UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

IMPLEMENTACION Y ANALISIS COMPARATIVO DE DISTINTOS METODOS DE DETECCION Y CORRECCION DE ERRORES

T E S I S

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A :

LUIS MANUEL VAZQUEZ NICOLAS

FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS: M. EN C. JOSE DE JESUS GALAVIZ CASAS

2005

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

Implementación y análisis comparativo de distintos métodos de detección y corrección de errores

realizado por Luis Manuel Vázquez Nicolás

con número de cuenta 09510155-2 , quien cubrió los créditos de la carrera de:

Licenciado en Ciencias de la Computación

Dicho trabajo cuenta con nuestro voto aprobatorio.

A t e n t a m e n t e

Director de Tesis Propietario	M. en C. José de Jesús Galaviz Casas
Propietario	Dra. Elisa Viso Gurovich
Propietario	Dr. Sergio Rajsbaum Gorodezky
Suplente	M. en I. María de Luz Gasca Soto
Suplente	Lic. en C. C. Francisco Lorenzo Solsona Cruz

Consejo Departamental de

Dr. Francisco Hernández Quiroz

Para mis papás

Silvestre

y

María Luisa

Para mis hermanas

Lizbeth

y

Ana María

Agradecimientos

Hay muchas personas a las que quisiera agradecer. Sin embargo, en este momento puedo olvidar algunas. Si esto ocurre, de antemano mil disculpas.

Quisiera empezar agradeciendo a la Universidad Nacional Autónoma de México, en particular a la Facultad de Ciencias en la cual obtuve la formación y la oportunidad de estudiar Ciencias de la Computación.

Enseguida deseo agradecer a mi apreciable director de tesis, el M. en C. José de Jesús Galavíz Casas, por su paciencia, sus consejos y el inmejorable asesoramiento que me brindó. Además no puedo dejar de agradecer cada clase que tomé con él.

También deseo agradecer a mis sinodales el Dr. Sergio Rajsbaum, la M. en C. María de Luz Gasca y el Lic. Francisco Lorenzo Solsona por su disposición y sus invaluables consejos que me proporcionaron en este trabajo. No quisiera desaprovechar esta oportunidad para agradecer a la Dra. Elisa Viso no solamente por sus consejos en la revisión de este trabajo si no también por sus importantes enseñanzas en los dos cursos de la licenciatura en los que tuve el privilegio de ser su alumno.

Ahora deseo agradecer a mis maravillosos papás Ma. Luisa y Silvestre a los que ni escribiendo mil líneas podría terminar de agradecer su comprensión, apoyo, cariño y tantas, tantas cosas. Naturalmente, agradezco a mis hermanas Ana María y Lizbeth por su compañía, apoyo, enorme cariño, consejos y por estar siempre conmigo, las quiero muchísimo.

Deseo agradecer a mis tíos Guadalupe, Cecilio y Eziquio por su inestimable apoyo y consejos.

No quisiera dejar de mencionar a mi queridísimo amigo Alberto por haber escuchado muchas veces lo que estaba pensando, aún cuando aparentemente carecía de lógica y darme su apoyo siempre.

Le doy las gracias a mi niño Lucky por su comprensión al resignarse a verme sentado mucho tiempo frente a la computadora, escuchar las versionas previas y aceptar que no puedo darle el tiempo que sin duda merece. También agradecer y disculparme con mi otro niño que ahora ya no está.

Y dejando lo principal para el final, deseo agradecer a Dios.

Índice general

Índice de figuras

Introducción

La teoría de códigos e información encuentra sus bases establecidas en el artículo *"A Mathematical Theory of Communication"*, por C. E. Shannon, publicado en 1948 [16]. Conociendo el hecho que los medios de transmisión (canales) no son ciento por ciento confiables, podemos considerar que el principal problema de dicha teoría es: ¿cómo saber en qué momento ocurrió un error? y ¿existe la posibilidad de recuperar la información original?

A partir de la publicación de este artículo se han ido desarrollando elementos que proponen distintas formas de atacar el problema.

Hoy en día estas preguntas cobran mayor importancia. En la misma medida en que las comunicaciones cambian de análogicas a digital, se busca el uso de los códigos binarios para la detección y recuperación de errores.

Nuestro propósito con este trabajo es buscar el comportamiento que se presenta con los siguientes códigos detectores y correctores de errores: Hamming, Golay, Reed-Muller y Reed-Solomon. Estos códigos han sido ampliamente usados para distintas aplicaciones con el objetivo de minimizar las probabilidades de error. Nuestra intención es implementarlos y observar su comportamiento cuando son sometidos al ruido blanco y al ruido de ráfagas, en particular.

Capítulo 1

Elementos básicos de la Teoría de Códigos e Información

1.1. Introducción

Existen tres razones principales para codificar datos:

- Eficiencia. Relacionada con la teoría de la información.
- Corrección y detección de errores. Relacionada con la teoría de códigos.
- Ocultamiento de la información. Relacionada con la criptología.

Nuestro trabajo se concentra en la teoría de códigos.

1.2. Cadenas y códigos

Para que podamos iniciar este estudio, necesitamos establecer los conceptos básicos para tener un lenguaje común al describir el desarrollo y el tratamiento de los problemas más adelante. Por tal motivo presentamos las siguientes definiciones que siguen el enfoque de Steven Roman [15].

Definición 1.2.1. Una *cadena* es una sucesión finita de elementos de un conjunto $S = \{s_1, s_2, ..., s_n\}$ de n símbolos distintos, al cual se le denomina *alfabeto*.

Cabe señalar que no nos interesa la semántica del mensaje y sólo nos interesan los símbolos como unidades de transmisión.

Dado que la cadena es finita podemos hablar de su longitud, es decir, si tenemos una cadena $\mathbf{x} = x_1x_2...x_k$ entonces la longitud de $\mathbf{x}$ es el número de símbolos que forman la cadena y la denotaremos como $\mathbf{len(x) = k}$.

La operación fundamental entre las cadenas es la *concatenación*, la cual definimos enseguida.

Definición 1.2.2. La *concatenación* es una operación binaria sobre un conjunto de cadenas **W** tal que:

$$\mathrm{m}:\mathbf{W} \times \mathbf{W} \rightarrow \mathbf{W}$$

El efecto de concatenar una cadena **x** con la cadena **z** se logra haciendo seguir a la cadena **x** con los símbolos de **z**.

Con base en lo anterior, definimos:

Definición 1.2.3. Sea $\mathbf{w} = \mathbf{xz}$

- **x** es un *prefijo* de **w**.
- **z** es un *sufijo* de **w**.

Definición 1.2.4. La *cardinalidad* de un conjunto de cadenas A, es el número de elementos que pertenecen al conjunto A y se denota como $\mid A \mid$.

Definición 1.2.5. Definimos el conjunto A^n como el conjunto de todas las cadenas de longitud n que comparten el mismo alfabeto.

1.2.1. Distancia de Hamming

Es conveniente que dispongamos de una medida que indique que tan distintas son dos cadenas entre sí, que tan alejadas están. Para ello utilizaremos la *distancia de Hamming*.

Definición 1.2.6. La *distancia de Hamming* entre dos cadenas cualesquiera $\mathbf{x}, \mathbf{y} \in S^n$, denotadas con $\mathrm{d}(\mathbf{x},\mathbf{y})$, es el número de posiciones en las que **x** difiere de **y**.

La distancia de Hamming es una verdadera métrica para S^n, dado que satisface las siguientes tres propiedades:

1. Positiva definida.
   $\mathrm{d}(\mathbf{x},\mathbf{y}) \geq 0$ y $\mathrm{d}(\mathbf{x},\mathbf{y}) = 0$ sí y solo si $\mathbf{x} = \mathbf{y}$
2. Simétrica.
   $\mathrm{d}(\mathbf{x},\mathbf{y}) = \mathrm{d}(\mathbf{y},\mathbf{x})$
3. Desigualdad del triángulo.
   $\mathrm{d}(\mathbf{x},\mathbf{z}) \leq \mathrm{d}(\mathbf{x},\mathbf{y}) + \mathrm{d}(\mathbf{y},\mathbf{z})$

1.2.2. Códigos

Definición 1.2.7. Un *código* es un conjunto cualquiera de cadenas sobre un cierto alfabeto.

Definición 1.2.8. Definimos a A^* como el conjunto de todas las palabras sobre A incluyendo la palabra de longitud cero.

Definición 1.2.9. Sea $A = \{a_1, a_2, ..., a_r\}$ un conjunto finito de r símbolos al que llamaremos *alfabeto del código*. Un código $r-$ario sobre A es un subconjunto C de A^*. El número r es llamado la base del código.

Ejemplo 1.2.10. Si $A = \{0, 1\}$, un ejemplo de un código binario es $C = \{0, 10, 11\}$.

Hay ciertos tipos de códigos que son convenientes porque nos facilitan la tarea de interpretar las palabras, es decir, el proceso de decodificación. Una característica de gran utilidad es que un código sea unívocamente decodificable.

Definición 1.2.11. Un código es *unívocamente decodificable* si no hay dos sucesiones diferentes de palabras de código que representan la misma cadena sobre A, siendo A el alfabeto del código.

Ejemplo 1.2.12. $C = \{0, 01, 001\}$ es unívocamente decodificable.

Definición 1.2.13. Un código *instantáneo* es un código unívocamente decodificable en el que la decodificación puede hacerse leyendo sucesivamente de izquierda a derecha, la palabra de código.

Ejemplo 1.2.14. $C = \{0, 10, 110\}$ es un código instantáneo.

En los códigos instantáneos ninguna palabra completa del código constituye un prefijo, es decir, una subcadena inicial de otra palabra más larga.

Diremos que un código tiene la propiedad de prefijo si ninguna palabra de código es prefijo de otra palabra de código.

Con estos elementos cubrimos los conceptos esenciales al hablar sobre los códigos en general. En el siguiente capítulo presentaremos los conceptos relacionados con la corrección y detección de errores.

Capítulo 2

Corrección y detección de errores

2.1. Introducción

Sabemos que los canales no son 100% confiables. Por tal motivo, debemos aprender a trabajar con los errores, de tal forma que podamos minimizar sus consecuencias. En este capítulo estudiaremos cómo se pueden detectar los errores y cuándo es posible corregirlos.

Podemos suponer que los símbolos viajan uno a la vez a través del canal de transmisión, esto es, de forma serial. Así que cada uno de ellos se ve expuesto a la posibilidad de ser alterado individualmente. Si nosotros al recibirlos, sólo los consideramos de forma individual, no tenemos forma de detectar si ocurrió un error o no. Por ejemplo, considérese un canal simétrico binario, como el ilustrado en la Figura 2.1, si se envía un cero y se recibe un uno, el receptor no se da cuenta del que se cometió un error, porque recibir un uno es tan válido como recibir un cero. Por otro lado, si consideramos grupos de símbolos entonces podemos decir si el grupo de símbolos que llegó es válido.

Resulta conveniente entonces considerar a los símbolos elementales agrupados en palabras o bloques de longitud fija. A un código constituido de esta manera se le denomina código de bloque.

Definición 2.1.1. Un *código de bloque* $r-$ario C sobre un conjunto A, es un subconjunto no vacío de A^n. Los elementos de C se denominan *palabras del código*, y n es la longitud del código.

A un código con M palabras de longitud n sobre un alfabeto de r símbolos se le denomina un *(n,M)-código r-ario.*

Una vez que establecemos un código de bloque sobre el canal, estamos determinando el conjunto de palabras válidas, esto es, quiénes son los grupos válidos de símbolos.

Definición 2.1.2. Un *canal de comunicación* consiste de un *alfabeto del canal* finito, $A = \{a_1, ..., a_r\}$ y un conjunto de *probabilidades del canal* [1], $P(a_j \text{ recibido} \mid a_i \text{ enviado})$ que satis-

[1] También se les llama probabilidades hacia adelante o probabilidades de transición.

facen

$$\sum_{j=1}^{r} P(a_j \text{ recibido} \mid a_i \text{ enviado}) = 1, \quad \forall i.$$

Cabe señalar que estamos suponiendo que estas probabilidades no cambian con el tiempo.

Definición 2.1.3. Se dice que un canal de comunicación es *sin memoria* si la salida de cualquier símbolo que sea transmitido es independiente de la salida del símbolo transmitido previamente. En términos más formales, si $\mathbf{c} = c_1...c_n$ y $\mathbf{x} = x_1...x_n$ son palabras de longitud n sobre el alfabeto A, la probabilidad $P(\mathbf{x} \text{ recibido} \mid \mathbf{c} \text{ enviado})$ es el producto:

$$P(\mathbf{x} \text{ recibido} \mid \mathbf{c} \text{ enviado}) = \prod_{i=1}^{n} P(x_i \text{ recibido} \mid c_i \text{ enviado}).$$

2.2. Tipos de canal

Existen distintos tipos de canales. De hecho, la mayoría de los canales que se presentan en la realidad difieren de los modelos más comunes que se manejan en la teoría de códigos. Sin embargo, estos modelos son de suma utilidad para entender el comportamiento de los canales reales.

Las Figuras 2.1 y 2.2 presentan dos tipos de canales básicos: el canal simétrico binario y el canal de borrado binario.

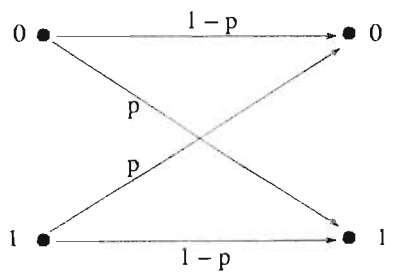


Figura 2.1: Canal simétrico binario.

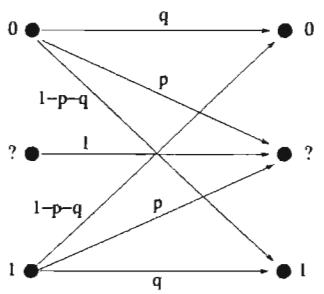


Figura 2.2: Canal de borrado binario.

La *matriz del canal* está definida por las probabilidades del canal, que se colocan de la siguiente forma:

$$\begin{bmatrix} p(y_1|x_1) & p(y_2|x_1) & \dots & p(y_t|x_1) \\ p(y_1|x_2) & p(y_2|x_2) & \dots & p(y_t|x_2) \\ & & \vdots & \\ p(y_1|x_s) & p(y_2|x_s) & \dots & p(y_t|x_s) \end{bmatrix}$$

Obsérvese que cada renglón de la matriz del canal corresponde a un solo símbolo de entrada, y cada columna corresponde a un único símbolo de salida.

Se puede usar esta matriz para definir algunos tipos especiales de canales.

Definiciones

Sean **X** y **Y** variables aleatorias que corresponden a la entrada y a la salida, respectivamente.

1. Intuitivamente, un canal es *sin pérdida* si la entrada **X** está completamente determinada por la salida **Y**. Concretamente, un canal es *sin pérdida* si se cumple que:

   a) Existen subconjuntos ajenos no vacíos $B_1, ..., B_s$ del alfabeto de salida con la propiedad que $P(Y \in B_i \mid X = x_i) = 1$ donde $i = 1, ..., s$.

   b) Para cualquier distribución de probabilidad del alfabeto de entrada, si $\mathrm{p}(y_j) \neq 0$ entonces existe un x_i para el cual $\mathrm{p}(x_i|y_j) = 1$.

2. Intuitivamente, un canal es *determinístico* si la salida **Y** está completamente determinada por la entrada **X**. Concretamente, un canal es *determinístico* si se cumple que:

   a) Para toda x_i existe una y_j para el cual $\mathrm{p}(y_j|x_i)$=1.

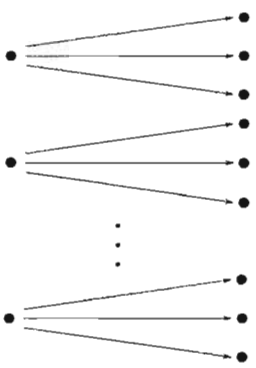
3. Un canal es *sin ruido* si es tanto sin pérdida como determinístico. Equivalentemente, un canal es *sin ruido* si existe una función inyectiva φ del alfabeto de entrada $\{x_1, \dots, x_s\}$ al alfabeto de salida $\{y_1, \dots, y_t\}$ para la cual $\mathrm{p}(\varphi(x_i)|x_i) = 1$ para toda i.

4. Intuitivamente, un canal es *inútil* si el conocimiento sobre la entrada **X** no nos dice nada sobre la salida **Y**. Específicamente un canal es *inútil* si cualquiera de las siguientes condiciones se cumplen.

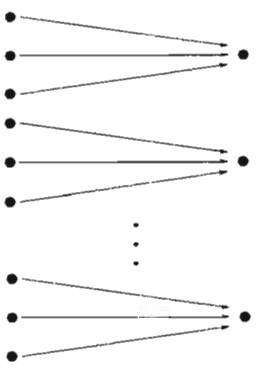
   a) Los renglones de la matriz del canal son iguales.

   b) La entrada **X** y la salida **Y** son independientes para todas las distribuciones de probabilidad sobre la entrada.

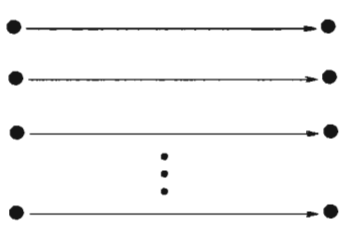
En la Figura 2.3 se ilustran estas definiciones.



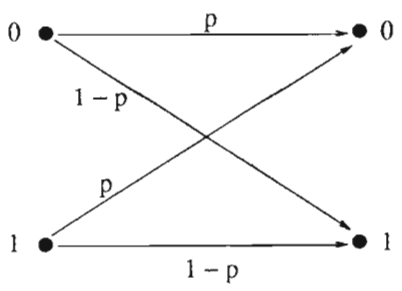
Canal sin pérdida.



Canal determinístico.



Canal sin ruido.



Canal inútil.

Figura 2.3
Tipos de canales

2.3. Canal de Gilbert-Elliot

En los tipos de canales anteriores, la posibilidad de que un bit sea alterado es independiente de las posibles alteraciones de los bits adyacentes. Sin embargo, a veces durante una transmisión, los errores que ocurren no son independientes entre sí. De tal suerte que si un bit es alterado entonces aumenta la probabilidad de que los bits consecutivos sean alterados también hasta llegar a un momento en donde la probabilidad de alteración va disminuyendo hasta que nuevamente la transmisión es con muy pocos errores. A este bloque de bits erróneos consecutivos se le conoce como *ráfaga*[2].

El modelo del canal de Gilbert-Elliot se propone simular el comportamiento de un canal con ráfagas. El modelo que aquí presentamos está basado en el artículo de Mushkin [12] y en el material de Sun [17]. Este modelo consiste en una cadena de Markov, donde hay dos estados, vamos a llamarlos el estado *bueno* y el estado *malo*. En cada estado hay un canal simétrico binario. La probabilidad de cruzamiento del estado bueno es muy baja y la probabilidad de cruzamiento del estado malo es muy alta. A estas probabilidades las denotaremos como P_{c_1} y P_{c_2}, respectivamente.

Además de las probabilidades anteriores, existen otras cuatro probabilidades, que se muestran en el Cuadro 2.1. Este modelo se ilustra en la Figura 2.4.

p_{bb}	Probabilidad de pasar del estado bueno al estado bueno.
p_{bm}	Probabilidad de pasar del estado bueno al estado malo.
p_{mm}	Probabilidad de pasar del estado malo al estado malo.
p_{mb}	Probabilidad de pasar del estado malo al estado bueno.

Cuadro 2.1: Probabilidades de transición.

En este modelo se parte del hecho que la probabilidad de pasar del estado bueno al estado malo es baja, y la probabilidad de pasar del estado malo al estado bueno es alta. Esta última probabilidad determinará la longitud de la ráfaga, entre más alta sea, la longitud de la ráfaga aumentará.

El modelo de Gilbert-Elliot es una cadena de Markov ergódica, como se señala en [12], porque:

- Sus dos estados se intercomunican, es decir, la probabilidad de transición entre ellos es distinta de cero.
- Es *acíclica* porque tiene ciclos de período uno.

Por tratarse de una cadena de Markov ergódica entonces

$$\lim_{n\to\infty} P_{ij}^n = \Pi_j \quad \text{y} \quad \sum_j \Pi_j = 1$$

donde P_{ij} es la entrada (i, j) en la matriz de transiciones y Π_j es la probabilidad de estar en el estado j.

[2]Es más conocido por el término en inglés *burst error*.

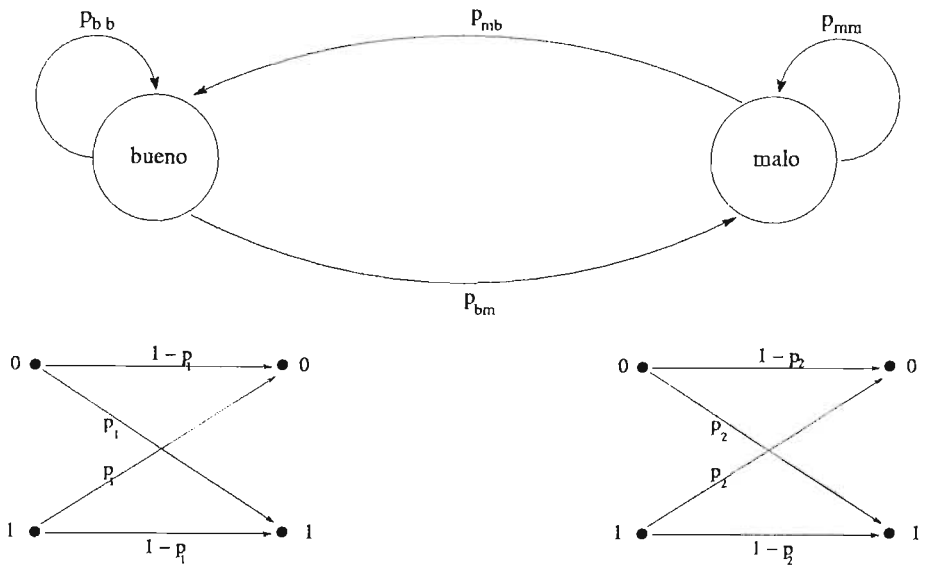


Figura 2.4: Canal de Gilbert-Elliot.

Las ráfagas que afectan a los canales son ocasionadas por causas externas, que pueden ir desde factores del clima, como la temperatura, el viento, las descargas eléctricas o bien interferencias con otras transmisiones.

2.4. Regla de decisión

Cuando estamos transmitiendo la información a través del canal, el receptor debe decidir cómo interpretar las palabras que están llegando (independientemente de que sean correctas o no), es decir, el receptor debe hacer una suposición plausible acerca de cuál de las palabras válidas le fué enviada, con base en la palabra recibida.

Definición 2.4.1. Sea C un (n, M)-código sobre el alfabeto A. Supóngase que la palabra de código c no contiene el símbolo "?" (en caso que lo contenga, simplemente se reemplaza por otro símbolo). Una *regla de decisión* para C es una función $f : A^n \to C \cup \{?\}$. Al proceso de aplicar la regla de decisión se le llama *decodificación.* Si $\mathbf{x}$ es una palabra en A^n entonces la regla de decisión f *decodifica* $\mathbf{x}$ en la palabra de código $f(\mathbf{x})$ en caso de que $f(\mathbf{x}) \in C$. De lo contrario, si $f(\mathbf{x}) = ?$ entonces se declara un *error de decodificación.* Este concepto se ilustra en la Figura 2.5.

Para enunciar este concepto de una forma más intuitiva, diremos que una regla de decisión es un procedimiento que decide, dada la cadena recibida, qué cadena de código le corresponde o si han ocurrido errores.

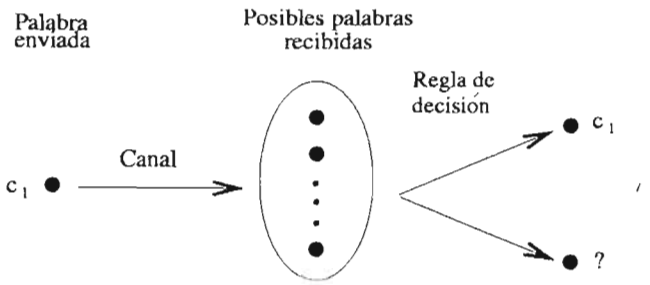


Figura 2.5: Regla de decisión.

La elección de la mejor regla de decisión depende de las características del canal y debemos mantener en mente que *nunca* podemos estar completamente seguros de que la regla de decisión entrega la palabra de código que realmente fué enviada.

Naturalmente, deseamos tener la regla de decisión que se equivoque lo menos posible, es decir, dado que se envió la palabra $\mathbf{c}$ y se recibió una determinada palabra $\mathbf{d}$, deseamos que $f(\mathbf{d}) = \mathbf{c}$ la mayor parte de las veces. Y como queremos que se equivoque lo menos posible, necesitamos calcular que tan probable es decodificar correctamente. Así tenemos que

$$P(\textit{decodificar correctamente}) = \sum_{x \in A^n} P(f(x) \textit{ enviado} \mid x \textit{ recibido}) P(x \textit{ recibido})$$

Definición 2.4.2. Definimos al *observador ideal* como cualquier esquema de decisión f con la propiedad de que

$$P(f(x) \textit{ enviado} \mid x \textit{ recibido}) = \max_{\mathbf{c} \in C} \{P(\mathbf{c} \textit{ enviado} \mid x \textit{ recibido})\}$$

para todas las posibles cadenas recibidas x.

Sin embargo, la principal desventaja del observador ideal es que depende de la distribución de probabilidad del alfabeto de entrada. Por lo tanto si esta distribución cambia es probable que cambie también el observador ideal. Por otro lado, al pretender implementarlo, nos damos cuenta que necesitamos almacenar $|A^n|$ probabilidades por palabra de código.

Una forma de evitar esta dependencia es suponer una distribución de probabilidad uniforme. Es decir,

$$P(\mathbf{c} \textit{ enviado}) = \frac{1}{M},$$

donde M es el tamaño del código.

Esto nos lleva a la siguiente definición.

Definición 2.4.3. Una *regla de decisión de máxima verosimilitud* es una regla de decisión f que maximiza las probabilidades del canal, es decir,

$$P(x \text{ recibido} \mid f(x) \text{ enviado}) = \max_{\mathbf{x} \in C} P(x \text{ recibido} \mid \mathbf{c} \text{ enviado})$$

para toda $x \in A^n$.

De tal manera que cuando la distribución de probabilidad de entrada es uniforme, el observador ideal es aquel que aplica la regla de decisión de máxima verosimilitud.

2.5. Decodificación del vecino más cercano

Ahora que contamos con el concepto de la regla de decisión de máxima verosimilitud, vamos a usarlo. En particular con el canal simétrico binario.

Como los canales de comunicación no son tan malos, podemos asumir que la probabilidad de que ocurra un error (probabilidad de cruzamiento) es menor que $\frac{1}{2}$. De hecho, un canal que tuviera una probabilidad de cruce ligeramente inferior a $\frac{1}{2}$ sería extremadamente malo. Por lo tanto, la probabilidad de que un símbolo se reciba correctamente es $1 - p$. Así, la probabilidad de que una palabra se reciba sin errores es

$$P(\textit{palabra correcta}) = (1-p)^n$$

La probabilidad de que una palabra de código $\mathbf{c}$ difiera de una palabra $\mathbf{x}$ en exactamente k posiciones[3] es:

$P(k$ símbolos erróneos en lugares específicos$) = p^k(1-p)^{n-k}$

De tal manera que si la palabra de código $\mathbf{c}$ difiere de una palabra $\mathbf{x}$ en exactamente k posiciones entonces

$$P(\mathbf{x} \textit{ recibida} \mid \mathbf{c} \textit{ enviada}) = p^k(1-p)^{n-k}$$

A continuación enunciaremos un teorema importante en relación con la utilidad de la distancia de Hamming. El lector puede consultar la demostración en [15].

Teorema 2.5.1. Para un canal simétrico binario con probabilidad de error $p < 1/2$, la regla de decisión de máxima verosimilitud es la que elige la palabra de código cuya distancia de Hamming a la palabra recibida $\mathbf{x}$ es mínima.

