Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería de la Computación

Algoritmo eficiente para el cálculo de las distribuciones de pesos de todos los códigos cíclicos irreducibles sobre campos finitos pequeños

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

Maestro en Ciencia e Ingeniería de la Computación

PRESENTA:

Félix Alejandro Hernández Fuentes

DIRECTOR DE TESIS:

Dr. Gerardo Vega Hernández

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Ciudad Universitaria, Cd. de México. Octubre 2018

A mis papás Blanca y Lorenzo,
a mi hermana Jenny,
a mi esposa Paulina,
con mucho cariño.

Resumen

En teoría de códigos, un problema interesante pero al mismo tiempo difícil es determinar la distribución de pesos de un código dado. Esto es importante ya que al conocer esta distribución podemos determinar las capacidades de detección y corrección de errores del código.

El problema resulta aún más interesante para códigos cíclicos pues estos poseen una rica estructura algebraica que puede ser usada de muchas formas. Por ejemplo, en telecomunicaciones esta estructura es empleada comúnmente para diseñar algoritmos de codificación y decodificación eficientes.

De entre los códigos cíclicos existen los códigos cíclicos irreducibles, que son aquellos códigos cuyo polinomio de chequeo de paridad es irreducible. Estos códigos han sido ampliamente utilizados a lo largo de las últimas décadas siendo la exploración espacial uno de sus más notables usos.

En el presente trabajo se utiliza la riqueza algebraica que poseen estos códigos para presentar un algoritmo que determina de manera eficiente TODAS las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión k, con k pequeña, definidos sobre un campo finito pequeño con q elementos $\mathbb{F}_q$.

El algoritmo hace uso de la identidad de McEliece [4, Lema 2.8] con la cual es posible calcular los pesos de los códigos cíclicos irreducibles a través de una suma exponencial particular.

En el trabajo se incluye también el código fuente en lenguaje de programación C de una implementación del algoritmo propuesto.

Palabras clave: *Códigos cíclicos irreducibles, distribución de pesos, sumas exponenciales.*

Agradecimientos

A mis padres y mi hermana, por el constante apoyo recibido durante toda mi vida, por el cariño, la paciencia y sus cuidados.

A Paulina, por siempre estar ahí, por todo el amor y los consejos, por permitirme ser tu compañero (ahora de por vida) a lo largo de estos últimos 8 años.

A mis amigos de *La Jaula*, por tantos buenos momentos y por su amistad.

Al Dr. Gerardo Vega, por la oportunidad, la confianza y por todas sus enseñanzas a lo largo de estos últimos dos años.

A mis profesores y compañeros del posgrado, por su dedicación, apoyo, enseñanzas y su amistad.

A mis sinodales, por el valioso tiempo que dedicaron a la revisión y corrección de este trabajo.

A la Universidad Nacional Autónoma de México (UNAM) y al Posgrado en Ciencia e Ingeniería de la Computación, por haberme brindado la oportunidad de desarrollar mis estudios de posgrado.

Al Consejo Nacional de Ciencia y Tecnología, por el financiamiento económico recibido para la realización de mis estudios de posgrado.

Tambien, agradezco el apoyo parcial del PAPIIT-UNAM IN109818 para la realización de este trabajo.

Índice general

Capítulo 1

Introducción

Un problema fundamental en las comunicaciones es el de reconstruir en un punto (receptor) la información trasmitida desde otro punto (emisor). La información se envía a través de un canal de comunicación (por ejemplo la línea telefónica, un CD, radio, etc.) el cual comúnmente introduce errores en ella al momento de su transmisión o almacenamiento debido a distintos factores como pueden ser: el ruido en la línea telefónica, daño en el CD, radiación de distintas fuentes, etc. En este sentido la teoría de códigos tiene como objetivo principal proteger la información contra errores que de manera natural e inevitable la modifican. Esto lo realiza a través del estudio de las propiedades de los códigos correctores detectores de error.

Una familia de códigos correctores detectores de error de gran interés e importancia son los llamados códigos cíclicos los cuales han sido ampliamente usados en productos de electrónica de consumo, tecnologías de transmisión de datos, sistemas de difusión, entre otros. Esto es así ya que dichos códigos poseen una rica estructura algebraica que, entre otras cosas, permite el diseño de algoritmos de codificación y decodificación eficientes.

Ejemplos de códigos cíclicos son: los códigos BCH utilizados comúnmente en la comunicación satelital, en los códigos de barra de dos dimensiones (código QR), en el almacenamiento de datos (discos de estado sólido); los códigos LCD recientemente utilizados en criptografía; los códigos Reed-Solomon utilizados en la telefonía móvil, en la transmisión digital de radio y televisión.

Un problema matemático de interés es utilizar esta riqueza algebraica que poseen los códigos cíclicos para determinar la distribución de sus pesos, hecho que consiste en clasificar a todos los vectores de un código[1] de acuerdo a su peso de Hamming[2], lo cual resulta importante ya que al conocer esta distribución es posible determinar las capacidades de detección y corrección de errores del código. Un ejemplo de este concepto se muestra a continuación.

Ejemplo 1. Sea $\mathcal{C}$ el código cíclico generado por el polinomio $x^2 + 2$ definido sobre el campo primo[3] $\mathbb{F}_3 = \{0, 1, 2\}$, de longitug 4 y dimensión 2. Como $\mathcal{C}$ es un código

[1] En el contexto de teoría de códigos, a los vectores que forman parte de un código se les llama palabras de código.

[2] Para un vector v con un número finito de coordenadas, el peso de Hamming de v está dado por el número de coordenadas distintas de cero en v.

[3] Se dice que un campo finito es primo si éste no tiene subcampos propios.

lineal tiene asociado una matriz generadora la cual está dada por:

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

Utilizando a G para el cálculo de todas las palabras de código de $\mathcal{C}$, se tiene que:

$$\mathcal{C} = \{(0,0,0,0), (0,1,0,2), (1,0,2,0), (0,2,0,1), (2,0,1,0) \\ (1,1,2,2), (1,2,2,1), (2,2,1,1), (2,1,1,2)\}.$$

Entonces, la distribución de pesos para $\mathcal{C}$ es:

$$A_0 = 1, A_2 = 4, A_4 = 4,$$

es decir, existe una palabra de código con peso de Hamming 0, 4 de peso 2 y 4 de peso 4. Y así, la distancia mínima para el código terciario $\mathcal{C}$ es 2, lo cual implica que este código es capaz de detectar un error pero no de corregirlo.

Hay que notar que $\mathcal{C}$ es ejemplo de un código cíclico irreducible, que son aquellos códigos cíclicos cuyo polinomio de chequeo de paridad es irreducible y, además, son los de interés para este trabajo. Cabe mencionar que estos códigos de bloque han sido ampliamente utilizados a lo largo de las últimas décadas siendo la exploración espacial uno de sus más notables usos.

Volviendo al Ejemplo 1, en resumen, para obtener la distribución de pesos de un código dado $\mathcal{C}$ lo que regularmente se hace es: calcular la matriz generadora asociada al código, con ella se calculan todas las palabras de código de $\mathcal{C}$, y posteriormente se obtiene el peso de Hamming de cada una de ellas para así clasificarlas. Note que este método resulta impráctico. Si se tuviera un código con miles de palabras de código, obtener la distribución de sus pesos se volvería una tarea muy costosa.

El objetivo de este trabajo es presentar un algoritmo para poder determinar, sin la necesidad de invertir recursos en el cálculo de palabras de código, TODAS las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión k, con k pequeña, definidos sobre un campo finito pequeño con q elementos $\mathbb{F}_q$.

Por un resultado de Schmidt y White [4, Observación 2.10] se verá que para el algoritmo que se propone en este trabajo será suficiente restringirse al caso en que q es un número primo pues el caso en que q es una potencia de un primo se puede extender fácilmente a partir del primero.

El algoritmo se basa en un resultado de Robert J. McEliece [4, Lema 2.8] con el cual es posible calcular los pesos de los códigos cíclicos irreducibles a través de una suma exponencial particular. Suma que, como se verá más adelante, no resulta trivial de evaluar.

También se presenta una implementación del algoritmo propuesto en lenguaje de programación C que, además de obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles, regresa el polinomio de chequeo de paridad asociado a cada código para que, si así se decide, éste pueda ser implementado.

El programa antes mencionado toma sólo dos parámetros para realizar los cálculos: un número primo p pequeño y un polinomio primitivo de grado k, con k pequeña, definido sobre el campo primo con p elementos $\mathbb{F}_p$.

Aunque el algoritmo es propio para campos finitos en general, el programa que aquí se presenta funciona de manera correcta y eficiente cuando los códigos son de dimensión pequeña y además están definidos sobre campos primos pequeños. De otra forma el programa podría tomar mucho tiempo para finalizar su ejecución y arrojar los resultados correctos. Un ejemplo del posible uso de esta herramienta se muestra a continuación.

Ejemplo 2. Suponga que se desea trabajar con un código cíclico irreducible de dimensión 4 definido sobre $\mathbb{F}_7$, el campo primo con 7 elementos. Además suponga que se desea que el código tenga pocos pesos (códigos de gran importancia práctica en criptografía ya que son útiles en el desarrollo de esquemas de intercambio de secretos). Entonces, basta con dar al programa los valores de entrada 7 y un polinomio primitivo de grado 4 definido sobre $\mathbb{F}_7$, por ejemplo $x^4 + x^3 + 6x + 3$, para que éste devuelva en cuestión de pocos segundos todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 4 definidos sobre un alfabeto con 7 elementos. Con base en la salida del programa se puede elegir el código cíclico irreducible que más se ajuste a las necesidades y, puesto que la herramienta devuelve el polinomio de chequeo de paridad asociado a cada código, es posible proceder a su implementación.

El presente trabajo está organizado de la siguiente manera: en el Capítulo 2 se establece la notación que se estará utilizando a lo largo de este escrito, también se describen los códigos cíclicos así como algunos de sus resultados más relevantes. En el Capítulo 3 se introducen las sumas exponenciales, herramienta con la cual fue desarrollado el algoritmo propuesto. El Capítulo 4 está dedicado a la descripción del algoritmo para el cálculo de la distribución de pesos de códigos cíclicos irreducibles sobre campos finitos pequeños. En la última parte de este trabajo se dan las conclusiones y se incluye un apéndice con el código fuente de la implementación del algoritmo.

Capítulo 2

Antecedentes

El presente Capítulo tiene como fin describir a los códigos cíclicos, algunos de sus resultados más relevantes, y también mencionar otras definiciones que serán de utilidad para el desarrollo de este escrito. A lo largo de este trabajo se usará la siguiente:

2.1. Notación

Mediante p, t, q, k y Δ, se denotarán cinco enteros positivos tales que p es un número primo, $q = p^t$ y $\Delta = (q^k - 1)/(q - 1)$. Si v y w son enteros tales que $\text{mcd}(v, w) = 1$ (donde $\text{mcd}(v, w)$ denota al máximo común divisor de v y w), entonces el entero más pequeño m para el cual $w^m \equiv 1$ (mód v) es llamado el *orden multiplicativo de* w módulo v y será denotado como $\text{ord}_v(w)$. Con γ se denotará a un elemento primitivo del campo finito con q^k elementos $\mathbb{F}_{q^k}$, es decir, $< \gamma >= \mathbb{F}_{q^k}^*$[1]. Para un entero positivo s se define como ξ_s a la s-ésima raíz compleja de la unidad $e^{2\pi\sqrt{-1}/s}$, es decir, $\xi_s := e^{2\pi\sqrt{-1}/s}$.

2.2. Grupos cíclicos

Como se mencionó, el objetivo de este trabajo es presentar un algoritmo para poder determinar todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles definidos sobre un campo finito pequeño. Un campo es una estructura algebraica que consta de dos subestructuras algebraicas, a saber, un grupo aditivo y un grupo multiplicativo, ambos conmutativos (ver, por ejemplo, [2, Capítulo 1, Definición 1.29, p. 11]). Cuando el campo es finito, su grupo multiplicativo resulta cíclico (ver, por ejemplo, [2, Capítulo 2, Teorema 2.8, p. 46]). Dicha estructura cíclica es utilizada constantemente a lo largo de este trabajo y es por ello que a continuación se enuncia su definición formal.

Definición 1. *[2, Capítulo 1, Definición 1.3, p. 3] Un grupo multiplicativo G es llamado cíclico si existe un elemento $a \in G$ tal que para cualquier $b \in G$ existe un entero j con $b = a^j$. Dicho elemento a es llamado el generador del grupo cíclico. Cuando esto suceda se escribirá $< a >= G$.*

[1] $\mathbb{F}_{q^k}^* := \mathbb{F}_{q^k} \backslash \{0\}$

El siguiente Teorema establece hechos importantes acerca de los grupos cíclicos.

Teorema 1. *[2, Capítulo 1, Teorema 1.15, p. 7]*

(1) Todo subgrupo de un grupo cíclico es cíclico.

(2) En un grupo cíclico finito $< a >$ de orden m, el elemento a^l, $l \in \mathbb{Z}$, genera un subgrupo de orden $m/\mathrm{mcd}(l, m)$.

