Una Introducción a la Teoría de Códigos Convolucionales

Viviana Carolina Guerrero Pantoja

UNIVERSIDAD DEL VALLE
FACULTAD DE CIENCIAS NATURALES Y EXACTAS
DEPARTAMNETO DE MATEMÁTICAS
PROGRAMA DE MAESTRÍA EN CIENCIAS MATEMÁTICAS
SANTIAGO DE CALI, 2018

Una Introducción a la Teoría de Códigos Convolucionales

Viviana Carolina Guerrero Pantoja

Trabajo de grado presentado como requisito parcial para optar al título de Magíster en Ciencias Matemáticas

Director:
John Hermes Castillo Gómez, Ph.D.
Universidad de Nariño

Co-director:
Juan Miguel Velásquez Soto, Ph.D.
Universidad del Valle

UNIVERSIDAD DEL VALLE
FACULTAD DE CIENCIAS NATURALES Y EXACTAS
DEPARTAMNETO DE MATEMÁTICAS
PROGRAMA DE MAESTRÍA EN CIENCIAS MATEMÁTICAS
SANTIAGO DE CALI, 2018

Agradecimentos

Quiero expresar por medio de este trabajo mi gratitud, a Dios por haberme permitido ingresar al programa de Maestría en Ciencias Matemáticas de la Universidad del Valle y vivir esta experiencia que aunque no ha sido fácil, logre cumplir esta meta de forma satisfactoria.

A todas aquellas personas que han influido en mi carrera, por su apoyo incondicional tanto a nivel académico como personal. De manera especial al profesor Jhon Castillo, director de este trabajo, quien creyó en mí desde el inicio de mis estudios en pre-grado, por alentarme a continuar con mis estudios y me ha apoyado de manera personal y académica en todo este proceso; gracias por la paciencia, tiempo y dedicación dada a la orientación de este trabajo sin lo cual no habría sido posible su realización. Así como también al profesor Juan Miguel Velásquez, co-director de este trabajo, su apoyo y consejos a nivel institucional y personal, fueron fundamentales en éste proceso. Agradezco también a los profesores Diana Bueno y Yamidt Bermúdez, jurados de este trabajo, por el tiempo dedicado a la revisión y por las sugerencias que permitieron mejorarlo.

Expreso además mis agradecimientos a la Universidad del Valle por darme la oportunidad de continuar con mi formación tanto personal como académica. A los docentes del departamento de matemáticas que contribuyeron en mi formación dentro del programa de Maestría en Ciencias Matemáticas. Especialmente al profesor Álvaro Garzón, sus enseñanzas, su disciplina y su exigencia me aportaron mucho y me inspiran a seguir en este camino.

A mis compañeros de estudio, por todo su apoyo y por los buenos momentos compartidos; especialmente a mis amigas Luz Elena Dominguez y Yazmin Rivera que fueron un gran apoyo pues siempre estuvieron y sé que estarán en adelante en los momentos más difíciles para ser esa voz de aliento cuando creo no poder seguir.

A los profesores del departamento de matemáticas de la Universidad de Nariño: Catalina Rúa, John Castillo, Fernando Benavides y Wilson Mutis, quienes siempre me motivaron a seguir con mis estudios y por brindarme su amistad y apoyo.

Al grupo de investigación ALTENUA: Álgebra, Teoría de Números y Aplicaciones ERM, en especial a su director el profesor Carlos Trujillo por confiar en mí y mediante el proyecto de investigación 110356935047 "Construcciones de Conjuntos $B_h[g]$, propiedad de Midy, y algunas aplicaciones" de COLCIENCIAS, me apoyó con la financiación inicial de mis estudios de maestría.

Finalmente agradezco a las personas que siempre han estado presentes en cada etapa de mi vidas apoyándome incondicionalmente; mis padres y familiares. A Jesús Fajardo que fue

mi mayor apoyo en todo este proceso, por entender mi ausencia y motivarme siempre. Y Andrés Jaramillo mi gran amigo.

Resumen

La teoría de codificación es una de las diversas ramas de la matemática que es atractiva debido a la interacción activa entre las invenciones sofisticadas de ingeniería y las matemáticas abstractas. En este documento se presentan algunos de los conceptos básicos de los códigos convolucionales, los cuales son un tipo de códigos correctores de errores diferentes de los códigos de bloque, cuya principal diferencia entre ellos es la introducción del concepto de memoria para el caso convolucional. Sin embargo, se muestra que algunas de las definiciones aplicables a los códigos de bloque lineales y cíclicos pueden darse también en el contexto de los códigos convolucionales.

Además, se expone cómo construir el codificador físico de un código convolucional, cuatro representaciones analíticas y dos representaciones gráficas para realizar el proceso de codificación y la forma de pasar de una representación a otra. También se presenta una breve descripción de los códigos convolucionales cíclico tomando como principal referencia el artículo "On Cyclic Convolutional Codes" de la autoría de Heide Gluesing-Luerssen y Wiland Schmale del año 2004. En este trabajo se muestra como el programa computacional `SAGE` es una herramienta útil para facilitar algunos cálculos, que a mano pueden resultar tediosos; permitiendo exhibir ejemplos más grandes a los que suelen encontrarse en la literatura. Este objetivo se logra a partir de la implementación de algoritmos basados en los conceptos aquí presentados.

Palabras clave: Código Convolucional, Código Convolucional Cíclico, σ-ciclicidad, Factores invariantes, Anillo de polinomios torcido.

Abstract

Coding theory is one of several branches of mathematics that is attractive because of the active interaction between sophisticated engineering inventions and abstract mathematics. This dissertation presents some of the basic concepts of convolutional codes, which are a type of error-correcting codes different from block codes, whose main difference between them is the introduction of the concept of memory on the convolutional case. However, it is shown how some of the definitions applicable to linear and cyclic block codes can also be given in the context of convolutional codes.

In addition, we explain how to construct the physical encoder of a convolutional code, four analytical representations and two graphic representations to perform the encoding process and how to pass from one representation to another; as well as a brief description of the algebraic structure of cyclical convolutional codes taking as main reference the article "On Cyclic Convolutional Codes" wrote by Heide Gluesing-Luerssen and Wiland Schmale in 2004. This work shows how the mathematical software system `SAGE` is a useful tool to facilitate some calculations, which can be tedious by hand; so we can exhibit larger examples than those usually found in the literature. This objective is achieved through the implementation of algorithms based on the concepts presented here.

Índice general

Lista de símbolos

$\mathbb{N}$	Conjunto de números naturales.
$\mathbb{N}_0$	Conjunto de números naturales unido con el cero.
$\mathbb{F}_q$	Campo finito de orden q, donde q es una potencia de un primo.
$\mathbb{F}$	Campo arbitrario.
$\mathbb{F}[z]$	Anillo de polinomios en la indeterminada z.
$\mathbb{F}(z)$	Campo de funciones racionales.
$\mathbb{F}[[z]]$	Anillo de series de potencia formales.
$\mathbb{F}((z))$	Campo de series formales de Laurent.
R_n	Anillo cociente de polinomios módulo $x^n - 1$.
$R[x; \alpha, \delta]$	Anillo de polinomios torcido sobre R.
$R_n[z; \sigma]$	Anillo de polinomios torcido sobre R_n.
gr	Grado de un polinomio.
ker	Kernel.
det	Determinante.
gen	Generado.
Aut	Automorfismos.
im	Imagen.

Índice de figuras

Índice de algoritmos

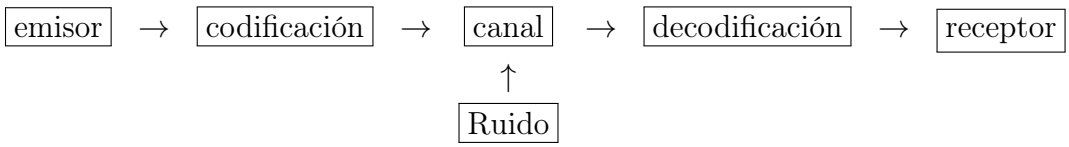
Introducción

Un código es el resultado de la aplicación de un sistema de signos y reglas que permiten cambiar la forma de un mensaje para que pueda ser transmitido a través de un medio. En nuestra cotidianidad hacemos uso de ciertos tipos de códigos como son: el lenguaje cotidiano, la notación musical y las señales de tránsito, entre otros; los cuales no son códigos en el sentido matemático, pero en su esencia tienen los mismos objetivos. El lenguaje cotidiano es un código de longitud variable, el cual no tiene buenas propiedades en cuanto a la detección y corrección de errores. En este caso, algunas veces es posible detectar errores por sintaxis o contexto, pero en ciertas ocasiones es muy difícil corregirlo. Por ejemplo, si al transmitir un mensaje se recibe la palabra "Xo", esta puede ser interpretada por su receptor como "No", "Yo" o "Lo" que son palabras código dependiendo del contexto; pues en el caso de que esta se encuentre en la frase "Xo te quiero" las posibles opciones serían "No te quiero" o "Yo te quiero" sin embargo, no se puede asegurar cual de los dos fue el mensaje enviado; es decir es posible detectar el error pero no necesariamente corregirlo. Inclusive, algunas veces no se puede ni siquiera detectar el error, por ejemplo si queremos transmitir la palabra "pila", en un listado de compras familiares, se podría cometer un error al escribir en nuestro teléfono y enviar "pala".

En general cuando se transmite un mensaje este es enviado a través de un medio, el cual se denomina *canal* y se escoge dependiendo del tipo de información. Para iniciar el proceso es necesario expresar el mensaje, denominado *palabras fuente*, en términos acordes a los que el canal está capacitado para enviar como *palabras código*. A este proceso se le denomina *codificación*. Una vez se codifica el mensaje, este es llevado a través del canal hasta el punto donde es recibido por el receptor en los mismos términos que maneja el canal.

Luego se requiere otro proceso, el cual tiene como objetivo que el receptor entienda el mensaje y sea capaz de detectar o corregir los posibles errores generados en la transmisión del mensaje original. Estos errores pueden originarse por distintos fenómenos entre los cuales podrían mencionarse: errores humanos, ruido, interferencias electromagnéticas, perturbaciones eléctricas naturales (tales como relámpagos) o por fuentes artificiales, por ejemplo las líneas de transmisión de alto voltaje, circuitos de conmutación de una computadora digital cercana, entre otras. Una manera de esquematizar todo el proceso aquí descrito es mediante

el siguiente diagrama.



El origen de la Teoría de Códigos está ligado con el trabajo del ingeniero electrónico y matemático estadounidense Claude Shannon y su artículo "A Mathematical Theory of Communication" [25], el cual dio origen a la Teoría de Códigos y la Teoría de la Información. La Teoría de Códigos Correctores de Errores, tiene como objetivo principal construir códigos que permitan enviar la mayor información posible, detectar los errores producidos en la transmisión y corregirlos. Sin embargo, el artículo de Shannon no contenía una construcción explícita de tales códigos. Desde entonces diversas técnicas matemáticas se utilizan con este fin, dando como resultado diferentes familias de códigos, tales como los códigos de bloque, códigos lineales, códigos cíclicos, códigos convolucionales, códigos proyectivos, entre otros. En la actualidad son varias las ramas de la matemática involucradas en la construcción de estos códigos; entre ellas están Álgebra Lineal, Teoría de Grupos, Teoría de Anillos, Teoría de Campos Finitos, Teoría de Módulos, Geometría Algebraica, Combinatoria, Teoría de Numeros.

El concepto de código convolucional fue presentado por Peter Elias [4] en 1955, mientras que la teoría matemática de estos códigos fue desarrollada por G. David Forney Jr. [5] en la década de 1970, y posteriormente Robert McEliece [17] en 1998 dio un planteamiento desde el álgebra moderna. Un abordaje profundo puede consultarse en los libros de Shun Lin y Daniel Costello [15], Philippe Piret [20], Ajay Dholakia [3], Rolf Johannesson y Kamil Sh. Zigangirov [14], entre otros.

Los códigos convolucionales y los códigos de bloque son los tipos de códigos más utilizados en la práctica de la ingeniería, por su sencillez y utilidad. Estos códigos, no solo poseen buenas propiedades generales sino también algoritmos eficientes de codificación y decodificación, hecho que conduce a la necesidad de mantener una fundamentación matemática fuerte para el diseño de códigos útiles. En consecuencia, la teoría de codificación es una de las diversas ramas de la matemática que es atractiva debido a la interacción activa entre las invenciones sofisticadas de ingeniería y matemáticas abstractas.

Los códigos convolucionales son un tipo de códigos correctores de errores diferentes de los códigos de bloque. La principal diferencia entre ellos es la introducción del concepto de ***memoria***, esto es la codificación en un tiempo t no depende solo de la secuencia de entrada en ese instante sino también de un número finito de entradas anteriores. Sin embargo algunas de las definiciones aplicables a los códigos de bloque lineales y cíclicos pueden darse también en el contexto de los códigos convolucionales, por ejemplo algunas de ellas se muestran en la tabla 1.

Códigos de Bloque Lineales	Códigos Convolucionales
Peso de Hamming	Peso de Hamming de un vector polinómico
Distancia	Distancia entre dos vectores polinómicos
Distancia Mínima	Distancia Libre
Matriz Generadora	Matriz Generadora
Matriz de Control de Paridad	Matriz de Control de Paridad
Ciclicidad	σ-Ciclicidad

Tabla 1: *Algunos conceptos semejantes.*

En este documento se presentan los códigos convolucionales desde dos puntos de vista, inicialmente a partir de las series de Laurent, se muestra las diferentes formas de representar tanto las k-secuencias de información como las n-secuencias código, y el proceso para pasar de una representación a otra. Sin embargo, también se muestra un segundo enfoque, ya no en el contexto de las series formales de potencias sino tomando las palabras código como elementos de un anillo de polinomios el cual se usa principalmente para trabajar con códigos convolucionales cíclicos.

Adicionalmente, se expone como construir el codificador físico de un código convolucional y cómo usarlo para realizar el proceso de codificación. También se muestra como determinar las secuencias generadoras discretas a partir del codificador físico. Se exhiben diferentes representaciones de un codificador convolucional y la operación convolucional usando diferentes procedimientos, como por ejemplo: matriz generadora discreta semi-infinita, secuencias generadoras polinomiales; para los cuales se construyeron algoritmos implementados en el programa computacional `SAGE`. Por otro lado se estudian dos representaciones gráficas como son: el diagrama de estado y el diagrama enrejado y cómo codificar una secuencia de información a partir de estas.

En este trabajo se muestra que es posible definir el concepto de código convolucional dual y se muestra como obtener la matriz de control de paridad mediante la forma de Smith de una matriz generadora polinomial del código convolucional dado. Finalmente, se presenta el estado del arte acerca de la estructura algebraica de los códigos convolucionales cíclicos tomando como referencia principal el artículo "On Cyclic Convolutional Codes" de Heide Gluesing-Luerssen y Wiland Schmale [9], mostrando como `SAGE` puede ser una herramienta útil para conseguir resultados en proyectos de investigación futuros.

Este trabajo de investigación se divide en tres capítulos y contiene un apéndice en el cual se exponen los algoritmos con una implementación en el programa computacional `SAGE` que se usan en el desarrollo de este documento para facilitar algunos cálculos. En el Capítulo 1 se presentan algunos conceptos básicos de Códigos Lineales y Teoría de Módulos necesarios para el desarrollo de los temas de los capítulos siguientes, así como también la definición de forma de Smith de una matriz polinomial y un método para determinarla, y el concepto de anillo de polinomios torcido que es fundamental en el desarrollo del Capítulo 3.

El segundo capítulo contiene inicialmente la notación para elementos en un código con-

volucional, su definición formal y la matriz generadora del mismo, se presentan diferentes representaciones para el codificador convolucional y la operación convolucional como son: codificador físico, secuencias generadoras discretas, matriz generadora discreta semi-infinita, generadores polinomiales, matriz generadora polinomial y dos representaciones gráficas; se expone un algoritmo de decodificación gráfico conocido como el algoritmo de Viterbi y para finalizar la definición de código convolucional dual y un método para determinar la matriz de control de paridad a partir de una matriz generadora polinomial dada.

En el último capítulo se exponen las bases para el estudio de los códigos convolucionales cíclicos, algunas implementaciones en `SAGE` que son útiles para determinar una matriz generadora de un código convolucional σ-cíclico y cierta relación con los números de Midy.

Capítulo 1

Preliminares

En este capítulo con el fin de facilitar la comprensión de los temas a tratar en los demás capítulos, se presenta una serie de nociones y resultados básicos de códigos lineales y teoría de módulos, así como también el concepto de forma normal de Smith de una matriz polinomial y la definición formal de anillo de polinomios torcido.

1.1. Códigos Lineales

En esta sección se presentan algunos de los conceptos básicos de códigos de bloque, códigos lineales y códigos lineales cíclicos que pueden darse también en el contexto de los códigos convolucionales como se verá en el desarrollo de los Capítulos 2 y 3.

Un conjunto finito $A = \{a_1, a_2, \ldots, a_q\}$ se denomina *alfabeto*, los elementos de A se llaman *símbolos*, una sucesión de n símbolos se denomina una *n-úpla* o *palabra de longitud n* sobre A. Sea A^n el conjunto de todas las n-uplas sobre A, así cualquier subconjunto no vacío C de A^n es un *código de bloque q-ario*. Cada elemento en C se llama *palabra código* y si C contiene M elementos, entonces se dice que el código C tiene longitud n y tamaño M o simplemente que C es un (n, M)-código. Si C no es de bloque entonces C se denomina código de longitud variable.

Un ejemplo de código de longitud variable es el código C sobre el alfabeto $A = \{0, 1\}$ definido por $C = \{(0), (1, 0), (1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 1, 1)\}$.

Usualmente, A se toma como el campo finito $\mathbb{F}_q$, donde q es una potencia de un primo. Así,

cuando el alfabeto es $\mathbb{F}_2, \mathbb{F}_3$ ó $\mathbb{F}_4$, se dice que C es un *código binario, ternario o cuaternario*, respectivamente.

Ejemplo 1.1. Los siguientes son códigos de bloque

i) $C_1 = \{(0,0,0,0),(1,0,1,0),(0,1,0,1),(1,1,1,1)\}$ sobre $\mathbb{F}_2$ es un $(4,4)$-código binario.

ii) $C_2 = \{(2,1,0),(1,2,0),(1,2,2),(2,1,1)\}$ sobre $\mathbb{F}_3$ es un $(3,4)$-código ternario.

La capacidad de detectar o corregir errores de un código viene dada en términos de la distancia mínima, la cual se define a partir de la distancia de Hamming.

Definición 1.1.
La distancia de Hamming entre n-uplas es el número de componentes en que difieren, $d(v,v') = \#\{i : v_i \neq v'_i\}$. *La distancia mínima de un código es la menor distancia de Hamming entre las palabras del código,*

$$d(C) = \min_{x,y \in C}\{d(x,y)\}.$$

Un (n,M)-código de bloque con distancia mínima d es un (n,M,d)-código. Es bien conocido que un código C con distancia mínima d, puede detectar $(d-1)$ errores y corregir $\lfloor (d-1)/2 \rfloor$ errores.

Ejemplo 1.2.
Se denotan las palabras del código C_1 del ejemplo 1.1, como $c_0 = (0,0,0,0)$, $c_1 = (1,0,1,0)$, $c_2 = (0,1,0,1)$ y $c_3 = (1,1,1,1)$.
Las distancias de Hamming son:

$$d(c_0,c_1) = d(c_0,c_2) = d(c_1,c_3) = d(c_2,c_3) = 2 \quad \text{y} \quad d(c_0,c_3) = d(c_1,c_2) = 4.$$

Luego la distancia mínima de C_1 es 2, esto es C_1 es un $(4,4,2)$-código. El cual puede detectar 3 errores y corregir 1 error.

Observación 1.1. El concepto de distancia de Hamming se utiliza en el proceso de decodificación. Sea C un código, supóngase que palabras código son enviadas sobre un canal de

comunicación. Si x es una palabra recibida, la *decodificación por distancia mínima* decodifica x a c_x, si $d(x, c_x)$ es mínima entre todas las palabras código, es decir

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

Ejemplo 1.3.

Para el código $C = \{(0,1,1,0,1), (0,0,0,1,1), (1,0,1,1,0), (1,1,0,0,0)\}$. Se usa la decodificación por distancia mínima para decodificar la palabra recibida $(0,1,1,1,1)$.

Se calcula las distancias:

$$d((0,1,1,1,1), (0,1,1,0,1)) = 1,$$

$$d((0,1,1,1,1), (0,0,0,1,1)) = 2,$$

$$d((0,1,1,1,1), (1,0,1,1,0)) = 3,$$

$$d((0,1,1,1,1), (1,1,0,0,0)) = 4.$$

Se decodifica $(0,1,1,1,1)$ como la palabra código $(0,1,1,0,1)$.

Definición 1.2.

El peso de Hamming de una palabra x en $\mathbb{F}_q^n$ es el número de sus componentes no nulas, $wt(x) = \#\{i : x_i \neq 0\}$. El peso mínimo de Hamming de C, denotado por $wt(C)$ es el mínimo de los pesos de todas las palabras no nulas de C. Es decir,

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

Ejemplo 1.4.

Nuevamente se considera el código C_1 del ejemplo 1.1 y las palabras de este como se denotaron en el ejemplo 1.2. Luego, los pesos de Hamming de las palabras no nulas del código C_1 son: $wt(c_1) = 2$, $wt(c_2) = 2$ y $wt(c_3) = 4$. Por tanto el peso mínimo de C_1 es $wt(C) = 2$.

Definición 1.3.

Un código lineal q-ario C de longitud n es un subespacio vectorial de $\mathbb{F}_q^n$ de dimensión k. En este caso se dice que C es un $[n, k]$-código. Además, una matriz generadora de C es una matriz $G \in \mathbb{F}_q^{k \times n}$ cuyas filas forman un base para C.

Cabe notar que la matriz generadora siempre existe y es de rango k, por tal motivo un código lineal se puede ver también como la imagen de la transformación lineal inyectiva $\varphi : \mathbb{F}_q^k \to \mathbb{F}_q^n$ definida por $\phi(u) = uG$ para todo $u \in \mathbb{F}_q^k$ y G una matriz generadora. Esta transformación representa la codificación.

Ejemplo 1.5.

Si se consideran los códigos C_1 y C_2 del ejemplo 1.1. Se puede ver que el código C_1 es un código lineal ya que cumple la condición de ser cerrado bajo la suma de sus elementos condición que es necesaria para ser subespacio vectorial, la cual además es suficiente para el caso de los subespacios sobre $\mathbb{F}_2$ pues los escalares son 0 y 1. Pero el código C_2 no es lineal dado que el elemento $(0,0,0) = (2,1,0) + (1,2,0) \notin C_2$.

El código lineal C_1 tiene a lo más dos elementos linealmente independientes (L.I), por ejemplo $(1,0,1,0)$ y $(0,1,0,1)$ son L.I., pero también lo son $(0,1,0,1)$ y $(1,1,1,1)$. Así que matrices generadoras para el código C_1 son:

$$G_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \quad \text{o} \quad G_2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

En general, determinar la distancia mínima de un (n, M)-código de bloque arbitrario, es una tarea que puede llevar mucho tiempo, dado que es necesario calcular $\binom{M}{2}$ distancias de Hamming. Pero, para un $[n, k]$-código lineal sobre $\mathbb{F}_q$ este cálculo se simplifica considerablemente, ya que la distancia mínima satisface la siguiente igualdad, ver [16, Teorema 4.3.8]

$$d(C) = \min_{x,y \in C}\{d(x,y)\} = \min_{c \in C}\{w(c)\} = wt(C).$$

por lo cual solo se requiere calcular $M - 1$ pesos posibles donde $M = q^k$ es el tamaño de C.

Ejemplo 1.6.

Para determinar la distancia mínima del código lineal C_1 en lugar de calcular las 6 distancias que se calcularon en el ejemplo 1.2, basta con calcular los 3 pesos posibles que se calcularon en el ejemplo 1.4.

Un $[n, k]$-código lineal C por ser un subespacio vectorial, tiene asociado un subespacio

vectorial denominado el dual $C^{\perp}$, el cual también es un código lineal y se define a continuación

Definición 1.4.

Sea C un $[n,k]$-código lineal. El conjunto

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

es el código dual de C.

De esta manera el *código dual* de un código lineal, se define como el complemento ortogonal del código lineal C. Además, se puede probar que $C \oplus C^{\perp} = \mathbb{F}_q^n$.

Dado que $C^{\perp}$ es un código lineal, este tiene una matriz generadora H mediante la cual se define la matriz de control de paridad de C. Así, una *matriz de control de paridad* de C es una matriz generadora de $C^{\perp}$, tal matriz siempre existe y es de dimensión $n-k \times n$. Además, si G es una matriz generadora de C se satisface que $GH^T = 0$, lo cual implica que para todo $c \in C$, $cH^T = 0$.

Ejemplo 1.7.

Determinar el código dual $C^{\perp}$ del código lineal binario

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

Una base para C es $\{(1,0,1,1,0), (0,1,0,1,1)\}$. De ahí que una matriz generadora para C es

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

Ahora, se calcula una base para el espacio nulo del sistema $\mathbf{x}\, G = \mathbf{0}$, con $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$ en $\mathbb{F}_2^5$. El generado por estos elementos por definición es el código dual de C

$$\mathbf{x}\, G^T = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} x_1 + x_3 + x_4 & x_2 + x_4 + x_5 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}.$$

Así una base para $C^{\perp}$ es $\{(1,0,1,0,0),(1,1,0,1,0),(0,1,0,0,1)\}$ por lo tanto

$$\begin{aligned} C^{\perp} =&\{(0,0,0,0,0),(1,0,1,0,0),(1,1,0,1,0),(0,1,0,0,1),\\ &(0,0,1,1,1),(1,1,1,0,1),(0,1,1,1,0),(1,1,0,1,1)\}. \end{aligned}$$

Una matriz de control de paridad de C es

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

Usando la matriz de control de paridad, David Slepian en 1960 dio un método general de decodificación para códigos lineales [27], del cual se puede encontrar mayor información en [16, Sec. 4.8].

Definición 1.5.

Sea C un $[n,k]$-código con matriz de paridad H. Si $v \in \mathbb{F}_q^n$, el síndrome de v se define por $s(v) = s_H(v) = vH^T$.

En particular para cada $e \in \mathbb{F}_q^n$ todos los elementos del conjunto $e + C = \{e + c : c \in C\}$ tienen el mismo síndrome, pues por linealidad

$$(e+c)H^T = eH^T + cH^T = eH^T \quad \text{para todo} \quad c \in C$$

La decodificación por síndromes, que asigna a cada síndrome el vector de peso mínimo en el conjunto $e+C$ correspondiente llamado *líder*, permitirá decodificar cualquier palabra código enviada según la capacidad de detección y corrección de errores.

Ejemplo 1.8.

Considere el código C del ejemplo 1.7. Suponga que se recibió la palabra $x = (1,1,0,1,1)$. Se usa la matriz de control de paridad

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

para decodificar la palabra recibida.

Primero se calculan los conjuntos $e + C$:

$$C_0 = (0,0,0,0,0) + C = \{(0,0,0,0,0), (1,0,1,1,0), (0,1,0,1,1), (1,1,1,0,1)\}$$

$$C_1 = (0,0,0,0,1) + C = \{(0,0,0,0,1), (1,0,1,1,1), (0,1,0,1,0), (1,1,1,0,0)\}$$

$$C_2 = (0,0,0,1,0) + C = \{(0,0,0,1,0), (1,0,1,0,0), (0,1,0,0,1), (1,1,1,1,1)\}$$

$$C_3 = (0,0,1,0,0) + C = \{(0,0,1,0,0), (1,0,0,1,0), (0,1,1,1,1), (1,1,0,0,1)\}$$

$$C_4 = (0,1,0,0,0) + C = \{(0,1,0,0,0), (1,1,1,1,0), (0,0,0,1,1), (1,0,1,0,1)\}$$

$$C_5 = (1,0,0,0,0) + C = \{(1,0,0,0,0), (0,0,1,1,0), (1,1,0,1,1), (0,1,1,0,1)\}$$

$$C_6 = (0,0,1,0,1) + C = \{(0,0,1,0,1), (1,0,0,1,1), (0,1,1,1,0), (1,1,0,0,0)\}$$

$$C_7 = (0,1,1,0,0) + C = \{(0,0,0,0,0), (1,1,0,1,0), (0,0,1,1,1), (1,0,0,0,1)\}$$

Como, $s(x) = xH^T = (1,0,1)$ y el síndrome de los elementos de C_5 es el mismo, además la palabra de peso mínimo en C_5 es $(1,0,0,0,0)$. Por tanto la palabra x se decodifica como $c = x + (1,0,0,0,0) = (1,1,0,1,1) + (1,0,0,0,0) = (0,1,0,1,1)$.

Una clase importante de códigos lineales son los códigos cíclicos, para los cuales su estructura matemática facilita los procesos de codificación y decodificación. Además, un código cíclico de longitud n es determinado totalmente por un polinomio de grado menor que n, denominado polinomio generador.

Definición 1.6.

Un código lineal $C \subset \mathbb{F}_q^n$ es cíclico si

$$c_0c_1\cdots c_{n-2}c_{n-1} \in C \quad \textit{implica} \quad c_{n-1}c_0c_1\cdots c_{n-2} \in C.$$

Es decir, según esta definición, un código lineal C es cíclico si es cerrado bajo el desplazamiento cíclico de las letras que componen cada palabra.

Ejemplo 1.9.

Sean $C_3 = \{(0,0,0), (1,1,0), (1,0,1), (0,1,1)\}$ y $C_4 = \{(0,0,0), (1,0,1), (0,1,0), (1,1,1)\}$ $[3,4]$-códigos lineales sobre $\mathbb{F}_2$. El código C_3 es cíclico, pero C_4 no lo es pues la palabra $(1,1,0) \notin C_4$ y esta se obtiene de la palabra código $(1,0,1)$ luego de un desplazamiento

cíclico.

Ahora se presenta una forma de caracterizar un código cíclico, que permite ver la estructura algebraica que estos poseen. Si C es un código lineal sobre $\mathbb{F}_q^n$, entonces a cada palabra código $c = (c_0, c_1, \ldots, c_{n-1}) \in C$, se le asocia un polinomio en el anillo cociente $R_n = \frac{\mathbb{F}_q[x]}{\langle x^n - 1 \rangle}$ de las clases residuales de $\mathbb{F}_q[x]$ módulo $x^n - 1$ de la siguiente manera,

$$\begin{aligned} \phi : \ \mathbb{F}_q^n &\longrightarrow R_n \\ c &\longrightarrow \phi(c) = [c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}]. \end{aligned}$$

donde ϕ es un isomorfismo de espacios vectoriales de C sobre el subespacio $\phi(C)$ de R_n. Además, se puede probar que C es un código cíclico, si y solo si, $\phi(C)$ es un ideal del anillo cociente R_n. En consecuencia estudiar códigos cíclicos es equivalente a estudiar ideales del anillo R_n, con la particularidad de que R_n es un dominio de ideales principales.

Luego para todo ideal no cero I de R_n existe un único polinomio mónico de grado minimal que lo genera. El único polinomio mónico se denomina el *polinomio generador* de I. Para un código cíclico C sobre $\mathbb{F}_q^n$, el polinomio generador de $\phi(C)$ es llamado también el polinomio generador de C. Esto evidencia una ventaja ya que los códigos cíclicos, pueden ser determinados completamente mediante un polinomio de grado menor que n, donde n es la longitud del código.

Sin embargo, un código cíclico C puede ser generado por otros polinomios distintos al polinomio generador. Por lo tanto, se adopta la notación $C = \langle\langle p(x) \rangle\rangle$ que significa que C es el ideal generado por $p(x)$ y que $p(x)$ es el polinomio generador de C.

Existen dos maneras de codificar mensajes utilizando códigos cíclicos. Una de ellas es dado $C = \langle\langle g(x) \rangle\rangle$ un $[n, n-r]$-código cíclico, con $grad(g(x)) = r$. Se puede codificar el mensaje q-ario $r(x)$ de longitud $n - r$, de la siguiente forma $c(x) = r(x)g(x)$.

Sea $g(x)$ el polinomio generador de un $[n, n-r]$-código cíclico C. Se puede probar que $g(x)$ divide a $x^n - 1$ en R_n, luego $x^n - 1 = g(x)h(x)$ donde $h(x) \in \mathbb{F}_q[x]$ es un polinomio de grado $n - r$. El polinomio $h(x)$ se denomina el *polinomio de control* de C.

Sea $h(x) = \Sigma_{i=0}^{k} a_i x^i$ un polinomio de grado k sobre $\mathbb{F}_q$ se define el *polinomio recíproco* $h_R(x)$ como

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

Si L es un código cíclico, entonces el código dual $L^{\perp}$ es también un código cíclico el cual es generado por $h_0^{-1}h_R(x)$ donde h_0 es el término independiente del polinomio de control de L y $h_R(x)$ su polinomio recíproco.

1.2. Forma de Smith de una matriz polinomial

A continuación se introducen dos matrices cuadradas invertibles con entradas en $\mathbb{F}_q[z]$, que se denominan elementales con ellas se realizan operaciones elementales, las cuales son el producto del efecto de multiplicar a izquierda o a derecha por estas matrices.

Sean $p(z) \in \mathbb{F}_q[z]$ y e_{ij} una matriz cuadrada con 1 en la posición (i, j) y 0 en las demás posiciones. Luego, $P_{ij} = \mathbf{1} - e_{ii} - e_{jj} + e_{ij} + e_{ji}$ es la matriz elemental invertible por medio de la cual se realiza la permutación de filas i y j o las columnas i y j dependiendo de si multiplica a la izquierda o derecha de una matriz. Es claro que $P_{ij}^{-1} = P_{ij}$.

Ahora se considera la matriz $T_{ij}(p(z)) = \mathbf{1} + p(z)e_{ij}$, la cual es invertible dado que

$$T_{ij}(p(z))T_{ij}(-p(z)) = (\mathbf{1} + p(z)e_{ij})(\mathbf{1} - p(z)e_{ij}) = \mathbf{1}.$$

Esta matriz lleva a cabo la adición de los elementos de la i-ésima fila (i-ésima columna) multiplicados por $p(z)$ a los elementos de la j-ésima fila (j-ésima columna) si la multiplicación se realiza a izquierda (derecha) de una matriz.

A continuación, se presenta la descomposición en factores invariantes de una matriz que se basa en el siguiente resultado algebraico, ver [13, p. 181].

Teorema 1.1. *(Forma de Smith)*
Sea $G(z)$ una matriz polinomial con entradas en $\mathbb{F}_q[z]$ de tamaño $k \times n$, con $k \leq n$ de rango r. Entonces $G(z)$ puede escribirse de la siguiente manera

$$G(z) = A(z)\Gamma(z)B(z), \tag{1.1}$$

donde $A(z)$ y $B(z)$ son matrices polinomiales invertibles con entradas en $\mathbb{F}_q[z]$ de tamaño $k \times k$ y $n \times n$ respectivamente y $\Gamma(z)$ es una matriz diagonal de tamaño $k \times n$ cuyos elementos no nulos $\gamma_i(z) \in \mathbb{F}_q[z]$ con $1 \leq i \leq r$ satisfacen que $\gamma_i(z) \mid \gamma_{i+1}(z)$, dicha matriz $\Gamma(z)$ se denomina la forma de Smith de $G(z)$ y la expresión en la ecuación (1.1) se llama

descomposición en factores invariantes de $G(z)$.

Además, si $\Delta_i(z) \in \mathbb{F}_q[z]$ *es el máximo común divisor de los* $i \times i$ *menores de* $G(z)$, *entonces* $\gamma_i(z) = \Delta_i(z)/\Delta_{i-1}(z)$ *para* $1 \leq i \leq r$ *y* $\Delta_0(z) = 1$ *por convención.*

Demostración.

Si $G(z) = \mathbf{0}$, no hay nada que probar. Así se asume que $G(z) \neq \mathbf{0}$.