A las palabras de código con mínima distancia de una palabra $\mathbf{x}$ se les denomina las *vecinas más cercanas de* x y la regla de decisión que se acaba de mencionar se le denomina por lo tanto *de vecino más cercano.*

[3] Obsérvese que en la siguiente fórmula se hace hincapié de que se tratan de lugares específicos. De lo contrario, tendríamos que calcular $\binom{n}{k}$ y multiplicarlo por la fórmula que se presenta.

2.6. La distancia mínima de un código

Al hablar de la regla de decisión del vecino más cercano, nos empezamos a dar cuenta que nos conviene que las palabras del código sean tan diferentes como sea posible entre ellas mismas, es decir, que la distancia de Hamming entre ellas sea lo suficientemente grande.

Definición 2.6.1. Sea u un entero positivo. Un código C es *detector de u errores* si para cualquier palabra que presente de 1 a u errores, dicha palabra *no* es una palabra de código. Un código C es *detector de exactamente u errores* si es detector de u errores, pero no de $u+1$ errores.

Definición 2.6.2. Sea v un entero positivo. Un código C es *corrector de v errores* si al decodificar con la regla de decisión del vecino más cercano, es posible corregir v errores o menos. Un código es *exactamente corrector de v errores* si es corrector de v errores, pero no de $v+1$ errores.

Así, la cantidad de errores que pueden detectarse o corregirse depende de la distancia mínima entre cualesquiera dos palabras del código.

Definición 2.6.3. Sea C un código con al menos dos palabras. La *distancia mínima del código*, que denotaremos con $d(C)$ es la distancia más pequeña entre las distintas palabras del código:

$$d(C) = min\{d(\mathbf{c_i}, \mathbf{c_j}) \mid \mathbf{c_i}, \mathbf{c_j} \in C, \mathbf{c_i} \neq \mathbf{c_j}\}$$

dado que $\mathbf{c_i} \neq \mathbf{c_j}$, $d(C) \geq 1$.

Definición 2.6.4. Un código $r-$ario con M palabras diferentes, de longitud n y distancia mínima d es un (n, M, d)-código $r-$ario.

Por último, tenemos el siguiente teorema que será muy útil en este trabajo. Su demostración puede ser consultada en [14].

Teorema 2.6.5. Si C es un (n, M, d)-código entonces es detector de exactamente $d-1$ errores y corrector de exactamente $\lfloor \frac{d-1}{2} \rfloor$.

2.7. Códigos maximales

Un código maximal es aquel que posee todas las palabras de código posibles, dadas ciertas características.

Definición 2.7.1. Un (n, M, d)-código es maximal si no está contenido en un código más grande con la misma distancia mínima, es decir, si no está contenido en un $(n, M+1, d)$-código.

La demostración del siguiente teorema puede consultarse en [5].

Teorema 2.7.2. Un (n, M, d)-código es maximal si y sólo si, para todas las palabras $\mathbf{x} \in A^n$, hay una palabra de código $\mathbf{c}$ con la propiedad que $d(\mathbf{x}, \mathbf{c}) < d$.

2.8. Probabilidad de error al decodificar

Naturalmente, lo que pretendemos es poder decodificar tan correctamente como sea posible. Sin embargo, antes debemos precisar que significa "tan correctamente como sea posible".

La probabilidad de que ocurran $k \leq n$ errores en una palabra de longitud n transmitida por un canal simétrico binario con probabilidad de error p es:

$$P_k = \binom{n}{k} p^k (1-p)^{n-k}.$$

Si el lector desea ver la demostración del siguiente teorema, puede consultarla en [14].

Teorema 2.8.1. Para un canal simétrico binario y usando decodificación del vecino más cercano, la probabilidad de error al decodificar una palabra, es decir, la probabilidad de que la palabra decodificada no sea la enviada, satisface:

$$\sum_{k-d}^{n} \binom{n}{k} p^k (1-p)^{n-k} \leq P(error\ decodificacion) \leq 1 - \sum_{k=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{k} p^k (1-p)^{n-k}.$$

2.9. Códigos perfectos y esferas de empaque

Hemos estado hablando sobre la distancia entre dos palabras de código, el siguiente paso es definir qué es una vecindad. El punto alrededor del cual se establece esta vecindad, o más propiamente, la esfera, es una palabra de código $\mathbf{x}$. Y los elementos que caen dentro de esta esfera son las palabras vecinas más cercanas a $\mathbf{x}$.

Definición 2.9.1. Definimos $\mathbb{Z}_r$ como el conjunto de los enteros $\{0, 1, \ldots, r-1\}$.

Definición 2.9.2. Definimos $\mathbb{Z}_r^n$ como el conjunto de todas las cadenas de longitud n cuyo alfabeto es $\mathbb{Z}_r$.

Definición 2.9.3. Sea $\mathbf{x}$ una cadena en $\mathbb{Z}_r^n$ y sea $\rho \geq 0$. La esfera $S_r^n(\mathbf{x}, \rho)$ con centro en $\mathbf{x}$ y radio ρ es el conjunto de todas las cadenas en $\mathbb{Z}_r^n$ cuya distancia a $\mathbf{x}$ es, a lo más ρ. En símbolos,

$$S_r^n(\mathbf{x}, \rho) = \{y \in \mathbb{Z}_r^n \mid d(\mathbf{x}, \mathbf{y}) \leq \rho\}$$

Definición 2.9.4. El volumen de una esfera $S_r^n(\mathbf{x}, \rho)$ es el número de cadenas en ella.

Así que el volumen de una esfera de radio ρ es:

$$V_r^n(\rho) = \sum_{k=0}^{\rho} \binom{n}{k} (r-1)^k;$$

en $\mathbb{Z}_2^n$

$$V_r^2(\rho) = \sum_{k=0}^{\rho} \binom{n}{k}.$$

Definición 2.9.5. Sea C un (n, M, d)-código r-ario. El *radio de empaque* de C, que denotamos como $Pr(C)$, es el radio más grande posible para un conjunto de esferas disjuntas centradas en una palabra de código. Al conjunto de esferas $S_r^n(c_i, Pr(C))$ $(c_i \in C)$, se le denomina las *esferas de empaque* para C.

En las Figuras 2.6 y 2.7 queremos ilustrar el concepto de las esferas de empaque, donde los puntos negros indican las palabras de código y los círculos alrededor de ellos son las esferas.

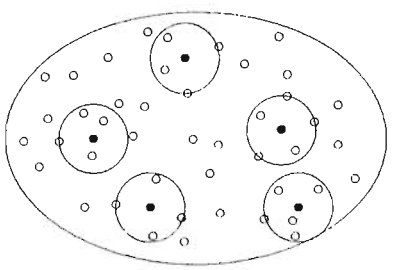


Figura 2.6: Esferas de empaque.

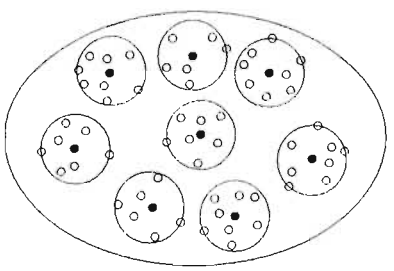


Figura 2.7: Código perfecto.

Las demostraciones que corresponden tanto al teorema como al corolario siguientes, pueden ser consultadas en [14].

Teorema 2.9.6. El radio de empaque de un (n, M, d)-código es $\mathrm{Pr(C)} = \lfloor \frac{d-1}{2} \rfloor$.

Corolario 2.9.7. Un código es corrector de exactamente v errores si y sólo si $\mathrm{Pr}(C) = v$.

Definición 2.9.8. Un (n, M, d)-código r-ario $C = \{c_1, ..., c_M\}$ con alfabeto $\mathbb{Z}_r$ es *perfecto* si las esferas de empaque son una partición de $\mathbb{Z}_r^n$.

Entonces, un código es perfecto si no hay palabras $\mathbf{x} \in A^n$ que queden fuera de alguna esfera de empaque.

2.10. Equivalencia de códigos

Definición 2.10.1. Dos códigos son equivalentes si uno puede ser obtenido del otro por una combinación de operaciones de los dos tipos siguientes:

1. Permutar las posiciones de todas las palabras del código.
2. Dada una posición específica, aplicar una permutación a los símbolos en esa posición.

Afortunadamente, podemos encontrar una forma más clara de dar esta definición en [5], la cual presentamos a continuación:

Supongamos que tenemos un (n, M, d)-código r-ario C y que acomodamos cada una de las palabras del código en un renglón de una matriz de n columnas y M renglones a la que llamaremos $\mathcal{M}$. Las reglas del juego de equivalencia son las siguientes:

1. Cambiar el orden de las columnas de la matriz.
2. Dada una columna y una permutación de los símbolos en el alfabeto A del código (es decir una función biyectiva de A en A): aplicar la permutación a los elementos de la columna.
3. Cambiar un renglón por otro. Esta última regla no aparece arriba, pero es evidente, solo altera el orden en el que son listadas las palabras del código.

Teorema 2.10.2. Si el alfabeto A de un código contiene el símbolo 0 entonces cualquier código sobre A es equivalente a uno que contiene la palabra $\mathbf{0}$.

2.11. Tasa de transmisión y tasa de corrección

Hay un compromiso entre las dos cualidades deseables en un código: *eficiencia* y *robustez.*

Definición 2.11.1. La *tasa de transmisión* de un (n, M)-código C, $r-$ario es:

$$R(C) = \frac{log_r(M)}{n} \leq 1.$$

Definición 2.11.2. La *tasa de corrección de error* de un (n, M, d)-código C, es:

$$\delta(C) = \frac{\lfloor \frac{d-1}{2} \rfloor}{n},$$

el número de errores que pueden corregirse en cada palabra entre el tamaño total de la palabra.

Este tipo de medidas son importantes, pues nos ayudan a distinguir de una manera proporcional la calidad de envío de información por un lado y la capacidad de corregir errores por otro. Sin embargo, podríamos decir, que ya sabemos medir cuántos errores puede corregir un código conociendo la distancia mínima d. Bajo este contexto, podríamos pensar a primera vista que si

un código corrige cien errores es un buen código, pero pensemos por ejemplo que se trataba de una palabra de longitud mil. Entonces, ya no lo consideraríamos una alta corrección de errores. Por este motivo necesitamos este tipo de tasas, para observar las medidas de forma proporcional.

Capítulo 3

Códigos lineales

3.1. Introducción

Si bien conocemos los resultados más importantes para la detección y corrección de errores, existe un tipo de códigos que ha sido ampliamente estudiado, los códigos lineales. De hecho, esta tesis se propone estudiar algunos de estos códigos. En este capítulo, nos vamos a concentrar más en entenderlos que en ser rigurosos con las demostraciones, las cuales pueden ser consultadas en [14, 15].

En general, los códigos lineales son más fáciles de describir que los otros códigos, es decir, es más fácil implementar la codificación y decodificación de los mensajes.

Teorema 3.1.1. Si p es un primo, el conjunto $\mathbb{Z}_p^n$ (las cadenas de longitud n sobre $\mathbb{Z}_p$), junto con las operaciones de adición y de multiplicación escalar de $\mathbb{Z}_p$, es un *espacio vectorial* sobre $\mathbb{Z}_p$.

Definición 3.1.2. Un subconjunto S no vacío de $\mathbb{Z}_p^n$ es un subespacio vectorial de $\mathbb{Z}_p^n$ si el conjunto S, junto con las operaciones de adición y multiplicación por escalar, heredadas de $\mathbb{Z}_p^n$, es en sí mismo un espacio vectorial.

3.2. Códigos lineales

Definición 3.2.1. Un código C es lineal si $C \subseteq \mathbb{Z}_p^n$, es decir si C es un subespacio vectorial de $\mathbb{Z}_p^n$. Si C tiene dimensión k y distancia mínima $d(C) = d$ entonces C es un **[n,k,d]-código**.

Teorema 3.2.2. Un *[n,k,d]-código lineal p-ario* tiene p^k palabras de código, esto es, se trata de un *[n,p^k,d]-código lineal.*

3.2.1. El peso del código

Definición 3.2.3. El *peso* de una palabra es el número de símbolos distinto de cero, que contiene la palabra.

Definición 3.2.4. El *peso* de un código C, denotado por $w(C)$, es el mínimo peso entre todas las palabras del código distintas de cero.

Teorema 3.2.5. Si C es un código lineal entonces $d(C) = w(C)$.

3.2.2. La matriz generadora de un código lineal

Otra de las ventajas de trabajar con códigos lineales, es que, como se trata de un espacio vectorial, basta con tener unas cuantas palabras de código que constituyan la base[1] para describir todo el código.

Definición 3.2.6. Sea C un código lineal, con base $B = b_1, b_2, \ldots, b_k$ donde

$$\begin{aligned} b_1 &= b_{11}b_{12}\ldots b_{1n} \\ b_2 &= b_{21}b_{22}\ldots b_{2n} \\ &\vdots \\ b_k &= b_{k1}b_{k2}\ldots b_{kn} \end{aligned}$$

La *matriz generadora* del código C es aquella que tiene como renglones las palabras de código de la base.

$$\begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ & & \vdots & \\ b_{k1} & b_{k2} & \ldots & b_{kn} \end{bmatrix}$$

Teorema 3.2.7. Sea G una matriz con elementos en $\mathbb{Z}_p$. G es una matriz generadora para algún código lineal C sobre $\mathbb{Z}_p$ si y sólo si los renglones de G son linealmente independientes.

3.3. Corrección de errores

En los códigos lineales es más fácil el proceso de decodificación, esencialmente vamos a construir una matriz de la siguiente forma.

Sea $C = \{c_1, c_2, \ldots, c_M\}$ un $[n, k, d]$-código p–ario, con tamaño $M = p^k$, y sea $c_1 = \mathbf{0}$.

Vamos a construir una tabla que contendrá a todo $\mathbb{Z}_p^n$. El primer renglón de la tabla contiene las palabras de código, con c_1 en la primera posición,

0	c_2	c_3	...	c_M

[1] Recuérdese que, de acuerdo con el álgebra lineal, una base de un espacio vectorial V es un conjunto de vectores linealmente independientes de V cuya cardinalidad es igual a la dimensión del espacio vectorial. En nuestro caso, a cada una de las palabras se le considera un vector y el campo sobre el que trabajamos es $\mathbb{Z}_p^n$.

Enseguida, escogemos una cadena de $\mathbb{Z}_p^n$ con el menor peso que *no* aparezca ya en la tabla, llamémosla f_2, y coloquémosla abajo de **0**. Entonces sumamos f_2 a cada una de las palabras de código que están en el primer renglón, y ponemos los resultados en el segundo renglón.

0	c_2	c_3	...	c_M
f_2	$f_2 + c_2$	$f_2 + c_3$	...	$f_2 + c_M$

Otra vez, escogemos la cadena con el menor peso que no aparezca ya en la tabla, la ponemos en la primera columna, y terminamos de construir el resto del renglón sumando f_3 con las palabras de código,

0	c_2	c_3	...	c_M
f_2	$f_2 + c_2$	$f_2 + c_3$	...	$f_2 + c_M$
f_3	$f_3 + c_2$	$f_3 + c_3$	...	$f_3 + c_M$

Continuando de esta forma hasta que esté todo $\mathbb{Z}_p^n$, es decir, hemos contruido la siguiente tabla, la cual recibe el nombre de *arreglo estándar*.

0	c_2	c_3	...	c_M
f_2	$f_2 + c_2$	$f_2 + c_3$	...	$f_2 + c_M$
f_3	$f_3 + c_2$	$f_3 + c_3$	...	$f_3 + c_M$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
f_q	$f_q + c_2$	$f_q + c_3$	...	$f_q + c_M$

A cada renglón del arreglo estándar se le llama **coset**. Y a la primera cadena de cada renglón se le denomina **líder del coset**.

Teorema 3.3.1. Sea C un $[n, k, d]$-código lineal con arreglo estándar A.

1. Cada cadena en $\mathbb{Z}_p^n$ aparece exactamente una vez en A.
2. El número de renglones de A es p^{n-k}.
3. Dos cadenas **x** y **y** en $\mathbb{Z}_p^n$ están en el mismo coset de A si y sólo si su diferencia **x** - **y** es una palabra de código.

Teorema 3.3.2. Sea C un $[n, k]$-código p–ario, con arreglo estándar A. Para cualquier cadena **x** en $\mathbb{Z}_p^n$, la palabra de código **c** que está al principio de la columna es la palabra de código más cercana a **x**.

3.4. La probabilidad de decodificar correctamente

En códigos lineales tenemos una expresión exacta para calcular la probabilidad de decodificar correctamente.

Teorema 3.4.1. Sea C un $[n,k]$-código binario y supóngase que los líderes del coset en un arreglo estándar C son $f_1, f_2, \ldots, f_q$. Entonces, si suponemos un canal binario simétrico con probabilidad de cruzamiento p, la probabilidad que una cadena recibida sea decodificada correctamente es

$$P(\textit{decodificar correctamente}) = \sum_{i=1}^{q} p^{w(f_i)}(1-p)^{n-w(f_i)}$$

Si decimos que w_i es el número de líderes del coset de peso i, entonces esta probabilidad es

$$P(\textit{decodificar correctamente}) = \sum_{i=1}^{n} w_i p^i (1-p)^i$$

3.5. El dual de un código

Hemos visto que usar la estructura de espacio vectorial en los códigos nos ha ayudado a simplificar la forma de codificación, decodificación y la representación del código, entre otras cosas. Al estar manejando los conceptos del álgebra lineal, nos apoyamos en una teoría sólida con muchos resultados ya demostrados. Vamos a usar uno de estos resultados para construir un nuevo código a partir de un código dado: estamos hablando del dual de un código.

Sin embargo, necesitamos establecer previamente los elementos que nos van a permitir llegar a este concepto.

Definición 3.5.1. Sean $\mathbf{x} = x_1x_2\ldots x_n$ y $\mathbf{y} = y_1y_2\ldots y_n \in \mathbb{Z}_p^n$, donde p es un primo. El *producto interno* de $\mathbf{x}$ y $\mathbf{y}$, denotado por $\mathbf{x}\cdot\mathbf{y}$, es el elemento de $\mathbb{Z}_p$ definido por

$$\mathbf{x}\cdot\mathbf{y} = x_1y_1 + x_2y_2 + \ldots + x_ny_n$$

donde tanto la suma como el producto se toman del campo $\mathbb{Z}_p$.

Definición 3.5.2. Sean $\mathbf{x}$ y $\mathbf{y} \in \mathbb{Z}_p^n$. Si $\mathbf{x}\cdot\mathbf{y} = 0$ entonces decimos que son ortogonales.

Ahora, si tomamos una cadena $\mathbf{a} \in \mathbb{Z}_p^n$, denotaremos como $\{\mathbf{a}\}^\perp$ al conjunto de todas las cadenas en $\mathbb{Z}_p^n$ que son ortogonales a $\mathbf{a}$. Es decir,

$$\{\mathbf{a}\}^\perp = \{\mathbf{b} \in \mathbb{Z}_p^n \mid \mathbf{a}\cdot\mathbf{b} = 0\}.$$

A este conjunto se le denomina el *complemento ortogonal* de $\mathbf{a}$.

Teorema 3.5.3. Para cualquier cadena $\mathbf{a} \in \mathbb{Z}_p^n$, el conjunto $\{\mathbf{a}\}^\perp$ es un código lineal.

Generalizando la definición del complemento ortogonal tenemos la siguiente definición.

Definición 3.5.4. Sea $A = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_s}\} \subseteq \mathbb{Z}_p^n$ un código. Al conjunto

$$A^{\perp} = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_s}\}^{\perp} = \{\mathbf{x} \in \mathbb{Z}_p^n \mid \mathbf{x} \cdot \mathbf{a}_i = 0 \textit{ para toda } i\}$$

se le llama el *complemento ortogonal* de A.

Enseguida, exponemos el resultado que es el motivo de las definiciones anteriores.

Teorema 3.5.5. El *complemento ortogonal* $A^{\perp}$ de cualquier código A es un código lineal y se le conoce como *código dual* de A.

La importancia del teorema anterior radica en la posibilidad de construir un código lineal y las ventajas que se tienen al trabajar con este tipo de códigos.

3.6. El síndrome

Uno de los aspectos más importantes que tenemos al trabajar con códigos lineales es el síndrome. El síndrome es una cadena de bits, la cual, si no hubo errores durante la transmisión de la palabra de código, es **0**. Como veremos más adelante, el síndrome puede tener otras propiedades en ciertos tipos de códigos lineales.

Definición 3.6.1. Una *matriz verificadora de paridad* para un $[n, k]$-código lineal C es una matriz P con la propiedad que

$$C = \{x \in \mathbb{Z}_p^n \mid xP^t = 0\}$$

Definición 3.6.2. Sea P una matriz verificadora de paridad de un código lineal $C \subseteq \mathbb{Z}_p^n$. El *síndrome* $S(\mathbf{x})$ de una cadena $\mathbf{x} \in \mathbb{Z}_p^n$ es el producto $\mathbf{x}P^t$.

El siguiente teorema ofrece algunos resultados importantes sobre el síndrome.

Teorema 3.6.3. Sea P una matriz verificadora de paridad de un código lineal $C \subseteq \mathbb{Z}_p^n$ y sea S la función síndrome de P. Entonces, para toda $\mathbf{x}$,$\mathbf{y} \in \mathbb{Z}_p^n$ y $\alpha \in \mathbb{Z}_p$ se cumple:

1. Si P tiene el tamaño $m \times n$ entonces $S(\mathbf{x}) \in \mathbb{Z}_p^m$.
2. $S(\mathbf{x} + \mathbf{y}) = S(\mathbf{x}) + S(\mathbf{y})$.
3. $S(\alpha\mathbf{x}) = \alpha S(\mathbf{x})$.
4. $\mathbf{x} \in C$ si y sólo si $S(\mathbf{x}) = \mathbf{0}$.

A continuación se expone uno de los teoremas más importantes sobre el síndrome.

Teorema 3.6.4. Sea $C \subseteq \mathbb{Z}_p^n$ un código lineal. Dos cadenas $\mathbf{x}$ y $\mathbf{y}$ están en el mismo coset de cualquier arreglo estándar de C si y sólo si tienes el mismo *síndrome*.

Este es un teorema importante porque nos permite simplificar la decodificación de las palabras de código. Por un lado, llamemos $\mathbf{x}$ a la palabra de código que vamos a decodificar. La cadena de error $\mathbf{e}$ tal que $\mathbf{c} = \mathbf{x}$ - $\mathbf{e}$, es el líder del coset. En conclusión, como las cadenas de error siempre van a ser los líderes del coset, ya no necesitamos almacenar todo el arreglo estándar, solo los líderes del coset con su respectivo síndrome. Este arreglo más pequeño recibe el nombre de **tabla del síndrome** de C.

Líder del coset	*Síndrome*
$\mathbf{0}$	$\mathbf{0}$
$\mathbf{f_2}$	$S(\mathbf{f_2})$
$\mathbf{f_3}$	$S(\mathbf{f_3})$
$\vdots$	$\vdots$
$\mathbf{f_q}$	$S(\mathbf{f_q})$

Así, el procedimiento de decodificación se resume en los siguientes pasos:

- Calcular el síndrome $S(\mathbf{x})$ de la palabra recibida.
- Buscar el síndrome $S(\mathbf{x})$ en la *tabla del síndrome* para encontrar el correspondiente *líder del coset*.
- Restar la palabra recibida con el líder del coset para obtener la *palabra de código* $\mathbf{c}$ más cercana a $\mathbf{x}$, $\mathbf{c} = \mathbf{x} - \mathbf{f_i}$.

Habiendo terminado con la presentación de los códigos lineales vamos a estudiar un enfoque basado en polinomios de estos códigos en el siguiente capítulo.

Capítulo 4

Códigos cíclicos y polinomios

4.1. Polinomios

Veremos que es conveniente manejar con polinomios la representación de algunos códigos para comprenderlos y deducir propiedades de una forma más adecuada. Queremos aclarar que no tenemos la intención de extendernos en este tema sino de introducir los conceptos que consideramos necesarios para establecer la relación entre los polinomios y los códigos. Las demostraciones de los resultados mostrados en este capítulo pueden ser consultadas en *Coding Theory: The Essentials* por Hoffman [8] y en *Álgebra Lineal. Álgebra Multilineal y k-teoría algebraica clásica.* por Lluis-Puebla [11].

Antes que nada, debemos definir lo que entendemos por polinomio.

Definición 4.1.1. Un *polinomio de grado n sobre K* es una expresión de la forma $a_0 + a_1x + \ldots + a_nx^n$ donde los coeficientes $a_0, a_1, \ldots, a_n$ son elementos de K y $a_n \neq 0$.

El conjunto de polinomios sobre K se denota como $K[x]$ y sus elementos son denotados por $f(x), g(x), p(x)$, etcétera.

4.1.1. Operaciones entre polinomios

Estamos acostumbrados a trabajar con polinomios sobre el campo de los reales. Sin embargo, nosotros vamos a trabajar sobre un campo distinto en donde se cumple que $1 + 1 = 0$ y en consecuencia $x^k + x^k = 0$. Esta propiedad ocasiona algunas variantes en la suma y la multiplicación. Por ejemplo, si $f(x) = 1 + x$ y $h(x) = 1 + x + x^2 + x^3 + x^4$ tenemos que:

a) $f(x) + g(x) = 1 + x + 1 + x + x^2 + x^3 + x^4 = x^2 + x^3 + x^4$.

b) $f(x)g(x) = 1(1 + x + x^2 + x^3 + x^4) + x(1 + x + x^2 + x^3 + x^4) = 1 + x + x^2 + x^3 + x^4 + x + x^2 + x^3 + x^4 + x^5 = 1 + x^5$.

Una vez que ya sabemos cómo sumar y multiplicar polinomios conviene que recordemos cómo dividir polinomios. El algoritmo es similar al que usamos cotidianamente, pero respetando las reglas del campo.

Algoritmo 4.1.2. Algoritmo de la División. Sean $f(x)$ y $h(x) \in K[x]$ con $h(x) \neq 0$. Entonces existen polinomios únicos $q(x)$ y $r(x) \in K[x]$ tal que

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

con $r(x) = 0$ o el $grado(r(x)) < grado(h(x))$.

Al polinomio $q(x)$ se le denomina *cociente* y al polinomio $r(x)$ se le denomina *residuo*.

Ejemplo 4.1.3. Sea $f(x) = x + x^2 + x^6 + x^7 + x^8$ y $h(x) = 1 + x + x^2 + x^4$.

$$\begin{array}{r|l} & x^4 + x^3 \\ \hline x^4 + x^2 + x + 1 & x^8 + x^7 + x^6 + x^2 + x \\ & \underline{x^8 + x^6 + x^5 + x^4} \\ & \quad x^7 + x^5 + x^4 + x^2 + x \\ & \quad \underline{x^7 + x^5 + x^4 + x^3} \\ & \quad\quad x^3 + x^2 + x \end{array}$$

El cociente es $q(x) = x^3 + x^4$ y el residuo es $r(x) = x + x^2 + x^3$.

4.1.2. Correspondencia entre palabras y polinomios

El polinomio $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$ de grado a lo más $n-1$ sobre K puede ser considerado como la palabra $v = a_0a_1a_2 \ldots a_{n-1}$ de longitud n en K^n. Por ejemplo, si $n = 7$,

polinomio	*palabra*
$1 + x + x^2 + x^4$	1110100
$1 + x^4 + x^5 + x^6$	1000111
$1 + x + x^3$	1101000

Definición 4.1.4. Decimos que $f(x)$ *módulo* $h(x)$ es $r(x)$ si $r(x)$ es el residuo cuando $f(x)$ es dividido entre $h(x)$, lo cual escribimos como $r(x) = f(x) \mod h(x)$.