(3) Sea m el orden del grupo cíclico finito $< a >$. Si d es un divisor positivo de m, entonces $< a >$ contiene un y sólo un subgrupo de índice d[2]. Para cualquier divisor positivo f de m, $< a >$ contiene precisamente un subgrupo de orden f.

Dicho lo anterior, se presentan a continuación los códigos cíclicos y algunos de sus resultados más importantes.

2.3. Códigos cíclicos

Sea n entero positivo y sean q y k como antes. Un $[n, k]$ código lineal $\mathcal{C}$ definido sobre el campo finito con q elementos $\mathbb{F}_q$ es un subespacio vectorial de $\mathbb{F}_q^n$ de dimensión k. Si además siempre que $(c_0, c_1, \ldots, c_{n-1}) \in \mathcal{C}$ entonces $(c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in \mathcal{C}$, el código es llamado *cíclico*.

A los vectores que forman parte de $\mathcal{C}$ se les llama *palabras de código*. Si $v \in \mathcal{C}$, entonces el *peso de Hamming* de v, $\mathrm{w}(v)$, es el número de coordenadas distintas de cero en v.

A continuación se enuncia la definición de distribución de pesos de un código.

Definición 2. *[3, Capítulo 2, p. 40] Sea $\mathcal{C}$ un $[n, k]$ código lineal sobre $\mathbb{F}_q$. Sea A_i el número de palabras de código con peso de Hamming i. Los números*

$$A_0 = 1, A_1, \ldots, A_n$$

son llamados la distribución de pesos de $\mathcal{C}$.

Observación 1. El valor más pequeño de i, $i > 0$, para el cual A_i es distinto de cero corresponde a la *distancia mínima* del código $\mathcal{C}$. Dicho parámetro determina las capacidades de corrección y detección de errores del código (ver, por ejemplo, [3, Capítulo 1, Teorema 2, p. 10]). Es por esta razón que el problema de determinar la distribución de pesos de un código es de gran importancia.

Un código lineal $\mathcal{C}$ de longitud n, dimensión k y distancia mínima d es llamado un $[n, k, d]$ código.

Ahora bien, si se identifica al vector $c = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ con el polinomio

$$c(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} \in \mathbb{F}_q[x]/(x^n - 1),$$

[2]Para un grupo cíclico finito $< a >$, un subgrupo de índice d es aquel cuyo generador está dado por el elemento a^d.

entonces cualquier código $\mathcal{C}$ de longitud n sobre $\mathbb{F}_q$ puede verse como un subconjunto del anillo cociente $\mathbb{F}_q[x]/(x^n-1)$. Así, un código lineal $\mathcal{C}$ será cíclico si y sólo si el subconjunto correspondiente en $\mathbb{F}_q[x]/(x^n-1)$ es un ideal del anillo $\mathbb{F}_q[x]/(x^n-1)$, es decir, si $\mathcal{C}$ es lineal y además si $c(x)$ está en $\mathcal{C}$ entonces también lo estará $xc(x)$ (ver [3, Capítulo 7, p. 189]).

2.3.1. Polinomio generador

Un tipo particular de ideal es el *ideal principal* que consiste de todos los múltiplos de un polinomio fijo $g(x)$ por elementos de $\mathbb{F}_q[x]/(x^n-1)$, éste se denota como

$$< g(x) >,$$

donde $g(x)$ es llamado el *polinomio generador* del ideal. De hecho todos los ideales de $\mathbb{F}_q[x]/(x^n-1)$ son principales, es decir, todo código cíclico tiene un polinomio generador. El siguiente es un Teorema bien conocido que establece esta y otras propiedades básicas acerca de los códigos cíclicos.

Teorema 2. *[3, Capítulo 7, Teorema 1, p. 190] Sea $\mathcal{C}$ un ideal distinto de cero en $\mathbb{F}_q[x]/(x^n-1)$, es decir, un código cíclico de longitud n definido sobre $\mathbb{F}_q$. Entonces:*

(a) Existe un único polinomio mónico $g(x)$ de grado mínimo en $\mathcal{C}$.

(b) $\mathcal{C} =< g(x) >$, es decir, $g(x)$ es el polinomio generador de $\mathcal{C}$.

(c) $g(x)$ divide al polinomio x^n-1.

(d) Todo $c(x) \in \mathcal{C}$ puede ser expresado de manera única como $c(x) = f(x)g(x)$ en $\mathbb{F}_q[x]$, con $f(x) \in \mathbb{F}_q[x]$, y si $\text{grado}(g(x)) = r$ entonces $\text{grado}(f(x)) \leq n-r-1$. La dimensión de $\mathcal{C}$ es $n-r$. Por tanto el mensaje $f(x)$ se convierte en la palabra de código $f(x)g(x)$.

(e) Si $g(x) = g_0 + g_1x + \cdots + g_rx^r$, entonces $\mathcal{C}$ es generado (como subespacio de $\mathbb{F}_q^n$) por las filas de la matriz generadora

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_r & & 0 \\ & g_0 & g_1 & \cdots & g_{r-1} & g_r & \\ & & & \cdots & \cdots & & \\ 0 & & g_0 & \cdots & \cdots & & g_r \end{bmatrix} = \begin{bmatrix} g(x) & & & & \\ & xg(x) & & & \\ & & & \cdots & \\ & & & & x^{n-r-1}g(x) \end{bmatrix}.$$

2.3.2. Polinomio de chequeo de paridad

Sea $\mathcal{C} =< g(x) >$ un código cíclico de longitud n definido sobre $\mathbb{F}_q$ con $\text{grado}(g(x)) = r$. Por el Teorema 2 se sabe que $g(x)$ divide a x^n-1, entonces

$$h(x) = (x^n-1)/g(x) = h_0 + h_1x + \cdots + h_{n-r-1}x^{n-r-1} + x^{n-r}$$

es llamado el *polinomio de chequeo de paridad* de $\mathcal{C}$. Note que la dimensión de $\mathcal{C}$ corresponde al grado de $h(x)$. Si $h(x)$ es irreducible entonces $\mathcal{C}$ es llamado un *código cíclico irreducible.*

2.3.3. Factorización de $x^n - 1$

Como el polinomio generador de un código cíclico de longitud n sobre $\mathbb{F}_q$ debe ser un factor del polinomio $x^n - 1$, es necesario estudiar la factorización de $x^n - 1$ e introducir el concepto de campo de descomposición de $x^n - 1$.

Observación 2. El polinomio $x^n - 1$ no tiene factores repetidos sobre $\mathbb{F}_q$ si y sólo si $\text{mcd}(n, q) = 1$ (ver [3, Capítulo 7, p. 196]). De esta manera, se asume a lo largo de este trabajo que $\text{mcd}(n, q) = 1$.

El siguiente resultado será útil para lo que se describe más adelante.

Lema 1. *[3, Capítulo 4, Lema 9, p. 102] Si $x, r, s \in \mathbb{Z}$, con $x \geq 2$, $r, s \geq 1$, entonces*

$$x^s - 1 \mid x^r - 1 \iff s \mid r.$$

Como una consecuencia directa del Lema anterior se tiene el siguiente:

Corolario 1. *Sea* $\text{ord}_n(q) = k$*, entonces*

$$x^n - 1 \mid x^{q^k - 1} - 1.$$

Pero $x^n - 1 \nmid x^{q^s - 1} - 1$ *para* $0 < s < k$.

Con lo anterior se tiene que las raíces de $x^n - 1$, que son llamadas las *n-ésimas raíces de la unidad*, viven en $\mathbb{F}_{q^k}$, la extensión de grado k de $\mathbb{F}_q$, y en ningún campo más chico.

Como ya se mencionó, $x^n - 1$ no tiene factores repetidos y por tanto tiene n raíces distintas. Así, existen n distintos elementos $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ en $\mathbb{F}_{q^k}$ (las n-ésimas raíces de la unidad) tal que

$$x^n - 1 = \prod_{i=0}^{n-1} (x - \alpha_i).$$

Por esta razón $\mathbb{F}_{q^k}$ es entonces llamado el *campo de descomposición* para $x^n - 1$.

Para hablar de la factorización de $x^n - 1$ sobre $\mathbb{F}_q$ es necesario introducir las clases ciclotómicas mód n.

Sea $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$ el anillo de enteros módulo n. Para cualquier s en $\mathbb{Z}_n$, la *clase ciclotómica de s módulo n* se define como

$$C_s = \left\{ s, sq, sq^2, \ldots, sq^{l_s - 1} \right\} \pmod{n} \subseteq \mathbb{Z}_n,$$

donde s es llamado el *representante de la clase* y l_s es el entero positivo más pequeño tal que $s \equiv sq^{l_s} \pmod{n}$, con $|C_s| = l_s$. Sea $\Omega_{(n,q)}$ el conjunto de todos los representantes diferentes de las clases ciclotómicas. Se tiene entonces que $C_s \cap C_t = \emptyset$ para cualesquiera dos elementos distintos s y t en $\Omega_{(n,q)}$, y

$$\bigcup_{s \in \Omega_{(n,q)}} C_s = \mathbb{Z}_n. \tag{2.1}$$

Por tanto, las distintas clases ciclotómicas módulo n generan una partición de $\mathbb{Z}_n$.

Ahora, sea γ como en la notación introducida a principio de este Capítulo. Sea $\beta = \gamma^{(q^k-1)/n}$. Note entonces que β es una n-ésima raíz primitiva de la unidad en $\mathbb{F}_{q^k}$ pues no solo cumple con satisfacer la ecuación $x^n - 1 = 0$ ($\beta^n = \gamma^{q^k-1} = 1$) sino también, por el Teorema 1, se tiene que β es un generador del (único) subgrupo cíclico de $\mathbb{F}_{q^k}^*$ de orden n, $< \beta >$. El *polinomio mínimo* $M_s(x)$ de β^s sobre $\mathbb{F}_q$ es el polinomio mónico de grado menor sobre $\mathbb{F}_q$ tal que $M_s(\beta^s) = 0$. Este polinomio está dado por

$$M_s(x) = \prod_{i \in C_s} \left(x - \beta^i\right) \in \mathbb{F}_q[x], \tag{2.2}$$

que es irreducible sobre $\mathbb{F}_q$ (ver [3, Capítulo 4, Propiedad (M1), p. 99]). Se sigue entonces de (2.1) que

$$x^n - 1 = \prod_{s \in \Omega_{(n,q)}} M_s(x),$$

que es la factorización de $x^n - 1$ en factores irreducibles sobre $\mathbb{F}_q$.

2.4. Otras definiciones

A continuación se presenta una función importante de $\mathbb{F}_{q^k}$ a $\mathbb{F}_q$ que resulta ser lineal y regular (ver [2, Capítulo 2, Teorema 2.23, p. 51]).

Definición 3. *[2, Capítulo 2, Definición 2.22, p. 50] Para $\alpha \in \mathbb{F}_{q^k}$, la traza $\mathrm{Tr}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha)$ de α sobre $\mathbb{F}_q$ se define como*

$$\mathrm{Tr}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha) := \alpha + \alpha^q + \cdots + \alpha^{q^{k-1}}.$$

Si $\mathbb{F}_q$ no tiene subcampos propios, es decir, es un campo primo, entonces $\mathrm{Tr}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha)$ será llamada la traza absoluta de α y será denotada simplemente como $\mathrm{Tr}(\alpha)$.

Observe que calcular la traza de un elemento $\alpha \in \mathbb{F}_{q^k}$ no es una tarea sencilla. Dicho cálculo involucra la construcción del campo finito $\mathbb{F}_{q^k}$, lo cual a su vez implica llevar a cabo múltiples divisiones sintéticas (pues los elementos del campo finito son las clases residuales de $\mathbb{F}_q[x]$ módulo un polinomio primitivo[3] de grado k definido sobre $\mathbb{F}_q$) y también operaciones con la aritmética particular del campo finito $\mathbb{F}_q$.

Ahora, sea γ como antes. Para cualquier entero a, se denotará como $h_a(x) \in \mathbb{F}_q[x]$ al polinomio mínimo de γ^{-a}.

Se concluye este Capítulo dando una definición de código cíclico irreducible en términos de la función traza, que además resultará de utilidad para los fines de este trabajo.

[3]Un polinomio primitivo es el polinomio mínimo de un elemento primitivo del campo finito $\mathbb{F}_{q^k}$.

Definición 4. *[4, Definición 2.2] Sean q y k como antes. Sea n un divisor positivo de $q^k - 1$, $u = (q^k - 1)/n$, y sea ω una n-ésima raíz primitiva de la unidad en $\mathbb{F}_{q^k}$. Entonces*

$$\mathcal{C}(q,k,u) := \left\{ c(y) := \left(\mathrm{Tr}_{\mathbb{F}_{q^k}/\mathbb{F}_q}\left(y\omega^i\right)\right)_{i=0}^{n-1} \;\middle|\; y \in \mathbb{F}_q^k \right\},$$

es llamado un código cíclico irreducible sobre $\mathbb{F}_q$.

De aquí en adelante, para cada $y \in \mathbb{F}_{q^k}$, $\mathrm{w}(y)$ denotará al peso de Hamming de la palabra de código $c(y) \in \mathcal{C}(q,k,u)$.