Sea $g_{ij}(z)$ un elemento no nulo de $G(z)$ con $gr(g_{ij}(z))$ minimal. Por medio transformaciones elementales por fila y columna, se lleva dicho elemento a la posición $(1,1)$. Supóngase ahora que esta allí. Sean $\alpha_{ij}(z)$ con $1 \leq i \leq k$ y $1 \leq j \leq n$ los elementos de esta nueva matriz. Para $j > 1$, se divide un elemento de la primera fila $\alpha_{1j}(z)$ por $\alpha_{11}(z)$, entonces se tiene que

$$\alpha_{1j}(z) = \alpha_{11}(z)\beta_j(z) + \beta_{1j}(z), \text{ donde } gr(\beta_{1j}(z)) < gr(\alpha_{11}(z)).$$

Ahora, se suma la primera columna multiplicada por $-\beta_j(z)$ a la j-ésima columna. Esta operación elemental reemplaza $\alpha_{1j}(z)$ por $\beta_{1j}(z)$. Si $\beta_{1j}(z) \neq 0$ se obtiene una matriz equivalente para la cual se ha reducido el grado mínimo de una entrada no nula. Se repite el procedimiento original a esta nueva matriz. Del mismo modo, si

$$\alpha_{j1}(z) = \alpha_{11}(z)\beta_j(z) + \beta_{j1}(z), \text{ donde } gr(\beta_{j1}(z)) < gr(\alpha_{11}(z)),$$

entonces por medio de operaciones elementales por fila del mismo tipo del caso anterior, permiten obtener una matriz para la cual se reduce el grado mínimo de las entradas no cero de la primera columna. Dado que el grado es siempre un entero no negativo en un número finito de aplicaciones de este proceso se obtiene una matriz equivalente $G'(z) = (\beta_{ij}(z))$ en la cual $\beta_{11}(z)$ tiene grado minimal y $\beta_{11} \mid \beta_{1j}$ y $\beta_{11} \mid \beta_{i1}$ para todo $i > 1$ y todo $j > 1$. Entonces por medio de transformaciones elementales en las filas y columnas se obtiene una matriz equivalente de la forma

$$\begin{pmatrix} \beta_{11}(z) & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & G''(z) & \\ 0 & & & \end{pmatrix}.$$

Sea $\gamma_1(z) = \beta_{11}(z)$. A continuación se demuestra que $\gamma_1(z)$ divide a todos los elementos en $G''(z) = (\delta_{ij}(z))$. Suponga que $\gamma_1(z) \nmid \delta_{ij}(z)$, entonces se suma la j-ésima columna a la

primera obteniendo una nueva columna. Repetir el primer proceso proporciona un elemento en la posición $(1,1)$ con menor grado que $\beta_{11}(z)$, lo cual es una contradicción.

Luego, repitiendo el procedimiento inicial a la sub-matriz $G''(z)$ se obtiene una matriz equivalente de la forma

$$\begin{pmatrix} \beta_{11}(z) & 0 & 0 & \cdots & 0 \\ 0 & \delta_{11}(z) & 0 & \cdots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & & G'''(z) & \\ 0 & 0 & & & \end{pmatrix}.$$

Así $\gamma_2(z) = \delta_{11}(z)$, si se procede como antes en un número finito de pasos se obtiene la matriz diagonal $\Gamma(z)$ cuyos elementos no nulos satisfacen que $\gamma_i(z) \mid \gamma_{i+1}(z)$.

Ahora, se prueba que Δ_i no se afecta por operaciones elementales de fila o columna en $G(z)$. Sea $A(z) = (a_{ij}(z))$ de tamaño $b \times b$ con entradas en $\mathbb{F}_q[z]$ cualquier producto de operaciones elementales de fila. La entrada (i,j) de $A(z)G(z)$ es $\sum_k a_{ik}(z)g_{kj}(z)$. Esto prueba que las filas de $A(z)G(z)$ son combinación lineal de de las filas de $G(z)$. Por lo tanto, los $i \times i$ menores de $A(z)G(z)$ son combinación lineal de los $i \times i$ menores de $G(z)$. Así, el máximo común divisor de todos los $i \times i$ menores de $G(z)$ es un divisor del máximo común divisor de todos los $i \times i$ menores de $A(z)G(z)$. Dado que el determinante de $A(z)$ es 1 esta es invertible, $A^{-1}(z)$ existe y es una matriz polinomial de tamaño $b \times b$. Repitiendo el argumento anterior para la matriz $A^{-1}(z)(A(z)G(z))$ se garantiza que el máximo común divisor de todos los $i \times i$ menores de $A(z)G(z)$ son un divisor de todos los $i \times i$ menores de $A^{-1}(z)(A(z)G(z)) = G(z)$. Por lo tanto, el máximo común divisor de todos los menores de $G(z)$ y de $A(z)G(z)$ son el mismo. Así $\Delta_i(z)$ es invariante bajo operaciones elementales de fila. Con un argumento análogo, se puede probar que $\Delta_i(z)$ es invariante bajo operaciones elementales por columna.

La forma de $\Gamma(z)$ muestra que

$$\begin{aligned} \Delta_1(z) &= \gamma_1(z), \\ \Delta_2(z) &= \gamma_1(z)\gamma_2(z), \\ &\vdots \\ \Delta_r(z) &= \gamma_1(z)\gamma_2(z)\ldots\gamma_r(z). \end{aligned}$$

Luego, si $\Delta_0(z) = 1$ se tiene que

$$\gamma_i(z) = \frac{\Delta_i(z)}{\Delta_{i-1}(z)}, \text{ para } 1 \leq i \leq r.$$

La unicidad de los $\Delta_i(z)$ implica la unicidad de los $\gamma_i(z)$ para todo i. □

Ejemplo 1.10.

Determinar la forma de Smith de la matriz polinomial binaria

$$G(z) = \begin{pmatrix} 1+z & z & 1 \\ z^2 & 1 & 1+z+z^2 \end{pmatrix}$$

y su descomposición en factores invariantes $G(z) = A(z)\Gamma(z)B(z)$.

Para determinar la forma de Smith de la matriz $G(z)$ se procede como en la demostración del Teorema 1.1, así primero se debe ubicar en la posición $(1,1)$ un elemento de grado mínimal; para ello se permutan las columnas 1 y 3 así

$$G(z)P_{13} = \begin{pmatrix} 1 & z & 1+z \\ 1+z+z^2 & 1 & z^2 \end{pmatrix}.$$

Luego, para hacer 0 los elemenetos en las posiciones $(1,2)$ y $(1,3)$ se realizan operaciones del tipo T_{ij} como sigue

$$(G(z)P_{13})T_{12}(z)T_{13}(1+z) = \begin{pmatrix} 1 & 0 & 0 \\ 1+z+z^2 & 1+z+z^2+z^3 & 1+z^2+z^3 \end{pmatrix}.$$

A continuación se hace 0 la posición $(2,1)$, para ello basta con multiplicar por la izquierda la matriz anterior por $T_{21}(1+z+z^2)$ y se tiene

$$T_{21}(1+z+z^2)(G(z)P_{13}T_{12}(z)T_{13}(1+z)) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1+z+z^2+z^3 & 1+z^2+z^3 \end{pmatrix} = G'(z).$$

Siguiendo la técnica de la prueba se divide $1+z^2+z^3$ por $1+z+z^2+z^3$ obteniendo

$$1+z^2+z^3 = (1+z+z^2+z^3)1+z,$$

de lo cual se sigue que se debe sumar la columna 2 a la 3 de la matriz $G'(z)$ para obtener

$$G'(z)T_{23}(1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1+z+z^2+z^3 & z \end{pmatrix}.$$

Ahora, en la matriz obtenida en el paso anterior se permutan las columnas 2 y 3 y se tiene

$$(G'(z)T_{23}(1))P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & z & 1+z+z^2+z^3 \end{pmatrix}.$$

Ahora se debe dividir $1+z+z^2+z^3$ por z esto es

$$1+z+z^2+z^3 = z(1+z+z^2)+1.$$

Luego, se multiplica la columna 2 por $1+z+z^2$ y se le suma a la columna 3, así

$$(G'(z)T_{23}(1)P_{23})T_{23}(1+z+z^2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & z & 1 \end{pmatrix}.$$

Nuevamente es necesario intercambiar las columnas 2 y 3 como sigue

$$(G'(z)T_{23}(1)P_{23}T_{23}(1+z+z^2))P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & z \end{pmatrix}.$$

Luego se divide z por 1 obteniendo que $z = 1(z)+0$, así finalmente a la columna 3 se le suma z veces la columna 2, esto es

$$(G'(z)T_{23}(1)P_{23}T_{23}(1+z+z^2)P_{23})T_{23}(z) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \Gamma(z).$$

El proceso termina dado que la matriz $G(z)$ tiene rango 2 y de su forma de Smith se tiene que los factores invariantes son todos iguales a 1.

Finalmente para obtener la descomposición de la matriz $G(z)$, basta con rastrear hacia atrás las matrices elementales usadas en el proceso para obtener $\Gamma(z)$ y multiplicar esta por sus inversas como sigue

$$G(z) = T_{21}(1+z+z^2)\Gamma(z)(T_{23}(z)P_{23}T_{23}(1+z+z^2)P_{23}T_{23}(1)T_{13}(1+z)T_{12}(z)P_{13})$$

De lo cual se concluye que

$$A(z) = T_{21}(1+z+z^2) = \begin{pmatrix} 1 & 0 \\ 1+z+z^2 & 1 \end{pmatrix}$$

y

$$\begin{aligned} B(z) &= T_{23}(z)P_{23}T_{23}(1+z+z^2)P_{23}T_{23}(1)T_{13}(1+z)T_{12}(z)P_{13} \\ &= \begin{pmatrix} 1+z & z & 1 \\ 1+z^2+z^3 & 1+z+z^2+z^3 & 0 \\ z+z^2 & 1+z+z^2 & 0 \end{pmatrix}. \end{aligned}$$

Por lo tanto, se tiene que la siguiente descomposición de la matriz $G(z)$

$$G(z) = \begin{pmatrix} 1 & 0 \\ 1+z+z^2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1+z & z & 1 \\ 1+z^2+z^3 & 1+z+z^2+z^3 & 0 \\ z+z^2 & 1+z+z^2 & 0 \end{pmatrix}.$$

1.3. Módulos

En esta sección se presentan algunos conceptos y propiedades básicas de la teoría de módulos, puesto que se usan a lo largo del documento y son necesarios principalmente en el Capítulo 3.

Definición 1.7. *(R-módulo)*

Sea R un anillo. Un grupo abeliano M escrito aditivamente, se denomina un R-módulo izquierdo si para cada $a \in R$ y cada $m \in M$ se tiene que un producto $am \in M$ tal que:

i) $(a+b)m = am + bm$

ii) $a(m_1 + m_2) = am_1 + am_2$

iii) $a(bm) = (ab)m$

iv) $1m = m$ *para todo* $a, b \in R$ *y* $m, m_1, m_2 \in M$.

Análogamente, se define R-módulo a derecha, considerando la multiplicación de elementos de M por elementos de R al lado derecho. Ahora se presenta la definicón de R-submódulo.

Definición 1.8. *(R-submódulo)*
Sea M un módulo sobre un anillo R. Un subconjunto no vacio $N \subset M$ se llama un R-submódulo de M si se cumplen las siguientes condiciones:

i) Para todo $x, y \in N$ se tiene que $x + y \in N$.

ii) Para todo $r \in R$ y todo $n \in N$ se cumple que $rn \in N$.

A continuación, se presentan las definiciones básicas para extender la noción de base de un espacio vectorial, para el caso de un módulo definido sobre un anillo.

Definición 1.9. *(Conjunto generador)*
Un conjunto $S = \{s_i\}_{i \in I}$ de elementos de un R-módulo M es un conjunto de generadores de M si $M = RS$; es decir, si todo elemento de M puede escribirse como una combinación lineal finita de elementos de S con coeficientes en R.

Definición 1.10. *(Independencia lineal)*
Un conjunto $S = \{s_i\}_{i \in I}$ de elementos de un R-módulo M es linealmente independiente o simplemente R-libre si, para cualquier combinación lineal finita de elemento en S con coeficientes en R

$$r_{i_1} s_{i_1} + r_{i_2} s_{i_2} + \cdots + r_{i_t} s_{i_t} = 0,$$

implica que $r_{i_1} = r_{i_2} = \cdots = r_{i_t} = 0$

Definición 1.11. *(Base o R-base)*
Un conjunto $S = \{s_i\}_{i \in I}$ de elementos de un R-módulo M es una base de Msobre R o una R-base si, S es un conjunto de generadores linealmente independientes.

Definición 1.12. *(R-módulo libre)*
Un R-módulo M es libre si este tiene una base.

El siguiente teorema es fundamental en la prueba de la Proposición 3.1.

Teorema 1.2.

Sean R un dominio de ideales principales, F un R-módulo libre de dimensión finita y M un submódulo de F. Entonces M es libre y su dimensión es menor o igual que la dimensión de F.

Demostración.

Sean $\{x_1, x_2, \cdots, x_n\}$ una base de F y M_r la intersección de M con $\langle x_1, \cdots, x_r\rangle$ el módulo generado por $x_1, \cdots, x_r$.

Entonces $M_1 = M \cap \langle x_1\rangle$ es un módulo de $\langle x_1\rangle$ y es del tipo $\langle a_1 x_1\rangle$ para algún $a_1 \in R$. Así, M_1 es cero o libre de dimensión 1.

Ahora como hipótesis de inducción se asume que M_r es libre de dimensión menor o igual que r.

Sea $\mathscr{A}$ el conjunto que consiste de todos los elementos $a \in R$ tales que existe un elemento $x \in M$ que puede escribirse en la forma

$$x = b_1 x_1 + \cdots + b_r x_r + a x_{r+1} \text{ con } b_i \in R.$$

Luego $\mathscr{A}$ es un ideal de R y es principal, generado por un elemento a_{r+1}.

Si $a_{r+1} = 0$, entonces $M_{r+1} = M_r$ y termina el paso inductivo.

Si $a_{r+1} \neq 0$, sea $w \in M_{r+1}$ tal que el coeficiente de w con respecto a x_{r+1} es a_{r+1}, esto es

$$w = d_1 x_1 + d_2 x_2 + \cdots + d_r x_r + a_{r+1} x_{r+1}.$$

Sea $x \in M_{r+1}$ entonces

$$x = b_1 x_1 + b_2 x_2 + \cdots + b_r x_r + b_{r+1} x_{r+1} \text{ con } b_i \in R,$$

pero $M_{r+1} = M \cap \langle x_1, \cdots x_{r+1}\rangle$ luego $x \in M$ y $b_{r+1} \in \mathscr{A}$; es decir, existe $c \in R$ tal que $b_{r+1} = c a_{r+1}$, de ahí que

$$x = b_1 x_1 + b_2 x_2 + \cdots + b_r x_r + c a_{r+1} x_{r+1};$$

y se tiene

$$x - cw = c_1 x_1 + c_2 x_2 + \cdots + c_r x_r + 0 x_{r+1} \text{ con } c_i = b_i - c d_i \in R.$$

Así, $x - cw \in M_r$. Luego todo $x \in M_{r+1}$ se puede escribir como

$$x = c_1x_1 + c_2x_2 + \cdots + c_rx_r + cw \text{ para algún } c \in R,$$

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

Ahora, suponga que $M_r \cap \langle w \rangle \neq \langle 0 \rangle$. Sea $z \in M_r \cap \langle w \rangle$ entonces

$$z = b_1x_1 + b_2x_2 + \cdots + b_rx_r \quad \text{y} \quad z = g(d_1x_1 + \cdots + d_rx_r + a_{r+1}x_{r+1}) \quad \text{con} \quad g \neq 0 \in R.$$

Luego $0 = c_1x_1 + c_2x_2 + \cdots + c_rx_r + ga_{r+1}x_{r+1}$ con $c_i = b_i - gd_i$, lo cual contradice que $\{x_1, x_2, \cdots, x_{r+1}\}$ es un conjunto linealmente independiente ya que $ga_{r+1} \neq 0$. Por lo tanto $M_r \cap \langle w \rangle = \langle 0 \rangle$.

Así, M_{r+1} se puede escribir como una suma directa de M_r y $\langle w \rangle$, lo cual garantiza que M_{r+1} es libre de dimensión menor o igual que $r + 1$. □

Definición 1.13. *(Suma directa)*
Sean R un anillo, M un R-módulo izquierdo y $N_1, N_2, \ldots, N_k$ R-submódulos izquierdos de M. Entonces M es la suma directa de los submódulos M_i que se denota $M = \oplus_{i=1}^{k} N_i$, si las siguientes condiciones se cumplen:

i) $M = N_1 + N_2 + \cdots + N_k$.

ii) Para todo $1 \leq i \leq k$, $N_i \cap (\sum_{i \neq j} N_j) = \{0\}$

Definición 1.14. *(Sumando directo)*
Un submódulo N de un R-módulo M es un sumando directo de M si existe otro módulo N' tal que $M = N \oplus N'$.

Los Lemas 1.1 y 1.2 sobre sumandos directos que se presentan a continuación, se requieren para garantizar la veracidad de la Proposición 3.2.

Lema 1.1. *Sean R un anillo, M un R-módulo izquierdo y $N_1, N_2, \ldots, N_k$ R-submódulos izquierdos de M. Entonces las siguientes afirmaciones son equivalentes.*

i) $M = N_1 \oplus N_2 \oplus \cdots \oplus N_k$.

ii) Todo elemento $m \in M$ puede escribirse de forma única como

$$m = n_1 + n_2 + \cdots + n_k \text{ con } n_i \in N_i, 1 \leq i \leq k$$

Demostración.

i) $\Rightarrow$ *ii)* Suponga que M es suma directa de los R-submódulos izquierdos $N_1, N_2, \ldots, N_k$; luego para cada $m \in M$ existen $n_1, n_2, \ldots, n_k$ con $n_i \in N_i$ para $1 \leq i \leq k$ tales que $m = n_1 + n_2 + \cdots + n_k$. Ahora, suponga que esta representación no es única, es decir $m = n_1' + n_2' + \cdots + n_k'$ con $n_i' \in N_i$ para $1 \leq i \leq k$ de ahí que

$$n_1 + n_2 + \cdots + n_k = n_1' + n_2' + \cdots + n_k'.$$

Luego,

$$n_i - n_i' = n_1 - n_1' + n_2 - n_2' + \cdots + n_{i-1} - n_{i-1}' + n_{i+1} - n_{i+1}' + \cdots + n_k - n_k'$$

con $n_i - n_i' \in N_i$ y $n_1 - n_1' + n_2 - n_2' + \cdots + n_{i-1} - n_{i-1}' + n_{i+1} - n_{i+1}' + \cdots + n_k - n_k' \in \sum_{i \neq j} N_j$, y de la Definición 1.13 se tiene que $N_i \cap (\sum_{i \neq j} N_j) = \{0\}$, así $n_i - n_i' = 0$. Por lo tanto $n_i = n_i'$ para todo $1 \leq i \leq k$.

ii) $\Rightarrow$ *i)* Suponga que para cada $m \in M$ se puede escribir de forma única como

$$m = n_1 + n_2 + \cdots + n_k \text{ con } n_i \in N_i, \text{ para } 1 \leq i \leq k.$$

De la hipótesis se tiene que $M = N_1 + N_2 + \cdots + N_k$. Se prueba a continuación que $N_i \cap (\sum_{i \neq j} N_j) = \{0\}$. Para ello suponga que existe $x \in N_i \cap (\sum_{i \neq j} N_j)$, entonces $x = x + 0$, con $x \in N_i$ y $0 \in \sum_{i \neq j} N_j$; también $x = 0 + x$ con $0 \in N_i$ y $x \in \sum_{i \neq j} N_j$ así dado que la representación de $x \in M$ es única se tiene que $x = 0$. □

Lema 1.2.

Sea R un anillo. Sean M un R-módulo izquierdo y N un R-submódulo de M. Entonces N es un sumando directo de M si y solo si existe una función $\pi : M \to N$ que es R-lineal y la identidad en N.

Demostración.

Si N es un sumando directo de M, entonces existe un R-submódulo N' tal que $M = N \oplus N'$.

Luego cualquier $m \in M$ puede expresarse de forma única como $m = n+n'$ para algún $n \in N$ y $n' \in N'$. Ahora sea $\pi : M \to N$ definida por $\pi(m) = n$ donde n es el elemento único de N en la representación de m. Ahora sean $m_1, m_2 \in M$ y $r \in R$, entonces $\pi(m_1 + m_2) = \pi((n_1+n_2)+(n_1'+n_2')) = n_1+n_2 = \pi(m_1)+\pi(m_2)$ y $\pi(rm_1) = \pi(rn_1+rn_1') = rn_1 = r\pi(m_1)$. Así π es R-lineal y es la función identidad en N.

Recíprocamente, sean $m \in M$, $n = \pi(m)$ y $n' = m-n$ luego $\pi(n') = \pi(m-n) = 0$. Sea N' el kernel de π, entonces $n' \in N'$. Ahora para $r \in R$, $\pi(rn') = \pi(rm-rn) = \pi(rm)-\pi(rn) = r(\pi(m) - \pi(n)) = 0$ lo cual implica que $rn' \in N'$. Así N' es un R-modulo izquierdo de M. Dado que $M = N + N'$ y $N \cap N' = 0$ se tiene que N es un sumando directo de M. □

La Definición 1.15 y el Teorema 1.3 son necesarios en la Sección 3.2 para la demostración de la Proposición 3.6.

Definición 1.15. *(A Invariante)*
Un subespacion $R' \subset R$ se denomina invariante con respecto al operador A o A invariante si $AR' \subset R'$; es decir, si $x \in R'$ implica $Ax \in R'$. En otras palabras, el operador A lleva un vector del subespacio R' en otro vector del mismo subespacio.

Teorema 1.3. *(Teorema de Descomposición de un Espacio en Subespacios Invariantes)*
Sean E un k-espacio vectorial, A un operador de E. Si el polinomio minimal $\psi(\lambda)$ se descompone como producto de dos polinomios mónicos coprimos $\psi_1(\lambda)$ y $\psi_2(\lambda)$; es decir, $\psi(\lambda) = \psi_1(\lambda)\psi_2(\lambda)$. Entonces, E se descompone en suma directa de subespacios invariantes por A

$$E = \ker \psi_1(A) \oplus \ker \psi_2(A),$$

con $\ker \psi_i(A) = \{x \in E : \psi_i(A)x = 0\}$ para $i = 1, 2$. Además, el polinomio minimal de $\ker \psi_i(A)$ es $\psi_i(\lambda)$ para $i = 1, 2$.

Demostración. Ver [7, p.179] □

1.4. Anillos de polinomios torcidos

En esta sección se presentan los requisitos para que un anillo S sea un *anillo de polinomios torcido*[1] en una indeterminada x sobre algún anillo R. Si se quiere que los coeficientes de los polinomios estén a la izquierda, entonces los elementos de S deben tener la forma $r_0 + r_1x + \cdots + r_nx^n$ para $n \in \mathbb{Z}^+$ y $r_i \in R$. Para que los coeficientes y grados estén bien definidos, se requiere que dos polinomios sean iguales solo si sus coeficientes coinciden. Esto significa que hasta ahora se pide que S sea un R-módulo izquierdo con base $\{1, x, x^2, x^3, \dots\}$.

Además, un requisito básico que se pide es que la multiplicación de polinomios debe respetar grados en la medida de lo posible; en particular el grado de un producto de polinomios no debe ser mayor que la suma de los grados de los factores. Para satisfacer este requisito, es suficiente asumir que, para cada $r \in R$ el producto xr pueda escribirse con coeficientes a la izquierda y tenga grado a lo más 1. En otras palabras $xr = r'x + r''$ para algunos $r', r'' \in R$. Dado que se supone que los coeficientes de un polinomio son únicos r' y r'' deben depender únicamente de r. Por lo tanto, deben existir funciones α , $\delta : R \to R$ tales que $xr = \alpha(r)x + \delta(r)$para todo $r \in R$. Ahora, por el hecho de estar trabajando en un anillo cabe preguntar ¿qué propiedades se imponen a α y δ?

La ley distributiva requiere que $x(r+s) = xr+xs$ para todo $r, s \in R$, de donde se deduce que α y δ deben ser funciones aditivas. Dado que $x1 = x$, se tiene que $\alpha(1) = 1$ y $\delta(1) = 0$. La ley asociativa requiere que $x(rs) = (xr)s$ para todo $r, s \in R$, por lo cual

$$\alpha(rs)x + \delta(rs) = (\alpha(r)x + \delta(r))s = \alpha(r)(\alpha(s)x + \delta(s)) + \delta(r)s.$$

Por lo tanto, se tiene $\alpha(rs) = \alpha(r)\alpha(s)$ y $\delta(rs) = \alpha(r)\delta(s) + \delta(r)s$ para $r, s \in R$. En particular, α debe ser un endomorfismo en el anillo R. La función δ es similar a una derivación excepto por la aparición de α y esta se denomina una *derivación torcida*[2] con respecto a α o simplemente una *α-derivación.*

Definición 1.16. *(α-derivación)*

Sea α un endomorfismo en el anillo R. Una α-derivación en R es cualquier función aditiva $\delta : R \to R$ tal que $\delta(rs) = \alpha(r)\delta(s) + \delta(r)s$ para todo par $r, s \in R$ y $\delta(1) = 0$.

[1] En inglés, Skew polynomials rings.
[2] En ingles, skew derivation.

Observación 1.2. Estrictamente hablando, se definió una α-derivación izquierda, aunque en este trabajo no será necesario el concepto de α-derivación derecha, esta se puede definir como una función aditiva $\delta : R \to R$ tal que $\delta(rs) = \delta(r)\alpha(s) + r\delta(s)$ para todo par $r, s \in R$ y $\delta(1) = 0$.

En el caso de una derivación ordinaria, no es necesario incluir la condición $\delta(1) = 0$, pero en la definición de una α-derivación sí, ya que se deduce de la regla del *producto torcido*[3]. Si α es la función identidad en R, entonces la α-derivación δ es precisamente una derivación ordinaria.

Lo anterior lleva a construir un anillo de polinomios torcido en el cual la multiplicación es afectada por un endomorfismo α de anillos y una α-derivación δ.

Definición 1.17. *(Anillo de polinomios torcido sobre R)*
Sea R un anillo, α un endomorfismo del anillo R y δ una α-derivación en R. Se denota $S = R[x; \alpha, \delta]$ al conjunto tal que

a) S es un anillo, que contiene a R como subanillo.

b) x es un elemento de S.

c) S es un R-modulo libre izquierdo con base $\{1, x, x^2, \dots\}$.

d) $xr = \alpha(r)x + \delta(r)$ para todo $r \in R$.

El anillo S se denomina un anillo de polinomios torcido sobre R.

Observación 1.3. Cabe aclarar que algunos autores prefieren que sus anillos de polinomios torcidos tengan coeficientes a la derecha. Para lograr esto se inicia con un anillo R, α un endomorfismo de R y δ una α-derivación derecha en R. El correspondiente anillo de polinomios torcido es un R-modulo libre derecho con base $\{1, x, x^2, \dots\}$, donde $xr = x\alpha(r) + \delta(r)$ para todo $r \in R$.

Hasta el momento, la definición dada es solamente una lista de condiciones, pero no se ha probado que un anillo de la forma $R[x; \alpha, \delta]$ existe, ni que este sea único salvo isomorfismos.

[3]En inglés, skew product.

Por otra parte, una vez descrita una regla de multiplicación explicita para $R[x;\alpha,\delta]$, el proceso para demostrar que este es un anillo no es el habitual de verificar los axiomas de anillo; sino que se prueba la existencia de $R[x;\alpha,\delta]$ sin definir explícitamente la multiplicación. Dicha prueba puede encontrarse en [10, Prop. 2.3]. En la Sección 3.2 del Capítulo 3 se trabaja con un tipo particular de estos anillos esto es el caso cuando la α-derivación $\delta = 0$, y en esta caso se tiene que $R[x;\alpha,0] = R[x;\alpha]$.

A partir del item *d)* de la Definición 1.17 se puede determinar una formula general para expresar $x^i r$, para cualquier $i \in \mathbb{N}$ y $r \in R$, como un polinomio con coeficientes a izquierda. Por ejemplo,

$$\begin{aligned}
x^3r = \;& x^2(xr) = x^2(\alpha(r)x + \delta(r)) = x^2\alpha(r)x + x^2\delta(r) = x(x\alpha(r))x + x(x\delta(r)) \\
=\;& x(\alpha(\alpha(r))x + \delta(\alpha(r)))x + x(\alpha(\delta(r)) + \delta(\delta(r))) \\
=\;& (x\alpha^2(r))x^2 + (x\delta(\alpha(r)))x + (x\alpha(\delta(r)))x + x\delta^2(r) \\
=\;& [\alpha(\alpha^2(r))x + \delta(\alpha^2(r))]x^2 + [\alpha(\delta(\alpha(r)))x + \delta(\delta(\alpha(r)))]x \\
& +[\alpha(\alpha(\delta(r)))x + \delta(\alpha(\delta(r)))]x + [\alpha(\delta^2(r))x + \delta(\delta^2(r))] \\
=\;& \alpha^3(r)x^3 + [\delta\alpha^2(r) + \alpha\delta\alpha(r) + \alpha^2\delta(r)]x^2 + [\delta^2\alpha(r) + \delta\alpha\delta(r) + \alpha\delta^2(r)]x + \delta^3(r).
\end{aligned}$$

Capítulo 2

Códigos Convolucionales

Este capítulo se divide en cinco secciones, en la primera se presenta la definición de un código convolucional desde el punto de vista de las series de Laurent y en el mismo sentido la definición de codificador convolucional, las diferentes formas de representar tanto las k-secuencias de información como las n-secuencias código; la segunda sección está dedicada a el codificador físico de un código convolucional; en la tercera sección se muestran las representaciones analíticas y algunas gráficas relacionadas a un codificador convolucional y la operación de codificación convolucional. En la cuarta sección se expone la decodificación mediante el algoritmo de Viterbi y finalmente en la última sección se introduce el concepto de código dual de un código convolucional. La principal referencia de este capítulo es el libro de A. Dholakia, ver [3].

2.1. Definición y notación

La definición de código convolucional puede presentarse de dos formas diferentes. Una consiste en definir primero el codificador y a partir de este establecer el código y la otra es hacerlo en el sentido contrario; sin embargo en los dos casos se define el mismo objeto. En este documento se asumirá la posición de definir primero el código y luego el codificador.

Además, estas definiciones pueden darse en diferentes niveles de generalidad, lo que depende de la estructura escogida tanto para el alfabeto como para los elementos del código. Los elementos de estos conjuntos se pueden tomar de las siguientes estructuras algebraicas

infinitas, siendo $\mathbb{F}$ un campo arbitrario: $\mathbb{F}[z]$ el anillo de polinomios en la indeterminada z con coeficientes en el campo $\mathbb{F}$, ver [12, p. 149]; $\mathbb{F}(z) = \{p/q : p, q \in \mathbb{F}[z] \text{ y } q \neq 0\}$ el campo de funciones racionales en z sobre $\mathbb{F}$, ver [1, p. 342]; $\mathbb{F}[[z]] = \{\Sigma_{i=0}^{\infty} f_i z^i : f_i \in \mathbb{F}\}$ el anillo de series de potencias formales en z sobre $\mathbb{F}$, ver [12, p. 154] o $\mathbb{F}((z)) = \{\Sigma_{i=-\infty}^{\infty} f_i z^i : f_i \in \mathbb{F}\}$ el campo de series formales de Laurent, ver [18, p. 873].

Para el caso del campo de las series formales de Laurent, cabe recordar la forma en que se construyen los inversos multiplicativos. Dado que los elementos en el conjunto de las series formales de potencias $\mathbb{F}[[z]]$, son de la forma $f(z) = \sum_{n=0}^{\infty} a_n z^n$ con coeficientes en el campo $\mathbb{F}$, este conjunto es un dominio entero con la adición y la multiplicación usuales. Una serie formal de potencias $f(z)$ admite un inverso multiplicativo $f^{-1}(z)$ en $\mathbb{F}[[z]]$ si y solo si $a_0 \neq 0$, ver [18, Teorema 1]. Ahora, si $f(z)$ es cualquier elemento no nulo de $\mathbb{F}((z))$ entonces no todos sus coeficientes son cero y si e es el menor índice tal que $a_e \neq 0$, $f(z)$ se puede reescribir como $f(z) = z^e h(z)$ para algún $h(z) \in \mathbb{F}[[z]]$ el cual admite un inverso multiplicativo. Luego, $f^{-1}(z) = z^{-e} h^{-1}(z)$ es el inverso multiplicativo de $f(z)$ el cual es un elemento de $\mathbb{F}((z))$. Por lo tanto, $\mathbb{F}((z))$ junto con la adición y multiplicación usuales forman un campo.

Antes de dar las definiciones de código y codificador convolucional se presentarán algunas notaciones previas que facilitarán su comprensión.

Notación 2.1.

Dado $k \in \mathbb{Z}^+$, se entiende por una k-úpla de información en el tiempo t a un elemento de $\mathbb{F}^k$, el cual usualmente se denota con $\boldsymbol{u}_t = (u_t^{(1)}, u_t^{(2)}, \ldots, u_t^{(k)})$. Igualmente, a partir de las k-úplas de información $\boldsymbol{u}_t$ se construye una k-secuencia de información, como un elemento de $\mathbb{F}^k((z))$ y que se representa como

$$\boldsymbol{u}(z) = \sum_{t=-\infty}^{\infty} \boldsymbol{u}_t z^t, \tag{2.1}$$

donde para algún $r \in \mathbb{Z}$ se tiene que $\boldsymbol{u}_t = \boldsymbol{0}$ para $t < r$.

Similarmente, para $n \in \mathbb{Z}^+$, con $\boldsymbol{v}_t = (v_t^{(1)}, v_t^{(2)}, \ldots, v_t^{(n)})$ se representa una n-úpla código en el tiempo t, luego la n-secuencia código definida a partir de las n-úplas código $\boldsymbol{v}_t$, puede representarse como

$$\boldsymbol{v}(z) = \sum_{t=-\infty}^{\infty} \boldsymbol{v}_t z^t, \tag{2.2}$$

donde para algún $s \in \mathbb{Z}$ se tiene que $\boldsymbol{v}_t = \mathbf{0}$ para $t < s$, el cual es un elemento de $\mathbb{F}^n((z))$.

La notación anterior extiende aquella utilizada en la teoría de códigos de bloque para referirse a una palabra mensaje y una palabra código, ver [11, p. 1].

De lo anterior se sigue que,

$$\begin{aligned}\boldsymbol{u}(z) &= \textstyle\sum_{t=-\infty}^{\infty} \boldsymbol{u}_t z^t = \boldsymbol{u}_r z^r + \boldsymbol{u}_{r+1} z^{r+1} + \cdots \\ &= (u_r^{(1)}, u_r^{(2)}, \ldots, u_r^{(k)}) z^r + (u_{r+1}^{(1)}, u_{r+1}^{(2)}, \ldots, u_{r+1}^{(k)}) z^{r+1} + \cdots \\ &= (u_r^{(1)} z^r, u_r^{(2)} z^r, \ldots, u_r^{(k)} z^r) + (u_{r+1}^{(1)} z^{r+1}, u_{r+1}^{(2)} z^{r+1}, \ldots, u_{r+1}^{(k)} z^{r+1}) + \cdots \\ &= (u_r^{(1)} z^r + u_{r+1}^{(1)} z^{r+1} + \cdots, u_r^{(2)} z^r + u_{r+1}^{(2)} z^{r+1} + \cdots, \ldots, u_r^{(k)} z^r + u_{r+1}^{(k)} z^{r+1} + \cdots).\end{aligned}$$

Ahora, si se hace

$$\boldsymbol{u}^{(i)} = (u_r^{(i)}, u_{r+1}^{(i)}, \ldots) \text{ y } \boldsymbol{u}^{(i)}(z) = u_r^{(i)} z^r + u_{r+1}^{(i)} z^{r+1} + \cdots, \tag{2.3}$$

para $i = 1, 2, \ldots, k$, entonces la k-secuencia de información $\boldsymbol{u}(z)$ puede representarse como el vector $\boldsymbol{u}(z) = (\boldsymbol{u}^{(1)}(z), \boldsymbol{u}^{(2)}(z), \ldots, \boldsymbol{u}^{(k)}(z))$ en $\mathbb{F}((z))^k$ y de esta manera se tiene la relación que permite representar $\boldsymbol{u}(z)$ por $\boldsymbol{u} = (\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(k)})$.

Similarmente, la n-secuencia código, puede simbolizarse por medio del vector $\boldsymbol{v}(z) = (\boldsymbol{v}^{(1)}(z), \boldsymbol{v}^{(2)}(z), \ldots, \boldsymbol{v}^{(n)}(z))$ en $\mathbb{F}((z))^n$ y puede asociarse a $\boldsymbol{v} = (\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(n)})$.

Aunque se puede dar una definición general de códigos convolucionales sobre cualquier campo en este trabajo se usarán campos finitos $\mathbb{F}_q$, donde q es una potencia de un primo.

Definición 2.1.

Un (n, k)-código convolucional C es un $\mathbb{F}_q((z))$-subespacio del espacio vectorial $\mathbb{F}_q((z))^n$ de dimensión k, donde $\mathbb{F}_q$ es un campo finito. La velocidad del código es $R = k/n$.

Ahora, un codificador es una transformación que acepta una k-úpla de información $\boldsymbol{u}_t$ y produce una n-úpla código $\boldsymbol{v}_t$ como salida en el instante t. A diferencia de lo que sucede en los códigos de bloque, esta transformación tiene memoria, es decir la n-úpla código $\boldsymbol{v}_t$ no depende solo de la k-úpla de información $\boldsymbol{u}_t$ sino también de algunas k-úplas de información $\boldsymbol{u}_l$ para $l < t$. Cuando se utilizan las m k-úplas de información anteriores se dice que la memoria es m y se dirá que es un (n, k, m)-código convolucional.