Además, decimos que dos funciones $f(x)$ y $p(x)$ son *equivalentes módulo* $h(x)$ si y solamente si tienen el mismo residuo cuando son divididos por $h(x)$, es decir,

$$f(x) \mod h(x) = r(x) = p(x) \mod h(x).$$

Esta relación se denota como

$$f(x) \equiv p(x)(\mod h(x)).$$

Ejemplo 4.1.5. Sea $h(x) = 1 + x^5$ y $f(x) = 1 + x^4 + x^9 + x^{11}$. Entonces dividir $f(x)$ entre $h(x)$ da como residuo $r(x) = 1 + x$. Decimos que $r(x) = f(x) \mod h(x)$.

Por otro lado, si $p(x) = 1 + x^6$, entonces $1 + x \equiv 1 + x^6(\mod (1 + x^5))$ y por lo tanto, en este caso, $p(x) \equiv f(x)(\mod h(x))$.

Como podemos ver la equivalencia mantiene algunas propiedades que vamos a formalizar en el siguiente teorema.

Lema 4.1.6. Si $f(x) \equiv g(x)(\mod h(x))$ entonces,

$$\begin{aligned} f(x)+p(x) &\equiv g(x)+p(x)(\mod h(x)) \quad y \\ f(x)p(x) &\equiv g(x)p(x)(\mod h(x)). \end{aligned}$$

Para terminar esta sección sobre polinomios, deseamos hacer hincapié en que no hemos exigido que el campo al que pertenecen los coeficientes de un polinomio, sea binario. En este trabajo, sólo usaremos campos finitos.

4.2. Códigos cíclicos

Hay un tipo especial de códigos lineales llamados códigos cíclicos. Estos códigos mantienen propiedades que facilitan las tareas de codificación y decodificación.

Definición 4.2.1. El *desplazamiento cíclico* $\pi(v)$ de la palabra $v = a_0a_1 \ldots a_{n-1}$ es la palabra $\pi(v) = a_{n-1}a_0 \ldots a_{n-2}$.

Por ejemplo,

v	10110	111000	0000	1011
$\pi(v)$	01011	011100	0000	1101

Definición 4.2.2. Un código C es *cíclico* si el desplazamiento cíclico de cada palabra de código es también una palabra de código.

Cabe señalar que $\pi(v)$ es una función lineal y que por tal motivo para detectar si un código lineal C es cíclico, basta con observar si al aplicar la operación $\pi(v)$ sobre los elementos de la base, se cumple que las palabras resultantes son palabras del código C.

4.2.1. Construcción de un código cíclico lineal

El procedimiento que a continuación presentamos describe la construcción de un código cíclico.

1. Sea v una palabra de longitud n.
2. Formar un conjunto $S = \{v, \pi(v), \pi^2(v), \ldots, \pi^{n-1}(v)\}$.
3. Sea[1] $C = < S >$.

Se dice que tal palabra v es un *generador* del código cíclico C y se dice que el código C es el *código cíclico lineal más pequeño* que contiene a v. La idea principal del procedimiento anterior se basa en el hecho que describe el siguiente lema.

Lema 4.2.3. Sea C un código cíclico con palabras de longitud n y sea $v \in C$. Entonces para cualquier polinomio $a(x), c(x) = a(x)v(x) \mod (1 + x^n)$.

[1] Recuérdese que estamos considerando que el código C es un espacio lineal y que la notación $< S >$ se interpreta como el espacio lineal generado por el conjunto S.

4.2.2. Propiedades

Como hemos podido observar, aprovechando las características de los códigos lineales cíclicos podemos hacer más eficientes la descripción y el manejo de estos códigos. A continuación presentamos un teorema que describe al polinomio más importante en un código cíclico.

Teorema 4.2.4. Sea C un código cíclico. Entonces existe un polinomio único $g(x)$ en C tal que es mónico y tiene el grado más pequeño entre todos los polinomios distintos de cero. Además, $C =<< g(x) >>$.

A dicho polinomio $g(x)$ le llamamos el *polinomio generador* de C.

Enseguida presentamos un teorema muy útil que resume muchos datos importantes de estos códigos.

Teorema 4.2.5. Sea C un código cíclico de longitud n y sea $g(x)$ el polinomio generador. Si $n - k = grado(g(x))$ entonces:

1. C tiene dimensión k.
2. Las palabras de código correspondientes a $g(x), xg(x), \ldots, x^{k-1}g(x)$ forman una base para C.
3. $c(x) \in C$ si y solamente si $c(x) = a(x)g(x)$ para algún polinomio $a(x)$ con $grado(a(x)) < k$.

Podemos encontrar una propiedad muy importante del polinomio generador en el siguiente teorema. Sin embargo, el resultado que nos será útil lo tenemos en el corolario que se deriva de dicho teorema.

Teorema 4.2.6. $g(x)$ es el polinomio generador de un código cíclico lineal de longitud n si y solamente si $g(x)$ divide a $1 + x^n$, es decir, si $1 + x^n = h(x)g(x)$.

Corolario 4.2.7. El polinomio generador $g(x)$ para el código cíclico lineal más pequeño de longitud n que contiene la palabra v es el *máximo común divisor* entre $v(x)$ y $1 + x^n$.

4.3. Codificación y decodificación

En esta sección describiremos los procesos generales de codificación y decodificación que se pueden usar con los códigos lineales. Sin embargo, como veremos en los siguientes capítulos, regularmente cada código lineal tiene sus propios procesos que proporcionan ventajas propias del tipo de código que se está ocupando.

4.3.1. Codificación

Queremos empezar a hablar acerca de la codificación con un concepto muy importante, la matriz generadora. La forma más simple de encontrar una matriz generadora para un código cíclico lineal se remite al uso del Teorema 4.2.5 con el que construimos la siguiente matriz:

$$\begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$$

Ejemplo 4.3.1. Sea C el código cíclico lineal de longitud $n = 7$ con polinomio generador $g(x) = 1 + x + x^3$ de grado $n - k = 3$. Por lo tanto la dimensión $k = 4$, y la base es,

$$\begin{array}{rcl} g(x) & = & 1 + x + x^3 \\ xg(x) & = & x + x^2 + x^4 \\ x^2g(x) & = & x^2 + x^3 + x^5 \\ x^3g(x) & = & x^3 + x^4 + x^6. \end{array}$$

Y la matriz generadora para C es

$$G = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix}$$

Observando la matriz generadora podemos darnos cuenta que la longitud del mensaje que se codifica[2] debe ser k (el número de renglones) y la longitud de la palabra de código es n (el número de columnas). Los k dígitos de *información* que serán codificados, pueden ser pensados como un polinomio $a(x) = a_0 + a_1x + \ldots + a_{k-1}x^{k-1}$ al que llamamos el *polinomio de información* o el *polinomio del mensaje.*

Hasta este momento, sabemos que podemos codificar usando la matrix generadora. Sin embargo, hay una forma más sencilla y ésta consiste en multiplicar el polinomio $a(x)$ con el polinomio $g(x)$.

Por ejemplo, si $g(x) = 1 + x^2 + x^3$ es el polinomio generador de un código cíclico lineal de longitud 7, a(x) es la representación polinomial de la palabra a codificar y la dimensión es $k = 4$, tenemos:

a) $a(x) = 1 + x^3$.
$c(x) = (1 + x^3)(1 + x^2 + x^3) = 1 + x^2 + x^3 + x^3 + x^5 + x^6 = 1 + x^2 + x^5 + x^6 \longleftrightarrow 1010011$.
Por lo tanto el mensaje 1001 corresponde a la palabra de código 1010011.

b) $a(x) = x$.
$c(x) = (x)(1 + x^2 + x^3) = x + x^3 + x^4 \longleftrightarrow 0101100$ Por lo tanto el mensaje 0100 corresponde a la palabra de código 0101100.

[2]Por codificar entenderemos la multiplicación por la izquierda del mensaje con la matriz generadora.

4.3.2. Decodificación

En el proceso de decodificación tenemos que incluir el procedimiento que nos permita recuperar el mensaje (dentro de la capacidad de corrección del código) con los posibles errores ocurridos en la palabra recibida, es decir, obtener $c(x)$ (la palabra de código enviada) a partir de $w(x)$ (la palabra recibida), tal que $c(x) = w(x) + e(x)$.

Definición 4.3.2. El *polinomio del síndrome*, $s(x)$, se define como $s(x) = w(x) \mod g(x)$, en donde $s(x) = 0$ si y solamente si $w(x)$ es una palabra de código.

Algoritmo 4.3.3. Algoritmo de decodificación de códigos cíclicos lineales.

1. Calcular el polinomio del síndrome $s(x) = w(x) \mod g(x)$, donde w es la palabra recibida.
2. Por cada $i \geq 0$, calcula $s_i = s_i(x) \mod g(x)$ hasta encontrar un síndrome s_j tal que el peso de s_j sea menor o igual a t, donde t es el número de errores que puede corregir el canal.
3. El polinomio de error de máxima probabilidad es $e(x) = x^{n-j}s_j(x) \mod (1 + x^n)$.

Ejemplo 4.3.4. Sea $g(x) = 1+x^4+x^6+x^7+x^8$ el generador de un código cíclico lineal corrector de $t = 2$ errores y de longitud $n = 15$.

Supongamos que recibimos la palabra
$w = 001000001110110$ que es equivalente a $w(x) = x^2 + x^8 + x^9 + x^{10} + x^{12} + x^{13}$.

Buscamos el síndrome con peso $\leq t = 2$.

$$\begin{array}{rcrcl} s(x) & = & w(x) \mod g(x) & = & x + x^2 + x^3 \\ s_1(x) & = & xs(x) & = & x^2 + x^3 + x^4 \mod g(x) = x^2 + x^3 + x^4 \\ s_2(x) & = & x^2s(x) & = & x^3 + x^4 + x^5 \mod g(x) = x^3 + x^4 + x^5 \\ s_3(x) & = & x^3s(x) & = & x^4 + x^5 + x^6 \mod g(x) = x^4 + x^5 + x^6 \\ s_4(x) & = & x^4s(x) & = & x^5 + x^6 + x^7 \mod g(x) = x^5 + x^6 + x^7 \\ s_5(x) & = & x^5s(x) & = & x^6 + x^7 + x^8 \mod g(x) = 1 + x^4 \end{array}$$

Cuando $j = 5$ nos detenemos porque el peso de $s_5(x) \leq 2$.

Así que,

$e(x) = x^{n-5}s_5(x) \mod (1 + x^{15}) = x^{10}(1 + x^4) \mod (1 + x^{15}) = x^{10} + x^{14} \mod (1 + x^{15}) = x^{10} + x^{14}$.

Por lo tanto,

$$\begin{array}{rcl} c(x) & = & w(x) + e(x) \\ & = & x^2 + x^8 + x^9 + x^{10} + x^{12} + x^{13} + x^{10} + x^{14} \\ & = & x^2 + x^8 + x^9 + x^{12} + x^{13} + x^{14} \end{array}$$

Entonces $c(x) = 001000001100111$.

4.4. Códigos duales

Algo interesante en los códigos cíclicos lineales es que el dual de un código cíclico es también un código cíclico, como se establece en el siguiente teorema.

Teorema 4.4.1. Si C es un código cíclico lineal de longitud n y dimensión k con generador $g(x)$ y si $1 + x^n = g(x)h(x)$ entonces $C^{\perp}$ es un código cíclico de dimensión $n - k$ con generador $x^k h(x^{-1})$.

4.5. Comentarios

En este capítulo hemos revisado los conceptos y teoremas relacionados con los códigos lineales, el uso de polinomios y los códigos cíclicos, veremos más adelante que estos conceptos nos ayudarán a entender algunos códigos que veremos en los siguientes capítulos.

Capítulo 5

Códigos de Hamming y Códigos de Golay

5.1. Códigos de Hamming

Los códigos de Hamming han sido ampliamente estudiados y es una referencia obligada al estudiar códigos lineales. Nuestra intención con este capítulo es plantear cuáles son las características de estos códigos que le han ganado tanta popularidad y explicar la implementación que seguiremos en este trabajo.

El material que aquí presentamos, incluyendo las demostraciones de los resultados que están en este capítulo, se basa en *Coding and Information Theory* por Hamming [7] y en *Introduction to Coding Theory* por Lint [10].

5.1.1. Introducción

Al trabajar con códigos detectores y correctores de errores, empezamos a manejar el concepto de *bit de paridad*. Los bits de paridad son bits que se le agregan al *mensaje*[1] para formar una palabra de código y su función radica en mantener un número par (o impar) de unos en subconjuntos de bits de la palabra de código. Por ejemplo, supongamos que solamente vamos a agregar un bit de paridad al siguiente mensaje:

$$c_1 = 110010.$$

Tomaremos como subconjunto de bits a toda la palabra, y como deseamos mantener una paridad par, contamos el número de unos que constituye el mensaje, es decir, calculamos su peso. Si el peso es par entonces agregamos un 0, si no agregamos un 1. En este caso,

[1] El mensaje es la cadena de bits que se desea transmitir.

$$w(c_1) = 3;$$

como el 3 es impar entonces agregamos un 1 al final de la palabra,

$$\textit{palabra de código} = 1100101.$$

Así, cuando el receptor reciba las palabras de código debe contar un número par de unos. De lo contrario, se sabe que ocurrió un error.

No obstante, es posible que se cometan errores en la palabra y que, por mera casualidad, el resultado de estos errores constituya una palabra con una cantidad par de unos. Si esto ocurre, el receptor no tiene forma de detectar el error y simplemente acepta la palabra. Con esto deseamos enfatizar en que siempre trabajamos con este riesgo y nos limitamos a disminuirlo.

Retomando los códigos de Hamming, los bits de paridad juegan un papel similar que en el ejemplo anterior. Sólo que serán varios bits los que trabajen sobre el mismo mensaje, cada uno estableciendo la paridad de un subconjunto de bits.

En este sentido, una de las ideas centrales es que cada uno de los subconjuntos mantienen intersecciones no vacías, pero ninguno de los subconjuntos puede ser obtenido sumando algunos de ellos.

Hamming menciona que una de las ideas principales de estos códigos consiste en que el síndrome es **0** cuando no ocurrió ningún error. De lo contrario, el síndrome se interpreta como un número, y este número señala la posición[2] del bit erróneo.

5.1.2. Propiedades

Los códigos de Hamming mantienen una distancia mínima de 3. Con la información que establecimos en el capítulo 2, podemos inferir que se trata de un código que detecta hasta 2 errores y corrige 1 error.

Por otro lado, recordemos que cuando construimos la matriz estándar se presentaba cierta ambigüedad, la cual no se presenta si se trata de un código perfecto. Afortunadamente, los códigos de Hamming cumplen con esta característica como se menciona en el siguiente teorema.

Teorema 5.1.1. Los códigos de Hamming son códigos perfectos.

5.1.3. Matrices de Hamming

Mencionamos en el capítulo anterior a la matriz de verificación de paridad. A partir de esta matriz vamos a empezar a construir estos códigos.

Las *matrices de Hamming* son matrices verificadores de paridad[3] que pueden ser construidas fácilmente y se denotan con $H_p(h)$ donde p es la cantidad de dígitos (en nuestro caso $p = 2$, pues trabajamos con alfabetos binarios) y h es el número de renglones de la matriz.

[2] Las posiciones de una palabra inician de derecha a izquierda, empezando por la posición 1.

[3] Recordemos que la matriz verificadora de paridad nos ayuda en el proceso de decodificación para obtener el síndrome.

Construcción

Nuestro objetivo es construir la matriz de verificación de paridad de Hamming p-aria de h renglones. Primero mencionaremos los pasos para construirla y después mostraremos un ejemplo.

1. Si se desea construir la matriz $H_p(h)$ se estará trabajando sobre $\mathbb{Z}_p^h$. El primer paso es listar todas las cadenas que pertenezcan a $\mathbb{Z}_p^h$.

2. Cada cadena se interpretará como un número, donde la posición más significativa es la que se encuentra más a la izquierda y cuyo dígito sea distinto de cero.

3. Viéndolos como números, sabemos que el dígito de mayor valor es $p-1$.

4. Se excluyen todos aquellos números que tengan en la posición más significativa el dígito $p-1$, se excluye al **0** también.

5. Con los números restantes se forma la matriz, donde se coloca a cada número de manera decreciente, en forma de columna, con la posición menos significativa al inicio de la columna y la más significativa al final de la misma.

Ejemplo 5.1.2. Deseamos construir la matriz $H_3(3)$.

1. Se lista $\mathbb{Z}_3^3$. Sabemos que $h = 3$ es el número de renglones de la matriz de Hamming y $p = 3$ indica el alfabeto que se está manejando, por lo que, en este caso estamos hablando de $\mathbb{Z}_3 = \{0, 1, 2\}$.

000	100	200
001	101	201
002	102	201
010	110	210
011	111	211
012	112	212
020	120	220
021	121	221
022	122	222

2. Si los observamos como números, el menor es **0** y el mayor 222.

3. El dígito con mayor valor es el 2.

4. Excluimos a aquellos números que tienen al 2 en la posición más significativa y también excluimos a la palabra **0**.

~~000~~	100	~~200~~
001	101	~~201~~
~~002~~	102	~~201~~
010	110	~~210~~
011	111	~~211~~
012	112	~~212~~
~~020~~	120	~~220~~
~~021~~	121	~~221~~
~~022~~	122	~~222~~

5. Con los números restantes formamos la matriz, donde cada uno se escribe como columna.

$$H_3(3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{bmatrix}$$

Características

Steven Roman [15] menciona algunas características de las matrices de Hamming, que consideramos conveniente señalar aquí.

1. La matriz de Hamming $H_p(h)$ tiene tamaño $h \times n$, donde $n = \frac{p^h-1}{p-1}$.
2. Los renglones de $H_p(h)$ son linealmente independientes sobre $\mathbb{Z}_p$.
3. Las primeras tres columnas de $H_p(h)$ son linealmente dependientes.

Estas características se sintetizan en el siguiente teorema.

Teorema 5.1.3. La matriz de Hamming $H_p(h)$ es una matriz verificadora de paridad para un código lineal $[n, k, d]$-código $H_p(h)$ sobre $\mathbb{Z}_p$ con los parámetros

$$n = \frac{p^h - 1}{p - 1},\ k = n - h,\ d = 3.$$

A este código se le denomina el *código de Hamming p-ario con parámetro h.* De tal forma que $H_p(h)$ es corrector de un error.

5.1.4. Descripción del algoritmo

Con toda la información que contamos, es el momento adecuado de mencionar el algoritmo para codificar y decodificar usando códigos de Hamming binarios. Primero describiremos cuáles son los pasos en la codificación y después en la decodificación.

Esencialmente, esta descripción del algoritmo se puede revisar en [5, 7].

Para codificar:

1. Recibimos el mensaje que se desea codificar. Llamaremos n_1 a la longitud del mensaje.

2. Calculamos cuántos bits de paridad se deben agregar en el mensaje. Esto es equivalente a resolver la siguiente desigualdad, donde m es el entero más pequeño que la cumple y m es la cantidad de bits de paridad.
$$2^m \geq m + n_1 + 1,$$

3. Formamos la palabra de código de longitud $n = n_1 + m$.
Sabemos que tenemos que agregar m bits de paridad. Para saber qué posiciones deben ser verificadas por cada bit de paridad vamos a observar la siguiente tabla. La primera columna señala la posición del bit erróneo.

Posición	Representación binaria
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
⋮	⋮

Si nos fijamos en el bit menos significativo en la columna de la representación binaria, vemos que está encendido en las posiciones 1,3,5,7,9,11,13, ... Por lo tanto, el primer bit de paridad va a verificar las posiciones 1,3,5,7,9,11,13,....

Si nos fijamos en el segundo bit menos significativo, vemos que está encendido en las posiciones 2,3,6,7,10,11,14,15 .... Por lo tanto, el segundo bit de paridad cubre dichas posiciones.

Al fijarnos en el tercer bit menos significativo, vemos que está encendido en las posiciones 4,5,6,7,12,13,14,15, ..., es decir, el tercer bit cubre dichas posiciones. El siguiente bit de paridad cubre las posiciones 8,9,10,11,12,13,14,15,24,25, ... Y así seguimos el mismo procedimiento hasta que sepamos qué posiciones cubre cada uno de los m bits de paridad.

4. Una vez que sabemos cuáles posiciones cubre cada bit de paridad, vamos a insertarlos en el mensaje para formar la palabra de código.

   Cada uno de los bits de paridad se coloca en la posición 2^h donde h = 0,1,2, ..., es decir, el primer bit de paridad se coloca en la posición $2^0 = 1$ de la palabra de código, el segundo en la posición $2^1 = 2$, el tercero en la posición $2^2 = 4$, y así sucesivamente.

5. La cadena con el mensaje y los bits de paridad que se insertaron en el paso anterior constituyen la palabra de código que vamos a enviar.

Para decodificar:

1. Recibimos la palabra de longitud n.
2. Calculamos el síndrome.

   Para llevar a cabo este paso, se debe multiplicar la matriz de Hamming por la cadena recibida, que se coloca como matriz columna. Sin embargo, existen atajos que nos permiten calcular el síndrome más rápidamente; en esta tesis tomaremos el siguiente.

   a) Conocemos cuáles son las posiciones que verifica cada bit de paridad.

Posición del bit de paridad	Subconjunto de bits que verifica
$2^0 = 1$	1,3,5,7,9,11,13, ...
$2^1 = 2$	2,3,6,7,10,11,14,15, ...
$2^2 = 4$	4,5,6,7,12,13,14,15, ...
$\vdots$	$\vdots$

   b) El bit menos significativo que forma parte del síndrome, se calcula aplicando la operación $\oplus$[4] entre los bits que pertenecen al subconjunto correspondiente al primer bit de paridad, es decir, el que está en la posición $2^0 = 1$.

   c) El segundo bit menos significativo del síndrome se calcula de la misma forma, aplicando la operación $\oplus$ sobre el subconjunto de bits correspondiente al bit de paridad en la posición $2^1 = 2$.

   d) Se calculan los bits restantes del síndrome siguiendo este procedimiento hasta terminar los bits de paridad de la palabra recibida.

3. Si el síndrome es igual a **0** entonces no hubo errores y nos dirigimos al paso 6, si no nos vamos al siguiente paso.
4. Considerar al síndrome como un número. Este número señalará la posición del bit que se debe corregir.
5. Se niega el bit indicado por la posición que acabamos de calcular en el paso anterior, es decir, estamos corrigiendo la palabra.
6. Recuperamos el mensaje eliminando los bits de paridad.

5.1.5. Ejemplos

Ejemplo 5.1.4. Supongamos que queremos enviar el mensaje 101001 de longitud $n_1 = 6$.

[4] $\oplus$ indica la operación XOR.

Proceso de codificación

- Calculamos cuántos bits de paridad se deben agregar al mensaje.

$$2^m \geq m + 6 + 1.$$

$m = 4$, ya que 4 es el entero más pequeño que cumple la desigualdad anterior.

$$2^4 \geq 4 + 6 + 1,$$

$$m = 4, n_1 = 6 \text{ y } n = 10.$$

- Copiamos los bits del mensaje en sus nuevas posiciones:

		1		0	1	0		0	1
$2^0 = 1$	$2^1 = 2$	3	$2^2 = 4$	5	6	7	$2^3 = 8$	9	10

- Calculamos los bits de paridad:

1. Primer bit. Posiciones: 3,5,7 y 9.
   $1 \oplus 0 \oplus 0 \oplus 0 = 1$

1		1		0	1	0		0	1
$2^0 = 1$	$2^1 = 2$	3	$2^2 = 4$	5	6	7	$2^3 = 8$	9	10

2. Segundo bit. Posiciones: 3,6,7 y 10.
   $1 \oplus 1 \oplus 0 \oplus 1 = 1$

1	1	1		0	1	0		0	1
$2^0 = 1$	$2^1 = 2$	3	$2^2 = 4$	5	6	7	$2^3 = 8$	9	10

3. Tercer bit. Posiciones: 5,6 y 7.
   $0 \oplus 1 \oplus 0 = 1$

1	1	1	1	0	1	0		0	1
$2^0 = 1$	$2^1 = 2$	3	$2^2 = 4$	5	6	7	$2^3 = 8$	9	10

4. Cuarto bit. Posiciones: 9 y 10.
   $0 \oplus 1 = 1$

1	1	1	1	0	1	0	1	0	1
$2^0 = 1$	$2^1 = 2$	3	$2^2 = 4$	5	6	7	$2^3 = 8$	9	10

- El resultado de la codificación es la *palabra de código* 1111010101.

Supongamos que ocurre un error durante la transmisión cuando enviamos la palabra de código 1111010101 en la posición 8, es decir, que se recibe la palabra 1111010001.

Proceso de decodificación

- Calcular el síndrome:

  1. Bit menos significativo del síndrome. Posiciones: 1,3,5,7 y 9,
     $1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$.
  2. Segundo bit menos significativo del síndrome. Posiciones: 2,3,6,7 y 10,
     $1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$.
  3. Tercer bit menos significativo del síndrome. Posiciones: 4,5,6 y 7,
     $1 \oplus 0 \oplus 1 \oplus 0 = 0$.
  4. Cuarto bit menos significativo del síndrome. Posiciones: 8,9 y 10.
     $0 \oplus 0 \oplus 1 = 1$

     El síndrome es 1000.

- $1000_2 = 8_{10}$.
- Se concluye que hubo un error en la posición 8 y corregimos dicha posición.
- El receptor decide que la palabra de código enviada es 1111010101.
- Recuperamos el mensaje de la palabra de código.
- El mensaje supuesto es 101001.

Como vimos en el ejemplo anterior el algoritmo de codificación y decodificación proporcionó resultados correctos aun cuando se presentó un error durante la transmisión. Sin embargo, ahora vamos a suponer que ocurren dos errores en la misma palabra de código.

Ejemplo 5.1.5. Supongamos que los errores ocurrieron en las posiciones 3 y 7, es decir, el receptor toma la palabra 1101011101.

- Calcular el síndrome.

  1. Bit menos significativo del síndrome. Posiciones: 1,3,5,7 y 9,
     $1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$.
  2. Segundo bit menos significativo del síndrome. Posiciones: 2,3,6,7 y 10,
     $1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0$.
  3. Tercer bit menos significativo del síndrome. Posiciones: 4,5,6 y 7,
     $1 \oplus 0 \oplus 1 \oplus 1 = 1$.

4. Cuarto bit menos significativo del síndrome. Posiciones: 8,9 y 10,
   $1 \oplus 0 \oplus 1 = 0$.

   El síndrome es 0100.

- $0100_2 = 4_{10}$.
- Se concluye que hubo un error en la posición 4 y corregimos dicha posición.
- El receptor decide que la palabra de código enviada es 1100011101.
- Recuperamos el mensaje de la palabra de código.
- El mensaje es 001101.

Pero en realidad, enviaron el mensaje 101001.

Es éste el tipo de errores que pueden ocurrir con los códigos de Hamming.

5.1.6. Observaciones

Hemos visto cómo se pueden cometer errores en los códigos de Hamming. Y reiteramos, no existe un código que siempre proporcione la respuesta correcta. Sin embargo, los códigos de Hamming ofrecen las facilidades de codificación y decodificación con las que cumplen pocos códigos.

Por otro lado, a medida que la longitud del mensaje crece, es menor la proporción entre la longitud del mensaje y la cantidad de bits de paridad que se agregan, es decir, aumenta la tasa de transmisión.

5.2. Códigos de Golay

Golay publicó un artículo de una página en 1949 donde presentó estos códigos que llevan su nombre. Aun cuando son cuatro los códigos de Golay, a nosotros sólo nos interesa $\mathcal{G}_{24}$, el cual, junto con $\mathcal{G}_{23}$, son los códigos binarios más famosos desde el punto de vista de J. H. van Lint, tanto por razones teóricas como prácticas.

El código de Golay $\mathcal{G}_{24}$, se describe a partir de la siguiente matriz[5] a la que llamaremos **A**.

[5]Muchos autores prefieren sustituir el 0 por un · para que sea más clara la visualización de la matriz; nosotros seguiremos esta notación.

$$\mathbf{A} = \begin{bmatrix} \cdot & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot \\ 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 \\ 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 \\ 1 & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 \\ 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 \\ 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 \\ 1 & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot \\ 1 & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot \\ 1 & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

La matriz generadora del código está determinada de la siguiente manera: $G_{24} = (\mathbf{I}_{12}|\mathbf{A})$, donde $\mathbf{I}_{12}$ es la matriz identidad de 12 renglones y 12 columnas.