Observación 3. Por el teorema de Delsarte (ver, por ejemplo, [1]), el polinomio de chequeo de paridad para $\mathcal{C}(q,k,u)$ es $h_{-u}(x)$. Además, $\mathcal{C}(q,k,u)$ es un $[n,k']$ código, con $k' = \mathrm{grado}(h_{-u}(x)) = \mathrm{ord}_n(q)$, que es un divisor de k.

Observación 4. Sean a y b enteros distintos tal que $\mathrm{mcd}(q^k-1,a) = \mathrm{mcd}(q^k-1,b) = \frac{q^k-1}{n}$, entonces note que γ^a y γ^b son dos raíces n-ésimas primitivas de la unidad en $\mathbb{F}_{q^k}$. Así, por la Definición 4, la distribución de pesos para $\mathcal{C}(q,k,a)$ y $\mathcal{C}(q,k,b)$ será la misma.

Capítulo 3

Sumas exponenciales

Las sumas exponenciales son herramientas importantes en teoría de números para resolver problemas que involucran números enteros, y números reales en general, que a menudo son intratables por otros métodos. Sumas análogas pueden ser consideradas en el contexto de campos finitos resultando ser de gran utilidad en varias aplicaciones sobre tales campos, como es el caso del presente trabajo donde son usadas para calcular los pesos de los códigos cíclicos irreducibles.

Un rol básico en las sumas exponenciales para campos finitos es jugado por un grupo especial de homomorfismos conocidos como *caracteres*. Se distinguen dos tipos de caracteres, a saber aditivos y multiplicativos, dependiendo de cuando se haga referencia al grupo aditivo o al grupo multiplicativo del campo finito. Los caracteres entonces son funciones del grupo aditivo o multiplicativo de dicho campo hacia las raíces complejas de la unidad. Las sumas exponenciales son formadas utilizando los valores de uno o más caracteres y posiblemente combinando éstos con funciones de pesos u otras funciones.

Uno de los tipos de sumas exponenciales para campos finitos más importantes son las sumas Gaussianas ya que mezclan caracteres de ambas estructuras del campo. También aparecen en otros contextos en álgebra y teoría de números (ver, por ejemplo, [2, Capítulo 5, p. 162]). Para este trabajo son de interés pues una identidad de Robert J. McEliece [4, Lema 2.8] muestra que los pesos de los códigos cíclicos irreducibles pueden ser expresados como combinaciones lineales de sumas Gaussianas. Y es precisamente este resultado sobre el cual se basa el algoritmo propuesto en este trabajo.

El presente Capítulo tiene como objetivo introducir de manera formal los conceptos antes mencionados. Como se verá más adelante, la evaluación de la suma exponencial en la identidad de McEliece [4, Lema 2.8] no resulta trivial, de hecho, la suma opera sobre el campo de los números complejos $\mathbb{C}$. En este sentido, otro objetivo de este Capítulo es analizar algunas propiedades de las sumas Gaussianas con la intención de simplificar el cálculo de la identidad antes mencionada.

3.1. Caracteres

Sean p, q, k y ξ_s como a principios del Capítulo 2. Sea $L = \mathbb{F}_{q^k}$. Considere primero al grupo aditivo de L. La función $\psi_1 : L \longrightarrow \mathbb{C}$ definida como

$$\psi_1(c) := \xi_p^{\mathrm{Tr}(c)}, \quad \forall c \in L,$$

es llamada *caracter aditivo canónico* de L. Donde, como ya se dijo, $\mathrm{Tr} : L \longrightarrow \mathbb{F}_p$ denota la función traza absoluta.

Considere ahora al grupo multiplicativo de L, es decir, $L^* := L\backslash\{0\}$. Sea γ como en el Capítulo anterior. Para cada $j \in \{0, 1, \ldots, q^k - 2\}$, la función $\chi_j : L^* \longrightarrow \mathbb{C}$ con

$$\chi_j(\gamma^l) := \xi_{q^k-1}^{jl}, \quad l \in \{0, 1, \ldots, q^k - 2\},$$

define un *caracter multiplicativo* de L^*. El caracter multiplicativo χ_0 tal que

$$\chi_0(c) = 1, \quad \forall c \in L^*,$$

es llamado caracter multiplicativo *trivial*. También, a cada caracter χ se le asocia su caracter *conjugado* $\bar{\chi}$ definido como

$$\bar{\chi}(c) := \overline{\chi(c)} = \chi(c^{-1}), \quad \forall c \in L^*.$$

Sea L° el conjunto de caracteres multiplicativos en L^*. Un caracter multiplicativo χ de L^* puede ser no trivial en L^* pero aún así aniquilar a todo un subgrupo H de L^* en el sentido de que $\chi(h) = 1$, $\forall h \in H$. El subconjunto de caracteres multiplicativos en L^* que aniquilan a un subgrupo dado $H \subseteq L^*$, es decir,

$$\{\chi \in L^\circ \mid \chi(h) = 1, \ \forall h \in H\},$$

es llamado el *conjunto aniquilador de H en L°*.

3.2. Sumas Gaussianas

Antes de presentar la identidad de McEliece [4, Lema 2.8], se recuerda la definición de suma Gaussiana.

Definición 5. *[4, Definición 2.6] Con la misma notación, sea χ caracter multiplicativo de L^*. Entonces, se define su suma Gaussiana como*

$$G_L(\chi) := \sum_{x \in L^*} \chi(x)\xi_p^{\mathrm{Tr}(x)} \in \mathbb{C}.$$

A continuación se enuncia la identidad de McEliece.

Lema 2. *[4, Lema 2.8] Sea u divisor de $\Delta = (q^k - 1)/(q - 1)$ y $n := (q^k - 1)/u$. Sea $U =< \gamma^u >$, es decir, U es el subgrupo de L^* de índice u. Sea*

$$\Gamma := \{\chi \in L^\circ \mid \chi(a) = 1, \ \forall a \in U\},$$

es decir, Γ *es el conjunto aniquilador de* U *en* L°*. Para* $0 \leq j < q^k - 1$*, el peso de Hamming de la palabra de código* $c(\gamma^j) \in \mathcal{C}(q,k,u)$ *está dado por*

$$\mathrm{w}(\gamma^j) = \frac{q-1}{qu}\left(q^k - \sum_{\chi \in \Gamma \setminus \{\chi_0\}} G_L(\chi)\bar{\chi}(\gamma^j)\right). \tag{3.1}$$

Observación 5. Sean $q = p^t, k, u$ y n como en el Lema 2. Por un resultado de Schmidt y White [4, Observación 2.10], se sabe que el código $\mathcal{C}(q,k,u)$ tendrá el mismo número de pesos distintos que el código $\mathcal{C}(p,kt,u)$. Más aún, los pesos de estos códigos difieren sólo por el factor constante $(q-1)p/(p-1)q$. Entonces, para estudiar a los códigos $\mathcal{C}(q,k,u)$, basta considerar el caso en que $q = p$ y $q-1$ divide a $n = (q^k-1)/u$. Por tal motivo, de aquí en adelante se asume que $q = p$ y $p-1$ divide a $n = (p^k-1)/u$. Es decir, ahora se tiene que $L = \mathbb{F}_{p^k}$, $< \gamma >= L^*$ y $\Delta = (p^k-1)/(p-1)$.

En el Lema 2, como Γ es el conjunto aniquilador de U en L° se cumple que

$$|\Gamma| = \frac{|L^*|}{|U|} = \frac{p^k-1}{\frac{p^k-1}{u}} = u,$$

(ver [2, Capítulo 5, Teorema 5.6, p. 165]).

Ahora bien, note que la evaluación de (3.1) no resulta trivial. De hecho, como se dijo, la suma opera con raíces complejas de la unidad. A manera de simplificar el cálculo de esta última expresión, se analizará la siguiente suma exponencial:

$$\tau(\gamma^j) := \sum_{\chi \in \Gamma \setminus \{\chi_0\}} G_L(\chi)\bar{\chi}(\gamma^j) \in \mathbb{C}, \tag{3.2}$$

con $j \in \{0, \ldots, p^k - 2\}$.

Así pues, sea $\chi \in \Gamma \setminus \{\chi_0\}$ (es decir, χ es un caracter multiplicativo en Γ que no es el trivial). Considere las siguientes propiedades:

Lema 3. *Con la misma notación, si* u *divide a* Δ *entonces para cualquier entero* $0 \leq i < \Delta$ *se cumplirá:*

(P1) $\chi(\gamma^{\Delta+i}) = \chi(\gamma^i)$.

(P2) $\mathrm{Tr}(\gamma^{\Delta+i}) = 0 \iff \mathrm{Tr}(\gamma^i) = 0$.

(P3) $\left|\left\{0 \leq l < p^k - 1 \mid l \pmod{\Delta} = i\right\}\right| = p-1$.

(P4) $\mathrm{Tr}(\gamma^i) \neq 0 \Longrightarrow \left\{\mathrm{Tr}\left(\gamma^{i+\Delta\cdot l}\right) \;\middle|\; 0 \leq l < p-1\right\} = \{1, 2, \ldots, p-1\} = \mathbb{F}_p^*$.

Demostración.
(P1) En efecto,

$$\chi(\gamma^{\Delta+i}) = \chi(\gamma^\Delta)\chi(\gamma^i) = 1 \cdot \chi(\gamma^i) = \chi(\gamma^i),$$

pues u divide a Δ y χ es trivial en U.

(P2) Note que,

$$\mathrm{Tr}(\gamma^{\Delta+i}) = \mathrm{Tr}(\gamma^{\Delta} \cdot \gamma^{i}) = \gamma^{\Delta} \cdot \mathrm{Tr}(\gamma^{i}),$$

pues, por el Teorema 1 (2), se sabe que $< \gamma^{\Delta} >= \mathbb{F}_p^* \subsetneq L^* =< \gamma >$ y además la traza es una función lineal. Adicionalmente, $\gamma^{\Delta} \neq 0$, pues es un generador de $\mathbb{F}_p$, y se sigue el resultado deseado.

(P3) Observe que l debe ser de la forma $l = b\Delta + i$, para algún entero b, así l (mód Δ) $= i$. Ahora, Δ e i están fijas, sólo es posible variar b y, por las restricciones sobre l, esto se puede hacer $0 \leq b < p-1$.

(P4) Se tiene que,

$$\mathrm{Tr}(\gamma^{i+\Delta\cdot l}) = \mathrm{Tr}(\gamma^{i} \cdot \gamma^{\Delta\cdot l}) = \gamma^{\Delta\cdot l} \cdot \mathrm{Tr}(\gamma^{i}),$$

pues $\gamma^{\Delta\cdot l} \in < \gamma^{\Delta} >= \mathbb{F}_p^*$ y la traza es una función lineal. Entonces

$$\mathrm{Tr}(\gamma^{i+\Delta\cdot l}) = \gamma^{\Delta\cdot l} \cdot \alpha, \quad \text{con } \alpha \in \mathbb{F}_p^*, \text{ pues } \mathrm{Tr}(\gamma^{i}) \neq 0.$$

Note que en la última igualdad sólo es posible variar l ($0 \leq l < p-1$) y, para cada una, $\gamma^{\Delta\cdot l} \cdot \alpha$ representa a un elemento único. Si $\gamma^{\Delta\cdot l_1} \cdot \alpha = \gamma^{\Delta\cdot l_2} \cdot \alpha$ con $l_1 \neq l_2$, entonces $\gamma^{\Delta\cdot l_1} = \gamma^{\Delta\cdot l_2}$ (aplicando ley de la cancelación), y así $l_1 = l_2$ (pues Δ está fija) lo cual es una contradicción. □

Ahora, se definen los siguientes conjuntos de índices:

$$TZ := \left\{0 \leq i < \Delta \;\middle|\; \mathrm{Tr}(\gamma^{i}) = 0\right\},$$

$$TnZ := \{0, 1, 2, \ldots, \Delta - 1\} \setminus TZ.$$

Y se define $\delta := |TZ|$. Observe que $\delta = \frac{p^{k-1}-1}{p-1}$, ya que

$$|\{b \in L \mid \mathrm{Tr}(b) = 0\}| = p^{k-1},$$

debido a que la traza es una función regular (ver, por ejemplo, [2, Capítulo 2, Teorema 2.23, p. 51]).