A continuación se procede a dar la definición de un codificador para un código convolucional dado.

Definición 2.2.

Un (n, k, m)-codificador convolucional para el código convolucional C es una matriz $G(z)$ de tamaño $k \times n$, con entradas en el subconjunto $\mathbb{F}_q[z]$ de $\mathbb{F}_q((z))$, cuyas filas generan al (n, k, m)-código convolucional C.

En consecuencia, un (n, k, m)-código convolucional C que tenga como codificador convolucional la matriz $G(z)$, es la imagen de la función lineal $\mathbb{F}_q((z))^k \to \mathbb{F}_q((z))^n$, definida por $\boldsymbol{u}(z)G(z)$.

Observación 2.1. En las definiciones anteriores, los elementos de un código convolucional en realidad son series infinitas. Este es el enfoque tradicional que se puede encontrar en la literatura sobre códigos convolucionales, ver [3, 14, 17]; sin embargo en este trabajo se considerarán elementos con soporte finito tanto para las k-secuencias de información como para las n-secuencias código.

De esta manera, las expresiones dadas en (2.1), (2.2) y (2.3) se pueden presentar como sigue; $\boldsymbol{u}(z) = \sum_{t=0}^{N-1} \boldsymbol{u}_t z^t$ para las k-secuencias de información; $\boldsymbol{v}(z) = \sum_{t=0}^{M-1} \boldsymbol{v}_t z^t$ para las n-secuencias código y $\boldsymbol{u}^{(i)} = (u_0^{(i)}, u_1^{(i)}, \ldots, u_{N-1}^{(i)})$ y $\boldsymbol{u}^{(i)}(z) = u_0^{(i)} + u_1^{(i)} z + \cdots + u_{N-1}^{(i)} z^{N-1}$ para las componentes de $\boldsymbol{u}$ y $\boldsymbol{u}(z)$.

En consecuencia, en lugar de trabajar con elementos del $\mathbb{F}_q((z))$-espacio vectorial $\mathbb{F}_q((z))^n$ el trabajo se desarrollará en el $\mathbb{F}_q[z]$-módulo $\mathbb{F}_q[z]^n$. En este contexto los códigos convolucionales son $\mathbb{F}_q[z]$-submódulos de $\mathbb{F}_q[z]^n$ como se verá en la Sección 3.1 del Capítulo 3; bajo esta definición alternativa note que tanto las k-secuencias de información como las n-secuencias código son finitas a diferencia del contexto de las series de Laurent en el cual estas son infinitas.

En las siguientes secciones para facilitar la presentación en los ejemplos se consideran secuencias finitas.

2.2. Codificador físico de un código convolucional

El esquema de codificación de un codificador convolucional se puede implementar usando registros de desplazamiento lineal o tambien conocidos como circuitos secuenciales lineales[1],

[1] En inglés, Linear Shift-Registers o Linear Sequential Circuits (LSC).

el cual consiste de una red con un número finito de terminales de entrada y de salida. Las señales aplicadas a las terminales de entrada son tomadas del campo finito $\mathbb{F}_q$ y se aplican simultáneamente a todas las terminales de entrada en instantes discretos de tiempo. La red en el registro de desplazamiento lineal consta de una interconexión de un número finito de componentes principales.

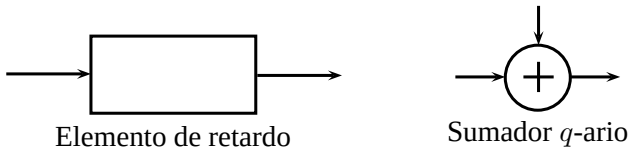


Figura 2.1: *Elemento de retardo y sumador q-ario.*

Las componentes principales de un registro de desplazamiento lineal son por una parte *sumadores* que sirven para implementar suma módulo q y por otra parte *elementos de retardo* también llamados *elementos de memoria* usados para retrasar o almacenar una señal de entrada por unidad de tiempo. En la figura 2.1 se muestra un elemento retardo con una entrada y una salida y un sumador q-ario con dos entradas y una salida.

En general, un elemento de retardo almacena un elemento de entrada para cada instante de tiempo, después se sustituye por otro elemento de entrada o se mueve al siguiente elemento de retardo o es descartado. El sumador toma todas las señales de entrada y de retardo que están relacionados con él, los suma módulo q y produce la salida en cada instante de tiempo. Finalmente, las secuencias de salida son intercaladas por un serializador para formar una secuencia de salida única.

A continuación se considera el siguiente codificador físico

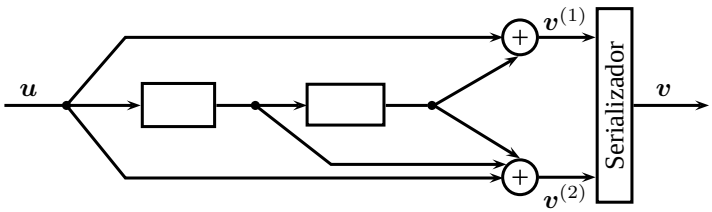


Figura 2.2: *Codificador Convolucional de velocidad* $R = 1/2$.

Como se puede observar el codificador consiste de una terminal de entrada, dos elementos de retardo, dos sumadores y tiene dos salidas. Luego, la velocidad del codificador anterior

es $R = 1/2$. Esto es, la 1-secuencia de información $\boldsymbol{u} = (u_0, u_1, \ldots, u_{N-1})$ se desplaza desde la izquierda un elemento a la vez y dos secuencias códigos $\boldsymbol{v}^{(1)}$ y $\boldsymbol{v}^{(2)}$ son las salidas de los sumadores módulo q. Por tanto, en la unidad de tiempo t el elemento de información actual es u_t, los elementos de retardo almacenados en las cajas son u_{t-1} y u_{t-2}, respectivamente; los cuales para el tiempo inicial $t = 0$ se asume que contienen ceros; y las dos salidas son $v_t^{(1)}$ y $v_t^{(2)}$. La primera salida es la suma módulo q de u_t y u_{t-2}, mientras que la segunda salida es la suma módulo q de u_t, u_{t-1} y u_{t-2}; es decir, la operación de codificación puede ser descrita por medio de las siguientes ecuaciones

$$v_t^{(1)} = u_t + u_{t-2}(\text{mód}q) \quad \text{y} \quad v_t^{(2)} = u_t + u_{t-1} + u_{t-2}(\text{mód}q).$$

Después de determinar las salidas respectivas en cada tiempo, los elementos de retardo se desplazan de la siguiente forma, u_t toma el lugar de u_{t-1}, u_{t-1} sustituye a u_{t-2} y u_{t-2} es descartado puesto que no se vuelve a utilizar. Este proceso se detiene cuando los elementos de retardo vuelven a su estado inicial; esto es todos iguales a cero, por lo cual para este caso después del tiempo $t = N - 1$ se requiere adicionar dos entradas de ceros que corresponde al número de elementos de retardo.

Finalmente, el serializador intercala las componentes de $\boldsymbol{v}^{(1)}$ y $\boldsymbol{v}^{(2)}$ para formar la salida

$$\boldsymbol{v} = (v_0^{(1)}, v_0^{(2)}, v_1^{(1)}, v_1^{(2)}, v_2^{(1)}, v_2^{(2)}, \ldots, v_{N+1}^{(1)}, v_{N+1}^{(2)}).$$

Además, se puede ver que la diferencia básica entre los códigos de bloque y los códigos convolucionales, es que en los primeros el elemento codificado en el tiempo t depende solo de la secuencia de información en este tiempo; mientras que en los códigos convolucionales la secuencia código en el tiempo t no depende exclusivamente de la secuencia de información en ese instante, sino también de m secuencias de información anteriores; es decir, un código convolucional requiere memoria, pero un código de bloque no.

En general, la velocidad del codificador es $R = k/n$ donde en cada instante de tiempo, k es el número de elementos de entrada y n es el número de elementos de salida. Si se utiliza un (n, k, m)-codificador convolucional, para codificar una k-secuencia de información de longitud N, en el proceso son codificados $k \cdot N$ datos para obtener una n-secuencia código de longitud $M = N + m$, la cual tendrá $n \cdot M$ datos de salida, donde los últimos $n \cdot m$

datos son generados por los m bloques de ceros que se le adjuntan a las k-secuencias de información para terminar el proceso.

A continuación se presenta un ejemplo en el cual las entradas se toman sobre el campo $\mathbb{F}_2$ que es generalmente el campo más usado en la literatura.

Ejemplo 2.1.

Considere el codificador convolucional de la figura 2.2 y la 1-secuencia de información $\boldsymbol{u} = 1101001$. Suponga que los elementos de retardo iniciales son ceros.

Los cálculos para cada tiempo se presentan en tabla 2.1.

Tiempo	Entrada	Elementos de Retardo		Salidas	
t	u_t	u_{t-1}	u_{t-2}	$v_t^{(1)} = u_t + u_{t-2}(\text{mód}2)$	$v_t^{(2)} = u_t + u_{t-1} + u_{t-2}(\text{mód}2)$
0	1	0	0	$1+0=1$	$1+0+0=1$
1	1	1	0	$1+0=1$	$1+1+0=0$
2	0	1	1	$0+1=1$	$0+1+1=0$
3	1	0	1	$1+1=0$	$1+0+1=0$
4	0	1	0	$0+0=0$	$0+1+0=1$
5	0	0	1	$0+1=1$	$0+0+1=1$
6	1	0	0	$1+0=1$	$1+0+0=1$
7	0	1	0	$0+0=0$	$0+1+0=1$
8	0	0	1	$0+1=1$	$0+0+1=1$

Tabla 2.1: *Codificación de 1101001 usando el codificador de la figura 2.2.*

El proceso se detiene en el tiempo $t = 8$ dado que para $t = 9$ y en adelante, tanto la entrada como los elementos de retardo son cero y por tanto las salidas serán siempre cero.
Así, se obtiene $\boldsymbol{v}^{(1)} = 111001101$ y $\boldsymbol{v}^{(2)} = 100011111$ y la 2-secuencia código es

$$\boldsymbol{v} = 11\ \ 10\ \ 10\ \ 00\ \ 01\ \ 11\ \ 11\ \ 01\ \ 11.$$

2.3. Representación de codificadores convolucionales

A continuación se expone la representación de un codificador convolucional y la operación convolucional mediante; secuencias generadoras discretas, matriz generadora discreta semi-infinita, generadores polinomiales, matriz generadora polinomial y dos representaciones gráficas que son: el diagrama de estado y el diagrama enrejado; y la forma de pasar de una representación a otra.

2.3.1. Representación analítica

En esta parte se exponen las diferentes representaciones analíticas, relacionadas a un codificador convolucional y la operación de codificación convolucional.

1. Secuencias generadoras discretas

Para el caso de un $(n, 1, m)$-código convolucional, un codificador convolucional de velocidad $R = 1/n$ es representado por las 1-secuencias generadoras $\boldsymbol{g}^{(j)} = (g_0^{(j)}, g_1^{(j)}, \ldots, g_m^{(j)})$ para $j = 1, 2, \ldots, n$. La operación de codificación convolucional $*$, la cual denota la convolución discreta de la 1-secuencia de información $\boldsymbol{u} = (u_0, u_1, u_2, \ldots, u_{N-1})$ con cada 1-secuencia generadora, se expresa mediante

$$\boldsymbol{v}^{(j)} = \boldsymbol{u} * \boldsymbol{g}^{(j)}, \text{ para } j = 1, 2, \ldots, n, \tag{2.4}$$

donde cada componente de $\boldsymbol{v}^{(j)}$ se calcula por medio de la siguiente regla $v_t^{(j)} = \sum_{l=0}^{m} u_{t-l} g_l^{(j)}$ con $u_{t-l} = 0$ si $t < l$.

Así las 1-secuencias código $\boldsymbol{v}^{(j)} = (v_0^{(j)}, v_1^{(j)}, v_2^{(j)}, \ldots, v_{(N+m)-1}^{(j)})$ con $1 \leq j \leq n$, dependen de la 1-secuencia de información $\boldsymbol{u}$.

La n-secuencia generadora compuesta que se construye al serializar las 1-secuencias generadoras $\boldsymbol{g}^{(j)}$, con $1 \leq j \leq n$ es

$$\boldsymbol{g} = (g_0^{(1)}, g_0^{(2)}, \ldots, g_0^{(n)}, g_1^{(1)}, g_1^{(2)}, \ldots, g_1^{(n)}, \ldots, g_m^{(1)}, g_m^{(2)}, \ldots, g_m^{(n)}).$$

El proceso anteriormente descrito es el fundamento del algoritmo `secgen` A.3. ver, p. 99, implementado en el programa computacional `SAGE`.

Observación 2.2. Dado que las secuencias generadoras especifican las entradas a cada sumador binario en el codificador físico, para obtenerlas a partir del codificador convolucional físico binario se procede como sigue. Para $l = 0, 1, \ldots, m$ y $j = 1, 2, \ldots, n$.

i) $g_l^{(j)} = 0$, si u_{t-l} no está conectado con la salida j.

ii) $g_l^{(j)} = 1$, si u_{t-l} está conectado con la salida j.

En lo que sigue se da un ejemplo de estos conceptos.

Ejemplo 2.2.

Considere el codificador convolucional del $(2,1,2)$-código convolucional de la figura 2.2. Entonces las 1-secuencias generadoras para este codificador son

$$\boldsymbol{g}^{(1)} = (1,0,1) \quad \text{y} \quad \boldsymbol{g}^{(2)} = (1,1,1).$$

La 2-secuencia generadora compuesta es

$$\boldsymbol{g} = (11\ \ 01\ \ 11).$$

Si se considera la 1-secuencia de información $\boldsymbol{u} = 11$, y se calcula la 2-secuencia código, se debe calcular $\boldsymbol{v}^{(1)} = \boldsymbol{u} * \boldsymbol{g}^{(1)}$ y $\boldsymbol{v}^{(2)} = \boldsymbol{u} * \boldsymbol{g}^{(2)}$. Para ello se calculan las componentes de cada vector como sigue

$$\begin{aligned}
v_0^{(1)} &= \textstyle\sum_{i=0}^{2} u_{0-i} g_i^{(1)} = u_0 g_0^{(1)} + u_{-1} g_1^{(1)} + u_{-2} g_2^{(1)} = 1(1) + 0(0) + 0(1) = 1.\\
v_1^{(1)} &= \textstyle\sum_{i=0}^{2} u_{1-i} g_i^{(1)} = u_1 g_0^{(1)} + u_0 g_1^{(1)} + u_{-1} g_2^{(1)} = 1(1) + 1(0) + 0(1) = 1.\\
v_2^{(1)} &= \textstyle\sum_{i=0}^{2} u_{2-i} g_i^{(1)} = u_2 g_0^{(1)} + u_1 g_1^{(1)} + u_0 g_2^{(1)} = 0(1) + 1(0) + 1(1) = 1.\\
v_3^{(1)} &= \textstyle\sum_{i=0}^{2} u_{3-i} g_i^{(1)} = u_3 g_0^{(1)} + u_2 g_1^{(1)} + u_1 g_2^{(1)} = 0(1) + 0(0) + 1(1) = 1.\\
v_0^{(2)} &= \textstyle\sum_{i=0}^{2} u_{0-i} g_i^{(2)} = u_0 g_0^{(2)} + u_{-1} g_1^{(2)} + u_{-2} g_2^{(2)} = 1(1) + 0(1) + 0(1) = 1.\\
v_1^{(2)} &= \textstyle\sum_{i=0}^{2} u_{1-i} g_i^{(2)} = u_1 g_0^{(2)} + u_0 g_1^{(2)} + u_{-1} g_2^{(2)} = 1(1) + 1(1) + 0(1) = 0.\\
v_2^{(2)} &= \textstyle\sum_{i=0}^{2} u_{2-i} g_i^{(2)} = u_2 g_0^{(2)} + u_1 g_1^{(2)} + u_1 g_2^{(2)} = 0(1) + 1(1) + 1(1) = 0.\\
v_3^{(2)} &= \textstyle\sum_{i=0}^{2} u_{3-i} g_i^{(2)} = u_3 g_0^{(2)} + u_2 g_1^{(2)} + u_0 g_2^{(2)} = 0(1) + 0(1) + 1(1) = 1.
\end{aligned}$$

De lo anterior se tiene que $\boldsymbol{v}^{(1)} = (1111)$ y $\boldsymbol{v}^{(2)} = (1001)$. Por tanto, la 2-secuencia código resultante es $\boldsymbol{v} = (11\ \ 10\ \ 10\ \ 11)$.

Usando el algoritmo `secgen` A.3 implementado en `SAGE`, las entradas son: la 1-secuencia de información $\boldsymbol{u} = 11$ que se representa por `u=vector(GF(2),[1,1])` y las 1-secuencias generadoras $\boldsymbol{g}^{(1)} = (1,0,1)$ y $\boldsymbol{g}^{(2)} = (1,1,1)$ que se ingresan en una lista `g=[g1,g2]` donde `g1=vector(GF(2),[1,0,1])` y `g2=vector(GF(2),[1,1,1])` y finalmente después de llamar el algoritmo se obtiene por salida

```
secgen(u,g)
    (1, 1, 1, 0, 1, 0, 1, 1)
```

este vector es la 2-secuencia código $\boldsymbol{v}$ que se calculó previamente.

Ejemplo 2.3.

Considere el $(5,1,3)$-codificador convolucional con secuencias generadoras $\boldsymbol{g}^{(1)} = 1000$, $\boldsymbol{g}^{(2)} = 1010$, $\boldsymbol{g}^{(3)} = 0100$, $\boldsymbol{g}^{(4)} = 0111$ y $\boldsymbol{g}^{(5)} = 1111$. Para codificar la 1-secuencia de información $\boldsymbol{u} = 101101011$ usando el algoritmo A.3 se obtiene que las entradas en `SAGE` son

```
u=vector(GF(2),[1,0,1,1,0,1,0,1,1]);
g1=vector(GF(2),[1,0,0,0]); g2=vector(GF(2),[1,0,1,0]);
g3=vector(GF(2),[0,1,0,0]); g4=vector(GF(2),[0,1,1,1]);
g5=vector(GF(2),[1,1,1,1]);
g=[g1,g2,g3,g4,g5]
```

Al llamar el algoritmo `secgen` se tiene que la 5-secuencia código es

```
secgen(u,g)
(1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1,
 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1)
```

Ahora, un codificador convolucional, de un (n,k,m)-código convolucional con $k > 1$ el cual tiene velocidad de codificación $R = k/n$ es representado por nk 1-secuencias generadoras $\boldsymbol{g}_i^{(j)} = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, \ldots, g_{i,m}^{(j)})$ con $i = 1, 2, \ldots, k$ y $j = 1, 2, \ldots, n$. La operación de codificación convolucional, es la convolución discreta de la secuencia de información con las secuencias generadoras y se expresa por

$$\boldsymbol{v}^{(j)} = \sum_{i=1}^{k} \boldsymbol{u}^{(i)} * \boldsymbol{g}_i^{(j)} \text{ para } j = 1, 2, \ldots, n\,, \tag{2.5}$$

donde cada componente de $\boldsymbol{v}^{(j)}$ se calcula por medio de la regla

$$v_t^{(j)} = \sum_{i=1}^{k} \sum_{l=0}^{m} u_{t-l}^{(i)} g_{i,l}^{(j)} \quad \text{con } u_{t-l}^{(i)} = 0 \text{ si } t < l. \tag{2.6}$$

Así, cada una de las n 1-secuencias código $\boldsymbol{v}^{(j)} = (v_0^{(j)}, v_1^{(j)}, \ldots, v_{(N+m)-1}^{(j)})$ para $1 \leq j \leq n$, depende de cada una de las 1-secuencias de información $\boldsymbol{u}^{(i)} = (u_0^{(i)}, u_1^{(i)}, \ldots, u_{N-1}^{(i)})$ con $1 \leq i \leq k$ y dicha dependencia esta determinada por las 1-secuencias generadoras.

La n-secuencia generadora compuesta para la i-ésima entrada es

$$\boldsymbol{g}_i = (g_{i,0}^{(1)}, g_{i,0}^{(2)}, \ldots, g_{i,0}^{(n)}, \ldots, g_{i,m}^{(1)}, g_{i,m}^{(2)}, \ldots, g_{i,m}^{(n)}), \tag{2.7}$$

para $1 \leq i \leq k$, la cual se obtienen al serializar las n 1-secuencias generadoras $\boldsymbol{g}_i^{(j)}$ para un i fijo.

Observación 2.3. Si $k > 1$, las secuencias generadoras se obtienen del codificador físico binario como sigue. Para $l = 0, 1, \ldots, m$, $i = 1, 2, \ldots, k$ y $j = 1, 2, \ldots, n$.

i) $g_{i,l}^{(j)} = 0$, si $u_{t-l}^{(i)}$ no está conectado con la salida j.

ii) $g_{i,l}^{(j)} = 1$, si $u_{t-l}^{(i)}$ está conectado con la salida j.

Ejemplo 2.4.

En la figura 2.3 se muestra un codificador convolucional de un $(3, 2, 2)$-código convolucional binario. El codificador tiene dos registros de desplazamiento cada uno con una 1-secuencia de información $\boldsymbol{u}^{(i)}$ con $i = 1, 2$ como entradas para cada instante de tiempo.

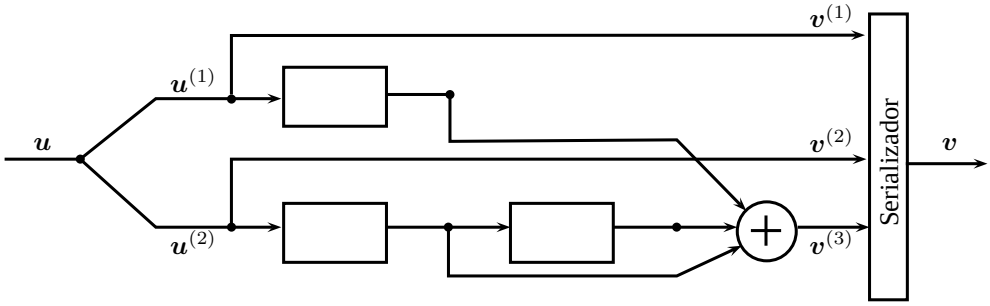


Figura 2.3: *Codificador Convolucional de velocidad* $R = 2/3$.

El codificador tiene seis 1-secuencias generadoras, que son

$$\begin{aligned} \boldsymbol{g}_1^{(1)} &= (1, 0, 0), & \boldsymbol{g}_2^{(1)} &= (0, 0, 0), \\ \boldsymbol{g}_1^{(2)} &= (0, 0, 0), & \boldsymbol{g}_2^{(2)} &= (1, 0, 0), \\ \boldsymbol{g}_1^{(3)} &= (0, 1, 0), & \boldsymbol{g}_2^{(3)} &= (0, 1, 1). \end{aligned}$$

Las dos 3-secuencias generadoras compuestas para este codificador son

$$\boldsymbol{g}_1 = (100 \quad 001 \quad 000) \qquad y \qquad \boldsymbol{g}_2 = (010 \quad 001 \quad 001).$$

Ahora, considere la 2-secuencia de información $\boldsymbol{u} = 11 \quad 01$ y suponga que los elementos de retardo iniciales son cero.

Las 1-secuencias de información son $\boldsymbol{u}^{(1)} = (1,0)$ y $\boldsymbol{u}^{(2)} = (1,1)$. De ahí se obtiene que las salidas son

$$\begin{aligned}
\boldsymbol{v}^{(1)} &= \boldsymbol{u}^{(1)} * \boldsymbol{g}_1^{(1)} + \boldsymbol{u}^{(2)} * \boldsymbol{g}_2^{(1)} = (1,0,0,0) + (0,0,0,0) = (1,0,0,0),\\
\boldsymbol{v}^{(2)} &= \boldsymbol{u}^{(1)} * \boldsymbol{g}_1^{(2)} + \boldsymbol{u}^{(2)} * \boldsymbol{g}_2^{(2)} = (0,0,0,0) + (1,1,0,0) = (1,1,0,0),\\
\boldsymbol{v}^{(3)} &= \boldsymbol{u}^{(1)} * \boldsymbol{g}_1^{(3)} + \boldsymbol{u}^{(2)} * \boldsymbol{g}_2^{(3)} = (0,1,0,0) + (0,1,0,1) = (0,0,0,1).
\end{aligned}$$

Por tanto, la 3-secuencia código es $\boldsymbol{v} = (110 \quad 010 \quad 000 \quad 001)$.

Ahora, para usar el algoritmo `ksecgen` A.4, ver p. 99, implementado en `SAGE`, considere las siguientes entradas: la 2-secuencia de información $\boldsymbol{u} = (11 \quad 01)$ la cual se representa por una lista de vectores `u=[u1,u2]` donde `u1=vector(GF(2),[1,0])` y `u2=vector(GF(2),[1,1])` y las seis 1-secuencias generadoras usadas anteriormente en una lista de listas de vectores así, `g=[[g11,g12,g13],[g21,g22,g23]]` donde

`g11=vector(GF(2),[1,0,0])`, `g12=vector(GF(2),[0,0,0])`,

`g13=vector(GF(2),[0,1,0])`, `g21=vector(GF (2),[0,0,0])`,

`g22=vector(GF(2),[1,0,0])` y `g23=vector(GF(2),[0,1,1])`.

Luego, de llamar el algoritmo `ksecgen` se obtiene el vector

```
ksecgen(u,g)
(1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1)
```

el cual representa la 3-secuencia código $\boldsymbol{v}$ obtenida anteriormente.

Ejemplo 2.5.

Considere nuevamente el $(3,2,2)$-codificador de la figura 2.3, lo cual implica que se usarán las secuencias generadoras determinadas en el ejemplo 2.4, a continuación por medio del algoritmo `ksecgen` implementado en `SAGE`, se determina la 3-secuencia código de la 2-secuencia de información $\boldsymbol{u} = 10 \quad 11 \quad 01 \quad 10 \quad 10 \quad 01 \quad 11$. Luego las entradas en `SAGE` son

```
u1=vector(GF(2),[1,1,0,1,1,0,1]);
u2=vector(GF(2),[0,1,1,0,0,1,1]);
u=[u1,u2];
g11=vector(GF(2),[1,0,0]); g12=vector(GF(2),[0,0,0]);
```

```
g13=vector(GF(2),[0,1,0]); g21=vector(GF(2),[0,0,0]);
g22=vector(GF(2),[1,0,0]); g23=vector(GF(2),[0,1,1]);
g=[[g11,g12,g13],[g21,g22,g23]]
```

finalmente se tiene por salida el vector

```
ksecgen(u,g)
(1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1)
```

que representa la 3-secuencia código.

2. Matriz generadora discreta semi-infinita.

La ecuación (2.6) permite calcular cada componente de una n-secuencia código $\boldsymbol{v}^{(j)}$, a continuación se muestra la forma de representar la codificación de una k-secuencia de información por medio de un producto de matrices.

De la ecuación (2.6) se tiene

$$v_t^{(j)} = \sum_{i=1}^{k}\sum_{l=0}^{m} u_{t-l}^{(i)} g_{i,l}^{(j)} \quad \text{con } u_{t-l}^{(i)} = 0 \text{ si } t < l.$$

Luego,

$$\begin{aligned} v_t^{(j)} \quad &= u_t^{(1)} g_{1,0}^{(j)} + u_{t-1}^{(1)} g_{1,1}^{(j)} + u_{t-2}^{(1)} g_{1,2}^{(j)} + \cdots + u_{t-m}^{(1)} g_{1,m}^{(j)} + \\ &\quad u_t^{(2)} g_{2,0}^{(j)} + u_{t-1}^{(2)} g_{2,1}^{(j)} + u_{t-2}^{(2)} g_{2,2}^{(j)} + \cdots + u_{t-m}^{(2)} g_{2,m}^{(j)} + \cdots + \\ &\quad u_t^{(k)} g_{k,0}^{(j)} + u_{t-1}^{(k)} g_{k,1}^{(j)} + u_{t-2}^{(k)} g_{k,2}^{(j)} + \cdots + u_{t-m}^{(k)} g_{k,m}^{(j)}, \end{aligned}$$

con $u_{t-l}^{(i)} = 0$ si $t < l$.

Ahora, dado que para $0 \leq l \leq m$, se tiene que $\boldsymbol{u}_{t-l} = (u_{t-l}^{(1)}, u_{t-l}^{(2)}, \ldots, u_{t-l}^{(k)})$ con $\boldsymbol{u}_{t-l} = \boldsymbol{0}$ si $t < l$, denotando

$$G_l^{(j)} = \begin{pmatrix} g_{1,l}^{(j)} \\ g_{2,l}^{(j)} \\ \vdots \\ g_{k,l}^{(j)} \end{pmatrix},$$

se tiene que $v_t^{(j)} = \boldsymbol{u}_t G_0^{(j)} + \boldsymbol{u}_{t-1} G_1^{(j)} + \boldsymbol{u}_{t-2} G_2^{(j)} + \cdots + \boldsymbol{u}_{t-m} G_m^{(j)}$.

De lo anterior, es posible determinar la componente $\boldsymbol{v}_t$ de la n-secuencia código $\boldsymbol{v}$ por medio de una multiplicación de matrices. Para $0 \leq l \leq m$ sea

$$G_l = \begin{pmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & \vdots & & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{pmatrix}, \tag{2.8}$$

entonces un bloque codificado $\boldsymbol{v}_t$ está dado por

$$\boldsymbol{v}_t = \boldsymbol{u}_t G_0 + \boldsymbol{u}_{t-1} G_1 + \cdots + \boldsymbol{u}_{t-m} G_m, \tag{2.9}$$

donde $\boldsymbol{u}_{t-l} = \boldsymbol{0}$ para $t < l$. Note que las componentes de la matriz G_l son las entradas de las n-secuencias generadoras compuestas $\boldsymbol{g}_i$ de la ecuación (2.7) .

De esta manera la k-secuencia de información $\boldsymbol{u}$ se codifica por medio de de la ecuación

$$\boldsymbol{v} = \boldsymbol{u}G, \tag{2.10}$$

con G una matriz de bloques construida a partir de las matrices G_l de la siguiente manera

$$G = \begin{pmatrix} G_0 & G_1 & G_2 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_{m-1} & G_m & \\ & & G_0 & \cdots & G_{m-2} & G_{m-1} & G_m \\ & & & \ddots & & & \ddots \end{pmatrix}, \tag{2.11}$$

donde las entradas en blanco corresponden a la matriz nula de tamaño $k \times n$ y la dimensión de la matriz G depende de la longitud de la k-secuencia de información de entrada; en efecto, si $\boldsymbol{u}$ tiene longitud N la matriz generadora es de tamaño $N \times (N + m)$.

La matriz G se conoce como la matriz generadora discreta semi-infinita para el (n, k, m)-código convolucional C y las matrices G_l para $0 \leq l \leq m$ son las submatrices generadoras.

Observación 2.4. Dado que las 1-secuencias generadoras para un $(n, 1, m)$-codígo convolucional son de la forma $\boldsymbol{g}^{(j)} = (g_0^{(j)}, g_1^{(j)}, \ldots, g_m^{(j)})$ para $j = 1, 2, \ldots, n$, se tiene que para $l = 0, 1, \ldots, m$

$$G_l = \begin{pmatrix} g_l^{(1)} & g_l^{(2)} & \cdots & g_l^{(n)} \end{pmatrix},$$

y la matriz generadora discreta semi-infinita es

$$G = \begin{pmatrix} g_0^{(1)} g_0^{(2)} \cdots g_0^{(n)} & g_1^{(1)} g_1^{(2)} \cdots g_1^{(n)} & \cdots & g_m^{(1)} g_m^{(2)} \cdots g_m^{(n)} & & \\ & g_0^{(1)} g_0^{(2)} \cdots g_0^{(n)} & \cdots & g_{m-1}^{(1)} g_{m-1}^{(2)} \cdots g_{m-1}^{(n)} & g_m^{(1)} g_m^{(2)} \cdots g_m^{(n)} & \\ & & \ddots & & & \ddots \end{pmatrix}. \tag{2.12}$$

La codificación usando las componentes de la matriz generadora discreta semi-infinita se implementó en SAGE en un algoritmo útil para cualquier valor de k; dicho algoritmo se denomina `matgen` A.5, ver, p. 100.

Ejemplo 2.6.

La matriz generadora semi-infinita G para el $(2, 1, 2)$-codificador de la figura 2.2 es

$$G = \begin{pmatrix} 11 & 01 & 11 & & & \\ & 11 & 01 & 11 & & \\ & & 11 & 01 & 11 & \\ & & & \ddots & & \ddots \end{pmatrix}.$$

Note que la primera fila de la matriz generadora G es la 2-secuencia generadora compuesta, la segunda fila se obtiene por desplazamiento de la primera fila dos posiciones a la derecha y sucesivamente la fila siguiente es la anterior desplazada dos posiciones a la derecha.

Por otro lado, la 2-secuencia código correspondiente a la 1-secuencia de información $\boldsymbol{u} = (1 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1)$ de longitud 7 se obtiene realizando la siguiente multiplicación

$$\boldsymbol{u}G = (1 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1) \begin{pmatrix} 11 & 01 & 11 & & & & & & \\ & 11 & 01 & 11 & & & & & \\ & & 11 & 01 & 11 & & & & \\ & & & 11 & 01 & 11 & & & \\ & & & & 11 & 01 & 11 & & \\ & & & & & 11 & 01 & 11 & \\ & & & & & & 11 & 01 & 11 \end{pmatrix},$$

y su producto o salida es $\boldsymbol{v} = (11 \;\; 10 \;\; 10 \;\; 00 \;\; 01 \;\; 11 \;\; 11 \;\; 01 \;\; 11)$, la cual es una 2-secuencia código de longitud 9, es decir la salida consta de 18 datos.

Para usar el algoritmo `matgen` A.5 que tiene por entradas la 1-secuencia de información $\boldsymbol{u} = (1 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1)$ y las componentes de la matriz generadora semi-infinita, se debe tener en cuenta que $\boldsymbol{u}$ está representado por un vector con entradas en el campo finito $\mathbb{F}_2$ esto es `u=vector(GF(2),[1,1,0,1,0,0,1]);` y una lista de vectores con entradas en el mis-

mo campo que son las componentes de la matriz generadora G, así `G=[G0,G1,G2]` donde `G0=Matrix(GF(2),1,2,[1,1])`, `G1=Matrix(GF(2),1,2,[0,1])` y `G2=Matrix(GF(2),1,2,[1,1])`. En consecuencia, se obtiene una lista de vectores

```
matgen(u,G)
[(1, 1),(1, 0),(1, 0),(0, 0),(0, 1),(1, 1),(1, 1),(0, 1),(1, 1)]
```

que representa la 2-secuencia código obtenida anteriormente.

Note que la secuencia obtenida en el ejemplo anterior es la misma que se obtuvo con el codificador físico y con las secuencias generadoras en los ejemplos 2.1 y 2.2, respectivamente.

Ejemplo 2.7.

Se considera el $(5, 1, 3)$-codificador convolucional cuyas componentes de la matriz generadora son $G_0 = (1, 1, 0, 0, 1)$, $G_1 = (0, 0, 1, 1, 1)$, $G_2 = (0, 1, 0, 1, 1)$ y $G_3 = (0, 0, 0, 1, 1)$ para codificar la 1-secuencia de información $\boldsymbol{u} = (1, 0, 1, 1, 0, 1, 0, 1, 1)$ usando el algoritmo A.5 se tiene

```
G0=Matrix(GF(2),1,5,[1,1,0,0,1]); G1=Matrix(GF(2),1,5,[0,0,1,1,1]);
G2=Matrix(GF(2),1,5,[0,1,0,1,1]); G3=Matrix(GF(2),1,5,[0,0,0,1,1]);
G=[G0,G1,G2,G3];
u=vector(GF(2),[1,0,1,1,0,1,0,1,1]);
matgen(u,G)
[(1,1,0,0,1), (0,0,1,1,1), (1,0,0,1,0), (1,1,1,0,1),
 (0,1,1,0,0), (1,0,0,0,1), (0,0,1,0,0), (1,0,0,1,0),
 (1,1,1,0,1), (0,1,1,0,0), (0,1,0,0,0), (0,0,0,1,1)]
```

Ejemplo 2.8.