Como tanto la matriz $\mathbf{I}_{12}$ y la matriz $\mathbf{A}$ son simétricas entonces la matriz transpuesta de G_{24} queda constituida por $G_{24}^t = \frac{\mathbf{I}_{12}}{\mathbf{A}}$.

Este código de Golay es un (24,4096,8)-código. Por lo tanto puede corregir hasta 3 errores y detectar hasta 7 errores.

5.2.1. Descripción del algoritmo

La descripción del algoritmo que aquí presentamos proviene de [5], pues consideramos que es la explicación más apropiada para nuestro trabajo.

Para codificar el mensaje sólo debemos multiplicar la transpuesta de la matriz G_{24} por el mensaje. En este algoritmo el mensaje mantiene una longitud fija de 12 bits. Por lo tanto, se lleva a cabo la multiplicación

$$G_{24}^t \times \textit{mensaje} = \textit{palabra de código},$$

en donde la palabra de código es de 24 bits.

Enseguida describiremos el proceso de decodificación en donde denotaremos las operaciones sobre bits XOR y OR como $\oplus$ y $+$, respectivamente.

1. Buscar el patrón de error $\mathbf{e}$.
   - *a)* Se calcula el síndrome $\mathbf{s} = G_{24}\mathbf{d}$.
   - *b)* Si $w(\mathbf{s}) \leq 3$ entonces el patrón de error es $\mathbf{s} << 12$.
   - *c)* Si $w(\mathbf{s}) > 3$ entonces buscamos un índice $i \in \{0, \ldots, 11\}$ tal que la columna i–ésima de la matriz $\mathbf{A}$ difiera de $\mathbf{s}$ a lo más en 2 bits. Si existe dicha columna, el patrón de error es $\mathbf{e} = ((\mathbf{s} \oplus \mathbf{A}[i]) << 12) + \mathbf{I}_{12}[i]$.
   - *d)* Si no hubo tal columna entonces obtenemos $\mathbf{s}_1 = \mathbf{A}\mathbf{s}$.

e) Si $w(\mathbf{s}_1) \leq 3$ entonces $\mathbf{e} = \mathbf{s}_1$.

f) Si $w(\mathbf{s}_1) > 3$ entonces buscamos un índice $i \in \{0, \ldots, 11\}$ tal que la columna $i-$ésima de la matriz $\mathbf{A}$ difiera de $\mathbf{s}_1$ a lo más en 2. Si dicha columna existe, el patrón de error es $\mathbf{e} = (\mathbf{s}_1 \oplus \mathbf{A}[i]) + (\mathbf{I}_{12}[i] << 12)$.

g) Si no existe dicha columna entonces ocurrieron muchos errores y no podemos obtener la palabra de código.

h) Eliminamos los últimos 12 bits de la palabra de código $\mathbf{c}$ para obtener el mensaje m.

2. Aplicar la operación $\oplus$ entre la palabra $\mathbf{e}$ y la palabra recibida $\mathbf{d}$ para obtener la palabra de código con mayor probabilidad en haber sido enviada, es decir, $\mathbf{e} \oplus \mathbf{d} = \mathbf{c}$.

5.2.2. Ejemplo

Proceso de codificación

1. El mensaje que se va a codificar es $m = 110110001101$.

2. Multiplicar $G_{24}^t m$.

$$\begin{pmatrix}
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot \\
1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 \\
1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 \\
1 & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot \\
1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 \\
1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 \\
1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 \\
1 & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot \\
1 & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot \\
1 & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\
1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1
\end{pmatrix}^t
\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}
=
\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

3. Por lo tanto, la palabra codificada es 110110001101010101000011.

Proceso de decodificación

Supongamos que durante la transmisión ocurrieron 3 errores, en las posiciones[6] 8, 12 y 16, de tal manera que la palabra con errores es

$$\mathbf{d} = 110110011100010001000011.$$

1. Vamos a calcular el síndrome $\mathbf{s} = G_{24}\mathbf{d}$.

   Donde G_{24} es la siguiente matriz.

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

2. El resultado de la multiplicación es el síndrome $\mathbf{s} = 101110011010$.

3. El peso del síndrome es $w(\mathbf{s}) = 7$.

4. Como $w(\mathbf{s}) > 3$ entonces buscamos la columna en la matriz $\mathbf{A}$ que difiera a lo más en 2 bits del síndrome. Tal columna es la columna $i = 3$, recuérdese que el índice i empieza en 0.

5.

$$\begin{array}{rcccccccccccccc}
 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & = & \mathbf{s} \\
\oplus & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & = & \mathbf{A}[3] \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & &
\end{array}$$

6. $000000010001 << 12 = 000000010001000000000000$

7.

$$\begin{array}{rcccccccccccccccccccccccc}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = \mathbf{I}[3] \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
\end{array}$$

8. Para terminar con el procedimiento, aplicamos la operación $\oplus$ entre la palabra recibida y el patrón de error

[6] Recordemos que las posiciones en una palabra inician en 1.

$$
\begin{array}{cccccccccccccccccccccccc}
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & = & \mathbf{d} \\
\oplus & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & \mathbf{e} \\
\hline
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & = & \mathbf{c}
\end{array}
$$

9. Por lo tanto, el mensaje se constituye por los primeros 12 bits de la palabra de código $\mathrm{c} = 110110001101010101000011$, es decir, $m = 110110001101$.

5.3. Comentarios

En este capítulo hemos revisado los códigos de Golay y Hamming, los cuales se caracterizan por presentar un proceso más sencillo de decodificación. Cabe señalar que en estos códigos sabemos en cuáles posiciones de la palabra de código se encuentran los bits que conforman el mensaje original. Dicho en otras palabras, si la palabra recibida es una palabra de código entonces sabemos perfectamente cuáles son las posiciones que ocupan los bits de redundancia y simplemente los eliminamos para obtener el supuesto mensaje enviado. No obstante, en los siguientes códigos que revisaremos en los capítulos subsecuentes, no se da este tipo de comportamiento.

Capítulo 6

Códigos de Reed-Muller

Los códigos de Reed-Muller fueron desarrollados por I. S. Reed y D. H. Muller en 1954. Una de las aplicaciones más conocidas de estos códigos fue durante la transmisión de fotografías en blanco y negro de Marte por el Mariner 9 en 1972. La descripción y el enfoque de los códigos de Reed-Muller que aquí presentamos está basada en [3, 8].

6.1. Preliminares

6.1.1. Vectores

Las palabras de un código de Reed-Muller forman un espacio vectorial, es decir, se trata de un código lineal. Podemos considerar cada palabra como un vector de longitud 2^m, donde m es un entero positivo y cada entrada del vector es un número de $\mathbb{Z}_2 = \{0, 1\}$. Para manipular los vectores usaremos tres operaciones: la suma, la multiplicación y el producto punto.

En general, usaremos $\mathbb{Z}_m$ para denotar el conjunto de enteros $\{0, 1, 2, \ldots, m-1\}$.

En $\mathbb{Z}_2$ la suma y la multiplicación se definen como,

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

Para dos vectores $\mathbf{x} = (x_1, x_2, ..., x_n)$ y $\mathbf{y} = (y_1, y_2, ..., y_n)$, la suma se define por

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, x_2 + y_2, ..., x_n + y_n),$$

donde cada x_i o y_i es 1 ó 0.

La *suma de un escalar* $a \in \mathbb{Z}_2$ con un vector $\mathbf{x}$ se define por

$$a + x = (a + x_1, a + x_2, \ldots, a + x_n).$$

El *complemento* $\bar{\mathbf{x}}$ de $\mathbf{x}$ es el vector igual a $1 + \mathbf{x}$.

La *multiplicación* se define por la fórmula,

$$\mathbf{x} * \mathbf{y} = (x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n).$$

La *multiplicación de una constante* $\mathbf{a} \in \mathbb{Z}_2$ con un vector $\mathbf{x}$ se define por

$$a * \mathbf{x} = (a * x_1, a * x_2, \ldots, a * x_n).$$

El *producto punto* entre $\mathbf{x}$ y $\mathbf{y}$ se define por

$$\mathbf{x} \cdot \mathbf{y} = x_1 * y_1 + x_2 * y_2 + \ldots + x_n * y_n.$$

6.1.2. Funciones

Estamos acostumbrados que al usar una notación posicional, el dígito menos significativo se encuentra más a la izquierda. Sin embargo, en este caso nos conviene tener un *orden inverso*, esto es que el dígito menos significativo se encuentra más a la derecha. Por ejemplo, pensemos que disponemos de 4 dígitos binarios. Bajo esta representación, usualmente el uno tiene la forma 0001, pero en el orden inverso es 1000.

Si ordenamos el conjunto de elementos de $\mathbb{Z}_2^m$ siguiendo el criterio del orden inverso entonces decimos que tenemos el *orden estándar de* $\mathbb{Z}_2^m$.

Ejemplo 6.1.1. El orden estándar de $\mathbb{Z}_2^2$ es (00, 10, 01, 11) y el orden estándar de $\mathbb{Z}_2^3$ es (000,100,010,110,001,101,011,111).

Ahora, construimos una función, f_I, que va de $\mathbb{Z}_2^m$ en {0,1}, de la siguiente forma:

Definición 6.1.2. Sea $I \subseteq \{0, 1, \ldots, m-1\}$. La función f_I se define como:

$$f_I(x_0, x_1, \ldots, x_{m-1}) = \begin{cases} \prod_{i \in I}(x_i + 1) & si\ I \neq \emptyset \\ 1 & si\ I = \emptyset \end{cases}$$

Esta función genera un vector $v_I = (f_I(u_0), f_I(u_1), \ldots, f_I(u_{2^m-1})) \in K^n$ donde $u_i \in K^m$, $n = 2^m$ y $u_0, u_1, \ldots, u_{2^m-1}$ es el orden estándar de los vectores en K^m.

Ejemplo 6.1.3. Sea m=4, así que $n = 2^4 = 16$. Sea $I = \{0, 3\}$ y deseamos encontrar v_I.

Entonces, de acuerdo con la definición anterior, $f_{\{0,3\}}(x_0, x_1, x_2, x_3) = (x_0 + 1)(x_3 + 1)$.

Para obtener v_I debemos de calcular $f_I(u_0), \ldots, f_I(u_{2^m-1})$.

$f_I(u_0) = f_{\{0,3\}}(0,0,0,0) = (0+1)(0+1) = 1$
$f_I(u_1) = f_{\{0,3\}}(1,0,0,0) = (1+1)(0+1) = 0$
$f_I(u_2) = f_{\{0,3\}}(0,1,0,0) = (0+1)(0+1) = 1$
$f_I(u_3) = f_{\{0,3\}}(1,1,0,0) = (1+1)(0+1) = 0$
$f_I(u_4) = f_{\{0,3\}}(0,0,1,0) = (0+1)(0+1) = 0$
$f_I(u_5) = f_{\{0,3\}}(1,0,1,0) = (1+1)(0+1) = 0$
$f_I(u_6) = f_{\{0,3\}}(0,1,1,0) = (0+1)(0+1) = 1$
$f_I(u_7) = f_{\{0,3\}}(1,1,1,0) = (1+1)(0+1) = 0$

$f_I(u_8) = f_{\{0,3\}}(0,0,0,1) = (0+1)(1+1) = 0$
$f_I(u_9) = f_{\{0,3\}}(1,0,0,1) = (1+1)(1+1) = 0$
$f_I(u_{10}) = f_{\{0,3\}}(0,1,0,1) = (0+1)(1+1) = 0$
$f_I(u_{11}) = f_{\{0,3\}}(1,1,0,1) = (1+1)(1+1) = 0$
$f_I(u_{12}) = f_{\{0,3\}}(0,0,1,1) = (0+1)(1+1) = 0$
$f_I(u_{13}) = f_{\{0,3\}}(1,0,1,1) = (1+1)(1+1) = 0$
$f_I(u_{14}) = f_{\{0,3\}}(0,1,1,1) = (0+1)(1+1) = 0$
$f_I(u_{15}) = f_{\{0,3\}}(1,1,1,1) = (1+1)(1+1) = 0$

Por lo tanto $v_I = v_{0,3} = (f_I(u_0), f_I(u_1), \ldots, f_I(u_{15})) = 1010001000000000$.

6.2. Códigos de Reed-Muller

Definición 6.2.1. Un *código de Reed-Muller de orden r*, $\mathcal{RM}(r,m)$, es el código lineal $\langle\{v_I | I \subseteq \mathbb{Z}_m, |I| \leq r\}\rangle$, donde r y m son enteros positivos tales que $r \leq m$.

La longitud del código es $n = 2^m$. La dimensión de un código de Reed-Muller es

$$k = 1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}.$$

Una base de este código lineal es $S = \{v_I | I \subseteq \mathbb{Z}_m, |I| \leq r\}$.

Teorema 6.2.2. La distancia mínima de $\mathcal{RM}(r,m)$ es 2^{m-r}.

6.2.1. Codificación

Sabemos que al trabajar sobre un código lineal, para codificar un mensaje basta con multiplicar dicho mensaje por la matriz generadora. Por lo tanto, es conveniente que encontremos esta matriz.

Ya conocemos una base del código, el conjunto S. Ahora, cada elemento que pertenece a la base formará un renglón de la matriz. Sin embargo, nos falta conocer el *orden* que deben seguir los elementos de la base para que la matriz esté en su forma canónica. Ordenaremos los vectores de la base considerando el siguiente criterio.

v_I está antes que v_J si se cumple cualquiera de las siguientes dos condiciones:

- $|I| < |J|$
- Si $|I| = |J|$ entonces $f_I(u_j) < f_J(u_j)$ y $f_I(u_i) < f_J(u_i)$ para $i > j$.

Ejemplo 6.2.3. Cuando $m = 3$ y $r = 1$, tenemos la siguiente matriz de codificación.

$$\mathcal{G}_{1,3} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} v_\emptyset \\ v_2 \\ v_1 \\ v_0 \end{matrix}$$

Si r = 2,

$$\mathcal{G}_{2,3} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} v_\emptyset \\ v_2 \\ v_1 \\ v_0 \\ v_{1,2} \\ v_{0,2} \\ v_{0,1} \end{matrix}$$

Si r = 3,

$$\mathcal{G}_{3,3} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} v_\emptyset \\ v_2 \\ v_1 \\ v_0 \\ v_{1,2} \\ v_{0,2} \\ v_{0,1} \\ v_{0,1,2} \end{matrix}$$

Más adelante mostraremos un ejemplo completo sobre cómo codificar y decodificar, el cual sin duda, aclarará más este procedimiento.

6.2.2. Decodificación

La decodificación se basa en el mecanismo que denominamos *por mayoría de votos*. Esto significa que el valor que más se repite es el que suponemos correcto. Sin embargo, si en determinado momento durante el proceso de decodificación existen exactamente la mitad de ceros y la mitad de unos, en otras palabras, que no hay una mayoría, significa que la cantidad de errores que ocurrieron rebasa la capacidad de corrección de ese código Reed-Muller en particular. Este comentario tiene el propósito de recordarnos[1] que no existe un código que siempre proporcione la respuesta correcta y nuestra tarea se limita a minimizar la probabilidad del error.

Para entender este procedimiento necesitamos algunos conceptos preliminares.

Definiciones

- Para cualquier $I \subseteq \mathbb{Z}_m$, se define $I^c = \mathbb{Z}_m \setminus I$ como el *complemento* de I en $\mathbb{Z}_m$.
- Sea $H_I = \{u \in \mathbb{Z}_2^m | f_I(u) = 1\}$.
- $f_I(x_0, \ldots, x_{m-1}) = \prod_{i \in I}(x_i + 1) = 1$ si y solamente si $x_i = 0$ para toda $i \in I$.
- Para cualquier $u = (u_0, u_1, \ldots, u_{m-1}) \in \mathbb{Z}_2^m$ y para cualquier $t = (t_0, t_1, \ldots, t_{m-1}) \in \mathbb{Z}_2^m$ definimos la función $f_{I,t}(u_0, u_1, \ldots, u_{m-1}) = f_I(u_0+t_0, u_1+t_1, \ldots, u_{m-1}+t_{m+1}) = f_I(u+t)$ y definimos $v_{I,t}$ como el vector que se forma usando la función $f_{I,t}$.

Algoritmo 6.2.4. Decodificación por mayoría de votos de $\mathcal{RM}(r, m)$.

1. Sea w la palabra recibida.
2. Sean $i = r$ y $w(r) = w$.
3. Obtener todos los posibles subconjuntos $J \subseteq Z_m$ tal que $|J| = i$, y para cada J se llevan a cabo los siguientes pasos:
   a) Obtener H_J.

[1] Siendo rigurosos, el único código en el que el receptor siempre sabe cuál fue la palabra enviada, es aquel que consiste de una sola palabra, el cual, como puede suponer el lector, no sirve para nuestros propósitos.

b) Para cada $t \in H_J$, calcular el producto punto entre $w(i)$ y $v_{J^c,t}$.

c) Tenemos $|H_J|$ productos del paso anterior. Si en estos productos se presenta una mayor cantidad de ceros que de unos entonces $m_J = 0$. Análogamente, si existen más unos que ceros entonces $m_J = 1$. Si existe la misma cantidad de ambos dígitos, se pide una retransmisión.

Cada m_J es un bit del supuesto mensaje, el orden en que obtengamos estos bits es el orden inverso que seguirán en la construcción del mensaje, es decir, el primer m_J que obtenemos es el último bit del supuesto mensaje y así sucesivamente.

4. Si $i > 0$ entonces calcular $w(i-1) = w(i) + \sum_{J \subseteq \mathbb{Z}_m} m_J v_J$ donde $|J| = i$.

5. Si el peso de $w(i-1)$ es menor o igual a $e = 2^{m-r-1} - 1$ entonces para toda $J \subseteq \mathbb{Z}_m$ con $|J| \leq r$, $m_J = 0$, y terminamos. De lo contrario, $i = i - 1$ y regresamos al paso 3.

6.3. Ejemplo

Codificación

Supongamos que deseamos codificar el mensaje $m = 01011000100$ de longitud 11 con el código $\mathcal{RM}(2,4)$. Este código tiene la capacidad de corrección de un error.

La matriz generadora del código de Reed-Muller $\mathcal{RM}(2,4)$ es la siguiente.

$$\mathcal{G}_{2,4} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Al multiplicar el mensaje m con la matrix $\mathcal{G}_{2,4}$ obtenemos la palabra de código $c = 0101100110100110$.

Decodificación

Supongamos que ocurrió un error en el quinto bit y la palabra recibida es $w = 0101000110100110$.

- Sea $w = 0101000110100110$.

- Sean $i = 2$ y $w(2) = w$.

J	t	$v_{J^c,t}$	$w \cdot v_{J^c,t}$	m_J
$\{0,1\}$	0000	1111000000000000	0	0
	0010	0000111100000000	1	
	0001	0000000011110000	0	
	0011	0000000000001111	0	
$\{0,2\}$	0000	1100110000000000	1	0
	0100	0011001100000000	0	
	0001	0000000011001100	0	
	0101	0000000000110011	0	
$\{1,2\}$	0000	1010101000000000	0	1
	1000	0101010100000000	1	
	0001	0000000010101010	1	
	1001	0000000001010101	1	
$\{0,3\}$	0000	1100000011000000	0	0
	0100	0011000000110000	0	
	0010	0000110000001100	1	
	0110	0000001100000011	0	
$\{1,3\}$	0000	1010000010100000	0	0
	1000	0101000001010000	0	
	0010	0000101000001010	1	
	1010	0000010100000101	0	
$\{2,3\}$	0000	1000100010001000	1	0
	1000	0100010001000100	0	
	0100	0010001000100010	0	
	1100	0001000100010001	0	

- Calculamos $w(1)$.

$$\begin{aligned} w(1) &= w(2) + \textstyle\sum_{J \subseteq \mathbf{Z}_4} m_J v_J \ \textit{donde}\ |J| = 2 \\ &= 1001000101100110. \end{aligned}$$

- Calculamos $e = 2^{4-2-1} - 1 = 2^1 - 1 = 1$. Como el peso de $w(1) \not\leq e$ entonces continuamos con el procedimiento.

- Sean $i = 1$ y $w(1) = 1001000101100110$.

J	t	$v_{J^c,t}$	$w \cdot v_{J^c,t}$	m_J
$\{0\}$	0000	1100000000000000	1	1
	0100	0011000000000000	1	
	0010	0000110000000000	0	
	0110	0000001100000000	1	
	0001	0000000011000000	1	
	0101	0000000000110000	1	
	0011	0000000000001100	1	
	0111	0000000000000011	1	
$\{1\}$	0000	1010000000000000	1	1
	1000	0101000000000000	1	
	0010	0000101000000000	0	
	1010	0000010100000000	1	
	0001	0000000010100000	1	
	1001	0000000001010000	1	
	0011	0000000000001010	1	
	1011	0000000000000101	1	
$\{2\}$	0000	1000100000000000	1	0
	1000	0100010000000000	0	
	0100	0010001000000000	0	
	1100	0001000100000000	0	
	0001	0000000010001000	0	
	1001	0000000001000100	0	
	0101	0000000000100010	0	
	1101	0000000000010001	0	
$\{3\}$	0000	1000000010000000	1	1
	1000	0100000001000000	1	
	0100	0010000000100000	1	
	1100	0001000000010000	1	
	0010	0000100000001000	0	
	1010	0000010000000100	1	
	0110	0000001000000010	1	
	1110	0000000100000001	1	

- Calculamos $w(0)$.

$$\begin{aligned} w(0) &= w(1) + \textstyle\sum_{J \subseteq \mathbb{Z}_4} m_J v_J \ \textit{donde}\ |J| = 1 \\ &= 0000100000000000. \end{aligned}$$

- Como el peso de $w(0) \leq e$ entonces las m_J restantes son igual a cero y terminamos el procedimiento.

- Por lo tanto el supuesto mensaje es $m = 01011000100$.

6.4. Comentarios

En este capítulo hemos presentado una forma general de codificar y decodificar de los códigos de Reed-Muller. Aun cuando existen formas más rápidas de decodificación cuando se trata de códigos de Reed-Muller de primer orden, es decir, cuando $r = 1$ en $\mathcal{RM}(r, m)$, hemos preferido usar el procedimiento más general presentado en esta capítulo para usarlo en la implementación de la familia de códigos de Reed-Muller.

En el siguiente capítulo explicaremos un código lineal de uso frecuente en nuestos días, los códigos Reed-Solomon.

Capítulo 7

Códigos Reed-Solomon

7.1. Introducción

I.S. Reed y G. Solomon publican en 1958 un artículo en el que dan a conocer los códigos que llevan sus nombres, los códigos Reed-Solomon. Actualmente, en el paso de la tecnología analógica a la tecnología digital, estos códigos han sido los más usados para la detección y corrección de errores. Para ejemplificar la popularidad de estos códigos se suelen mencionar sus aplicaciones en la transmisión de información a través del espacio y además su uso en los discos compactos.

Sin embargo, aun cuando podría parecer extraño en un principio, los códigos Reed-Solomon (RS) no son códigos binarios, pero tienen una representación binaria con la cual ha sido posible su uso. Enseguida presentamos los conceptos que nos van a permitir entender estos códigos.

7.2. El campo GF(2^r)

El objetivo que deseamos cubrir con esta sección es conocer cómo podemos construir el campo de Galois $GF(2^r)$ y cómo se llevan a cabo las operaciones en dicho campo. Por lo tanto, no seremos muy rigurosos con los elementos que presentamos.

El campo de Galois $GF(2^r)$ es un campo finito sobre el cual se construyen los códigos Reed-Solomon. Como ya habíamos mencionado estos códigos no son binarios. Del mismo modo, el campo $GF(2^r)$ tampoco lo es.

Conviene que el lector mantenga en mente los conceptos que presentamos en el capítulo 4 para entender con mayor facilidad las siguientes definiciones.

Definición 7.2.1. Un polinomio $d(x)$ es *divisor* o un *factor* de $f(x)$ si $f(x) = g(x)d(x)$.

Todo polinomio $f(x)$ tiene entre sus divisores al **1** y al mismo $f(x)$, a estos divisores se les llama *triviales*. Los divisores de $f(x)$ que no son triviales, los denominamos *propios*.

Definición 7.2.2. Sea K un campo. Se dice que un polinomio $f(x) \in K[x]$ es *irreducible sobre* K si no tiene divisores propios en $K[x]$; de lo contrario se dice que es *reducible* (o *factorizable*) sobre K.

Ejemplo 7.2.3. Algunos polinomios irreducibles son $f(x) = x$, $g(x) = 1 + x$ y $h(x) = x + x^2$. Algunos polinomios reducibles o factorizables son $p(x) = x^2$, $q(x) = 1+x^2$ y $r(x) = 1+x+x^2+x^3$.

Definición 7.2.4. Sea $p(x)$ un polinomio irreducible sobre el campo K de grado $r > 1$. $p(x)$ es *primitivo*, si $p(x)$ **no** es un divisor de $1 + x^m$ para cualquiera $m < 2^r - 1$.

Sabemos que podemos multiplicar y sumar polinomios módulo un polinomio $h(x)$[1]. Sin embargo, necesitamos que se cumplan ciertas propiedades al llevar a cabo las operaciones entre los polinomios de un conjunto $K^n[x]$, es decir, no solamente requerimos que K sea un campo, sino además necesitamos que $K^n[x]$ sea un campo, también. Para que esta última propiedad se cumpla, requerimos que al efectuar las operaciones entre polinomios módulo $h(x)$, el polinomio $h(x)$ sea *primitivo*. Lo cual crea un campo que conocemos como el campo de Galois, cuyo desarrollo se puede consultar en [6].

Definición 7.2.5. Un campo finito con p^r elementos se le llama un *campo de Galois* de orden p^r y se le denota por $GF(p^r)$.

En nuestro caso, por conveniencia $p = 2$; de tal forma, que estaremos trabajando sobre el campo $GF(2^r)$. Además, como se señala en [8], una importante característica del campo $GF(2^r)$ radica en que cada palabra distinta de cero puede ser representada por alguna potencia[2] de un elemento primitivo $\beta \in GF(2^r)$.

7.2.1. Construcción del campo de Galois $GF(2^r)$

Este proceso es sencillo, si tomamos como guía a los polinomios. Vamos a partir del hecho que existen los polinomios que llevan la siguiente secuencia:

$$0, x^0, x^1, x^2, \ldots, x^{2^r-2}.$$

Una vez que tenemos los 2^r polinomios $p(x)$ anteriores, se lleva a cabo la operación $p(x) \mod h(x)$, donde $h(x)$ es un polinomio primitivo.

Como vimos en la Sección 4.1.2, el polinomio $p(x)$ *mod* $h(x)$ tiene una representación binaria correspondiente. Y por último, el grado del polinomio x^m determina que la potencia correspondiente de β sea β^m, con lo que queda completamente determinado el campo de Galois.

Ejemplo 7.2.6. Queremos construir el campo $GF(2^4)$ con $h(x) = 1 + x + x^4$. Como $r = 4$ entonces la longitud de la palabra binaria es 4. Este campo contiene $2^4 = 16$ elementos y se observa en el Cuadro 7.1.

Sobre este campo es más fácil multiplicar. Por ejemplo

$$(0110)(1101) = \beta^5 \cdot \beta^7 = \beta^{12} = 1111.$$

[1] Véase Sección 4.1.1.