Dicho lo anterior, note que, para $\chi \in \Gamma \setminus \{\chi_0\}$:

$$\begin{aligned} G_L(\chi) &= \sum_{0 \leq i < p^k - 1} \chi(\gamma^{i}) \xi_p^{\mathrm{Tr}(\gamma^{i})} \\ &= \sum_{\substack{0 \leq i < \Delta \\ \mathrm{Tr}(\gamma^{i}) = 0}} \chi(\gamma^{i}) \underbrace{(\xi_p^0 + \cdots + \xi_p^0)}_{p-1 \text{ veces}} + \sum_{\substack{0 \leq i < \Delta \\ \mathrm{Tr}(\gamma^{i}) \neq 0}} \chi(\gamma^{i})(\xi_p^1 + \xi_p^2 + \cdots + \xi_p^{p-1}), \end{aligned}$$

por la propiedad (P1) del Lema 3 es posible acotar la suma a aquellas $i \in \{0, \ldots, \Delta - 1\}$. También es posible separar la suma en aquellos índices que pertenecen a TZ y aquellos que no. En la última igualdad, en la suma izquierda, se utilizan las propiedades (P2) y (P3), mientras que (P4) es utilizada para la suma de la derecha. Así,

$$G_L(\chi) = (p-1)\sum_{i\in TZ}\chi(\gamma^i) + \sum_{i\in TnZ}\chi(\gamma^i)(-1),$$

pues la suma de las p-ésimas raíces complejas de la unidad es cero, es decir, $\xi_p^0 + \xi_p^1 + \cdots + \xi_p^{p-1} = 0$, de donde se sigue que $\xi_p^1 + \xi_p^2 + \cdots + \xi_p^{p-1} = -1$, pues $\xi_p^0 = 1$. Entonces,

$$\begin{aligned} G_L(\chi) &= p\sum_{i\in TZ}\chi(\gamma^i) - \sum_{i\in TZ}\chi(\gamma^i) - \sum_{i\in TnZ}\chi(\gamma^i) \\ &= p\sum_{i\in TZ}\chi(\gamma^i) - \sum_{0\le i<\Delta}\chi(\gamma^i) \\ &= p\sum_{i\in TZ}\chi(\gamma^i) - 0, \end{aligned}$$

pues χ es un caracter multiplicativo no trivial de un grupo abeliano finito, entonces se cumple que $\sum_{c\in L^*}\chi(c) = 0$ (ver [2, Capítulo 5, Teorema 5.4, p. 164]).

Es decir, se tiene que:

$$G_L(\chi) = p\sum_{i\in TZ}\chi(\gamma^i), \tag{3.3}$$

para toda $\chi \in \Gamma\backslash\{\chi_0\}$.

Como se puede notar, se ha llegado a una expresión más sencilla para la suma Gaussiana. Ahora sólo es necesario calcular el conjunto TZ y trabajar con sus índices. Más aún, la parte correspondiente al caracter aditivo en la suma, $\xi_p^{\mathrm{Tr}(\gamma^i)}$, ha desaparecido.

Lo que resta para terminar este Capítulo es utilizar lo anterior con el fin de simplificar el cálculo de la suma (3.2). Sin embargo, antes de ello, es necesario introducir unos subconjuntos del conjunto TZ que crean una partición del mismo. Además, como se verá en el siguiente Capítulo, son estos subconjuntos los que en gran medida ayudarán a calcular los pesos de los códigos cíclicos irreducibles.

Lema 4. *Con la notación anterior se define*

$$TZ_u^j := \{i \in TZ \mid u \mid i-j\},$$

con $0 \le j < u$. *Se tiene entonces que:*

(a) *Para* $0 \le j \ne j' < u$, $TZ_u^j \cap TZ_u^{j'} = \emptyset$.

(b) $\bigcup_{j=0}^{u-1} TZ_u^j = TZ$

Demostración.

(a) Suponga que $TZ_u^j \cap TZ_u^{j'} \neq \emptyset$. Sea $i \in TZ_u^j \cap TZ_u^{j'}$, entonces se tiene que u divide a $i-j$ y a $i-j'$. De donde se sigue que $u \mid (i-j)-(i-j') = j'-j$, lo cual sólo pasa si $j'-j=0$, es decir, $j'=j$, una contradicción.

(b) Debe ser claro que $\bigcup_{j=0}^{u-1} TZ_u^j \subseteq TZ$. Resta por demostrar que $TZ \subseteq \bigcup_{j=0}^{u-1} TZ_u^j$. Sea $i \in TZ$. Si u divide a i entonces se sigue el resultado deseado. Suponga entonces que u no divide a i. Aplicando el algoritmo de la división a u y a i, se tiene que $i = uq + j$ con $q, j \in \mathbb{Z}$ y $0 \leq j < u$. Por tanto, $i - j = uq$, es decir, $u \mid i-j$, con $0 \leq j < u$. Y así, $i \in TZ_u^j$.

□

Volviendo a la suma exponencial (3.2), sea $j \in \{0, \ldots, u-1\}$. Utilizando (3.3) se tiene que:

$$\begin{aligned}
\tau(\gamma^j) &= p \sum_{\chi \in \Gamma \setminus \{\chi_0\}} \sum_{i \in TZ} \chi(\gamma^i)\bar{\chi}(\gamma^j) \\
&= p \sum_{\chi \in \Gamma \setminus \{\chi_0\}} \sum_{i \in TZ} \chi(\gamma^{i-j}) \\
&= p \sum_{i \in TZ} \sum_{\chi \in \Gamma \setminus \{\chi_0\}} \chi(\gamma^{i-j}) \\
&= p \left(\sum_{i \in TZ_u^j} \sum_{\chi \in \Gamma \setminus \{\chi_0\}} \chi(\gamma^{i-j}) + \sum_{i \in TZ \setminus TZ_u^j} \sum_{\chi \in \Gamma \setminus \{\chi_0\}} \chi(\gamma^{i-j}) \right),
\end{aligned}$$

donde en la última igualdad se separa la suma en aquellos $i \in TZ$ tales que u divide a $i-j$ y cuando no lo hace. Note que, cuando u divide a $i-j$ pasa que: $\chi(\gamma^{i-j}) = 1$. Pero si u no divide a $i-j$ se tiene que: como $i-j \neq 0$ (si $i-j=0$, entonces u sí dividiría a $i-j$), $\sum_{\chi \in \Gamma} \chi(\gamma^{i-j}) = 0$ (ver [2, Capítulo 5, Teorema 5.4, p. 164]). De donde se sigue que $\sum_{\chi \in \Gamma \setminus \{\chi_0\}} \chi(\gamma^{i-j}) = -1$ pues $\chi_0(\gamma^{i-j}) = 1$. Se tiene entonces que

$$\begin{aligned}
\tau(\gamma^j) &= p \left(\sum_{i \in TZ_u^j} |\Gamma \setminus \{\chi_0\}| + \sum_{i \in TZ \setminus TZ_u^j} (-1) \right) \\
&= p \left(\left|TZ_u^j\right|(u-1) - \left|TZ \setminus TZ_u^j\right| \right),
\end{aligned}$$

pues, como se mencionó, $|\Gamma| = u$. Así

$$\begin{aligned}
\tau(\gamma^j) &= p \left(\left|TZ_u^j\right|(u-1) - \left(|TZ| - \left|TZ_u^j\right|\right) \right) \\
&= p \left(\left|TZ_u^j\right|u - \left|TZ_u^j\right| + \left|TZ_u^j\right| - |TZ| \right) \\
&= p \left(\left|TZ_u^j\right|u - \delta \right),
\end{aligned}$$

donde debe recordarse que $\delta = |TZ| = \frac{p^{k-1}-1}{p-1}$. De esta manera, se tiene que

$$\tau(\gamma^j) = \sum_{\chi \in \Gamma \setminus \{\chi_0\}} G_L(\chi)\bar{\chi}(\gamma^j) = p\left(\left|TZ_u^j\right|u - \delta\right), \tag{3.4}$$

con $0 \leq j < u$.

En esta última expresión para la suma exponencial $\tau(\gamma^j)$ en (3.2) ya no es necesario operar con raíces complejas de la unidad, ahora su evaluación sólo depende del cálculo del conjunto TZ y de sus subconjuntos TZ_u^j, con $j \in \{0, \ldots, u-1\}$, pues p, u y δ son conocidas.

A pesar de que el conjunto TZ no requiere del uso de herramienta compleja para su cálculo, éste sigue siendo difícil de generar al involucrar el cálculo de la traza absoluta de algunos elementos de L^*.

En el siguiente Capítulo se usará lo antes demostrado para simplificar el cálculo de (3.1) y, con ello, presentar el algoritmo que permitirá calcular todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles definidos sobre campos finitos pequeños.

Capítulo 4

Algoritmo para el cálculo de las distribuciones de pesos de códigos cíclicos irreducibles sobre campos finitos

Este capítulo se centra en describir el algoritmo para obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles definidos sobre un campo finito. Sin embargo su uso es propio para campos finitos pequeños y para códigos cuya dimensión no sea tan grande.

Sean $p, k, \Delta, \delta, \gamma$ y u como antes. Sea $j \in \{0, \ldots, u-1\}$. Sustituyendo (3.4) en (3.1), se tiene que:

$$\begin{aligned}
\mathrm{w}(\gamma^j) &= \frac{p-1}{pu}\left(p^k - p\left(\left|TZ_u^j\right|u - \delta\right)\right) \\
&= \frac{p-1}{u}\left(p^{k-1} + \frac{p^{k-1}-1}{p-1} - \left|TZ_u^j\right|u\right) \\
&= \frac{p-1}{u}\left(\Delta - \left|TZ_u^j\right|u\right) \\
&= (p-1)\left(\frac{\Delta}{u} - \left|TZ_u^j\right|\right).
\end{aligned}$$

Es decir:

$$\mathrm{w}(\gamma^j) = (p-1)\left(\frac{\Delta}{u} - \left|TZ_u^j\right|\right).$$

Más aún, sea:

$$F_j := \left\{0 \leq i < u \mid \left|TZ_u^i\right| = \left|TZ_u^j\right|\right\},$$

y debido a que en (3.1) se tomó $\chi \in \Gamma \backslash \{\chi_0\}$, se cumple que

$$\mathrm{w}(\gamma^{j+u}) = \mathrm{w}(\gamma^j),$$

pues

$$\bar{\chi}(\gamma^{j+u}) = \bar{\chi}(\gamma^j) \cdot \bar{\chi}(\gamma^u) = \bar{\chi}(\gamma^j) \cdot 1,$$

ya que χ es un caracter multiplicativo en $\mathbb{F}_{p^k}$ que es trivial en $< \gamma^u >$.

Por tanto, se concluye que la frecuencia con la que aparecerá el peso $\mathrm{w}(\gamma^j)$ en el código cíclico $\mathcal{C}(p,k,u)$ será $|F_j|\frac{p^k-1}{u}$, es decir:

Peso	Frecuencia
$(p-1)\left(\frac{\Delta}{u} - \lvert TZ_u^j\rvert\right)$	$\lvert F_j\rvert\frac{p^k-1}{u}$

Tabla 4.1: Distribución de pesos para $\mathcal{C}(p,k,u)$

De esta manera, por lo visto en el Capítulo 3, se ha obtenido una expresión muy sencilla para calcular el peso de Hamming de las palabras de código de $\mathcal{C}(p,k,u)$, y la frecuencia con la que aparecerán en el código, que sólo depende del cálculo del conjunto TZ y de sus subconjuntos TZ_u^j pues las demás incógnitas se obtienen de manera inmediata al ser p y k conocidas.

Precisamente la construcción del conjunto TZ es la rutina que más consume recursos en el algoritmo pues, como se mencionó, involucra el cálculo de las trazas absolutas de los elementos de $\mathbb{F}_{p^k}^* =< \gamma >$. Afortunadamente, este cálculo debe realizarse una única vez y sólo para algunos elementos del campo finito, a saber, para: γ^i con $i \in \{0,\ldots,\Delta-1\}$. Después ya sólo se debe trabajar con los índices i para los cuales se encontró que $\mathrm{Tr}(\gamma^i) = 0$, pues son estos índices los que se reparten entre los subconjuntos TZ_u^j para crear la partición de TZ. Y que a su vez, como puede observarse en la Tabla 4.1, son estos subconjuntos los que en buena medida se encargan de obtener la distribución de pesos del código.

Entonces, el algoritmo propuesto se limita a manipular números enteros y ya no a raíces complejas de la unidad como en un principio.

Note que el cálculo del conjunto TZ implica la construcción del campo finito $\mathbb{F}_{p^k}$. Es por esta razón que el algoritmo toma como valores de entrada un número primo p pequeño y un polinomio primitivo $f(x) \in \mathbb{F}_p[x]$, con $\mathrm{grado}(f(x)) = k$ (k pequeña), pues son estos dos elementos los necesarios para construir al campo finito en cuestión: $\mathbb{F}_{p^k} = \mathbb{F}_p[x]/(f(x)) = \mathbb{F}_p(\gamma)$, con $f(\gamma) = 0$.

En la Obervación 3 se mencionó que $\mathcal{C}(p,k,u)$ es un código cíclico irreducible de longitud $n = (p^k-1)/u$ y dimensión $k' = \mathrm{ord}_n(p)$, con k' divisor de k. Para el algoritmo que se propone serán de interés aquellos códigos $\mathcal{C}(p,k,u)$ cuya dimensión sea exactamente igual a k, es decir, aquellos códigos que cumplan con que $k' = k$, pues únicamente para estos códigos el campo de descomposición del polinomio x^n-1 es $\mathbb{F}_{p^k}$.

En caso de que $k' < k$, el campo $\mathbb{F}_{p^{k'}}$ (subcampo de $\mathbb{F}_{p^k}$) será el campo de descomposición para el polinomio x^n-1. Y en este sentido, para obtener las distribuciones de pesos de los códigos cíclicos irreducibles de dimensión k' sobre $\mathbb{F}_p$ será suficiente trabajar con el subcampo $\mathbb{F}_{p^{k'}} \subsetneq \mathbb{F}_{p^k}$. Es decir, para el algoritmo que aquí se propone será suficiente con cambiar el polinomio primitivo de grado k por uno que ahora tenga grado k'.

