha & \dots & \alpha^{n-1} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \alpha^{k-1} & \dots & \alpha^{(k-1)(n-1)} \end{pmatrix} \text{ y } H = \begin{pmatrix} 1 & \alpha & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & \dots & \alpha^{2(n-1)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \alpha^{n-k} & \dots & \alpha^{(n-k)(n-1)} \end{pmatrix}$$

Matrices, respectivamente, generatriz y de control del código asociadas.

Capítulo 2

Códigos localmente recuperables

La mayor parte de los resultados expuestos a lo largo de este capítulo se encuentran en el artículo [1].
Un código localmente recuperable, LRC de forma abreviada, es un código de corrección de errores donde todo borrón en una coordenada de la palabra del código puede ser recuperado utilizando sólo un conjunto de otras coordenadas, sin tener que acudir a la capacidad correctora global del código. Entendemos borrón como el fallo en una coordenada conocida de la palabra. Es decir, no sólo sabemos que hay un fallo, sino que también sabemos en que coordenada ha ocurrido. Esto puede ser por una codificación errónea o bien por que no tenemos acceso a esa parte de la información, algo relativamente común en la práctica, para sistemas de almacenamiento distribuido de la información, como se motivaba en la introducción.
Veremos el dilema que existe entre mantener una baja localidad para las coordenadas del código y la capacidad del mismo de corregir borrones, así como su relación con la distancia mínima.

2.1. Introducción

En términos de códigos, se considera un código lineal $\mathcal{C}$ de parámetros $[n, k, d]$ sobre un cuerpo finito $\mathbb{F}_q$. Supongamos que la codificación de una palabra $\mathbf{x} \in \mathbb{F}_q^k$ está dada por el vector

$$\mathcal{C}(\mathbf{x}) = (\mathbf{c}_1 \cdot \mathbf{x}, \ldots, \mathbf{c}_n \cdot \mathbf{x}) \in \mathbb{F}_q^n$$

$\mathcal{C}$ queda definido por el conjunto $\mathbf{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_n\} \subset \mathbb{F}_q^k$, conjunto el cual debe tener rango k. Recordemos que podemos determinar una matriz generatriz del código utilizando estos n vectores como columnas. Notemos la diferencia entre los dos términos, $\mathcal{C}$ y $\mathbf{C}$, donde el primero denota el código en sí, mientras que el segundo es un conjunto de vectores de $\mathbb{F}_q^k$ que definen el código.
Vamos con unas primeras definiciones y resultados:

Lema 2.1.1. *$d(\mathcal{C}) = d$ si y solo si para todo conjunto $S \subseteq \mathbf{C}$ con Rango a lo sumo $k-1$, se tiene que $|S| \leq n - d$.*

Demostración. Las técnicas de esta demostración exceden los propósitos del trabajo. El lector interesado en la demostración de este resultado puede encontrar la prueba en [9, Th. 1.1.6]. □

Definición 2.1.2. Para cada $\mathbf{c}_i \in \mathbf{C}$ se define la localidad de $\mathbf{c}_i$ como:

$$Loc(\mathbf{c}_i) = min\{r : \exists R \subseteq \mathbf{C} \text{ con } \#R = r \text{ tal que } \mathbf{c}_i = \sum_{j \in R} \lambda_j \mathbf{c}_j\}.$$

Definimos la localidad del código $\mathcal{C}$ como:

$$Loc(\mathcal{C}) = max_{i \in \{1,\cdots,n\}} Loc(\mathbf{c}_i).$$

Definición 2.1.3. Diremos que $\mathcal{C}$ tiene localización de la información r si existe $\mathbf{I} \subseteq \mathbf{C}$ de rango máximo tal que $Loc(\mathbf{c}) \leq r$, para todo $\mathbf{c} \in \mathbf{I}$.

Para un código $\mathcal{C}$ así definido podemos tomar una base de $\mathbb{F}_q^k$ como conjunto I y particionar $\mathbf{C}$ como $I = \{\mathbf{e}_1, \cdots, \mathbf{e}_k\}$, parte correspondiente a los símbolos o coordenadas de información, y $\mathbf{C} \backslash I = \{\mathbf{c}_{k+1}, \cdots, \mathbf{c}_n\}$, parte correspondiente a los símbolos o coordenadas de control. Definimos así un código en forma sistemática.
A continuación damos de nuevo los conceptos de localidad en términos de coordenadas, lo que será de utilidad más adelante.

Definición 2.1.4. Diremos que una coordenada $i \in \{1, \ldots, n\}$ es localmente recuperable con localidad r si existe un conjunto de recuperación $R \subseteq \{1, \ldots, n\}$ con $i \notin R, \#R = r$ tal que para cada $\mathbf{x} \in \mathcal{C}$, un borrón en la coordenada x_i de $\mathbf{x}$ se puede recuperar utilizando los elementos x_j, con $j \in R$.

Definición 2.1.5. Un código es localmente recuperable con localidad menor o igual que r si lo es cada coordenada. La localidad de $\mathcal{C}$ es el mínimo entero que verifica esta condición.

Definición 2.1.6. Diremos que $\mathcal{C}$ es un (r, d)-código si tiene localización de la información r y distancia mínima d.

Notemos que para todo código $\mathcal{C}$ el conjunto de las relaciones de dependencia lineal de a lo sumo $r + 1$ elementos de $\mathbf{C}$ definen un hipergrafo $H_r(V, E)$, donde $V = \{1, \cdots, n\}$ es el conjunto de vértices del grafo, y se puede establecer una correspondencia biyectiva con los n elementos de $\mathbf{C}$.
Por otro lado, cada arista del conjunto E se corresponde con un subconjunto $S \subseteq V$ que verifica que $|S| \leq r + 1$ y además

$$\sum_{i \in S} \lambda_i \cdot \mathbf{c}_i = 0, \text{ con } \lambda_i \neq 0 \text{ para todo } i \in S.$$

Equivalentemente, S es una arista de H si se corresponde con el soporte de una palabra de $\mathcal{C}^{\perp}$ con peso de Hamming a lo sumo $r + 1$. Si ningún vértice de H es aislado entonces todas las coordenadas de $\mathcal{C}$ tienen localidad r. Es más, un código $\mathcal{C}$ tiene localización de la información r si el conjunto de puntos correspondientes a vértices contenidos en alguna arista de H tiene rango máximo.
Cuando el contexto no de lugar a confusión con el valor de r escribiremos de forma simplificada $H(V, E)$ o simplemente H.

Ejemplo 2.1.7. Un código $\mathcal{C}$ MDS de parámetros $[n, k, d]$ verifica que su localidad es k. Los códigos Reed-Solomon son un ejemplo: Mediante interpolación polinómica, k datos correctos son suficientes para recuperar la información perdida en un borrón.

Esta idea plantea algunos inconvenientes. Si durante el proceso de recuepración de x_i alguna de las coordenadas del conjunto R contiene algún error, la recuperación de x_i será también erronea y ambos errores quedarán sin detectar. Es más, estos nuevos errores podrían superar la capacidad correctora global del código $\mathcal{C}$ original y resultar así imposible su recuperación, incluso utilizando todo el codigo. Por ello resultará interesante utilizar conjuntos de recuperación que también permitan detectar errores locales en las coordenadas usadas durante el proceso de recuperación, como veremos en el apartado final de este trabajo, con los códigos localmente recuperables con detección de errores locales.

2.2. Cota inferior y teorema de estructura

En esta sección trabajaremos con códigos sistemáticos con localización de la información r. Dados k, r y d, nuestro objetivo es minimizar la longitud n. Dado que nuestro código es sistemático, esto es equivalente a minimizar la redundancia $h = n - k$. Daremos una cota inferior para este parámetro.

Teorema 2.2.1. *Para cualquier código lineal de parámetros $[n, k, d]_q$ con localización de la información r se tiene que:*

$$n - k \geq \left\lceil \frac{k}{r} \right\rceil + d - 2.$$

Demostración. Nuestra intención es construir un conjunto $S \subseteq \mathbf{C}$ con $Rank(S) \leq k - 1$ y aplicar el lema 2.1.1. El siguiente algoritmo sistematiza la obtención de este conjunto S:

1: $i \leftarrow 10$
2: $S_0 \leftarrow \{\}$
3: **while** $Rank(S_{i-1}) \leq k - 2$ **do**
4: Tomamos $\mathbf{c}_i \in \mathbf{C} \setminus S_{i-1}$ tal que existe una hiperarista $T_i \subset H$ con $\mathbf{c}_i \in T_i$
5: **if** $Rank(S_{i-1} \cup T_i) < k$ **then**
6: $S_i \leftarrow S_{i-1} \cup T_i$
7: **else**
8: Tomamos $T' \subset T_i$ tal que $Rank(S_{i-1} \cup T')$
9: $S_i \leftarrow S_{i-1} \cup T'$
10: **end if**
11: $i \leftarrow i + 1$
12: **end while**

Veámoslo en detalle:
El hecho de que $Rank(S_{i-1}) \leq k - 2$ y $Rank(I) = k$ garantiza la existencia de $\mathbf{c}_i$ como se pide. Ahora Sea l el número de veces que aumenta el cardinal del conjunto S_i. El conjunto final S_l verifica $Rank(S_l) = k - 1$, luego $k - 1 \leq |S|$. Definimos ahora s_i, t_i para medir el aumento del cardinal y del rango del conjunto S_i, respectivamente:

$$s_i = |S_i| - |S_{i-1}|$$

$$t_i = Rank(S_i) - Rank(S_{i-1}), \text{ donde } Rank(S_l) = \sum_{i=1}^{l} t_i = k - 1.$$

Analizamos dos situaciones posibles, en función de si la condición $Rank(S_{i-1} \cup T_i) = k$ se alcanza alguna vez o no. De hecho, la condición solo podría darse en el último paso, $i = l$.

- Caso 1:
  Supongamos que $Rank(S_{i-1} \cup T_i) \leq k-1$ de principio a fin. En cada paso añadimos $s_i \leq r+1$ vectores al conjunto. Notemos que dichos vectores siempre verifican que alguna combinación lineal no trivial nos da un vector (posiblemente nulo) del espacio vectorial $\langle S_{i-1} \rangle$. Por tanto, tenemos que:

  $$t_i \leq s_i - 1 \leq r, \text{ luego hay } l \geq \left\lceil \frac{k-1}{r} \right\rceil \text{ pasos, por tanto:}$$

  $$|S| = \sum_{i=1}^{l} s_i \geq \sum_{i=1}^{l} (t_i + 1) = k - 1 + l \geq k - 1 + \left\lceil \frac{k-1}{r} \right\rceil \geq k + \left\lceil \frac{k}{r} \right\rceil - 2$$

  donde la última desigualdad es una igualdad en el caso de que $r = 1$ o bien $k \equiv 1 \bmod r$.
  Por el lema 2.1.1 podemos afirmar que $n - d \geq |S|$, luego en virtud de la cadena de desigualdades recién probada:

  $$n - d \geq k + \left\lceil \frac{k}{r} \right\rceil - 2, \text{ como se quería demostrar.}$$

- Caso 2:
  Supongamos que $Rank(S_{l-1} \cup T_l) = k$. Como el rango de nuestro conjunto aumenta en a lo sumo r unidades en cada paso se tiene que $l \geq \left\lceil \frac{k}{r} \right\rceil$. Para $i \leq l-1$, añadimos un conjunto T_i de $s_i \leq r+1$ vectores. Notemos de nuevo el hecho de que dichos vectores siempre verifican que alguna combinación lineal no trivial nos da un vector (posiblemente nulo) del espacio vectorial $\langle S_{i-1} \rangle$. Por lo tanto, $Rank(S_i)$ aumenta en t_i, donde $t_i \leq s_i - 1$.
  En el paso l-ésimo añadimos a S el conjunto $T' \subset T_l$. Esto incrementa el rango de S en $t_l \geq 1$ unidades (pues $Rank(S) \leq k-2$ desde el principio) y aumenta el cardinal de S en $s_l \geq t_l$ unidades. Por tanto:

  $$|S| = \sum_{i=1}^{l} s_i \geq \sum_{i=1}^{l-1} (t_i + 1) - t_l \geq k + \left\lceil \frac{k}{r} \right\rceil - 2.$$

  De nuevo, teniendo en cuenta el lema 2.1.1, $|S| \leq n - d$, por tanto:

  $$n - d \geq k + \left\lceil \frac{k}{r} \right\rceil - 2, \text{ como se quería demostrar.}$$

□

Definición 2.2.2. Diremos que un (r, d)-código $\mathcal{C}$ es óptimo si sus parámetros verifican la igualdad en la cota del teorema anterior.

Los códigos pirámide son un ejemplo de (r, d)-códigos óptimos, para todos los valores r, d y k cuando q es suficientemente grande. Aprovechamos este momento para ilustrar con un ejemplo como son estos códigos pirámide.

Ejemplo 2.2.3. Utilizamos este ejemplo para mostrar un código pirámide LRC de parámetros $[n = 9, k = 6]$ y localidad $r = 3$. Partimos con la matriz generatriz G_{RS} de un código RS $\mathcal{C}_{RS}$ de parámetros $[8, 6]$, en forma sistemática,

$$G_{RS} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & g_{11} & g_{12} \\ 0 & 1 & 0 & 0 & 0 & 0 & g_{21} & g_{22} \\ 0 & 0 & 1 & 0 & 0 & 0 & g_{31} & g_{32} \\ 0 & 0 & 0 & 1 & 0 & 0 & g_{41} & g_{42} \\ 0 & 0 & 0 & 0 & 1 & 0 & g_{51} & g_{52} \\ 0 & 0 & 0 & 0 & 0 & 1 & g_{61} & g_{62} \end{pmatrix}$$

Obtenemos una matriz generatriz del código pirámide asociado dividiendo una de las $n - k = 2$ columnas de control de la matriz original y reordenando las columnas como sigue:

$$G_{Pyr} = \begin{pmatrix} 1 & 0 & 0 & g_{11} & 0 & 0 & 0 & 0 & g_{12} \\ 0 & 1 & 0 & g_{21} & 0 & 0 & 0 & 0 & g_{22} \\ 0 & 0 & 1 & g_{31} & 0 & 0 & 0 & 0 & g_{32} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & g_{41} & g_{42} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & g_{51} & g_{52} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & g_{61} & g_{62} \end{pmatrix}$$

Obtenemos como resultado la matriz generatriz de un código LRC de parámetros $[n = 9, k = 6, r = 3]$ óptimo, respecto a la cota tipo Singleton del teorema 2.2.1, veámoslo:
Por un lado tenemos que

$$d_{RS} = n_{RS} - k_{RS} + 1 = 8 - 6 + 1 = 3,$$

pues los códigos RS son un caso particular de código MDS y por ello alcanzan la cota de Singleton. Además, la distancia mínima d_{Pyr} es al menos d_{RS}, pues la distancia mínima de un código lineal es precisamente el mínimo peso de Hamming, como se definía en el primer capítulo, luego

$$d_{Pyr} \geq d_{RS} = 3.$$

Por otro lado, se verifica la desigualdad de tipo Singleton del teorema 2.2.1, luego

$$d_{Pyr} \leq n - k + 1 - \left(\left\lceil \frac{k}{r} \right\rceil - 1 \right) = 9 - 6 + 1 - \left(\left\lceil \frac{6}{3} \right\rceil - 1 \right) = 3,$$

luego la distancia alcanza la cota y podemos afirmar que es óptimo.
De forma general, dividiendo una de las columnas de control de la matriz original y reordenándolas como en este ejemplo, podemos construir a partir de un código RS de parámetros $[n, k]$ un nuevo código pirámide $\mathcal{C}_{Pyr}$ con localización en la información r de parámetros

$$[n_{Pyr} = n + \left\lceil \frac{k}{r} \right\rceil - 1, k_{Pyr} = k].$$

Para una información más detallada sobre este tipo de códigos correctores se recomienda consultar [4].

La prueba del teorema 2.2.1 nos da información acerca de cómo es la estructura de los (r, d)-códigos óptimos. Podemos pensar en el algoritmo como un proceso de maximizar la fracción

$$\frac{|S|}{Rank(s)} = \frac{\sum_{i=1}^{l} s_i}{\sum_{i=1}^{l} t_i}$$

Con esta idea en mente, en el paso i-ésimo podemos elegir $\mathbf{c}_i$ de manera que s_i/t_i esté maximizado, es decir, introducir el mayor número de vectores aumentando el rango lo menos posible. Una longitud óptima para este código nos debería dar el mismo valor de $|S|$ para esta o para cualquier elección de $\mathbf{c}_i$. Esta observación nos da una idea de la dependencia local en los códigos óptimos, como la dada en el siguiente teorema de estructura:

Teorema 2.2.4. *Sea $\mathcal{C}$ un código de parámetros $[n, k, d]_q$ con localización de la información r. Supongamos que $r|k$, $r < k$ y*

$$n = k + \frac{k}{r} + d - 2$$

es decir, verifica la igualdad del teorema anterior.
Entonces las hiperaristas del hipergrafo $H(V, E)$ son disjuntas y cada una contiene exactamente $r + 1$ vértices.

Demostración. Usaremos el algoritmo utilizado en la demostración del teorema anterior para obtener un conjunto S y las secuencias $\{s_i\}$ y $\{t_i\}$. Estudiaremos de forma separada el caso $r = 1$.

- Supongamos $r = 1$:
  puesto que $t_i \leq r = 1$ para todo i, caemos en el primer caso, $Rank(s_{i-1} \cup T_i) \leq k-1$, para todo i. Combinando ahora la cadena de desigualdades obtenida en el teorema anterior, el lema 2.1.1 y la última condición impuesta por el enunciado del teorema se tiene que

  $$|S| = \sum_{i=1}^{l} s_i = \sum_{i=1}^{l} t_i + l = 2k - 2.$$

  Además, $\sum_{i=1}^{l} t_i = k - 1$, concluimos que $l = k - 1$ y $s_i = 2,\ t_i = 1$ para todo i. Notemos que estas dos últimas condiciones hacen imposible la existencia de aristas de cardinal igual a 1 (vértices aislados) y también la posibilidad de intersecciones no vacías de aristas.