[2] Recuérdese que por tratarse de un campo, se cumple que $a^b a^c = a^{b+c}$ y además $(a^b)^c = a^{b \cdot c}$

palabra	polinomio en	x	(módulo h(x))	potencia de β
0000	0	=	$0 \ mod \ h(x)$	—
1000	1		$1 \ mod \ h(x)$	$\beta^0 = 1$
0100	x	=	$x \ mod \ h(x)$	β
0010	x^2	=	$x^2 \ mod \ h(x)$	β^2
0001	x^3	=	$x^3 \ mod \ h(x)$	β^3
1100	$1+x$	=	$x^4 \ mod \ h(x)$	β^4
0110	$x+x^2$	=	$x^5 \ mod \ h(x)$	β^5
0011	x^2+x^3	=	$x^6 \ mod \ h(x)$	β^6
1101	$1+x+x^3$	=	$x^7 \ mod \ h(x)$	β^7
1010	$1+x^2$	=	$x^8 \ mod \ h(x)$	β^8
0101	$x+x^3$	=	$x^9 \ mod \ h(x)$	β^9
1110	$1+x+x^2$	=	$x^{10} \ mod \ h(x)$	β^{10}
0111	$x+x^2+x^3$	=	$x^{11} \ mod \ h(x)$	β^{11}
1111	$1+x+x^2+x^3$	=	$x^{12} \ mod \ h(x)$	β^{12}
1011	$1+x^2+x^3$	=	$x^{13} \ mod \ h(x)$	β^{13}
1001	$1+x^3$	=	$x^{14} \ mod \ h(x)$	β^{14}

Cuadro 7.1: Construcción de $GF(2^4)$ con el polinomio $h(x) = 1 + x + x^4$.

Y si queremos sumar palabras,

$$(0110) + (1101) = (1011) = \beta^{13}.$$

Aun cuando parece que en la suma, la operación no se simplifica en lo absoluto, resultará útil más adelante cuando necesitemos reducir términos en donde llevaremos a cabo operaciones como la siguiente:

$$\beta + \beta^2 = (0100) + (0010) = (0110) = \beta^5.$$

7.2.2. Propiedades

En este capítulo presentamos una lista de resultados que nos proporcionan información importante sobre el campo de Galois y los códigos Reed-Solomon. La demostración de los mismos pueden ser consultados en [8].

Teorema 7.2.7. Sean $\alpha_1, \alpha_2, \ldots, \alpha_t$ elementos distintos de cero en el campo $GF(2^r)$. Entonces el polinomio $g(x) = (\alpha_1 - x)(\alpha_2 - x)\ldots(\alpha_2 - x)$ genera un código cíclico lineal de longitud $2^r - 1$ sobre el campo $GF(2^r)$.

Teorema 7.2.8. Sea C un código cíclico lineal de longitud n sobre el campo $GF(2^r)$. Entonces cada palabra de código $c(x)$ puede ser escrita de forma única como $m(x)g(x)$ para algún $m(x)$ en $GF(2^r)[x]$ de grado menor que $n - grado(g(x))$. También, $g(x)$ divide a $f(x)$ si y solamente si $g(x)$ divide a $1 + x^n$ y $f(x)$ es una palabra de código.

Corolario 7.2.9. Sea $g(x)$ un polinomio con grado $n - k$. Si $g(x)$ genera un código cíclico lineal C sobre $GF(2^r)$ de longitud $n = 2^r - 1$, y dimensión k, entonces

$$G = \begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$$

es una matriz generadora para C, y el número de palabras de código en C es $(2^r)^k$.

7.3. Códigos Reed-Solomon

Definición 7.3.1. Un código binario *Reed-Solomon* $RS(2^r, \delta)$ es un código cíclico lineal sobre $GF(2^r)$ con polinomio generador $g(x) = (\beta^{m+1}+x)(\beta^{m+2}+x)\ldots(\beta^{m+\delta-1}+x)$ para algún entero m y algún elemento primitivo $\beta \in GF(2^r)$.

Ahora, presentamos un teorema múy útil, cuya demostración se puede consultar en [8].

Teorema 7.3.2. Si C es un código $RS(2^r, \delta)$ entonces

a) $n = 2^r - 1$,

b) $k = 2^r - \delta$,

c) $d = \delta$, y

d) $|C| = 2^{rk}$.

Antes de mostrar un ejemplo en donde usemos los resultados del teorema anterior, deseamos hacer notar que cualquier código $RS(2^r, \delta)$, puede ser representado como un código binario. Este proceso, consiste en reemplazar cada dígito, de cada palabra de código, por la palabra binaria de longitud r dada por $GF(2^r)$. Por lo tanto, si el código original tiene una longitud de $n = 2^r - 1$ entonces el código binario correspondiente tiene una longitud de $n' = r(2^r - 1)$.

Sea $c \in C$; denotaremos como $\hat{c}$ la representación binaria correspondiente a c, y como $\hat{C}$ el código binario correspondiente a C.

Ejemplo 7.3.3. Sea C el código $RS(2^2, 2)$ con polinomio generador $g(x) = \beta + x$ y donde $GF(2^2)$ se construye usando $1 + x + x^2$. Usando el Teorema 7.3.2 tenemos:

a) $n = 2^2 - 1 = 3$,

b) $k = 2^2 - 2 = 4 - 2 = 2$,,

c) $d = 2$, y

d) $|C| = 2^{2\cdot 2} = 2^4 = 16$.

Del Corolario 7.2.9 tenemos que la matriz generadora es

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

Ahora, construimos el campo $GF(2^2)$ con el polinomio $h(x) = 1 + x + x^2$.

palabra	polinomio en x (módulo h(x))	potencia de β
00	$0 = 0 \mod h(x)$	—
10	$1 = 1 \mod h(x)$	$\beta^0 = 1$
01	$x = x \mod h(x)$	β
11	$1+x = x^2 \mod h(x)$	β^2

Como el campo $GF(2^2)$ tiene 4 elementos y además la dimensión del código es $k = 2$ (o sea que la longitud de los mensajes que podemos codificar es 2) entonces tenemos $4^2 = 16$ posibles mensajes, u, que podemos codificar.

A continuación presentamos una tabla donde mostramos cada uno de los 16 mensajes y su correspondiente palabra de código.

$\hat{u}$	u	$c = uG$	$\hat{c}$
0000	00	000	000000
1000	10	$\beta 10$	011000
0100	$\beta 0$	$\beta^2\beta 0$	110100
1100	$\beta^2 0$	$1\beta^2 0$	101100
0010	01	$0\beta 1$	000110
1010	11	$\beta\beta^2 1$	011110
0110	$\beta 1$	$\beta^2 01$	110010
1110	$\beta^2 1$	111	101010

$\hat{u}$	u	$c = uG$	$\hat{c}$
0001	0β	$0\beta^2\beta$	001101
1001	1β	$\beta\beta\beta$	010101
0101	$\beta\beta$	$\beta^2 1\beta$	111001
1101	$\beta^2\beta$	10β	100001
0011	$0\beta^2$	$01\beta^2$	001011
1011	$1\beta^2$	$\beta 0\beta^2$	010011
0111	$\beta\beta^2$	$\beta^2\beta^2\beta^2$	111111
1111	$\beta^2\beta^2$	$1\beta\beta^2$	100111

Cuadro 7.2: Construcción del código $RS(2^2, 2)$ con $g(x) = \beta + x$ y su correspondiente representación binaria .

7.3.1. Decodificación

Si bien el siguiente proceso de decodificación no es el más rápido, es importante porque nos sirve para entender un algoritmo más eficiente, el cual veremos más adelante.

Hasta ahora, todos los códigos que habíamos manejado eran binarios, de tal suerte que sólo necesitábamos conocer las posiciones que creíamos erróneas y negar el bit correspondiente. Sin embargo, como los códigos Reed-Solomon no son binarios, necesitamos detectar las posibles posiciones erróneas y detectar por cuál elemento que está en $GF(2^r)$ se debe sustituir dicha posición.

Nos referiremos a cada posible posición errónea con un *número de posición errónea*. La *magnitud del error* de una posición errónea i es el elemento $\alpha \in GF(2^r)$ que se ubica en la posición i que consideramos el patrón de error más probable.

Ejemplo 7.3.4. Consideremos un código $RS(8, 3)$. Si $c = \beta^3\beta^4\beta^0 0000$ fué transmitida y supongamos que se recibió $w = \beta^3\beta^4\beta^5 0000$, entonces el patrón de error más probable es $c + w = e = 00\beta^4 0000$. Así que el *número de posición errónea* es β^2 y la correspondiente *magnitud del error* es β^4.

A continuación presentamos el algoritmo de decodificación, cuyo desarrollo puede ser consultado en [8].

Algoritmo 7.3.5. Sea a la palabra de código enviada que pertenence al código $RS(2^r, \delta)$ con polinomio generador $g(x) = (\beta^{m+1} + x)\ldots(\beta^{m+\delta-1} + x)$. Sea $t = [(\delta - 1)/2]$. Supongamos que se recibe la palabra w. Para encontrar la posible palabra de código enviada, se llevan a cabo los siguientes pasos:

1. Se calcula $s_j = w(\beta^j)$ para $m + 1 \leq j \leq m + 2t$.
2. Se establece $e = t$, y se encuentra el rango de la matriz extendida M'.

$$M' = \begin{bmatrix} s_{m+1} & s_{m+2} & \cdots & s_{m+e+1} \\ s_{m+2} & s_{m+3} & \cdots & s_{m+e+2} \\ \vdots & \vdots & & \vdots \\ s_{m+e} & s_{m+e+1} & \cdots & s_{m+2e} \end{bmatrix} \tag{7.1}$$

3. Sea e el rango de M'. Se resuelve el siguiente sistema lineal para $\sigma_0, \sigma_1, \ldots, \sigma_{e-1}$.

$$\begin{bmatrix} s_{m+1} & s_{m+2} & \cdots & s_{m+e} \\ s_{m+2} & s_{m+3} & \cdots & s_{m+e+1} \\ \vdots & \vdots & & \vdots \\ s_{m+e} & s_{m+e+1} & \cdots & s_{m+2e-1} \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_{e-1} \end{bmatrix} = \begin{bmatrix} s_{m+e+1} \\ s_{m+e+2} \\ \vdots \\ s_{m+2e} \end{bmatrix} \tag{7.2}$$

4. Se encuentran las raíces de $\sigma_A(x) = \sigma_0 + \sigma_1 x + \ldots + x^e$; estas raíces son los *números de posiciones erróneas* $a_1, \ldots, a_e$.
5. Se resuelve el siguiente sistema de ecuaciones para $b_1, \ldots, b_e$; las cuales son las magnitudes de error correspondientes a $a_1, \ldots, a_e$. Por lo tanto, el patrón de error más probable c_e está completamente determinado.

$$\begin{bmatrix} a_1^{m+1} & a_2^{m+1} & \cdots & a_e^{m+1} \\ a_1^{m+2} & a_2^{m+2} & \cdots & a_e^{m+2} \\ \vdots & \vdots & & \vdots \\ a_1^{m+e} & a_2^{m+e} & \cdots & a_e^{m+e} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_e \end{bmatrix} = \begin{bmatrix} s_{m+1} \\ s_{m+2} \\ \vdots \\ s_{m+e} \end{bmatrix} \tag{7.3}$$

6. Para encontrar la supuesta palabra de código sumamos el patrón de errores c_e con la palabra recibida w.

$$c = c_e + w$$

7. Por último para obtener el supuesto mensaje m_s, dividimos el polinomio equivalente a la supuesta palabra de código $c(x)$, entre el polinomio generador $g(x)$.

$$m_s(x) = c(x) \div g(x),$$

y obtenemos la cadena de Galois m_s equivalente al polinomio $m_s(x)$.

Ahora presentamos un ejemplo del uso de este algoritmo.

Ejemplo 7.3.6. Sea $g(x) = (1+x)(\beta+x)(\beta^2+x)(\beta^3+x)(\beta^4+x)(\beta^5+x)$ el polinomio generador de un código $RS(2^4, 7)$, donde $GF(2^4)$ se construye usando $1+x+x^4$, es decir, se trata del campo que construimos en el Ejemplo 7.2.6. Por lo tanto, $m = -1$ y $t = 3$.

Supongamos que la palabra recibida es
$w(x) = 1 + \beta^4 x + \beta x^3 + \beta^9 x^5 + x^6$ la cual es equivalente a $1\beta^4 0\beta 0\beta^9 100000000$.

Ahora.
$g(x) = (1+x)(\beta+x)(\beta^2+x)(\beta^3+x)(\beta^4+x)(\beta^5+x) = 1+\beta^4 x+\beta^2 x^2+\beta x^3+\beta^{12}x^4+\beta^9 x^5+x^6$.

1. Se calculan $s_j = w(\beta^j)$ para $0 \leq j \leq 5$.

$$\begin{array}{rcl}
s_0 = w(\beta^0) = 1+\beta^4(\beta^0)+\beta(\beta^0)^3+\beta^9(\beta^0)^5+(\beta^0)^6 & = & \beta+\beta^4+\beta^9 = \beta^7. \\
s_1 = w(\beta) = 1+\beta^4(\beta)+\beta(\beta)^3+\beta^9(\beta)^5+(\beta)^6 & = & 1+\beta^5+\beta^4+\beta^{14}+\beta^6 = \beta^0. \\
s_2 = w(\beta^2) = 1+\beta^4(\beta^2)+\beta(\beta^2)^3+\beta^9(\beta^2)^5+(\beta^2)^6 & = & 1+\beta^6+\beta^7+\beta^4+\beta^{12} = \beta^9. \\
s_3 = w(\beta^3) = 1+\beta^4(\beta^3)+\beta(\beta^3)^3+\beta^9(\beta^3)^5+(\beta^3)^6 & = & 1+\beta^7+\beta^{10}+\beta^9+\beta^3 = \beta^{12}. \\
s_4 = w(\beta^4) = 1+\beta^4(\beta^4)+\beta(\beta^4)^3+\beta^9(\beta^4)^5+(\beta^4)^6 & = & 1+\beta^8+\beta^{13}+\beta^{14}+\beta^9 = \beta^9. \\
s_5 = w(\beta^5) = 1+\beta^4(\beta^5)+\beta(\beta^5)^3+\beta^9(\beta^5)^5+(\beta^5)^6 & = & 1+\beta^9+\beta+\beta^4+1 = \beta^7.
\end{array}$$

2. Se establece $e = t = 3$, y se encuentra el rango de la matriz extendida M'.

$$M' = \begin{bmatrix} s_0 & s_1 & s_2 & s_3 \\ s_1 & s_2 & s_3 & s_4 \\ s_2 & s_3 & s_4 & s_5 \end{bmatrix} = \begin{bmatrix} \beta^7 & 1 & \beta^9 & \beta^{12} \\ 1 & \beta^9 & \beta^{12} & \beta^9 \\ \beta^9 & \beta^{12} & \beta^9 & \beta^7 \end{bmatrix}$$

$$\leftrightarrow \begin{bmatrix} \beta^7 & 1 & \beta^9 & \beta^{12} \\ 0 & \beta^{12} & \beta^7 & \beta^6 \\ 0 & \beta^7 & \beta^2 & \beta \end{bmatrix} \leftrightarrow \begin{bmatrix} \beta^7 & 1 & \beta^9 & \beta^{12} \\ 0 & \beta^{12} & \beta^7 & \beta^6 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Por lo tanto, el rango de M' es 2.

3. Sea $e = 2$.

$$\begin{bmatrix} s_0 & s_1 \\ s_1 & s_2 \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} s_2 \\ s_3 \end{bmatrix}$$

Esto es,

$$\begin{bmatrix} \beta^7 & 1 \\ 1 & \beta^9 \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \beta^9 \\ \beta^{12} \end{bmatrix}$$

$$\leftrightarrow \begin{bmatrix} \beta^7 & 1 \\ 0 & \beta^{12} \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \beta^9 \\ \beta^7 \end{bmatrix}$$

Así que tenemos:

$$\beta^7\sigma_0 + \sigma_1 = \beta^9 \tag{7.4}$$
$$\beta^{12}\sigma_1 = \beta^7 \tag{7.5}$$

De la ecuación (7.5) tenemos que $\sigma_1 = \beta^{10}$.

Sustituyendo σ_1 en la ecuación (7.4) tenemos que $\beta^7\sigma_0 = \beta^9 + \beta^{10} = \beta^{13}$. Así que $\sigma_0 = \beta^6$

Por lo tanto $\sigma_0 = \beta^6$ y $\sigma_1 = \beta^{10}$.

4. Buscamos las raíces de $\sigma_A(x) = \sigma_0 + \sigma_1 x + x^2$.

   $\sigma_A(x) = \beta^6 + \beta^{10}x + x^2 = (x + \beta^2)(x + \beta^4)$.

   Por lo tanto $a_1 = \beta^2$ y $a_2 = \beta^4$. Estos valores constituyen los números de las posibles posiciones erróneas.

5. Resolvemos el siguiente sistema de ecuaciones lineales para b_1 y b_2.

$$\begin{bmatrix} a_1^0 & a_2^0 \\ a_1^1 & a_2^1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \end{bmatrix}$$

   Esto es,

$$\begin{bmatrix} (\beta^2)^0 & (\beta^4)^0 \\ (\beta^2)^1 & (\beta^4)^1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \beta^7 \\ 1 \end{bmatrix}$$

$$\leftrightarrow \begin{bmatrix} 1 & 1 \\ 0 & \beta^{10} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \beta^7 \\ \beta^7 \end{bmatrix}$$

   Así que tenemos:

$$b_1 + \beta^{12} = \beta^7 \tag{7.6}$$

$$\beta^{10}b_2 = \beta^7. \tag{7.7}$$

   De la ecuación (7.7) tenemos que $b_2 = \beta^{12}$.

   Sustituyendo b_2 en la ecuación (7.6) tenemos que $b_1 = \beta^7 + \beta^{12} = \beta^2$.

   Por lo tanto las magnitudes de error de las posiciones $a_1 = \beta^2$ y $a_2 = \beta^4$ son $b_1 = \beta^2$ y $b_2 = \beta^{12}$, respectivamente.

   Así que el patrón de error c_e es:

0	0	β^2	0	β^{12}	0	0	0	0	0	0	0	0	0	0
β^0	β^1	$a_1 = \beta^2$	β^3	$a_2 = \beta^4$	β^5	β^6	β^7	β^8	β^9	β^{10}	β^{11}	β^{12}	β^{13}	β^{14}

6. La posible palabra de código enviada es

$$c = w + c_e = 1\beta^4 0\beta 0\beta^9 100000000 + 00\beta^2 0\beta^{12} 0000000000 = 1\beta^4\beta^2\beta\beta^{12}\beta^9 100000000.$$

7. Recuperamos el mensaje.

$$m_s(x) = c(x) \div g(x)$$
$$m_s(x) = (1+\beta^4 x+\beta^2 x^2+\beta x^3+\beta^{12}x^4+\beta^9 x^5+x^6) \div (1+\beta^4 x+\beta^2 x^2+\beta x^3+\beta^{12}x^4+\beta^9 x^5+x^6)$$
$$m_s(x) = 1.$$

Ahora, si bien ya hemos obtenido el supuesto mensaje, el lector no debe olvidar que este mensaje es un polinomio en el campo de Galois $GF(2^4)$, por lo que primero traducimos el polinomio $m_s(x)$ en la cadena de Galois m_s, esto es:

$m_s(x) = 1$ es equivalente a $m_s = 100000000$.

Por último, traducimos la cadena de Galois m_s en la cadena binaria correspondiente, como se mostró en la Sección 7.2.1.

$m_s = 100000000$ es equivalente a 100000000000000000000000000000000000.

7.4. Algoritmo Berlekamp-Massey

La dificultad más fuerte que se presentaba con los códigos Reed-Solomon, y en la que aún se sigue trabajando, es la complejidad del proceso de decodificación. Uno de los procedimientos más usados para llevar a cabo la decodificación es conocido como el algoritmo de Berlekamp-Massey, el cual presentamos en esta sección.

La idea central de este algoritmo es optimizar la obtención del polinomio $\sigma(x)$, esto es, el polinomio localizador de errores.

Algoritmo 7.4.1. Sea w la palabra recibida que fue codificada usando el código $RS(2^r, \delta)$ con polinomio generador $g(x) = (\beta^{m+1} + x) + (\beta^{m+2} + x) + \ldots + (\beta^{m+\delta-1} + x)$, y sea $t = [(\delta - 1)/2]$ la capacidad de corrección del código. Se decodifica w como sigue:

1. Se calculan $s_j = w(\beta^j)$ para $m + 1 \leq j \leq m + 2t$.

2. Se definen
$$\begin{aligned} q_{-1}(x) &= 1 + s_{m+1}x + s_{m+2}x^2 + \ldots + s_{m+2t}x^{2t}, \\ q_0(x) &= s_{m+1} + s_{m+2}x + \ldots + s_{m+2t}x^{2t-1}, \\ p_{-1}(x) &= x^{2t+1}, \text{ y} \\ p_0(x) &= x^{2t}. \end{aligned}$$
Sean $d_{-1} = -1$, $d_0 = 0$ y $z_0 = -1$.

3. Para $1 \leq i \leq 2t$, se definen q_i, p_i, d_i y z_i como sigue:
Si $q_{i-1}(0) = 0$ entonces
$$\begin{aligned} q_i(x) &= q_{i-1}(x)/x, \\ p_i(x) &= p_{i-1}(x)/x, \\ d_i &= d_{i-1} + 1, \text{ y} \\ z_i &= z_{i-1}. \end{aligned}$$
Pero si $q_{i-1}(0) \neq 0$ entonces
$$\begin{aligned} q_i(x) &= (q_{i-1}(x) + (q_{i-1}(0)q_{z_{i-1}}(0))q_{z_{i-1}}(x))x, \\ p_i(x) &= (p_{i-1}(x) + (q_{i-1}(0)q_{z_{i-1}}(0))p_{z_{i-1}}(x))x, \\ d_i &= 1 + \min d_{i-1}, d_{z_{i-1}}, \text{ y} \\ z_i &= \begin{cases} i-1 & \text{si } d_{i-1} \geq d_{z_{i-1}}; \\ z_{i-1} & \text{en otro caso.} \end{cases} \end{aligned}$$

4. Si no se ha rebasado la capacidad de corrección del código, es decir, que el número de errores ocurridos e no ha superado a t, entonces el polinomio $p_{2t}(x)$ presenta los *mismos coeficientes* que tiene el polinomio $\sigma(x)$. Ambos polinomios tienen la misma cantidad de sumandos y tienen el mismo grado, para construir $\sigma(x)$ tomaremos los coeficientes de $p_{2t}(x)$ en el orden inverso.

   Dicho en símbolos, tenemos:

   Sea e el número de errores ocurridos. El polinomio corrector de errores $\sigma(x)$ tiene la siguiente forma:

$$\sigma(x) = p_{2t}(e) + p_{2t}(e-1)x + \ldots + p_{2t}(1)x^{e-1} + x^e.$$

Ejemplo 7.4.2. Considérese el Ejemplo 7.3.6.

1. En dicho ejemplo tenemos los siguientes síndromes:

$$\begin{array}{lll} s_0 = \beta^7, & & s_3 = \beta^{12}, \\ s_1 = \beta^0, & & s_4 = \beta^9, \\ s_2 = \beta^9, & & s_5 = \beta^7. \end{array}$$

2. Definimos:

$$\begin{array}{rcl} q_{-1}(x) & = & 1 + \beta^7 x + x^2 + \beta^9 x^3 + \beta^{12} x^4 + \beta^9 x^5 + \beta^7 x^6 \\ q_{-1}(x) & = & \beta^7 x + x + \beta^9 x^2 + \beta^{12} x^3 + \beta^9 x^4 + \beta^7 x^5 \\ p_{-1}(x) & = & x^7 \\ p_0(x) & = & x^6 \end{array}$$

   Sean $d_{-1} = -1$, $d_0 = 0$ y $z_0 = -1$

3. Para $1 \leq i \leq 6$, se definen q_i, p_i, d_i y z_i de la siguiente forma:

   - **i = 1**
     $q_0(0) = \beta^7$.
     Como $\beta^7 \neq 0$, tenemos:

$$\begin{array}{rcll} q_1(x) & = & \dfrac{q_0(x) + \dfrac{q_0(0)}{q_{-1}(0)} q_{-1}(x)}{x} & = \beta^3 + x + \beta^{13} x^2 + \beta^{14} x^3 + \beta^{14} x^4 + \beta^{14} x^5 \\ p_1(x) & = & \dfrac{p_0(x) + \dfrac{q_0(0)}{q_{-1}(0)} p_{-1}(x)}{x} & = 1 + \beta^7 x \\ d_1 & = & 1 + \min\{d_0, d_{-1}\} = 1 + \min\{0, -1\} & = 0 \\ z_1 & = & 1 - 1 \ \text{ porque } d_0 \geq d_{-1} & = 0 \end{array}$$

- **i = 2**
  $q_1(0) = \beta^3$.
  Como $\beta^3 \neq 0$, tenemos:

$$\begin{aligned}
q_2(x) &= \frac{q_1(x) + \frac{q_1(0)}{q_0(0)} q_0(x)}{x} &&= \beta^{12} + \beta^7 x + \beta^6 x^2 + \beta^{12} x^3 \\
p_2(x) &= \frac{p_1(x) + \frac{q_1(0)}{q_0(0)} p_0(x)}{x} &&= 1 + \beta^8 x \\
d_2 &= 1 + \text{mín}\{d_1, d_0\} = 1 + \text{mín}\{0, 0\} &&= 1 \\
z_2 &= 2 - 1 \quad \text{ya que } d_1 \geq d_0 &&= 1
\end{aligned}$$

- **i = 3**
  $q_2(0) = \beta^{12}$.
  Como $\beta^{12} \neq 0$, tenemos:

$$\begin{aligned}
q_3(x) &= \frac{q_2(x) + \frac{q_2(0)}{q_1(0)} q_1(x)}{x} &&= \beta^9 + \beta^{10} x + \beta^9 x^2 \\
p_3(x) &= \frac{p_2(x) + \frac{q_2(0)}{q_1(0)} p_1(x)}{x} &&= 1 + \beta^{12} x + \beta x^2 \\
d_3 &= 1 + \text{mín}\{d_2, d_1\} = 1 + \text{mín}\{1, 0\} &&= 1 \\
z_3 &= 3 - 1 \quad \text{pues } d_2 \geq d_1 &&= 2
\end{aligned}$$

- **i = 4**
  $q_3(0) = \beta^0$.
  Como $\beta^0 \neq 0$, tenemos:

$$\begin{aligned}
q_4(x) &= \frac{q_3(x) + \frac{q_3(0)}{q_2(0)} q_2(x)}{x} &&= 0 \\
p_4(x) &= \frac{p_3(x) + \frac{q_3(0)}{q_2(0)} p_2(x)}{x} &&= 1 + \beta^{10} x + \beta^6 x^2 \\
d_4 &= 1 + \text{mín}\{d_3, d_2\} = 1 + \text{mín}\{1, 1\} &&= 2 \\
z_4 &= 4 - 1 \quad \text{porque } d_3 \geq d_2 &&= 3
\end{aligned}$$

- **i = 5**
  $q_4(0) = 0$.
  Como $q_4(0) = 0$, tenemos:

$$\begin{aligned} q_5(x) &= \frac{q_4(x)}{x} &&= 0 \\ p_5(x) &= \frac{p_4(x)}{x} &&= x + \beta^{10}x^2 + \beta^6 x^3 \\ d_5 &= d_4 + 1 &&= 3 \\ z_5 &= z_4 &&= 3 \end{aligned}$$

- **i = 6**
  $q_5(0) = 0$.
  Como $q_5(0) = 0$, tenemos:

$$\begin{aligned} q_6(x) &= \frac{q_5(x)}{x} &&= 0 \\ p_6(x) &= \frac{p_5(x)}{x} &&= 1 + \beta^{10}x + \beta^6 x^2 \\ d_6 &= d_5 + 1 &&= 4 \\ z_6 &= z_5 &&= 3 \end{aligned}$$

Por último, sabemos que

$$p_{2t}(x) = p_6(x) = 1 + \beta^{10}x + \beta^6 x^2,$$

y tomando los coeficientes del polinomio anterior en orden inverso, construimos el polinomio detector de errores $\sigma(x)$:

$$\sigma(x) = \beta^6 + \beta^{10}x + x^2.$$

7.5. Comentarios

Con esto concluimos nuestra revisión sobre los códigos Reed-Solomon. Como el lector pudo observar, el proceso de decodificación fue más elaborado que aquellos presentados anteriormente. Debido a la complejidad de la decodificación de estos códigos, se dificultó que se empezara su uso en aplicaciones concretas, es en los años recientes cuando se han podido implementar con buenos resultados.