En el caso particular que $k' = 1$, es decir, si $u = \Delta$ entonces el código $\mathcal{C}(p, k, \Delta)$ tendrá longitud $n = p - 1$ y dimensión 1, pues $\mathrm{ord}_{p-1}(p) = 1$. Y entonces, dicho código será equivalente a un código de repetición de longitud $p - 1$ y por tanto su distribución de pesos estará dada por: $A_0 = 1, A_{p-1} = p - 1$.

Algoritmo eficiente para el cálculo de todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión k (pequeña) definidos sobre un campo primo pequeño $\mathbb{F}_p$

```
1: procedure DistrPCodIrred(p,pol)
2:     /* donde p es un número primo y pol es el polinomio primitivo
3:        de grado k definido sobre F_p */
4:     k = grado(pol)
5:     Δ = (p^k − 1)/(p − 1)
6:     δ = (p^{k−1} − 1)/(p − 1)
7:
8:     /* Construcción del campo finito
9:        F_{p^k} = F_p[x]/(pol) = F_p(γ), con pol(γ) = 0 */
10:    define gama[p^k − 1]
11:    for (i = 0; i < p^k − 1; i++) do
12:        gama[i] = residuo(γ^i/(pol))        /* es decir, gama[i] almacena el residuo resultante de realizar la división sintética de γ^i entre el polinomio primitivo pol */.
13:
14:    /* Cálculo de la función traza absoluta
15:        Tr(γ^i) = γ^i + (γ^i)^p + ··· + (γ^i)^{p^{k−1}}
16:        el conjunto TZ, representado por el arreglo Traza, sólo se calcula una vez */
17:    define Traza[Δ]
18:    for (i = 0; i < Δ; i++) do
19:        define tem = 0
20:        for (j = 0, m = 1; j < k; j++, m = (m · p) (mód p^k − 1)) do
21:            tem+ = gama[(i · m) (mód p^k − 1)]
22:        Traza[i] = tem
23:
24:    /* Cálculo de la distribución de pesos */
25:    for (u = 1; u < Δ; u++) do
26:        if Δ ≡ 0 (mód u) then        /* u debe ser un divisor de Δ */
27:            n = (p^k − 1)/u
28:            if ord_n(p) = k then        /* C debe ser un código de dimensión k */
29:                print "La distribución de pesos de C(p, k, u) es: "
30:                /* Cálculo de los subconjuntos TZ_u^j */
31:                define F[δ + 1]
32:                define TZ[u]
33:                for (j = 0; j < u; j++) do
34:                    for (i = j, z = 0; i < Δ; i+ = u) do
35:                        if Traza[i] = 0 then
36:                            z++
37:                    TZ[j] = z
38:                    F[z]++
39:                for (i = δ; i ≥ 0; i−−) do
40:                    if F[i] ≠ 0 then
41:                        print "A_{(p−1)(Δ/u−i)} = F[i](p^k−1/u) "
42:            else
43:                print "No aplica, es de dimensión menor."
```

Se finaliza este Capítulo con un par de ejemplos de la ejecución del algoritmo. Mientras se ejemplifica lo antes visto, se hace referencia a las funciones que realizan dichas acciones en la implementación del algoritmo que se encuentra en el Apéndice A de este trabajo.

Ejemplo 3. Suponga que se desea obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 4 definidos sobre el campo primo con 3 elementos, $\mathbb{F}_3$.

Primero es necesario construir la extensión $\mathbb{F}_{3^4}$, para ello se debe encontrar un polinomio primitivo de grado 4 definido sobre $\mathbb{F}_3$, por ejemplo $x^4 + x + 2$. Entonces $\mathbb{F}_{3^4} = \mathbb{F}_3[x]/(x^4 + x + 2) = \mathbb{F}_3(\gamma)$, con $\gamma^4 + \gamma + 2 = 0$, es el campo buscado.

Se tiene entonces que $p = 3$, $k = \text{grado}(x^4 + x + 2) = 4$, $\Delta = \frac{3^4-1}{3-1} = 40$ y $\delta = \frac{3^3-1}{3-1} = 13$.

Siguiendo con el algoritmo, se debe construir el conjunto TZ. Para ello es necesario calcular $\text{Tr}(\gamma^i)$, la traza absoluta, para los elementos γ^i con $i \in \{0, \ldots, \Delta - 1\}$ y seleccionar aquellos índices i para los cuales se cumpla que $\text{Tr}(\gamma^i) = 0$. En la implementación del Apéndice A, la función `Tr` lleva a cabo dicho cálculo.

$$
\begin{aligned}
\text{Tr}(\gamma^0) &= \gamma^0 + \gamma^0 + \gamma^0 + \gamma^0 = 1 + 1 + 1 + 1 = 1 \\
\text{Tr}(\gamma) &= \gamma + \gamma^3 + \gamma^{3^2} + \gamma^{3^3} = \gamma + \gamma^3 + \gamma^9 + \gamma^{27} = 0 \\
\text{Tr}(\gamma^2) &= \gamma^2 + \gamma^{2\cdot 3} + \gamma^{2\cdot 3^2} + \gamma^{2\cdot 3^3} = \gamma^2 + \gamma^6 + \gamma^{18} + \gamma^{54} = 0 \\
&\vdots \\
\text{Tr}(\gamma^{37}) &= \gamma^{37} + \gamma^{37\cdot 3} + \gamma^{37\cdot 3^2} + \gamma^{37\cdot 3^3} = \gamma^{37} + \gamma^{31} + \gamma^{13} + \gamma^{39} = 2 \\
\text{Tr}(\gamma^{38}) &= \gamma^{38} + \gamma^{38\cdot 3} + \gamma^{38\cdot 3^2} + \gamma^{38\cdot 3^3} = \gamma^{38} + \gamma^{34} + \gamma^{22} + \gamma^{66} = 2 \\
\text{Tr}(\gamma^{39}) &= \gamma^{39} + \gamma^{39\cdot 3} + \gamma^{39\cdot 3^2} + \gamma^{39\cdot 3^3} = \gamma^{39} + \gamma^{37} + \gamma^{31} + \gamma^{13} = 2
\end{aligned}
$$

Nota: para obtener la expresión polinomial del elemento γ^i basta tomar el residuo resultante de realizar la división sintética de γ^i entre $\gamma^4 + \gamma + 2$ con la aritmética del campo primo $\mathbb{F}_3$. Por ejemplo, para γ^9 se tiene que su expresión polinomial está dada por $\gamma^3 + \gamma^2 + \gamma$, para γ^{27} es $\gamma^3 + 2\gamma^2 + \gamma$, y es por ello que

$$\text{Tr}(\gamma) = \gamma + \gamma^3 + \gamma^9 + \gamma^{27} = \gamma + \gamma^3 + (\gamma^3 + \gamma^2 + \gamma) + (\gamma^3 + 2\gamma^2 + \gamma) = 3(\gamma + \gamma^2 + \gamma^3) = 0.$$

Se tiene entonces que

$$TZ = \{1, 2, 3, 5, 6, 9, 14, 15, 18, 20, 25, 27, 35\}.$$

Note que en efecto $\delta = |TZ|$. Ahora, sea u divisor de $\Delta = 40$, es decir, $u \in \{1, 2, 4, 5, 8, 10, 20\}$. Para cada u se calcula la distribución de pesos para el código $\mathcal{C}(3, 4, u)$. Hay que recordar que la longitud del código $\mathcal{C}(3, 4, u)$ está dada por $n = (p^k - 1)/u$. La función `encuentraDistribucion` es la encargada de realizar el cálculo de las distribuciones de pesos en la implementación del Apéndice A.

(a) $u = 1$

Entonces, $n = \frac{3^4-1}{1} = 80$ y $\text{ord}_{80}(3) = 4 = k$.

Para $0 \leq j < 1$:

Si $j = 0$, se tiene que

$|TZ_1^0| = |\{i \in TZ \mid 1 \mid i\}| = |TZ| = 13$.

Y entonces,

$|F_0| = |\{0 \leq i < 1 \mid |TZ_1^i| = |TZ_1^0|\}| = |\{0\}| = 1$.

Por tanto, $\mathcal{C}(3, 4, 1)$ es un código cíclico irreducible de longitud $\boldsymbol{n = 80}$, dimensión $\boldsymbol{\text{ord}_{80}(3) = 4}$, cuya distribución de pesos es:

- $\diamond$ $\boldsymbol{A_0 = 1}$, pues la palabra de código cero siempre está presente en los códigos lineales;
- $\diamond$ $\boldsymbol{A_{54} = 80}$, pues

$$\text{w}(\gamma^0) = (3-1)\left(\frac{40}{1} - 13\right) = 54$$

$$Frecuencia = 1\left(\frac{80}{1}\right) = 80,$$

siguiendo lo establecido en la Tabla 4.1.

(b) $u = 2$

Entonces, $n = \frac{3^4-1}{2} = 40$ y $\text{ord}_{40}(3) = 4 = k$.

Para $0 \leq j < 2$:

Si $j = 0$: $|TZ_2^0| = |\{i \in TZ \mid 2 \mid i\}| = |\{2, 6, 14, 18, 20\}| = 5$.

Si $j = 1$: $|TZ_2^1| = |\{i \in TZ \mid 2 \mid i - 1\}| = |\{1, 3, 5, 9, 15, 25, 27, 35\}| = 8$.

Y entonces,

$|F_0| = |\{0 \leq i < 2 \mid |TZ_2^i| = |TZ_2^0|\}| = |\{0\}| = 1$,

$|F_1| = |\{0 \leq i < 2 \mid |TZ_2^i| = |TZ_2^1|\}| = |\{1\}| = 1$.

Por tanto, $\mathcal{C}(3, 4, 2)$ es un código cíclico irreducible de longitud $\boldsymbol{n = 40}$, dimensión $\boldsymbol{\text{ord}_{40}(3) = 4}$, cuya distribución de pesos es:

- $\diamond$ $\boldsymbol{A_0 = 1}$, por la misma razón del inciso anterior;
- $\diamond$ $\boldsymbol{A_{24} = 40}$, pues

$$\text{w}(\gamma^1) = (3-1)\left(\frac{40}{2} - 8\right) = 24$$

$$Frecuencia = 1\left(\frac{80}{2}\right) = 40;$$

- $\diamond$ $\boldsymbol{A_{30} = 40}$, pues

$$\text{w}(\gamma^0) = (3-1)\left(\frac{40}{2} - 5\right) = 30$$

$$Frecuencia = 1\left(\frac{80}{2}\right) = 40.$$

(c) $u = 4$

Entonces, $n = \frac{3^4-1}{4} = 20$ y $\mathrm{ord}_{20}(3) = 4 = k$.

Para $0 \leq j < 4$:

Si $j = 0$: $|TZ_4^0| = |\{i \in TZ \mid 4 \mid i\}| = |\{20\}| = 1$.

Si $j = 1$: $|TZ_4^1| = |\{i \in TZ \mid 4 \mid i-1\}| = |\{1, 5, 9, 25\}| = 4$.

Si $j = 2$: $|TZ_4^2| = |\{i \in TZ \mid 4 \mid i-2\}| = |\{2, 6, 14, 18\}| = 4$.

Si $j = 3$: $|TZ_4^3| = |\{i \in TZ \mid 4 \mid i-3\}| = |\{3, 15, 27, 35\}| = 4$.

Y entonces,

$|F_3| = |\{0 \leq i < 4 \mid |TZ_4^i| = |TZ_4^3|\}| = |\{1, 2, 3\}| = 3$,

$|F_0| = |\{0 \leq i < 4 \mid |TZ_4^i| = |TZ_4^0|\}| = |\{0\}| = 1$.

Por tanto, $\mathcal{C}(3, 4, 4)$ es un código cíclico irreducible de longitud $\boldsymbol{n = 20}$, dimensión $\boldsymbol{\mathrm{ord}_{20}(3) = 4}$, cuya distribución de pesos es:

- $\diamond$ $\boldsymbol{A_0 = 1}$;
- $\diamond$ $\boldsymbol{A_{12} = 60}$, pues

$$\mathrm{w}(\gamma^3) = (3-1)\left(\frac{40}{4} - 4\right) = 12$$

$$Frecuencia = 3\left(\frac{80}{4}\right) = 60;$$

- $\diamond$ $\boldsymbol{A_{18} = 20}$, pues

$$\mathrm{w}(\gamma^0) = (3-1)\left(\frac{40}{4} - 1\right) = 18$$

$$Frecuencia = 1\left(\frac{80}{4}\right) = 20.$$

(d) $u = 5$

Entonces, $n = \frac{3^4-1}{5} = 16$ y $\mathrm{ord}_{16}(3) = 4 = k$.

Para $0 \leq j < 5$:

Si $j = 0$: $|TZ_5^0| = |\{i \in TZ \mid 5 \mid i\}| = |\{5, 15, 20, 25, 35\}| = 5$.

Si $j = 1$: $|TZ_5^1| = |\{i \in TZ \mid 5 \mid i-1\}| = |\{1, 6\}| = 2$.

Si $j = 2$: $|TZ_5^2| = |\{i \in TZ \mid 5 \mid i-2\}| = |\{2, 27\}| = 2$.