- Supongamos $r > 1$:
  Puesto que $r|k$ se tiene que

  $$k - 1 + \left\lceil \frac{k-1}{r} \right\rceil > k + \left\lceil \frac{k}{r} \right\rceil - 2$$

  Es decir, la expresión de la izquierda es una cota inferior más precisa para $|S|$. Por consiguiente, estamos en el segundo caso de la demostración del teorema anterior,

$Rank(S_{l-1} \cup T_l) = k$. Como vimos en la demostración del teorema anterior vemos que:

$$|S| = \sum_{i=1}^{l} s_i \geq \sum_{i=1}^{l-1}(t_i + 1) - t_l \geq k + \lceil \frac{k}{r} \rceil - 2.$$

Combinando este resultado con el Lema 2.1.1 y las hipótesis del enunciado se tiene que

$$|S| = \sum_{i=1}^{l} s_i = \sum_{i=1}^{l} t_i + l - 1 = k + \frac{k}{r} - 2.$$

Observamos que $\sum_{i=1}^{l} t_i = k - 1$ y por tanto $l = \frac{k}{r}$. Esto junto con la restricción $t_i \leq r$ implica que $t_j = r - 1$ para algún $j \in \{1, \cdots, l\}$ y $t_i = r$ para $i \neq j$. Además podemos afirmar que $j = l$, veámoslo. Supongamos que $j < l$, entonces se tiene que $\sum_{i \leq l-1} t_i = k - r - 1$ y $t_l = r$, luego estaríamos en el primer caso en lugar del segundo.
Supongamos ahora que existe una arista T tal que $|T| \leq r$. Si se añade esta arista a S en el primer paso del algoritmo tendríamos $t_i \leq r - 1$. Ahora supongamos que $T_1 \cap T_2 \neq \emptyset$. Observamos que esto implica que $Rank(T_1 \cap T_2) < 2r$. Por tanto, si añadimos las aristas T_1 y T_2 a S se tiene que $t_1 + t_2 \leq 2r - 1$. Claramente estas dos condiciones nos llevan a contradicción en caso de que $l = \frac{k}{r} \geq 3$. De hecho, también lleva a contradicción para el caso $\frac{k}{r} = 2$, pues nos llevaría de nuevo al primer caso.

□

El teorema de estructura nos dice que cuando d es suficientemente pequeño, los (r, d)-códigos óptimos tienen una estructura bastante rígida. Formalizamos esto a continuación con la definición de código canónico.

Definición 2.2.5. Sea $\mathcal{C}$ un código sistemático de parámetros $[n, k, d]_q$ con localización de la información r, donde $r|k$, $r < k$ y $n = k + \frac{k}{r} + d - 2$. Diremos que el código $\mathcal{C}$ es canónico si el conjunto $\mathbf{C}$ puede ser particionado en tres conjuntos $\mathbf{C} = I \cup \mathbf{C}' \cup \mathbf{C}''$ tales que:

1. $I = \{\mathbf{e}_1, \cdots, \mathbf{e}_k\}$.
2. $\mathbf{C}' = \{\mathbf{c}'_1, \cdots, \mathbf{c}'_{\frac{k}{r}}\}$, donde $w(\mathbf{c}'_i) = r$, para todo $i \in \{1, \cdots, \frac{k}{r}\}$. Además, los soportes de estos vectores son conjuntos disjuntos y constituyen una partición del conjunto $\{1, \cdots, k\}$.
3. $\mathbf{C}'' = \{\mathbf{c}''_1, \cdots, \mathbf{c}''_{d-2}\}$, donde $w(\mathbf{c}''_i) = k$ para todo $i \in \{1, \cdots, d-2\}$.

Claramente cualquier código canónico es en particular sistemático y tiene localización de la información r. La propiedad de la distancia requiere de la elección correcta de los vectores para los conjuntos $\mathbf{C}'$ y $\mathbf{C}''$. Los códigos pirámide son de nuevo un ejemplo. Notemos que, como $r < k$, hay siempre una distinción entre los elementos de $\mathbf{C}'$ y $\mathbf{C}''$.

Teorema 2.2.6. *Supongamos que $d < r + 3$, $r < k$, y $r|k$. Sea $n = k + \frac{k}{r} + d - 2$. En estas condiciones se tiene que todo código sistemático de parámetros $[n, k, d]_q$ con localización de la información r es un código canónico.*

Demostración. Sea $\mathcal{C}$ un código sistemático de parámetros $[n,k,d]_q$ y localización de la información r. Empezaremos viendo que el hipergrafo $H(V,E)$ tiene exactamente $\frac{k}{r}$ aristas.
Como $\mathcal{C}$ es sistemático se tiene que $I = \{\mathbf{e}_1, \cdots, \mathbf{e}_k\} \subset \mathbf{C}$. Por el teorema anterior sabemos que $H(V,E)$ tiene un total de m aristas regulares de cardinal $r+1$, y cada vértice asociado a un elemento de I está en una de estas aristas. Además, como los vectores de I son linealmente independientes, se tiene que cada arista contiene al menos un vértice de $\mathbf{C} \setminus I$, y como mucho r del conjunto I. Por tanto se tiene que $m \geq \frac{k}{r}$. Demostremos que se cumple la igualdad. Razonamos por reducción al absurdo, supongamos que $m \geq \frac{k}{r} + 1$. Como las aristas son regulares y disjuntas, se tiene que:

$$n \geq m(r+1) \geq k + \frac{k}{r} + r + 1 > k + \frac{k}{r} + d - 2.$$

Esto entra en contradicción con la condición inicial $n = k + \frac{k}{r} + d - 2$, por lo tanto $m = \frac{k}{r}$. Esto quiere decir que cada arista T_i contiene exactamente r vértices $\mathbf{e}_{i_1}, \cdots, \mathbf{e}_{i_r} \in I$, y un vértice $\mathbf{c}_i \in \mathbf{C} \setminus I$, luego:

$$\mathbf{c}_i = \sum_{j=1}^{r} \lambda_{i_j} \cdot \mathbf{e}_{i_j}.$$

Como además los conjuntos o aristas T_i son disjuntos dos a dos, los vectores $\mathbf{c}_1, \cdots, \mathbf{c}_{\frac{k}{r}}$ tienen soportes disjuntos formando una partición del conjunto $\{1, \cdots, k\}$.
Para terminar con la demostración veamos que los vectores restantes $\mathbf{c}_1^{''}, \cdots, \mathbf{c}_{d-2}^{''}$ cumplen que $w(\mathbf{c}_i^{''}) = k$. Para ello consideremos la codificación de los vectores $\mathbf{e}_j$. Notemos que $\mathbf{e}_i \cdot \mathbf{e}_j \neq 0$ si y solo si $i = j$, y $\mathbf{c}_i^{''} \cdot \mathbf{e}_j \neq 0$ si y solo si $j \in Sop(\mathbf{c}_i^{''})$. Por consiguiente, sólo dos de estos productos puede ser no nulo. Como la distancia mínima del código es d, todos los $d-2$ productos $\mathbf{c}_i^{''} \cdot \mathbf{e}_j$, con $i \in \{1, \cdots, d-2\}$ son no nulos, para todo $j \in \{1, \cdots, k\}$, por lo tanto $w(c_i^{''}) = k$ para todo $i \in \{1, \cdots, d-2\}$ como queríamos ver. □

La cota que se obtiene como resultado es lo suficientemente buena como para distinguir el caso particular en el que sólo se pide localidad r en los símbolos de información del código del caso más general en el que se pide localidad r para todos los símbolos.
El siguiente corolario se sigue del hecho de que el hipergrafo $H(V,E)$ debe contener $n - \frac{k}{r}(r+1) = d-2$ vértices aislados que no forman parte de ninguna relación lineal que involucre $r+1$ elementos.

Corolario 2.2.7. *Supongamos que $2 < d < r+3$, y $r|k$. Sea $n = k + \frac{k}{r} + d - 2$. En estas condiciones no existen códigos lineales de parámetros $[n,k,d]$ con localidad r.*

2.3. Códigos canónicos. Localidad de símbolos de control

El anterior teorema nos da una idea aproximada a cerca de los (r,d)-códigos óptimos en el caso particular en el que $d < r+3$ y $r|k$.
Para un código $\mathcal{C}$, el conjunto de coordenadas $\mathbf{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_n\}$ puede ser particionado en conjuntos $I, \mathbf{C}^{'}, \mathbf{C}^{''}$, donde para cada elemento $\mathbf{c} \in I \cup \mathbf{C}^{'}$ tengamos $Loc(\mathbf{c}) = r$, y para cada elemento $\mathbf{c}^{''} \in \mathbf{C}^{''}$ tengamos $Loc(\mathbf{c}^{''}) > r$. Resulta natural preguntarse cómo de baja pude ser la localidad de los símbolos de $\mathbf{C}^{''}$. En esta sección abordaremos esta pregunta.

2.3.1. Localidad para símbolos de control. Cota inferior

Teorema 2.3.1. *Sea $\mathcal{C}$ un (r,d)-código sistemático de parámetros $[n,k,d]_q$. Supongamos $d < r+3$, $r < k$ y $r|k$. Entonces existen $\frac{k}{r}$ símbolos de control de $\mathcal{C}$ con localidad r, y los $d-2$ símbolos de control restantes de $\mathcal{C}$ tienen localidad mayor o igual que $k-(\frac{k}{r}-1)(d-3)$.*

Demostración. El teorema 2.2.6 garantiza que un código $\mathcal{C}$ con estas características es canónico. Sea $\mathbf{C} = I \cup \mathbf{C}' \cup \mathbf{C}''$ la partición canónica de las coordenadas de $\mathcal{C}$. Claramente, para $\frac{k}{r}$ cualesquiera símbolos $\mathbf{c}' \in \mathbf{C}'$ se tiene localidad a lo sumo r. Ahora daremos cotas inferiores para los símbolos de $\mathbf{C}' \cup \mathbf{C}''$.

- Empezamos con símbolos $\mathbf{c}'' \in \mathbf{C}''$:
  Para todo $j \in \{1, \cdots, \frac{k}{r}\}$ definimos el subconjunto $\mathcal{R}_j \subseteq \mathbf{C}$ al que llamaremos *fila*. Sea $S_j = Sop(\mathbf{c}'_j)$. La fila j-ésima contiene al vector $\mathbf{c}'_j$, los r vectores unidad en el soporte de $\mathbf{c}'_j$ y al conjunto $\mathbf{C}''$,

$$\mathcal{R}_j = \{\mathbf{c}'_j\} \cup \left(\bigcup_{i \in S_j} \mathbf{e}_i \right) \cup \mathbf{C}''.$$

  Observamos que, si nos restringimos a $I \cup \mathbf{C}'$, las filas $\{\mathcal{R}_j\}_{j \in \{1,\cdots,\frac{k}{r}\}}$ forman una partición de $\{1, \cdots, n\}$. Consideramos ahora un símbolo arbitrario $\mathbf{c}'' \in \mathbf{C}''$ y sea $l = Loc(\mathbf{c}'')$. Entonces se tiene que:

$$\mathbf{c}'' = \sum_{i \in L} \mathbf{c}_i, \tag{2.1}$$

  donde $|L| = l$. A continuación, veremos que para cada fila $\mathcal{R}_j$ se tiene que

$$|L \cap \mathcal{R}_j| \geq r. \tag{2.2}$$

  Esta última condición ligada a la estructura de los conjuntos $\mathcal{R}_j$ nos lleva a la desigualdad que pide demostrar el teorema.
  Para probar la desigualdad (2.2) consideramos el código

$$\mathcal{C}_j = \{\mathcal{C}(\mathbf{x}) : \mathbf{x} \in \mathbb{F}_q^k \text{ tal que } Sop(\mathbf{x}) \subseteq S_j\}. \tag{2.3}$$

  Notemos que

$$Sop(\mathcal{C}_j) = \bigcup_{\mathbf{y} \in \mathcal{C}_j} Sop(\mathbf{y}) = \mathcal{R}_j, \text{ y } dim(\mathcal{C}_j) = r,$$

  pues las coordenadas de $\mathcal{C}$ fuera de $\mathcal{R}_j$ no dependen de las coordenadas de información en S_j. Observando que la distancia mínima del código $\mathcal{C}_j$ es al menos d, $d_{\mathcal{C}_j} \geq d$, y además $|\mathcal{R}_j| = r + d - 1$, se concluye que $\mathcal{C}_j$ es un código MDS si se restringe a su soporte. Veamos esto con más detalle:
  Recordamos que la cota Singleton afirma que

$$d_{\mathcal{C}_j} \leq n_{\mathcal{C}_j} - k_{\mathcal{C}_j} + 1 = |\mathcal{R}_j| - r + 1,$$

  luego, se tiene que $d_{\mathcal{C}_j} \leq r + d - 1 - r + 1 = d \Rightarrow d_{\mathcal{C}_j} \leq d$. Esto, junto con la desigualdad $d_{\mathcal{C}_j} \geq d$ vista anteriormente nos da la igualdad en la cota de Singleton, es decir, $\mathcal{C}_j$ es efectivamente un código MDS.

podemos entonces garantizar que para r símbolos cualesquiera de $\mathcal{C}_j$ se tiene que son independientes entre sí.
Queda remarcar que la igualdad en la ecuación (2.1) restringida a las coordenadas en S_j nos da una relación de dependencia lineal entre, a lo sumo, $|\mathcal{R}_j \cap L| + 1$ símbolos de $\mathcal{C}_j$.

- Veámoslo ahora para símbolos de $\mathbf{C}'$:
Elegimos ahora un elemento $\mathbf{c}_j' \in \mathbf{C}'$. Un razonamiento similar al anterior demuestra que $Loc(\mathbf{c}_j') < r$. Se tiene una relación de dependencia lineal para a lo sumo $r+1$ coordenadas del código $\mathcal{C}_j$ de parámetros $[r+d-1, r, d]_q$ definido antes, restringido a su soporte.

□

Podemos observar que la cota dada en este teorema es próxima a k sólo cuando r es grande y d pequeño, respectivamente. Para otros casos, el teorema no descarta la existencia de códigos canónicos con baja localidad para todos los símbolos del código, incluidos aquellos en $\mathbf{C}''$.
En la siguiente sección veremos códigos que verifican precisamente esto. En particular veremos que se puede garantizar la igualdad en la cota dada en el teorema anterior, bajo las condiciones adecuadas.

2.3.2. Localidad para símbolos de control. Cota superior

Los teoremas 2.3.5 y 2.3.6 serán los resultados principales de esta sección. El primero da una cota inferior general que coincide con la dada en el teorema anterior, el segundo nos da una familia concreta de códigos para el caso particular en el que $d = 4$. Introducimos ahora unos conceptos necesarios para la demostración del primero de estos teoremas:

Definición 2.3.2. Sea un subespacio lineal $L \subseteq \mathbb{F}_q^n$ y $S \subseteq \{1, \cdots, n\}$ un conjunto con $|S| = k$. En estas condiciones diremos que S es un k-núcleo de L si para todo $\mathbf{v} \in L$ se tiene que $sop(\mathbf{v}) \nsubseteq S$.

No es difícil notar que S es un k-núcleo para L si y solo si S es un subconjunto de algún conjunto de coordenadas de información en el espacio $L^\perp$. En otras palabras, S es un k-núcleo para L si y solo si k columnas de la matriz de dimensiones $(n - dim(L)) \times n$ generatriz de $L^\perp$ que se corresponden a los elementos de S son linealmente independientes.

Definición 2.3.3. Sea $L \subseteq \mathbb{F}_q^n$ un espacio lineal. Tomamos $\{\mathbf{c}_1, \cdots, \mathbf{c}_n\}$ un conjunto de n vectores de $\mathbb{F}_q^k$. Diremos que los vectores $\{\mathbf{c}_i\}$ están en posición general respecto a L si se cumplen las siguientes condiciones:

1. Para todo $\mathbf{v} \in L$ se tiene que $\sum_{i=1}^{n} \mathbf{v}_i \cdot \mathbf{c}_i = 0$.
2. Para todo S, k-núcleo para L, se tiene que $Rank(\{\mathbf{c}_i\}_{i \in S}) = k$.

Observamos que para cualquier L y para cualquier elección de $k = n - dim(L)$ columnas de una matriz de control de L de dimensión $k \times n$ se tiene que estas están en posición general respecto de S.
El siguiente lema afirma la existencia de vectores que están en posición general sujetos

a un espacio lineal arbitrario, siempre que el dominio subyacente sea lo suficientemente grande.

Lema 2.3.4. *Sea $L \subseteq \mathbb{F}_q^n$ un espacio lineal y k un número entero positivo. Supongamos que $q > k \cdot n^k$, entonces existe una familia de vectores $\{\boldsymbol{c}_i\}_{i\in\{1,\cdots,n\}}$ en $\mathbb{F}_q^k$ de forma que estos están en posición general respecto a L.*

Demostración. Construimos una matriz M de dimensiones $k \times n$, con coeficientes en $\mathbb{F}_q$, tomando cada fila de forma aleatoria (uniforme e independiente) del espacio lineal $L^\perp$. Tomamos las n columnas de M como los vectores $\{\mathbf{c}_i\}$. Observamos que la primera propiedad de la definición anterior se verifica siempre. Es más, vemos que nuestra elección de M induce una distribución uniforme en cada conjunto de k columnas de M que forman un k-núcleo.
La segunda condición de la definición se verifica siempre y cuando cada menor de orden $k \times k$ de la matriz M, que se corresponden con los k-núcleos, sean invertibles. Este caso se da con probabilidad al menos:

$$1 - \binom{n}{k}\left(1 - \prod_{i=1}^{k}(1 - \frac{1}{q^i})\right) = 1 - \binom{n}{k}\left(1 - (1 - \frac{1}{q})^k\right) \geq 1 - n^k \cdot \frac{k}{q} > 0,$$

pues $n^k \cdot \frac{k}{q} < 1$ por hipótesis. □

Vamos ahora con el resultado principal de esta sección.

Teorema 2.3.5. *Sea $2 < d < r + 3$, $r < k$ y $r|k$. Sea $q > k \cdot n^k$ una potencia prima. Sea $n = k + \frac{k}{r} + d - 2$, entonces existe un código $\mathcal{C}$ sistemático de parámetros $[n, k, d]_q$ con localización de la información r, donde $\frac{k}{r}$ símbolos de control tienen localidad r y los $d - 2$ restantes tienen localidad $k - (\frac{k}{r} - 1)(d - 3)$.*

Demostración. Sea $t = \frac{k}{r}$. Tomamos $t + 1$ subconjuntos de $\{1, \cdots, n\}$, $P_0, P_1, \cdots, P_t$ que verifiquen las siguientes propiedades:

1. $|P_0| = k - (t - 1)(d - 3) + 1$.
2. Para todo $i \in \{1, \cdots, t\}$ se tiene que $|P_i| = r + 1$.
3. Para todo $i, j \in \{1, \cdots, t\}$ con $i \neq j$ se tiene que $P_i \cap P_j = \emptyset$.
4. Para todo $i \in \{1, \cdots, t\}$ se tiene que $|P_0 \cap P_i| = r - d + 3$.