En este momento ya conocemos los cuatro códigos lineales que mencionamos en la presentación: Golay, Hamming, Reed-Muller y Reed-Solomon. Así que en el siguiente capítulo solo resta llevar a cabo la comparación entre éstos usando los dos tipos de ruido al simular la transmisión de los bits: el ruido blanco y el ruido con ráfagas.

Capítulo 8

Comparación de los códigos

En este capítulo mostramos los resultados de llevar a cabo la comparación entre los códigos: Hamming, Golay, Reed-Muller y Reed-Solomon. Sin embargo, antes de mostrar los resultados señalaremos algunas notas de implementación que se siguieron para la programación de los mismos.

8.1. Implementación del modelo de transmisión

Las implementaciones que se hicieron de cada uno de estos códigos siguen los algoritmos de codificación y decodificación que se vieron en los capítulos anteriores. Sin embargo, destacaremos algunas cualidades de dicha implementación.

Nuestra intención, no es abrumar al lector con los detalles de cada programa, pues cada uno de ellos ya está documentado en el programa mismo, si no de proporcionar un esquema general del diseño que les dió origen.

8.1.1. Implementación de los códigos

Usamos el paradigma de orientación a objetos para la programación de los códigos. Bajo este paradigma, pensamos en cada uno de los códigos como una clase. Ahora, considerando que todos son códigos lineales, podemos encontrar elementos comunes como fue el proceso de codificación de un mensaje. Por tal motivo, se implementó una clase abstracta a la que llamamos *AbstractErrorDC* para reunir estos elementos.

Por otro lado, hicimos uso de una interfaz que proporciona las declaraciones comunes a todos los códigos lineales.

La programación de estos códigos se llevó a cabo usando el lenguaje de programación Java, el cual se apega al paradigma de orientación a objetos.

Las siguientes clases constituyen la parte más importante, pues forman una biblioteca de códigos que se usa para llevar a cabo los experimentos mostrados en el resto del capítulo.

- AbstractErrorDC.java
- ErrorDC.java
- Golay.java
- Hamming.java
- ReedMuller.java
- ReedSolomon.java

En la Figura 8.1 mostramos cuál es la relación entre estas clases usando la notación UML.

A excepción de la clase Golay, la cual implementa el código G_{24}, el resto de las clases implementan las familias de códigos, es decir, no se implementó un código específico si no una forma general para instanciar el código solicitado por el usuario.

8.1.2. Implementación de los canales de transmisión

Para poder simular la transmisión de los bits de información y someterlos a ruido, se implementaron los modelos de canales de transmisión revisados en el Capítulo 2: el canal simétrico binario y el canal de ráfagas. Este último usando el modelo de Gilbert-Elliot que se revisó en la Sección 2.3.

Las clases que se crearon para implementar estos canales son:

- Channel.java. Es una interfaz que mantiene la declaración común a ambos tipos de canales.
- BSChanel.java. Implementa el canal simétrico binario.
- GEChannel.java. Implementa el canal de Gilbert-Elliot.

Por otro lado se implementaron dos clases que simulan la transmisión de los bits, los cuales son leídos de un archivo generado aleatoriamente, usando cualquiera de los dos tipos de canales. Estas clases son:

- EncodeDecodeFile.java. Esta clase se encarga de leer o escribir las cadenas de bits en los archivos fuente y destino, respectivamente. Además se encarga de usar cualquiera de los códigos lineales que se implementaron, para codificar o decodificar la cadena de bits.
- FileTransmission.java. En esta clase se lleva a cabo la simulación de la transmisión de los bits sucesivamente usando cualquiera de los dos modelos de canales de transmisión.

En la Figura 8.2 mostramos la relación entre las clases mencionadas.

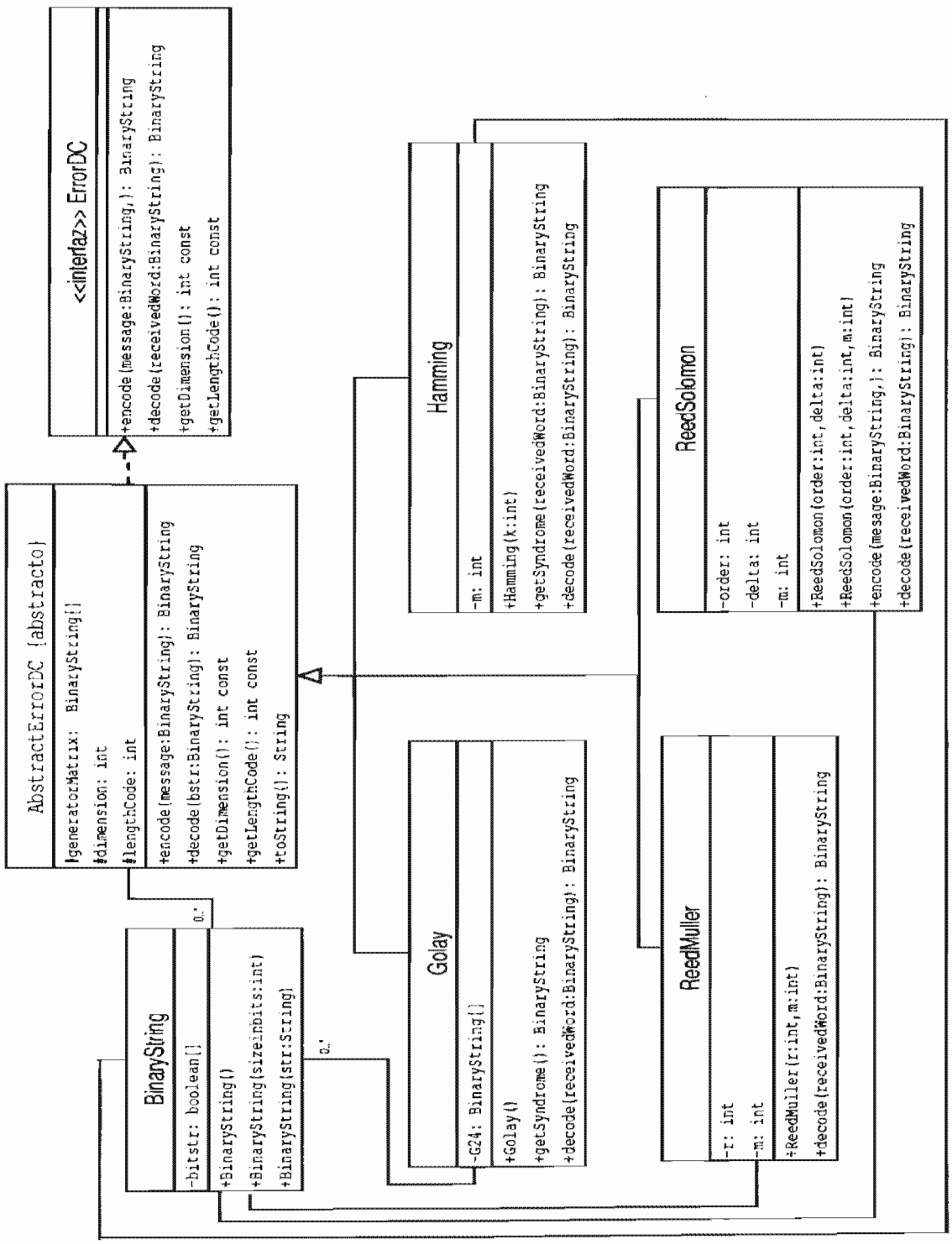


Figura 8.1: Diagrama de clases de los códigos correctores y detectores de errores.

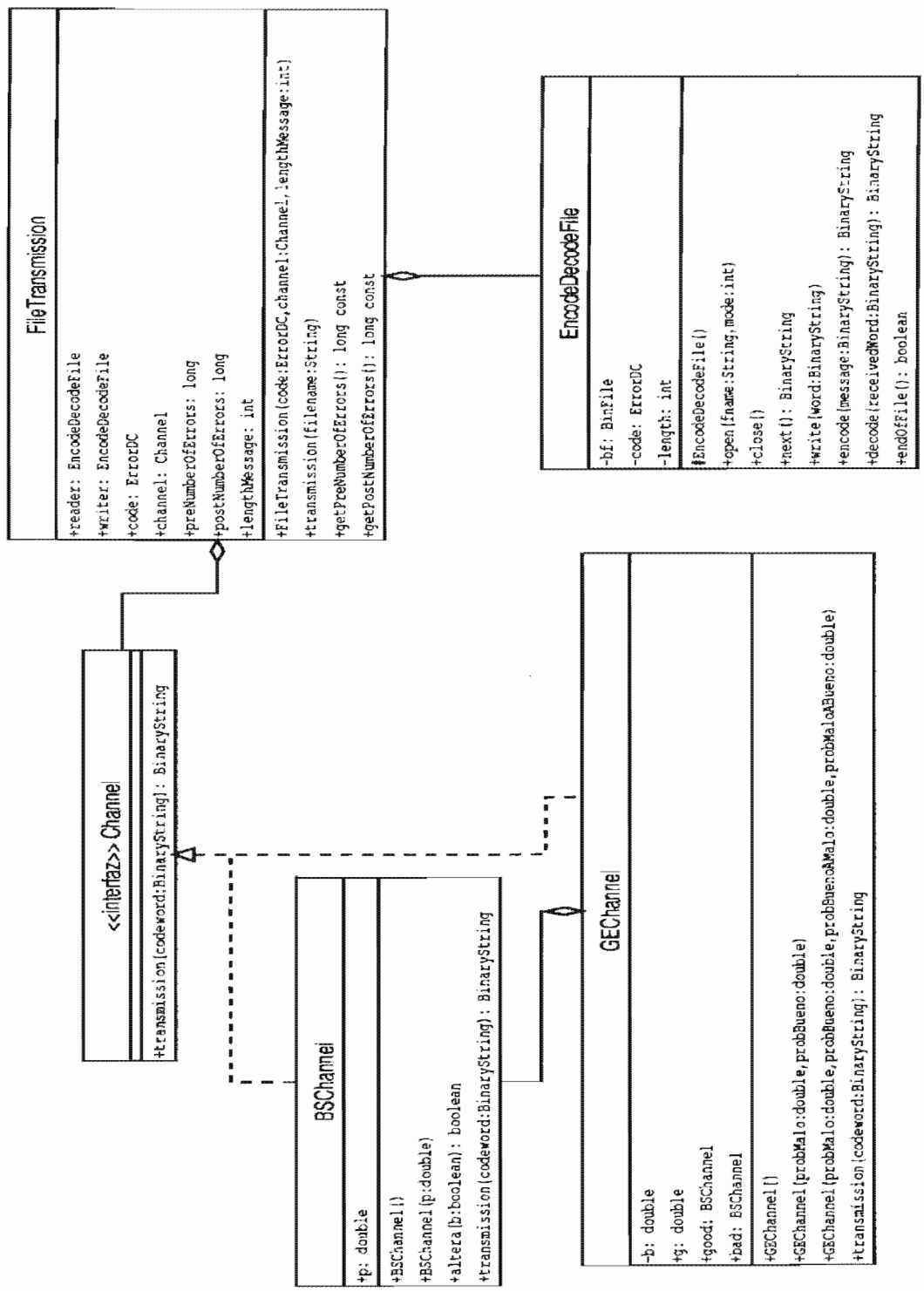


Figura 8.2: Diagrama de clases de los tipos de canales de transmisión.

8.1.3. Comentarios finales sobre la implementación

Aún cuando se implementaron más clases para facilitar el registro de los experimentos que se muestran en las siguientes secciones de este capítulo, la idea general de la implementación se resume en las clases anteriores representadas en sus respectivas figuras. Si el lector desea ver los detalles de los programas que se hicieron como parte de este trabajo, puede observar el código mismo o bien la documentación en formato HTML generada por la herramienta *javadoc*[1].

8.2. Modelo experimental

Ahora contamos con lo necesario para llevar a cabo la comparación entre los códigos que revisamos. En términos generales lo que haremos es realizar experimentos[2], los cuales consisten en simular la transmisión de un archivo por un canal con ruido y comparar los resultados arrojados por la transmisión con los datos originales.

Hipótesis

Considerando que conocemos las capacidades de corrección de cada código, suponemos que si un código tiene una capacidad de corrección de d bits por palabra de código de longitud n entonces tendrá un comportamiento satisfactorio en canales con probabilidad de cruzamiento inferior a $\frac{1}{n}$. Por otro lado, creemos que debido al tipo de decodificación que usa el código Reed-Solomon, es este código el que presentará un mejor desempeño.

En nuestro modelo experimental tendremos presente las siguientes variables:

- Tipo de código.
  - Longitud del mensaje.
  - Longitud de la palabra de código.
  - Capacidad de corrección por palabra de código.
  - Cardinalidad.
- Tipo de canal.
  - Probabilidad(es) del canal.

Los elementos que mediremos como resultado de cada experimento son:

T_e	Tasa de errores antes de la decodificación.
T_{ep}	Tasa de errores después de la decodificación.
T_d	Tasa de errores de los bits de datos antes de la decodificación.
T_{dp}	Tasa de errores de los bits de datos después de la decodificación.

[1] http://pateame.fciencias.unam.mx/tc/ligas.html

[2] Usando el enfoque de la probabilidad, entendemos por experimento el proceso por medio del cual una observación queda registrada.

Cada experimento arroja elementos importantes que debemos mantener presentes. Para facilitar el manejo de estos elementos usaremos la notación que presentamos en el siguiente cuadro:

P_c	Probabilidad de cruzamiento del canal.
B_b	Número total de bits transmitidos por el canal.
B_e	Número total de bits erróneos **antes** de la decodificación.
B_{ep}	Número total de bits erróneos **después** de la decodificación.
D_b	Número total de bits de datos transmitidos por el canal.
D_e	Número de bits de datos erróneos **antes** de la decodificación.
D_{ep}	Número de bits de datos erróneos **después** de la decodificación.

Cuadro 8.1: Notación

Por otro lado, en relación a las tasas de error tenemos:

$$T_e = \frac{B_e}{B_b}, \qquad T_d = \frac{D_e}{D_b},$$

$$T_{ep} = \frac{B_{ep}}{B_b}, \qquad T_{dp} = \frac{D_{ep}}{D_b}.$$

Además consideraremos la tasa de transmisión y la tasa de corrección que definimos en la Sección 2.11, las cuales se representan como $R(C)$ y $\delta(C)$ respectivamente, donde C es un (n, M) código.

$$R(C) = \frac{log_2(M)}{n} \qquad \delta(C) = \frac{\lfloor \frac{d-1}{2} \rfloor}{n}$$

8.2.1. Canales de transmisión usuales

Antes de empezar con los experimentos deseamos proporcionar los siguientes datos sobre las tasas de error en distintos canales. Los datos del Cuadro 8.2 fueron obtenidos de [2, 4, 9].

Canal	**Tasa de error**
CD-ROM	10^{-4}
Telefonía estándar	10^{-4}
Telefonía de alta calidad	10^{-6}
Microondas	10^{-6}
Fast Ethernet	10^{-10}
Fibra óptica	10^{-12}

Cuadro 8.2: Tasas de error

8.3. Comportamiento de los códigos usando el canal simétrico binario

El canal simétrico binario fué discutido en la Sección 2.2. En este tipo de canal el principal elemento es la probabilidad de cruzamiento del canal P_c, debido a que ésta define la cantidad de ruido en el mismo.

Recuérdese que el canal simétrico binario es un canal *sin memoria*, esto es, que el error al cual se ve afectado un bit es independiente de los errores ocurridos en el resto de los bits. Esto ocasiona que los errores estén distribuidos a lo largo de las palabras de bits que son transmitidas.

Por último, en cuanto a la información que presentamos en este capítulo, se proporciona una tabla con los resultados de los experimentos de someter un archivo a distintos valores de la probabilidad de cruzamiento del canal.

Por cada valor de la probabilidad de cruzamiento del canal (P_c), se simuló la transmisión del archivo por el canal, diez veces. Sin embargo, en las tablas que presentan los resultados, sólo se exponen los promedios de estos diez experimentos por cada renglón.

8.3.1. Códigos de Golay

A continuación presentamos las características del código de Golay G_{24} y los resultados de los experimentos usando el canal simétrico binario.

Tipo de código:	Golay	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	12 bits	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	24 bits		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	3 bits		0.14, 0.16, 0.2, 0.3,
Cardinalidad (M):	4096		0.4 y 0.5.
Tasa de transmisión:	0.5	**Tasa de corrección:**	0.125

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	16392	18.7	0.0	0.0011	0.0	8196	10.8	0.0	0.0013	0.0
0.005	16392	79.3	0.0	0.0048	0.0	8196	41.8	0.0	0.0051	0.0
0.01	16392	169.4	0.0	0.0103	0.0	8196	84.6	0.0	0.0103	0.0
0.02	16392	328.7	3.2	0.0200	0.0001	8196	164.3	1.9	0.0200	0.0002
0.04	16392	652.5	47.2	0.0398	0.0028	8196	325.7	24.4	0.0397	0.0029
0.06	16392	989.5	179.6	0.0603	0.0109	8196	492.9	87.2	0.0601	0.0106
0.08	16392	1297.2	395.8	0.0791	0.0241	8196	646.2	197.2	0.0788	0.0240
0.1	16392	1632.7	777.6	0.0996	0.0474	8196	819.5	388.8	0.0999	0.0474
0.14	16392	2319.8	1852.0	0.1415	0.1129	8196	1159.3	921.9	0.1414	0.1124
0.16	16392	2628.5	2325.4	0.1603	0.1418	8196	1308.3	1160.1	0.1596	0.1415
0.2	16392	3279.4	3351.8	0.2000	0.2044	8196	1643.4	1677.5	0.2005	0.2046
0.3	16392	4937.0	5326.2	0.3011	0.3249	8196	2454.0	2648.2	0.2994	0.3231
0.4	16392	6569.1	6811.6	0.4007	0.4155	8196	3299.1	3418.6	0.4025	0.4171
0.5	16392	8225.7	8224.2	0.5018	0.5017	8196	4114.9	4098.4	0.5020	0.5000

Tabla 8.2.1.1. Resultados de los experimentos realizados con el código de Golay G_{24} sobre un canal simétrico binario.

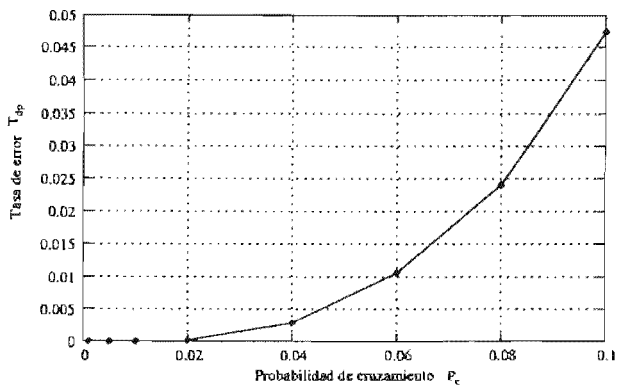


Figura 8.3: Comportamiento del código de Golay G_{24}.

Comentarios sobre el código de Golay G_{24}

La capacidad de corrección del código de Golay G_{24} es de 1 bit por cada 8 bits. De tal suerte que si el canal tiene una probabilidad de cruzamiento $P_c = 0.1$, supondríamos que el código puede recuperar todos los errores, pero esto ¡no ocurrió! ¿por qué?

Al observar con cuidado la transmisión podemos ver que hay una gran cantidad de palabras de código que no necesariamente presentaron 3 bits erróneos. Esto significa que hay palabras de código que tendrán más de 3 bits erróneos, lo que *excede* la capacidad de corrección del código. Esta observación no es específica del código de Golay, sino que es general al comportamiento de los códigos frente a la cantidad de ruido que presenta cualquier canal.

No obstante, ya aclarado el comportamiento anterior, podemos observar que bajo las tres primeras probabilidades de cruzamiento se tiene una recuperación *completa* del archivo. Esto nos muestra un muy buen desempeño del código.

8.3.2. Códigos de Hamming

Presentaremos los resultados de la transmisión de los bits usando los códigos de Hamming $H_2(4)$, $H_2(8)$ y $H_2(12)$.

Código de Hamming $H_2(4)$

Tipo de código:	$H_2(4)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	4 bits	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	7 bits		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	1 bit		0.14, 0.16, 0.2, 0.3,
Cardinalidad (M):	16		0.4 y 0.5.
Tasa de transmisión:	0.5714	**Tasa de corrección:**	0.142857

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	14336	14.6	0.0	0.0010	0.0	8192	7.9	0.0	0.0009	0.0
0.005	14336	70.2	3.9	0.0048	0.0002	8192	40.3	2.3	0.0049	0.0002
0.01	14336	143.1	9.9	0.0099	0.0006	8192	83.1	5.6	0.0101	0.0006
0.02	14336	283.9	49.5	0.0198	0.0034	8192	160.3	28.2	0.0195	0.0034
0.04	14336	568.2	178.7	0.0396	0.0124	8192	324.4	103.2	0.0395	0.0125
0.06	14336	854.6	382.5	0.0596	0.0266	8192	487.5	217.8	0.0595	0.0265
0.08	14336	1147.1	654.6	0.0800	0.0456	8192	662.8	374.3	0.0809	0.0456
0.1	14336	1439.1	959.8	0.1003	0.0669	8192	833.3	551.4	0.1017	0.0673
0.14	14336	2018.8	1678.4	0.1408	0.1170	8192	1157.1	952.6	0.1412	0.1162
0.16	14336	2290.8	2054.9	0.1597	0.1433	8192	1318.3	1185.5	0.1609	0.1447
0.2	14336	2876.8	2813.9	0.2006	0.1962	8192	1645.1	1607.4	0.2008	0.1962
0.3	14336	4294.3	4642.7	0.2995	0.3238	8192	2457.3	2661.2	0.2999	0.3248
0.4	14336	5740.3	6033.3	0.4004	0.4208	8192	3293.8	3457.0	0.4020	0.4219
0.5	14336	7129.2	7151.5	0.4972	0.4988	8192	4079.0	4092.2	0.4979	0.4995

Tabla 8.2.2.1. Resultados de los experimentos realizados con el código de Hamming $H_2(4)$ sobre un canal simétrico binario.

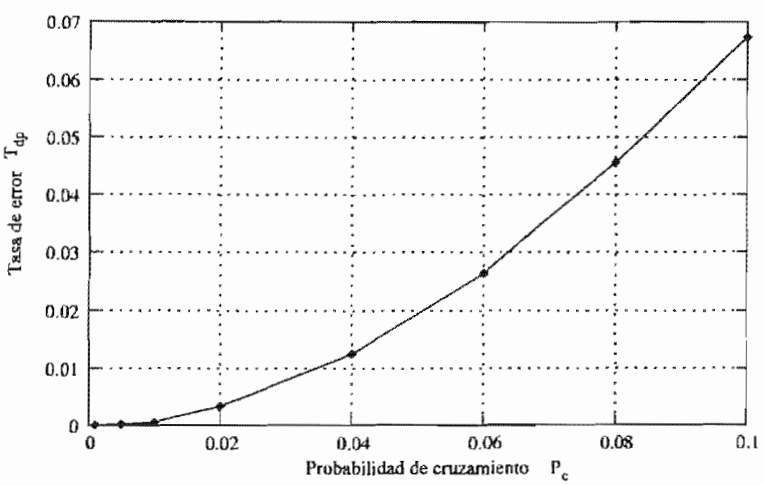


Figura 8.4: Comportamiento del código de Hamming $H_2(4)$.

De los tres códigos de Hamming que revisamos en esta sección, son precisamente los códigos $H_2(4)$ los que presentan una tasa corrección más alta. Sin embargo, veremos más adelante que entre los códigos de Golay, Hamming, Reed-Muller y Reed-Solomon, son los códigos de Hamming los que tienen el desempeño más bajo. Sin embargo, no deben de ser ignorados ya que siempre el código corrector de errores más adecuado esta en función de las características del canal sobre el cual se transmitirán los datos y además son los códigos de Hamming los que presentan el procedimiento de decodificación más sencillo.

Código de Hamming $H_2(8)$

Tipo de código:	$H_2(8)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	8 bits	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	12 bits		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	1 bit		0.14, 0.16, 0.2, 0.3,
Cardinalidad (M):	256		0.4 y 0.5.
Tasa de transmisión:	0.6667	**Tasa de corrección:**	0.08333

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	12288	13.1	0.0	0.0010	0.0	8192	8.3	0.0	0.0010	0.0
0.005	12288	65.5	3.4	0.0053	0.0002	8192	42.1	2.3	0.0051	0.0002
0.01	12288	121.4	19.9	0.0098	0.0016	8192	81.4	13.5	0.0099	0.0016
0.02	12288	250.0	67.0	0.0203	0.0054	8192	166.1	45.9	0.0202	0.0056
0.04	12288	483.5	234.0	0.0393	0.0190	8192	321.5	155.2	0.0392	0.0189
0.06	12288	745.2	492.7	0.0606	0.0400	8192	495.6	325.6	0.0604	0.0397
0.08	12288	973.7	765.4	0.0792	0.0622	8192	650.8	507.4	0.0794	0.0619
0.1	12288	1234.3	1083.8	0.1004	0.0881	8192	830.9	721.6	0.1014	0.0880
0.14	12288	1727.3	1738.0	0.1405	0.1414	8192	1147.4	1146.8	0.1400	0.1399
0.16	12288	1968.5	2065.4	0.1601	0.1680	8192	1315.4	1369.9	0.1605	0.1672
0.2	12288	2460.3	2655.0	0.2002	0.2160	8192	1640.0	1766.0	0.2001	0.2155
0.3	12288	3706.7	3957.9	0.3016	0.3220	8192	2473.2	2649.1	0.3019	0.3233
0.4	12288	4933.2	5093.8	0.4014	0.4145	8192	3284.4	3391.8	0.4009	0.4140
0.5	12288	6142.4	6141.7	0.4998	0.4998	8192	4087.1	4092.3	0.4989	0.4995

Tabla 8.2.2.2. Resultados de los experimentos realizados con el código de Hamming $H_2(8)$ sobre un canal simétrico binario.

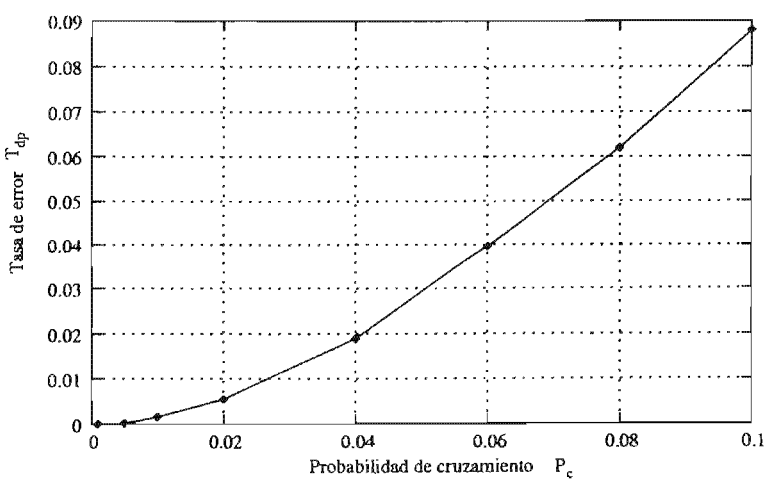


Figura 8.5: Comportamiento del código de Hamming $H_2(8)$.