Si $j = 3$: $|TZ_5^3| = |\{i \in TZ \mid 5 \mid i-3\}| = |\{3, 18\}| = 2$.

Si $j = 4$: $|TZ_5^4| = |\{i \in TZ \mid 5 \mid i-4\}| = |\{9, 14\}| = 2$.

Y entonces,

$|F_4| = |\{0 \leq i < 5 \mid |TZ_5^i| = |TZ_5^4|\}| = |\{1, 2, 3, 4\}| = 4$,

$|F_0| = |\{0 \leq i < 5 \mid |TZ_5^i| = |TZ_5^0|\}| = |\{0\}| = 1$.

Por tanto, $\mathcal{C}(3, 4, 5)$ es un código cíclico irreducible de longitud $\boldsymbol{n = 16}$, dimensión $\boldsymbol{\mathrm{ord}_{16}(3) = 4}$, cuya distribución de pesos es:

- $\diamond$ $\boldsymbol{A_0 = 1}$;

- $\diamond$ $\boldsymbol{A_{12} = 64}$, pues

$$\mathrm{w}(\gamma^4) = (3-1)\left(\frac{40}{5} - 2\right) = 12$$

$$Frecuencia = 4\left(\frac{80}{5}\right) = 64;$$

- $\diamond$ $\boldsymbol{A_6 = 16}$, pues

$$\mathrm{w}(\gamma^0) = (3-1)\left(\frac{40}{5} - 5\right) = 6$$

$$Frecuencia = 1\left(\frac{80}{5}\right) = 16.$$

(e) $u = 8$

Entonces, $n = \frac{3^4-1}{8} = 10$ y $\mathrm{ord}_{10}(3) = 4 = k$.

Para $0 \leq j < 8$:

Si $j = 0$: $|TZ_8^0| = |\{i \in TZ \mid 8 \mid i\}| = |\emptyset| = 0$.

Si $j = 1$: $|TZ_8^1| = |\{i \in TZ \mid 8 \mid i-1\}| = |\{1, 9, 25\}| = 3$.

Si $j = 2$: $|TZ_8^2| = |\{i \in TZ \mid 8 \mid i-2\}| = |\{2, 18\}| = 2$.

Si $j = 3$: $|TZ_8^3| = |\{i \in TZ \mid 8 \mid i-3\}| = |\{3, 27, 35\}| = 3$.

Si $j = 4$: $|TZ_8^4| = |\{i \in TZ \mid 8 \mid i-4\}| = |\{20\}| = 1$.

Si $j = 5$: $|TZ_8^5| = |\{i \in TZ \mid 8 \mid i-5\}| = |\{5\}| = 1$.

Si $j = 6$: $|TZ_8^6| = |\{i \in TZ \mid 8 \mid i-6\}| = |\{6, 14\}| = 2$.

Si $j = 7$: $|TZ_8^7| = |\{i \in TZ \mid 8 \mid i-7\}| = |\{15\}| = 1$.

Y entonces,

$|F_7| = |\{0 \leq i < 8 \mid |TZ_8^i| = |TZ_8^7|\}| = |\{4, 5, 7\}| = 3$,

$|F_6| = |\{0 \leq i < 8 \mid |TZ_8^i| = |TZ_8^6|\}| = |\{2, 6\}| = 2$,

$|F_3| = |\{0 \leq i < 8 \mid |TZ_8^i| = |TZ_8^3|\}| = |\{1, 3\}| = 2$,

$|F_0| = |\{0 \leq i < 8 \mid |TZ_8^i| = |TZ_8^0|\}| = |\{0\}| = 1$.

Por tanto, $\mathcal{C}(3, 4, 8)$ es un código cíclico irreducible de longitud $\boldsymbol{n = 10}$, dimensión $\boldsymbol{\mathrm{ord}_{10}(3) = 4}$, cuya distribución de pesos es:

- $\diamond$ $\boldsymbol{A_0 = 1}$;
- $\diamond$ $\boldsymbol{A_8 = 30}$, pues

$$\mathrm{w}(\gamma^7) = (3-1)\left(\frac{40}{8} - 1\right) = 8$$

$$Frecuencia = 3\left(\frac{80}{8}\right) = 30;$$

- $\diamond$ $\boldsymbol{A_6 = 20}$, pues

$$\mathrm{w}(\gamma^6) = (3-1)\left(\frac{40}{8} - 2\right) = 6$$

$$Frecuencia = 2\left(\frac{80}{8}\right) = 20;$$

◇ $\boldsymbol{A_4 = 20}$, pues

$$\mathrm{w}(\gamma^3) = (3-1)\left(\frac{40}{8} - 3\right) = 4$$

$$Frecuencia = 2\left(\frac{80}{8}\right) = 20;$$

◇ $\boldsymbol{A_{10} = 10}$, pues

$$\mathrm{w}(\gamma^0) = (3-1)\left(\frac{40}{8} - 0\right) = 10$$

$$Frecuencia = 1\left(\frac{80}{8}\right) = 10.$$

(f) Para $u = 10$ se tiene que $n = 8$ y entonces $\mathrm{ord}_8(3) = 2 \neq 4 = k$. Por tanto este caso no es aplicable pues la dimensión del código $\mathcal{C}(3, 4, 10)$ es 2 y, como se mencionó, el campo de descomposición para el polinomio $x^8 - 1$ es $\mathbb{F}_{3^2} \subsetneqq \mathbb{F}_{3^4}$, es decir, para este caso el conjunto TZ se debe recalcular utilizando un polinomio primitivo de grado 2.

Note que $< \gamma^{10} >= \mathbb{F}_{3^2}^*$. Entonces, para obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 2 sobre el campo primo $\mathbb{F}_3$ basta con ejecutar de nueva cuenta el algoritmo pero ahora con los valores de entrada 3 y el polinomio primitivo $h_{-10}(x)$ (es decir, el polinomio mínimo del elemento $\gamma^{10} \in \mathbb{F}_{3^4}^*$).

(g) Para $u = 20$ se tiene que $n = 4$ y $\mathrm{ord}_4(3) = 2 \neq 4 = k$. Tampoco es un caso aplicable. Al igual que en el inciso anterior, el campo de descomposición para el polinomio $x^4 - 1$ es $\mathbb{F}_{3^2}$.

En resumen, se tiene que todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 4 definidos sobre el campo primo $\mathbb{F}_3$ están dadas de la siguiente manera:

Código	Parámetros $[n, k, d]$	Distribución de pesos
$\mathcal{C}(3, 4, 1)$	$[80, 4, 54]$	$A_0 = 1, A_{54} = 80$
$\mathcal{C}(3, 4, 2)$	$[40, 4, 24]$	$A_0 = 1, A_{24} = 40, A_{30} = 40$
$\mathcal{C}(3, 4, 4)$	$[20, 4, 12]$	$A_0 = 1, A_{12} = 60, A_{18} = 20$
$\mathcal{C}(3, 4, 5)$	$[16, 4, 6]$	$A_0 = 1, A_{12} = 64, A_6 = 16$
$\mathcal{C}(3, 4, 8)$	$[10, 4, 4]$	$A_0 = 1, A_8 = 30, A_6 = 20, A_4 = 20, A_{10} = 10$

Tabla 4.2: Tabla con todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 4 definidos sobre el campo primo $\mathbb{F}_3$.

Ahora suponga que desea trabajar con el código del inciso (e): $\mathcal{C}(3, 4, 8)$. Por (2.2), el polinomio mínimo para el elemento $\gamma^8 \in \mathbb{F}_{3^4}$ está dado por

$$M(x) = \prod_{i \in \{1,3,9,7\}} \left(x - \left(\gamma^8\right)^i\right) = x^4 + 2x^3 + x^2 + 2x + 1 \in \mathbb{F}_3[x],$$

además, por la Observación 3, $M(x) = h_{-8}(x)$ corresponde al polinomio de chequeo de paridad del código $\mathcal{C}(3, 4, 8)$. En la implementación del Apéndice A la función encargada de realiza el cálculo del polinomio mínimo asociado al código es `encuentraPolMino`.

Entonces, si se desea implementar este último código basta con realizar la siguiente división sintética

$$g(x) = (x^{10} - 1)/h_{-8}(x) = x^6 + x^5 + 2x + 2 \in F_3[x],$$

para obtener el polinomio generador del código y, posteriormente, proceder como en el Teorema 2 ($\mathcal{C}(3, 4, 8) =< x^6 + x^5 + 2x + 2 >\subseteq F_3[x]/(x^{10} - 1)$).

Por otro lado, retomando lo dicho en el inciso (f), el polinomio mínimo para el elemento $\gamma^{10} \in \mathbb{F}_{3^4}^*$ está dado por

$$h_{-10}(x) = \prod_{i\in\{1,3\}} \left(x - \left(\gamma^{10}\right)^i\right) = x^2 + x + 2 \in \mathbb{F}_3[x],$$

Entonces, para obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 2 sobre el campo primo $\mathbb{F}_3$ basta con ejecutar de nueva cuenta el algoritmo pero ahora con los valores de entrada 3 y el polinomio primitivo $h_{-10}(x)$.

Para ilustrar cómo a través del algoritmo propuesto también es posible obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles sobre un campo finito con $q = p^t$ elementos (es decir, un campo no primo), se tiene el siguiente ejemplo:

Ejemplo 4. Se desea obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 2 definidos sobre el campo finito con 9 elementos, $\mathbb{F}_9$.

Para este caso se tiene que $q = p^t = 3^2$, $k = 2$, $\Delta = (q^k - 1)/(q - 1) = \frac{80}{8} = 10$ y $u \in \{1, 2, 5\}$, donde para cada u la longitud del código está dada por $n = (q^k - 1)/u$. Entonces, los códigos cíclicos irreducibles de los cuales se espera obtener su distribución de pesos son los siguientes:

Código	Parámetros $[n, k]$
$\mathcal{C}(9, 2, 1)$	$[80, 2]$
$\mathcal{C}(9, 2, 2)$	$[40, 2]$
$\mathcal{C}(9, 2, 5)$	$[16, 2]$

Por la Observación 5, se sabe que basta calcular la distribución de pesos para los siguientes códigos:

Código	Parámetros $[n, k]$
$\mathcal{C}(3, 4, 1)$	$[80, 4]$
$\mathcal{C}(3, 4, 2)$	$[40, 4]$
$\mathcal{C}(3, 4, 5)$	$[16, 4]$

Pues ambos tienen el mismo número de pesos. Además, los pesos de estos últimos códigos tan sólo difieren de los otros por el factor constante: $\frac{(q-1)p}{(p-1)q} = \frac{4}{3}$.

Entonces, utilizando los resultados del Ejemplo 3, se tiene que todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 2 definidos sobre el campo finito $\mathbb{F}_9$, están dadas de la siguiente manera:

Código	Parámetros $[n, k, d]$	Distribución de pesos
$\mathcal{C}(9, 2, 1)$	$[80, 2, 72]$	$A_0 = 1, A_{72} = 80$
$\mathcal{C}(9, 2, 2)$	$[40, 2, 32]$	$A_0 = 1, A_{32} = 40, A_{40} = 40$
$\mathcal{C}(9, 2, 5)$	$[16, 2, 8]$	$A_0 = 1, A_{16} = 64, A_8 = 16$

Tabla 4.3: Tabla con todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 2 definidos sobre el campo finito $\mathbb{F}_9$.

Conclusiones

En este trabajo se presentó un algoritmo que permite determinar de manera eficiente TODAS las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión k, con k pequeña, definidos sobre un campo finito pequeño con q elementos $\mathbb{F}_q$.

Por un resultado de Schmidt y White [4, Observación 2.10], para el algoritmo propuesto fue suficiente restringirse al caso en que q es un número primo pues el caso en que q es una potencia de un primo se puede extender fácilmente a partir del primero.

El algoritmo se basa en la identidad de Robert J. McEliece [4, Lema 2.8] con la cual es posible calcular los pesos de los códigos cíclicos irreducibles a través de una suma exponencial particular.

La evaluación de dicha suma compleja no resulta inmediata, pero gracias al estudio de sus propiedades en el Capítulo 3 se obtuvo una expresión sencilla para esta suma que sólo depende del cálculo del conjunto TZ y de sus subconjuntos TZ_u^j, pues las demás incógnitas se obtienen de manera inmediata al ser p y k conocidas. Se debe tomar en cuenta que a pesar de que la expresión que se obtuvo es sencilla, ésta involucra al cálculo de la función traza absoluta para algunos elementos del campo finito, lo cual no resulta fácil de obtener.

Al final fue posible reducir la identidad de McEliece a una expresión que sólo requiere manipular algunos números enteros y ya no a números complejos como en un principio.

También se presenta en el Apéndice A de este trabajo una implementación del algoritmo propuesto en lenguaje de programación C que, además de dar todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles, da el polinomio de chequeo de paridad asociado a cada código para que, si así se decide, éste pueda ser implementado.

El programa toma sólo dos parámetros para realizar los cálculos: un número primo p, con $p \in \{2, 3, 5, 7\}$, y un polinomio primitivo de grado k, con $k \in \{2, 3, 4, 5\}$, definido sobre $\mathbb{F}_p$. Dicha herramienta pretende ser un auxiliar en investigaciones que involucren el uso de códigos cíclicos irreducibles definidos sobre campos finitos pequeños.