En este ejemplo se determina la matriz generadora semi-infinita G para el $(3, 2, 2)$-codificador de la figura 2.3. De las 3-secuencias generadoras compuestas obtenidas en el ejemplo 2.4 se obtiene que las componentes de la matriz generadora son

$$G_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad G_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad G_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Por lo tanto, la matriz generadora discreta es

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

Luego, se considera la 2-secuencia de información $\boldsymbol{u} = (11 \;\; 01)$, y se determina la 3-secuencia código usando la ecuación (2.9), dado que $\boldsymbol{u}$ es una 2-secuencia de longitud $N = 2$ entonces la 3-secuencia código tendrá longitud $M = N + m = 4$, de ahí que se debe calcular $\boldsymbol{v}_t$ para $t = 0, 1, 2, 3$, esto es

$$\boldsymbol{v}_0 = \boldsymbol{u}_0 G_0 + \boldsymbol{u}_{-1} G_1 + \boldsymbol{u}_{-2} G_2,$$

y como $0 < 1 < 2$, se tiene $\boldsymbol{u}_{-1} = \boldsymbol{u}_{-2} = (0 \;\; 0)$ así

$$\boldsymbol{v}_0 = \boldsymbol{u}_0 G_0 = (1 \;\; 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = (1 \;\; 1 \;\; 0).$$

Ahora $\boldsymbol{v}_1 = \boldsymbol{u}_1 G_0 + \boldsymbol{u}_0 G_1 + \boldsymbol{u}_{-1} G_2$, pero $1 < 2$ entonces $\boldsymbol{u}_{-1} = (0 \;\; 0)$, de ahí que

$$\boldsymbol{v}_1 = \boldsymbol{u}_1 G_0 + \boldsymbol{u}_0 G_1 = (0 \;\; 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + (1 \;\; 1) \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

esto es

$$\boldsymbol{v}_1 = (0 \;\; 1 \;\; 0) + (0 \;\; 0 \;\; 0) = (0 \;\; 1 \;\; 0).$$

Luego $\boldsymbol{v}_2 = \boldsymbol{u}_2 G_0 + \boldsymbol{u}_1 G_1 + \boldsymbol{u}_0 G_2$, donde $\boldsymbol{u}_2 = (0 \;\; 0)$, de ahí que

$$\boldsymbol{v}_2 = u_1 G_1 + u_0 G_2 = (0 \;\; 1) \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} + (1 \;\; 1) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

así

$$\boldsymbol{v}_2 = (0 \;\; 0 \;\; 1) + (0 \;\; 0 \;\; 1) = (0 \;\; 0 \;\; 0).$$

Finalmente, $\boldsymbol{v}_3 = \boldsymbol{u}_3 G_0 + \boldsymbol{u}_2 G_1 + \boldsymbol{u}_1 G_2$, con $\boldsymbol{u}_3 = \boldsymbol{u}_2 = (0 \;\; 0)$, luego

$$\boldsymbol{v}_3 = u_1 G_2 = (0 \;\; 1) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (0 \;\; 0 \;\; 1).$$

Por lo tanto, la 3-secuencia código es $v = (110 \;\; 010 \;\; 000 \;\; 001)$, la cual coincide con la obtenida en el ejemplo 2.4 usando las secuencias generadoras.

Ahora, se considera el algoritmo `matgen` A.5 el cual tiene por entradas un vector y una lista de matrices que representan la 2-secuencia de información y las componentes de la matriz generadora, respectivamente, de ahí que el vector de entrada $\boldsymbol{u}$ se representa por `u=vector(GF(2),[1,1,0,1])` y la lista por `G=[G0,G1,G2]` con

```
G0=Matrix(GF(2),2,3,[1,0,0,0,1,0]);
G1=Matrix(GF(2),2,3,[0,0,1,0,0,1]);
G2=Matrix(GF(2),2,3,[0,0,0,0,0,1])
```

De esta manera, al llamar el algoritmo se consigue

```
matgen(u,G)
 [(1, 1, 0), (0, 1, 0), (0, 0, 0), (0, 0, 1)]
```

que es una lista de vectores que representa la 3-secuencia código.

Ejemplo 2.9.

Usando el codificador del ejemplo 2.8 y con ayuda del algoritmo A.5 se codifica la 2-secuencia de información $\boldsymbol{u} = (10 \;\; 11 \;\; 01 \;\; 10 \;\; 10 \;\; 01 \;\; 11)$ como sigue

```
G0=Matrix(GF(2),2,3,[1,0,0,0,1,0]);
G1=Matrix(GF(2),2,3,[0,0,1,0,0,1]);
G2=Matrix(GF(2),2,3,[0,0,0,0,0,1]);
G=[G0,G1,G2];
u=vector(GF(2),[1,0,1,1,0,1,1,0,1,0,0,1,1,1]);
matgen(u,G)
[(1, 0, 0), (1, 1, 1), (0, 1, 0), (1, 0, 0), (1, 0, 0),
 (0, 1, 1), (1, 1, 1), (0, 0, 1), (0, 0, 1)]
```

en este caso se obtiene la misma 3-secuencia código encontrada en el ejemplo 2.5.

3. Generadores polinomiales

Como se hizo en la notación 2.1 con las k-secuencias de información y las n-secuencias código las secuencias generadoras $\boldsymbol{g}_i^{(j)}$ para $i = 1, 2, \ldots, k$ y $j = 1, \ldots, n$ se pueden representar como polinomios en la indeterminada z, como sigue.

Dado que $\boldsymbol{g}_i^{(j)} = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, \ldots, g_{i,m}^{(j)})$ entonces $\boldsymbol{g}_i^{(j)}(z) = g_{i,0}^{(j)} + g_{i,1}^{(j)} z + \cdots + g_{i,m}^{(j)} z^m$ para cada $i = 1, 2, \ldots, k$ y $j = 1, \ldots, n$.

De esta manera, la operación de convolución descrita en la ecuación (2.5) se puede representar polinomialmente como

$$\boldsymbol{v}^{(j)}(z) = \sum_{i=1}^{k} \boldsymbol{u}^{(i)}(z) \boldsymbol{g}_i^{(j)}(z), \tag{2.13}$$

se observa que cada coeficiente $v_t^{(j)}$ está dado por

$$v_t^{(j)}(z) = \sum_{i=1}^{k} \sum_{l=0}^{m} u_{t-l}^{(i)} g_{i,l}^{(j)}, \tag{2.14}$$

lo que verifica la ecuación (2.6).

Note que para el caso de un $(n, 1, m)$- codificador convolucional la ecuación (2.13) se transforma en

$$\boldsymbol{v}^{(j)}(z) = \boldsymbol{u}(z) \boldsymbol{g}^{(j)}(z). \tag{2.15}$$

Aquí, la multiplicación de polinomios se lleva a cabo sobre el anillo de polinomios $\mathbb{F}_q[z]$.

Ejemplo 2.10.

Los polinomios generadores del $(2, 1, 2)$-codificador de la figura 2.2 son

$$\boldsymbol{g}^{(1)}(z) = 1 + z^2 \qquad \boldsymbol{g}^{(2)}(z) = 1 + z + z^2.$$

Ahora usando el polinomio de información $\boldsymbol{u}(z) = 1 + z$, que corresponde a la 1-secuencia de información del ejemplo 2.2, se determina $\boldsymbol{v}^{(1)}(z)$ y $\boldsymbol{v}^{(2)}(z)$ calculando los siguientes productos de polinomios

$$\boldsymbol{v}^{(1)}(z) = \boldsymbol{u}(z) * \boldsymbol{g}^{(1)}(z) = z^3 + z^2 + z + 1 \qquad \boldsymbol{v}^{(2)}(z) = \boldsymbol{u}(z) * \boldsymbol{g}^{(2)}(z) = z^3 + 1$$

Por lo tanto la 2-secuencia polinomial es $\boldsymbol{v}(z) = (z^3 + z^2 + z + 1, z^3 + 1)$.

Si se usa del algoritmo `secgenpol` A.6, ver p. 102, implementado en `SAGE` se obtiene que las entradas son

```
P.<z> = GF(2)[]
g1=1+z^2; g2=1+z+z^2;
g=[g1,g2]
```

```
u=1+z
```

al llamar el algoritmo `secgenpol`, se obtiene por salida

```
secgenpol(u,g)
[z^3 + z^2 + z + 1, z^3 + 1]
```

Ejemplo 2.11.

Se considera el $(5, 1, 3)$-codificador convolucional con secuencias generadoras polinomiales $\boldsymbol{g}^{(1)}(z) = 1$, $\boldsymbol{g}^{(2)}(z) = 1 + z^2$, $\boldsymbol{g}^{(3)}(z) = z$, $\boldsymbol{g}^{(4)}(z) = z + z^2 + z^3$ y $\boldsymbol{g}^{(5)}(z) = 1 + z + z^2 + z^3$, para codificar la 1-secuencia de información $\boldsymbol{u}(z) = 1 + z + z^2 + z^3 + z^5 + z^7 + z^8$ usando el algoritmo A.6 se tiene

```
P.<z> = GF(2)[]
g1=1; g2=1+z^2; g3=z; g4=z+z^2+z^3; g5=1+z+z^2+z^3
g=[g1,g2,g3,g4,g5]
u=1+z^2+z^3+z^5+z^7+z^8
secgenpol(u,g)
 [z^8+z^7+z^5+z^3+z^2+1, z^10+z^9+z^8+z^4+z^3+1,
  z^9+z^8+z^6+z^4+z^3+z, z^11+z^7+z^2+z, z^11+z^8+z^5+z^3+z+1]
```

Es decir, la secuencia codigo obtenida es

$$\begin{aligned}\boldsymbol{v}(z) =&(z^8 + z^7 + z^5 + z^3 + z^2 + 1, z^{10} + z^9 + z^8 + z^4 + z^3 + 1, z^9 + z^8 + z^6 + z^4 + z^3 + z,\\ & z^{11} + z^7 + z^2 + z, z^{11} + z^8 + z^5 + z^3 + z + 1).\end{aligned}$$

Ejemplo 2.12.

Análogamente al ejemplo 2.10, los polinomios generadores del $(3, 2, 2)$-codificador de la figura 2.3 son

$$\begin{aligned} \boldsymbol{g}_1^{(1)} &= 1, & \boldsymbol{g}_2^{(1)} &= 0,\\ \boldsymbol{g}_1^{(2)} &= 0, & \boldsymbol{g}_2^{(2)} &= 1,\\ \boldsymbol{g}_1^{(3)} &= z, & \boldsymbol{g}_2^{(3)} &= z + z^2. \end{aligned}$$

La codificación para la 2-secuencia de información del ejemplo 2.4 usando la representación polinomial $\boldsymbol{u}(z) = (1, 1 + z)$, luego se debe calcular $\boldsymbol{v}^{(j)}$ para $j = 1, 2, 3$ usando la ecuación

(2.15), esto es

$$\begin{aligned}
\boldsymbol{v}^{(1)}(z) &= \textstyle\sum_{i=1}^{2} \boldsymbol{u}^{(i)}(z)\boldsymbol{g}_i^{(1)}(z) \\
&= \boldsymbol{u}^{(1)}(z)\boldsymbol{g}_1^{(1)}(z) + \boldsymbol{u}^{(2)}(z)\boldsymbol{g}_2^{(1)}(z) = 1(1) + (1+z)(0) = 1 \\
\boldsymbol{v}^{(2)}(z) &= \textstyle\sum_{i=1}^{2} \boldsymbol{u}^{(i)}(z)\boldsymbol{g}_i^{(2)}(z) \\
&= \boldsymbol{u}^{(1)}(z)\boldsymbol{g}_1^{(2)}(z) + \boldsymbol{u}^{(2)}(z)\boldsymbol{g}_2^{(2)}(z) = 1(0) + (1+z)(1) = 1+z \\
\boldsymbol{v}^{(3)}(z) &= \textstyle\sum_{i=1}^{2} \boldsymbol{u}^{(i)}(z)\boldsymbol{g}_i^{(3)}(z) \\
&= \boldsymbol{u}^{(1)}(z)\boldsymbol{g}_1^{(3)}(z) + \boldsymbol{u}^{(2)}(z)\boldsymbol{g}_2^{(3)}(z) = 1(z) + (1+z)(z+z^2) = z^3.
\end{aligned}$$

Por tanto, $\boldsymbol{v}(z) = (1, 1+z, z^3)$.

Ahora, con el algoritmo `ksecgenpol` A.7, ver p. 102, las entradas son

```
P.<z> = GF(2)[]
g11=1+0*z; g12=0+0*z; g13=z
g21=0+0*z; g22=1+0*z; g23=z+z^2
g=[[g11,g12,g13],[g21,g22,g23]]
u1=1; u2=1+z
u=[u1,u2]
```

después de llamar el algoritmo `ksecgenpol` en SAGE se tiene por salida

```
ksecgenpol(u,g)
[1, z + 1, z^3]
```

Ejemplo 2.13.

Considere el $(3, 2, 2)$-codificador cuyos polinomios generadores son

$$\begin{aligned}
\boldsymbol{g}_1^{(1)} &= 1, & \boldsymbol{g}_2^{(1)} &= 0, \\
\boldsymbol{g}_1^{(2)} &= 0, & \boldsymbol{g}_2^{(2)} &= 1, \\
\boldsymbol{g}_1^{(3)} &= z, & \boldsymbol{g}_2^{(3)} &= z + z^2.
\end{aligned}$$

Luego, para codificar la 2-secuencia polinomial $\boldsymbol{u}(z) = (1+z+z^3+z^4+z^6, z+z^2+z^5+z^6)$, usando el algoritmo A.7 implementado en `SAGE`, se tiene

```
P.<z> = GF(2)[]
g11=1+0*z; g12=0+0*z; g13=z
```

```
g21=0+0*z; g22=1+0*z; g23=z+z^2
g=[[g11,g12,g13],[g21,g22,g23]]
u1=1+z+z^3+z^4+z^6; u2=z+z^2+z^5+z^6
u=[u1,u2]
ksecgenpol(u,g)
[z^6 + z^4 + z^3 + z + 1, z^6 + z^5 + z^2 + z,
 z^8 + z^7 + z^6 + z^5 +z]
```

4. Matriz generadora polinomial

A partir de la representación polinomial de las secuencias generadoras, se tiene que la matriz generadora semi-infinita dada en la ecuación (2.11) tiene su respectiva representación polinomial, llamada la matriz generadora polinomial y está dada por

$$G(z) = \begin{pmatrix} \boldsymbol{g}_1^{(1)}(z) & \boldsymbol{g}_1^{(2)}(z) & \cdots & \boldsymbol{g}_1^{(n)}(z) \\ \boldsymbol{g}_2^{(1)}(z) & \boldsymbol{g}_2^{(2)}(z) & \cdots & \boldsymbol{g}_2^{(n)}(z) \\ \vdots & \vdots & & \vdots \\ \boldsymbol{g}_k^{(1)}(z) & \boldsymbol{g}_k^{(2)}(z) & \cdots & \boldsymbol{g}_k^{(n)}(z) \end{pmatrix}.$$

donde, cada entrada $\boldsymbol{g}_i^{(j)}(z)$ es un polinomio. Así, la operación de codificación está dada por $\boldsymbol{v}(z) = \boldsymbol{u}(z)G(z)$.

Los polinomios generadores $\boldsymbol{g}_i^{(j)}(z)$ para cualquier i dado y para todo $j = 1, \ldots, n$ detectan la influencia de la entrada i-ésima en todas las n salidas. El polinomio generador o polinomios de mayor grado para un i dado, determina el número máximo de símbolos de información pasados del flujo de entrada que necesitan ser almacenados en el i-ésimo registro de desplazamiento.

Además, note que la matriz $G(z)$ puede obtenerse también a partir de las submatrices generadoras dadas en la ecuación (2.8) como sigue

$$G(z) = G_0 + G_1 z + G_2 z^2 + \cdots + G_m z^m. \tag{2.16}$$

A continuación se presenta ejemplos en los cuales para la codificación se hace uso de la matriz generadora polinomial.

Ejemplo 2.14.

La matriz generadora polinomial del $(2, 1, 2)$-codificador de la figura 2.2 es

$$G_1(z) = \begin{pmatrix} \boldsymbol{g}^{(1)}(z) & \boldsymbol{g}^{(2)}(z) \end{pmatrix} = \begin{pmatrix} 1+z^2 & 1+z+z^2 \end{pmatrix}. \tag{2.17}$$

Para codificar la 1-secuencia de información polinomial $\boldsymbol{u}(z) = 1 + z$, basta con realizar el producto escalar entre $\boldsymbol{u}(z)$ y $G_1(z)$. Así, la 2-secuencia código es

$$\boldsymbol{v}(z) = (z^3 + z^2 + z + 1 \quad z^3 + 1).$$

Haciendo uso del algoritmo `matgenpol` A.8, ver p. 103, se tiene que las entradas son

```
P.<z> = GF(2)[]
g1=1+z^2; g2=1+z+z^2
g=vector(P,[g1,g2])
u=1+z
```

después de llamar el algoritmo la salida es

```
matgenpol(u,g)
(z^3 + z^2 + z + 1, z^3 + 1)
```

Ejemplo 2.15.

Considere el $(5, 1, 3)$-codificador convolucional para el cual una matriz generadora polinomial es

$$\begin{aligned} G(z) &= \begin{pmatrix} \boldsymbol{g}^{(1)}(z) & \boldsymbol{g}^{(2)}(z) & \boldsymbol{g}^{(3)}(z) & \boldsymbol{g}^{(4)}(z) & \boldsymbol{g}^{(5)}(z) \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1+z^2 & z & z+z^2+z^3 & 1+z+z^2+z^3 \end{pmatrix}. \end{aligned}$$

Para codificar la 1-secuencia de información $\boldsymbol{u}(z) = 1 + z + z^2 + z^3 + z^5 + z^7 + z^8$ se usa el algoritmo A.8 y se tiene:

```
P.<z> = GF(2)[]
g1=1; g2=1+z^2; g3=z; g4=z+z^2+z^3; g5=1+z+z^2+z^3
g=vector(P,[g1,g2,g3,g4,g5])
u=1+z^2+z^3+z^5+z^7+z^8
matgenpol(u,g)
(z^8+z^7+z^5+z^3+z^2+1, z^10+z^9+z^8+z^4+z^3+1,
```

```
z^9+z^8+z^6+z^4+z^3+z, z^11+z^7+z^2+z, z^11+z^8+z^5+z^3+z+1)
```

Ejemplo 2.16.

La matriz generadora polinomial del $(3, 2, 2)$-codificador de la figura 2.3 es

$$G_2(z) = \begin{pmatrix} \boldsymbol{g}_1^{(1)}(z) & \boldsymbol{g}_1^{(2)}(z) & \boldsymbol{g}_1^{(3)}(z) \\ \boldsymbol{g}_2^{(1)}(z) & \boldsymbol{g}_2^{(2)}(z) & \boldsymbol{g}_2^{(3)}(z) \end{pmatrix} = \begin{pmatrix} 1 & 0 & z \\ 0 & 1 & z+z^2 \end{pmatrix}. \tag{2.18}$$

En este caso para realizar la codificación de una 2-secuencia de información $\boldsymbol{u}(z) = (1 \quad 1+z)$, es suficiente con realizar el producto del vector $\boldsymbol{u}(z)$ por la matriz generadora, esto es

$$\boldsymbol{u}(z)G_2(z) = (1 \quad 1+z) \begin{pmatrix} 1 & 0 & z \\ 0 & 1 & z+z^2 \end{pmatrix} = \begin{pmatrix} 1 & 1+z & z^3 \end{pmatrix}.$$

Por lo tanto, la 3-secuencia código es $\boldsymbol{v}(z) = (1 \quad 1+z \quad z^3)$.

Para usar el algoritmo `kmatgenpol` A.9, ver p. 104 , es necesario en SAGE escribir las entradas como sigue

```
P.<z> = GF(2)[]
g11=1+0*z; g12=0+0*z; g13=z
g21=0+0*z; g22=1+0*z; g23=z+z^2
G=Matrix(P,2,3,[g11,g12,g13,g21,g22,g23])
u1=1; u2=1+z
U=vector(P,[u1,u2])
```

Después se llama el algoritmo para obtener el secuencia código

```
kmatgenpol(U,G)
(1, z + 1, z^3)
```

Ejemplo 2.17.

Considere el $(3, 2, 2)$-codificador cuya matriz generadora polinomial es la misma del ejemplo 2.16, para codificar la 2-secuencia de información polinomial

$$\boldsymbol{u}(z) = (1 + z + z^3 + z^4 + z^6, z + z^2 + z^5 + z^6),$$

con ayuda del algoritmo `kmatgenpol` A.9. Se tiene

```
P.<z> = GF(2)[]
g11=1+0*z; g12=0+0*z; g13=z
g21=0+0*z; g22=1+0*z; g23=z+z^2
G=Matrix(P,2,3,[g11,g12,g13,g21,g22,g23])
u1=1+z+z^3+z^4+z^6; u2=z+z^2+z^5+z^6
U=vector(P,[u1,u2])
kmatgenpol(U,G)
(z^6+z^4+z^3+z+1, z^6+z^5+z^2+z, z^8+z^7+z^6+z^5+z)
```

2.3.2. Representaciones gráficas.

Otra forma de representar el proceso de codificación convolucional es mediante gráficas. En este apartado se exponen dos representaciones gráficas, las cuales serán necesarias para la decodificación mediante el algoritmo de Viterbi, ver Sección 2.4.1.

1. Diagrama de estado.

Cada matriz polinómica de un código convolucional se puede asociar con un diagrama de estado que permite hacer la codificación. El diagrama de estado está íntimamente relacionado con el diagrama de registro de desplazamiento lineal y proporciona una representación visual para encontrar la salida en cualquier tiempo t.

Primero, se presenta la construcción del diagrama de estado para un $(n, 1, m)$-código convolucional.

Definición 2.3.

El estado de un $(n, 1, m)$-codificador en el tiempo t consiste de los contenidos en los elementos de retardo en su respectivo registro de desplazamiento lineal en el tiempo t que entraron anteriormente; es decir, para el tiempo t el estado está dado por $(u_{t-1}, u_{t-2}, \ldots, u_{t-m})$.

En el ejemplo 2.14, el estado del codificador $G_1(z) = (1 + z^2 \quad 1 + z + z^2)$ en el tiempo t, es el contenido de los dos elementos de retardo del registro de desplazamiento lineal; esto es (u_{t-1}, u_{t-2}).

En general, $\boldsymbol{v} = \boldsymbol{u}G$ produce n ecuaciones para calcular los $v_t^{(j)}$, con $1 \leq j \leq n$ en el tiempo t en términos del estado $(u_{t-1}, u_{t-2}, \ldots, u_{t-m})$ y la entrada u_t. Por lo tanto, a partir de estas ecuaciones si se conocen el estado en el tiempo t y la entrada u_t, se puede calcular la salida $\boldsymbol{v}_t$.

Definición 2.4.

*El **conjunto de estados** de un codificador de un $(n, 1, m)$-código convolucional es el conjunto de todas las m-úplas binarias ordenadas; es decir, $\mathbb{F}_2^m$.*

El conjunto de estados es de tamaño 2^m y representa todos los posibles estados que un codificador puede tener en cualquier instante de tiempo.

Definición 2.5.

*El **diagrama de estados** es un grafo dirigido etiquetado el cual se determina de la siguiente manera. Los vértices del grafo son el conjunto de estados. Hay dos tipos de aristas dirigidas: sólida y discontinua. Una arista dirigida desde un vértice a otro es sólida si la entrada es 0 y discontinua si la entrada es 1; la arista es etiquetada con la salida. En otras palabras, una arista dirigida desde el vértice $(u_{t-1}, u_{t-2}, \ldots, u_{t-m})$ hacia el vértice $(u_t, u_{t-1}, u_{t-2}, \ldots, u_{t-(m+1)})$ es sólida si $u_t = 0$ y discontinua si $u_t = 1$ y es etiquetado con $(v_t^{(1)}, \ldots, v_t^{(n)})$.*

Note que al ver los vértices en los extremos de la arista, se determina el contenido del registro de desplazamiento lineal en el tiempo t y por tanto se puede calcular la etiqueta de la arista, dependiendo si la arista es continua o discontinua.

Ejemplo 2.18.

En la figura 2.4 se muestra el diagrama de estados para la matriz generadora $G_1(z)$ del ejemplo 2.14. Se presentan dos versiones del diagrama de estados. Se puede codificar cualquier mensaje atravesando el diagrama en la figura 2.4 *a*) siempre iniciando en el estado 00 y terminando en el mismo estado.

Por ejemplo, si 101 se va a codificar, comenzará en el estado 00, pasa por la arista discontinua pues la primera entrada es 1; esta arista termina en el estado 10 y tiene por salida 11. Luego pasa por la arista solida, pues la siguiente entrada es 0, desde el estado

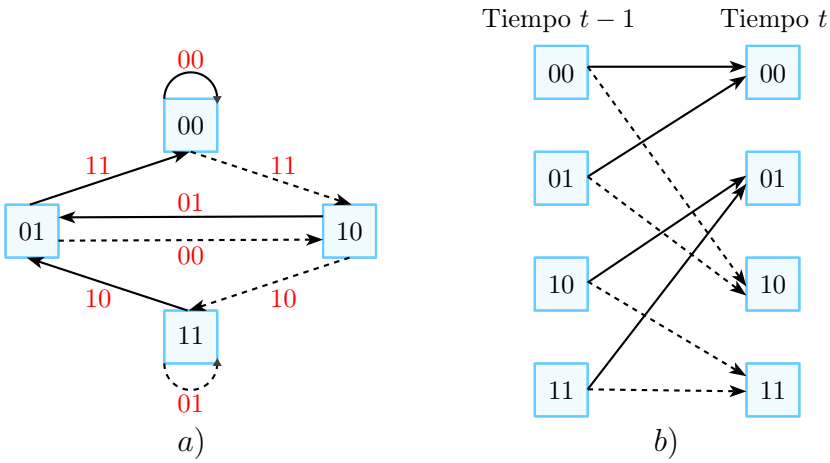


Figura 2.4: *Diagrama de estados del codificador de la figura 2.2*

10 al estado 01 y salida 01. Ahora, atraviesa el borde discontinuo, dado que la entrada es 1, desde el estado 01 al estado 10 y salida 00. Ya no hay más entradas, pero aún no se ha llegado al estado 00, por lo cual las siguientes entradas serán 0 hasta llegar al estado de todos cero. Por lo tanto se atraviesa desde el estado 10 hasta el estado 01 por la arista solida con salida 01; finalmente se pasa del estado 01 al estado 00 a lo largo de la arista solida con salida 11. En consecuencia, la codificación de 101 es por tanto la recopilación de las salidas que se fueron obteniendo en el proceso; esto es 11 01 00 01 11.

La figura 2.4 *b*) es esencialmente el mismo diagrama ampliado para mostrar los estados en el tiempo $t-1$ y t.

La misma idea general es valida para un (n,k)-código convolucional con matriz generadora polinomial $G(z) = (\boldsymbol{g}_i^{(j)}(z))$ con $k > 1$. Para $1 \leq i \leq k$, sea

$$m_i = \max_{1 \leq j \leq n} \operatorname{gr}(\boldsymbol{g}_i^{(j)}(z))$$

el máximo grado entre las entradas de la fila i; m_i se denomina ***el grado de la*** i***-ésima fila de*** $G(z)$. Se adopta la convención de que el polinomio cero tiene grado $-\infty$. Sea $\boldsymbol{u} = (\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(k)})$ una k-secuencia de información siendo $u_t^{(i)}$ la i-ésima entrada en el tiempo t. Sea $\boldsymbol{u}G = \boldsymbol{v} = (\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)})$ la n-secuencia código resultante. Cada $v_t^{(j)}$ es una combinación de algunos de los $u_T^{(i)}$ con $T \in \{t, t-1, \ldots, t-m_i\}$. De ahí, si se conocen las entradas $u_T^{(i)}$ y todas las entradas de orden i desde el tiempo $t-1$ hasta regresar al tiempo $t-m_i$, se puede determinar la salida. Así que en el instante t, se dice que el codificador

está en el estado $(u_{t-1}^{(1)}, \ldots, u_{t-m_1}^{(1)}, u_{t-1}^{(2)}, \ldots, u_{t-m_2}^{(2)}, \ldots, u_{t-1}^{(k)}, \ldots, u_{t-m_k}^{(k)})$. El ***conjunto de estados de*** G es el conjunto de todas las $(m_1 + m_2 + \cdots + m_k)$-úplas binarias ordenadas. El conjunto de estados es de tamaño $2^{m_1+m_2+\cdots+m_k}$ y representa todos los posibles estados que un codificador puede tener en cualquier tiempo. Note que la memoria m del codificador G es el máximo de los m_i; en particular cuando $k = 1$, se tiene que $m = m_1$.

El diagrama de estados para el caso $k > 1$ se construye como en el caso $k = 1$. Los vértices del grafo dirigido etiquetado son el conjunto de estados. Sin embargo esta vez, deben haber 2^k tipos de aristas que representan cada una de las posibles entradas $(u_t^{(1)}, u_t^{(2)}, \ldots, u_t^{(k)})$. De nuevo la arista se etiqueta con la salida $(v_t^{(1)}, v_t^{(2)}, \ldots, v_t^{(k)})$.

Ejemplo 2.19.

En la figura 2.5 se muestra el diagrama de estados para el $(3, 2, 2)$-código convolucional cuyo codificador físico se muestra en la figura 2.3 y tiene matriz generadora polinimial

$$G_2(z) = \begin{pmatrix} 1 & 0 & z \\ 0 & 1 & z + z^2 \end{pmatrix},$$

del ejemplo 2.16.

Para la construcción del diagrama de estados es necesaria la siguiente información

1. Para cada instante de tiempo t el estado es de la forma $(u_{t-1}^{(1)}, u_{t-1}^{(2)}, u_{t-2}^{(2)})$, dado que el grado de la fila 1 y 2 de la matriz $G_2(z)$ son respectivamente $m_1 = 1$ y $m_2 = 2$.

2. El diagrama de estados está compuesto por 8 vértices, que son los elementos del conjunto de estados $\mathbb{F}_2^3 = \{(0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,0), (1,0,1), (0,1,1), (1,1,1)\}$.

3. Las entradas en cada instante de tiempo son de la forma $(u_t^{(1)}, u_t^{(2)})$. Las posibles entradas son:
   - $(0,0)$ representada en el diagrama por una línea solida roja ——
   - $(0,1)$ representada en el diagrama por una línea discontinua negra - - - -
   - $(1,0)$ representada en el diagrama por una línea solida negra ——
   - $(1,1)$ representada en el diagrama por una línea discontinua roja - - - -

4. La salida para cada instante de tiempo es $\boldsymbol{v}_t = (v_t^{(1)}, v_t^{(2)}, v_t^{(3)})$ donde cada componente

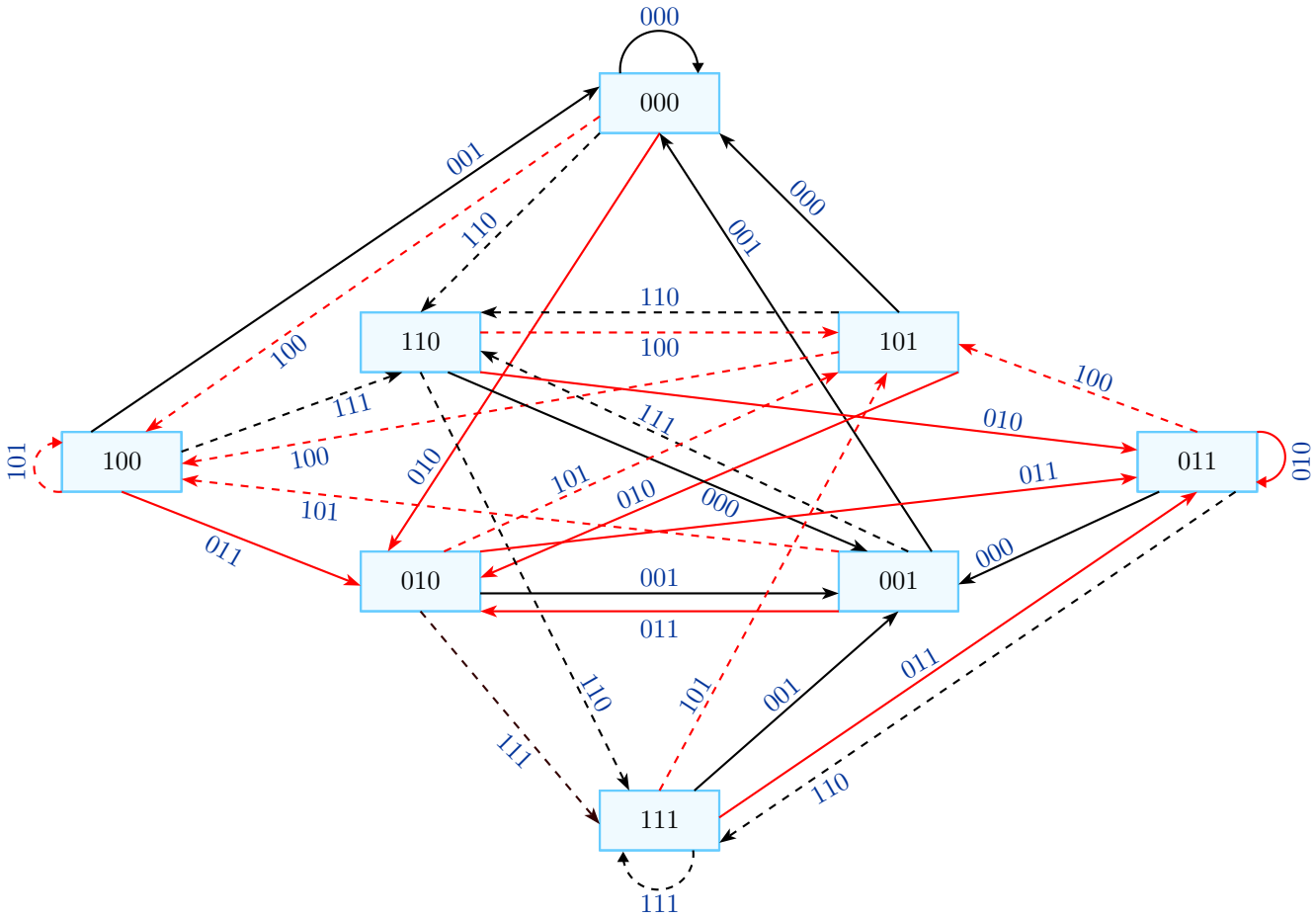


Figura 2.5: *Diagrama de estados del codificador de la Figura 2.3*

está determinada por las siguientes ecuaciones

$$v_t^{(1)} = u_t^{(1)}, \quad v_t^{(2)} = u_t^{(2)} \quad \text{y} \quad v_t^{(3)} = u_{t-1}^{(1)} + u_{t-1}^{(2)} + u_{t-2}^{(2)} (\text{mód} 2). \tag{2.19}$$

2. Diagrama enrejado

El diagrama enrejado para un codificador de un código convolucional es solamente una extensión del diagrama de estados comenzando en el instante $t = 0$. Se ilustra esto en la figura 2.6 para la matriz generadora $G_1(z)$ del $(2, 1)$-código convolucional del ejemplo 2.14.

Los estados son etiquetados $a = 00$, $b = 01$, $c = 10$ y $d = 11$. Observe que el enrejado es una extensión de la figura 2.4 *b*), entendiendo que en el instante $t = 0$, el único estado utilizado es el estado a; entonces, en el tiempo $t = 1$ los únicos estados que se pueden alcanzar son a y c, por lo cual son los únicos dibujados hasta ese momento. La codificación se puede lograr recorriendo el enrejado de izquierda a derecha iniciando en el estado a y terminando

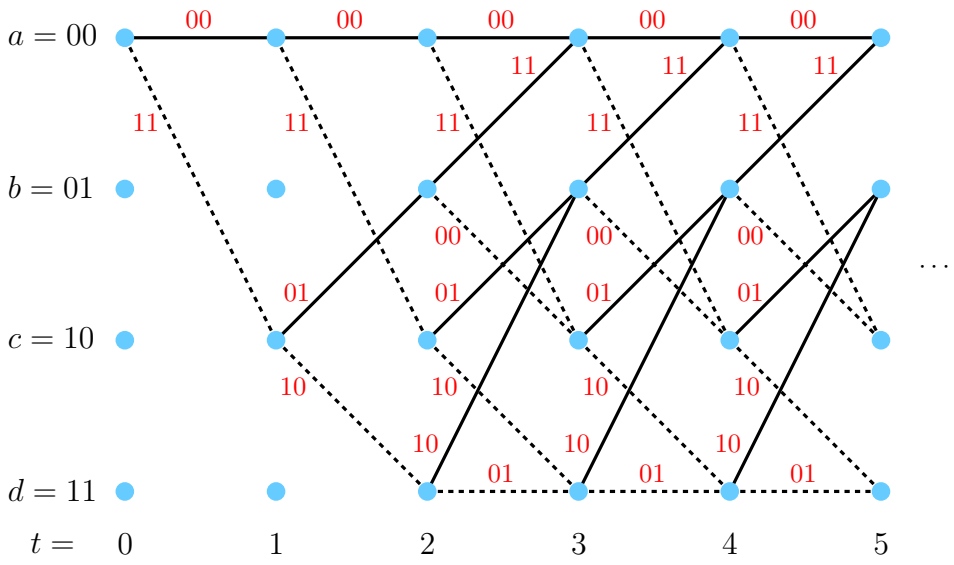


Figura 2.6: *Enrejado del codificador* $G_1(z) = (1 + z^2 \quad 1 + z + z^2)$

en el estado a. El estado $s \in \{a, b, c, d\}$ en el tiempo t se denota por s_t. Por ejemplo, para codificar 101 se pasa por la trayectoria $a_0c_1b_2c_3b_4a_5$ iniciando en una arista discontinua, a continuación una solida, luego otra discontinua y finalmente dos aristas solidas, ya que se deben adicionar dos ceros como entrada para terminar en el estado a. Esto produce la 2-secuencia código 11 01 00 01 11 luego de escribir las etiquetas en las aristas de la trayectoria recorrida.