Para cada conjunto P_i, con $0 \leq i \leq t$, tomamos un vector $\mathbf{v}_i \in \mathbb{F}_q^n$ tal que $Sop(\mathbf{v}_i) = P_i$. Nos aseguramos que las entradas no nulas de $\mathbf{v}_0$ contengan los mismos valores. También nos aseguramos de que para todo $i \in \{1, \cdots, t\}$ las coordenadas no nulas de $\mathbf{v}_i$ contengan distintos valores. Notemos que la cota inferior dada para q garantiza que se puedan dar todas estas propiedades.
Para un conjunto finito A, denotamos por A^0 a un conjunto obtenido a partir de A al que se le ha quitado como mucho un elemento. Notemos que para todo $i \in \{1, \cdots, t\}$ y para todos $\alpha, \beta \in \mathbb{F}_q \setminus \{0\}$ se tiene que:

$$Sop(\alpha\mathbf{v}_0 + \beta\mathbf{v}_i) = (P_0 \setminus P_i) \sqcup (P_0 \cap P_i)^0 \sqcap (P_i \setminus P_0). \tag{2.4}$$

Consideramos el espacio vectorial $L = \langle \mathbf{v}_i : i \in \{0, 1, \cdots, t\}\rangle$, y sea $M = P_0 \setminus (\bigsqcup_{i=1}^{t} P_i)$. Se observa que

$$\#M = k - (t-1)(d-3) + 1 - t(r-d+3) = d-2.$$

Ahora bien, para cada $\mathbf{v} \in L$, de acuerdo con la ecuación (2.4), existe un conjunto $T \subset \{1, \cdots, t\}$ tal que, o bien tenemos

$$Sop(\mathbf{v}) = \bigsqcup_{i \in T} P_i,$$

o bien se tiene que

$$Sop(\mathbf{v}) = M \bigsqcup_{i \in \{1,\cdots,t\} \setminus T} (P_0 \cap P_i) \bigsqcup_{i \in T} (P_0 \cap P_i)^o \bigsqcup_{i \in T} (P_i \setminus P_0). \tag{2.5}$$

Observamos que dado un conjunto $K \subseteq \{1, \cdots, n\}$ con $|K| = k$, es un k-núcleo para L si y solo si para todo $i \in \{1, \cdots, t\}$ se tiene que $P_i \nsubseteq K$ y:

- o bien $M \nsubseteq K$
- o bien $M \subseteq K$ y existe $i \in \{1, \cdots, t\}$ tal que:
  - o bien $|P_i \cap P_0 \cap K| < r - d + 2$
  - o bien $|P_i \cap P_0 \cap K| = r - d + 2$ y $P_i \setminus P_0 \nsubseteq K$.

Sea $I \subseteq \{1, \cdots, n\}$ tal que $M \cap I = \emptyset$ y además para todo $i \in \{1, \cdots, t\}$ se tiene que $|I \cap P_i| = r$. Por lo visto justo antes, I es un k-núcleo para L.
Utilizando el lema previo, podemos obtener una familia de vectores $\mathbf{C} = \{\mathbf{c}_i\}_{i \in I}$, con $\mathbf{c}_i \in \mathbb{F}_q^k$, los cuales están en posición general respecto de el espacio L. Tomamos estos vectores como base del espacio vectorial $\mathbb{F}_q^k$ y consideramos el código $\mathcal{C}$ obtenido mediante el conjunto $\mathbf{C}$ como se indicaba al principio de este capítulo, es decir,

$$\mathcal{C} = \{(\mathbf{x} \cdot \mathbf{c}_1, \cdots, \mathbf{x} \cdot \mathbf{c}_n) \in \mathbb{F}_q^n : \mathbf{x} \in \mathbb{F}_q^k\} \tag{2.6}$$

Es casi inmediato notar que efectivamente $\mathcal{C}$ es un código sistemático con localización de la información r.
Para los t símbolos de control en el conjunto $(\bigsqcup_{i \in \{1,\cdots,t\}} P_i) \setminus I$ se tiene también localidad r. Es más, todos los $d-2$ símbolos de control en el conjunto M tienen localidad $k - (t-1)(d-3)$. Queda probar que el código $\mathcal{C}$ tiene distancia mínima

$$d = n - k - t + 2 \tag{2.7}$$

De acuerdo con el lema 2.1.1, $d = n - |S|$, donde $S \subseteq \{1, \cdots, n\}$ es el conjunto más grande que verifica que el conjunto de vectores $\{\mathbf{c}_i\}_{i \in S}$ no tiene rango máximo. De acuerdo con la definición 2.3.3, para cualquier K, k-núcleo de L, se tiene que $Rank(\{\mathbf{c}_i\}_{i \in K}) = k$, por tanto, para deducir la igualdad (3.4) basta con ver que todo conjunto $S \subseteq \{1, \cdots, n\}$ con $|S| = k + t - 1$ contiene un k-núcleo para L. Para probar esto haremos un análisis caso por caso.
Sea $S \subseteq \{1, \cdots, n\}$ con $|S| = k + t - 1$ un conjunto arbitrario. Definimos $b = \#\{i \in \{1, \cdots, n\} : P_i \subseteq S\}$. Puesto que $t(r+1) > |S|$, tenemos que $b \leq t - 1$. Separamos ahora en casos:

- Caso 1: $M \nsubseteq S$
  Quitamos $t-1$ elementos del conjunto S para obtener un conjunto $K \subseteq S$ con $|K| = k$, tal que para todo $i \in \{1, \cdots, t\}$, $P_i \nsubseteq K$. Teniendo en cuenta el razonamiento hecho tras la figura (2.5) se tiene que efectivamente K es un k-núcleo.
- Caso 2: $M \subseteq S$ y $b \leq t-2$
  Descartamos de nuevo $t-1$ elementos de S para obtener un conjunto $K \subseteq S$ con $|K| = k$ tal que $M \nsubseteq K$ y tal que para todo $i \in \{1, \cdots, t\}$, $P_i \nsubseteq K$. Teniendo en cuenta el razonamiento hecho tras la figura (2.5) se tiene que efectivamente K es un k-núcleo.
- Caso 3: $M \subseteq S$ y $b = t-1$
  Sea $i \in \{1, \cdots, t\}$ tal que $P_i \nsubseteq S$. Dicho i es único. Notemos que:

  $$|P_i \cap S| = k + t + 1 + (d-2) - (t-1)(r+1) = r - d + 2$$

  También se observa que $|P_i \setminus P_0| = r + 1 - (r - d + 3) = d - 2 \geq 1$. Combinando estas dos últimas observaciones se deduce que:

  - o bien $|P_i \cap P_0 \cap S| < r - d + 2$
  - o bien $|P_i \cap P_0 \cap S| < r - d + 2$ y $P_i \setminus P_0 \nsubseteq S$.

  Por último, descartamos $t-1$ elementos de S para obtener un conjunto $K \subseteq S$ con $|K| = k$ tal que para todo $i \in \{1, \cdots, t\}$ se tiene que $P_i \nsubseteq K$. Aplicando esto último junto con el razonamiento hecho tras la figura (2.5) se tiene que K es un k-núcleo.

□

Este teorema nos da una construcción general para (r, d)-códigos que no sólo son óptimos respecto a la localidad de la información y redundancia, sino también con respecto a la localidad de símbolos de control. Sin embargo, este resultado es débil en dos aspectos. En primer lugar la construcción no es explícita, y en segundo lugar requiere de un cuerpo subyacente relativamente grande.
El siguiente teorema da una construcción explícita que sirve también sobre cuerpos más pequeños, en el caso concreto para códigos con distancia mínima $d = 4$.

Teorema 2.3.6. *Sean r, k con $r < k$ y $r|k$, enteros positivos. Sea $q \geq r+2$ una potencia prima. Sea $n = k + \frac{k}{r} + 2$, entonces existe un código sistemático $\mathcal{C}$ de parámetros $[n, k, d]_q$ con localización de la información r, donde $\frac{k}{r}$ símbolos de control tienen localidad r y los 2 símbolos de control restantes tienen localidad $k - \frac{k}{r} + 1$.*

Demostración. Tomamos un código ε sistemático, arbitrario, de parámetros $[r+3, r, 4]_q$. Por ejemplo, se pude tomar ε como un Reed-Solomon. Consideramos la codificación:

$$\varepsilon(\mathbf{y}) = (\mathbf{y},\ \mathbf{p}_0 \cdot \mathbf{y},\ \mathbf{p}_1 \cdot \mathbf{y},\ \mathbf{p}_2 \cdot \mathbf{y}).$$

Puesto que ε es un código MDS, todos los vectores $\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2\}$ tienen peso r, por tanto, para ciertos coeficientes $\{\alpha_j\}_{j \in \{1, \cdots, r\}}$ no nulos, se tiene que:

$$\mathbf{p}_1 = \sum_{j=1}^{r-1} \alpha_j \cdot \mathbf{e}_j + \alpha_r \mathbf{p}_2, \tag{2.8}$$

donde $\{\mathbf{e}_j\}_{j\in\{1,\cdots,r\}}$ son vectores unidad r-dimensionales.
Para definir un código $\mathcal{C}$ sistemático, particionamos el vector de entrada $\mathbf{x} \in \mathbb{F}_q^k$ en $t = \frac{k}{r}$ vectores r-dimensionales $\mathbf{y}_1, \cdots, \mathbf{y}_t \in \mathbb{F}_q^r$. Definimos:

$$\mathcal{C}(\mathbf{x}) = \left(\mathbf{y}_1, \cdots, \mathbf{y}_r, \mathbf{p}_0 \cdot \mathbf{y}_1, \cdots, \mathbf{p}_0 \cdot \mathbf{y}_t, (\mathbf{p}_1 \sum_{i\in\{1,\cdots,t\}} \mathbf{y}_i), (\mathbf{p}_2 \sum_{i\in\{1,\cdots,t\}} \mathbf{y}_i) \right), \tag{2.9}$$

Se observa que las $k + t$ primeras coordenadas de $\mathcal{C}$ tienen localidad r. Veamos que las dos últimas coordenadas tiene localidad $k - t + 1$. De la ecuación (2.8) se tiene que

$$\left(\mathbf{p}_1 \sum_{i\in\{1,\cdots,t\}} \mathbf{y}_i \right) = \sum_{j=1}^{r-1} \sum_{i=1}^{t} \alpha_j \mathbf{y}_i(j) + \alpha_r \left(\mathbf{p}_2 \sum_{i\in\{1,\cdots,t\}} \mathbf{y}_i \right),$$

donde $\mathbf{y}_i(j)$ denota el elemento j-ésimo del vector $\mathbf{y}_i$.
Por tanto la penúltima coordenada de $\mathcal{C}$ puede recuperarse accediendo a $(r-1)t$ coordenadas de información y a la última coordenada. De forma similar, la última coordenada se puede recuperar de $k - t$ coordenadas de información y la penúltima coordenada.
Para ver que el código $\mathcal{C}$ tiene distancia 4 damos un algoritmo que corrige 3 borrones en $\mathcal{C}$. El algoritmo consta de dos pasos:

- Paso 1:
  Para todo $i \in \{1, \cdots, t\}$, nos referimos a un conjunto $(\mathbf{y}_i, \mathbf{p}_0 \cdot \mathbf{y}_i)$ de $r+1$ coordenadas de $\mathcal{C}$ como un único bloque. Recorreremos los t bloques, si encontramos un bloque donde hay un borrón en algún símbolo, lo recuperaremos a partir de los otros símbolos del bloque.

- Paso 2:
  Observemos que tras llevar a cabo el primer paso puede haber a lo sumo un bloque que tenga borrones. Si no existe dicho bloque, en el primer paso habremos recuperado satisfactoriamente todos los símbolos de información y habremos terminado.
  En caso contrario, supongamos que el único bloque con borrones es $(\mathbf{y}_j, \mathbf{p}_0 \cdot \mathbf{y}_j)$, para algún $j \in \{1, \cdots, t\}$. Puesto que conocemos todos los vectores $\mathbf{y}_i$ con $i \neq j, i \in \{1, \cdots, t\}$, a partir de los símbolos $(\mathbf{p}_1 \sum_{i\in\{1,\cdots,t\}} \mathbf{y}_i)$ y $(\mathbf{p}_2 \sum_{i\in\{1,\cdots,t\}} \mathbf{y}_i)$, si estos símbolos no tienen borrones, recuperamos los símbolos $\mathbf{p}_1 \cdot \mathbf{y}_j$ y $\mathbf{p}_1 \cdot \mathbf{y}_j$.
  Finalmente, aplicamos el proceso de descodificación del código ε para recuperar el vector $\mathbf{y}_j$ de a lo sumo 3 borrones en

$$\varepsilon(\mathbf{y}_j) = (\mathbf{y}_j, \mathbf{p}_0 \cdot \mathbf{y}_j, \mathbf{p}_1 \cdot \mathbf{y}_j, \mathbf{p}_2 \cdot \mathbf{y}_j).$$

□

2.4. Códigos no canónicos

En esta sección veremos que los códigos canónicos detallados en las dos secciones anteriores no son las únicas familias de (r, d)-códigos óptimos, respecto a la cota dada en el teorema 2.2.1 para la distancia mínima. Si se relajan las condiciones exigidas en el teorema 2.3.5 se pueden obtener otras familias. Veremos a continuación una de estas familias, la cual nos da localidad uniforme para todos los símbolos del código.

Teorema 2.4.1. *Sea n, k, r y $d \geq 2$ enteros positivos. Sea $q > kn^k$ una potencia prima. Supongamos que $(r+1)|n$ y*

$$n - k = \left\lceil \frac{k}{r} \right\rceil + d - 2.$$

Entonces existe un código de parámetros $[n, k, d]_q$ donde todo símbolo tiene localidad r.

Demostración. Sea $t = \frac{n}{r+1}$. Particionamos el conjunto $\{1, \cdots, n\}$ en t subconjuntos disjuntos $P_1, \cdots, P_t$, cada uno de tamaño $t+1$. Para cada $i \in \{1, \cdots, t\}$ tomamos un vector $\mathbf{v}_i \in \mathbb{F}_q^n$ tal que $Sop(\mathbf{v}_i) = P_i$. Ponemos ahora todas las coordenadas no nulas de los vectores $\mathbf{v}_i$ iguales a 1. Consideramos el espacio lineal $L = \langle \mathbf{v}_i : i \in \{1, \cdots, t\} \rangle$. Para cada $\mathbf{v} \in L$ se tiene que:

$$Sop(\mathbf{v}) = \bigsqcup_{i \in T} P_i, \text{ para algún } T \subseteq \{1, \cdots, t\}.$$

Observamos que el conjunto $K \subseteq \{1, \cdots, n\}$ con $|K| = k$, es un k-núcleo para L si y solo si para todo $i \in \{1, \cdots, t\}$ se tiene que $P_i \nsubseteq k$. De hecho, las condiciones del teorema implican que $k \leq n - t$, por tanto existen dichos k-núcleos para L.
Utilizamos el lema 2.3.4 para obtener los vectores $\{\mathbf{c}_i\}_{i \in \{1, \cdots, t\}} \in \mathbb{F}_q^k$ que están en posición general respecto al espacio L. Consideramos el código $\mathcal{C}$ definido del mismo modo que en (2.6).
Notamos que $\mathcal{C}$ tiene dimensión k y localidad r para cada símbolo. Resta probar que el código tiene distancia mínima

$$d = n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{2.10}$$

Nuestra prueba se basa en el lema 2.1.1. Sea $S \subseteq \{1, \cdots, n\}$ un conjunto arbitrario tal que el rango $Rank(\{\mathbf{c}_i\}_{i \in S}) < k$. Claramente S no contiene ningún k-núcleo para L. Consideramos:

$$b = \#\{i \in \{1, \cdots, t\} : P_i \subseteq S\}.$$

Tenemos que $|S| - b \leq k - 1$ descartando b elementos de S, uno para cada $P_i \subseteq S$, convirtiendo S en un $(|S| - b)$-núcleo. De hecho tenemos $br \leq k - 1$, puesto que descartando un elemento de cada $P_i \subseteq S$ obtenemos un br-código en S. Combinando las dos últimas desigualdades concluimos que

$$|S| \leq k - \left\lfloor \frac{k-1}{r} \right\rfloor - 1.$$

Combinando esta desigualdad con la identidad $\lfloor \frac{k-1}{r} \rfloor = \lceil \frac{k}{r} \rceil - 1$ y de nuevo el lema 3.1.1, obtenemos la igualdad pedida en (2.10):

$$d = n - k - \left\lceil \frac{k}{r} \right\rceil + 2.$$

□

Capítulo 3

Familia de códigos localmente recuperables óptimos

En este capítulo presentamos una familia de códigos localmente recuperables que, con los parámetros de localidad y longitud fijos, alcanzan el máximo valor posible para su distancia mínima, propiedad que debían tener los códigos a los que llamábamos óptimos en el capítulo anterior. Las palabras de estos códigos serán obtenidas como evaluaciones de polinomios específicos construidos sobre cuerpos finitos, y se reducirán a códigos Reed-Solomon cuando la localidad del código sea igual a la longitud.
El cardinal del alfabeto utilizado en el código para la mayoría de parámetros será ligeramente mayor que la longitud del código.
El proceso de recuperación consistirá en interpolación polinómica sobre r puntos, donde r es el grado de recuperación o localidad del código.
La fuente principal para el desarrollo de este capítulo ha sido el artículo [2].

3.1. Introducción

Mencionamos dos de las construcciones más conocidas y utilizadas para estos códigos óptimos, la primera de ellas se puede ver en detalle en [7]. En este artículo se propone una construcción a dos niveles basada en los códigos Gabidulin combinado con un $(r+1, r)$-código de control. La segunda construcción, detallada en [8] utiliza dos niveles de códigos MDS, uno de ellos Red-Solomon y un $(r+1, r)$-código, también MDS con otras particularidades. Un inconveniente bastante común de estas construcciones es el tamaño del alfabeto del código, el cual es una función exponencial de la longitud del código, lo que dificulta su implementación. En este capítulo enmendaremos este inconveniente, presentando una generalización natural de la construcción de los códigos Reed-Solomon sobre un alfabeto de cardinal comparable a la longitud del código n.
A continuación, repetimos unas definiciones dadas en capítulos previos, pero esta vez en términos distintos, lo cual facilitará el trabajo a lo largo del capítulo. Para cada $a \in \mathbb{F}_q$ consideramos los siguientes subconjuntos de $\mathcal{C}$:

$$\mathcal{C}(i, a) = \{\mathbf{x} \in \mathcal{C} : \mathbf{x}_i = a\}, \text{ para cada } i \in \{1, \cdots, n\}.$$

Definición 3.1.1. Diremos que $\mathcal{C}$ tiene localidad r si para cada $i \in \{1, \cdots, n\}$ existe un subconjunto $I_i \subset \{1, \cdots, n\} \setminus i$, con $|I_i| \leq r$ tal que las restricciones de los conjuntos

$\mathcal{C}(i,a)$ a las coordenadas de I, para diferentes valores de a, son disjuntos, es decir,

$$\mathcal{C}_{I_i}(i,a) \cap \mathcal{C}_{I_i}(i,a') = \emptyset,\ a \neq a',$$

donde la notación $\mathcal{C}_I$ denota la restricción del código $\mathcal{C}$ al conjunto de coordenadas $I \subset \{1, \cdots, n\}$. El código $\mathcal{C}_{I_i \cup \{1,\cdots,i\}}$ recibe el nombre de código local de $\mathcal{C}$.

Dos propiedades deseables a la hora de obtener códigos con localidad r fija son tener una distancia mínima relativamente grande con respecto a la longitud del código n y mantener la tasa de transmisión de información del código, $\frac{k}{n}$, lo más alta posible.