Código de Hamming $H_2(12)$

Tipo de código:	$H_2(12)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	12 bits	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	17 bits		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	1 bit		0.14, 0.16, 0.2, 0.3,
Cardinalidad (M):	4096		0.4 y 0.5.
Tasa de transmisión:	0.7058	**Tasa de corrección:**	0.0588

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	11628	13.3	0.0	0.0011	0.0	8208	8.9	0.0	0.0010	0.0
0.005	11628	58.7	5.8	0.0050	0.0004	8208	38.8	4.3	0.0047	0.0005
0.01	11628	117.7	27.1	0.0101	0.0023	8208	80.9	18.7	0.0098	0.0022
0.02	11628	226.8	82.3	0.0195	0.0070	8208	159.3	59.2	0.0194	0.0072
0.04	11628	461.4	298.4	0.0396	0.0256	8208	327.3	213.5	0.0398	0.0260
0.06	11628	699.3	584.0	0.0601	0.0502	8208	494.9	413.9	0.0602	0.0504
0.08	11628	940.1	886.0	0.0808	0.0761	8208	660.7	628.4	0.0804	0.0765
0.1	11628	1161.8	1169.5	0.0999	0.1005	8208	821.4	831.3	0.1000	0.1012
0.14	11628	1639.7	1770.1	0.1410	0.1522	8208	1165.6	1260.5	0.1420	0.1535
0.16	11628	1856.6	2023.9	0.1596	0.1740	8208	1317.3	1440.7	0.1604	0.1755
0.2	11628	2322.1	2512.0	0.1996	0.2160	8208	1634.8	1771.5	0.1991	0.2158
0.3	11628	3501.2	3642.7	0.3011	0.3132	8208	2460.1	2566.0	0.2997	0.3126
0.4	11628	4622.7	4691.0	0.3975	0.4034	8208	3266.6	3315.6	0.3979	0.4039
0.5	11628	5821.3	5823.4	0.5006	0.5008	8208	4123.3	4123.7	0.5023	0.5024

Tabla 8.2.2.3. Resultados de los experimentos realizados con el código de Hamming $H_2(12)$ sobre un canal simétrico binario.

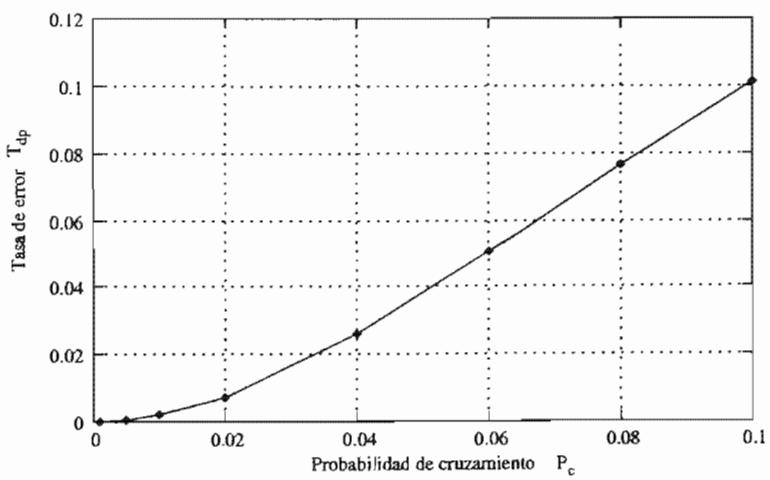


Figura 8.6: Comportamiento del código de Hamming $H_2(12)$.

Comentarios sobre los códigos de Hamming

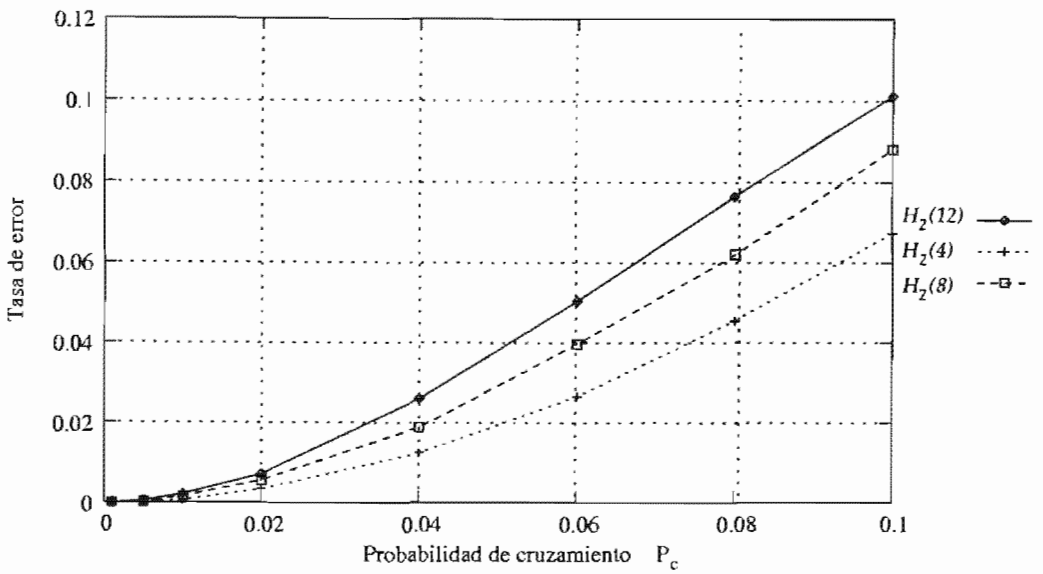


Figura 8.7: Comportamiento de los códigos de Hamming.

Como se puede ver en la Figura 8.7, el código de Hamming $H_2(4)$ tiene el mejor desempeño. Sin embargo, también tiene la menor capacidad de transmisión.

Obsérvese que cuando $P_c = 0.1$, el código de Hamming $H_2(4)$ tiene una pérdida del archivo de 0.2767 y que el código de Hamming $H_2(12)$ tiene una pérdida del archivo de 0.4733. Dada esta situación, valdría la pena disminuir la capacidad de transmisión para obtener una mayor recuperación de los errores.

8.3.3. Códigos de Reed-Muller

El código de Reed-Muller es un código más complejo que el código de Hamming o el código de Golay. Sin embargo, ofrece una mayor capacidad de elección en lo que se refiere a la longitud del mensaje y a la capacidad de corrección de bits, aunque esto ocasione que disminuya la tasa de transmisión. Como el lector recordará, para construir un código de Reed-Muller, se requieren dos parámetros r y m. En este caso se trata de un $\mathcal{RM}(2,4)$ con capacidad de corrección de 1 bit por cada palabra de código de 16 bits.

A continuación presentamos los resultados de la transmisión de estos códigos usando un canal simétrico binario.

Código de Reed-Muller $RM(2,4)$

Tipo de código:	$RM(2,4)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	11 bits	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	16 bits		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	1 bit		0.14, 0.16, 0.2, 0.3,
Cardinalidad (M):	2048		0.4 y 0.5.
Tasa de transmisión:	0.6875	**Tasa de corrección:**	0.0625

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	11920	10.8	0.0	0.0009	0.0	8195	56.1	0.0	0.0068	0.0
0.005	11920	63.4	18.2	0.0053	0.0015	8195	331.8	11.3	0.0404	0.0013
0.01	11920	118.6	71.8	0.0099	0.0060	8195	602.0	36.3	0.0734	0.0044
0.02	11920	243.7	224.4	0.0204	0.0188	8195	1177.1	125.2	0.1436	0.0152
0.04	11920	472.3	781.0	0.0396	0.0655	8195	1985.2	429.7	0.2422	0.0524
0.06	11920	704.0	1392.8	0.0590	0.1168	8195	2582.6	785.8	0.3151	0.0958
0.08	11920	949.8	2148.8	0.0796	0.1802	8195	2985.0	1196.9	0.3642	0.1460
0.1	11920	1199.7	2858.8	0.1006	0.2398	8195	3331.9	1623.8	0.4065	0.1981
0.14	11920	1648.2	3992.6	0.1382	0.3349	8195	3740.4	2343.7	0.4564	0.2859
0.16	11920	1913.5	4419.4	0.1605	0.3707	8195	3838.8	2604.8	0.4684	0.3178
0.2	11920	2380.0	5112.2	0.1996	0.4288	8195	3933.1	3097.9	0.4799	0.3780
0.3	11920	3579.5	5812.6	0.3002	0.4876	8195	4100.0	3763.8	0.5003	0.4592
0.4	11920	4785.6	5934.0	0.4014	0.4978	8195	4093.4	4026.1	0.4994	0.4912
0.5	11920	5952.9	5946.4	0.4994	0.4988	8195	4070.3	4090.9	0.4966	0.4991

Tabla 8.2.3.1. Experimentos usando el código Reed-Muller $RM(2,4)$ en el canal simétrico binario.

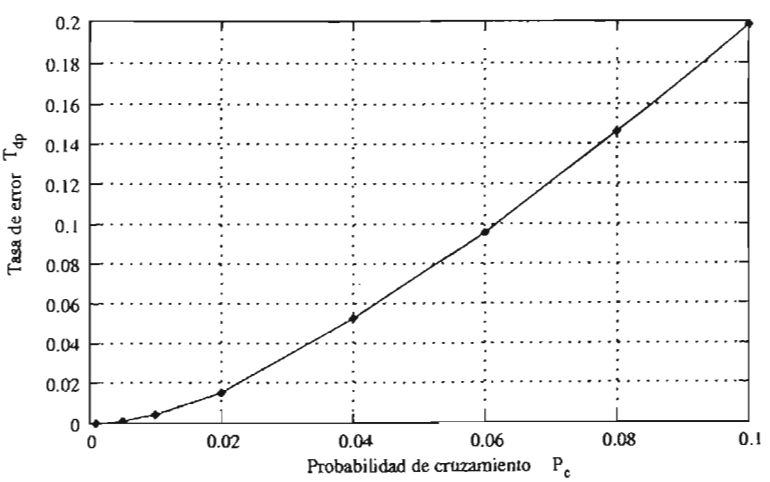


Figura 8.8: Comportamiento del código de Reed-Muller $RM(2,4)$.

Código de Reed-Muller $RM(1,3)$

Tipo de código:	$RM(1,3)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	8 bits	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	4 bits		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	1 bit		0.14, 0.16, 0.2, 0.3,
Cardinalidad (M):	16		0.4 y 0.5.
Tasa de transmisión:	0.5	**Tasa de corrección:**	0.125

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	16384	15.5	0.0	0.0009	0.0	8192	31.5	0.0	0.0038	0.0
0.005	16384	77.6	7.2	0.0047	0.0004	8192	153.4	3.7	0.0187	0.0004
0.01	16384	176.9	17.2	0.0107	0.0010	8192	346.2	9.3	0.0422	0.0011
0.02	16384	325.9	74.0	0.0198	0.0045	8192	621.3	39.7	0.0758	0.0048
0.04	16384	655.3	269.2	0.0399	0.0164	8192	1158.2	140.2	0.1413	0.0171
0.06	16384	989.2	549.6	0.0603	0.0335	8192	1617.0	289.0	0.1973	0.0352
0.08	16384	1310.5	890.4	0.0799	0.0543	8192	2001.6	472.2	0.2443	0.0576
0.1	16384	1646.0	1312.0	0.1004	0.0800	8192	2373.8	685.8	0.2897	0.0837
0.14	16384	2308.2	2271.6	0.1408	0.1386	8192	2879.2	1203.1	0.3514	0.1468
0.16	16384	2608.8	2683.2	0.1592	0.1637	8192	3035.2	1412.6	0.3705	0.1724
0.2	16384	3276.3	3657.2	0.1999	0.2232	8192	3425.9	1973.7	0.4182	0.2409
0.3	16384	4910.7	5614.4	0.2997	0.3426	8192	3877.4	3034.8	0.4733	0.3704
0.4	16384	6565.3	7022.0	0.4007	0.4285	8192	4054.8	3769.3	0.4949	0.4601
0.5	16384	8215.2	7798.0	0.5014	0.4759	8192	4117.8	4121.3	0.5026	0.5030

Tabla 8.2.3.2. Experimentos usando el código Reed-Muller $RM(1,3)$ sobre el canal simétrico binario.

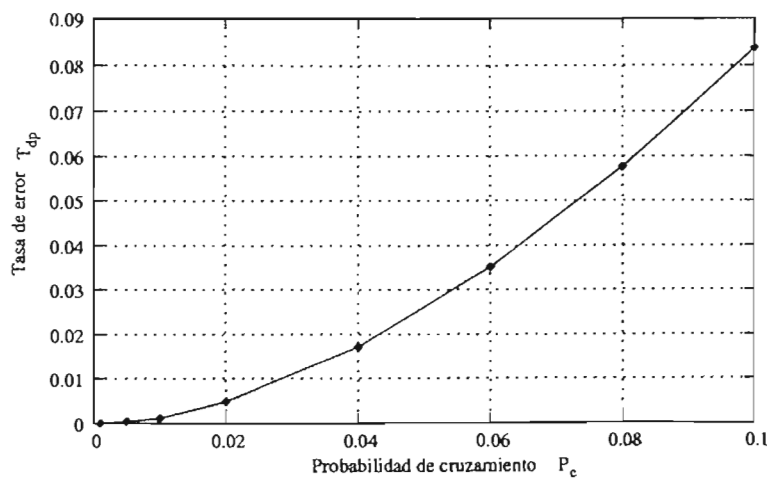


Figura 8.9: Comportamiento del código de Reed-Muller $RM(1,3)$.

Código de Reed-Muller $RM(1,4)$

Tipo de código:	$RM(1,4)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	5 bits	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	16 bits		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	3 bits		0.14, 0.16, 0.2, 0.3,
Cardinalidad (M):	32		0.4 y 0.5.
Tasa de transmisión:	0.3125	**Tasa de corrección:**	0.1875

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	26224	24.6	0.0	0.0009	0.0	8195	59.2	0.0	0.0072	0.0
0.005	26224	127.4	0.0	0.0048	0.0	8195	302.4	0.0	0.0369	0.0
0.01	26224	261.8	0.8	0.0099	0.0003	8195	602.3	0.4	0.0734	0.0004
0.02	26224	529.6	2.4	0.0201	0.0009	8195	1151.4	0.7	0.1405	0.0008
0.04	26224	1039.4	32.8	0.0396	0.0012	8195	1907.6	8.1	0.2327	0.0009
0.06	26224	1600.2	161.6	0.0610	0.0061	8195	2610.3	43.6	0.3185	0.0053
0.08	26224	2109.9	328.8	0.0804	0.0125	8195	3043.5	89.6	0.3713	0.0109
0.1	26224	2614.7	735.2	0.0997	0.0280	8195	3309.2	194.9	0.4038	0.0237
0.14	26224	3703.2	1960.8	0.1412	0.0747	8195	3721.4	544.2	0.4541	0.0664
0.16	26224	4182.6	2716.0	0.1594	0.1035	8195	3823.1	761.3	0.4665	0.0928
0.2	26224	5215.6	4472.0	0.1988	0.1705	8195	3983.5	1271.1	0.4860	0.1551
0.3	26224	7869.2	8919.2	0.3000	0.3401	8195	4088.0	2665.5	0.4988	0.3252
0.4	26224	10468	11629.	0.3991	0.4434	8195	4079.1	3740.6	0.4977	0.4564
0.5	26224	13125	12796.	0.5004	0.4879	8195	4102.5	4112.5	0.5006	0.5018

Tabla 8.2.3.3. Experimentos usando el código Reed-Muller $RM(1,4)$ sobre el canal simétrico binario.

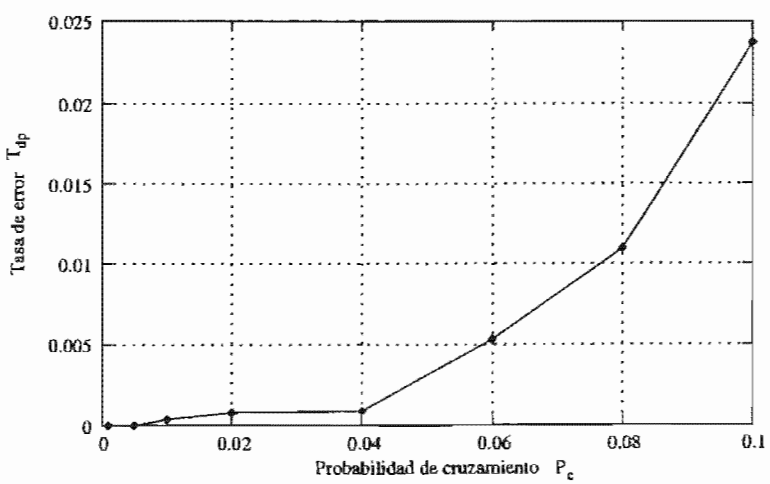


Figura 8.10: Comportamiento del código de Reed-Muller $RM(1,4)$.

Comentarios sobre los códigos Reed-Muller

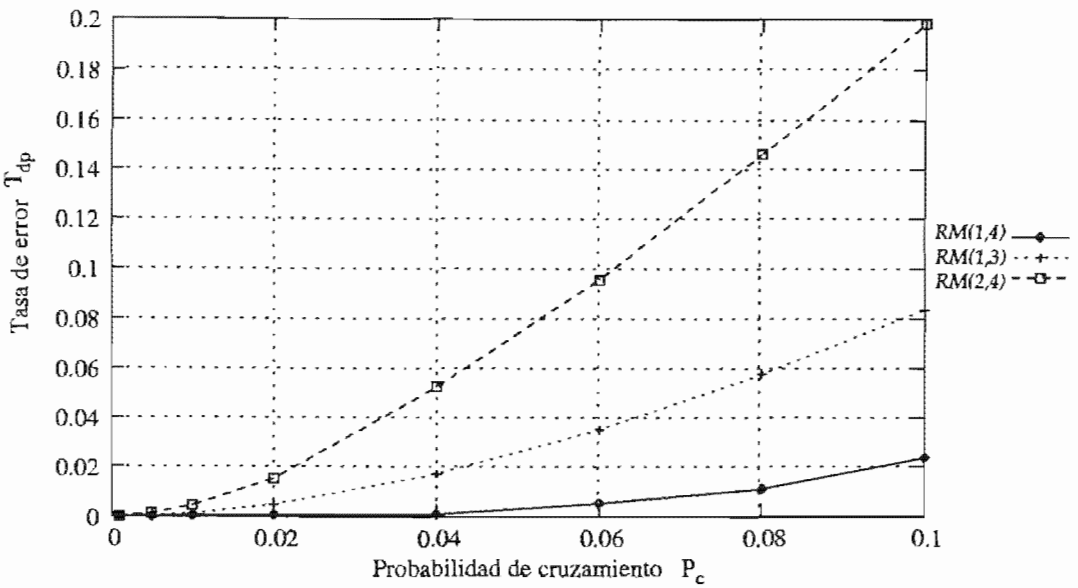


Figura 8.11: Comportamiento de los códigos de Reed-Muller.

Aun cuando la mayor recuperación de errores se tiene en el código $RM(1,4)$, este código tiene una tasa de transmisión de 0.3125. De tal forma que más de la mitad de los bits que se transmiten por el canal, son bits agregados para la recuperación de errores. Sin embargo, durante la transmisión de fotografías en 1969 y 1977 hechas por el Mariner, se ocupó el código de Reed-Muller $RM(1,5)$, el cual tiene una tasa de transmisión de 0.1875, pero tiene una tasa de corrección de 0.21875.

8.3.4. Códigos de Reed-Solomon

En esta sección presentamos los resultados de la simulación de la transmisión de bits usando un canal simétrico binario con los códigos Reed-Solomon $RS(2^3,3)$, $RS(2^4,9)$ y $RS(2^5,17)$.

Recuérdese que los códigos Reed-Solomon no son binarios, por lo tanto llevar a cabo esta simulación implica más tiempo que los códigos anteriores. Aquí intervienen dos factores:

- La traducción constante que se realiza entre el campo de Galois y el alfabeto binario, y
- La dificultad del proceso de decodificación.

Por último deseamos recordar al lector que en principio los códigos Reed-Solomon se pensaban para usarse con ruido con ráfagas y no con ruido blanco, el cual es el manejado en esta sección.

Código de Reed-Solomon $RS(2^3, 3)$

Tipo de código:	$RS(2^3, 3)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	5 símbolos	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	7 símbolos		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	1 símbolo		0.14, 0.16, 0.2,0.3,
Longitud del símbolo:	3 bits		0.4 y 0.5.
Cardinalidad (M):	32768		
Tasa de transmisión:	0.714285	**Tasa de corrección:**	0.0476 - 0.14285

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	11529	242.6	206.5	0.0210	0.0179	8235	453.7	181.3	0.0550	0.0220
0.003	11529	270.8	225.2	0.0234	0.0195	8235	508.6	192.3	0.0617	0.0233
0.005	11529	297.3	240.9	0.0257	0.0208	8235	620.5	222.3	0.0753	0.0269
0.008	11529	321.6	254.0	0.0278	0.0220	8235	681.3	251.5	0.0827	0.0305
0.01	11529	349.7	281.7	0.0303	0.0244	8235	745.7	280.4	0.0905	0.0340
0.02	11529	461.9	380.2	0.0400	0.0329	8235	1096.5	435.9	0.1331	0.0529
0.03	11529	568.1	495.6	0.0492	0.0429	8235	1339.1	583.0	0.1626	0.0707
0.04	11529	709.0	662.7	0.0614	0.0574	8235	1644.9	830.2	0.1997	0.1008
0.05	11529	792.9	765.9	0.0687	0.0664	8235	1840.1	981.6	0.2234	0.1191
0.08	11529	1139.0	1237.7	0.0987	0.1073	8235	2344.2	1599.9	0.2846	0.1942
0.1	11529	1359.9	1526.1	0.1179	0.1323	8235	2676.9	1975.1	0.3250	0.2398
0.2	11529	2490.1	2797.2	0.2159	0.2426	8235	3431.1	3100.1	0.4166	0.3764
0.3	11529	3629.3	3855.3	0.3147	0.3344	8235	3770.5	3623.3	0.4578	0.4399
0.4	11529	4742.6	4856.4	0.4113	0.4212	8235	3979.7	3906.6	0.4832	0.4743
0.5	11529	5866.0	5859.3	0.5088	0.5082	8235	4148.6	4142.1	0.5037	0.5029

Tabla 8.2.4.1. Experimentos usando el código Reed-Solomon $RS(2^3, 3)$ sobre el canal simétrico binario.

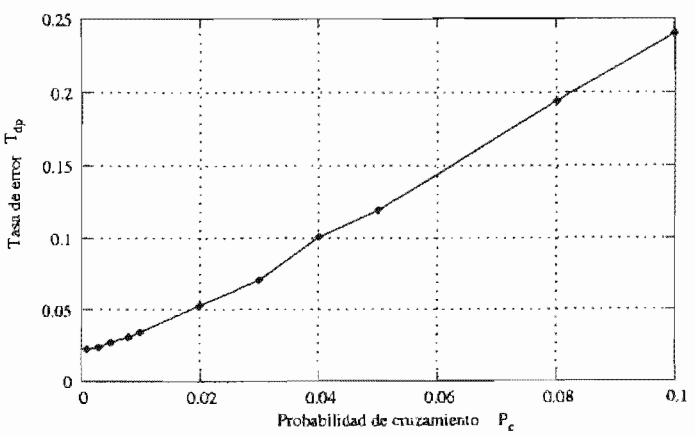


Figura 8.12: Comportamiento del código de Reed-Solomon $RS(2^3, 3)$.

Código de Reed-Solomon $RS(2^4, 9)$

Tipo de código:	$RS(2^4, 9)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	7 símbolos	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	15 símbolos		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	4 símbolos		0.14, 0.16, 0.2, 0.3,
Longitud del símbolo:	4 bits		0.4 y 0.5.
Cardinalidad (M):	268435456		
Tasa de transmisión:	0.46666	**Tasa de corrección:**	0.066 - 0.266

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	17580	17.8	0.9	0.0010	0.00005	8204	51.4	0.0	0.0062	0.0
0.005	17580	87.8	6.5	0.0049	0.0003	8204	237.6	0.0	0.0289	0.0
0.01	17580	173.7	12.8	0.0098	0.0007	8204	494.4	1.1	0.0602	0.0001
0.02	17580	338.6	27.3	0.0192	0.0015	8204	861.7	11.6	0.1050	0.0014
0.04	17580	701.2	180.8	0.0398	0.0102	8204	1552.3	212.2	0.1892	0.0258
0.06	17580	1066.4	524.3	0.0606	0.0298	8204	2113.6	684.5	0.2576	0.0834
0.08	17580	1400.8	1020.2	0.0796	0.0580	8204	2460.0	1339.4	0.2998	0.1632
0.1	17580	1752.4	1583.9	0.0996	0.0900	8204	2782.9	1981.7	0.3392	0.2415
0.14	17580	2443.5	2603.9	0.1389	0.1481	8204	3162.3	2946.6	0.3854	0.3591
0.16	17580	2808.8	3054.3	0.1597	0.1737	8204	3263.6	3211.8	0.3978	0.3914
0.2	17580	3528.3	3815.5	0.2006	0.2170	8204	3494.6	3510.9	0.4259	0.4279
0.3	17580	5291.5	5500.4	0.3009	0.3128	8204	3802.3	3818.7	0.4634	0.4654
0.4	17580	7046.2	7142.5	0.4008	0.4062	8204	3977.6	3981.9	0.4848	0.4853
0.5	17580	8804.7	8793.6	0.5008	0.5002	8204	4123.9	4111.7	0.5026	0.5011

Tabla 8.2.4.2. Experimentos usando el código Reed-Solomon $RS(2^4, 9)$ sobre el canal simétrico binario.

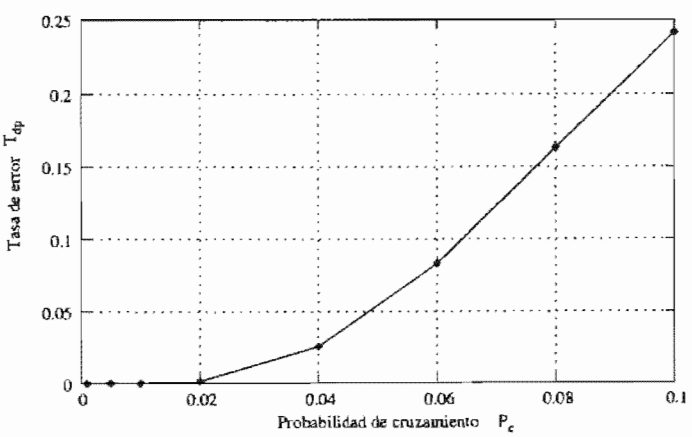


Figura 8.13: Comportamiento del código de Reed-Solomon $RS(2^4, 9)$.

Código de Reed-Solomon $RS(2^5, 17)$

Tipo de código:	$RS(2^5, 17)$	**Tipo de canal:**	Canal simétrico binario.
Longitud del mensaje:	15 símbolos	Probabilidad (P_c):	0.001, 0.005, 0.01, 0.02,
Longitud de la palabra de código:	31 símbolos		0.04, 0.06, 0.08, 0.1,
Capacidad de corrección por palabra:	8 símbolos		0.14, 0.16, 0.2, 0.3,
Longitud del símbolo:	5 bits		0.4 y 0.5.
Cardinalidad (M):	3.77×10^{22}		
Tasa de transmisión:	0.4838	**Tasa de corrección:**	0.05161 - 0.2580

P_c	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.001	17360	18.3	0.3	0.0010	0.00001	8400	174.3	0.0	0.0207	0.0
0.005	17360	85.6	3.9	0.0049	0.0002	8400	746.9	0.0	0.0889	0.0
0.01	17360	173.0	4.9	0.0099	0.002	8400	1356.9	0.0	0.1615	0.0
0.02	17360	349.6	14.4	0.0201	0.008	8400	2118.7	12.1	0.2522	0.0014
0.04	17360	697.4	179.7	0.0401	0.0103	8400	3012.8	410.6	0.3586	0.0488
0.06	17360	1045.9	711.7	0.0602	0.0409	8400	3429.8	1717.5	0.4083	0.2044
0.08	17360	1391.5	1364.8	0.0801	0.0786	8400	3566.7	2870.1	0.4246	0.3416
0.1	17360	1719.1	1831.4	0.0990	0.1054	8400	3750.6	3528.6	0.4465	0.4200
0.14	17360	2430.7	2602.0	0.1400	0.1498	8400	3908.1	3908.5	0.4652	0.4652
0.16	17360	2747.9	2921.2	0.1582	0.1682	8400	3933.4	3942.7	0.4682	0.4693
0.2	17360	3488.4	3651.3	0.2009	0.2103	8400	3999.8	4019.7	0.4761	0.4785
0.3	17360	5208.7	5306.6	0.3000	0.3056	8400	4095.4	4099.0	0.4875	0.4879
0.4	17360	6947.0	6997.9	0.4001	0.4031	8400	4145.3	4133.9	0.4934	0.4921
0.5	17360	8654.8	8662.3	0.4985	0.4989	8400	4196.8	4207.5	0.4996	0.5008

Tabla 8.2.4.3. Experimentos usando el código Reed-Solomon $RS(2^5, 17)$ sobre el canal simétrico binario.