2.4. Decodificación

En esta sección se expone un método de decodificación para un (n, k, m)-código convolucional llamado Algoritmo de Viterbi y en el cual es necesario el uso de las representaciones gráficas expuestas en la Sección 2.3.2. Cabe aclarar que este método es eficiente para valores de n y k pequeños.

2.4.1. El algoritmo de Viterbi

El algoritmo de Viterbi utiliza el diagrama enrejado de un codificador de un código convolucional para determinar la decodificación de una secuencia recibida. La versión del algoritmo que se presenta realiza la decodificación al vecino más cercano, que no es otra cosa que la decodificación por distancia mínima, ver observación 1.1 . Supóngase que una

k-secuencia de información de longitud N se codifica usando una matriz generadora con memoria m. El algoritmo requiere que se considere la posición del enrejado que inicia en el instante $t = 0$ y termina en el momento $M = N + m$; este se denomina el enrejado truncado N. La figura 2.7 muestra el enrejado truncado en $N = 6$ para la matriz generadora $G_1(z)$ del ejemplo 2.14 que se utiliza para decodificar un mensaje de longitud 6.

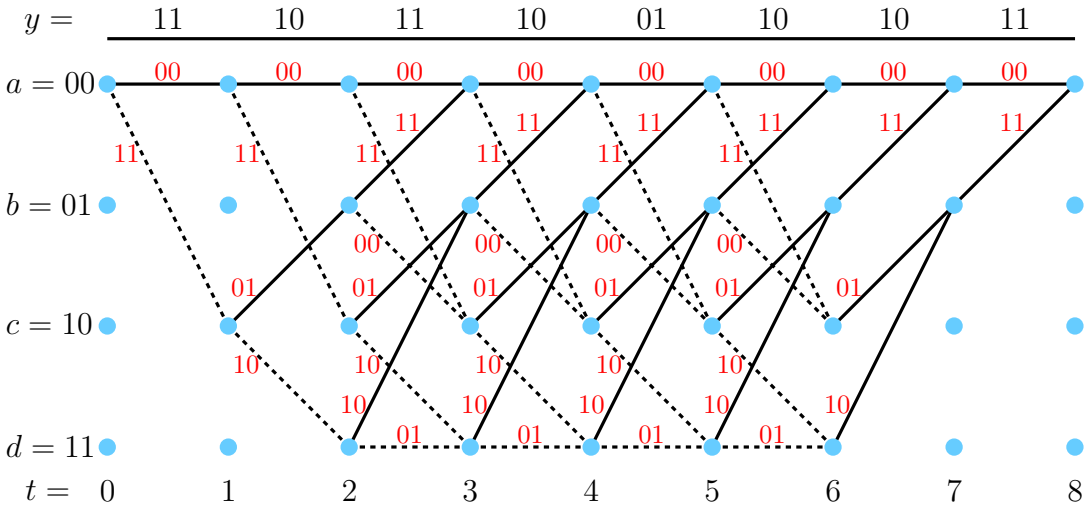


Figura 2.7: *Enrejado truncado $N = 6$ del codificador $G_1(z) = (1 + z^2 \quad 1 + z + z^2)$.*

Suponga que la k-secuencia de información $\boldsymbol{u}^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, \ldots, u_k^{(i)})$ con $0 \leq i \leq N - 1$ es codificada usando la matriz generadora G para generar una n-secuencia código $\boldsymbol{v}^{(j)} = (v_0^{(j)}, v_1^{(j)}, v_2^{(j)}, \ldots, v_{(N+m)-1}^{(j)})$ con $1 \leq j \leq n$. En lo que sigue se asume que $\boldsymbol{c}^{(j)} = (c_0^{(j)}, c_1^{(j)}, c_2^{(j)}, \ldots, c_{(N+m)-1}^{(j)})$ con $1 \leq j \leq n$ es recibida.

A continuación se presentan algunas definiciones necesarias para el algoritmo de Viterbi.

Definición 2.6.

Si e es la arista que conecta el estado s del instante $t - 1$ con el estado s' en el tiempo t, entonces el peso de la arista e es la distancia de Hamming entre la etiqueta de la arista e y la componente $\boldsymbol{c}_{t-1}$ de la n-secuencia recibida.

Por otro lado, el peso de una trayectoria P a través del enrejado es la suma de los pesos de las aristas de la trayectoria.

De la definición anterior se tiene que el peso de las aristas y de las trayectorias dependen del vector recibido.

Definición 2.7.

Se denota el estado cero por a y en el tiempo 0 por a_0. Suponga que P es una trayectoria en el enrejado que inicia en a_0 y termina en el instante t en el estado s. La trayectoria P se dice es una trayectoria sobreviviente en el instante t, si su peso es el más pequeño entre los pesos de todos los caminos que inician en a_0 y terminan en el estado s en el tiempo t. La colección de todas las trayectorias sobrevivientes que inician en a_0 y terminan en el estado s en el instante t se denota por $\mathcal{S}(s,t)$.

Si P es una trayectoria en el enrejado, partiendo de a_0 y terminando en el instante I, se define $\boldsymbol{c}_P$ a la n-secuencia código asociada con P, donde $\boldsymbol{c}_{Pt}$ es la etiqueta en la arista en P desde el estado en el tiempo t hasta el estado en el tiempo $t+1$ para $0 \leq t \leq I$. Adicionalmente, se define como $\boldsymbol{u}_P$ el mensaje asociado con P donde $\boldsymbol{u}_{Pt}$ es la entrada identificada por el tipo de arista en P desde el estado en el instante t hasta el estado en el instante $t+1$ para $0 \leq t \leq \text{mín}\{I, N\}$.

Algoritmo 2.1.

El algoritmo de decodificación de Viterbi consta de los siguientes pasos

1. Dibujar el diagrama enrejado truncado en N de $G(z)$, reemplazando las etiquetas de las aristas por el peso de cada arista. Denote con a el estado $\mathbf{0}$.

2. Calcular $\mathcal{S}(s,1)$ para todos los estados s utilizando el enrejado del paso 1.

3. Repetir lo siguiente para $t = 2, 3, \ldots, M$ usando el enrejado del paso 1. Suponiendo que $\mathcal{S}(s,t-1)$ fue calculado para todos los estados s, calcular $\mathcal{S}(s,t)$ para todo s de la siguiente manera. Para cada estado s' y cada arista e desde s' a s, formar la trayectoria P conformada por P' seguida de e donde $P' \in \mathcal{S}(s,t-1)$. Incluir P en $\mathcal{S}(s,t)$ unicamente si tiene el peso más pequeño entre todas las trayectorias.

4. Un vecino más cercano a $\boldsymbol{c}$ es cualquier $\boldsymbol{c}_P$ para $P \in \mathcal{S}(s,M)$ obtenido a partir del mensaje dado por $\boldsymbol{u}_P$.

Ejemplo 2.20.

En este ejemplo se ilustra el algoritmo de Viterbi para decodificar la 2-secuencia recibida

$y = 11\ 10\ 11\ 10\ 01\ 10\ 10\ 11$ para lo cual inicialmente se usa el enrejado de la figura 2.7, en el cual aparece y en la parte superior. El enrejado del paso 1 se muestra en la figura 2.8.

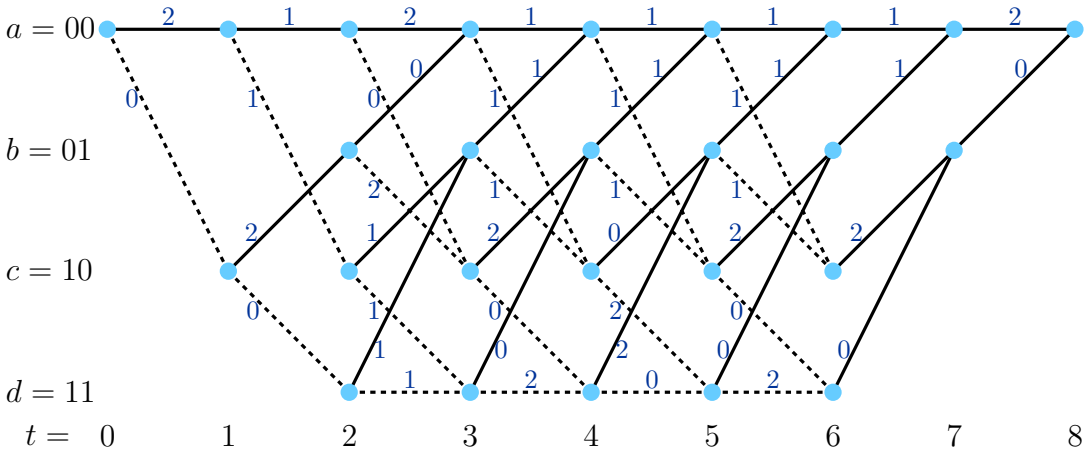


Figura 2.8: *Pesos en el enrejado de la figura 2.7.*

Ahora para el paso 2, se tiene

$$\mathcal{S}(a,1) = \{a_0a_1\}, \qquad \mathcal{S}(b,1) = \phi,$$
$$\mathcal{S}(c,1) = \{a_0c_1\}, \qquad \mathcal{S}(d,1) = \phi.$$

A partir de estos conjuntos se puede calcular $\mathcal{S}(s,2)$ usando la regla del paso 3 y se obtiene

$$\mathcal{S}(a,2) = \{a_0a_1a_2\}, \qquad \mathcal{S}(b,2) = \{a_0c_1b_2\},$$
$$\mathcal{S}(c,2) = \{a_0a_1c_2\}, \qquad \mathcal{S}(d,2) = \{a_0c_1d_2\}.$$

Se observa que las trayectorias en $\mathcal{S}(s,2)$ inician con a_0a_1 o a_0c_1, las cuales son las trayectorias de $\mathcal{S}(s',1)$. Se puede construir $\mathcal{S}(a,3)$ a partir de los caminos en cada $\mathcal{S}(s,2)$ y se tiene que solo hay dos posibilidades, una es $a_0a_1a_2$ seguida por la arista a_2a_3 resultando la trayectoria $a_0a_1a_2a_3$ y la otra es $a_0c_1b_2$ seguida por la arista b_2a_3 resultando la trayectoria $a_0c_1b_2a_3$; sin embargo, la primera tiene peso 5 y la segunta tiene peso 2, por lo tanto $\mathcal{S}(a,3) = \{a_0c_1b_2a_3\}$. El cálculo de los otros $\mathcal{S}(s,3)$ es similar. De esta forma se continúa hasta calcular $\mathcal{S}(a,8)$. Todas las trayectorias sobreviviventes se muestran en la figura 2.9, en la cual para hacer los cálculos más sencillos se ha graficado cada trayectoria con el peso en color rojo bajo su vértice final.

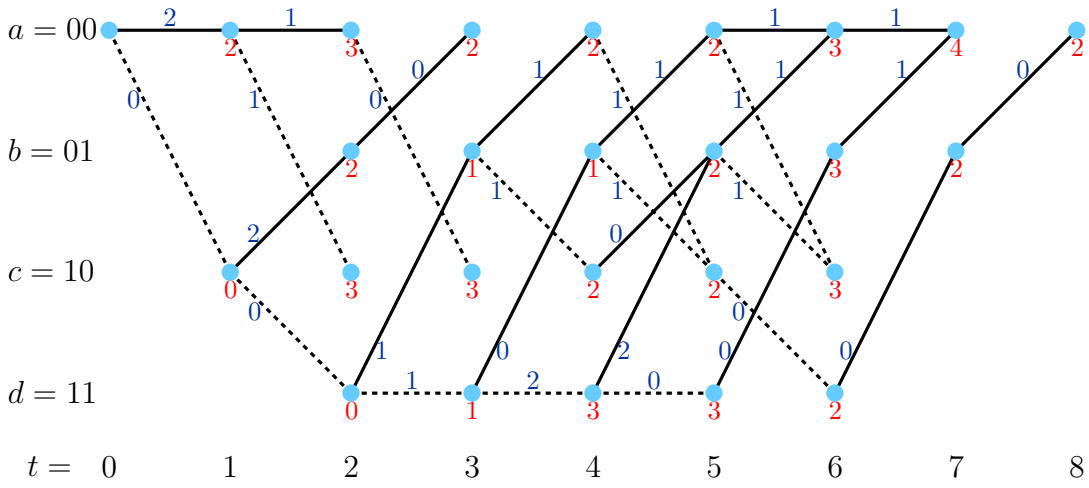


Figura 2.9: *Decodificación en el enrejado de la figura 2.7.*

Note que algunas trayectorias se interrumpen, esto se debe a que cuando se añade una arista más a la trayectoria, la trayectoria resultante tiene mayor peso que otras trayectorias que terminan en el mismo vértice.

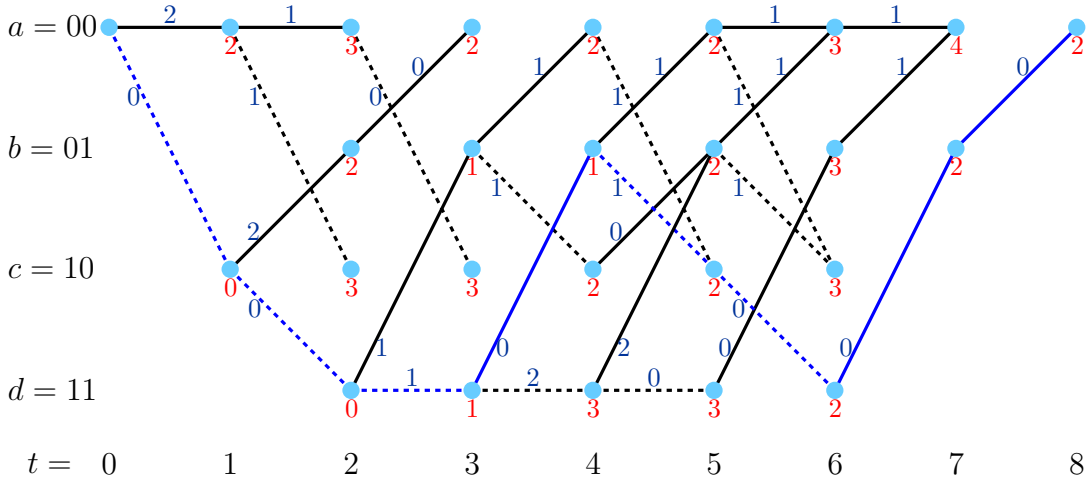


Figura 2.10: *Trayectoria P el enrejado de la figura 2.7.*

Para finalizar con el paso 4, note que $\mathcal{S}(a, 8)$ contiene solo una trayectoria esta es $P = a_0c_1d_2d_3b_4c_5d_6b_7a_8$ la cual tiene peso 2 y se resalta en color azul en la figura 2.10. Así, la palabra código $\boldsymbol{c}_P$ obtenida de la trayectoria P en la figura 2.7 es 11 10 10 10 00 10 10 11 que se diferencia en 2 posiciones de la 2-secuencia recibida. Además, como aristas solidas provienen de entrada 0 y las aristas discontinuas de entradas 1 entonces, de las 6 primeras aristas en la trayectoria P se puede deducir que la 1-secuencia de información enviada fue 111011.

2.5. Código convolucional dual

En esta sección se presenta la definición de código convolucional dual y su relación con el síndrome de una n-secuencia recibida, como también un método para determinar una matriz de control de paridad polinomial a partir de una matriz generadora polinomial.

De igual forma que para el caso de los códigos lineales un código convolucional es un subespacio vectorial y dado que todo subespacio vectorial tiene su respectivo complemento ortogonal surge el hecho de que cada código convolucional también tiene su respectivo código convolucional dual.

Definición 2.8.

Sea C un (n,k)-código convolucional. Entonces su código convolucional dual $C^{\perp}$ es un subespacio $(n-k)$-dimensional del espacio vectorial $\mathbb{F}_q((z))^n$, el cual es ortogonal a C.

El código convolucional dual $C^{\perp}$ es generado por cualquier codificador $H(z)$ de velocidad $R = n - k/n$ tal que

$$G(z)H(z)^T = \mathbf{0}. \tag{2.20}$$

La matriz $H(z)$ se denomina *la matriz de control de paridad* del código C. Un método para determinar $H(z)$ a partir de la matriz $G(z)$ se presenta más adelante en esta sección.

El codificador de un código convolucional dual, puede representarse por medio de cualquiera de los métodos expuestos en la Sección 2.3, pues este sigue siendo el codificador de un código convolucional.

2.5.1. Codificador polinomial de un código convolucional dual

A continuación se presenta un método para determinar una matriz generadora polinomial $H(z)$ de un código convolucional dual $C^{\perp}$, a partir de una matriz generadora polinomial $G(z)$ de un código convolucional C mediante la aplicación del Teorema 1.1, el cual permite determinar la descomposición en factores invariantes de $G(z)$.

Sea $G(z)$ una matriz generadora polinomial de un código convolucional C. Del Teorema 1.1 se tiene que una descomposición en factores invariantes de $G(z)$ tiene la forma

$$G(z) = A(z)\Gamma(z)B(z),$$

donde $A(z)$ es una matriz de operaciones elementales por fila de tamaño $k \times k$ y $B(z)$ es una matriz de operaciones elementales por columna de tamaño $n \times n$, luego tales matrices son polinomiales e invertibles y $B^{-1}(z)$ existe.

Ahora, sea $B_1(z)$ la matriz compuesta por las últimas $(n-k)$-columnas de $B^{-1}(z)$. Entonces, la matriz de control de paridad $H(z)$ para el código convolucional dual puede definirse como

$$H(z) = B_1(z)^T. \tag{2.21}$$

La matriz $H(z)$ definida en la ecuación (2.21) tiene rango $n-k$. A continuación se garantiza que $H(z)$ satisface la condición de la ecuación (2.20); esto es $G(z)B_1(z) = \mathbf{0}$.

Como $G(z) = A(z)\Gamma(z)B(z)$ entonces $G(z)B_1(z) = A(z)\Gamma(z)B(z)B_1(z)$. Sea $I_n = [\delta_{ij}]$ tal que $\delta_{ij} = 1$ si $i = j$ y $\delta_{ij} = 0$ si $i \neq j$. Dado que $B_1(z)$ es la matriz formada por las últimas $n-k$ columnas de $B(z)^{-1}$ y sabiendo que $B(z)B(z)^{-1} = I_n$ se tiene

$$B(z)B_1(z) = \begin{pmatrix} \delta_{1,k+1} & \delta_{1,k+2} & \delta_{1,k+3} & \cdots & \delta_{1,n} \\ \delta_{2,k+1} & \delta_{2,k+2} & \delta_{2,k+3} & \cdots & \delta_{2,n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \delta_{k+1,k+1} & \delta_{k+1,k+2} & \delta_{k+1,k+3} & \cdots & \delta_{k+1,n} \\ \delta_{k+2,k+1} & \delta_{k+2,k+2} & \delta_{k+2,k+3} & \cdots & \delta_{k+2,n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \delta_{n,k+1} & \delta_{n,k+2} & \delta_{n,k+3} & \cdots & \delta_{n,n} \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{k\times(n-k)} \\ I_{n-k} \end{pmatrix}.$$

Ahora, como $G(z)$ es de rango k, del Teorema 1.1 se concluye que $G(z)$ tiene k factores invariantes, esto es

$$\Gamma(z) = \begin{pmatrix} \gamma_1 & 0 & 0 & \cdots & 0 & \\ 0 & \gamma_2 & 0 & \cdots & 0 & \\ 0 & 0 & \gamma_3 & \cdots & 0 & \mathbf{0}_{k\times(n-k)} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \\ 0 & 0 & 0 & \cdots & \gamma_k & \end{pmatrix}.$$

Así, $\Gamma(z)B(z)B_1(z) = \mathbf{0}_{k\times(n-k)}$, por lo tanto $G(z)H(z)^T = \mathbf{0}$ con $H(z)$ como se definió en la ecuación (2.21).

Note que la matriz de control de paridad de un código convolucional no es única, ya que

es posible obtener otra si $H(z)$ se multiplica por cualquier matriz cuadrada no singular de tamaño $(n-k)$.

Ejemplo 2.21.

La matriz generadora polinomial del $(2,1,2)$-codificador de la figura 2.2 es

$$G(z) = \begin{pmatrix} 1+z^2 & 1+z+z^2 \end{pmatrix}.$$

Para determinar la matriz de control de paridad de este codificador se debe encontrar una descomposición en factores invariantes de $G(z)$ para lo cual se usa el proceso descrito en la demostración del Teorema 1.1, como sigue.

Dado que $deg(1+z^2) = deg(1+z+z^2) = 2$ no se requiere transformar la matriz $G(z)$,pues un elemento de grado mínimo ya se encuentra en la posición (1,1). Luego se divide $1+z+z^2$ entre $1+z^2$, y se tiene que

$$1+z+z^2 = (1+z^2)1+z \text{ con } deg(z) < deg(1+z^2).$$

A continuación se trasforma la matriz $G(z)$ sumandole a la columna 2 la columna 1 esto es:

$$G(z)T_{12}(1) = \begin{pmatrix} 1+z^2 & z \end{pmatrix}.$$

Obteniendo una matriz $G_1(z) = (1+z^2 \quad z)$, en la cual el grado mínimo de las entradas no nulas es menor que en $G(z)$ y debido a que la componente $(1,2)$ es diferente de cero se repite el proceso aplicado a la matriz $G_1(z)$.

Como $deg(z) < deg(1+z^2)$, primero se debe permutar las columnas 1 y 2, esto es

$$G_1(z)P_{12} = \begin{pmatrix} z & 1+z^2 \end{pmatrix}.$$

Ahora, al dividir $1+z^2$ entre z se obtiene

$$1+z^2 = z(z)+1 \text{ con } deg(1) < deg(z).$$

Se transforma la matriz $G_1(z)P_{12}$ sumandole a la columna 2 la columna 1 multiplicada por z obteniendo

$$G_1(z)P_{12}T_{12}(z) = (z \quad 1) = G_2(z).$$

Nuevamente, se obtiene una matriz $G_2(z)$ en la que el mínimo de las entras no nulas es menor que en $G_1(z)$ y cuya componente $(1,2)$ es no nula, así que se debe repetir el proceso a la matriz $G_2(z)$.

Dado que $deg(1) < deg(z)$ se intercambia las columnas 1 y 2 de la matriz $G_2(z)$ y luego se suma la columna 1 multiplicada por z a la columna 2 y se tiene

$$G_2(z)P_{12}T_{12}(z) = \begin{pmatrix} 1 & 0 \end{pmatrix}.$$

En este caso la componente $(1,2)$ es cero, por lo cual el proceso termina así

$$\Gamma(z) = G(z)T_{12}(1)P_{12}T_{12}(z)P_{12}T_{12}(z) = \begin{pmatrix} 1 & 0 \end{pmatrix}.$$

Luego, $G(z) = \Gamma(z)T_{12}(z)P_{12}T_{12}(z)P_{12}T_{12}(1)$ de ahí que $B(z) = T_{12}(z)P_{12}T_{12}(z)P_{12}T_{12}(1)$ esto es

$$B(z) = \begin{pmatrix} 1+z^2 & 1+z+z^2 \\ z & 1+z \end{pmatrix} \quad \text{y} \quad B(z)^{-1} = \begin{pmatrix} 1+z & 1+z+z^2 \\ z & 1+z^2 \end{pmatrix}.$$

Así,

$$B_1(z) = \begin{pmatrix} 1+z+z^2 \\ 1+z^2 \end{pmatrix}.$$

Por lo tanto,

$$H(z) = \begin{pmatrix} 1+z+z^2 & 1+z^2 \end{pmatrix}.$$

Ejemplo 2.22.

La matriz generadora polinomial del $(3,2,2)$-codificador de la figura 2.3 es

$$G_2(z) = \begin{pmatrix} 1 & 0 & z \\ 0 & 1 & z+z^2 \end{pmatrix}.$$

Para determinar la matriz de control de paridad de este codificador se busca una descomposición en factores invariantes de $G_2(z)$. Así se obtiene

$$\Gamma(z) = G_2(z)T_{13}(z)T_{23}(z+z^2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Luego $G_2(z) = \Gamma(z)T_{23}(z+z^2)T_{13}(z)$, de ahí que $B(z) = T_{23}(z+z^2)T_{13}(z)$ y

$$B(z)^{-1} = B(z) = \begin{pmatrix} 1 & 0 & z \\ 0 & 1 & z+z^2 \\ 0 & 0 & 1 \end{pmatrix}.$$

Por lo tanto

$$H(z) = \begin{pmatrix} z & z+z^2 & 1 \end{pmatrix},$$

es la matriz de control de paridad buscada.

2.5.2. Ex-síndrome y síndrome convolucional

David Forney Jr. en [6] llama a la transpuesta de la matriz de control de paridad $H(z)$ el *ex-síndrome*[2] convolucional el cual tiene la propiedad,

$$\boldsymbol{v}(z)H(z)^T = \mathbf{0} \text{ si y sólo si } \boldsymbol{v}(z) \in C. \tag{2.22}$$

Sea $\boldsymbol{r}(z)$ la n-secuencia recibida cuando la n-secuencia código $\boldsymbol{v}(z)$ fue transmitida por un canal. La secuencia $\boldsymbol{r}(z) \in \mathbb{F}_q((z))^n$ puede ser diferente de la n-secuencia código transmitida $\boldsymbol{v}(z)$ dado que el canal puede introducir errores. Sea $\boldsymbol{e}(z)$ la n-secuencia de error, esto es

$$\boldsymbol{r}(z) = \boldsymbol{v}(z) + \boldsymbol{e}(z). \tag{2.23}$$

En la siguiente definición se evidencia que el ex-síndrome es útil en la formación de la $(n-k)$-secuencia *síndrome* correspondiente a la n-secuencia recibida $\boldsymbol{r}(z)$.

Definición 2.9.

La $(n-k)$-secuencia síndrome $\boldsymbol{s}(z)$ se define por

$$\boldsymbol{s}(z) = \boldsymbol{r}(z)H(z)^T. \tag{2.24}$$

Luego, de las ecuaciones (2.23) y (2.24) se tiene

$$\boldsymbol{s}(z) = \boldsymbol{v}(z)H(z)^T + \boldsymbol{e}(z)H(z)^T,$$

[2]En inglés, syndrome-former

pero como $\boldsymbol{v}(z) \in C$ se tiene $\boldsymbol{v}(z)H(z)^T = \mathbf{0}$ así

$$\boldsymbol{s}(z) = \boldsymbol{e}(z)H(z)^T.$$

Por lo tanto, el síndrome depende solo de los errores introducidos por el canal y no de la k-secuencia de información transmitida.

Como se hizo para el caso de la matriz generadora polinomial en la ecuación (2.16), el ex-síndrome $H(z)^T$ puede expresarse por medio de la siguiente ecuación

$$H(z)^T = H_0^T + H_1^T z + H_2^T z^2 + \cdots + H_{m_s}^T z^{m_s}, \tag{2.25}$$

donde H_i^T es una matriz con entradas en $\mathbb{F}_q$ de tamaño $n \times (n-k)$ para cada $0 \leq i \leq m_s$ y la memoria del ex-síndrome es m_s. En general, la memoria de un ex-síndrome no es igual a la memoria de la matriz generadora $G(z)$.

A partir de la ecuación (2.25), la condición dada en (2.22) puede expresarse como

$$v_t H_0^T + v_{t-1} H_1^T(z) + \cdots + v_{t-m_s} H_{m_s}^T(z)^{m_s} = \mathbf{0}, \text{ para } 0 \leq t \leq N + m - 1 \tag{2.26}$$

Luego, para cualquier n-secuencia código $\boldsymbol{v} = (\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{N+m-1})$ se tiene de forma equivalente que

$$\boldsymbol{v}H^T = \mathbf{0}$$

donde

$$H^T = \begin{pmatrix} H_0^T & H_1^T & H_2^T & \cdots & H_{m_s}^T & & \\ & H_0^T & H_1^T & \cdots & H_{m_s-1}^T & H_{m_s}^T & \\ & & H_0^T & \cdots & H_{m_s-2}^T & H_{m_s-1}^T & H_{m_s}^T \\ & & & \ddots & & & \ddots \end{pmatrix}, \tag{2.27}$$

es una matriz semi-infinita ex-síndrome correspondiente a la matriz semi-infinita generadora G dada en la ecuación (2.11) y las matrices H_p^T con $0 \leq p \leq m_s$ son las respectivas submatrices ex-síndrome. De lo anterior se sabe que

$$GH^T = \mathbf{0}.$$

Además, tanto las secuencias generadoras discretas como las polinomiales del ex-síndrome son las componentes de las matrices generadoras en su respectiva representación, de igual

forma que en la Sección 2.3 para el caso de una matriz generadora de un código convolucional. Dado que el síndrome se puede obtener por medio del producto de matrices este también se puede configurar en un circuito secuencial lineal, en dicho circuito se implementa la ecuación matricial $\boldsymbol{s} = \boldsymbol{r}H^T$.

A continuación, se presentan ejemplos de lo expuesto en esta subsección.

Ejemplo 2.23.

Para el $(2, 1, 2)$-codificador de la figura 2.2 en el ejemplo 2.21 se obtuvo que una matriz de control de paridad es

$$H(z) = \begin{pmatrix} 1 + z + z^2 & 1 + z^2 \end{pmatrix}.$$

Luego, el correspondiente ex-síndrome convolucional es

$$H(z)^T = \begin{pmatrix} 1 + z + z^2 \\ 1 + z^2 \end{pmatrix}.$$

De ahí que las submatrices ex-síndrome son

$$H_0^T = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad H_1^T = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ y } \quad H_2^T = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Así, la matriz semi-infinita ex-síndrome es

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

Cuyo circuito secuencial lineal correspondiente, que consta de dos entradas $r^{(1)}$, $r^{(2)}$ y una salida $\boldsymbol{s}$ se muestra en la figura 2.11

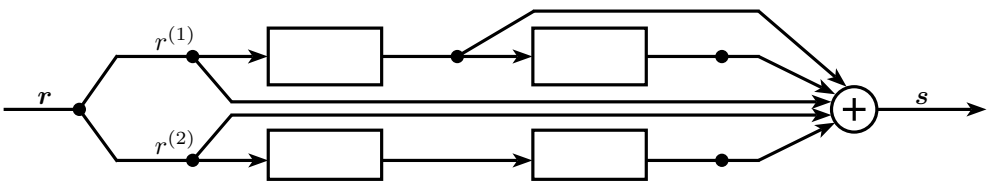


Figura 2.11: *Un ex-síndrome para el codificador la figura 2.2.*

Ejemplo 2.24.

Considere el $(3, 2, 2)$-codificador de la figura 2.3, para el cual en el ejemplo 2.22 se concluyó que

$$H(z) = \begin{pmatrix} z & z+z^2 & 1 \end{pmatrix},$$

es una matriz de control de paridad.

Ahora, su correspondiente ex-síndrome convolucional es

$$H(z)^T = \begin{pmatrix} z \\ z+z^2 \\ 1 \end{pmatrix}.$$

Luego, las submatrices ex-síndrome son

$$H_0^T = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad H_1^T = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \text{ y } \quad H_2^T = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Así, la matriz semi-infinita ex-síndrome es

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

Finalmente, su circuito secuencial lineal consta de tres entradas $r^{(1)}$, $r^{(2)}$, $r^{(3)}$ y una salida $\boldsymbol{s}$ el cual se muestra en la figura 2.12

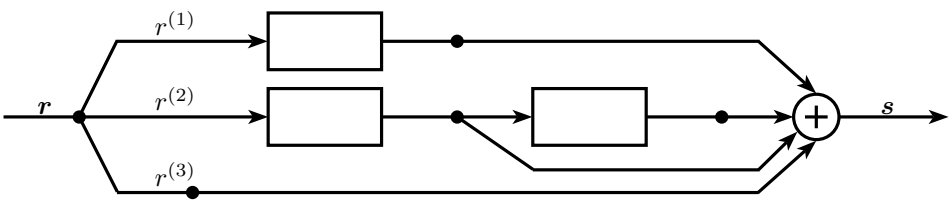


Figura 2.12: *Un ex-síndrome para el codificador la figura 2.3.*

Capítulo 3

Códigos Convolucionales Cíclicos

En este capítulo se presenta la definición de código convolucional cíclico para lo cual en adelante $\mathbb{F}_q$ es un campo finito y n un entero positivo tal que $q \nmid n$. El número n denota la longitud del código; cabe resaltar que la condición impuesta es la misma que se hace en la construcción de códigos de bloque cíclicos, esto para que el polinomio $x^n - 1$ se factorice en polinomios irreducibles distintos en $\mathbb{F}_q$.

3.1. Conceptos básicos

Antes de la definición de código convolucional cíclico se presentan algunos conceptos básicos sobre submódulos y sumandos directos de $\mathbb{F}_q[z]^n$ que se usarán más adelante.

Proposición 3.1.

Sea V un $\mathbb{F}_q[z]$-submódulo de $\mathbb{F}_q[z]^n$. Entonces se satisfacen las siguientes propiedades.

i) V tiene una base finita y toda base de V tiene el mismo cardinal, el cual se denomina rango de V.

ii) Si $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r \in \mathbb{F}_q[z]^n$ forman un conjunto generador de V, entonces

$$V = \operatorname{im} M = \{\boldsymbol{u}M : \boldsymbol{u} \in \mathbb{F}_q[z]^r\}, \text{ donde } M = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_r \end{pmatrix} \in \mathbb{F}_q[z]^{r \times n}.$$

iii) Sean $P \in \mathbb{F}_q[z]^{r \times r}$ y M como en ii) con sus r filas linealmente independientes, esto es de rango r. Entonces $V = \operatorname{im}(PM)$ si y solo si P es invertible sobre $\mathbb{F}_q[z]$.

Demostración.

i) Como $\mathbb{F}_q$ es un campo, entonces $\mathbb{F}_q[z]$ es un dominio de ideales principales y dado que $\mathbb{F}_q[z]^n$ es un $\mathbb{F}_q[z]$-módulo libre entonces, por el Teorema 1.2, el submódulo V de $\mathbb{F}_q[z]^n$ tiene una base.

A continuación, se prueba que toda base de V tiene el mismo número de elementos,esto es igual cardinal. Para ello se verifica que si $S = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ es una base para V y $T = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_r\}$ es un conjunto linealmente independiente de vectores en V, entonces $r \leq n$.

Considere el conjunto $T_1 = \{\boldsymbol{w}_1, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$. Como S genera a V, T_1 también lo genera y puesto que $\boldsymbol{w}_1$ es una combinación lineal de los vectores de S, T_1 es linealmente dependiente. Entonces, algún $\boldsymbol{v}_j$ es una combinación lineal de los vectores que le preceden en T_1.

Luego, $S_1 = T_1 \setminus \{\boldsymbol{v}_j\}$ genera a V y sea $T_2 = \{\boldsymbol{w}_2, \boldsymbol{w}_1, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{j-1}, \boldsymbol{v}_{j+1}, \ldots, \boldsymbol{v}_n\}$, entonces T_2 es linealmente dependiente, luego algún vector en T_2 es una combinación lineal de los vectores precedentes en T_2. Pero, como T es un conjunto linealmente independiente dicho vector no puede ser $\boldsymbol{w}_1$, así que dicho vector es $\boldsymbol{v}_i$ para algún $i \neq j$. Este proceso se repite un número finito de veces. Si se eliminan todos los vectores de S antes que se pueda incluir todos los vectores de T, entonces el conjunto resultante es un subconjunto de T linealmente dependiente, lo cual es una contradicción. Esto significa que en el proceso los elementos de S no se pueden agotar antes de incluir todos los elementos de T; es decir $r \leq n$.