Teorema 3.1.2. *Sea $\mathcal{C}$ un código localmente recuperable de parámetros (n,k,r) sobre $\mathbb{F}_q$. Entonces la tasa de transmisión de información verifica que*

$$\frac{k}{n} \leq \frac{r}{r+1}.$$

Demostración. Las técnicas de esta demostración exceden los propósitos del trabajo. El lector interesado en la demostración de este resultado puede encontrar la prueba en apéndice del artículo [2]. □

3.2. Construcción del código

En esta sección construiremos un código lineal localmente recuperable óptimo (respecto a la cota del teorema 2.2.1, al igual que en el capítulo previo) de parámetros $[n,k,r]$ sobre un cuerpo finito $\mathbb{F}_q$, donde q es una potencia de un número primo, mayor o igual que n. En la primera versión que daremos para esta construcción supondremos que k es divisible por r. A lo largo de toda la sección supondremos que n es divisible por $r+1$.En la última sección veremos como estas condiciones se pueden relajar.

3.2.1. Construcción general

Comenzamos con un método general de construcción de códigos lineales localmente recuperables. Después veremos como algunos de estos códigos tienen distancia mínima óptima. Los códigos se construirán como evaluación de polinomios, en la misma línea que otras construcciones algebraicas. A diferencia de los conocidos Reed-Solomon, los códigos que construiremos serán evaluados en un conjunto específico de puntos del cuerpo $\mathbb{F}_q$, con $q > n$. El ingrediente principal para esta construcción será el polinomio $g(x) \in \mathbb{F}_q[x]$, el cual debe satisfacer las siguientes condiciones:

1. El grado de g es exactamente $r+1$.
2. Existe una partición $\mathcal{A} = \{A_1, \cdots, A_{\frac{n}{r+1}}\}$ de un conjunto $A \subseteq \mathbb{F}_q$ de tamaño n, en conjuntos de $r+1$ elementos, tal que el polinomio g es constante en cada conjunto A_i de la partición $\mathcal{A}$, es decir:

$$\text{Para todo } i \in \{1, \cdots, \frac{n}{r+1}\},\ \alpha, \beta \in A_i \text{ se tiene que } g(\alpha) = g(\beta).$$

Diremos que un polinomio es *bueno* si satisface estas dos condiciones. La siguiente construcción se basa en la existencia de estos polinomios.

Construcción 1: Códigos localmente recuperables de parámetros (n, k, r).
Sea $n \leq q$ la longitud del código deseada. Sea $\mathcal{A} \subset \mathbb{F}_q,\ |A| = n$ y sea $g(x)$ un polinomio *bueno* para la partición $\mathcal{A}$ del conjunto A. Para la codificación de un mensaje $\mathbf{a} \in \mathbb{F}_q^k$, escrito de la forma $\mathbf{a} = (a_{ij};\ i \in \{0, \cdots, r-1\},\ j \in \{0, \cdots, \frac{k}{r}-1\})$ definimos el polinomio codificador:

$$f_{\mathbf{a}}(x) = \sum_{i=0}^{r-1} f_i(x)x^i, \tag{3.1}$$

donde

$$f_i(x) = \sum_{j=0}^{\frac{k}{r}-1} a_{ij}g(x)^j, \text{ para cada } i \in \{0, \cdots, r-1\} \tag{3.2}$$

Llamamos a los polinomios f_i polinomios coeficiente. La palabra codificada para $\mathbf{a}$ se puede ver como una evaluación del polinomio $f_{\mathbf{a}}$ en cada uno de los puntos del conjunto A. En otras palabras, el código $\mathcal{C}$ localmente recuperable de parámetros (n, k, r) se define como el conjunto:

$$\mathcal{C} = \{(f_{\mathbf{a}}(\alpha),\ \alpha \in A : \mathbf{a} \in \mathbb{F}_q^k\} \subseteq \mathbb{F}_q^n. \tag{3.3}$$

Llamaremos normalmente localización a los elementos del conjunto A, y a los elementos del vector $(f_{\mathbf{a}}(\alpha))$ símbolos de la palabra codificada. La recuperación local se lleva a cabo de la siguiente forma:
Recuperación del borrón: Supongamos que el símbolo del borrón está asociado a la localización $\alpha \in A_j$, donde A_j es uno de los elementos de la partición $\mathcal{A}$. Sea $(c_\beta,\ \beta \in A_j \setminus \alpha)$, vector que denota los restantes r símbolos en las localizaciones del conjunto A_j. Para obtener el valor de $c_\alpha = f_{\mathbf{a}}(\alpha)$, calculamos el polinomio $\delta(x)$ de grado menor o igual que r tal que $\delta(\beta) = c_\beta$, para todo $\beta \in A_j \setminus \alpha$, es decir, mediante interpolación en estos puntos obtenemos

$$\delta(x) = \sum_{\beta \in A_j \setminus \alpha} c_\beta \prod_{\beta' \in A_j \setminus \{\alpha, \beta\}} \frac{x - \beta'}{\beta - \beta'}, \tag{3.4}$$

y sea $c_\alpha = \delta(\alpha)$. Llamamos a $\delta(x)$ el polinomio de decodificación de el símbolo c_α. Por lo tanto, para obtener el símbolo del borrón, es necesario realizar una interpolación polinómica de r símbolos conocidos en su conjunto de recuperación. Este procedimiento de recuperación se empleará en todos las construcciones de este capítulo. En el siguiente teorema veremos que estos códigos que acabamos de construir son óptimos con respecto a la cota dada en el teorema 2.2.1, y justifica la validez del proceso de recuperación.

Teorema 3.2.1. *El código lineal $\mathcal{C}$ definido en (3.3) tiene dimensión k y es un código localmente recuperable óptimo de parámetros $[n, k, r]$, es decir, alcanza la igualdad en la cota dada en el teorema 2.2.1.*

Demostración. Notemos que para $i \in \{0, \cdots, r-1\}$, $j \in \{0, \cdots, \frac{k}{r}-1\}$, los k polinomios $g(x)^j x^i$ tienen todos distinto grado, por lo tanto son linealmente independientes sobre $\mathbb{F}$. En otras palabras, la aplicación $a \mapsto f_a(x)$ es inyectiva. Por (3.1) y (3.2) deducimos que el grado del polinomio $f_a(x)$ es a lo sumo

$$\left(\frac{k}{r} - 1\right)(r+1) + r - 1 = k + \frac{k}{r} - 2 \leq n - 2,$$

donde la última desigualdad de sigue del teorema 3.1.2. Esto implica que dos polinomios de codificación distintos f_α y f_β nos dan dos codificaciones diferentes, luego la dimensión del código debe ser k. En efecto, la codificación es lineal y la distancia mínima satisface

$$d \geq n - max\{deg(f_{\mathbf{a}}) : \mathbf{a} \in \mathbb{F}_q^k\} = n - \frac{k}{r} + 2.$$

Este resultado, junto con la desigualdad del teorema 2.2.1 asegura que la distancia mínima es óptima, es decir, alcanza la igualdad como se pedía demostrar.
Veamos ahora la propiedad de localidad. Sea A_j un elemento de la partición $\mathcal{A}$ y supongamos que el símbolo asociado al borrón de nuestra palabra del código es $c_\alpha = f_a(\alpha)$, donde $\alpha \in A_j$ es un elemento del cuerpo. Definimos el polinomio de decodificación

$$\xi(x) = \sum_{i=0}^{r-1} f_i(\alpha), \tag{3.5}$$

donde los $f_i(x)$ son los coeficientes que definíamos en (3.2). Veremos que $\xi(x)$ es el mismo polinomio que $\delta(x)$ definido en (3.4). Cada $f_i(x)$ es una combinación lineal de potencias de g, por lo tanto es también constante en el conjunto A_j, es decir, para cualquier $\beta \in A_j$ y para cualquier coeficiente polinomial f_i con $i \in \{1, \cdots, r-1\}$ se tiene que

$$f_i(\beta) = f_i(\alpha). \tag{3.6}$$

Teniendo en cuenta los resultados (3.5) y (3.6) se tiene que, para todo β en A_j,

$$\xi(\beta) = \sum_{i=0}^{r-1} f_i(\alpha)\beta^i = \sum_{i=0}^{r-1} f_i(\beta)\beta^i = f_a(\beta).$$

En otras palabras, los valores del polinomio de codificación $f_a(x)$ y el polinomio de decodificación $\xi(x)$ coinciden en las localizaciones de A_j. Como el grado del polinomio $\xi(x)$ es a lo sumo $r-1$, podemos obtenerlo interpolando a partir de los r símbolos de c_β, donde $\beta \in A_j \backslash\, \alpha$, como en (3.4). Una vez calculado el polinomio $\xi(x)$ podemos obtener el símbolo perdido en el borrón calculando $\xi(\alpha)$. Es decir, para concluir con la demostración hemos probado que el símbolo que queríamos recuperar, c_α, se ha podido recuperar utilizando tan solo otros r símbolos de la palabra del código. □

Como consecuencia del teorema, vemos que el polinomio $\delta(x)$ satisface la condición $\delta(\alpha) = f_a(\alpha)$ para todo $\alpha \in A_j$, es decir, queda determinado por el índice j del conjunto de recuperación A_j. En otras palabras, el polinomio $\delta(x)$ es el mismo para cada par de elementos $\alpha_1, \alpha_2 \in A_j$.

Ejemplo 3.2.2. Construiremos un código LRC óptimo sobre $\mathbb{F}_q$ de parámetros $(n = 9, k = 4, r = 2)$. Puesto que necesitamos al menos 9 puntos distintos del cuerpo donde hacer la evaluación, debemos considerar $q \geq 9$. Definiremos el código $\mathcal{C}$ sobre $\mathbb{F}_{13}$.
La dificultad de la *construcción* 1 está en encontrar un polinomio $g(x)$ de grado $r+1 = 3$ que sea constante en 3 conjuntos disjuntos de tamaño 3. Más adelante daremos una construcción sistemática para la obtención de estos polinomios. Consideramos la siguiente partición:

$$\mathcal{A} = \{A_1 = \{1, 3, 9\},\ A_2 = \{2, 6, 5\},\ A_3 = \{4, 12, 10\}\}.$$

Notemos que el polinomio $g(x) = x^3$ es constante en los conjuntos A_i. Más adelante detallaremos un método para su obtención.
Sea $\mathbf{a} = (a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}) \in \mathbb{F}_{13}^4$, es decir, el vector de información de longitud $k = 4$ sobre $\mathbb{F}_{13}$, y definimos el polinomio de codificación como se había planteado:

$$\begin{aligned} f_{\mathbf{a}}(x) &= (a_{0,0} + a_{0,1}g(x)) + x(a_{1,0} + a_{1,1}g(x)) \\ &= (a_{0,0} + a_{0,1}x^3) + x(a_{1,0} + a_{1,1}x^3) \\ &= a_{0,0} + a_{1,0}x + a_{0,1}x^3 + a_{1,1}x^4. \end{aligned}$$

La palabra del código $\mathbf{c} \in \mathcal{C}$ asociada a la palabra $\mathbf{a}$ se obtiene como la evaluación del polinomio $f_{\mathbf{a}}$ en todos los puntos de los conjuntos de la partición $\mathcal{A}$, es decir, $\mathbf{c} = (f_{\mathbf{a}}(\alpha) : \alpha \in \cup_{i=1}^{3} A_i)$. Puesto que el grado de $f_{\mathbf{a}}$ es a lo sumo 4, la distancia mínima es como mínimo 5, y de hecho debe ser 5, de acuerdo con la cota dada en el teorema 2.2.1. Por ejemplo, para $\mathbf{a} = (1, 1, 1, 1)$, obtenemos la palabra del código asociada como

$$\begin{aligned} &(f_{\mathbf{a}}(1),\ f_{\mathbf{a}}(3),\ f_{\mathbf{a}}(9),\ f_{\mathbf{a}}(2),\ f_{\mathbf{a}}(6),\ f_{\mathbf{a}}(5),\ f_{\mathbf{a}}(4),\ f_{\mathbf{a}}(12),\ f_{\mathbf{a}}(10)) \\ &\qquad = (4, 8, 7, 1, 11, 2, 0, 0, 0). \end{aligned}$$

Supongamos que tenemos un borrón en la primera coordenada, es decir hemos perdido el valor $f_{\mathbf{a}}(1)$. Por la construcción que hemos seguido, este valor podrá ser recuperado utilizando otras 2 coordenadas de la palabra codificada, concretamente a las coordenadas correspondientes a las posiciones 3 y 9. Utilizando la construcción del polinomio dada en (3.4), obtenemos el polinomio $\delta(x) = 2x + 2$, y calculando $\delta(1) = 4$ obtenemos el valor que queríamos recuperar, $f_{\mathbf{a}}(1) = 4$.
Este ejemplo se puede ver con más detalle en la sección 5.1 de este trabajo.

Hacemos ahora algunas observaciones a cerca de los resultados vistos:

1. La *construcción* 1 es una extensión directa de los códigos Reed-Solomon clásicos, ambos son evaluaciones de algunos polinomios definidos por el vector del mensaje. de hecho, en el caso particular en el que $r = k$, nuestra construcción es precisamente la de los códigos Reed-Solomon. Notemos que, en ese caso, cada coeficiente del polinomio es constante y por lo tanto la construcción del código no requiere de un polinomio *bueno*. Por la misma razón, para un código Reed-Solomon el conjunto de recuperación A se puede elegir arbitrariamente como un subconjunto de $\mathbb{F}_q$, mientras que la condición $r < k$ impone restricciones en la elección de dichas coordenadas.

2. En la *construcción* 1 hemos supuesto que k es divisible por r, sin embargo, veamos que esta restricción puede ser fácilmente eliminada. Supongamos que r no divide a k y definimos el coeficiente $f_{\mathbf{a}}$ de la fórmula (3.2) de la siguiente manera:

$$f_i(x) = \sum_{j=0}^{s(k,r,i)} a_{ij} g(x)^j, \text{ para } i \in \{0, \cdots, r-1\}, \tag{3.7}$$

donde

$$s(k, r, i) = \begin{cases} \left\lfloor \frac{k}{r} \right\rfloor & si \quad i < k \ mod \ r \\ \left\lfloor \frac{k}{r} \right\rfloor - 1 & si \quad i \geq k \ mod \ r \end{cases}$$

Se ve de forma sencilla que k símbolos de información definen r de los coeficientes, y el polinomio de codificación resultante $f_{\mathbf{a}}$ tiene grado a lo sumo $k + \frac{k}{r} - 2$. El resto de la construcción no se ve afectada por este cambio.

3.2.2. Construcción de códigos LRC óptimos. Estructura algebraica del cuerpo.

La dificultad principal de la construcción vista en la sección anterior está en encontrar un polinomio *bueno*, $g(x)$, junto con la correspondiente partición del subconjunto A del cuerpo $\mathbb{F}_q$. En esta sección veremos cómo construir $g(x)$ utilizando grupos multiplicativos y aditivos de $\mathbb{F}_q$.
El grupo multiplicativo $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$ es cíclico, y el grupo aditivo $\mathbb{F}_q^+$ es isomorfo al producto de l copias del grupo aditivo $\mathbb{F}_p^l$, donde $q = p^l$, siendo p la característica del cuerpo. Los siguientes resultados construyen polinomios buenos a partir de cualquier subgrupo de $\mathbb{F}_q^*$ o de $\mathbb{F}_q^+$.

Proposición 3.2.3. *Sea H un subgrupo de $\mathbb{F}_q^*$ o de $\mathbb{F}_q^+$. El polinomio anulador del subgrupo*

$$g(x) = \prod_{h \in H} (x - h) \tag{3.8}$$

es constante en cada cogrupo de H (también llamados clases laterales de H).

Demostración. Supongamos que H es un subgrupo multiplicativo y sean $a,\ a\bar{h}$ dos elementos del cogrupo $aH = \{ah : h \in H\}$, donde $\bar{h} \in H$. Entonces,

$$g(a\bar{h}) = \prod_{h \in H} (a\bar{h} - h) = \bar{h}^{|H|} \prod_{h \in H} (a - h\bar{h}^{-1})$$

$$= \prod_{h \in H} (a - h) = g(a).$$

La demostración para subgrupos aditivos es análoga. □

Notemos que si H es un subgrupo multiplicativo de $\mathbb{F}_q^*$ entonces, en la ecuación (3.8), podemos escribir el polinomio g como $g(x) = x^{|H|} - 1$. De forma equivalente podemos tomar $g(x) = x^{|H|}$.
Estos polinomios anuladores de subgrupos forman una clase de polinomios buenos, que pueden ser utilizados en la construcción de nuestros códigos óptimos. La partición $\mathcal{A}$ es una unión de cogrupos de H, luego la longitud del código n puede ser cualquier múltiplo de $r + 1$, satisfaciendo $n \leq q - 1$ (o simplemente $n \leq q$ para el caso de grupos aditivos). Por tanto el cardinal de un subgrupo divide al del grupo y se tiene que $q \ mod\ (r + 1)$ es 1 (o bien 0 para el caso de grupos aditivos).
los parámetros de los códigos LRC construidos utilizando subgrupos se definen de forma natural en función de los posibles tamaños de los subgrupos. Notemos que en el ejemplo anterior se utiliza el subgrupo multiplicativo $H = \{1, 3, 9\}$ del cuerpo $\mathbb{F}_{13}$, cuyo anulador es $g(x) = x^3 - 1$. En el ejemplo se utiliza otro polinomio *bueno*, $g(x) = x^3$.

Ejemplo 3.2.4. En este ejemplo construiremos un código LRC de parámetros $(n = 12, k = 6, r = 3)$ y distancia mínima $d = 6$ sobre el cuerpo finito $\mathbb{F}_{13}$, utilizando un subgrupo multiplicativo de este. Notemos que 5 verifica $5^{r+1} = 5^4 = 1\ mod\ 13$, es más, es una raíz cuarta de la unidad, por tanto, el polinomio $g(x) = x^4$ es constante en los subgrupos del grupo cíclico $H = \{1, 5, 12, 8\}$ generado por las potencias de 5. Notemos que el polinomio g construido en la proposición anterior es $g(x) = x^4 - 1$, aunque utilizaremos el polinomio $g(x) = x^4$ en su lugar. Puesto que los polinomios $1,\ x^4 - 1$ generan el mismo

subespacio que los polinomios $1,\ x^4$, los códigos que se obtienen como resultado serán equivalentes. Utilizaremos los cogrupos de $H,\ 1H,\ 2H = 3H$ y $4H$ para obtener una partición de $\mathbb{F}_{13}$,

$$\mathcal{A} = \{A_1 = \{1, 5, 12, 8\},\ A_2 = \{2, 10, 11, 3\},\ A_3 = \{4, 7, 9, 6\}\}$$

para el vector de información $(a_0,\ a_1,\ a_2,\ a_4,\ a_5,\ a_6)$ definimos el polinomio de codificación de acuerdo con la figura (3.1):

$$f_{\mathbf{a}}(x) = \sum_{i=0,\ i\neq 3}^{6} a_i x^i = f_0(x) + f_1(x)x + f_2(x)x^2,$$

donde los coeficientes del polinomio son

$$f_0(x) = a_0 + a_4x^4,\ f_1(x) = a_1 + a_5x^4,\ f_0(x) = a_2 + a_6x^4.$$

La palabra codificada correspondiente se obtiene evaluando el polinomio $f_{\mathbf{a}}(x)$ para cada $x \in \mathbb{F}_{13}^*$.
Este ejemplo está desarrollado con más detalle en la sección 5.2 de este trabajo.