Por ejemplo, existen caracterizaciones de códigos cíclicos irreducibles $\mathcal{C}(q = p^t, k, u)$ sobre campos finitos a partir de los valores de u. Si $u = 1$, el código cíclico irreducible $\mathcal{C}(q, k, 1)$ tendrá un sólo peso [5, Teorema 1]. Si $u \geq 2$, $f = \mathrm{ord}_u(p)$ es un número par y $p^{f/2} \equiv -1 \pmod{u}$, el código $\mathcal{C}(q, k, u)$ será un código cíclico irreducible se-

miprimitivo de dos pesos [5, Teorema 7]. Para estos valores de u, las distribuciones de pesos obtenidas por el programa pudieron ser confirmadas. Sería interesante ejecutar el programa, para distintos valores de p y k, y analizar su salida para otros valores de u. Esto con el fin de encontrar condiciones suficientes y necesarias para caracterizar de alguna manera la distribución de pesos de otras familias de códigos cíclicos irreducibles sobre campos finitos.

Como trabajo futuro se puede complementar la implementación realizada. Aunque el algoritmo es propio para campos finitos en general, el programa funciona de manera correcta y eficiente cuando los códigos son de dimensión pequeña ($k \in \{2, 3, 4, 5\}$) y además están definidos sobre campos primos pequeños ($p \in \{2, 3, 5, 7\}$). De no ser así, el programa puede llegar a tomar mucho tiempo para finalizar los cálculos y arrojar los resultados correctos. Utilizando mejores funciones o rutinas auxiliares en la implementación, se podría reducir su tiempo de ejecución. Y así, se podría trabajar con códigos cíclicos irreducibles definidos sobre campos primos con más elementos o códigos cuya dimensión sea más grande.

También, se podría complementar el programa para que tome en cuenta el resultado de Schmidt y White [4, Observación 2.10] y así extender su funcionamiento para cualquier campo finito pequeño y ya no sólo al caso primo.

Se debe recordar que el programa únicamente obtiene las distribuciones de pesos de aquellos códigos $\mathcal{C}(p, k, u)$ cuya dimensión, $k' = \mathrm{ord}_n(p)$, es igual a k. Se podría complementar el programa para que trabaje de forma recursiva, es decir, primero determinando las distribuciones de pesos de aquellos códigos cuya dimensión sea k (es decir, trabajar con el campo $\mathbb{F}_{p^k}$) y posteriormente determinando las distribuciones de pesos para los códigos cuya dimensión sea menor que k (es decir, trabajar con el subcampo $\mathbb{F}_{p^{k'}} \subsetneqq \mathbb{F}_{p^k}$), pues para obtener estas últimas distribuciones basta ejecutar de nueva cuenta el programa con los valores de entrada p y el polinomio mínimo asociado al elemento $\gamma^{\frac{p^k-1}{p^{k'}-1}} \in \mathbb{F}^*_{p^k} =< \gamma >$, el cual es calculado por el programa en la primera ejecución.

Apéndice A

Implementación del algoritmo

A continuación se presenta una implementación en lenguaje de programación C del algoritmo propuesto. El programa *DistrPCodIrred.c*, como se mencionó, toma dos argumentos: un número primo p pequeño ($p \in \{2, 3, 5, 7\}$) y un polinomio primitivo de grado k definido sobre $\mathbb{F}_p$, donde también se espera que k no sea muy grande ($k \in \{2, 3, 4, 5\}$).

Por ejemplo, si se desea obtener todas las posibles distribuciones de pesos de los códigos cíclicos irreducibles de dimensión 4 definidos sobre el campo primo $\mathbb{F}_7$, primero se debe encontrar un polinomio primitivo de grado 4 en $\mathbb{F}_7[x]$, por ejemplo $x^4 + x^3 + 6x + 3$, y luego ejecutar:

```
$ ./DistrPCodIrred -p 7 "(x^4+x^3+6x+3)"

Tabla de sumar GF(7)
 0  1  2  3  4  5  6
 1  2  3  4  5  6  0
 2  3  4  5  6  0  1
 3  4  5  6  0  1  2
 4  5  6  0  1  2  3
 5  6  0  1  2  3  4
 6  0  1  2  3  4  5

Tabla de multiplicar GF(7)
 0  0  0  0  0  0  0
 0  1  2  3  4  5  6
 0  2  4  6  1  3  5
 0  3  6  2  5  1  4
 0  4  1  5  2  6  3
 0  5  3  1  6  4  2
 0  6  5  4  3  2  1

Usaremos el pol. primitivo: (x^4+x^3+6x+3)
p=7  k=4  Delta:=(p^k-1)/(p-1)=400  n:=(p^k-1)/u

u    Código     Long(n) Dim  Pol. Mín.              Distribución de pesos
1    C(7,4,1)   2400    4    (x^4+x^3+6x+3)         A[2058]=2400
2    C(7,4,2)   1200    4    (x^4+6x^3+x^2+6x+2)    A[1008]=1200 A[1050]=1200
4    C(7,4,4)   600     4    (x^4+x^3+3x^2+3x+4)    A[504]=1800 A[546]=600
5    C(7,4,5)   480     4    (x^4+2x^3+3x^2+4x+5)   A[378]=480 A[420]=1920
8    C(7,4,8)   300     4    (x^4+5x^3+4x^2+x+2)    A[252]=2100 A[294]=300
```

```
10   C(7,4,10)  240     4    (x^4+2x^3+3x^2+4)      A[168]=240 A[210]=2160
16   C(7,4,16)  150     4    (x^4+4x^3+3x^2+x+4)    A[114]=300 A[120]=600
                                                    A[126]=300 A[132]=600
                                                    A[138]=300 A[144]=150
                                                    A[150]=150
20   C(7,4,20)  120     4    (x^4+2x^3+3x^2+3x+2)   A[84]=240 A[96]=600
                                                    A[102]=480 A[108]=480
                                                    A[114]=600
25   C(7,4,25)  96      4    (x^4+2x^2+3)           A[42]=96 A[84]=2304
40   C(7,4,40)  60      4    (x^4+2x^3+x^2+3x+4)    A[36]=120 A[42]=240
                                                    A[48]=720 A[54]=780
                                                    A[60]=540
50   C(7,4,50)  48      2    (x^2+2x+3)             No aplicable
80   C(7,4,80)  30      4    (x^4+5x^3+4x^2+6x+2)   A[12]=60 A[18]=300
                                                    A[24]=930 A[30]=1110
100  C(7,4,100) 24      2    (x^2+2x+2)             No aplicable
200  C(7,4,200) 12      2    (x^2+4)                No aplicable
```

En este ejemplo el programa muestra las tablas de sumar y multiplicar para el campo $\mathbb{F}_7$. Posteriormente, para cada u divisor de $\Delta = 400$ se muestra el código cíclico irreducible $\mathcal{C}(p, k, u)$, su longitud (n), su dimensión, el polinomio de chequeo de paridad asociado al código (Pol. Mín.) y por último la distribución de sus pesos. En caso de que la dimensión del código sea menor al grado del polinomio primitivo ingresado, no se presentará la distribución de sus pesos.

Si se desea implementar algún código de la lista, basta con realizar la siguiente división sintética para obtener el polinomio generador del código:

$$g(x) = (x^n - 1)/h(x),$$

donde $h(x)$ representa al polinomio de chequeo de paridad asociado al código y n a su longitud. Posteriormente, proceder como en el Teorema 2.

Lo que resta de este apéndice, es el código fuente del programa *DistrPCodIrred.c*. Su uso es permitido libremente con el reconocimiento de su origen.

DistrPCodIrred.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define N_max 85536
typedef unsigned char B8;

B8 Prim[N_max], invM[64], TSuma[128][128], TMult[128][128];
int P, k, traza[N_max], gama[N_max], egama[N_max];
int N; /* P^k-1 */
unsigned int Pk, Delta, delta;

char *sig[2] = {"","+"};

#define CM 32         /* CM=maximo valor de M */
int PI_Pr[6]={1,1,0,0,0,0}, PI_04[6]={1,1,1,0,0,0}, PI_08[6]={1,1,0,1,0,0},
    PI_16[6]={1,1,0,0,1,0}, PI_32[6]={1,0,1,0,0,1}, PI_09[6]={2,1,1,0,0,0},
    PI_27[6]={1,2,0,1,0,0}, PI_81[6]={2,1,0,0,1,0}, PI_25[6]={2,1,1,0,0,0},
```

```
    PI_49[6]={3,1,1,0,0,0};
int *PI, MPI[6], M, gp;

int leePol(f,p) /* Lee el polinomio primitivo p y lo almacena en el arreglo f */
B8 f[N_max];
char *p;
{
        int e;
        B8 a;

        memset(f,0,N_max);
        if(*(p++)!='(') { printf("Error 1\n"); exit(0); }
        while(*p!=')') {
            a=0;
            if(*p!='x') for(;(*p>47) && (*p<58);p++) a = a*10+(*p - 48);
            if(*p=='x') {
                p++;
                if(*p=='^') {
                    p++;
                    for(e=0;(*p>47) && (*p<58);p++) e = e*10+(*p - 48);
                } else e=1;
            } else e=0;
            if(!a) a++;
            f[e]=a;
            if(*p=='+') p++;
        }
}

int grado(x) /* Obtiene el grado del polinomio x */
B8 x[];
{
        int i;

        for(i=N_max-1;i>0;i--)
            if(x[i]) break;
        if((i==0)&&(!x[i])) return(-1);
        return(i);
}

int Divide(Q,R,A,B)   /* División sintética de A entre B */
B8 Q[], R[], A[], B[];
{
        int i, j, gb, gr, dif, fac;

        for(i=0;i<N_max;i++) { R[i] = A[i];  Q[i] = 0; }
        gb = grado(B);
        gr = grado(R);
        while(gr>=gb) {
            dif = gr - gb;
            fac = (invM[B[gb]]*R[gr])%P;
            Q[dif] = fac;
            fac = P-fac;
            for(i=gb;i>=0;i--)
                R[i+dif] = (R[i+dif]+fac*B[i])%P;
            gr = grado(R);
        }
}
```

```
int calinvMult() /* Calcula los inversos multiplicativos */
{
        int i, j;

        for(i=1;i<P;i++)
            for(j=1;j<P;j++)
                if(((i*j)%P)==1) invM[i] = j;
}

int codifica(X) /* Función que almacena al arreglo X en un entero */
B8 X[N_max];
{
        int i, gd, val, sal=0;

        gd=grado(X);
        for(i=0,val=1;i<=gd;i++,val*=P) sal += X[i]*val;
        return(sal);
}

int decodifica(X,val) /* Función inversa de codifica(X)  */
B8 X[N_max];
int val;
{
        int i;

        memset(X,0,N_max);
        for(i=0;val>0;i++,val/=P) X[i] = val % P;
}

int Tr() /* Calcula el arreglo traza */
{
        int i, j, m, l, r, val;
        B8 R[N_max], pg[N_max], Q[N_max], tem[N_max], X[N_max];

        memset(gama,0,N_max*sizeof(int));
        memset(egama,0,N_max*sizeof(int));
        memset(pg,0,N_max);
        pg[0]=1;
        for(i=0;i<N+1;i++) {
            Divide(Q,R,pg,Prim);
            val = codifica(R);
            gama[i] = val;
            egama[val] = i;
            for(j=0;j<k;j++) pg[k-j]=R[k-j-1];
            pg[0]=0;
        }
        egama[0] = -1;
        gama[N] = 1;
        for(i=0;i<Delta;i++) {
            memset(tem,0,N_max);
            for(j=0,m=1;j<k;j++,m = (m*P)%N) {
                decodifica(X,gama[(i*m)%N]);
                for(r=0;r<k;r++) tem[r] = (tem[r]+X[r])%P;
            }
            traza[i] = codifica(tem);
        }
        /*for(i=0;i<Delta;i++) {
            printf("gama^%02d = %02d  --> trazaA = %02d ;    ",
```

```
                    i,gama[i],traza[i]);
            printf("egama[%02d] = %02d\n",i,egama[i]);
        }*/
}

int imprime(x) /* Imprime x en formato polinomial */
B8 x[];
{
        char linea[512];
        int i, a, s=0;

        i = grado(x);
        printf("(");
        if(i>1) {
            s = 1;
            a = x[i];
            if(a!=1) printf("%dx^%d",a,i--);
            else printf("x^%d",i--);
        }
        for(;i>1;i--) {
            if(!x[i]) continue;
            a = x[i];
            if(a!=1) printf("+%dx^%d",a,i);
            else printf("+x^%d",i);
        }
        if(x[1]) {
            a = x[1];
            if(a!=1) printf("%s%dx",sig[s],a);
            else printf("%sx",sig[s]);
            s = 1;
        }
        if(x[0]) {
            a = x[0];
            printf("%s%d",sig[s],a);
        }
        printf(")");
}

int multN(pMin,pol,g) /* Función auxiliar de encuentraPolMino */
int pMin[N_max], pol[N_max], g;
{
        int i, j, l;
        B8 R[64][N_max], X[N_max];

        memset(R,0,64*N_max);
        for(i=0;i<2;i++)
            for(j=0;j<g;j++) {
                decodifica(X,gama[(pMin[j]+pol[i])%N]);
                for(l=0;l<k;l++)
                    R[i+j][l] = TSuma[R[i+j][l]][X[l]];
            }
        for(i=0;i<g+1;i++) pMin[i] = egama[codifica(&R[i][0])]%N;
}

int encuentraPolMino(RpMin,a) /* Encuentra el polinomio mínimo de gamma^a */
B8 RpMin[N_max];
int a;
{
```