Finalmente, si S y T son bases de V, de lo anterior se tiene que $n = r$.

ii) Sea $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r\} \subseteq \mathbb{F}_q[z]^n$ un conjunto generador de V. Es decir, todo elemento de V puede escribirse como una combinación lineal de elementos en el conjunto $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r\}$ con coeficientes en $\mathbb{F}_q[z]$, luego para $\boldsymbol{x} \in V$ se tiene

$$\boldsymbol{x} = u_1\boldsymbol{v}_1 + u_2\boldsymbol{v}_2 + \cdots + u_r\boldsymbol{v}_r, \text{ con } u_1, u_2, \ldots, u_r \in \mathbb{F}_q[z].$$

Ahora, si se define

$$\boldsymbol{u} = (u_1, u_2, \ldots, u_r) \in \mathbb{F}_q[z]^n \quad \text{y} \quad M = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_r \end{pmatrix} \in \mathbb{F}_q[z]^{r\times n},$$

se tiene que $\boldsymbol{x} = \boldsymbol{u}M$. Por lo tanto, $V = \operatorname{im} M = \{\boldsymbol{u}M : \boldsymbol{u} \in \mathbb{F}_q[z]^r\}$.

iii) Dada una matriz A, con $\rho(A)$ se denota el número de filas (o columnas) linealmente independientes de la matriz A, denominado el rango de A. Supóngase que $P \in \mathbb{F}_q[z]^{r\times r}$ es una matriz tal que $V = \operatorname{im} PM$. Por hipótesis se tiene que $V = \operatorname{im} M$, lo que significa que V es de rango r. De esta manera dado que PM es una matriz de tamaño $r \times n$ que genera a V, entonces PM tiene r filas linealmente independientes, esto es $\rho(PM) = r$.

Puesto que

$$\rho(PM) \leq \text{mín}\{\rho(P), \rho(M)\},$$

y como por hipótesis $\rho(M) = r$ entonces

$$r = \rho(PM) \leq \text{mín}\{\rho(P), r\} \leq \rho(P).$$

Además, P es de tamaño $r \times r$ luego $\rho(P) \leq r$, así $\rho(P) = r$. Por lo tanto, P es invertible.

Recíprocamente, supóngase que P es invertible sobre $\mathbb{F}_q[z]$. Entonces P^{-1} existe. Sea $v \in V$ luego, por el item *ii*) se tiene que

$$v = \boldsymbol{u}M = \boldsymbol{u}P^{-1}PM = \boldsymbol{u'}PM \in \operatorname{im}(PM),$$

con $\boldsymbol{u'} = \boldsymbol{u}P^{-1} \in \mathbb{F}_q[z]^r$; es decir, $V \subseteq \operatorname{im}(PM)$.

Ahora, sea $t \in \operatorname{im}(PM)$ entonces $t = \boldsymbol{u}PM$ para algún $\boldsymbol{u} \in \mathbb{F}_q[z]^r$, luego

$$t = \boldsymbol{u}PM = \boldsymbol{u''}M \in \operatorname{im}(M),$$

con $\boldsymbol{u''} = \boldsymbol{u}P \in \mathbb{F}_q[z]^r$, esto es $\operatorname{im}(PM) \subseteq V$. Esto es, $V = \operatorname{im} PM$. □

La Proposición 3.1 garantiza que todo $\mathbb{F}_q[z]$-submódulo V de $\mathbb{F}_q[z]^n$ es libre y puede escribirse como

$$V = \operatorname{im} G = \{\boldsymbol{u}G : \boldsymbol{u} \in \mathbb{F}_q[z]^k\},$$

donde k es el rango de V y $G \in \mathbb{F}_q[z]^{k\times n}$ es una matriz cuyas filas son una base de V. Cualquier matriz G de este tipo se denomina matriz generadora del módulo V. Además esta es única salvo la multiplicación izquierda por una matriz invertible; esto es para cualquier par de matrices $G, G' \in \mathbb{F}_q[z]^{k\times n}$ de rango k la identidad $\operatorname{im} G = \operatorname{im} G'$ es equivalente a $G' = PG$ para alguna matriz invertible $P \in \mathbb{F}_q[z]^{k\times k}$.

Se puede probar que si V es un sumando directo de $\mathbb{F}_q[z]^n$ de rango k, la forma de Smith de cualquier matriz generadora G de V esta dada por $\Gamma = (I_k \;\; 0_{k\times n-k})$, ver [9, Prop. 2.2].

Proposición 3.2.

Sea V un $\mathbb{F}_q[z]$-submódulo de $\mathbb{F}_q[z]^n$ y $G \in \mathbb{F}_q[z]^{k\times n}$ una matriz generadora de V. Entonces G es invertible a derecha si y solo si V es un sumando directo de $\mathbb{F}_q[z]^n$.

Demostración.

Sea $\widehat{G} \in \mathbb{F}_q[z]^{n\times k}$ la matriz inversa a derecha de G; es decir, tal que $G\widehat{G} = I_k$. Considere la función

$$\begin{array}{rccc} \pi: & \mathbb{F}_q[z]^n & \longrightarrow & V \\ & \boldsymbol{v} & \longrightarrow & \pi(\boldsymbol{v}) = \boldsymbol{v}\widehat{G}G. \end{array}$$

Por la definición de la función π y las propiedades de las operaciones de matrices se tiene que π es $\mathbb{F}_q[z]$-lineal. Ahora, como $V = \operatorname{im} G$ entonces para cada $\boldsymbol{v} \in V$ existe $\boldsymbol{u} \in \mathbb{F}_q[z]^k$ tal que $\boldsymbol{v} = \boldsymbol{u}G$. Luego

$$\pi(\boldsymbol{v}) = (\boldsymbol{u}G)\widehat{G}G = \boldsymbol{u}G = \boldsymbol{v};$$

es decir, π es la identidad en V. Así, por el Lema 1.2 se tiene que V es un sumando directo de $\mathbb{F}_q[z]^n$.

Recíprocamente, se tiene por hipótesis que V es un sumando directo de $\mathbb{F}_q[z]^n$ y G es una matriz generadora, luego la descomposición en factores invariantes de G es $G = A\Gamma B$, donde A y B son matriz invertibles de tamaño $k \times k$ y $n \times n$ respectivamente y $\Gamma = (I_k \;\; 0_{k\times n-k})$.

Sea $G' = B^{-1}\Gamma^T A^{-1}$ y dado que $\Gamma\Gamma^T = I_k$ se tiene que $GG' = (A\Gamma B)(B^{-1}\Gamma^T A^{-1}) = I_k$; esto es, G es invertible a derecha. □

Proposición 3.3.

Sea V un $\mathbb{F}_q[z]$-submódulo sumando directo de $\mathbb{F}_q[z]^n$. Entonces para todo $\boldsymbol{v} \in \mathbb{F}_q[z]^n$ y todo

$\lambda \in \mathbb{F}_q[z]$ no nulo se tiene que

$$\lambda \boldsymbol{v} \in V \text{ implica } \boldsymbol{v} \in V.$$

Demostración.

Sea $G \in \mathbb{F}_q[z]^{k\times n}$ una matriz generadora de V, dado que $\lambda \boldsymbol{v} \in V$ entonces $\lambda \boldsymbol{v} = \boldsymbol{u}G$ para algún $\boldsymbol{u} \in \mathbb{F}_q[z]^k$. Sea $\widehat{G} \in \mathbb{F}_q[z]^{n\times k}$ la matriz inversa a derecha de G y considere $\boldsymbol{u'} = \boldsymbol{v}\widehat{G}$, luego

$$\lambda \boldsymbol{u'} = \lambda \boldsymbol{v}\widehat{G} = \boldsymbol{u}G\widehat{G} = \boldsymbol{u}.$$

Así $\lambda \boldsymbol{u'}G = \boldsymbol{u}G = \lambda \boldsymbol{v}$ entonces $\lambda[\boldsymbol{u'}G - \boldsymbol{v}] = 0$, pero como $\lambda \neq 0$ y $\mathbb{F}_q[z]$ es un dominio entero se tiene que $\boldsymbol{u'}G = \boldsymbol{v}$, esto es $\boldsymbol{v} \in V$. □

Es bien conocido que en la teoría de espacios vectoriales todo subespacio es un sumando directo. Sin embargo, la Proposición 3.2 sugiere que esto no sucede en el caso de los $\mathbb{F}_q[z]$-submódulos del módulo $\mathbb{F}_q[z]^n$. A continuación se proporciona un ejemplo de esta situación.

Ejemplo 3.1.

Sea $\langle z \rangle = \{gz : g \in \mathbb{F}_q[z]\}$ el $\mathbb{F}_q[z]$-submódulo de $\mathbb{F}_q[z]$ generado por z. Supóngase que $\langle z \rangle$ es un sumando directo de $\mathbb{F}_q[z]$. Dado que $z \in \langle z \rangle$ por la Proposición 3.3 se tiene que $1 \in \langle z \rangle$ lo cual es una contradicción.

Proposición 3.4.

Sean V, W $\mathbb{F}_q[z]$-submódulos de $\mathbb{F}_q[z]^n$ que tienen el mismo rango y tales que $W \subseteq V$. Si $W = \operatorname{im} \widehat{G}$ con $\widehat{G}$ invertible a derecha entonces $V = W$.

Demostración.

Sean $G = (g_{ij}), \widehat{G} = (\widehat{g}_{ij}) \in \mathbb{F}_q[z]^{k\times n}$ tales que $V = \operatorname{im} G$ y $W = \operatorname{im} \widehat{G}$. Como $W \subseteq V$, se tiene

$$\operatorname{im} \widehat{G} = \{\boldsymbol{v}\widehat{G} : \boldsymbol{v} \in \mathbb{F}_q[z]^k\} \subseteq \{\boldsymbol{u}G : \boldsymbol{u} \in \mathbb{F}_q[z]^k\} = \operatorname{im} G. \tag{3.1}$$

Sea $\boldsymbol{e}_i \in \mathbb{F}_q[z]^k$ el vector que contiene un 1 en la i-ésima componente y cero en las demás, entonces $\boldsymbol{e}_i\widehat{G} = (\widehat{g}_{i1}, \widehat{g}_{i2}, \ldots, \widehat{g}_{in})$; es decir, la i-ésima fila de la matriz $\widehat{G}$ puede escribirse como $\boldsymbol{e}_i\widehat{G}$. De la ecuación (3.1), se tiene que existe $\boldsymbol{u}_i \in \mathbb{F}_q[z]^k$ tal que $\boldsymbol{e}_i\widehat{G} = \boldsymbol{u}_i G$ para $1 \leq i \leq k$, de ahí que existe una matriz $U \in \mathbb{F}_q[z]^{k\times k}$ tal que $\widehat{G} = UG$.

Ahora, sea $\widehat{G'}$ la inversa a derecha de $\widehat{G}$ entonces $I_k = \widehat{G}\,\widehat{G'} = UG\widehat{G'}$, luego el sistema $\boldsymbol{x}U = 0$ con $\boldsymbol{x} \in \mathbb{F}_q[z]^k$ tiene solución única $\boldsymbol{x} = 0$, esto es U es invertible. Así, por el item *iii*) de la Proposición 3.1 se tiene que $V = \operatorname{im} G = \operatorname{im} UG = \operatorname{im} \widehat{G} = W$. □

En la Sección 2.5 se presentó el concepto de dual de un código convolucional visto como un subespacio vectorial de $\mathbb{F}_q((z))^n$. Este concepto tiene sentido porque en un espacio vectorial cualquier subespacio es un sumando directo. De esta manera, se puede adaptar este concepto cuando se tiene sumando directo de $\mathbb{F}_q[z]^n$; por lo tanto usando las ideas dadas en la Subsección 2.5.1 se prueba el siguiente resultado.

Proposición 3.5.

Sea $V \subseteq \mathbb{F}_q[z]^n$ un submódulo y $G \in \mathbb{F}_q[z]^{k\times n}$ una matriz genradora de V invertible a derecha. Entonces existe una matriz $N \in \mathbb{F}_q[z]^{n\times(n-k)}$ tal que $V = \ker N = \{v \in \mathbb{F}_q[z]^n : vN = \mathbf{0}\}$.

A continuación se presenta la definición formal de un código convolucional en el contexto de $\mathbb{F}_q[z]$-submódulos de $\mathbb{F}_q[z]^n$. En [8, p. 4] se menciona que en [23, 24] se plantea y discute en detalle el problema de si la teoría algebraica de los códigos convolucionales debería desarrollarse desde el contexto de las series de Laurent o en el contexto de los polinomios. Además, en la observación 2.4 de [9, p.5] se menciona la existencia de una correspondencia uno a uno entre códigos convolucionales como subespacios de $\mathbb{F}_q((z))^n$ con matriz generadora polinómica y códigos convolucionales en el sentido de la siguiente definición.

Definición 3.1.

Sea $G \in \mathbb{F}_q[z]^{k\times n}$ una matriz de rango k. Un submódulo $C = \operatorname{im} G \subseteq \mathbb{F}_q[z]^n$ se denomina un código convolucional si la matriz G es invertible a derecha.

En la teoría clásica de los códigos convolucionales como subespacios de $\mathbb{F}_q((z))^n$ una matriz generadora polinomica invertible a derecha se denomina básica o matriz generadora básica, ver [14, p.80]. De la Proposición 3.2 se tiene que un código convolucional es un sumando directo de $\mathbb{F}_q[z]^n$ de rango k. Además, la Proposición 3.5 garantiza que cada código convolucional $C \subseteq \mathbb{F}_q[z]^n$ tiene una matriz de control de paridad; es decir, existe una matriz $H \in \mathbb{F}_q[z]^{n\times(n-k)}$ tal que $C = \ker H = \{\boldsymbol{v} \in \mathbb{F}_q[z]^n : \boldsymbol{v}H = 0\}$.

A continuación se presentan las definiciones de complejidad de un código convolucional, grado de un vector y grado fila de una matriz, conceptos necesarios para definir matriz

generadora minimal. En [8, 9] los autores aclaran que no se ha establecido una notación estándar para el parámetro llamado complejidad del código, por ejemplo en [14, p.83] se denomina longitud de restricción general[1].

Antes de presentar dichas definiciones cabe mencionar, que un *k-menor* de una matriz G es un determinante de tamaño $k \times k$ de la matriz G.

Definición 3.2.

Sean $\mathbb{F}_q$ un campo finito y $G \in \mathbb{F}_q[z]^{k\times n}$ una matriz de rango k. El número

$$\delta = \max\{\text{gr } \gamma : \gamma \text{ es un } k\text{-menor de } G\}, \tag{3.2}$$

se denomina la complejidad de la matriz G o del submódulo $\operatorname{im} G$.

Definición 3.3.

Sea G una matriz generadora de un código convolucional, entonces la complejidad del código es igual a la complejidad de la matriz G.

Naturalmente, un código convolucional de complejidad cero es un código de bloque, porque esto significa que las entradas de su matriz generadora son polinomios constantes; esto es, son elementos de $\mathbb{F}_q$.

Definición 3.4.

1. *Si $v = \sum_{j=0}^{N} \boldsymbol{v}_j z^j \in \mathbb{F}_q[z]^n$ donde $\boldsymbol{v}_j \in \mathbb{F}_q^n$, entonces el grado de v es N y se denota* $\text{gr } v = N$. *Además,* $\text{gr } 0 = -\infty$.

2. *Sea $G = (g_{ij}) \in \mathbb{F}_q[z]^{k\times n}$ entonces el grado de la i-ésima fila de G se denota ν_i y se define por* $\nu_i = \max_{1\le j\le n}\{\text{gr } g_{ij}\}$.

En [8] los autores afirman que en [5, Tma. 1] se demuestra que cada submódulo de $\mathbb{F}_q[z]^n$ tiene una matriz generadora minimal en el sentido de la siguiente definición.

Definición 3.5.

Sea $G \in \mathbb{F}_q[z]^{k\times n}$ una matriz de rango k y complejidad δ y sean $\nu_1, \nu_2, \ldots, \nu_k$ los grados de las filas de G, se dice que G es minimal si $\delta = \sum_{i=1}^{k} \nu_i$.

[1] En inglés, overall constraint length.

El concepto más importante para un código es su distancia, pues esta sirve para determinar su capacidad de corrección y detección de errores, por lo tanto, la calidad del código. La definición de distancia libre de un código convolucional se presenta a continuación.

Definición 3.6.

1. *El peso de la palabra* $v = \sum_{i=0}^{N} \boldsymbol{v}_i z^i \in \mathbb{F}_q[z]^n$*, se define como*

$$wt(v) = \sum_{i=0}^{N} wt(\boldsymbol{v}_i) \in \mathbb{N}_0,$$

   donde $wt(\boldsymbol{v}_i)$ *denota el peso de Hamming del vector constante* $\boldsymbol{v}_i \in \mathbb{F}_q^n$*.*

2. *La distancia libre de un código* $C \subseteq \mathbb{F}_q[z]^n$ *se define como*

$$d_{libre}(C) = \text{mín}\{wt(v) : v \in C \setminus \{0\}\}.$$

3.2. Ciclicidad

A continuación se expone la noción de ciclicidad en los códigos convolucionales, para esto se inicia recordando este concepto en códigos de bloque cíclicos. Un código de bloque $C \subseteq \mathbb{F}_q^n$ se dice cíclico si este es invariante bajo desplazamientos cíclicos, es decir, si

$$(v_0, v_1, \ldots, v_{n-1}) \in C \quad \text{entonces} \quad (v_{n-1}, v_0, \ldots, v_{n-2}) \in C, \tag{3.3}$$

o equivalentemente, $CS \subseteq C$ donde

$$S = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{F}_q^{n \times n}. \tag{3.4}$$

Una herramienta importante en la teoría de códigos de bloque cíclicos es la llamada representación polinomial. Esta se basa en el $\mathbb{F}_q$-isomorfismo, dado por

$$\begin{array}{rccc} \phi: & \mathbb{F}_q^n & \longrightarrow & R_n \\ & \boldsymbol{v} = (v_0, \ldots, v_{n-1}) & \longrightarrow & \phi(\boldsymbol{v}) = \sum_{i=0}^{n-1} v_i x^i \end{array},$$

donde $R_n = \mathbb{F}_q[x]/\langle x^n - 1\rangle$ es el anillo cociente definido por, $R_n = \{f \in \mathbb{F}_q[x] : \text{gr}(f) < n\}$ con multiplicación módulo $x^n - 1$.

La función ϕ interpreta el desplazamiento cíclico como la multiplicación por x. Como consecuencia, un código de bloque cíclico C puede ahora presentarse como un ideal $\phi(C)$ en R_n y viceversa, en otras palabras, un código de bloque C es cíclico si y solo si

$$a \in \phi(C) \qquad \text{implica} \qquad xa \in \phi(C).$$

Esto significa que estudiar códigos de bloque cíclicos en $\mathbb{F}_q^n$ es equivalente a estudiar ideales en R_n. Además, dado que R_n es un dominio de ideales principales, esto representa ciertas ventajas a la hora de estudiar la estructura de los códigos de bloque cíclicos, ver [11, Cap. 4], en particular cada código de bloque cíclico tiene un polinomio generador.

Para extender este concepto de los códigos de bloque cíclicos al contexto de códigos convolucionales se considera la siguiente función

$$\begin{array}{rccc} \varphi: & \mathbb{F}_q[z]^n & \longrightarrow & R_n[z] \\ & \sum_{j=0}^{M-1} \boldsymbol{v}_j z^j & \longrightarrow & \sum_{j=0}^{M-1} \phi(\boldsymbol{v}_j) z^j, \end{array} \tag{3.5}$$

donde $\boldsymbol{v}_j \in \mathbb{F}_q^n$ y $\phi(\boldsymbol{v}_j) \in R_n$ para todo j. Cabe observar que usando la notación del Capítulo 2 esta función puede expresarse de la siguiente manera

$$\begin{array}{rccc} \varphi: & \mathbb{F}_q[z]^n & \longrightarrow & R_n[z] \\ & \left(\sum_{i=0}^{M-1} a_{ij} z^i\right)_{j=0}^{n-1} & \longrightarrow & \sum_{i=0}^{M-1} \left(\sum_{j=0}^{n-1} a_{ij} x^j\right) z^i, \end{array} \tag{3.6}$$

donde $M-1$ es el máximo grado de los polinomios del vector de entrada.

Teorema 3.1.

La función φ dada por (3.5) es un $\mathbb{F}_q[z]$-isomorfismo de $\mathbb{F}_q[z]$-módulos.

Demostración.

Sean $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_q[z]^n$ y $r \in \mathbb{F}_q[z]$ con

$$\boldsymbol{u} = \sum_{j=0}^{M-1} \boldsymbol{u}_j z^j, \;\; \boldsymbol{v} = \sum_{i=0}^{N-1} \boldsymbol{v}_i z^i \;\text{ y }\; r = \sum_{t=0}^{h} r_t z^t.$$

Supóngase que $M-1 < N-1$ y asúmase que $\boldsymbol{u}_j = 0$ para $M \leq j \leq N-1$. Dado que ϕ es

un $\mathbb{F}_q$-isomorfismo se tiene

$$\begin{aligned}\varphi(\boldsymbol{u}+\boldsymbol{v}) &= \varphi\left(\sum_{i=0}^{N-1}(\boldsymbol{u}_i+\boldsymbol{v}_i)z^i\right) = \sum_{i=0}^{N-1}\phi(\boldsymbol{u}_i+\boldsymbol{v}_i)z^i = \sum_{i=0}^{N-1}[\phi(\boldsymbol{u}_i)+\phi(\boldsymbol{v}_i)]z^i \\ &= \sum_{i=0}^{N-1}\phi(\boldsymbol{u}_i)z^i + \sum_{i=0}^{N-1}\phi(\boldsymbol{v}_i)z^i = \varphi(\boldsymbol{u})+\varphi(\boldsymbol{v}).\end{aligned}$$

Ahora,

$$\begin{aligned}\varphi(r\boldsymbol{u}) &= \varphi\left(\left[\sum_{t=0}^{h} r_t z^t\right]\left[\sum_{j=0}^{M-1}\boldsymbol{u}_j z^j\right]\right) = \varphi\left(\sum_{k=0}^{h+M-1}\left[\sum_{p=0}^{k} r_p \boldsymbol{u}_{k-p}\right]z^k\right) \\ &= \sum_{k=0}^{h+M-1}\phi\left(\sum_{p=0}^{k} r_p\boldsymbol{u}_{k-p}\right)z^k = \sum_{k=0}^{h+M-1}\sum_{p=0}^{k} r_p\phi\left(\boldsymbol{u}_{k-p}\right)z^k \\ &= \left[\sum_{t=0}^{h} r_t z^t\right]\left[\sum_{j=0}^{M-1}\phi(\boldsymbol{u}_j)z^j\right] = r\varphi(\boldsymbol{u}).\end{aligned}$$

Además, si $\boldsymbol{u} \neq \boldsymbol{v}$ por la definición de igualdad de polinomios y de la función φ, se tiene que $\varphi(\boldsymbol{u}) \neq \varphi(\boldsymbol{v})$. Finalmente, si $T \in R_n[z]$ entonces

$$T = \sum_{i=0}^{m}\left(\sum_{j=0}^{n-1} a_{ij}x^j\right)z^i = \sum_{i=0}^{m}\phi(\boldsymbol{a}_i)z^i,$$

con $\boldsymbol{a}_i = (a_{i0}, a_{i1}, \ldots, a_{i,n-1}) \in \mathbb{F}_q^n$, esto es existe $\boldsymbol{x} = \sum_{i=0}^{m}\boldsymbol{a}_i z^i \in \mathbb{F}_q[z]^n$ tal que $\varphi(\boldsymbol{x}) = T$. □

En este sentido, sería muy natural definir códigos convolucionales cíclicos justo como se hace para códigos de bloque cíclicos, esto es que se cumpla la expresión (3.3) o dicho de otra manera que si C es un código convolucional se denomina cíclico si satisface que

$$\boldsymbol{v} \in \varphi(C) \text{ implica } x\boldsymbol{v} \in \varphi(C).$$

Sin embargo, P. Piret en [19] demostró que los códigos convolucionales que satisfacen esta condición son nuevamente códigos de bloque cíclicos y por tanto este análisis no conduciría al descubrimiento de nuevos códigos convolucionales. A continuación se presenta una demostración, basada en [9], de este resultado.

Proposición 3.6.

Sea $\mathcal{C} \subseteq \mathbb{F}_q[z]^n$ un código convolucional que satisface (3.3). Entonces $\mathcal{C}$ es un código de bloque.

Demostración.

Por hipótesis, $\mathcal{C}S \subseteq \mathcal{C}$ siendo S como en (3.4). Sea $A = xI_n - S$, luego para determinar el polinomio característico de S, $p_S(x)$, se calcula $\det(A)$, esto es

$$\det(A) = \det(xI_n - S) = \det \begin{pmatrix} x & -1 & 0 & 0 & \cdots & 0 \\ 0 & x & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & 0 & \cdots & x \end{pmatrix}_{n\times n}.$$

Realizando el cálculo de este determinante mediante el desarrollo de cofactores por la primera columna, se tiene

$$p_S(x) = x \det \begin{pmatrix} x & -1 & \cdots & 0 \\ 0 & x & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x \end{pmatrix}_{(n-1)\times(n-1)} + (-1)^{n+1}(-1) \det \begin{pmatrix} -1 & 0 & \cdots & 0 \\ x & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{pmatrix}_{(n-1)\times(n-1)},$$

de ahí que, $p_S(x) = x(x^{n-1}) + (-1)^{n+2}(-1)^{n-1} = x^n + (-1)^{2n+1}$. Así, el polinomio característico de S es $x^n - 1$; además el determinante del menor $(n, 1)$ de la matriz A es $\det(M_{n1}) = (-1)^{n-1}$, luego sin importar cual sea el determinante de los demás menores de orden $n - 1$ se tiene que el máximo común divisor de todos los menores de orden $n - 1$ de la matriz característica A es 1. Por lo tanto, el polinomio minimal de S es $x^n - 1$, ver [7, p. 90].

Sea $x^n - 1 = \pi_1 \cdots \pi_r$ la factorización en polinomios irreducibles diferentes sobre $\mathbb{F}_q$. Luego, por el Teorema 1.3, se tiene la descomposición

$$\mathbb{F}_q[z]^n = \ker \pi_1(S) \oplus \cdots \oplus \ker \pi_r(S),$$

en submódulos los cuales son sumandos directos minimales S-invariantes, ver Definición 1.15. Dado que C es un sumando directo, también se tiene

$$C = \oplus_{i\in T} \ker \pi_i(S), \text{ donde } T = \{i : \ker \pi_i(S) \cap C \neq \{0\}\}.$$

Supóngase que $\pi_i(x) = b_0 + b_1x + \cdots + b_tx^t$, por definición $\pi_i(S) = b_0I_n + b_1S + \cdots + b_tS^t$ luego para $\boldsymbol{v} = \sum_{j=0}^{M-1} \boldsymbol{a}_j z^j \in \ker \pi_i(S)$ para algún $i \in T$

$$\pi_i(S)(\boldsymbol{v}) = \boldsymbol{v}(b_0I_n + b_1S + \cdots + b_tS^t) = b_0\boldsymbol{v}I_n + b_1\boldsymbol{v}S + \cdots + b_t\boldsymbol{v}S^t = 0.$$

De lo cual se tiene

$$\begin{aligned}0 &= b_0(\boldsymbol{a}_0 + \boldsymbol{a}_1 z + \cdots + \boldsymbol{a}_{M-1} z^{M-1}) I_n + b_1(\boldsymbol{a}_0 + \boldsymbol{a}_1 z + \cdots + \boldsymbol{a}_{M-1} z^{M-1}) S + \cdots + \\ &\quad b_t(\boldsymbol{a}_0 + \boldsymbol{a}_1 z + \cdots + \boldsymbol{a}_{M-1} z^{M-1}) S^t \\ &= (b_0 \boldsymbol{a}_0 I_n + b_1 \boldsymbol{a}_0 S + \cdots + b_t \boldsymbol{a}_0 S^t) + (b_0 \boldsymbol{a}_1 I_n + b_1 \boldsymbol{a}_1 S + \cdots + b_t \boldsymbol{a}_1 S^t) z + \cdots + \\ &\quad (b_0 \boldsymbol{a}_{M-1} I_n + b_1 \boldsymbol{a}_{M-1} S + \cdots + b_t \boldsymbol{a}_{M-1} S^t) z^{M-1}.\end{aligned}$$

Lo que implica $(b_0 \boldsymbol{a}_j I_n + b_1 \boldsymbol{a}_j S + \cdots + b_t \boldsymbol{a}_j S^t) = 0$ para $0 \leq j \leq M-1$, esto si y solo si $\pi_i(S)(\boldsymbol{a}_j) = 0$ para $0 \leq j \leq M-1$ o equivalentemente $\boldsymbol{a}_j \in \ker \pi_i(S)$ para $0 \leq j \leq M-1$.

De ahí que, $\ker \pi_i(S)$ es generado como $\mathbb{F}_q[z]$-módulo por vectores constantes y dado que $\mathbb{F}_q^n$ es S invariante, se tiene que el $\mathbb{F}_q[z]$-módulo $\ker \pi_i(S)$ es generado por $\ker \pi_i(S) \cap \mathbb{F}_q^n$. Entonces C tiene una base en $\mathbb{F}_q^n$.

Esto conduce directamente a una matriz generadora constante y por tanto a un codificador constante para C. Esto significa que la complejidad es cero, y como se explica después de la Definición 3.3, esto implica que C es un código de bloque. □

Como se menciona en los artículos [8, 9] el resultado presentado en la Proposición 3.6 llevó a Piret en [19] a presentar una noción más general y compleja de ciclicidad para códigos convolucionales. En lugar de invarianza bajo desplazamientos cíclicos de C mediante la matriz de desplazamiento S dada en (3.4), dada por

$$\sum_{i=0}^{M-1} \boldsymbol{v}_i z^i \in C, \quad \text{implica} \quad \sum_{i=0}^{M-1} \boldsymbol{v}_i S z^i \in C,$$

Piret introduce una especie de grado de cuasi-ciclicidad. Precisamente él llamó a C un código convolucional cíclico, si existe algún m, el cual es coprimo con la longitud del código n, tal que

$$\sum_{i=0}^{M-1} \boldsymbol{v}_i z^i \in C, \quad \text{implica} \quad \sum_{i=0}^{M-1} \boldsymbol{v}_i S^{(m^i)} z^i \in C. \tag{3.7}$$

En el lenguaje polinomial, es decir en el anillo de polinomios $R_n[z]$, esto se interpreta como sigue

$$\sum_{i=0}^{M-1} \phi(\boldsymbol{v}_i) z^i \in \varphi(C), \quad \text{implica} \quad \sum_{i=0}^{M-1} \phi(\boldsymbol{v}_i) x^{(m^i)} z^i \in \varphi(C). \tag{3.8}$$

La siguiente proposición permite introducir una estructura de $\mathbb{F}_q$-álgebra en el $\mathbb{F}_q[z]$-

módulo izquierdo $R_n[z]$ que se extiende del álgebra $R_n[z]$.

Proposición 3.7.

Si m y n son coprimos, entonces la función $h : R_n \longrightarrow R_n$, definida por $h(x) = x^m$ es un $\mathbb{F}_q$-automorfismo en R_n.

Demostración.

Sean $a, b \in \mathbb{F}_q$ y $p(x) = \sum_{i=0}^{n-1} a_i x^i$, $q(x) = \sum_{i=0}^{n-1} b_i x^i, \in R_n$ entonces

$$\begin{aligned} h(ap(x) + bq(x)) =& h\left(\sum_{i=0}^{n-1}(aa_i + bb_i)x^i\right) = \sum_{i=0}^{n-1}(aa_i + bb_i)(x^m)^i \\ =& a\sum_{i=0}^{n-1} a_i(x^m)^i + b\sum_{i=0}^{n-1} a_i(x^m)^i \\ =& ah(p(x)) + bh(q(x)). \end{aligned}$$

Esto es h es un homomorfismo de $\mathbb{F}_q$-módulos. Además, por las definiciones de igualdad de polinomios y de la función se tiene que h es inyectiva. Para la sobreyectividad; dado que existen t y k en $\mathbb{Z}$ tales que $1 = mt + nk$, se sigue que

$$x = x^{mt+nk} = x^{mt}x^{nk} = x^{mt}(1) = x^{mt} = h(x^t),$$

esto es, la preimagen de x en R_n es x^t. De ahí que, si $p(x) = \sum_{i=0}^{n-1} a_i x^i \in R_n$ su preimagen es $g(x) = \sum_{i=0}^{n-1} a_i x^{ti}$.

Por lo tanto, la función h es un $\mathbb{F}_q$-automorfismo en R_n. □

En este sentido se puede ver, a partir de la ecuación (3.8) y la Proposición 3.7, que el concepto de ciclicidad de Piret es un caso particular de aquel dado por Roos en [21], el cual es una extensión natural para un $\mathbb{F}_q$-automorfismo σ arbitrario de R_n.

Cabe notar que en este documento cambia la escritura de los elementos en $\mathbb{F}_q[z]^n$ es diferente a la que es utilizada en los artículos [8, 9], pues en este trabajo los coeficientes de los polinomios en $\mathbb{F}_q[z]^n$ se escriben siempre a la izquierda de la variable z, como se puede observar por ejemplo en la ecuación (3.7) y de igual manera los elementos en $R_n[z]$ se escribirán con coeficientes a izquierda como se evidencia en la ecuación (3.8).

Antes de mostrar la generalización de Roos, se presenta el anillo de polinomios torcido

$R_n[z;\sigma]$ el cual es isomorfo a $\mathbb{F}_q[z]^n$ como $\mathbb{F}_q[z]$-módulo derecho y algunas de sus características.

Para hacerlo, tenga en cuenta que $\mathbb{F}_q$ puede considerarse como un subcampo del anillo R_n y en consecuencia R_n es una $\mathbb{F}_q$-álgebra. En lo que sigue se verá que el grupo de automorfismos $\mathrm{Aut}_{\mathbb{F}_q}(R_n)$ del $\mathbb{F}_q$-álgebra R_n juega un papel importante para la definición de códigos convolucionales. Además cada automorfismo $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$ está únicamente determinado por el valor de $\sigma(x) \in R_n$. En particular, $\sigma(x) = x$ es la identidad en R_n. Pero cabe aclarar que no todas las opciones para $\sigma(x)$ determinan un $\mathbb{F}_q$-automorfismo en R_n; en este sentido la Proposición 3.7 se constituye en una herramienta para construir automorfismos de R_n. Por otro lado, en [9, Cor. 3.2] se presenta una formula para calcular el cardinal de $\mathrm{Aut}_{\mathbb{F}_q}(R_n)$.

En [8] los autores comentan que la idea principal de Piret fue imponer una nueva estructura de anillo a $R_n[z]$ y llamar a un código convolucional cíclico si es un ideal derecho con respecto a la nueva estructura definida. Esta nueva estructura es no conmutativa y se basa en un automorfismo arbitrariamente elegido en R_n. Esta nueva estructura se muestra a continuación.

Definición 3.7.

Sea $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$. Se define el producto de $A = \sum_{i=0}^{N} a_i z^i$ y $B = \sum_{j=0}^{M} b_j z^j \in R_n[z]$ por

$$A *_\sigma B = \left(\sum_{i=0}^{N} a_i z^i\right) *_\sigma \left(\sum_{j=0}^{M} b_j z^j\right) = \sum_{k=0}^{N+M} \left(\sum_{i+j=k} a_i \sigma^i(b_j)\right) z^k,$$

para todo $N, M \in \mathbb{N}_0$ y $a_i, b_j \in R_n$.

El conjunto $R_n[z]$ dotado con la suma usual de polinomios, la multiplicación $*_\sigma$ y la multiplicación clásica para los coeficientes en el anillo cociente R_n, es un anillo de polinomios torcido el cual se denota por $R_n[z;\sigma]$. El anillo $R_n[z;\sigma]$ también se denomina álgebra de Piret con parámetros q, n, σ.

Observe que la multiplicación en $R_n[z;\sigma]$ es simplemente una extensión de la multiplicación conmutativa en R_n junto con la siguiente regla

$$z *_\sigma a = \sigma(a) z \text{ para todo } a \in R_n. \tag{3.9}$$

En particular, se tiene

$$z *_\sigma \lambda = \lambda z \text{ para todo } \lambda \in \mathbb{F}_q.$$

Por tanto, el producto habitual siempre que el factor de la izquierda esté en $\mathbb{F}_q[z]$; es precisamente

$$A *_\sigma B = \left(\sum_{i=0}^{N} a_i z^i\right) *_\sigma \left(\sum_{i=0}^{M} b_i z^i\right) = \sum_{k=0}^{N+M} \left(\sum_{k=i+j} a_i b_j\right) z^k,$$

para todo $A \in R_n[z]$ y $B \in \mathbb{F}_q[z]$.

Note que siempre se ponen los coeficientes de z a su izquierda. Esto es, por supuesto una cuestión de elección, pero la forma explícita de que el producto $*_\sigma$ no es conmutativo depende de ello. Usando la regla de multiplicación (3.9) los coeficientes siempre se pueden desplazar a la derecha de z si se aplica una potencia adecuada de σ^{-1}. Además, es claro que la indeterminada z no conmuta con los coeficientes de R_n a menos que σ sea la identidad, lo cual evidencia la importancia de diferenciar entre coeficientes izquierdos y derechos de z.

Dado que la multiplicación en el anillo R_n sigue siendo la usual, se tiene que R_n es un subanillo conmutativo de $R_n[z;\sigma]$. Además dado que $\sigma|_{\mathbb{F}_q} = id_{\mathbb{F}_q}$ se sigue que el anillo de polinomios $\mathbb{F}_q[z]$ es un subanillo conmutativo del anillo torcido $R_n[z;\sigma]$; como consecuencia $R_n[z;\sigma]$ tiene estructura de $\mathbb{F}_q[z]$-módulo izquierdo y derecho.