Ejemplo 3.2.5. En este ejemplo construiremos un código LRC óptimo utilizando un subgrupo aditivo del cuerpo. Sea α un elemento primitivo del cuerpo $\mathbb{F}_{2^4}$. Tomamos el subgrupo aditivo $H = \{x + y\alpha : x, y \in \mathbb{F}_2\}$. El polinomio $g(x)$ de la fórmula (3.8) será

$$g(x) = x(x + 1)(x + \alpha)(x + \alpha + 1)$$

$$= x^4 + (\alpha^2 + \alpha + 1)x^2 + (\alpha^2 + \alpha)x.$$

Construiremos un código LRC óptimo de parámetros $(n = 12, k = 6, r = 3)$ con distancia mínima $d = 6$. Para $i \in \{0, 1, 2\}$ los coeficientes del polinomio serán

$$f_i(x) = a_{i,0} + a_{i,1}g(x),$$

El subgrupo H es de orden 4, luego para tener 12 puntos de evaluación elegimos 3 cogrupos de H cualesquiera dentro de los 4 posibles, y evaluamos el polinomio de codificación

$$f_{\mathbf{a}}(x) = f_2(x)x^2 + f_1(x)x + f_0(x)$$

en los elementos de esos subgrupos elegidos. El teorema 3.2.1 garantiza que el código resultante tiene las propiedades exigidas. Comparando este código con un código MDS de parámetros $(12, 6)$ se observa que ambos códigos se pueden definir sobre $\mathbb{F}_{2^4}$, sin embargo, mediante la reducción de la distancia mínima, de 7 a 6 hemos conseguido reducir la localidad a la mitad, de 6 a 3.

Las estructuras aditivas y multiplicativas del cuerpo pueden ser combinadas en métodos más generales para la construcción de polinomios buenos. Para dos subconjuntos $G, H \subseteq \mathbb{F}_q$, diremos que H es cerrado para la multiplicación por G si para cada par de elementos $h \in H$, $g \in G$ se tiene que $hg \in H$.

Teorema 3.2.6. *Sean l, s, m números enteros tales que l divida a s, $p^l \bmod m = 1$, con p un número primo. Sea H un subgrupo aditivo del cuerpo $\mathbb{F}_{p^s}$ el cual es cerrado para la multiplicación por elementos del cuerpo $\mathbb{F}_{p^l}$, y sean $\alpha_1, \cdots, \alpha_m$ las raíces de la unidad de orden m en $\mathbb{F}_{p^s}$. Entonces para cualquier $b \in \mathbb{F}_{p^s}$ el polinomio*

$$g(x) = \prod_{i=1}^{m} \prod_{h \in H} (x + h + \alpha_i) \tag{3.9}$$

es constante en la unión de los siguientes cogrupos de H, $\cup_{1 \leq i \leq m}(H + b\alpha_i)$, y el cardinal de esta unión satisface que

$$|\cup_{1 \leq i \leq m} (H + b\alpha_i)| = \begin{cases} |H| & si \quad b \in H \\ \\ m|H| & si \quad b \notin H \end{cases}$$

Demostración. Sea $\bar{h} \in H$ y sea $\bar{h} + b\alpha_j$ un elemento arbitrario, entonces

$$\begin{aligned} g(\bar{h} + b\alpha_j) &= \prod_{i=1}^{m} \prod_{h \in H} (\bar{h} + b\alpha_j + h + \alpha_i) \\ &= \prod_{i=1}^{m} \prod_{h \in H} (b\alpha_j + h + \alpha_i) \\ &= \alpha_j^{-m|H|} \prod_{i=1}^{m} \prod_{h \in H} (b + h\alpha_j^{-1} + \alpha_i\alpha_j^{-1}) \\ &= \prod_{i=1}^{m} \prod_{h \in H} (b + h\alpha_j^{-1} + \alpha_i) \\ &= \prod_{i=1}^{m} \prod_{h \in H} (b + h + \alpha_i) \\ &= g(b), \end{aligned}$$

donde hemos hecho cambio de variables y utilizado la hipótesis de que H es cerrado para la multiplicación por cualquier raíz de la unidad de orden m, ya que es cerrado para la multiplicación por elementos de $\mathbb{F}_{p^l}$. Para probar la parte final relacionada con el cardinal del conjunto unión, consideramos dos raíces de la unidad de orden m, α_i, α_j, entonces

$$H + b\alpha_i = H + \alpha_j \Leftrightarrow b(\alpha_i - \alpha_j) \in H \Leftrightarrow b \in H,$$

donde el último paso se deduce del hecho de que $\alpha_i - \alpha_j$ es un elemento no nulo del cuerpo $\mathbb{F}_{p^l}$ y H es cerrado para la multiplicación por elementos de $\mathbb{F}_{p^l}$. □

Algunas observaciones sobre esta construcción:

1. Para la construcción de un polinomio *bueno* utilizando el resultado dado por el teorema 3.2.6 es necesario encontrar un subgrupo aditivo H de $\mathbb{F}_{p^s}$, el cual sea cerrado para la multiplicación por elementos de $\mathbb{F}_{p^l}$. Notemos que, ya que l divide a s, el cuerpo $\mathbb{F}_{p^l}$ se puede ver como un vector de dimensión $\frac{s}{l}$ sobre el cuerpo $\mathbb{F}_{p^l}$. Por lo tanto ningún subespacio H de dimensión $1 \leq t \leq \frac{s}{l}$ puede ser un subgrupo aditivo del cuerpo $\mathbb{F}_{p^s}$ que sea cerrado para la multiplicación por elementos de $\mathbb{F}_{p^l}$, y su cardinal sea $|H| = (p^l)^t = p^{lt}$.

2. Puesto que el grado del polinomio $g(x)$ en la fórmula (3.9) es $m|H|$, es claro que toma valores distintos sobre los diferentes conjuntos de $U = \cup_i (H + b\alpha_i)$. En otras palabras, $g(x)$ particiona $\mathbb{F}_{p^s}$ en $\frac{p^s - |H|}{m|H|}$ conjuntos de cardinal $m|H|$ y un conjunto de cardinal $|H|$, de acuerdo con los valores que toma en los elementos del cuerpo. Por consiguiente, sobre el cuerpo de cardinal p^s, uno puede construir un código LRC óptimo de longitud $n \leq p^s$ tal que $m|H|$ divida a la longitud del código n.

Supongamos que se quiere construir un código LRC sobre un cuerpo con una característica en concreto p, por ejemplo $p = 2$, entonces el teorema 3.2.6 nos da un método flexible para construir polinomios buenos para una amplia elección de parámetros. De forma más específica, sea m un entero no divisible por p, y sea l el menor entero que verifique que $p^l = 1 \ mod \ m$ (notemos que $l \leq \varphi(m)$, donde φ denota la función φ de Euler). En este caso es posible construir un polinomio bueno que sea constante en conjuntos de tamaño mp^t para cualquier entero t que sea múltiplo de l.

Ejemplo 3.2.7. Supongamos $p = 7$ y los parámetros para nuestro código serán $n = 28, r = 13$. Para construir un código LRC óptimo necesitamos construir primero un polinomio $g(x)$ que sea constante en dos subconjuntos disjuntos de tamaño $r + 1 = 14$ sobre alguna extensión del cuerpo $\mathbb{F}_7$. Escribimos $14 = 2 \cdot 7$, luego $m = 2, l = 1$. Utilizando lo visto en el teorema anterior se puede obtener el polinomio bueno sobre $\mathbb{F}_{7^2}$. Con más precisión, teniendo en cuenta la segunda observación que hacíamos anteriormente, el polinomio $g(x)$ nos proporciona una partición del cuerpo de cardinal 49 en 3 conjuntos de cardinal 14 y uno más de cardinal 7. Para la construcción del código de longitud $n = 28$ se pueden elegir dos conjuntos cualesquiera, entre los tres conjuntos disjuntos de cardinal 14. Notemos que la dimensión del código puede tomar cualquier valor que verifique $k \leq \frac{lr}{r+1} = 26$.
Vamos a diferenciar las construcciones posibles de polinomios buenos en función al valor de los parámetros. Supongamos que se quiere construir un polinomio bueno sobre una extensión de $\mathbb{F}_p$ que sea constante sobre subconjuntos disjuntos de cardinal mp^t, donde m y p son primos entre si, entonces:

1. Si $t = 0$, se pueden utilizar subgrupos multiplicativos de alguna extensión $\mathbb{F}_{p^l}$ satisfaciendo $p^l = 1 \ mod \ m$.

2. Si $t > 0$ y $m = 1$, podemos basarnos en subgrupos aditivos.

3. Si $t, m > 1$ y además t es múltiplo de l, donde l es el menor entero tal que $p^l = 1 \ mod \ m$, entonces la construcción se puede llevar a cabo combinando las estructuras aditivas y multiplicativas del cuerpo como en el teorema 3.2.6.

Existe un caso en el que no seremos capaces de construir estos polinomios buenos. Por ejemplo, utilizando la técnica que se acaba de exponer, no es posible construir un código con localidad $r = 5$ sobre una extensión del cuerpo $\mathbb{F}_2$. Esto se sigue del hecho de que el cardinal del conjunto es $r + 1 = 5 + 1 = 3 \cdot 2$, luego $m = 3$ y $l = 2$ es el menor entero que verifica que $2^l = 1 \ mod \ 3$, sin embargo, $t = 1$ no es un múltiplo de $l = 2$. Por otro lado, un argumento simple de conteo nos muestra que los polinomios buenos existen también para los casos sin resolver en los que el cuerpo subyacente $\mathbb{F}_q$ es suficientemente grande.

Proposición 3.2.8. *Consideramos un cuerpo finito $\mathbb{F}_q$, entonces existe un polinomio bueno de grado $r + 1$ que es constante en al menos $\left\lceil \frac{\binom{q}{r+1}}{q^r} \right\rceil$ conjuntos de cardinal $r + 1$.*

Demostración. Consideramos el conjunto

$$M_{q,r} = \{f \in \mathbb{F}_q[x] : f(x) = \prod_{i=1}^{r+1}(x - \alpha_i)\},$$

donde α_i, $i \in \{1, \cdots, r+1\}$ varían sobre todas las $\binom{q}{r+1}$ posibles elecciones de subconjuntos de cardinal $r+1$ sobre el cuerpo finito de q elementos. En otras palabras, $M_{q,r}$ es el conjunto de todos los polinomios mónicos de grado $r+1$ en $\mathbb{F}_q[x]$ que además tengan $r+1$ ceros distintos en $\mathbb{F}_q$. Diremos que dos polinomios $f(x) = x^{r+1} + \sum_{i=0}^{r} a_i x^i,\ g(x) \in M_{q,r}$ son equivalentes si difieren tan solo en una constante. Claramente esto define una relación de equivalencia sobre $M_{q,r}$, y el número de clases de equivalencia distintas es a lo sumo q^r de acuerdo al número de elecciones de r-uplas de los coeficientes $a_1, \cdots, a_r$. Por lo tanto existe una clase de equivalencia de cardinal al menos $\lceil \binom{q}{r+1}/q^r \rceil$. Sea f un representante de dicha clase de equivalencia, y notemos que es constante en el conjunto de ceros de cualquier otro polinomio g de su misma clase de equivalencia. Concluimos entonces que f es un polinomio bueno, el cual es constante en conjuntos de cardinal $r+1$, donde el número de conjuntos es, como poco, $\lceil \binom{q}{r+1}/q^r \rceil$. □

Cuando q es suficientemente grande, por ejemplo, $q > n(r+1)^r$, el número $\lceil \binom{q}{r+1}/q^r \rceil$ excede a $\frac{n}{r+1}$, el cual es el número deseado de conjuntos para la construcción. Por ejemplo, tomando $q = 2^{11}$, se observa que existe un polinomio $g \in \mathbb{F}_q[x]$ de grado $r+1 = 6$ el cual es constante en al menos 3 conjuntos disjuntos de cardinal 6. De hecho, vemos que $\frac{\binom{2^{11}}{6}}{(2^{11})^5} \approxeq 2{,}82$. Usando la construcción 1 y el polinomio g, podemos construir un código LRC óptimo sobre $\mathbb{F}_q$ de longitud $n = 18$, localidad $r = 5$ y de cualquier dimensión $k \leq 15$.

3.2.3. Vista general de la familia de códigos LRC

En esta sección estudiaremos con mayor detalle el conjunto de polinomios de la forma (3.1) y su relación con el espacio $\mathbb{F}^n$, generalizando así la construcción que hemos planteado. Sea $A \subseteq \mathbb{F}$, y sea $\mathcal{A}$ una partición de A en m conjuntos A_i. Consideramos el conjunto de polinomios $\mathbb{F}_{\mathcal{A}[x]}$ de grado menor que $|A|$ y que son constantes en cada conjunto de la partición:

$$\mathbb{F}_{\mathcal{A}}[x] = \{f \in \mathbb{F}[x] : f \text{ constante en } A_i,\ i \in \{1, \cdots, m\},\ deg(f) < |A|\} \tag{3.10}$$

El anulador de A es el polinomio mónico h de menor grado tal que $h(a) = 0$ si $a \in A$, es decir, $h(x) = \prod_{a \in A}(x - a)$. Observamos que el conjunto $\mathbb{F}_{\mathcal{A}}$ con las operaciones suma y producto usuales módulo $h(x)$ constituye un álgebra conmutativa unitaria. Como los polinomios de $\mathbb{F}_{\mathcal{A}}[x]$ son constantes en los conjuntos de $\mathcal{A}$, escribiremos $f(A_i)$ para hacer referencia al valor que toma el polinomio en los elementos del conjunto $A_i \in \mathcal{A}$. En general simplificaremos la notación del producto de polinomios, escribiendo fg en lugar de $fg \ mod\ h$. La siguiente proposición lista algunas de las propiedades del álgebra.

Proposición 3.2.9. *1. Sea $f \in \mathbb{F}_{\mathcal{A}[x]}$ un polinomio no constante, entonces $max_i|A_i| \leq deg(f) \leq |A|$.*

2. La dimensión $dim(\mathbb{F}_{\mathcal{A}}[x]) = m$, y los m polinomios $f_1, \cdots, f_m$ que satisfacen $f_i(A_i) = \delta_{i,j}$ y $deg(f_i) < |A|$ forman una base, donde $\delta_{i,j}$ denota la función del-

ta de Kronecker. Explícitamente:

$$f_i(x) = \sum_{a \in A_i} \prod_{b \in A \setminus a} \frac{x-b}{a-b}. \tag{3.11}$$

3. *Sean $\alpha_1, \cdots, \alpha_m$ elementos no nulos de $\mathbb{F}$, y sea g el polinomio de grado $deg(g) < |A|$ que satisface $g(A_i) = \alpha_i$ para todo $i \in \{1, \cdots, m\}$, es decir,*

$$g(x) = \sum_{i=1}^{m} \alpha_i \sum_{a \in A_i} \prod_{b \in A \setminus a} \frac{x-b}{a-b}.$$

Entonces los polinomios $1, g, \cdots, g^{m-1}$ forman una base de $\mathbb{F}_{\mathcal{A}}[x]$.

4. *Existen m números enteros $0 < d_0 < d_1 < \cdots < d_{m-1} < |A|$ tales que el grado de cada polinomio en $\mathbb{F}_{\mathcal{A}}[x]$ es d_i, para algún valor de i.*

Demostración. 1. Para un polinomio $f \in \mathbb{F}_{\mathcal{A}}[x]$ y un conjunto $A_i \in \mathcal{A}$ se tiene que el polinomio $f(x) - f(A_i)$ tiene como poco $|A_i|$ raíces en $\mathbb{F}$ y por lo tanto $deg(f) \geq |A_i|$.

2. Los m polinomios $f_1, \cdots, f_m$ definidos en (3.11) son linealmente independientes, supongamos que existen ciertos valores λ_i de forma que

$$\sum_{i=1}^{m} \lambda_i f_i(x) = 0,$$

luego para cualquier valor $j \in \{1, \cdots, m\}$ se tendrá que

$$\sum_{i=1}^{m} \lambda_i f_i(A_i) = \sum_{i=1}^{m} \lambda_i \delta_{i,j} = \lambda_j = 0.$$

Por definición, los polinomios $f_1, \cdots, f_m$ generan el espacio $\mathbb{F}_{\mathcal{A}}[x]$.

3. Aplicando lo demostrado en el apartado (2) basta con ver que los polinomios $1, g(x), \cdots, g(x)^{m-1}$ son linealmente independientes. Supongamos que para ciertos valores $\beta_j \in \mathbb{F}$ se tiene que

$$\sum_{j=1}^{m} \beta_j g^{j-1}(x) = 0. \tag{3.12}$$

Definimos la matriz $\mathcal{V} = (v_{i,j})$, donde $v_{i,j} = (g^{j-1}(A_i))$. De la ecuación (3.12) se sigue que $\mathcal{V} \cdot (\beta_1, \cdots, \beta_m)^{\perp} = 0$, sin embargo $\mathcal{V}$ es una matriz de Vandermonde definida por m valores distintos no nulos del cuerpo, por lo tanto es invertible y se concluye que $\beta_i = 0$, para todo i.

4. Sea $\{f_0, \cdots, f_{m-1}\}$ una base del espacio $\mathbb{F}_{\mathcal{A}}[x]$. Sin pérdida de generalidad podemos suponer que los grados de dichos polinomios son todos distintos, ya que si no fuera así podríamos realizar operaciones lineales sobre ellos y obtener lo buscado. Para ello, consideramos una matriz de dimensiones $m \times |A|$ cuyas m filas se corresponden con los m vectores de coeficientes de los polinomios f_i. Las filas de una forma escalonada de esta matriz se corresponderán con los elementos de una base, formada por polinomios de distinto grado. Sea $d_i = deg(f_i)$ y supongamos que $d_0 < d_1 <$

$\cdots < d_{m-1}$. Puesto que los polinomios constantes están contenidos en el álgebra se tiene que $d_0 = 0$ y se concluye el resultado.

□

Ahora consideraremos un caso particular de álgebra generada por un conjunto A de n elementos, suponiendo que la partición verifica $|A_i| = r + 1$ para cada valor de i.