```
        int g, i, pot=1;
        int pMin[N_max], pol[N_max], rM[N_max];

        memset(RpMin,0,N_max);
        pMin[0]=0;
        pot = a;
        g = 1;
        while(1) {
            if(P==2) pol[0] = pot;
            else pol[0] = (pot+(N/2))%N;
            pol[1] = 0;
            multN(pMin,pol,g);
            pot = (pot*P) % N;
            g++;
            if(pot==a) break;
        }
        for(i=0;i<g;i++) RpMin[i] = gama[pMin[i]];
}

int encuentraDistribucion(u) /* Encuentra distribución de pesos dada u */
int u;
{
    int i, j, z, orden, mu;
    int TZ[N_max], Z[526];
    B8 polMin[N_max];

    memset(TZ,0,N_max*sizeof(int));
    encuentraPolMino(polMin,u);
    for(orden=1,mu=u*P;orden<k;orden++,mu*=P) if((mu%N)==u) break;
    printf("%d\tC(%d,%d,%d)\t%d\t\t%d\t",u,P,k,u,N/u,orden);
    imprime(polMin); printf("\t\t");
    if(orden<k) {
        printf("No aplicable pues es de dimensión menor\n");
        return 0;
    }
    for(j=0;j<u;j++) {
        for(i=j,z=0;i<Delta;i+=u)
            if(!traza[i]) z++;
        TZ[z]++;
        Z[j]=z;
    }
    /*printf("\n"); for(j=0;j<u;j++) printf("Z[%d]=%d  ",j,Z[j]); printf("\n");*/
    for(i=delta;i>=0;i--) {
        /*if(TZ[i]) {
            printf("\nTZ[%d]=%d --> A[%d]=%d",i,TZ[i],(P-1)*((Delta/u)-i),TZ[i]*(N/u));
        }*/
        if(TZ[i]) printf("A[%d]=%d ",(P-1)*((Delta/u)-i),TZ[i]*(N/u));
    }
    printf("\n");
}

int valor(z) /* Función auxiliar de calTablas */
int z[];
{
        int i, v=0, w;

        for(i=0,w=1;i<M;i++,w*=P) v += ((z[i]%P)*w);
        return(v);
```

```
}

int gradoQ(x) /* Función auxiliar de calTablas */
int x[];
{
        int i;

        for(i=2*M-1;i>0;i--)
            if(x[i]) break;
        return(i);
}

int sumaQ(x,y,z)       /* Suma z(x) = x(x)+y(x) sin reducir modulo P */
int x[],y[],z[];  /* la reduccion va acargo de la funcion valor */
{
        int i;

        for(i=0;i<M;i++) z[i] = (x[i]+y[i]);
}

int multQ(x,y,z)       /* Multiplica z(x) = x(x)*y(x) sin reducir modulo P */
int x[],y[],z[];  /* la reduccion va acargo de la funcion valor */
{
        int sum[2*CM];
        int i, j, gs;

        memset(sum,0,2*M*sizeof(int));
        for(i=0;i<M;i++)                /* multiplica sum(x) = x(x)*y(x) */
            for(j=0;j<M;j++) sum[i+j] += y[j]*x[i];
        for(gs=gradoQ(sum);gs>=gp;gs--)  /* residuo z(x) = sum(x) Mod PI(x) */
            for(j=0;j<M;j++) sum[gs-M+j] += MPI[j]*sum[gs];  /* PI es mónico */
        memcpy(z,sum,M*sizeof(int));
}

/* Función que calcula las tablas de sumar y de
   multiplicar para la aritmética en el campo */
int calTablas()
{
    int Q=P;
    int i, j, k, l, NE=Q;
    int x[CM], y[CM], z[CM];

    switch(Q) {
        case 2:
            P=2;   PI=PI_Pr;  M=1;
            break;
        case 3:
            P=3;   PI=PI_Pr;  M=1;
            break;
        case 4:
            P=2;   PI=PI_04;  M=2;
            break;
        case 5:
            P=5;   PI=PI_Pr;  M=1;
            break;
        case 7:
            P=7;   PI=PI_Pr;  M=1;
            break;
```

```
case 8:
    P=2;   PI=PI_08;  M=3;
    break;
case 9:
    P=3;   PI=PI_09;  M=2;
    break;
case 11:
    P=11;  PI=PI_Pr;  M=1;
    break;
case 13:
    P=13;  PI=PI_Pr;  M=1;
    break;
case 16:
    P=2;   PI=PI_16;  M=4;
    break;
case 17:
    P=17;  PI=PI_Pr;  M=1;
    break;
case 19:
    P=19;  PI=PI_Pr;  M=1;
    break;
case 23:
    P=23;  PI=PI_Pr;  M=1;
    break;
case 25:
    P=5;   PI=PI_25;  M=2;
    break;
case 27:
    P=3;   PI=PI_27;  M=3;
    break;
case 29:
    P=29;  PI=PI_Pr;  M=1;
    break;
case 31:
    P=31;  PI=PI_Pr;  M=1;
    break;
case 32:
    P=2;   PI=PI_32;  M=5;
    break;
case 47:
    P=47;  PI=PI_Pr;  M=1;
    break;
case 49:
    P=7;  PI=PI_49;   M=2;
    break;
case 53:
    P=53;  PI=PI_Pr;  M=1;
    break;
case 71:
    P=71;  PI=PI_Pr;  M=1;
    break;
case 81:
    P=3;   PI=PI_81;  M=4;
    break;
    default:
    printf("Error campo inválido o no soportado\n");
    exit(-1);
    break;
```

```
    }
    gp = M;
    for(i=0;i<M+1;i++) MPI[i] = (P - PI[i])%P; /*MPI(x)=-PI(x) sobre GF(P)*/
    for(i=0;i<NE;i++) {            /* Tabla para sumar */
        for(k=0,l=i;k<M;k++,l/=P) x[k] = l % P;
        for(j=0;j<NE;j++) {
            for(k=0,l=j;k<M;k++,l/=P) y[k] = l % P;
            sumaQ(x,y,z);          /* Suma x+y y el resultado en z */
            TSuma[i][j] = (B8) valor(z);
        }
    }
    for(i=0;i<NE;i++) {            /* Tabla para multiplicar */
        for(k=0,l=i;k<M;k++,l/=P) x[k] = l % P;
        for(j=0;j<NE;j++) {
            for(k=0,l=j;k<M;k++,l/=P) y[k] = l % P;
            multQ(x,y,z);          /* Multiplica x*y y el resultado en z */
            TMult[i][j] = (B8) valor(z);
        }
    }
    printf("\nTabla de sumar GF(%d)\n",P);
    for(i=0;i<Q;i++) {
        for(j=0;j<Q;j++) printf("%2d ",TSuma[i][j]);
        printf("\n");
        }
    printf("\n");
    printf("Tabla de multiplicar GF(%d)\n",P);
    for(i=0;i<Q;i++) {
        for(j=0;j<Q;j++) printf("%2d ",TMult[i][j]);
        printf("\n");
    }
    printf("\n");
}

int main(argc, argv) /* Rutina principal */
int argc;
char *argv[];
{
    int off = 0, u, i;

    if(!strcmp(argv[1],"-p")) {
        P = atoi(argv[2]);
        off = 2;
    }
    leePol(Prim,argv[1+off]);
    calTablas();
    calinvMult();
    printf("Usaremos el pol. primitivo: "); imprime(Prim); printf("\n");
    k = grado(Prim);

        for(i=0,Qv=1;i<k-1;i++) Qv *= Q;
        delta=(Qv-1)/(Q-1);
        Qv *= Q;
        N = Qv-1;
        Delta = (N/(Q-1));

    for(i=0,Pk=1;i<k-1;i++) Pk *= P;
        delta=(Pk-1)/(P-1);
    Pk *= P;
```

```
    N = Pk - 1;
    Delta = (N/(P-1));
    printf("p=%d  k=%d  Delta:=(p^k-1)/(p-1)=%d  n:=(p^k-1)/u\n\n",P,k,Delta);
    printf("u\tCódigo\t\tLong(n)\t\tDim\tPol. Mín.\t\tDistribución de pesos\n");
    Tr();
    for(u=1;u<Delta;u++) {
        if((Delta%u)==0) encuentraDistribucion(u);
    }
    printf("\n");
}
c
```

Bibliografía

[1] Delsarte, P. (1975). On subfield subcodes of Reed-Solomon codes, *IEEE Trans. Inform. Theory*, 21(5), 575-576.

[2] Lidl, R. y Niederreiter, H. *Introduction to finite fields and their applications*. Cambridge Univeristy Press, Cambridge, Reino Unido, 1986.

[3] MacWilliams, F.J. y Sloane, N.J.A. *The Theory of Error-Correcting Codes*. North-Holland Publishing Company, Amsterdam, Países Bajos, 1997.

[4] Schmidt, B., y White, C. (2012). All two-weight irreducible cyclic codes?, *Finite Fields and Their Applications*, 8(1), 1–17.

[5] Vega, G. (2015). A critical review and some remarks about one- and two-weight irreducible cyclic codes. *Finite Fields and Their Applications*, 33, 1-13.

Lista de Símbolos

Nota. Los símbolos que aparecen sólo en un contexto restringido no se encuentran listados. De ser apropiado, se proporciona una referencia de página para la primera aparición del símbolo.

Símbolo	**Descripción**	**Página**
$\|A\|$	la cardinalidad (= número de elementos) del conjunto finito A	
$\mathbb{Z}$	el anillo de los números enteros	16
$\mathbb{C}$	el campo de los números complejos	21
$a \mid b$	a divide a b	
$a \equiv b \pmod{n}$	a es congruente con b módulo n	
$\mathrm{mcd}(a, b)$	el máximo común divisor de a y b	
$\mathrm{ord}_n(q)$	el orden multiplicativo de q módulo n	18
$< a >$	el grupo cíclico generado por a	15
$< a^d >$	el subgrupo cíclico de $< a >$ de índice d	
$\|G\|$	el orden del grupo finito G	
$\mathrm{grado}(g(x))$	el grado del polinomio $g(x)$	17
$< g(x) >$	el ideal principal generado por el polinomio $g(x)$	17
$\mathbb{Z}_n$	el anillo de los números enteros módulo n	18
C_s	la clase ciclotómica de s módulo n	18
$M_s(x)$	el polinomio mínimo de $\beta^s \in \mathbb{F}_{q^k}^*$ sobre $\mathbb{F}_q$	19
$\mathbb{F}_p$	el campo primo de orden p	22
$\mathbb{F}_q$	el campo finito de orden q	16
$\mathbb{F}_{q^k}$	la extensión de grado k de $\mathbb{F}_q$	18
$\mathbb{F}_{q^k}^*$	el grupo multiplicativo de elementos distintos de cero de $\mathbb{F}_{q^k}$	19
$\mathbb{F}_q[x]$	el anillo de polinomios con indeterminada x sobre el campo $\mathbb{F}_q$	17
$\mathbb{F}_q[x]/(x^n - 1)$	el anillo de polinomios con indeterminada x sobre el campo $\mathbb{F}_q$ de grado a lo más $n - 1$	16
$\mathbb{F}_q(\gamma)$	la extensión de $\mathbb{F}_q$ obtenida por agregar la raíz γ	30
$\mathrm{w}(v)$	el peso de Hamming de v	16
$[n, k]$	el código lineal de longitud n y dimensión k	16
$[n, k, d]$	el código lineal de longitud n, dimensión k y distancia mínima d	16
$\mathrm{Tr}_{\mathbb{F}_{q^k}/\mathbb{F}_q}(\alpha)$	la traza de $\alpha \in \mathbb{F}_{q^k}$ sobre $\mathbb{F}_q$	19
$\mathrm{Tr}(\alpha)$	la traza absoluta de $\alpha \in \mathbb{F}_{q^k}$	19
$h_a[x]$	el polinomio mínimo de $\gamma^{-a} \in \mathbb{F}_{q^k}^*$ sobre $\mathbb{F}_q$	19
$\mathcal{C}(q, k, u)$	código cíclico irreducible sobre $\mathbb{F}_q$	20
ξ_s	la s-ésima raíz compleja de la unidad $e^{2\pi\sqrt{-1}/s}$	22
ψ_1	el caracter aditivo canónico de $\mathbb{F}_{q^k}$	22
χ_0	el caracter multiplicativo trivial de $\mathbb{F}_{q^k}$	22
$\bar{\chi}$	el conjugado del caracter χ	22
L°	el conjunto de caracteres multiplicativos del grupo abeliano finito L	22
$G_L(\chi)$	suma Gaussiana del caracter multiplicativo $\chi \in L^*$	22
Γ	el conjunto aniquilador de U en L°	22