Como la nueva estructura de anillo en R_n está determina por el automorfismo σ, Gluesing-Luerssen y Schmale en [9] proponen el nombre de σ-ciclicidad para la estructura cíclica de los códigos convolucionales. En la siguiente definición se introduce esta noción para submódulos arbitrarios de $\mathbb{F}_q[z]^n$.

Definición 3.8.

Sean q y n primos relativos y $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$. Un submódulo C de $\mathbb{F}_q[z]^n$ se dice σ-cíclico si

$$g = \sum_{i=0}^{d} g_i z^i \in \varphi(C) \quad \textit{implica} \quad g *_\sigma x = \sum_{i=0}^{d} g_i \sigma^i(x) z^i \in \varphi(C). \tag{3.10}$$

En [9] los autores mencionan que, Roos en [21] extendió la ecuación (3.10) a una estructura de $\mathbb{F}_q[z]$-módulo derecho en $R_n[z]$, hecho que se utilizó para investigar en detalle la estructura de código convolucional σ-cíclico. Desafortunadamente, en este contexto no es posible construir polinomios generadores. Parece ser más útil usar $*_\sigma$ para una estructura de anillo no conmutativo como el anillo $R_n[z;\sigma]$ como se muestra a continuación.

Definición 3.9.

Sea $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$. Considere la función $\varphi : \mathbb{F}_q[z]^n \to R_n[z;\sigma]$ como en (3.5), donde ahora las imágenes $\varphi(v) = \sum_{j=0}^{M-1} \phi(\boldsymbol{v}_j) z^j$ son vistas como elementos de $R_n[z;\sigma]$. Un submódulo $S \subseteq \mathbb{F}_q[z]^n$ se dice σ-cíclico si $\varphi(S)$ es un ideal derecho de $R_n[z;\sigma]$. Un código convolucional $C \subseteq \mathbb{F}_q[z]^n$ se dice σ-cíclico si C es un sumando directo de $\mathbb{F}_q[z]^n$ y es un submódulo σ-cíclico.

Note que la función $\varphi : \mathbb{F}_q[z]^n \to R_n[z;\sigma]$ dada en la Definición 3.9 es un isomorfismo de $\mathbb{F}_q[z]$-módulos derechos. Sin embargo, φ no es un isomorfismo de $\mathbb{F}_q[z]$-módulos izquierdos. En efecto, sea $a \in R_n[z;\sigma]$ y $r = z \in \mathbb{F}_q[z]$ entonces

$$\varphi(ra) = \varphi\left(z \sum_{j=0}^{N} \boldsymbol{v}_j z^j\right) = \varphi\left(\sum_{j=0}^{N} z\boldsymbol{v}_j z^j\right) = \varphi\left(\sum_{j=0}^{N} \sigma(\boldsymbol{v}_j) z^{j+1}\right) = \sum_{j=0}^{N} \phi(\sigma(\boldsymbol{v}_j)) z^{j+1}.$$

Pero,

$$r\varphi(a) = z\varphi\left(\sum_{j=0}^{N} \boldsymbol{v}_j z^j\right) = z\left(\sum_{j=0}^{N} \phi(\boldsymbol{v}_j) z^j\right) = \sum_{j=0}^{N} z\phi(\boldsymbol{v}_j) z^j = \sum_{j=0}^{N} \sigma(\phi(\boldsymbol{v}_j)) z^{j+1}.$$

Así $\varphi(ra) \neq r\varphi(a)$ para $\sigma \neq id_{R_n}$.

Es importante tener en cuenta que con base en la Definición 3.9, un submódulo C de $\mathbb{F}_q[z]^n$ es σ-cíclico si y solo si su versión polinomial $\varphi(C)$ es un ideal derecho en $R_n[z;\sigma]$. Los códigos convolucionales σ-cíclicos son los $R_n[z;\sigma]$-submódulos de $R_n[z;\sigma]$ que son sumandos directos del $\mathbb{F}_q[z]$-módulo derecho $R_n[z;\sigma]$. Como se verá esto implica que son sumandos directos como $R_n[z;\sigma]$-módulos. En otra palabras, cada código σ-cíclico tiene un complemento que es σ-cíclico.

Ejemplo 3.2. Sea $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ y $n = 3$.

1. Considere el automorfismo σ dado por $\sigma(x) = \alpha^2 x$. Se quiere encontrar el código convolucional σ-cíclico más pequeño que contiene la palabra código

$$\boldsymbol{v} = (1 + z + z^2, \alpha + z + \alpha^2 z^2, \alpha^2 + z + \alpha z^2) \in \mathbb{F}_4[z]^3.$$

Primero que todo, $\varphi(C)$ debe contener el ideal en $R_n[z;\sigma]$ generado por el polinomio

$$g = \varphi(\boldsymbol{v}) = 1 + \alpha x + \alpha x^2 + (1 + x + x^2)\alpha^2 z + (\alpha^2 + \alpha x + x^2) z^2.$$

Un cálculo

$$x *_\sigma g = \alpha^2 + x + \alpha x^2 + (1 + x + x^2)\alpha^2 z + (\alpha^2 + \alpha x + x^2)z^2 = \alpha^2 g.$$

Así $x^2 *_\sigma g = \alpha g$. Además, se puede verificar que la matriz

$$G = \begin{pmatrix} 1 + z + z^2 & \alpha + z + \alpha^2 z^2 & \alpha^2 + z + \alpha z^2 \end{pmatrix}$$

es básica y por tanto $C = \operatorname{im} G$ es el código convolucional σ-cíclico más pequeño que contiene la palabra v. Este código pasa a ser bastante bueno, dado que se puede probar que $d_{free}(C) = 9$, que es el valor máximo de la distancia libre de cualquier código unidimensional de longitud 3 y complejidad 2, ver [22, Teorema 2.2]. Por tanto, C es un MDS-código en el sentido de [22, Def. 2.5].

2. Considérese también la situación en 1, con el automorfismo $\sigma = id$. En este caso la multiplicación por x se corresponde simplemente con el desplazamiento cíclico habitual y por lo tanto, el código convolucional σ-cíclico más pequeño C' que contiene la palabra $\boldsymbol{v}$ tiene que satisfacer $\operatorname{im} G' \subseteq C'$, donde

$$G' = \begin{pmatrix} 1 + z + z^2 & \alpha + z + \alpha^2 z^2 & \alpha^2 + z + \alpha z^2 \\ \alpha^2 + z + \alpha z^2 & 1 + z + z^2 & \alpha + z + \alpha^2 z^2 \\ \alpha + z + \alpha^2 z^2 & \alpha^2 + z + \alpha z^2 & 1 + z + z^2 \end{pmatrix}.$$

Dado que $\det G' \neq 0$, el código C' es 3-dimensional y por la Proposición 3.4, se tiene

$$C' = \operatorname{im} I_3 = \mathbb{F}_4[z]^3.$$

Por lo tanto, C' es un código de bloque y así este es un ejemplo del resultado en la Proposición 3.6.

3.3. Algunos cálculos con SAGE para códigos convolucionales cíclicos

En esta sección se exhibe una manera de usar el programa computacional `SAGE` para realizar implementaciones que son útiles para determinar una matriz generadora de un código convolucional σ-cíclico.

Ejemplo 3.3.

Sean $R_7 = \mathbb{F}_2[x]/\langle x^7 - 1\rangle$ y $\sigma(x) = x^5$ un elemento en $\mathrm{Aut}_{\mathbb{F}_2}(R_7)$. Dado

$$g = 1 + x^2 + x^3 + x^4 + (x + x^2 + x^3 + x^5)z + (1 + x + x^4 + x^6)z^2 \in R_7[z;\sigma], \tag{3.11}$$

entonces $\langle g\rangle = \{gf : f \in R_7[z;\sigma]\}$ es el ideal derecho generado por g en $R_7[z;\sigma]$. Además, para $C = \varphi^{-1}(\langle g\rangle) \subseteq \mathbb{F}_2[z]^7$ se prueba que C es un sumando directo de $\mathbb{F}_2[z]^7$, así C es un código convolucional σ-cíclico.

Sea $f = \sum_{i=0}^{N} f_i z^i \in R_7[z;\sigma]$. Como $f_i \in R_7$, entonces $f_i = a_{i0} + a_{i1}x + \cdots + a_{i6}x^6$ con $a_{ij} \in \mathbb{F}_2$ para $0 \le i \le N$ y $0 \le j \le 6$, luego

$$\begin{aligned}
gf &= g\sum_{i=0}^{N} f_i z^i = \sum_{i=0}^{N} g f_i z^i = \sum_{i=0}^{N} g(a_{i0} + a_{i0}x + \cdots + a_{i6}x^6)z^i \\
&= \sum_{i=0}^{N} (ga_{i0} + ga_{i0}x + \cdots + ga_{i6}x^6)z^i = \sum_{i=0}^{N} (ga_{i0} + gxa_{i0} + \cdots + gx^6 a_{i6})z^i \\
&= \sum_{i=0}^{N} ga_{i0}z^i + \sum_{i=0}^{N} gxa_{i0}z^i + \cdots + \sum_{i=0}^{N} gx^6 a_{i6}z^i \\
&= g\sum_{i=0}^{N} a_{i0}z^i + gx\sum_{i=0}^{N} a_{i0}z^i + \cdots + gx^6\sum_{i=0}^{N} a_{i6}z^i,
\end{aligned}$$

con $\sum_{i=0}^{N} a_{ij}z^i \in \mathbb{F}_2[z]$ para $0 \le j \le 6$. Por lo tanto

$$\langle g\rangle = \mathrm{gen}_{\mathbb{F}_2[z]}\{g, gx, \ldots, gx^6\}. \tag{3.12}$$

Luego, de la ecuación (3.12) y por medio del isomorfismo φ dado en (3.5) se tiene que,

$$C = \{uG : u \in \mathbb{F}_2[z]^7\}, \text{ donde } G = \begin{pmatrix} \varphi^{-1}(g) \\ \varphi^{-1}(gx) \\ \vdots \\ \varphi^{-1}(gx^6) \end{pmatrix} \in \mathbb{F}_2[z]^{7\times 7}. \tag{3.13}$$

Ahora, para calcular gx^j para $0 \le j \le 6$ se debe usar la regla de multiplicación dada en la ecuación (3.9) de la siguiente manera

$$\begin{aligned}
gx &= (x + x^3 + x^4 + x^5) + (1 + x + x^3 + x^6)z + (x + x^3 + x^4 + x^5)z^2, \\
gx^2 &= (x^2 + x^4 + x^5 + x^6) + (x + x^4 + x^5 + x^6)z + (1 + x + x^2 + x^5)z^2.
\end{aligned}$$

Los demás productos se obtienen como combinación lineal de los tres primeros elementos

como sigue

$$gx^3 = g + gx^2, \quad gx^4 = g + gx + gx^2, \quad gx^5 = g + gx, \quad gx^6 = gx + gx^2.$$

De ahí que $\langle g \rangle = \text{gen}_{\mathbb{F}_2[z]}\{g, gx, gx^2\}$ y dado que φ^{-1} es un isomorfismo se tiene que $C = \{uG : u \in \mathbb{F}_2[z]^3\}$, donde

$$G = \begin{pmatrix} \varphi^{-1}(g) \\ \varphi^{-1}(gx) \\ \varphi^{-1}(gx^2) \end{pmatrix} = \begin{pmatrix} 1+z^2 & z+z^2 & 1+z & 1+z & 1+z^2 & z & z^2 \\ z & 1+z+z^2 & 0 & 1+z+z^2 & 1+z^2 & 1+z^2 & z \\ z^2 & z+z^2 & 1+z^2 & 0 & 1+z & 1+z+z^2 & 1+z \end{pmatrix}.$$

Se puede verificar que G es invertible a derecha y minimal. Por lo tanto, $C \subset \mathbb{F}_2[z]^7$ es un código convolucional σ-cíclico.

Por supuesto, como se ve en el ejemplo anterior, los cálculos hechos generalmente son enredados de completar manualmente. De esta manera, en el apéndice se proponen los algoritmos: Imagen inversa de la función φ A.12, ver p. 105, Grado fila A.11, ver p. 105, y Complejidad A.10, ver p. 104, los cuales ayudan a simplificar estos procedimientos. De esta manera, los valores anteriores se pueden calcular de la siguiente manera.

Primero en SAGE se definen los anillos sobre los cuales se realizan las cuentas así, para construir el anillo de polinomios $\mathbb{F}_2[z]$ se define por `F.<x>=GF(2)[]`, para el anillo cociente R_7 está dado por `A.<y>=QuotientRing(F,F.ideal(x^7-1))`, el automorfismo sobre R_7 se define como sigue `sigma=A.hom([y+y^2+y^3])` y finalmente el anillo torcido en la variable z y automorfismo σ, $R_7[z;\sigma]$ es `S.<z>=A['z',sigma]`.

Luego, el generador g dado en la ecuación (3.11) se representa en SAGE como elemento en `S` por `g=1+y^2+y^3+y^4+(y+y^2+y^3+y^5)*z+(1+y+y^4+y^6)*z^2` ahora se calculan los productos gx, gx^2, que en este caso se representan por `g*y` y `g*y^2` respectivamente y se obtiene

```
g*y=(y^6+y^3+y^2+y)*z^2+(y^4+y^2+y+1)*z+y^5+y^4+y^3+y
g*y^2=(y^5+y^4+y^3+y)*z^2+(y^6+y^3+y+1)*z+y^6+y^5+y^4+y^2
```

Para los demás productos se pregunta si cumplen la igualdad esto es

```
g*y^3==g+g*y^2; True
g*y^4==g+g*y+g*y^2; True
```

```
g*y^5==g+g*y; True
g*y^6==g*y+g*y^2; True
```

Luego para obtener la matriz generadora se usa el algoritmo `phi_inv` A.12, implementado en `SAGE` y se procede como sigue

```
G=Matrix([phi_inv(g,7,2),phi_inv(g*y,7,2),phi_inv(g*y^2,7,2)])
[x^2+1    x^2+x    x+1      x+1  x^2+1        x    x^2]
[    x  x^2+x+1      0  x^2+x+1  x^2+1    x^2+1      x]
[  x^2    x^2+x  x^2+1        0    x+1  x^2+x+1  x + 1]
```

Para comprobar que la matriz `G` es básica basta con calcular su forma normal de Smith en `SAGE` y verificar que esta es $\Gamma = (I_3 \quad 0_{3\times 4})$, lo cual se cumple como se verá a continuación

```
G.smith_form()[0]
[1 0 0 0 0 0 0]
[0 1 0 0 0 0 0]
[0 0 1 0 0 0 0]
```

Finalmente, haciendo uso de los algoritmos `grado_fila` A.11 y `complexity` A.10 se comprueba que la matriz `G` es minimal verificando en `SAGE` la igualdad dada en la ecuación 3.5, esto es

```
complexity(G)==sum([grado_fila(G,i) for i in [0..2]])
True
```

La principal herramienta para describir los ideales derechos, y por ende los códigos convolucionales cíclicos, en $R_n[z;\sigma]$ es el hecho de que R_n es un álgebra semisimple. Dado que n y q son coprimos, el polinomio $x^n - 1$ es libre de cuadrados y se puede factorizar de la siguiente manera

$$x^n - 1 = \pi_1 \pi_2 \cdots \pi_r, \tag{3.14}$$

donde $\pi_1, \pi_2, \ldots, \pi_r \in \mathbb{F}_q[x]$ son polinomios irreducibles, mónicos y diferentes dos a dos. Los polinomios en (3.14) son ordenados de la siguiente manera

$$\mathrm{gr}_x\, \pi_1 = \cdots = \mathrm{gr}_x\, \pi_{r_1} < \cdots < \mathrm{gr}_x\, \pi_{r_1+\cdots+r_{s-1}+1} = \cdots = \mathrm{gr}_x\, \pi_{r_1+\cdots+r_s}, \tag{3.15}$$

donde $r_1 + \cdots + r_s = r$. Sea $r_0 = 0$ y $l_t = 1 + \sum_{\lambda=0}^{t-1} r_\lambda$ para $t = 1, \ldots, s$, y se obtiene la partición $\{1, \ldots, r\} = R^{(1)} \cup \cdots \cup R^{(s)}$ con $R^{(t)} = \{l_t, l_t + 1, \ldots, l_t + r_t - 1\}$. También es necesario el uso de la siguiente relación de equivalencia

$$k \simeq l \quad \text{ si y solo si } \quad \mathrm{gr}_x\, \pi_k = \mathrm{gr}_x\, \pi_l. \tag{3.16}$$

Por lo tanto, $k \simeq l$ si y solo si k y l pertenecen al mismo conjunto de índices $R^{(t)}$ para algún t.

El Teorema Chino del Resto, ver [12, Cap. III, Cor. 2.27] proporciona un isomorfismo de anillos

$$\begin{array}{rccc} \psi: & R_n & \longrightarrow & K_1 \times K_2 \times \cdots \times K_r \\ & a & \longrightarrow & [\rho_1(a), \rho_2(a), \ldots, \rho_r(a)]\,, \end{array} \tag{3.17}$$

donde $K_k = \mathbb{F}_q[x]/\langle \pi_k \rangle$ y ρ_k denota la proyección canónica en K_k. Note que $K_k \cong K_l$ si y solo si $k \simeq l$. Como se indica en la ecuación (3.17) los elementos en el producto directo se denotan por $[a_1, a_2, \ldots, a_r]$. Se puede verificar que los elementos

$$\varepsilon^{(k)} = \psi^{-1}([(\delta_{kj})_{1 \le j \le r}]) \text{ para } k = 1, 2, \ldots, r,$$

determinan de forma única el conjunto de idempotentes primitivos de R_n. El subcampo $K^{(k)} = \varepsilon^{(k)} R_n = \psi^{-1}(0 \times \cdots \times 0 \times K_{(k)} \times 0 \times \cdots \times 0)$ se denomina la k-ésima componente de R_n. Además, $R_n = K^{(1)} \oplus \cdots \oplus K^{(r)}$ probando que R_n es un anillo Artiniano semisimple, ver [12, Cap. IX, Tm. 3.7]. En particular, R_n tiene un número finito de ideales, cada uno de los cuales es isomorfo a un producto directo de campos. Además, $a \in R_n$ es una unidad en R_n si y solo si $\varepsilon^{(l)} a \neq 0$ para todo $l = 1, \ldots, r$.

Ahora, se expone el efecto de un automorfismo dado $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$ en las componentes. Se puede ver que para cada K se tiene que $\sigma(K^{(k)}) = K^{(l)}$ para algún l tal que $l \simeq k$. En otras palabras,

$$\sigma(\varepsilon^{(k)}) = \varepsilon^{(l)} \text{ para algún } l \text{ tal que } k \simeq l. \tag{3.18}$$

Lo cual da lugar a la siguiente definición

Definición 3.10.

Sea $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$. Se define la permutación $\Pi_\sigma \in S_r$ por medio de $\Pi_\sigma(k) = l$ donde l es tal que $\sigma(\varepsilon^{(k)}) = \varepsilon^{(l)}$ para todo $k = 1, \ldots, r$, con Π_σ se denota la permutación correspondiente

a σ. Además, define la relación de equivalencia $\simeq_\sigma$ en el conjunto de índices $\{1,\ldots,r\}$ vía $k \simeq_\sigma l$ si existe algún $i \in \mathbb{N}_0$ tal que $\sigma^i(\varepsilon^{(k)}) = \varepsilon^{(l)}$.

Por su puesto, la permutación Π_σ simplemente refleja la permutación inducida por σ en el conjunto $\{\varepsilon^{(1)},\ldots,\varepsilon^{(r)}\}$, es decir $\sigma(\varepsilon^{(k)}) = \varepsilon^{(\Pi_\sigma(k))}$. La relación de equivalencia $\simeq_\sigma$ puede también expresarse como $k \simeq_\sigma l$ si y solo si k y l pertenecen al mismo ciclo de permutación Π_σ. Dado que la permutación Π_σ satisface $\Pi_\sigma(R^{(t)}) = R^{(t)}$ para todo $t = 1,\ldots,r$, ver (3.18), se obtiene que cada uno de sus ciclos está contenido en uno de los conjuntos $R^{(t)}$. En otras palabras

$$k \simeq_\sigma l \text{ entonces } k \simeq l \text{ para todo } k,l \in \{1,\ldots,r\}.$$

La consideración anterior proporciona una forma alternativa de calcular los automorfismos en R_n lo cual se prueba en [9].

Observación 3.1.

Cada permutación $\Pi \in S_r$ que satisface $\Pi(R^{(k)}) = R^{(k)}$ para todo $k \in \{1,\ldots,r\}$ es la permutación Π_σ de un $\mathbb{F}_q$-automorfismo σ en R_n. Por lo tanto, σ es tal que $\sigma(K^{(k)}) = K^{(\Pi(k))}$ para todo $k = 1,\ldots,r$. Puesto que hay en general muchos isomorfismos entre $K^{(k)}$ y $K^{(\Pi(k))}$, la permutación Π no determina completamente el automorfismo. Más bien, se consiguen todos los automorfismo σ en R_n que satisfacen $\Pi_\sigma = \Pi$ para un automorfismo fijo entre $K^{(k)}$ y $K^{(\Pi(k))}$, usando el grupo de automorfismo $\mathrm{Aut}_{\mathbb{F}_q}(R_n)$ para presentar los restantes. Se puede probar que de esta forma se obtienen todos los automorfismo en R_n, ver [28]. Con esta consideración es posible calcular la cardinalidad del grupo de automorfismos. En efecto, note que $r_1!\ldots r_s!$ cuenta el número de todas la permutaciones Π que satisfacen $\Pi(R^{(t)}) = R^t$ para todo t. Como cada k está en uno de los conjuntos $R^{(t)} = \{l_t, l_t+1,\ldots,{}_t + r_t - 1\}$ y $\left|\mathrm{Aut}_{\mathbb{F}_q}(K^{(l_t)})\right| = \mathrm{gr}_x\,\Pi_{l_t}$, así

$$|\,\mathrm{Aut}_{\mathbb{F}_q}(R_n)| = (\mathrm{gr}_x\,\Pi_{l_1})^{r_1}\ldots(\mathrm{gr}_x\,\pi_{l_s})^{r_s} r_1!\ldots r_s!. \tag{3.19}$$

A partir de esta descripción del anillo semisimple R_n y sus automorfismos disponibles, ahora para algún $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$ se considera al álgebra de Piret $R_n[z;\sigma]$ sobre R_n. Este anillo es por supuesto, un R_n-módulo y como tal semisimple (es decir, todo R_n-submódulo de $R_n[z;\sigma]$ es un sumando directo, ver [12, Cap. IX. Tm. 3.7]). Sin embargo, para el estudio

de los códigos σ-cíclicos es necesario estudiar la estructura del anillo junto con la estructura de $\mathbb{F}_q$-módulo derecho. Esto se estudió detalladamente en [9] y lleva a lo siguiente.

Sabiendo que $1 = \varepsilon^{(1)} + \cdots + \varepsilon^{(r)}$ se puede escribir cada polinomio $f \in R_n[z;\sigma]$ en la forma

$$f = f^{(1)} + \cdots + f^{(r)} \text{ donde } f^{(k)} = f\varepsilon^{(k)}, \tag{3.20}$$

donde $f^{(k)}$ se denomina la k-ésima componente de f. Además, el conjunto

$$T_f = \{k \in \{1, \ldots, r\} : f^{(k)} \neq 0\},$$

se denomina *el soporte* de f. De la ecuación (3.9) $z^\mu \varepsilon^{(k)} = \varepsilon^{(k')} z^\mu$ para algún k' tal que $k \simeq_\sigma k'$. Por lo tanto, cada $f \in R_n[z;\sigma]$ puede escribirse como una combinación R_n-lineal de elementos de la forma

$$\varepsilon^{(k)} z^\mu, \text{ con } \mu \geq 0 \text{ y } k = 1, \ldots, r. \tag{3.21}$$

Se llama a estos elementos *los monomios de* $R_n[z;\sigma]$. En particular, la k-ésima componente $f^{(k)} = \varepsilon^{(k)} f$ de f satisface

$$f^{(k)} \in \text{gen}_{R_n}\{\varepsilon^{(k)} z^\mu : \mu \geq 0, k' \cong_\sigma k\}, \tag{3.22}$$

donde el generado debe ser entendido con respecto a los coeficientes izquierdos. Así, los coeficientes derechos de $f^{(k)}$ no están en $\varepsilon^{(k)} R_n$ sino más bien están en los campos $K^{(k')} = \varepsilon^{(k')} R_n$, donde $k' \simeq_\sigma k$. A partir de esto y la ortogonalidad de los idempotentes se sigue inmediatamente la ortogonalidad de las componentes correspondientes a los ciclos disjuntos, precisamente

$$f, g \in R_n[z;\sigma], k \not\simeq_\sigma l \text{ entonces } f^{(k)} g^{(l)} = g^{(l)} f^{(k)} = 0.$$

Ejemplo 3.4.

Se considera de nuevo el ejemplo 3.3 donde el polinomio $x^7 - 1 = \pi_1 \pi_2 \pi_3$ con $\pi_1 = x + 1$, $\pi_2 = x^3 + x + 1$ y $\pi_3 = x^3 + x^2 + 1$. Luego, en la notación de las ecuaciones (3.14) y (3.15) se tiene que $r_1 = 1$ y $r_2 = 2$ de ahí $r = 3$, $s = 2$, $l_1 = 1 + \sum_{\lambda=0}^{0} r_\lambda = 1$ y $l_2 = 1 + \sum_{\lambda=0}^{1} r_\lambda = 2$. Así $R^{(1)} = \{1\}$ y $R^{(2)} = \{2, 3\}$.

Además, se tiene que los idempotentes primitivos son

$$\varepsilon^{(1)} = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6, \quad \varepsilon^{(2)} = 1 + x + x^2 + x^4 \quad \text{y} \quad \varepsilon^{(3)} = 1 + x^3 + x^5 + x^6.$$

Para $\sigma(x) = x^5$ se tiene $\sigma(\varepsilon^{(1)}) = \varepsilon^{(1)}$, $\sigma(\varepsilon^{(2)}) = \varepsilon^{(3)}$ y $\sigma(\varepsilon^{(3)}) = \varepsilon^{(2)}$. En otras palabras σ induce la permutación $\Pi_\sigma = (1)(2,3)$.

Finalmente, el polinomio g dado en la ecuación (3.11) satisface que

$$g^{(1)} = g^{(2)} = 0 \;\; y \;\; g^{(3)} = (1 + x + x^2)\varepsilon^{(3)} + x\varepsilon^{(2)}z + x\varepsilon^{(3)}z^2 = g.$$

Tener esta descripción de los polinomios en el álgebra de Piret $R_n[z;\sigma]$ a mano, permite estudiar los ideales derechos. En [9] se estableció una teoría del tipo bases de Groebner para $R_n[z;\sigma]$. Se basa en los monomios dados en la ecuación (3.21) y conduce a un algoritmo de reducción al igual que para los polinimios conmutativos en varias variables. Esto se ve de la siguiente manera

Definición 3.11.

1. *Dados dos monomios $\varepsilon^{(k)}z^\mu$ y $\varepsilon^{(l)}z^\nu$ se define*

$$\varepsilon^{(k)}z^\mu < \varepsilon^{(l)}z^\nu \;\; si \; y \; solo \; si \;\; \mu < \nu \;\; o \;\; \mu = \nu \;\; y \;\; k < l.$$

2. *Para un polinomio $f = \sum_{\nu>0} f_\nu z^\nu = \sum_{\nu>0}\sum_{l=1}^{r} f_\nu \varepsilon^{(l)} z^\nu \in R_n[z:\sigma]$, sea $LM(f)$ el monomio $\varepsilon^{(k)}z^\mu$ más grande (con respecto al orden $<$) el cual tiene un coeficiente no cero en f, esto es para el cual $f_\mu \varepsilon^{(k)} \neq 0$. El monomio $LM(f)$ se denomina el monomio principal de f. Los sumandos $f_\nu \varepsilon^{(l)} z^\nu$ se denominan términos de f.*

3. *Un polinomio $f \in R_n[z;\sigma]$ se denomina reducido si para todo $k, l = 1, \ldots, r$ con $k \neq l$ ningún término distinto de cero de $f^{(k)}$ es divisible a izquierda por $LM(f^{(l)})$.*

4. *Un polinomio $f \in R_n[z;\sigma]$ se denomina un polinomio componente si $f = f^{(k)}$ para algún $k = 1, \ldots, r$.*

Se puede verificar que $<$ es un buen orden en el conjunto de monomios con respecto a la multiplicación en la medida que el producto sea diferente de cero. Tenga en cuenta que una componente $f^{(k)}$ siempre es reducida.

Son necesarios los siguientes resultados sobre los ideales principales derechos que se probaron en [9] para el caso de ideales principales izquierdos y en este corresponden a [9, Cor. 4.13, Prop. 7.10, Tm. 7.13]

Teorema 3.2. *Sea* $\sigma \in \mathrm{Aut}_{\mathbb{F}_q}(R_n)$. *Entonces*

1. *Cada ideal derecho* $\mathcal{I} \in R_n[z,\sigma]$ *tiene polinomio generador reducido. Más precisamente, existe un polinomio reducido* $g \in R_n[z,\sigma]$ *tal que*

$$\mathcal{I} = \{gf : f \in R_n[z,\sigma]\}.$$

Además, el generador reducido es único salvo multiplicación derecha por unidades en R_n.

2. *Sea* $C \subseteq \mathbb{F}_q[z]^n$ *un código convolucional* σ*-cíclico. Entonces el ideal* $\varphi(C)$ *es principal y así tiene un generador reducido* $g \in R_n[z;\sigma]$. *Además, el soporte de* g *satisface que* $T_g = T_{g_0}$, *donde* g_0 *denota el término constante de* g.

3. *Sea* $g \in R_n[z;\sigma]$ *un polinomio reducido. Entonces* $\varphi^{-1}(\langle g\rangle) \subseteq \mathbb{F}_q[z]^n$ *es un sumando directo de* $\mathbb{F}_q[z]^n$ *(por lo tanto es un código convolucional* σ*-cíclico) si y solo si existen* $a \in R_n$ *y una unidad* $v \in R_n[z;\sigma]$ *tales que* $g = av$.

4. *Sea* $g \in R_n[z;\sigma]$ *un polinomio reducido con soporte* T_g. *Para* $l \in T_g$ *sea* $\mathrm{gr}_x\, \pi_l = \kappa_l$, *donde* π_l *es como en* (3.14) *y escriba* $\kappa = \sum_{l\in T_g} \kappa_l$. *Entonces la matriz*

$$G = \left(\varphi^{-1}(x^i g^{(l)})\right)_{l\in T_g, 0\le i\le \kappa_l - 1}$$

es una matriz generadora minimal del submódulo $S = \mathrm{im}\, G \subseteq \mathbb{F}_q[z]^n$. *Como consecuencia* S *es un submódulo de rango* κ *y complejidad* $\delta = \sum_{l\in T_g} \kappa_l \,\mathrm{gr}_z\, g^{(l)}$.

El ejemplo 3.3 fue presentado en el artículo [8, Ejemplo. 2.3] para el automorfismo $\sigma(x) = x^5$, sin embargo con ayuda de la siguiente rutina implementada en SAGE este ejemplo se puede replicar para los automorfismos de R_7 siempre que $g \in R_7[z;\sigma]$ sea un polinomio compontente.

```
def comprueba(g,kl,n,q,p):
    F.<x> = GF(q)[]
    A.<y> = QuotientRing(F,F.ideal(x^n-1))
    sigma = A.hom([p])
    S.<z> = A['z',sigma]
    FS = matrix(GF(q),[[1,0,0,0,0,0,0],[0,1,0,0,0,0,0],[0,0,1,0,0,0,0]])
    G = matriz_red(g,kl,n,q)
    v = sum([grado_fila(G,i) for i in [0..kl-1]])
```

```
    return [FS==G.smith_form()[0], v==complexity(G)]
```

Precisamente, Cornelis Roos en [21, p.680] presenta la tabla 3.1 de todos los automorfismos de $R_7 = \mathbb{F}_2[x]/\langle x^7 - 1\rangle$.

x;	$x + x^3 + x^5$;	$x + x^2 + x^3 + x^4 + x^5$;
x^2;	$x^2 + x^3 + x^6$;	$x + x^2 + x^3 + x^4 + x^6$;
x^3;	$x + x^2 + x^3$;	$x + x^2 + x^3 + x^5 + x^6$;
x^4;	$x^4 + x^5 + x^6$;	$x + x^2 + x^4 + x^5 + x^6$;
x^5;	$x + x^4 + x^5$;	$x + x^3 + x^4 + x^5 + x^6$;
x^6;	$x^2 + x^4 + x^6$;	$x^2 + x^3 + x^4 + x^5 + x^6$.

Tabla 3.1: *Automorfismos de* $R_7 = \mathbb{F}_2[x]/\langle x^7 - 1\rangle$.

En la tabla 3.2, se presentan los automorfismos de R_7 para los que $g = g^{(3)}$, es decir g es un polinomio compontente y por lo tanto es reducido. Para un automorfismo σ en la tabla 3.2, se tiene que el submódulo $\langle g\rangle$ es un código convolucional σ-cíclico y la matriz generadora obtenida es básica y minimal.

x^3;	$x + x^2 + x^3$;	$x + x^2 + x^3 + x^5 + x^6$;
x^5;	$x + x^4 + x^5$;	$x + x^3 + x^4 + x^5 + x^6$;
x^6;	$x^2 + x^4 + x^6$;	$x^2 + x^3 + x^4 + x^5 + x^6$.

Tabla 3.2: *Automorfismos de* R_7 *para los cuales* g *es un polinomio componente.*

Por otro lado al tomar $\sigma = x + x^3 + x^5$, se tiene que $g = g^{(2)} + g^{(3)}$, donde

$$g^{(2)} = g\varepsilon^{(2)} = (x^5 + x^3 + x^2 + x) * z,$$
$$g^{(3)} = g\varepsilon^{(3)} = (x^6 + x^4 + x + 1) * z^2 + x^4 + x^3 + x^2 + 1.$$

Sin embargo, en este caso no es posible verificar que g es un polinomio reducido por tal motivo no se puede aplicar el Teorema 3.2 para determinar si g genera un código convolucional σ-cíclico y determinar una matriz generadora minimal.

Por lo anterior en este caso aquí se aplicará el Teorema 3.2 para $g^{(2)}$ y $g^{(3)}$ con ayuda de SAGE como se muestra a continuación

```
F.<x> = GF(2)[]
A.<y>=QuotientRing(F,F.ideal(x^7-1))
sigma=A.hom([y+y^3+y^5])
S.<z>=A['z',sigma]
g2=(y^5+y^3+y^2+y)*z
g3=(y^6+y^4+y+1)*z^2 + y^4+y^3+y^2+1
```

Luego, por medio del algoritmo A.13 se calculan las matrices generadoras G_2 y G_3 correspondientes

```
G2=matriz_red(g2,3,7,2)
G3=matriz_red(g3,3,7,2)
```

Finalmente se calcula la forma de Smith de cada matriz para determinar si estas generan o no un código convolucional σ-cíclico

```
G2.smith_form()[0]
[1 0 0 0 0 0 0]
[0 1 0 0 0 0 0]
[0 0 1 0 0 0 0]

G3.smith_form()[0]
[1 0   0   0 0 0 0]
[0 1   0   0 0 0 0]
[0 0 t^3+1 0 0 0 0]
```

De lo anterior se tiene que la matriz G_3 no es básica, lo que implica que para el automorfismo $\sigma = x + x^3 + x^5$, el submódulo $\langle g^{(3)} \rangle$ no es un código convolucional σ-cíclico puesto que no es un sumando directo, sin embargo el submódulo $\langle g^{(2)} \rangle$ si lo es.

Por medio de SAGE se puede verificar que se obtienen resultados similares para los automorfismos de R_7 que no se encuentran en la la tabla 3.2, demostrando así que con $g^{(2)}$ es posible construir un código convolucional σ-cíclico, pero con $g^{(3)}$ no.

3.4. Números de Midy y códigos convolucionales

Aplicando el Teorema 3.2 al ejemplo 3.3 y dado que del ejemplo 3.4 se tiene que $g = g^{(3)}$ es un polinomio componente, entonces una matriz generadora para el código está dada por

$$G = \begin{pmatrix} \varphi^{-1}(g) \\ \varphi^{-1}(gx) \\ \varphi^{-1}(gx^2) \end{pmatrix}.$$

De esta manera el rango de este código es $k = \operatorname{gr}_x \pi_3 = 3$ y la complejidad es $\delta = \operatorname{gr}_x \pi_3 \operatorname{gr}_z g^{(3)} = 3 \times 2 = 6$. En este ejemplo es importante que en la factorización de $x^7 - 1$ en $\mathbb{F}_2$ todos los polinomios (excepto $x - 1$) sean del mismo grado. Este resultado en realidad se puede obtener para $x^p - 1$ cuando p es un primo impar, pero esta misma situación se consigue para algunos números compuestos conocidos como los números de Midy, ver Teorema 3.4.