Corolario 3.2.10. *Supongamos que $d_1 = r + 1$, luego existe un polinomio $g \in \mathbb{F}_{\mathcal{A}}[x]$ de grado $r + 1$. Entonces $d_i = i(r + 1)$ para todo $i \in \{0, \cdots, m - 1\}$, y los polinomios $1, g, \cdots, g^{m-1}$ definidos en el apartado (3) de la proposición anterior forman una base de $\mathbb{F}_{\mathcal{A}}[x]$.*

Demostración. Si existe dicho polinomio g, entonces claramente toma valores distintos en los distintos conjuntos de la partición $\mathcal{A}$. De no ser así, para alguna constante $c \in \mathbb{F}$, el polinomio $g - c$ tendría al menos $2(r + 1)$ raíces, siendo de grado $r + 1$, lo cual es una contradicción. Por tanto, en virtud del apartado (3) de la proposición anterior las potencias de g constituyen una base del álgebra y se concluye el resultado. □

Notemos que el álgebra $\mathbb{F}_{\mathcal{A}}[x]$ de la *construcción* 1 contiene un polinomio *bueno* de grado $r + 1$, satisfaciendo las condiciones del corolario 3.2.10, y por lo tanto $\mathbb{F}_{\mathcal{A}}[x]$ puede ser generado por las potencias de dicho polinomio.
A continuación, utilizaremos las propiedades del álgebra de polinomios definida por la partición $\mathcal{A}$ para construir códigos LRC de parámetros (n, k, r).

Construcción 2: Sea $A \subset \mathbb{F}$, con $|A| = n$, y sea $\mathcal{A}$ una partición del conjunto A en $m = \frac{n}{r+1}$ conjuntos de cardinal $r + 1$. Sea Φ una aplicación inyectiva de $\mathbb{F}^k$ sobre el espacio de polinomios

$$\mathcal{F}^r_{\mathcal{A}} = \oplus_{i=0}^{r-1} \mathbb{F}_{\mathcal{A}}[x] x^i.$$

Notemos que el espacio $\mathcal{F}^r_{\mathcal{A}}$ es suma directa de espacios de dimensión m, luego $dim(\mathcal{F}^r_{\mathcal{A}}) = mr$. Por lo tanto dicha aplicación inyectiva exista si y solo si $k \leq mr = \frac{nr}{r+1}$.
La aplicación Φ envía elementos o mensajes de $\mathbb{F}^k$ a un conjunto de polinomios codificadores. Construiremos el código evaluando los polinomios $f \in \Phi(\mathbb{F}^k)$ en los puntos de A. Si Φ es una aplicación lineal entonces el resultante código será también lineal.
Esta construcción se basa en una aplicación arbitraria $\Phi : \mathbb{F}^k \to \mathcal{F}^r_{\mathcal{A}}$, y constituye una generalización de la *construcción* 1, la cual utiliza una aplicación lineal particular con el mismo propósito. A continuación escribiremos $f_{\mathbf{a}}(x) = \Phi(a)$.

Teorema 3.2.11. *La construcción 2 nos da un código LRC de parámetros (n, k, r) con distancia mínima d, verificando:*

$$d \geq n - max\{deg(f_a - f_b) \, : \, a, b \in \mathbb{F}^k\} \geq n - max\{deg(f_a) \, : \, a \in \mathbb{F}^k\}. \tag{3.13}$$

Demostración. Para probar que el código es localmente recuperable repetiremos la demostración del teorema 4.2.1. Para un mensaje dado por el vector $\mathbf{a} \in \mathbb{F}^k$, sea

$$f_{\mathbf{a}}(x) = \sum_{i=0}^{r-1} f_i(x) x^i, \tag{3.14}$$

donde los coeficientes $f_i(x)$ satisfacen $f_i \in \mathbb{F}_{\mathcal{A}}[x]$. Tomamos $j \in \{1, \cdots, m\}$ y supongamos que el símbolo asociado al borrón y que queremos recuperar es $f_a(\alpha)$, con $\alpha \in A_j$. Definimos el polinomio de decodificación

$$\delta(x) = \sum_{i=0}^{r-1} f_i(\alpha) x^i. \tag{3.15}$$

Notemos que $\delta(\alpha) = f_{\mathbf{a}}(\alpha)$, en virtud de los resultados (3.14) y (3.15). Como $f_i \in \mathcal{F}_{\mathcal{A}}[x]$, para todo $\beta \in A_j$ se tiene que $f_{\mathbf{a}}(\beta) = \delta(\beta)$. Es más, como $\delta(x)$ tiene grado a lo sumo $r-1$, este puede ser obtenido mediante interpolación de $f_a(\beta) = \delta(\beta)$, con $\beta \in A_j \setminus \alpha$. Concluimos entonces que el valor del símbolo perdido $f_{\mathbf{a}}(\alpha)$ puede ser recuperado accediendo a los r elementos restantes del conjunto A_j.
Para terminar la demostración resta probar la desigualdad de la ecuación (3.13). Sean $(f_{\mathbf{a}}(\alpha))_{\alpha \in A}$, $(f_{\mathbf{b}}(\alpha))_{\alpha \in A}$ dos elementos del código obtenidos como codificación de dos vectores distintos $\mathbf{a}$, $\mathbf{b}$. Como Φ es inyectiva y $deg(f_{\mathbf{a}} - f_{\mathbf{b}}) < n$, los vectores del código que se corresponden con $f_{\mathbf{a}}$ y $f_{\mathbf{b}}$ son distintos y la desigualdad es inmediata. □

3.2.4. Codificación sistemática de códigos LRC

En el momento de la implementación práctica, es preferible tener códigos en su forma sistemática para simplificar la recuperación de la información.
Notemos que todas las construcciones que hemos ido dando a lo largo de la sección pueden ser modificadas para soportar códigos sistemáticos sin afectar a la distancia mínima asociada, modificando los polinomios de codificación (3.1), (3.14). En particular la *construcción* 1 se puede modificar para obtener códigos LRC óptimos en su forma sistemática. Describiremos esa modificación de forma rápida en esta sección.
Sea $\mathcal{A} = \{A_1, \cdots, A_m\}$, con $m = \frac{n}{r+1}$, una partición del conjunto $A \subseteq \mathbb{F}$ en m conjuntos de cardinal $r+1$, donde $|A| = n$. Para cada $i \in \{1, \cdots, \frac{k}{r}\}$ consideramos $B_i = \{\beta_{i,1}, \cdots, \beta_{i,r}\}$ un subconjunto de A_i de cardinal r. En nuestra codificación sistemática los símbolos del mensaje estarán en las coordenadas con localizaciones en los conjuntos B_i.
Notemos que el álgebra $\mathbb{F}_{\mathcal{A}}[x]$ tiene una base de polinomios f_i que verifica

$$f_i(A_j) = \delta_{i,j}, \text{ para } i, j \in \{1, \cdots, m\}. \tag{3.16}$$

Para cada conjunto B_i definimos r polinomios $\phi_{i,j}(x)$, para cada $j \in \{1, \cdots, r\}$, de grado menor que r tal que

$$\phi_{i,j}(\beta_{i,l}) = \delta_{i,l}.$$

Estos polinomios se pueden encontrar de forma sencilla mediante interpolación de Lagrange. Para k símbolos de información $\mathbf{a} = (a_{j,i})$, con $i \in \{1, \cdots, k/r\}$ y $j \in \{1, \cdots, r\}$ definimos el polinomio de codificación como

$$f_{\mathbf{a}}(x) = \sum_{i=1}^{\frac{k}{r}} f_i(x) \left(\sum_{j=1}^{r} a_{i,j} \phi_{i,j}(x) \right). \tag{3.17}$$

Por tanto la codificación de un mensaje $\mathbf{a} \in \mathbb{F}^k$ viene dada por el vector $(f_{\mathbf{a}}(\alpha) : \alpha \in A)$, como vimos en la figura (3.3). Además se puede comprobar de forma sencilla que $f_{\mathbf{a}} \in \mathcal{F}_{\mathcal{A}}^r$, luego cada símbolo tiene localidad r. Es más, por definición se tiene que

$$f_{\mathbf{a}}(\beta_{i,j}) = a_{i,j}, \text{ con } i \in \{1, \cdots, k/r\},\ j \in \{1, \cdots, r\},$$

luego el código es de hecho sistemático.
Aunque la fórmula (3.17) nos da un código LRC sistemático de parámetros (n, k, r) esta no nos garantiza una distancia mínima óptima, respecto a la cota dada en el teorema 3.2.1, de forma general. Esto se debe a que la mejor cota para el grado del polinomio de codificación $f_{\mathbf{a}}(x)$ es $f_{\mathbf{a}} \leq n-1$. Si el álgebra $\mathbb{F}_{\mathcal{A}}[x]$ es generada por las potencias de un polinomio *bueno* $g(x)$ (como veíamos en la parte 3 de la proposición 3.2.9) entonces es posible construir un código LRC óptimo sistemático. De hecho, hay que reemplazar cada polinomio f_i de la fórmula (4.16) por el polinomio $\bar{f}_i$, el cual es una combinación lineal de los polinomios $1, g, \cdots, g^{(k/r)-1}$ y satisface $\bar{f}_i(A_j) = \delta_{i,j}$ para todo $j \in \{1, \cdots, k/r\}$. Esto es de hecho posible ya que la matriz $\mathcal{V} = (g^{j-1}(A_i))$ es una matriz de Vandermonde y por tanto invertible. Claramente el grado de cada polinomio $\bar{f}_i$ es a lo sumo $((k/r)-1)(r+1)$, por lo tanto el grado de $f_{\mathbf{a}}$ es a lo sumo $k + k/r - 2$, lo que implica una distancia mínima óptima.

3.3. Generalizaciones de la construcción principal

En esta sección generalizaremos las construcciones vistas en la sección previa de distintas formas. Empezaremos con la construcción de códigos LRC de longitud arbitraria, suprimiendo la condición de que n sea divisible por $r+1$.

3.3.1. Códigos de longitud arbitraria

Como hemos mencionado, las construcciones de la sección anterior requieren de la condición de que n sea múltiplo de $r+1$. Con el fin de generalizar esta construcción, modificaremos de estos códigos con el fin de relajar esta condición. Mientras que la distancia mínima que se presenta a continuación no siempre alcanzará la cota tipo Singleton del teorema 2.2.1. como sería deseable, veremos que para el caso de códigos lineales esta será como mucho una unidad menor que el valor máximo posible. La única restricción que debemos imponer será que $n \neq 1 \ mod \ (r+1)$. Al igual que en secciones anteriores, para un conjunto $M \subset \mathbb{F}$ denotamos por $h_M(x) = \prod_{\alpha \in M}(x - \alpha)$ al polinomio anulador del conjunto M. En la siguiente construcción supondremos que n no es múltiplo de $(r+1)$. Por simplicidad supondremos que k es divisible por r, ya que esta restricción puede ser fácilmente suprimida a cambio de una notación algo más complicada.

Construcción 3: Sea $\mathbb{F}$ un cuerpo finito, sea $A \subset \mathbb{F}$ un subconjunto tal que $|A| = n$, con $n = s \ mod \ (r+1), \ s \neq 1$. Sea $m = \left\lceil \frac{n}{r+1} \right\rceil$ y consideramos una partición $\mathcal{A} = \{A_1, \cdots, A_m\}$ tal que $|A_i| = r+1$ para cada $1 \leq i \leq m-1$, y $1 < |A_m| = s < r+1$.
Consideramos ahora la aplicación $\Phi_i : \mathbb{F}^{k/r} \rightarrow \mathbb{F}_{\mathcal{A}}[x]$, inyectiva para cada $i \in \{0, \cdots, r-1\}$. Además, supongamos que Φ_{s-1} es una aplicación con llegada en el subespacio de polinomios $\mathbb{F}_{\mathcal{A}}[x]$ que se anula en el conjunto A_m, es decir, el rango de la aplicación Φ_{s-1} es el espacio

$$\{f \in \mathbb{F}_{\mathcal{A}}[x] : f(\alpha) = 0 \text{ para todo } \alpha \in A_m\}.$$

Dado un vector de entrada $\mathbf{a} = (a_0, \cdots, a_{r-1}) \in \mathbb{F}^k$, donde cada elemento a_i es un vector de dimensión k/r. Se define el polinomio de codificación como sigue:

$$f_{\mathbf{a}}(x) = \sum_{i=0}^{s-1} \phi_i(a_i)x^i + \sum_{i=s}^{r-1} \Phi_i(a_i)x^{i-s}h_{A_m}(x) = \sum_{i=0}^{s-1} f_i(x)x^i + \sum_{i=s}^{r-1} f_i(x)x^{i-s}h_{A_m}(x), \quad (3.18)$$

donde $\Phi_i(a_i) = f_i(x) \in \mathbb{F}_{\mathcal{A}}[x]$. Finalmente, definimos el código como el subespacio imagen asociado a la aplicación, como describíamos en (3.3).

Teorema 3.3.1. *La construcción 3 define un código LRC de parámetros* (n, k, r).

Demostración. Cada símbolo $f_{\mathbf{a}}(\alpha)$ para α en uno de los conjuntos $A_1, \cdots, A_{m-1}$ puede ser recuperado utilizando el mismo procedimiento de decodificación que en la *Construcción 2*. Esto se sigue del hecho de que el polinomio $f_{\mathbf{a}}(x)$ es un elemento del espacio $\oplus_{i=0}^{r-1}\mathbb{F}_{\mathcal{A}}[x]x^i$, y por lo tanto el símbolo puede ser recuperado accediendo a otros r símbolos. El único caso especial es el de la recuperación de símbolos en el conjunto A_m. Por la definición de Φ_{s-1} y la ecuación (3.18), la restricción del polinomio de codificación $f_{\mathbf{a}}(x)$ al conjunto A_m es un polinomio de grado a lo sumo $s-2$. Por tanto, para la recuperación del valor $f_{\mathbf{a}}(\alpha)$ para un valor $\alpha \in A_m$, construimos el polinomio

$$\delta(x) = \sum_{i=0}^{s-2} f_i(\alpha)x^i$$

a partir del conjunto de $s-1$ valores, $\{\delta(\beta) = f_{\mathbf{a}}(\beta) : \beta \in A_m \setminus \{\alpha\}\}$. El elemento que se desea recuperar es de hecho $f_{\mathbf{a}}(\alpha) = \delta(\alpha)$, lo que prueba la propiedad de localidad del código. □

Construcción 4: Sea $\mathbb{F}$ un cuerpo finito, y sea $A \subset \mathbb{F}$ un subconjunto de cardinal n, con $n = s \ mod \ (r+1), \ s \notin \{0, 1\}$. Supongamos que $(k+1)$ es divisible por r.
Sea $\mathcal{A}$ una partición del conjunto A en m conjuntos $A_1, \cdots, A_m$, del mismo cardinal que en la construcción anterior. Sea $g(x)$ un polinomio de grado $r+1$, tal que sus potencias $1, g, \cdots, g^{m-1}$ generan el álgebra $\mathbb{F}_{\mathcal{A}}[x]$. Sin perdida de generalidad podemos asumir que g se anula en el conjunto A_m, si no fuera así, podríamos tomar las potencias del polinomio $g(x) - g(A_m)$ como la base del álgebra.
Sea $\mathbf{a} = (a_0, \cdots, a_{r-1}) \in \mathbb{F}^k$ el vector con la información de entrada, tal que cada $\mathbf{a}_i$ con $i \neq s-1$ es un vector de longitud $(k+1)/r$ y el elemento $\mathbf{a}_{s-1}$ es de longitud $\frac{k+1}{r} - 1$. Definimos el polinomio de codificación como

$$f_{\mathbf{a}}(x) = \sum_{i=0}^{s-2}\sum_{j=0}^{\frac{k+1}{r}-1} a_{i,j}g(x)^j x^i + \sum_{j=1}^{\frac{k+1}{r}-1} a_{s-1,j}g(x)^j x^{s-1} + \sum_{i=s}^{r-1}\sum_{j=0}^{\frac{k+1}{r}-1} a_{i,j}g^j(x)x^{i-s}h_{A_m}(x). \tag{3.19}$$

El código se define como el conjunto de las evaluaciones $f_{\mathbf{a}}(x)$ con $\mathbf{a} \in \mathbb{F}^k$.

Teorema 3.3.2. *El código dado por la construcción anterior es un código LRC de parámetros* (n, k, r)*, con distancia mínima* d *que verifica*

$$d \geq n - k - \left\lceil \frac{k}{r} \right\rceil + 1. \tag{3.20}$$

Notemos que la distancia mínima que se plantea es a lo sumo una unidad menor que el valor máximo posible.

Demostración. Notemos que la codificación es lineal y el polinomio de codificación de la figura (3.19) es de grado a lo sumo

$$\left(\frac{k+1}{r} - 1\right)(r+1) + (r-1) =$$

$$= k + 1 - r + \frac{k+1}{r} - 1 + r - 1 = k + \left\lceil \frac{k}{r} \right\rceil - 1,$$

de donde se sigue la cota de (3.20) pedida.
La propiedad de localidad del código se obtiene de forma similar a lo visto en la *construcción 3*. En efecto, si se requiere recuperar el símbolo $f_{\mathbf{a}}(\alpha)$ para cierto $\alpha \in A_m$, requerimos de un polinomio de grado a lo sumo $s-2$, obtenido a partir de un conjunto de $s-1$ puntos de interpolación. □

3.4. Conclusión

En este capítulo se han construido códigos que satisfacen la igualdad en la cota de tipo Singleton para la distancia mínima, enunciada en el teorema 2.2.1, para cualquier valor asignado al parámetro de la localidad r, $1 < r < k$. Estos códigos constituyen una generalización de los Reed-Solomon teniendo en cuenta el parámetro de localidad pedido.

Capítulo 4

Códigos Localmente Recuperables con Detección de Errores Locales

En este capítulo presentamos una variable de los códigos localmente recuperables, la cual nos permitirá, a demás, detectar errores en los conjuntos de recuperación que emplearemos en el proceso. Con esta nueva prestación podremos evitar añadir más errores y no superar así la capacidad correctora global del código. La fuente principal empleada para la redacción de este capítulo ha sido el artículo [3].

4.1. Introducción

Para tener un sistema de almacenamiento fiable y eficiente, queremos ser capaces de recuperar la información de un nodo cuando este falla o no está disponible acudiendo a la información almacenada en otros nodos. Daremos de nuevo definiciones y términos de forma que se amolden a los resultados que queremos mostrar en este capítulo. En términos de códigos, supongamos dado un código lineal $\mathcal{C}$ de parámetros $[n, k]$ sobre un cuerpo finito $\mathbb{F}_q^n$.

Definición 4.1.1. Diremos que una coordenada $i \in \{1, \ldots, n\}$ es localmente recuperable con localidad r si existe un conjunto de recuperación $R \subseteq \{1, \ldots, n\}$ con $i \notin R, \#R = r$ tal que para cada $\mathbf{x} \in \mathcal{C}$, un error en la coordenada x_i de $\mathbf{x}$ se puede recuperar utilizando los elementos x_j, con $j \in R$.

Definición 4.1.2. Un código es localmente recuperable con localidad menor o igual que r si así lo verifica cada coordenada. La localidad de $\mathcal{C}$ es el mínimo entero que verifica esta condición.

Ejemplo 4.1.3. Un código $\mathcal{C}$ MDS de parámetros $[n, k, d]$ verifica que su localidad es k. Los códigos Reed-Solomon son un ejemplo: Mediante interpolación polinómica, k datos correctos son suficientes para recuperar la información perdida en un borrón.

Esta idea plantea algunos inconvenientes. Si durante el proceso de recuperación de x_i alguna de las coordenadas del conjunto R contiene algún error, la recuperación de x_i será también errónea y ambos errores quedarán sin detectar. Es más, estos nuevos errores podrían superar la capacidad correctora global del código $\mathcal{C}$ original y resultar así imposible su recuperación, incluso utilizando el código completo. Por ello resultará interesante

utilizar conjuntos de recuperación que también permitan detectar errores locales en las coordenadas usadas durante el proceso de recuperación.
Consideramos un código $\mathcal{C}$ de parámetros $[n, k, d]$ y G una matriz generatriz asociada con columnas $\mathbf{c}_1 \dots \mathbf{c}_n$. Veamos algunas propiedades:

Proposición 4.1.4. *El conjunto $R \subset \{1, \dots, n\}$, con $i \notin R$, es un conjunto de recuperación para la coordenada i si verifica:*

$$\boldsymbol{c}_i \in \langle \boldsymbol{c}_j : j \in R \rangle, \text{ el espacio generado por } \{\boldsymbol{c}_j : j \in R\}.$$

Demostración. Para ver que un conjunto R con estas propiedades efectivamente constituye un conjunto de recuperación, veamos que para cada $\mathbf{x} \in \mathcal{C}$ se tiene que $x_i = \sum_{j \in R} \lambda_j x_j$, para ciertos $\lambda_j \in \mathbb{F}_q$.
Por un lado $\mathbf{c}_i \in \langle \mathbf{c}_j : j \in R \rangle \Rightarrow \exists\ (\alpha_j)_{j \in R}$ tal que $\mathbf{c}_i = \sum_{j \in R} \alpha_j \mathbf{c}_j$, en particular, para cada $m \in \{1, \dots, k\},\ c_{i,m} = \sum_{j \in R} \alpha_j c_{j,m}$.
Por otro lado las k filas de G constituyen una base de C, luego existen $(\beta_m)_{m \in \{1,\dots,k\}}$ tal que $\mathbf{x} = \sum_{m=1}^{k} \beta_m \mathbf{f}_m$, donde $\mathbf{f}_m$ denota la fila m-ésima de la matriz G definida en el enunciado. En particular $x_i = \sum_{m=1}^{k} \beta_m c_{i,m}$, y $x_j = \sum_{m=1}^{k} \beta_m c_{j,m},\ \forall j \in R$. Teniendo en cuenta lo anterior:
$x_i = \sum_{m=1}^{k} \beta_m c_{i,m} = \sum_{m=1}^{k} \beta_m \sum_{j \in R} \alpha_j c_{j,m} = \sum_{j \in R} \sum_{m=1}^{k} \alpha_j \beta_m c_{j,m} = \sum_{j \in R} \alpha_j \sum_{m=1}^{k} \beta_m c_{j,m} = \sum_{j \in R} \lambda_j x_j$,
como queríamos demostrar. □

Sea $\pi_R : \mathbb{F}_q^n \to \mathbb{F}_q^r$ la proyección sobre las coordenadas de R, donde $r = \#R$. Para cada $\mathbf{x} \in \mathbb{F}_q^n$ abreviaremos escribiendo $\mathbf{x}_R = \pi_R(\mathbf{x})$.
Consideramos los siguientes códigos a partir de un código $\mathcal{C}$, perforados y recortados, respectivamente:

- $\mathcal{C}[R] = \{\mathbf{x}_R : \mathbf{x} \in \mathcal{C}\} = \pi_R(\mathcal{C})$.
- $\mathcal{C}[[R]] = \{\mathbf{x}_R : \mathbf{x} \in \mathcal{C}, sop(\mathbf{x}) \subseteq R\}$.

El siguiente lema relaciona estos dos códigos:

Lema 4.1.5. *Sea $\mathcal{C}$ un código lineal y $R \subseteq \{1, \cdots, n\}$. Se tiene que*

$$\mathcal{C}[R]^{\perp} = \mathcal{C}^{\perp}[[R]].$$

Demostración. Sea $\mathbf{y}$ un vector cualquiera. Veremos que las condiciones de pertenencia a cada código son equivalentes.
Supongamos $\mathbf{y} \in \mathcal{C}[R]^{\perp}$. Por definición $\mathbf{y} \in \mathcal{C}[R]^{\perp} \Leftrightarrow \forall \mathbf{x}_R \in \mathcal{C}[R]$ se tiene que $\mathbf{x}_R \cdot \mathbf{y} = 0 \Leftrightarrow \sum_{j \in R} x_j y_j = 0$.
Por otro lado, supongamos $\mathbf{y} \in \mathcal{C}^{\perp}[[R]]$. Por definición $\mathbf{y} \in \mathcal{C}^{\perp}[[R]] \Leftrightarrow$ existe $\mathbf{x}' \in \mathcal{C}^{\perp}$ con $sop(\mathbf{x}') \subseteq R$ tal que $\mathbf{y} = \pi_R(\mathbf{x}')$, o equivalentemente, $y_j = x'_j$ para todo $j \in R$. De la

condición de que $\mathbf{x}' \in \mathcal{C}^{\perp}$ se obtiene que $\mathbf{x}' \cdot \mathbf{x} = 0$ para todo $\mathbf{x} \in \mathcal{C}$, y como $sop(\mathbf{x}) \subseteq R$ se tiene que $\mathbf{x}' \cdot \mathbf{x} = \sum_{j \in R} x_j' x_j = \sum_{j \in R} y_j x_j = 0$. □

Notemos que $\mathbf{c}_i \in \langle \mathbf{c}_j : j \in R \rangle \Leftrightarrow dim(\mathcal{C}[R]) = dim(\mathcal{C}[\overline{R}])$, donde $\overline{R} = R \cup \{i\}$. Esto quiere decir que el conjunto de recuperación no depende de la matriz generatriz G escogida. En este caso, existirá $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{F}_q^n$ tal que $\sum w_j \mathbf{c}_j = 0$, con $w_i \neq 0$, $w_j = 0 \ \forall j \notin \overline{R} \Rightarrow \mathbf{w} \in \mathcal{C}^{\perp}$, con $sop(\mathbf{w}) \subseteq \overline{R} \Rightarrow \mathbf{w}_{\overline{R}} \in \mathcal{C}^{\perp}[[\overline{R}]]$, luego:

Lema 4.1.6. *R es conjunto de recuperación para i si y solo si:*

$$Existe \quad \boldsymbol{w}_{\overline{R}} \in C^{\perp}[[\overline{R}]] \ con \ w_i \neq 0$$

En este caso, $\# \ R \geq d(C^{\perp}) - 1$

Demostración. La última desigualdad se cumple, puesto que $w(\mathbf{w}) \leq \#R + 1$, ya que $sop(\mathbf{w}) \subseteq \overline{R}$ y $w_i \neq 0$, y como $d(C^{\perp}) \leq w(\mathbf{w})$, se deduce que $d(C^{\perp}) \leq \#\overline{R} + 1$. □

Definición 4.1.7. Diremos que una coordenada i tiene localidad r_i, donde:

$$r_i = inf\{\#R : R \text{ conjunto de recuperación de } i\}$$

La localidad de un código $\mathcal{C}$ se define como $r = Max_{1 \leq i \leq n}\{r_i\}$.

Una palabra $\mathbf{w}_{\overline{R}} \in \mathcal{C}^{\perp}[[\overline{R}]]$, con $w_i \neq 0$, no solo aporta un conjunto de recuperación, sino también un método de recuperación de x_i. Para cada $\mathbf{x} \in \mathcal{C}$, $\mathbf{w} \cdot \mathbf{x} = 0$, ya que $\mathbf{w} \in \mathcal{C}^{\perp}$. Además $\mathbf{w} \cdot \mathbf{x} = \mathbf{w}_{\overline{R}} \cdot \mathbf{x}_{\overline{R}}$, pues $sop(\mathbf{w}) \subseteq \overline{R}$, luego $\mathbf{w}_{\overline{R}} \cdot \mathbf{x}_{\overline{R}} = w_i \cdot x_i + \mathbf{w}_R \cdot \mathbf{x}_R = 0 \Rightarrow x_i = -\frac{\mathbf{w}_R \cdot \mathbf{x}_R}{w_i}$.
En caso de fallo, un error sin detectar en algún x_j, $j \in R$, nos llevará a una recuperación errónea de x_i. Podríamos detectar ese error utilizando el código $\mathcal{C}$ al completo, lo que iría en contra del carácter local del método. Otra opción está en tomar más conjuntos de recuperación, pero esto aumentaría las posibilidades de error sin determinarnos dónde se encuentra este.

4.2. Códigos Localmente Recuperables con Detección de Errores

En esta sección modificaremos ligeramente la definición de conjunto de recuperación para permitir la detección de errores.

Lema 4.2.1. *$d(\mathcal{C}) \geq d \Leftrightarrow \forall S \subseteq \{1, \ldots, n\}$ con $\#S > n - d$, se tiene que $dim(\mathcal{C}[S]) = dim(\mathcal{C})$.*

Demostración. Veamos primero la implicación a la derecha. Veremos que $dim(\mathcal{C}[S]) = dim(\mathcal{C})$ observando que $\#\mathcal{C} = \#\mathcal{C}[S]$. Sean $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, con $\mathbf{x} \neq \mathbf{y}$. Puesto que, por hipótesis, $d(\mathcal{C}) \geq d$, $\mathbf{x}$ e $\mathbf{y}$ tienen al menos d coordenadas distintas, o equivalentemente tienen, a lo sumo, $n - d$ coordenadas iguales. Ahora bien, $\#S > n - d$, y $\pi_S(\mathbf{x})$ y $\pi_S(\mathbf{y})$ tienen $\#S$ coordenadas, luego necesariamente $\pi_S(\mathbf{x}) \neq \pi_S(\mathbf{y})$, para todo $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ con $\mathbf{x} \neq \mathbf{y}$, luego $\#\mathcal{C} = \#\mathcal{C}[S]$.
Para la otra implicación, veamos el contrarrecíproco. Supongamos $dim(\mathcal{C}[S]) < dim(\mathcal{C}) \Rightarrow$ existen $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ con $\mathbf{x} \neq \mathbf{y}$, tal que $\pi_S(\mathbf{x}) = \pi_S(\mathbf{y})$, luego $\mathbf{x}$ e $\mathbf{y}$ tienen al menos $\#S$

coordenadas iguales, o equivalentemente, como mucho $n-\#S$ distintas $\Rightarrow d(\mathcal{C}) \leq n-\#S$. Además, por hipótesis, $n-d<\#S$, luego $d(\mathcal{C}) \leq n-\#S < n+d-n = d \Rightarrow d(\mathcal{C}) < d$, como queríamos ver. □

Proposición 4.2.2. *Un conjunto $\overline{R}\backslash\{i\}$ es de recuperación para cada coordenada $i \in \overline{R} \Leftrightarrow 1 < d(\mathcal{C}[\overline{R}])$.*

Demostración. $\overline{R}\backslash\{i\} = R$ es conjunto de recuperación para i $\Leftrightarrow \mathbf{c}_i \in \langle \mathbf{c}_j, j \in R\rangle \Leftrightarrow dim(\mathcal{C}[R]) = dim(\mathcal{C}[\overline{R}])$. Por el lema previo, y teniendo en cuenta que $\#R = \#\overline{R} - 1$, se concluye que $1 < d(C[\overline{R}])$. □

Definición 4.2.3. Diremos que un conjunto $R \subseteq \{1,\ldots,n\}$ es un conjunto de recuperación con detección de $t \geq 0$ errores para una coordenada $i \notin R$ si verifica:

$$t+1 < d(C[\overline{R}]), \text{ donde } \overline{R} = R \cup \{i\}.$$

De forma abreviada diremos que R es un conjunto t-edr, de las siglas en inglés *t-error detecting recovery set*.

Por tanto, un conjunto con detección de t errores de recuperación para una coordenada i de R es en particular un conjunto de recuperación de i. Además, $R \cup \{i\}\backslash\{j\}$ es también un conjunto t-edr, para todo $j \in R$. Notemos que, si $\#R = r$, acorde con el lema previo, R es un t-edr conjunto para $i \Leftrightarrow dim(\mathcal{C}[S]) = dim(\mathcal{C}[\overline{R}]), \forall S \subseteq \overline{R}$ con $\#S \geq r-t$. En particular, $dim(\mathcal{C}[\overline{R}]) \leq r-t$.
Por otro lado, $t+1 < d(\mathcal{C}[\overline{R}]) \Rightarrow t+1 \leq d(\mathcal{C}[R])$, luego hasta t errores en cualquier palabra $\mathbf{x}_R$, con $\mathbf{x} \in \mathcal{C}$ serían detectados. Por tanto, cuando ocurren a lo sumo t errores en $\mathbf{x}_R$, un conjunto t-edr R o bien detecta estos errores o bien recupera el valor correcto de x_i.

Definición 4.2.4. Definimos la t-localidad de una coordenada i como el entero $min\{\#R :$ R conjunto t-edr para $i\}$. Se define entonces la t-localidad de $\mathcal{C}$ como:

$$r_t(\mathcal{C}) = r_t = max\{\text{t-localidad de } i : 1 \leq i \leq n\}.$$

Diremos que un código $\mathcal{C}$ es localmente recuperable con detección de t errores si para cada coordenada i existe un conjunto de recuperación que detecte t errores. De forma abreviada diremos que $\mathcal{C}$ es un código t-LREDC, de las siglas en inglés t-*Local Recovery Error Detecting Code*. Notemos que todo código $\mathcal{C}$ tal que $t+1 < d = d(\mathcal{C})$ es un t-LREDC, sin más que tomar $\overline{R} = \{1,\ldots,n\}$.

Ejemplo 4.2.5. Si *pinchamos* un código MDS un número menor de d veces obtenemos de nuevo un código MDS de la misma dimensión.
La t-localidad de un código MDS de parámetros $[n,k,d]$ con $t+1<d$ es $r_t = k+t$.

Introducimos una definición que utilizaremos en el siguiente resultado:

Definición 4.2.6. Sea $\mathcal{C}$ un código lineal de parámetros $[n,k]$. Para cualquier $r \leq k$, el r-ésimo peso generalizado de Hamming de $\mathcal{C}$ se define como:

$$d_r(\mathcal{C}) = min\{\#sop(D) : D \subseteq \mathcal{C}, \text{ subespacio vectorial con } dim(D) = r\}.$$

Recordamos que $\#sop(D) = w(D)$, peso de Hamming de D, donde $sop(D) = \{i : \text{ tal que existe } \mathbf{x} \in D, x_i \neq 0\}$. Es inmediato notar que para $r = 1$, $d_1(\mathcal{C})$ coincide con la distancia mínima del código. Las dos siguientes proposiciones proporcionan cotas para $r_t(\mathcal{C})$.

Proposición 4.2.7. *Sea $\mathcal{C}$ un código t-LREDC de parámetros $[n, k, d]$. La t-localidad de $\mathcal{C}$, r_t, verifica:*

$$r_t(\mathcal{C}) \geq d_{t+1}(\mathcal{C}^\perp) - 1$$

donde $d_{t+1}(\mathcal{C}^\perp)$ es el $(t+1)$-ésimo peso generalizado de Hamming de $\mathcal{C}^\perp$.

Demostración. Sea R un t-edr conjunto para una coordenada i. La condición $dim(\mathcal{C}[\overline{R}]) \leq \#R - t \Rightarrow dim(\mathcal{C}^\perp[\overline{R}]) \geq t + 1 \Rightarrow \#R \geq d_{t+1}(\mathcal{C}^\perp) - 1$, para cada coordenada, luego $r_t(\mathcal{C}) \geq d_{t+1}(\mathcal{C}^\perp) - 1$. □

La siguiente cota generaliza la dada en el teorema 2.2.1:

Proposición 4.2.8. *Sea $\mathcal{C}$ un código r-LREDC de parámetros $[n, k, d]$, entonces la t-localidad de $\mathcal{C}$, r_t, verifica:*

$$n + t + 2 \geq k + d + \left\lceil \frac{k}{r_t - t} \right\rceil (t + 1).$$

Demostración. La demostración de este resultado es similar a la del teorema 2.2.1, teniendo en cuenta los comentarios hechos tras la definición 4.2.3. □

De forma análoga al caso $t = 0$, diremos que nuestro código $\mathcal{C}$ es t-óptimo si su t-localidad r_t alcanza la igualdad en la ecuación anterior.

4.3. Ejemplo de Código LREDC

En esta sección construiremos un LREDC a partir de un código Reed-Solomon.
Como veremos a continuación, $RS(\mathbb{F}_q, m)^\perp = RS(\mathbb{F}_q, q - m - 2)$. Además, por ser un código MDS su localidad es precisamente k ($r_0 = k$), pues toda coordenada puede ser recuperada a partir de otras k cualquiera coordenadas.
Veamos que efectivamente $RS(\mathbb{F}_q, m)^\perp = RS(\mathbb{F}_q, q - m - 2)$. Sabemos que $RS(\mathbb{F}_q, m)$ tiene parámetros $[q, m + 1]$, luego $RS(\mathbb{F}_q, m)^\perp$ tendrá parámetros $[q, q - (m + 1)]$, por ser el espacio ortogonal asociado. Además, el código dual de un Reed-Solomon será también un Reed-Solomon. Luego siguiendo la notación utilizada antes, $RS(\mathbb{F}_q, m)^\perp = RS(\mathbb{F}_q, q - m - 2)$.
Sea $\mathcal{P} = \{p_1, \dots p_n\} \subseteq \mathbb{F}_q$. Consideramos el código $\mathcal{C} = RS(\mathcal{P}, k - 1) = \{(f(p_1), \dots, f(p_n)) : f \in \mathbb{F}_q[x]_{\leq k-1}\}$, de longitud $n = \#\mathcal{P}$ y dimensión $k \leq n - 2$. Sea $r = k + 1$. Tomamos $\overline{R}$, conjunto de $r + 1$ coordenadas con $\overline{\mathcal{R}} \subseteq \mathcal{P}$, entonces $\mathcal{C}[\overline{R}] = \{(f(p_{i_1}), \dots, f(p_{i_{r+1}})) : f \in \mathbb{F}_q[x]_{\leq k-1}\} = \pi_{\overline{R}}(\mathcal{C}) = RS(\overline{\mathcal{R}}, k-1)$, con $d(\mathcal{C}[\overline{R}]) = 3$. Puesto que estamos trabajando con un código RS, obtenemos esta distancia mínima simplemente despejando de la igualdad de Singleton para códigos MDS. Vemos también que $R = \overline{R} \setminus \{i\}$ es un conjunto de recuperación con detección de 1 error para cada $i \in \overline{R}$. Veamos cómo se lleva a cabo el proceso de recuperación, incluyendo la detección de errores:

Recordando lo visto en el Lema 4.1.5, se tiene que $\mathcal{C}[\overline{R}]^{\perp} = RS(\mathbb{F}_q, q-r)[[\overline{R}]]$, con $dim(C[\overline{R}]^{\perp}) = 2$, y $\mathcal{C}[R]^{\perp} = RS(\mathbb{F}_q, q-r)[[R]]$, con $dim(\mathcal{C}[R]^{\perp}) = 1$. Definimos ahora:

$$F(x) = \prod_{\gamma \in \mathbb{F}_q - \overline{\mathcal{R}}} (x - \gamma) \in \mathbb{F}_q[x]_{\leq q-r-1}. \tag{4.1}$$

Asumimos que la coordenada i-ésima se corresponde con el elemento $p_i \in \mathcal{P}$. Sea $\mathcal{R} = \overline{\mathcal{R}} \backslash \{p_i\}$, definimos los siguientes vectores:

$$\mathbf{z}_R = ev_{\mathcal{R}}((x - p_i)F(x)),\ \mathbf{w}_{\overline{R}} = ev_{\overline{\mathcal{R}}}(F(x)) \tag{4.2}$$

Así, $\langle \mathbf{z}_R \rangle = \mathcal{C}^{\perp}[[R]] = \mathcal{C}[R]^{\perp}$, $\mathbf{w}_{\overline{R}} \in \mathcal{C}^{\perp}[[\overline{R}]] = C[\overline{R}]^{\perp}$, con $w_i \neq 0$.
Sea $\mathbf{x} \in \mathcal{C}$ una palabra del código con un borrón en x_i, y a lo sumo un error en $\mathbf{x}_R$. El vector $\mathbf{z}_R$ permite detectar errores en $\mathbf{x}_R$, mientras que $\mathbf{w}_{\overline{R}}$ nos permite recuperar la coordenada x_i.
Suponiendo que $\mathbf{x}_R$ tiene como mucho un error, y teniendo en cuenta que $\langle \mathbf{z}_R \rangle = \mathcal{C}[R]^{\perp}$, se tiene que $\mathbf{z}_R \cdot \mathbf{x}_R \neq 0$ si $\mathbf{x}_R$ tiene 1 error, mientras que $\mathbf{z}_R \cdot \mathbf{x}_R = 0$ si $\mathbf{x}_R$ no contiene errores. En este último caso, podemos recuperar x_i calculando $x_i = -\frac{\mathbf{w}_R \cdot \mathbf{x}_R}{w_i}$. Notemos que este proceso no requiere de interpolación polinómica.
Se puede ver este ejemplo con mayor calidad de detalle en la sección 5.3 de este trabajo.
La 1-localidad de los códigos RS para los códigos de dimensión k es $r_1 = k + 1$. De hecho se alcanzan igualdades en las cotas dadas en las proposiciones previas, luego se dice que los códigos RS son 1-óptimos. Un razonamiento similar prueba que ocurre lo mismo en general para códigos MDS.
Cuando q es grande con respecto a r (como es deseable), el cálculo de $F(p)$ para $p \in \overline{\mathcal{R}}$ se puede hacer de forma más eficiente, desde el punto de vista computacional, notando que $\prod_{\lambda \in \mathbb{F}_q^*} \lambda = -1$. Comprobemos este último resultado.