Sean N y b enteros positivos primos relativos, $b > 1$ la base de numeración, $|b|_N$ el orden de b en el grupo multiplicativo $\mathbb{U}_N$ de enteros positivos menores que N y primos relativos con N, y $x \in \mathbb{U}_N$. Se sabe que al escribir la fracción $\frac{x}{N}$ en base b, ésta es periódica. El período denota la secuencia de dígitos en base b que se repite en tal expresión, se puede probar que $|b|_N$ es la longitud del período de la fracción $\frac{x}{N}$. Sean d, k enteros positivos con $|b|_N = dk$,

$d > 1$ y $\frac{x}{N} = 0.\overline{a_1 a_2 \cdots a_{|b|_N}}$, donde la barra indica el período y los a_i son dígitos en base b. A continuación se separa el período $a_1 a_2 \cdots a_{|b|_N}$ en d bloques de longitud k cada uno. De esta forma sea

$$A_j = [a_{(j-1)k+1} a_{(j-1)k+2} \cdots a_{jk}]_b$$

el número representado en base b por el j-ésimo bloque y $S_d(x) = \sum_{j=1}^{d} A_j$. Si para todo $x \in \mathbb{U}_N$, la suma $S_d(x)$ es múltiplo de $b^k - 1$ se dice que N tiene la propiedad de Midy para b y d.

Dados b y N se denota con $\mathcal{M}_b(N)$ el conjunto de todos los d tales que N tiene la propiedad de Midy para b y d y se llamará a este, el conjunto de Midy de N para la base b. Con $\nu_p(N)$ se indica el mayor exponente del primo p en la factorización prima de N.

Definición 3.12.
Se dice que N es un número de Midy base b, si N es un entero impar compuesto primo relativo con b y $|b|_N$ y tal que todo divisor $d > 1$ de $|b|_N$ pertenece a $\mathcal{M}_b(N)$.

En [2] se prueba que cuando N es un número de Midy base b se puede caracterizar a partir del orden módulo p, denotado por $|b|_p$, para cada divisor primo p de N.

Teorema 3.3.
N es un número de Midy base b si y sólo si $|b|_N = |b|_p$ para todo p divisor primo de N.

Por otro lado, el concepto de número de Midy se puede obtener usando la noción de clases q-ciclotómicas módulo n.

Definición 3.13 (Clase ciclotómica)**.**
Sea s un entero con $0 \leq s < n$. La clase q-ciclotómica de s módulo n se define como el conjunto

$$C_s = \{sq^i \bmod n : 0 \leq i < r\},$$

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

El menor elemento de C_s se llama el representante de clase. Además es fácil probar que

$$\mathbb{Z}_n = \bigcup_i C_i,$$

donde la unión se hace sobre los i representantes de clase y

$$C_i \cap C_j = \begin{cases} C_i, & \text{si } i \in C_j, \\ \emptyset, & \text{en caso contrario.} \end{cases}$$

De estas dos últimas afirmaciones se sigue que las clases q-ciclotómicas módulo n se constituyen en una partición de $\mathbb{Z}_n$. En [26] los autores Shevelev et al. introducen el concepto de overpseudoprimo base b de la siguiente manera.

Definición 3.14.
Un número compuesto n primo realtivo con b, se denomina un overpseudoprimo base b si satisface que

$$n = r_b(n)|b|_n + 1,$$

donde $r_b(n)$ denota el número de clases b-ciclotómicas módulo n diferentes a la que contiene el 0.

Observe que la definición anterior significa que un número n es un overseudoprimo base b si y solo si todas las clases ciclotómicas, excepto aquella que contiene al 0, tienen el mismo cardinal.

En [26] se demuestra que el concepto de número de Midy base b es equivalente al concepto de overpseudoprimo base b.

En la literatura el concepto clase q-ciclotómica módulo n aparece en el contexto de los códigos de bloque cíclicos, como una herramienta para encontrar la factorización en polinomios irreducibles del polinomio $x^n - 1$ en $\mathbb{F}_q$, como se muestra a continuación

Sea q una potencia de un primo, n primo relativo con q y α un elemento primitivo de $\mathbb{F}_{q^t}$, donde $t = |q|_n$. Se puede obtener una factorización prima de $x^n - 1$ de la siguiente manera.

Teorema 3.4 ([11, Teorema 4.1.1])**.**
Sea n un entero positivo primo relativo con q. Sean $t = |q|_n$ y α una raíz n-ésima primitiva de la unidad en $\mathbb{F}_{q^t}$. Entonces

1. *Para cada entero s con $0 \leq s < n$ el polinomio minimal de α^s en $\mathbb{F}_q$ está dado por*

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

2. *Además*

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

es una factorización de $x^n - 1$ en polinomios irreducibles en $\mathbb{F}_q$, donde s recorre los representantes de las clases q-ciclotómicas módulo n.

En consecuencia, cuando N es un número de Midy base b, el polinomio $\dfrac{x^N - 1}{x - 1}$ se puede expresar como producto de polinomios del mismo grado, esto es $|C_s|$ donde s es un representante de las clases q-ciclotómicas módulo n.

De lo anterior se tiene que

$$x^N - 1 = \pi_1 \pi_2 \cdots \pi_r, \tag{3.23}$$

donde $\pi_1 = x - 1, \pi_2, \ldots, \pi_r \in \mathbb{F}_q[x]$ son polinomios irreducibles, mónicos y diferentes dos a dos. Dado que los polinomios $\pi_2, \ldots, \pi_r$ son del mismo grado estos son ordenados en dos grupos de la siguiente manera

$$\mathrm{gr}_x(x - 1) < \mathrm{gr}_x \pi_2 = \mathrm{gr}_x \pi_3 = \cdots = \mathrm{gr}_x \pi_r, \tag{3.24}$$

luego de la ecuación (3.15) se sigue que $r_1 = 1$ y $r_2 = r - 1$. Sea $r_0 = 0$ y $l_t = 1 + \sum_{\lambda=0}^{t-1} r_\lambda$ para $t = 1, 2$, la partición $\{1, \ldots, r\} = R^{(1)} \cup R^{(2)}$ con $R^{(t)} = \{l_t, l_t + 1, \ldots, l_t + r_t - 1\}$.

Además, la ecuación (3.19) se transforma en

$$|\operatorname{Aut}_{\mathbb{F}_q}(R_n)| = (\operatorname{gr}_x \pi_2)^{r-1}(r-1)!. \tag{3.25}$$

Finalmente por el item 4 del Teorema 3.2 para cuando N es un número de Midy base q se tiene la siguiente estructura para la matriz generadora minimal de un código convolucional σ-ciclico y su complejidad.

Teorema 3.5.
Sean N un número de Midy base q y s un representante de una clase q-ciclotómica módulo N no nula. Sea $g \in R_n[z;\sigma]$ un polinomio reducido con soporte T_g. Para $l \in T_g$ sea $\kappa_l = \operatorname{gr}_x \pi_l$, donde π_l es como en (3.23). Entonces

1. *Una matriz generadora minimal del submódulo $S = \operatorname{im} G \subseteq \mathbb{F}_q[z]^N$ es*

$$G = \left(\varphi^{-1}(x^i g^{(l)})\right)_{l \in T_g, 0 \le i \le \kappa_l - 1}.$$

2. *Si $1 \in T_g$, entonces S es un submódulo de rango $\kappa = 1 + (|T_g| - 1) \operatorname{gr}_x \pi_2$ y la complejidad es $\delta = \operatorname{gr}_z g^{(1)} + \operatorname{gr}_x \pi_2 \sum_{\substack{l \in T_g \\ l \ne 1}} \operatorname{gr}_z g^l$.*

3. *Si $1 \notin T_g$, entonces S es un submódulo de rango $\kappa = |T_g| \operatorname{gr}_x \pi_2$ y la complejidad es $\delta = \operatorname{gr}_x \pi_2 \sum_{l \in T_g} \operatorname{gr}_z g^l$.*

En particular, si g es un polinomio componente, N es un número de Midy base q y s es un representante de una clase q-ciclotómica módulo N no nula, la matriz generadora minimal del submódulo $S = \operatorname{im} G \subseteq \mathbb{F}_q[z]^N$ es $G = (\varphi^{-1}(x^i g))$ donde $0 \le i < |C_s|$. El rango de este código es igual a $|C_s|$ y la complejidad es $\delta = |C_s| \operatorname{gr}_z g$.

Conclusiones

1. En este trabajo se describen conceptos fundamentales de la teoría básica sobre los códigos convolucionales, presentando ejemplos de los conceptos y los procesos de codificación y un método de decodificación; lo cual proporciona las bases para animar al lector en el estudio más a fondo de las cuestiones aquí tratadas. Cabe resaltar que la mayoría de los procesos aquí descritos se presentan de manera detallada, dado que en los textos estudiados se omiten pasos que dificultan comprender con claridad los conceptos y la manera como se realizan los procesos.

2. En el desarrollo del documento se evidencia cómo evolucionan los conceptos algebraicos desde los conceptos de códigos de bloque, pasando por códigos lineales hasta llegar a los códigos convolucionales.

3. Adicionalmente, se presenta la evolución del concepto de código convolucional cíclico hasta llegar a la estructura de los códigos convolucionales σ-cíclicos con base en el artículo "On Cyclic Convolutional Codes", reelaborando de manera detallada las pruebas necesarias para que estas sean más claras, además algunas son producción propia ante la ausencia de las mismas en los textos y artículos estudiados, por ejemplo las demostraciones de las Proposiciones 3.1, 3.2, 3.3 y 3.4 . También fue posible establecer algunas propiedades de la matriz generadora de un código convolucional σ-cíclico cuando su longitud es un número de Midy.

4. Finalmente, el trabajo sirve para comprobar que el programa computacional `SAGE`, es una herramienta útil para realizar cálculos permitiendo presentar ejemplos más grandes de los que se encuentra usualmente en la bibliografía consultada, de esta manera demostrando ser un recurso importante y el cual debe ser estudiado con mayor profundidad para conseguir resultados en proyectos de investigación futuros.

Apéndice A

Algoritmos e implementación en SAGE

En este apéndice se consigna el pseudocodigo de los algoritmos y su implementación en el programa computacional SAGE utilizados en el presente trabajo de investigación. De los cuales se hace una breve descripción de su fundamento teórico e implementación.

Algoritmo A.1. (Convolución discreta, `convodiscre`)
Para el caso de un $(n, 1, m)$-código convolucional, este algoritmo calcula la convolución discreta, de la 1-secuencia de información con una 1-secuencia generadora y se basa en la ecuación 2.4.

Algoritmo A.1 Convolución discreta

Entrada: 1-secuencia de información binaria $\boldsymbol{u}$, 1-secuencia generadora binaria $\boldsymbol{g}^{(j)}$.
Salida: 1-secuencia código $\boldsymbol{v}^{(j)}$.

1: Calcular $N =$ longitud de $\boldsymbol{u}$.
2: Calcular $m =$ longitud de $\boldsymbol{g}^{(j)}$.
3: $M = N + m$
4: $V =$ vector $\mathbf{0}$ de longitud M.
5: Agregar m ceros al comienzo y al final de $\boldsymbol{u}$.
6: **para** $t = 0 : M - 1$ **hacer**
7: $\quad V[t] = \boldsymbol{u}[t : t + m] \cdot \boldsymbol{g}^{(j)}$.
8: **fin para**
9: **devolver** vector V

Para su implementación en `SAGE`, el algoritmo recibe dos vectores `u` y `g` con entradas en el campo $\mathbb{F}_2$, donde u representa la 1-secuencia de información $\boldsymbol{u} = (u_0, u_1, u_2, \ldots, u_N)$ y g representa una de las 1-secuencias generadoras $\boldsymbol{g}^{(j)} = (g_0^{(j)}, g_1^{(j)}, \ldots, g_m^{(j)})$; finalmente la salida de este algoritmo es un vector `v` que representa la 1-secuencia codificada $\boldsymbol{v}^{(j)}$.

```
def convodiscre(u,g):
""" El algoritmo recibe por entradas dos vectores con entradas en el campo de
    Galois de orden 2 y da por salida un vector que es la
    convolución discreta de u con g."""
```

```
    N=len(u);
    m=len(g)-1;
    G0=list(g);
    G1=list(reversed(G0));
    G=vector(GF(2),G1);
    M=N+m;
    p=N+2*m;
    U=vector(GF(2),[0 for i in [0..p-1]]);
    for d in [0..N-1]:
        U[m+d]=u[d]
    V=vector(GF(2),[0 for s in [1..M]]);
    for k in [0..M-1]:
        f=k+m;
        U1=vector(GF(2),[U[i] for i in [k..f]]);
        V[k]=U1.inner_product(G);
    return(V);
```

Algoritmo A.2. (Serializador, `serializador`)
Este algoritmo, recibe como entrada una lista de vectores definidos sobre el campo $\mathbb{F}_2$ de igual longitud; intercala las componentes de cada vector en la lista formando un nuevo vector que será posteriormente la salida. Este algoritmo se usa para la codificación usando secuencias generadoras y también puede usarse para el calculo de las secuencias compuestas.

Algoritmo A.2 Serializador

Entrada: Lista L de vectores de igual longitud con entradas sobre el campo $\mathbb{F}_2$.
Salida: Vector con entradas sobre el campo $\mathbb{F}_2$.

1: Calcular $t =$ longitud de L.
2: Calcular $p =$ longitud de $L[1]$.
3: $h = t * p$
4: $V =$ vector **0** de longitud h.
5: **para** $k = 1 : p$ **hacer**
6: **para** $s = 1 : t$ **hacer**
7: $V[t * k + s] = L[s][k]$.
8: **fin para**
9: **fin para**
10: **devolver** vector V

Su implementación en `SAGE` es

```
def serializador(L):
""" Este algoritmo recibe por entrada una lista de vectores cuyas componentes
   pertenecen al campo de Galois de orden 2 y tiene por salida un vector """
    t=len(L);
    p=len(L[0]);
    h=t*p;
```

```
    V=vector(GF(2),[0 for s in [1..h]]);
    for k in [0..p-1]:
        for s in [0..t-1]:
            V[t*k+s]=L[s][k];
    return(V);
```

Algoritmo A.3. (Secuencias Generadoras, `secgen`)
Para el caso de un $(n, 1, m)$-código convolucional, este algoritmo calcula la n-secuencia código $\boldsymbol{v}$, de la 1-secuencia de información $\boldsymbol{u} = (u_0, u_1, u_2, \ldots, u_N)$, usando las 1-secuencias generadoras $\boldsymbol{g}^{(j)} = (g_0^{(j)}, g_1^{(j)}, \ldots, g_m^{(j)})$ con $1 \leq j \leq n$.

Algoritmo A.3 Secuencias Generadoras

Entrada: 1-secuencia de información binaria $\boldsymbol{u}$, una lista $\boldsymbol{g}$ de las 1-secuencias generadoras binarias $\boldsymbol{g}^{(j)}$.
Salida: n-secuencia código $\boldsymbol{v}$.
1: Calcular $n =$ longitud de $\boldsymbol{g}$.
2: Calcular $m =$ longitud de una 1-secuencias generadoras binarias $\boldsymbol{g}^{(j)}$ menos 1.
3: $V1 =$ lista vacía.
4: **para** $j = 1 : n$ **hacer**
5: Calcular la convolución discreta de $\boldsymbol{u}$ con $\boldsymbol{g}^{(j)}$ e insertar al final de $V1$.
6: **fin para**
7: $V =$ serializar $(V1)$.
8: **devolver** vector V.

Su implementación en `SAGE` es

```
def secgen(u,g):
""" Este algoritmo recibe un vector u que representa una 1-secuencia de
   informacion y una lista de vectores que son las secuencias generadoras y
   tiene por salida un vector que representa la secuencia código  """
    n=len(g);
    m=len(g[0])-1;
    V1=[];
    for j in [0..n-1]:
        V1.append(convodiscre(u,g[j]));
    V=serializador(V1)
    return(V);
```

Algoritmo A.4. (k-Secuencias Generadoras, `ksecgen`)
Para el caso de un $(n, k > 1, m)$-código convolucional, este algoritmo calcula la n-secuncia codificada, de la k-secuencia de información usando las $m + 1$ 1-secuencias generadoras. El algoritmo recibe una lista de vectores u con entradas en el campo $\mathbb{F}_2$, que representa la k-secuencia de información $\boldsymbol{u} = (u_0, u_1, u_2, \ldots, u_N)$ y una lista g de $m+1$ vectores con entradas en el campo $\mathbb{F}_2$ que son las 1-secuencias generadoras $\boldsymbol{g}^{(j)} = (g_0^{(j)}, g_1^{(j)}, \ldots, g_m^{(j)})$; finalmente la salida de este algoritmo es un vector v que representa la n-secuencia código de longitud $M = N + m$.

Algoritmo A.4 k-Secuencias Generadoras

Entrada: k-secuencia de información binaria $\boldsymbol{u}$, una lista $\boldsymbol{g}$ de las 1-secuencias generadoras binarias $\boldsymbol{g}_i^{(j)}$.
Salida: n-secuencia código $\boldsymbol{v}$.

1: Calcular n = longitud de $\boldsymbol{g}[0]$.
2: Calcular k = longitud de $\boldsymbol{u}$.
3: Calcular m = longitud de $\boldsymbol{g}[0][0]$ menos 1.
4: $V1$ = lista vacía.
5: **para** $j = 1 : n$ **hacer**
6: B = lista vacía.
7: **para** $i = 1 : k$ **hacer**
8: Calcular la convolución discreta de $\boldsymbol{u}[i]$ con $\boldsymbol{g}[i][j]$ e insertar al final de B.
9: **fin para**
10: Sumar los vectores en B e insertar el vector suma al final de $V1$
11: **fin para**
12: V = serializar $(V1)$.
13: **devolver** vector V.

Su implementación en `SAGE` es

```
def ksecgen(u,g):
""" Este algoritmo recibe una lista de vectores u que representa una k-
    secuencia de informacion y una lista de vectores que son las secuencias
    generadoras y tiene por salida un vector que representa la n-secuencia
    código """
    n=len(g[0]);
    k=len(u);
    m=len(g[0][0])-1;
    V1=[];
    for j in [0..n-1]:
        B=[];
        for i in [0..k-1]:
            B.append(convodiscre(u[i],g[i][j]));
        V1.append(sum(vector(v)for v in B));
    V=serializador(V1)
    return(V);
```

Algoritmo A.5. (Matriz Generadora, `matgen`)
Para un (n, k, m)-código convolucional este algoritmo se basa en la ecuación (2.9), recibe un vector u con entradas en el campo $\mathbb{F}_2$, que representa la k-secuencia de información $\boldsymbol{u} = (\boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N)$ y una lista G con las $m+1$ matrices G_l con entradas en el campo $\mathbb{F}_2$ que son las componentes de la matriz generadora semi-infinita como se muestra en la ecuación (2.11). Finalmente, la salida de este algoritmo es un vector v que representa la n-secuencia código de longitud $M = N + m$.

Algoritmo A.5 Matriz Generadora

Entrada: k-secuencia de información binaria $\boldsymbol{u}$, una lista $\boldsymbol{G}$ con las componentes G_l de la matriz generadora G.
Salida: n-secuencia código $\boldsymbol{v}$.

1: Calcular k = número de filas de la matriz $\boldsymbol{G}[0]$.
2: Calcular n = número de columnas de la matriz $\boldsymbol{G}[0]$.
3: Calcular m = longitud de $\boldsymbol{G}$ menos 1.
4: Calcular N = longitud de $\boldsymbol{u}$ divido k.
5: Calcular $M = N + m$.
6: Construir una lista U con N vectores de longitud k.
7: Agregar m vectores $\mathbf{0}$ de longitud k al comienzo y al final de U.
8: V = lista vacía.
9: **para** $t = 1 : M$ **hacer**
10: B = lista vacía.
11: **para** $i = 0 : m$ **hacer**
12: Calcular $U[t + m - i] \cdot \boldsymbol{G}[i]$ e insertar al final de B.
13: **fin para**
14: Sumar los vectores en B e insertar el vector suma al final de V
15: **fin para**
16: **devolver** vector V.

Su implementación en SAGE es

```
def matgen(u,G):
    """Este algoritmo recibe un vector u de tamaño N*k que representa una
       k-secuencia de información de longitud N y una lista de matrices que
       son las componentes de la matriz generadora y tiene por salida una
       lista de vectores que representa la n-secuencia código """
    n=G[0].ncols();
    k=G[0].nrows();
    N=len(u)/k;
    m=len(G)-1;
    M=N+m;
    h=2*m+N;
    U=[];
    for p in [0..N-1]:
        s=0;
        w=[];
        while s<=k-1:
            w.append(u[s+p*k]);
            s=s+1;
        x=vector(GF(2),w);
        U.append(x);
    r=vector(GF(2),[0 for i in [0..k-1]]);
    u1=[r for t in [0..h-1]];
    for d in [0..N-1]:
        u1[m+d]=U[d];
    V=[];
```

```
    for t in [0..M-1]:
        B=[];
        for i in [0..m]:
            B.append(u1[t+m-i]*G[i]);
        V.append(sum(vector(v)for v in B));
    return(V);
```

Algoritmo A.6. (Secuencias Generadoras Polinomial, `secgenpol`)
Para el caso de un $(n, 1, m)$-código convolucional, este algoritmo recibe una lista g con los n polinomios generadores y el polinomio de información u y tiene por salida una lista de polinomios v. Esta algoritmo se fundamenta en la ecuación (2.15).

Algoritmo A.6 Secuencias Generadoras Polinomial

Entrada: 1-secuencia de información polinomial $\boldsymbol{u}(z)$, una lista $\boldsymbol{g}$ con las secuencias generadoras polinimiales $\boldsymbol{g}^{(j)}(z)$.
Salida: n-secuencia código $\boldsymbol{v}(z)$.
1: Calcular $n =$ longitud de $\boldsymbol{g}$.
2: $V =$ lista vacía.
3: **para** $s = 1 : n$ **hacer**
4: Calcular $\boldsymbol{u} \cdot \boldsymbol{g}[s]$ e insertar al final de V.
5: **fin para**
6: **devolver** vector V.

Su implementación en `SAGE` es

```
def secgenpol(u,g):
    """ Este algoritmo recibe un polinomio u que representa una 1-secuencia de
        información y una lista de polinomios que son las secuencias
       generadoras y tiene por salida una lista de polinomios que representa
       la secuencia código  """
    n=len(g);
    V=[];
    for s in [0..n-1]:
        V.append(u*g[s]);
    return(V);
```

Algoritmo A.7. (k-Secuencias Generadoras Polinomial, `ksecgenpol`)
Para el caso de un (n, k, m)-código convolucional con $k > 1$, este algoritmo recibe una lista g con n listas con k polinomios generadores y una lista de polinomio de información u y tiene por salida una lista v con n polinomios código. Esta algoritmo se fundamenta en la ecuación (2.13).

Algoritmo A.7 k-Secuencias Generadoras Polinomial

Entrada: k-secuencia de información polinomial $\boldsymbol{u}(z)$, una lista $\boldsymbol{g}$ de listas, con las secuencias generadoras polinimiales $\boldsymbol{g}_i^{(j)}(z)$.
Salida: n-secuencia código $\boldsymbol{v}(z)$.
1: Calcular $n =$ longitud de $\boldsymbol{g}[0]$.
2: Calcular $k =$ longitud de $\boldsymbol{u}$.
3: $V =$ lista vacía.
4: **para** $j = 1 : n$ **hacer**
5: $B =$ lista vacía.
6: **para** $i = 1 : k$ **hacer**
7: Calcular $\boldsymbol{u}[i] \cdot \boldsymbol{g}[i][j]$ e insertar al final de B.
8: **fin para**
9: Sumar los vectores en B e insertar el vector suma al final de V.
10: **fin para**
11: **devolver** vector V.

Su implementación en SAGE es

```
def ksecgenpol(u,g):
    """ Este algoritmo recibe una lista de polinomios u que representa una k-
       secuencia de información y una lista de listas de polinomios que son
       las secuencias generadoras y tiene por salida lista de polinomios que
       representa la n-secuencia código """
    n=len(g[0]);
    k=len(u);
    V=[];
    for j in [0..n-1]:
        B=[];
        for i in [0..k-1]:
            B.append(u[i]*g[i][j]);
        V.append(sum(v for v in B));
    return(V);
```

Algoritmo A.8. (Matriz Generadora Polinomial, `matgenpol`)
Para el caso de un $(n, 1, m)$-código convolucional, este algoritmo recibe un polinomio de información u y un vector con los n polinomios generadores, es decir la matriz generadora polinomial y tiene por salida un vector de polinomios.

Algoritmo A.8 Matriz Generadora Polinomial

Entrada: 1-secuencia de información polinomial $\boldsymbol{u}(z)$, un vector $\boldsymbol{G}(z)$ con las secuencias generadoras polinimiales $\boldsymbol{g}^{(j)}(z)$.
Salida: n-secuencia código $\boldsymbol{v}(z)$.
1: Calcular $\boldsymbol{v}(z) = \boldsymbol{u}(z) \cdot \boldsymbol{G}(z)$.
2: **devolver** vector $\boldsymbol{v}(z)$.

Su implementación en SAGE es

```
def matgenpol(u,g):
    """Este algoritmo recibe un polinomio u que representa una 1-secuencia de
       información y un vector con entradas polinomiales es decir la matriz
       generadora polinomial y tiene por salida un vector de polinomios que
       representa la n-secuencia código """
    V=u*g;
    return(V);
```

Algoritmo A.9. (k Matriz Generadora Polinomial, `kmatgenpol`)
Para el caso de un (n, k, m)-código convolucional, este algoritmo recibe un vector de información U de longitud k, con entradas en $\mathbb{F}_2(z)$ y una matriz G de tamaño $k \times n$ cuyas componentes son los polinomio generadores y tiene por salida un vector de polinomios.

Algoritmo A.9 k-Matriz Generadora Polinomial

Entrada: k-secuencia de información polinomial $\boldsymbol{u}(z)$, una matriz $\boldsymbol{G}(z)$ con las secuencias generadoras polinomiales $\boldsymbol{g}_i^{(j)}(z)$.
Salida: n-secuencia código $\boldsymbol{v}(z)$.
1: Calcular $\boldsymbol{v}(z) = \boldsymbol{u}(z) \cdot \boldsymbol{G}(z)$.
2: **devolver** vector $\boldsymbol{v}(z)$.

Su implementación en `SAGE` es

```
def kmatgenpol(U,G):
    """Este algoritmo recibe un vector de polinomios U que representa una k-
       secuencia de información y una matriz G cuyas componentes son
       polinomios, es decir  la matriz generadora polinomial y tiene por
       salida un vector de polinomios que representa la n-secuencia código
       """
    V=U*G;
    return(V);
```

Algoritmo A.10. (Complejidad, `complexity`)
El algoritmo se fundamenta en la Definición 3.3. Recibe una matriz generadora $G \in \mathbb{F}_q[z]^{k\times n}$ de un código convolucional C y tiene por salida la complejidad δ del código.

Algoritmo A.10 Complejidad

Entrada: Una matriz G de tamaño $k \times n$ en $\mathbb{F}_q[z]^{k\times n}$.
Salida: Un entero δ que representa la complejidad del código.
1: Calcular $k =$ número de filas de G.
2: Calcular una lista m con todos los k-menores.
3: Calcular una lista s con los grados de los k-menores en m.
4: **devolver** $\delta = \max(s)$.

Su implementación en `SAGE` es

```
def complexity(G):
 """Este algoritmo recibe una matriz G que representan una matriz generadora
    de un (n,k)-código convolucional y tiene por salida la complejidad de G.
    """
    k=G.nrows()
    m=G.minors(k)
    s=[f.degree() for f in m]
    return max(Set(s))
```

Algoritmo A.11. (Grado fila de una matriz generadora, `grado_fila`)
El algoritmo de basa en el item 2 de la Definición 3.4. Tiene por entradas una matriz $G \in \mathbb{F}_q[z]^{k\times n}$ junto con un entero $1 \leq i \leq k$ y su salida el máximo grado de las entradas en la i-ésima fila de G.

Algoritmo A.11 Grado fila

Entrada: Una matriz G de tamaño $k \times n$ y dos enteros n e i, donde i representa el índice de una fila de la matriz dada.
Salida: El máximo grado de la i-ésima fila de G.
1: Calcular $n =$ número de columnas de la matriz G.
2: Calcular en una lista T el grado de cada una de las entradas de la i-ésima fila de la matriz G.
3: **devolver** máx(T).

Su implementación en SAGE es

```
def grado_fila(G,i):
 """Este algoritmo recibe una matriz G que representa una matriz generadora de
     un (n,k)-código convolucional junto con un entero i entre 1 y k. Este
    algoritmo tiene por salida el máximo grado de la i-ésima fila de G."""
    n=G.ncols()
    T=[G[i][j].degree() for j in [0..n-1]]
    return max(T)
```

Algoritmo A.12. (Imagen inversa de la función φ, `phi_inv`)
Este algoritmo recibe un polinomio $g \in R_n[z]$ y devuelve un vector $\boldsymbol{u} \in \mathbb{F}_q[z]^n$; es decir calcula $\varphi^{-1}(g)$ siendo φ la función definida en la ecuación (3.5).

Algoritmo A.12 Imagen inversa de la función φ

Entrada: Un polinomio $g \in R_n[z]$ y dos enteros n y q primos relativos.
Salida: Un vector $\boldsymbol{u} \in \mathbb{F}_q[z]^n$.

1: Construir el anillo de polinomios $\mathbb{F}_q[z]$.
2: Calcular el vector $C \in \mathbb{F}_q^n$ de coeficientes del polinomio g.
3: Calcular $s =$ longitud de C.
4: Crear el vector nulo $fila$ de longitud n.
5: **para** $k = 0 : n-1$ **hacer**
6: **para** $l = 0 : s-1$ **hacer**
7: Calcular los coeficientes de $C[l]$ como un vector.
8: Calcular $fila[k] = C[l][k] * t^l + fila[k]$.
9: **fin para**
10: **fin para**
11: **devolver** vector $fila$.

Su implementación en SAGE es

```
def phi_inv(g,n,q):
 """Este algoritmo recibe un polinomio g y devuelve un vector u, es decir
    calcula la imagen inversa de g bajo la función phi."""
    F.<t> = GF(q)[]
    C = g.coefficients()
    s = len(C)
    fila = [0 for i in [0..n-1]]
    for k in [0..n-1]:
        for l in [0..s-1]:
            fila[k]=C[l].matrix()[0][k]*t^l+fila[k]
    return fila
```

Algoritmo A.13. (Matriz generadora código convolucional σ-cíclico, `matriz_red`)
Este algoritmo se fundamenta en el item 4 del Teorema 3.2 para el caso particular de un polinomio componente. Así este algoritmo recibe un polinomio componente $g \in R_n[z]$ junto con dos enteros n y q primos relativos entre si y un entero k_l. La salida es una matriz generadora G del submódulo $\langle g \rangle \subseteq \mathbb{F}_q[z]^n$.

Algoritmo A.13 Matriz Generadora CCC

Entrada: Un polinomio componente $g \in R_n[z]$, dos enteros primos relativos n y q junto con un entero k_l .
Salida: Una matriz generadora G del submódulo $\langle g \rangle \subseteq \mathbb{F}_q[z]^n$.

1: $L =$ lista vacía.
2: **para** $i = 0 : kl - 1$ **hacer**
3: Calcular `phi_inv(g*y^i,n,q)` e insertar al final de la lista L.
4: **fin para**
5: Convertir la lista L a una matriz G con entradas en el anillo $\mathbb{F}_q[z]$.
6: **devolver** G.

Su implementación en SAGE es

```
def matriz_red(g,kl,n,q):
 """Este algoritmo recibe un polinomio componente g junto con dos enteros n y
    q primos relativos entre si y un entero kl. La salida es una matriz
    generadora G."""
    L=[]
    for i in [0..kl-1]:
        p=phi_inv(g*y^i,n,q)
        L.append(p)
        R.<t>=GF(q)[]
    return matrix(R,L)
```

Bibliografía

[1] M. Artin. *Algebra*. Pearson Prentice Hall, second edition edition, 2011. 24

[2] J. H. Castillo, G. García-Pulgarín, and J. Velásquez-Soto. De los números de midy a la primalidad. *Revista Integración*, 33(1):1–10, 2015. 92

[3] A. Dholakia. *Introduction to convolutional codes with applications*. In energineering and computer Science. Springer Science + Business Media, LLC. New York, 1994. xvi, 23, 26

[4] P. Elias. The Noisy Channel Coding Theorem for Erasure Channels. *Amer. Math. Monthly*, 81(8):853–862, 1974. xvi

[5] G. D. Forney, Jr. Convolutional codes. I. Algebraic structure. *IEEE Trans. Information Theory*, IT-16:720–738, 1970. xvi, 71

[6] G. D. Forney, Jr. Structural analysis of convolutional codes via dual codes. *IEEE Trans. Information Theory*, IT-19:512–518, 1973. 61

[7] F. R. Gantmacher. *The theory of matrices. Vol. 1*. AMS Chelsea Publishing, Providence, RI, 1998. Translated from the Russian by K. A. Hirsch, Reprint of the 1959 translation. 19, 75

[8] H. Gluesing-Luerssen and B. Langfeld. On the algebraic parameters of convolutional codes with cyclic structure. *J. Algebra Appl.*, 5(1):53–76, 2006. 70, 71, 76, 77, 78, 89

[9] H. Gluesing-Luerssen and W. Schmale. On cyclic convolutional codes. *Acta Appl. Math.*, 82(2):183–237, 2004. xvii, 68, 70, 71, 74, 76, 77, 78, 79, 86, 87, 88

[10] K. R. Goodearl and R. B. Warfield, Jr. *An introduction to noncommutative Noetherian rings*, volume 61 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, second edition, 2004. 22

[11] W. C. Huffman and V. Pless. *Fundamentals of error-correcting codes*. Cambridge University Press, Cambridge, 2003. 25, 73, 93

[12] T. W. Hungerford. *Algebra*, volume 73 of *Graduate Texts in Mathematics*. Springer-Verlag, New York-Berlin, 1980. Reprint of the 1974 original. 24, 85, 86

[13] N. Jacobson. *Basic Algebra*. Number v. 1 in Basic Algebra. W.H. Freeman, 1985. 9

[14] R. Johannesson and K. S. Zigangirov. *Fundamentals of convolutional coding*. IEEE Series on Digital & Mobile Communication. Wiley-IEEE Press, New York, 2015. xvi, 26, 70, 71

[15] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall computer applications in electrical engineering series. Prentice-Hal, 1983. xvi

[16] S. Ling and C. Xing. *Coding theory: A first course*. Cambridge University Press, Cambridge, 2004. 4, 6

[17] R. J. McEliece. The algebraic theory of convolutional codes. In *Handbook of coding theory, Vol. I, II*, pages 1065–1138. North-Holland, Amsterdam, 1998. xvi, 26

[18] I. Niven. Formal power series. *Amer. Math. Monthly*, 76:871–889, 1969. 24

[19] P. Piret. Structure and constructions of cyclic convolutional codes. *IEEE Trans. Information Theory*, IT-22(2):147–155, 1976. 74, 76

[20] P. Piret. *Convolutional codes*. MIT Press, Cambridge, MA, 1988. An algebraic approach. xvi

[21] C. Roos. On the structure of convolutional and cyclic convolutional codes. *IEEE Trans. Inform. Theory*, 25(6):676–683, 1979. 77, 79, 90

[22] J. Rosental and R. Smarandache. Maximum distance separable convolutional codes. *Appl. Algebra Engrg. Comm. Comput.*, (10):15–35, 1999. 81

[23] J. Rosenthal. Connections between linear systems and convolutional codes. In *Codes, systems, and graphical models (Minneapolis, MN, 1999)*, volume 123 of *IMA Vol. Math. Appl.*, pages 39–66. Springer, New York, 2001. 70

[24] J. Rosenthal, J. M. Schumacher, and E. V. York. On behaviors and convolutional codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1881–1891, 1996. Codes and complexity. 70

[25] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948. xvi

[26] V. Shevelev, G. García-Pulgarín, J. M. Velásquez-Soto, and J. H. Castillo. Overpseudoprimes, and mersenne and fermat numbers as primover numbers. *Journal Of Integer Sequences*, 15(7):1–10, 2012. 92, 93

[27] D. Slepian. Some further theory of group codes. *Bell System Tech. J.*, 39:1219–1252, 1960. 6

[28] M. Ventou. Automorphisms and isometries of some modular algebras. In *Algebraic algorithms and error correcting codes (Grenoble, 1985)*, volume 229 of *Lecture Notes in Comput. Sci.*, pages 202–210. Springer, Berlin, 1986. 86