Lema 4.3.1. *Consideramos $\mathbb{F}_q$ cuerpo finito de q elementos, y sea $\mathbb{F}_q^* = \mathbb{F}_q \backslash \{0\}$, entonces se tiene que:*

$$\prod_{\lambda \in \mathbb{F}_q^*} \lambda = -1.$$

Demostración. Nótese que $\mathbb{F}_q$ es un cuerpo, luego todo elemento no nulo tiene inverso, además, $a \in \mathbb{F}_q^*$ es inverso de si mismo si y solo si $a^2 - 1 = 0 \Leftrightarrow a = \pm 1$. Distingamos ahora dos casos:
Si q es un número par, la característica del cuerpo es 2, luego $1 = -1$, por lo tanto para todo $a \in \mathbb{F}_q^* \backslash \{1\}$ existe $a^{-1} \in \mathbb{F}_q^* \backslash \{1\}$, con $a^{-1} \neq a$, donde a^{-1} denota el inverso multiplicativo de a. Por tanto se tiene que:

$$\prod_{\lambda \in \mathbb{F}_q^*} \lambda = 1 \cdot \prod_{\lambda \in \mathbb{F}_q^* \backslash \{1\}} \lambda = 1 \cdot 1 = 1 = -1.$$

Si q es un número impar, $1 \neq -1$, por lo tanto para todo $a \in \mathbb{F}_q^* \backslash \{1, -1\}$ existe $a^{-1} \in \mathbb{F}_q^* \backslash \{1, -1\}$, con $a^{-1} \neq a$. Por lo tanto se tiene que:

$$\prod_{\lambda \in \mathbb{F}_q^*} \lambda = -1 \cdot 1 \cdot \prod_{\lambda \in \mathbb{F}_q^* \backslash \{1, -1\}} \lambda = -1 \cdot 1 = -1.$$

□

Sea entonces $\phi(p) = \prod_{\gamma \in \overline{\mathcal{R}}, \gamma \neq p} (p - \gamma)$.
El cálculo de $\phi(p)$ requiere de $r = \#\overline{\mathcal{R}} - 1$ multiplicaciones, en lugar de las $q - r - 1$ que requería $F(p)$ originalmente. Una vez obtenido el valor de $\phi(p)$, no es más que notar que $F(p) = -\phi(p)^{-1}$, y las operaciones se reducen al cálculo de un inverso multiplicativo. De este modo, la complejidad computacional del proceso de recuperación con detección de errores se reduce considerablemente.
Si para cada coordenada i arreglamos el correspondiente $\overline{R}$, entonces podremos calcular y almacenar los respectivos $\mathbf{w}_{\overline{R}}$. De ello podremos deducir $\mathbf{z}_R$ y completar el proceso de recuperación.

4.4. Conclusión

En este capítulo se ha propuesto una variación de los códigos localmente recuperables que además tienen la ventaja de detectar errores locales, mejorando así la fiabilidad y seguridad del sistema de recuperación.
Hemos visto más en detalle el caso de los códigos de tipo Reed-Solomon donde, en particular, es suficiente con añadir una coordenada más en el conjunto de recuperación para ser capaces de detectar errores.

Capítulo 5

Cálculos con software de álgebra computacional

5.1. Ejemplo 3.2.2

[11]:
```
#Ejemplo (3.2.3) de codigo LRC
n,k,r=9,4,2
F=GF(13)
R.<x>=F[]
A=matrix([[1,3,9],[2,6,5],[4,12,10]]) #Particion en forma de matriz
AListа=[]
for i in range(A.nrows()):
    for j in range(A.ncols()):
        AListа.append(A[i,j])          #Particion en forma de lista
#A1=[1,3,9]
#A2=[2,6,5]
#A3=[4,12,10]
v=(1,1,1,1)                            #Vector de ejemplo dado en el␣
 ↪capitulo 3
a=random_vector(F,k)
print(a,v)                             #Genero un mensaje aleatorio sin␣
 ↪codificar
```

```
(8, 11, 4, 12) (1, 1, 1, 1)
```

[12]:
```
#Construccion del polinomio bueno g(x) (figura (3.8))
g=[]
for j in range(A.nrows()):
    aux=1
    for i in range(len(A[j])):
        aux=aux*(x-A[j,i])
    g.append(aux)
#Tres polinomios buenos, uno para cada conjunto A_i de la particion
```

```
#Usaremos en su lugar x^3, ya que es valido para todos los 3␣
 ↪simultaneamente
g_final=x^3
print(g,g_final)
```

```
[x^3 + 12, x^3 + 5, x^3 + 1] x^3
```

[13]:
```
#Contruccion de f_a(x) polinomio codificador (depende del g(x) elegido)
f_a=[]
f_v=[]
for k in range(A.nrows()):
    faux=[]
    faux_v=[]
    for i in range(0,len(a),2):
        faux.append(a[i]+a[i+1]*g[k])
        faux_v.append(v[i]+v[i+1]*g[k])
    aux=0
    aux_v=0
    for j in range(len(faux)):
        aux=aux+(faux[j])*x^j;
        aux_v=aux_v+(faux_v[j])*x^j;
    f_a.append(aux)                          #Tres polinomios codificadores,␣
 ↪definen el mismo codigo
    f_v.append(aux_v)
f_a_final=0
f_v_final=0
faux=[]
faux_v=[]
for k in range(0,len(a),2):
    faux.append(a[k]+a[k+1]*g_final)
    faux_v.append(v[k]+v[k+1]*g_final)
aux=0
aux_v=0
for l in range(len(faux)):
    aux=aux+(faux[l])*x^l;
    aux_v=aux_v+(faux_v[l]*x^l)
f_a_final=aux
f_v_final=aux_v
print(f_a)
print(f_a_final)
print(f_v_final)                             #Utilizaremos el generado por␣
 ↪el g(x) generico
```

```
[12*x^4 + 11*x^3 + 5*x + 10, 12*x^4 + 11*x^3 + 12*x + 11, 12*x^4 +␣
 ↪11*x^3 + 3*x
+ 6]
12*x^4 + 11*x^3 + 4*x + 8
x^4 + x^3 + x + 1
```

[14]:
```
#Obtencion de la palabra codificada c
c=[]
for i in range(len(AListA)):
        c.append(f_a_final(AListA[i]))
c_v=[]
for j in range(len(AListA)):
        c_v.append(f_v_final(AListA[j]))
print(c,c_v)
```

```
[9, 2, 7, 10, 7, 11, 4, 5, 8] [4, 8, 7, 1, 11, 2, 0, 0, 0]
```

[15]:
```
#Obtención de delta, polinomio de recuperacion (depende del conjunto Ai
 ↪de recuperacion elegido)
delta=[]
for i in range(3):
    delta.append(c[3*i+1]*((x-A[i,2])/
 ↪(A[i,1]-A[i,2]))+c[3*i+2]*((x-A[i,1])/(A[i,2]-A[i,1])))
delta_v=[]
for j in range(3):
    delta_v.append(c_v[3*j+1]*((x-A[j,2])/
 ↪(A[j,1]-A[j,2]))+c_v[3*j+2]*((x-A[j,1])/(A[j,2]-A[j,1])))

#Tres deltas distintos, cada uno corrige para un conjunto Ai diferente
print(delta,delta_v)
```

```
[3*x + 6, 9*x + 5, 5*x + 10] [2*x + 2, 9*x + 9, 0]
```

[16]:
```
for i in range(len(AListA)):
    if i<r+1:
        print(delta[0](AListA[i]),delta_v[0](AListA[i]))
    elif i<2*(r+1):
        print(delta[1](AListA[i]),delta_v[1](AListA[i]))
    elif i<3*(r+1):
        print(delta[2](AListA[i]),delta_v[2](AListA[i]))
print(c,c_v) #Vemos como, con el delta apropiado, contruido a partir de
 ↪r=2 simbolos de la palabra codificada, permite recuperar cualquier
 ↪borron en c.
```

```
9 4
2 8
7 7
10 1
7 11
11 2
4 0
5 0
8 0
[9, 2, 7, 10, 7, 11, 4, 5, 8] [4, 8, 7, 1, 11, 2, 0, 0, 0]
```

5.2. Ejemplo 3.2.4

[1]:
```
#Ejemplo (3.2.4) de codigo LRC
n,k,r=12,6,3
F=GF(13)
R.<x>=F[]
A=matrix([[1,5,12,8],[2,10,11,3],[4,7,9,6]]) #Particion en forma de␣
 ↪matriz
AListA=[]
for i in range(A.nrows()):
    for j in range(A.ncols()):
        AListA.append(A[i,j])                #Particion en forma de␣
 ↪lista
#A1=[1,5,12,8]
#A2=[2,10,11,3]
#A3=[4,7,9,6]
a=random_vector(F,k)
a                                            #Mensaje aleatorio sin␣
 ↪codificar sobre F^k
```

[1]:
```
(8, 7, 11, 11, 4, 3)
```

[2]:
```
#Contruccion alternativa del polinomio codificador (figura (3.7))
f_a=0
for m in range(k+k/r-2+1):
    if m<r:
        f_a=f_a+a[m]*x^m
    if m>r:
        f_a=f_a+a[m-1]*x^m
f_a
```

[2]:
```
3*x^6 + 4*x^5 + 11*x^4 + 11*x^2 + 7*x + 8
```

[7]:
```
#Construccion del polinomio bueno g(x) (figura (3.8))
g=[]
for j in range(A.nrows()):
    aux=1
    for i in range(len(A[j])):
        aux=aux*(x-A[j,i])
    g.append(aux)
#Tres polinomios buenos, uno para cada conjunto A_i de la particion
g_final=x^4 #Utilizamos este en su lugar, valido para los 3 conjuntos␣
 ↪simultaneamente
print(g,g_final)
```

```
[x^4 + 12, x^4 + 10, x^4 + 4] x^4
```

[5]:
```
#Obtencion de la palabra codificada c a partir de cada polinomio␣
 ↪codificador
c=[]
for i in range(len(AListа)):
    c.append(f_a(AListа[i]))
c
```

[5]: [5, 8, 9, 2, 3, 8, 5, 5, 3, 8, 10, 4]

[8]:
```
#Obtencion de delta, polinomio de recuperacion (depende del conjunto de␣
 ↪recuperacion elegido)
delta=[]
for i in range(A.nrows()):
    delta.append(c[4*i+1]*((x-A[i,2])/(A[i,1]-A[i,2]))*((x-A[i,3])/
 ↪(A[i,1]-A[i,3]))+c[4*i+2]*((x-A[i,1])/(A[i,2]-A[i,1]))*((x-A[i,3])/
 ↪(A[i,2]-A[i,3]))+c[4*i+3]*((x-A[i,1])/(A[i,3]-A[i,1]))*((x-A[i,2])/
 ↪(A[i,3]-A[i,2])))
delta

#Tres deltas distintos, cada uno corrige para un conjunto Ai diferente
```

[8]: [x^2 + 11*x + 6, 7*x^2 + 6*x + 2, 12*x^2 + 4*x + 3]

[9]:
```
for i in range(len(AListа)):
    if i<r+1:
        print(delta[0](AListа[i]))
    elif i<2*(r+1):
        print(delta[1](AListа[i]))
    elif i<3*(r+1):
        print(delta[2](AListа[i]))
c #Vemos como, con el delta apropiado, construido a partir de r=3␣
 ↪simbolos de la palabra codificada, permite recuperar cualquier␣
 ↪borron en c.
```

```
5
8
9
2
3
8
5
5
3
8
10
4
```

[9]: [5, 8, 9, 2, 3, 8, 5, 5, 3, 8, 10, 4]

5.3. Ejemplo de Código LREDC 4.3

[1]:
```
q,m=13,5                          #aqui se pueden generalizar los␣
 ↪parametros del código
F=GF(q)
R.<x>=F[]
n=q                               #como simplificacion, tomamos n=q (->␣
 ↪P = F_q, conjunto de puntos de evaluacion)
k=m+1
baseR=[]
for l in range(k):
    baseR.append(x^l)             #base de R.<x>, grado <=5 (para la␣
 ↪contruccion de G y C)
r=k+1
nLista=[0..q-1]
SR=Subsets(nLista,r+1)

R_=SR.random_element()            #tomamos un conjunto de recuperacion␣
 ↪cualquiera
LR_=R_.list()

i=R_.random_element()             #coordenada aleatoria, la cual␣
 ↪recuperaremos
Si=Set([i])

R=R_.difference(Si)
LR=R.list()

#construccion de G, matriz generatriz del codigo
G=[]
for m in range(k):
    Gaux=[]
    for s in range(n):
        Gaux.append(baseR[m](s))
    G.append(Gaux)
G=matrix(F,G)
#codigo asociado a la matriz G
C=LinearCode(G)
print(i,LR_,LR)                   #mostramos coordenada a recuperar y␣
 ↪el conjunto de recuperacion que usaremos
C
```

```
4 [2, 3, 4, 6, 7, 8, 9, 10] [2, 3, 6, 7, 8, 9, 10]
```

[1]:
```
[13, 6] linear code over GF(13)
```

[2]:
```
c=C.random_element();             #elemento aleatorio de nuestro codigo
```

```
print(i,c[i])                    #mostramos coordenada y simbolo de la␣
 ↪palabra asociado que queremos recuperar
c
```

```
4 8
```

[2]:
```
(4, 11, 5, 5, 8, 11, 7, 7, 10, 12, 2, 10, 12)
```

[3]:
```
#calculo de F(x) (figura(4.1))
Fp=1
for j in range(q):
    if j not in LR_:
        Fp=Fp*(x-j)
#construccion de los vectores z y w que utilizaremos en el proceso␣
 ↪(figura(4.2))
z=[]
w=[]
wi=0
for j in range(len(LR_)):
        if LR_[j] == i:
            wi=Fp(LR_[j])                #almacenamos este valor,␣
 ↪necesario despues
        w.append(Fp(LR_[j]))
for l in range(len(LR)):
        z.append((LR[l]-i)*Fp(LR[l]))
print(z,w)
```

```
[1, 6, 6, 9, 4, 8, 5] [6, 7, 4, 3, 3, 1, 12, 3]
```

[4]:
```
#comprobacion de errores en el conjunto de recuepracion
err=0
for j in range(len(LR)):
    err=err+z[j]*c[LR[j]]
print(err)                                  #resultado del producto␣
 ↪z_R por c_R
if err == 0:
    print('no hay errores en R')
else:
    print('error en alguna coordenada de R')
```

```
0
no hay errores en R
```

[5]:
```
#recuperacion de ci
ci=0
for j in range(len(LR_)):
    if LR_[j] != i:
        ci=ci+w[j]*c[LR_[j]]
ci=-ci/wi
```

```
print(i,ci)
```

```
4 8
```

Bibliografía

[1] P. Gopalan, C. Huang, H. Simitci and S. Yekhanin. *On the locality of codeword symbols*, IEEE Trans. Inf. Theory, vol. 58, no. 11, pp. 6925-6934, Nov. 2012.

[2] I. Tamo and A. Barg. *A family of optimal locally recoverable codes*, IEEE Trans. Inf. Theory, vol. 60, no. 8, pp. 4661-4676, Aug. 2014.

[3] C. Munuera. *Locally recoverable codes with local error detection*, arXiv:1812.00834, Dec. 2018.

[4] C. Huang, M. Chen and J. Li. *Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems*, IEEE Interational Symposium on Network Computing and Applications, pages 79-86, Cambridge, Massachusetts, Jul. 2007.

[5] C. Munuera and J. Tena. *Codificación de la Información*, Universidad de Valladolid, 1997.

[6] V. Ramkumar et al. *Codes for distributed storage*, Chapter 31 in W. Huffman, J. Kim and P. Solé. *Concise Encyclopedia of Coding Theory*, Boca Raton, Mar. 2021.

[7] N. Silberstein, A. S. Rawat, O. O. Koyluoglu and S. Vishwanath. *Optimal locally repairable codes via rank-metric codes*, in Proc. IEEE ISIT, pp. 1819-1823, Jul. 2013.

[8] I. Tamo, D. S. Papailiopoulos and A. G. Dimakis. *Optimal locally repairable codes and connections to matroid theory*, in Proc. IEEE ISIT, no. 11, pp. 1814-1818, Jul. 2013.

[9] M. Tsfasman, S. Vladut and D. Nogin. *Algebraic Geometric Codes: Basic Notations*, Providence, RI: Amer. Math. Soc., 2013.