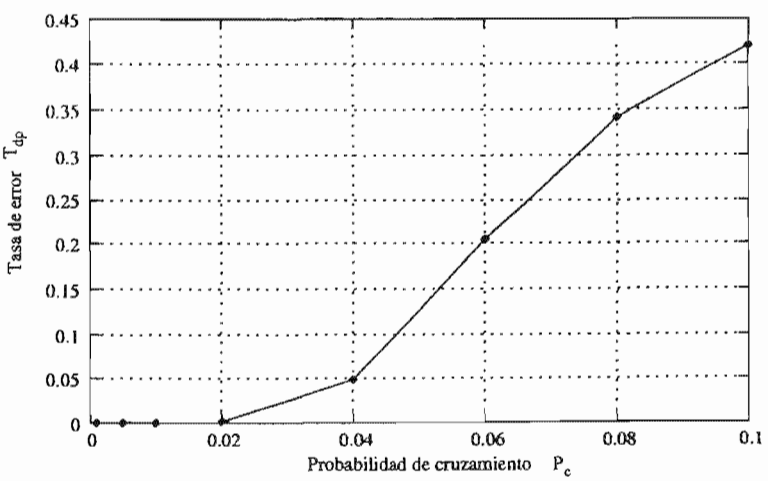


Figura 8.14: Comportamiento del código de Reed-Solomon $RS(2^5, 17)$.

Comentarios sobre los códigos de Reed-Solomon

Los códigos de Reed-Solomon son especiales, pues como no son binarios, es más difícil establecer su capacidad de corrección con objetividad en términos de bits. Como ya vimos en el capítulo anterior, estos códigos corrigen a nivel de símbolos, no de bits. De tal forma que si un símbolo que se compone de 5 bits, sólo presenta un 1 bit alterado, para los términos de corrección de Reed-Solomon, es exactamente igual a que este símbolo presente los 5 bits alterados.

Aun cuando esto pueda ser una ventaja, cuando trabajamos con canales donde los errores se distribuyen a lo largo de la palabra de código, entonces es muy fácil rebasar la capacidad de corrección de este código.

Por lo tanto, si se tiene un canal donde el comportamiento promedio consiste en tener muchos errores *distribuidos* a lo largo de la transmisión entonces el código de Reed-Solomon *no* es una buena opción.

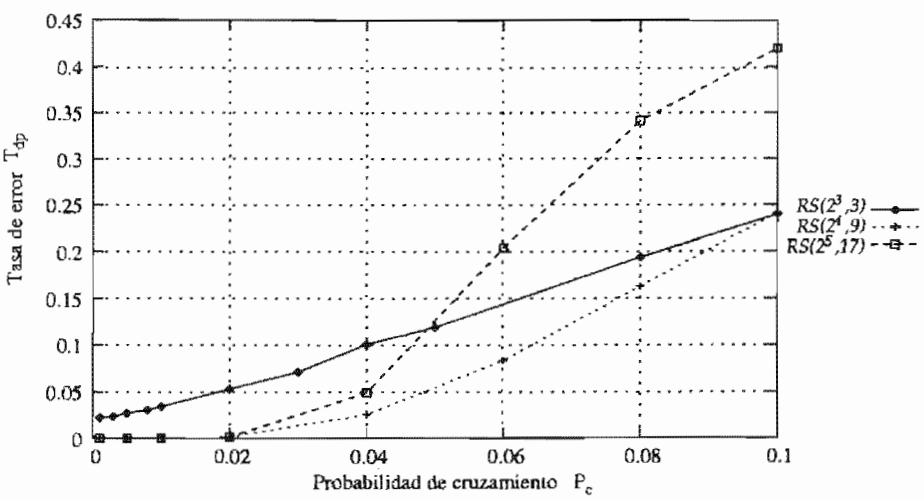


Figura 8.15: Comportamiento de los códigos de Reed-Solomon.

8.3.5. Comportamiento de los códigos en el canal simétrico binario

El canal simétrico binario se caracteriza porque los errores están distribuidos a lo largo de la transmisión. Como se puede observar en la Figura 8.16, el código que presentó una mejor respuesta es el código de $RM(1,4)$. Sin embargo, cabe señalar que el código de Golay G_{24} también tiene una alta recuperación de los errores y una mayor capacidad de transmisión que el código de Reed-Muller.

Obsérvese que el código $H_2(4)$ tiene una mayor capacidad de corrección que el código de Golay G_{24}; entonces ¿por qué el código de Golay tiene un mejor desempeño? Este comportamiento lo entendemos mejor observando la transmisión durante las primeras probabilidades de cruzamiento que se usaron en los experimentos. Por ejemplo, si se presenta la siguiente situación:

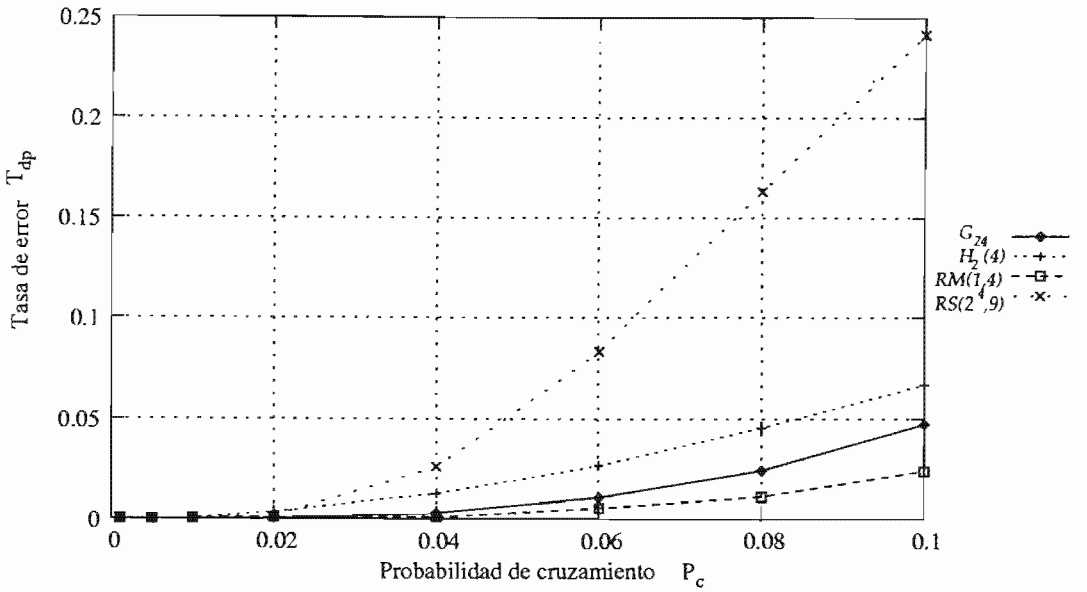


Figura 8.16: Comportamiento de los códigos en un canal simétrico binario.

110011001101010001010100	palabra enviada
100010001101010001010100	palabra recibida

Hay dos bits erróneos, en la segunda posición y en la sexta posición. Bajo estas circunstancias, el código $H_2(4)$ *no* puede recuperar la palabra de código original, porque solamente puede corregir 1 bit por cada 7 bits y en este caso hay dos bits erróneos en los primeros 7 bits. Por otro lado, el código de Golay G_{24} puede corregir los dos bits erróneos en la palabra recibida porque este código corrige 3 bits erróneos por cada 24 bits, sin importar si los bits erróneos estan juntos o distribuidos a lo largo de los 24 bits.

Debido a este tipo de comportamiento, el código de Golay tiene una mayor recuperación de los bits de información que el código de Hamming $H_2(4)$.

Por último, es el código de Reed-Solomon $RS(2^4, 9)$ el que presentó la menor recuperación de bits de información frente a este tipo de ruido, lo cual, es contrario a lo que habíamos supuesto inicialmente.

8.4. Comportamiento de los códigos usando el canal de Gilbert-Elliot

El modelo del canal de Gilbert-Elliot simula el comportamiento de canales con ráfagas. Se ha observado que es un comportamiento común en transmisiones donde el canal es confiable a

excepción de situaciones extremas que ocasionan un bloque continuo de bits erróneos. Este modelo fué revisado en la Sección 2.3 de este trabajo y seguiremos con la notación que se introdujo en dicha sección.

El lector recordará que se trabaja con cuatro probabilidades de transición y dos probabilidades de cruzamiento, correspondientes al estado bueno y al estado malo. Hemos escogido los siguientes valores en las probabilidades de transición porque se adaptaron mejor al tipo de ruido de ráfagas que deseábamos simular en un canal de Gilbert-Elliot.

$$\begin{aligned} p_{bm} &= 0.01 \qquad p_{bb} = 0.99 \\ p_{mb} &= 0.4 \qquad p_{mm} = 0.6 \end{aligned}$$

Estas probabilidades originan la siguiente matriz de transición:

$$M = \begin{pmatrix} 0.99 & 0.01 \\ 0.4 & 0.6 \end{pmatrix}$$

Considerando que el modelo es una cadena de Markov ergódica, como se mencionó en la Sección 2.3, tenemos el siguiente límite:

$$\lim_{n\to\infty} M^n = \begin{pmatrix} 0.9756 & 0.0243 \\ 0.9756 & 0.0243 \end{pmatrix}$$

De donde obtenemos que las probabilidades de estar en el estado bueno y en el estado malo son:

$$\begin{aligned} \Pi_b &= 0.9756 \\ \Pi_m &= 0.0243 \end{aligned}$$

Considerando los resultados anteriores podemos obtener la probabilidad de error a través de la siguiente expresión:

$$P_e = (0.9756)(P_{c_1}) + (0.0243)(P_{c_2})$$

En los experimentos manejaremos los siguientes valores para las probabilidades de cruzamiento:

$$P_{c_1} = 0.001, 0.005, 0.01, 0.05, 0.1,$$

$$P_{c_2} = 0.1, 0.5 \text{ y } 0.9$$

Por cada pareja de valores que tomen P_{c_1} y P_{c_2} simulamos la transmisión de un archivo por el canal de Gilbert-Elliot, diez veces. El promedio de los experimentos se presenta en un renglón de la tabla correspondiente al tipo de código que se usa para llevar a cabo el proceso de codificación y decodificación.

8.4.1. Códigos de Golay

Tipo de código:	Golay	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	12 bits	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	24 bits		0.05 y 0.1
Capacidad de corrección por palabra:	3 bits	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	4096		
Tasa de transmisión:	0.5	**Tasa de corrección:**	0.125

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	16392	54.5	0.0	0.0033	0.0	8196	25.7	0.0	0.0031	0.0
0.007308	0.005	0.1	16392	111.6	0.0	0.0068	0.0	8196	53.6	0.0	0.0065	0.0
0.012186	0.01	0.1	16392	197.4	0.4	0.012	0.0002	8196	95.6	0.1	0.0116	0.0001
0.0131256	0.001	0.5	16392	207.4	56.2	0.0126	0.0034	8196	93.0	23.6	0.0113	0.0028
0.017028	0.005	0.5	16392	275.9	56.0	0.0168	0.0034	8196	132.0	27.0	0.0161	0.0032
0.0219060	0.01	0.5	16392	340.6	53.4	0.0207	0.0032	8196	175.7	28.5	0.0214	0.0034
0.0228456	0.001	0.9	16392	335.5	169.8	0.0204	0.0103	8196	160.5	83.5	0.0195	0.0101
0.026748	0.005	0.9	16392	414.0	189.0	0.0252	0.0115	8196	197.9	87.5	0.0241	0.0106
0.031626	0.01	0.9	16392	505.9	222.0	0.0308	0.0135	8196	243.5	107.7	0.0297	0.0131
0.0512100	0.05	0.1	16392	833.6	103.0	0.0508	0.0062	8196	416.2	52.2	0.0507	0.0063
0.0609300	0.05	0.5	16392	996.4	242.2	0.0607	0.0147	8196	495.8	116.0	0.0604	0.0141
0.07065	0.05	0.9	16392	1129.1	432.0	0.0688	0.0263	8196	561.4	217.8	0.0684	0.0265
0.0999900	0.1	0.1	16392	1626.5	792.4	0.0992	0.0483	8196	811.4	401.6	0.0989	0.0489
0.10971	0.1	0.5	16392	1794.5	1007.8	0.1094	0.0614	8196	891.7	503.2	0.1087	0.0613
0.1194300	0.1	0.9	16392	1938.8	1237.6	0.1182	0.0755	8196	953.6	606.3	0.1163	0.0739

Tabla 8.3.1.1. Experimentos realizados usando el código de Golay con un canal de Gilbert-Elliot.

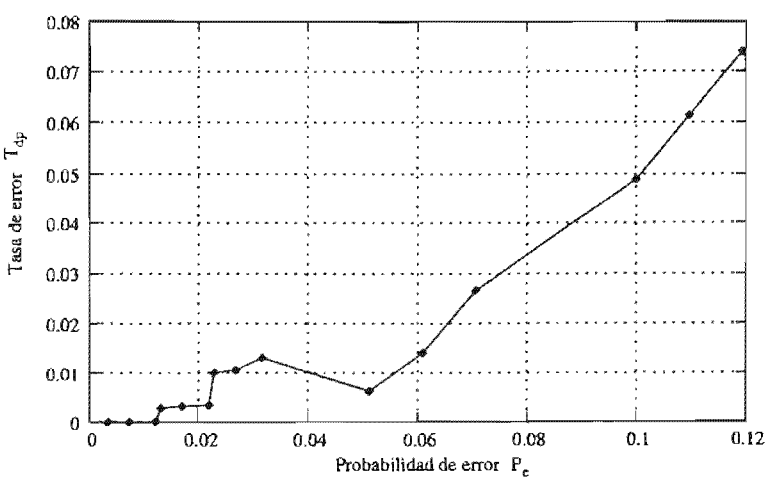


Figura 8.17: Comportamiento del código de Golay.

Comentarios sobre el código de Golay G_{24}

Aun cuando el código de Golay G_{24} no presenta un método de codificación enfocado en ráfagas, presenta un comportamiento favorable con las probabilidades más bajas de error que aquí manejamos. Sin embargo, a medida que P_{c_2} aumenta, tenemos una pérdida acelerada de la recuperación de los bits.

8.4.2. Códigos de Hamming

Los códigos de Hamming solo pueden corregir un error por palabra de código. En este caso, vamos a usar los códigos de Hamming $H_2(4)$, $H_2(8)$ y $H_2(12)$ para simular la transmisión de los bits usando un canal de Gilbert-Elliot.

Código de Hamming $H_2(4)$

Tipo de código:	$H_2(4)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	4 bits	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	7 bits		0.05 y 0.1
Capacidad de corrección por palabra:	1 bit	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	16		
Tasa de transmisión:	0.5714	**Tasa de corrección:**	0.1428

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	14336	45.9	8.4	0.0032	0.0005	8192	27.9	4.6	0.0034	0.0005
0.007308	0.005	0.1	14336	103.0	11.4	0.0071	0.0007	8192	60.6	5.9	0.0073	0.0007
0.012186	0.01	0.1	14336	163.0	19.3	0.0113	0.0013	8192	92.9	11.3	0.0113	0.0013
0.0131256	0.001	0.5	14336	149.2	104.1	0.0104	0.0072	8192	98.6	62.0	0.012	0.0075
0.017028	0.005	0.5	14336	209.8	110.5	0.0146	0.0077	8192	127.7	64.7	0.0155	0.0078
0.0219060	0.01	0.5	14336	281.1	125.8	0.0196	0.0087	8192	171.3	76.1	0.0209	0.0092
0.0228456	0.001	0.9	14336	262.4	229.1	0.0183	0.0159	8192	170.8	141.1	0.0208	0.0172
0.026748	0.005	0.9	14336	324.3	241.2	0.0226	0.0168	8192	203.4	144.8	0.0248	0.0176
0.031626	0.01	0.9	14336	398.3	258.8	0.0277	0.0180	8192	245.3	156.9	0.0299	0.0191
0.0512100	0.05	0.1	14336	725.0	288.2	0.0505	0.0201	8192	415.8	165.9	0.0507	0.0202
0.0609300	0.05	0.5	14336	865.8	440.7	0.0603	0.0307	8192	506.7	253.0	0.0618	0.0308
0.07065	0.05	0.9	14336	954.0	534.3	0.0665	0.0372	8192	565.3	313.2	0.0690	0.0382
0.0999900	0.1	0.1	14336	1416.4	925.1	0.0988	0.0645	8192	818.4	530.3	0.0999	0.0647
0.10971	0.1	0.5	14336	1541.2	1091.2	0.1075	0.0761	8192	897.0	625.2	0.1094	0.0763
0.1194300	0.1	0.9	14336	1673.7	1275.3	0.1167	0.0889	8192	976.8	739.7	0.1192	0.0902

Tabla 8.3.2.1. Experimentos realizados usando el código Hamming $H_2(4)$ con un canal de Gilbert-Elliot.

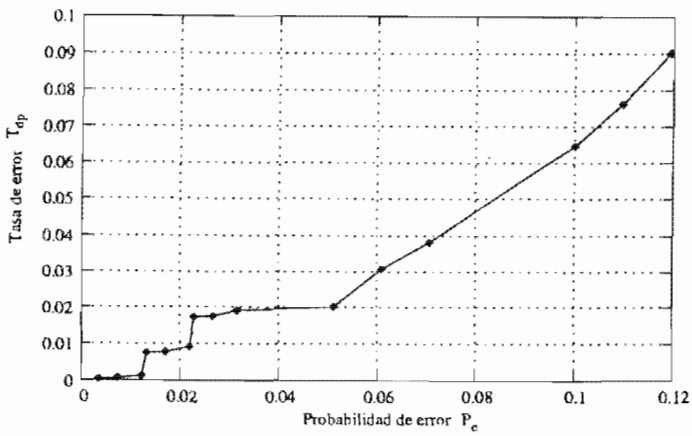


Figura 8.18: Comportamiento del código de Hamming $H_2(4)$.

Código de Hamming $H_2(8)$

Tipo de código:	$H_2(8)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	8 bits	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	12 bits		0.05 y 0.1
Capacidad de corrección por palabra:	1 bit	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	256		
Tasa de transmisión:	0.6667	**Tasa de corrección:**	0.0833

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	12312	40.4	8.2	0.0032	0.0006	8208	28.2	5.5	0.0034	0.0006
0.007308	0.005	0.1	12312	82.1	16.0	0.0066	0.0012	8208	55.9	9.9	0.0068	0.0012
0.012186	0.01	0.1	12312	146.5	25.2	0.0118	0.0020	8208	100.0	17.0	0.0121	0.0020
0.0131256	0.001	0.5	12312	141.9	103.9	0.0115	0.0084	8208	98.4	69.6	0.0119	0.0084
0.017028	0.005	0.5	12312	199.9	126.0	0.0162	0.0102	8208	139.3	87.4	0.0169	0.0106
0.0219060	0.01	0.5	12312	253.3	136.4	0.0205	0.011	8208	178.5	95.4	0.0217	0.0116
0.0228456	0.001	0.9	12312	244.0	218.6	0.0198	0.0177	8208	179.4	154.5	0.0218	0.0188
0.026748	0.005	0.9	12312	295.5	231.6	0.0240	0.0188	8208	208.4	159.2	0.0253	0.0193
0.031626	0.01	0.9	12312	356.7	251.7	0.0289	0.0204	8208	255.4	175.5	0.0311	0.0213
0.0512100	0.05	0.1	12312	626.2	356.5	0.0508	0.0289	8208	415.8	240.6	0.0506	0.0293
0.0609300	0.05	0.5	12312	729.7	495.7	0.0592	0.0402	8208	490.7	333.0	0.0597	0.0405
0.07065	0.05	0.9	12312	855.7	654.0	0.0695	0.0531	8208	580.1	441.5	0.0706	0.0537
0.0999900	0.1	0.1	12312	1240.2	1109.5	0.1007	0.0901	8208	819.4	736.4	0.0998	0.0897
0.10971	0.1	0.5	12312	1318.1	1203.2	0.107	0.0977	8208	880.4	801.2	0.1072	0.0976
0.1194300	0.1	0.9	12312	1440.5	1331.4	0.1169	0.1081	8208	974.1	898.9	0.1186	0.1095

Tabla 8.3.2.2. Experimentos realizados usando el código Hamming $H_2(8)$ con un canal de Gilbert-Elliot.

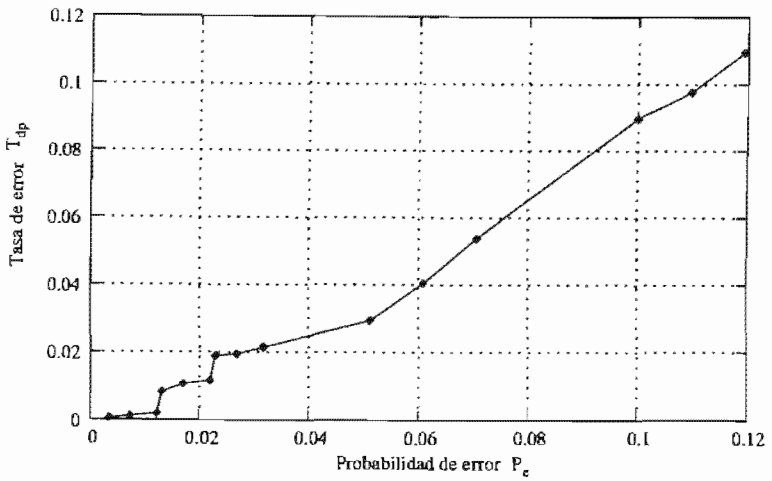


Figura 8.19: Comportamiento del código de Hamming $H_2(8)$.

Código de Hamming $H_2(12)$

Tipo de código:	$H_2(12)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	12 bits	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	17 bits		0.05 y 0.1
Capacidad de corrección por palabra:	1 bit	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	4096		
Tasa de transmisión:	0.7058	**Tasa de corrección:**	0.0588

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	11628	32.3	8.5	0.0027	7.3099	8208	24.4	6.3	0.0029	0.0007
0.007308	0.005	0.1	11628	79.3	15.8	0.0068	0.0013	8208	57.8	11.7	0.0070	0.0014
0.012186	0.01	0.1	11628	133.7	33.1	0.0114	0.0028	8208	95.1	21.1	0.0115	0.0025
0.0131256	0.001	0.5	11628	138.4	106.9	0.0119	0.0091	8208	100.0	75.3	0.0121	0.0091
0.017028	0.005	0.5	11628	188.5	125.0	0.0162	0.0107	8208	138.5	86.8	0.0168	0.0105
0.0219060	0.01	0.5	11628	250.6	161.7	0.0215	0.0139	8208	179.5	114.0	0.0218	0.0138
0.0228456	0.001	0.9	11628	268.0	258.9	0.0230	0.0222	8208	203.1	188.3	0.0247	0.0229
0.026748	0.005	0.9	11628	280.3	234.0	0.0241	0.0201	8208	205.4	167.8	0.0250	0.0204
0.031626	0.01	0.9	11628	350.4	277.9	0.0301	0.0238	8208	252.3	197.0	0.0307	0.0240
0.0512100	0.05	0.1	11628	589.9	436.7	0.0507	0.0375	8208	414.9	310.1	0.0505	0.0377
0.0609300	0.05	0.5	11628	700.3	578.8	0.0602	0.0497	8208	498.0	410.8	0.0606	0.05
0.07065	0.05	0.9	11628	797.1	697.4	0.0685	0.0599	8208	569.1	495.9	0.0693	0.0604
0.0999900	0.1	0.1	11628	1170.6	1197.6	0.1006	0.1029	8208	834.0	857.3	0.1016	0.1044
0.10971	0.1	0.5	11628	1272.8	1318.9	0.1094	0.1134	8208	911.7	949.6	0.111	0.1156
0.1194300	0.1	0.9	11628	1384.2	1437.2	0.1190	0.1235	8208	982.6	1020.4	0.1197	0.1243

Tabla 8.3.2.3. Experimentos realizados usando el código Hamming $H_2(12)$ con un canal de Gilbert-Elliot.

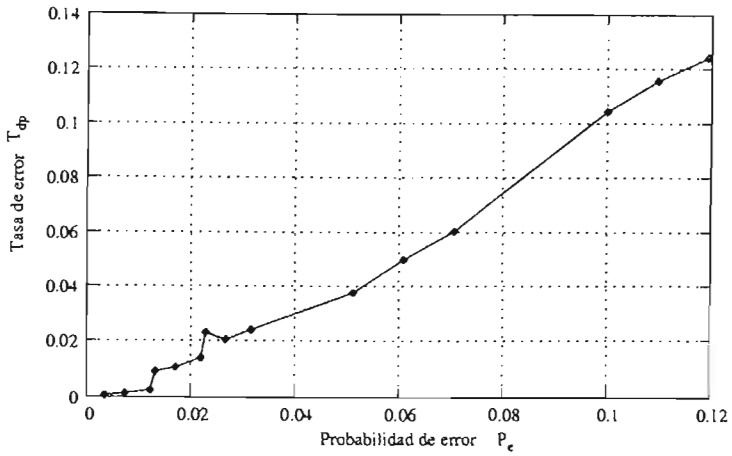


Figura 8.20: Comportamiento del código de Hamming $H_2(12)$.

Comentarios sobre los códigos de Hamming

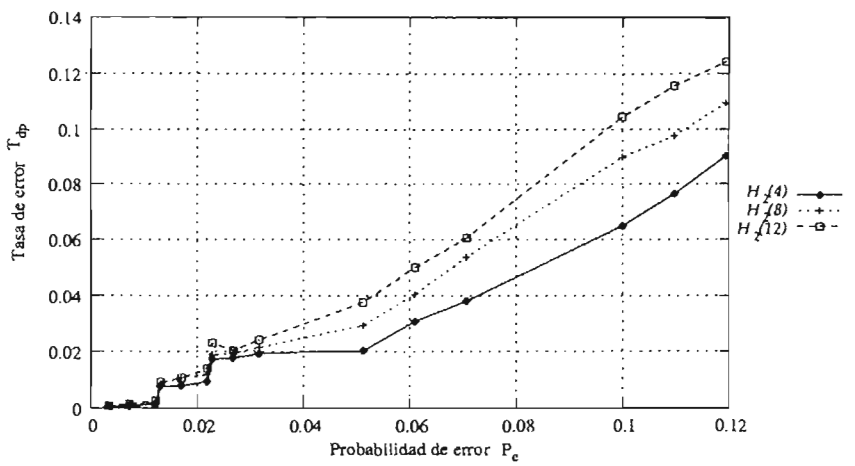


Figura 8.21: Comportamiento de los códigos de Hamming.

En el comportamiendo de los códigos de Hamming frente al ruido con ráfagas, aun cuando su recuperación de errores disminuye comparándolo con el ruído en el canal simétrico binario, considerando su desempeño se mantiene el orden entre los tres códigos de Hamming revisados. Esto es, nuevamente, el código que tienen una mayor recuperación de los errores es $H_2(4)$, seguido por $H_2(8)$ y por último $H_2(12)$.

Obsérvese que ninguno de ellos lleva a cabo un recuperación *completa* de los bits de información en ninguno de los casos.

8.4.3. Códigos de Reed-Muller

Código de Reed-Muller $RM(2,4)$

Tipo de código:	$RM(2,4)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	11 bits	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	16 bits		0.05 y 0.1
Capacidad de corrección por palabra:	1 bit	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	2048		
Tasa de transmisión:	0.6875	**Tasa de corrección:**	0.0625

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	11920	42.9	35.4	0.0035	0.0029	8195	207.9	18.1	0.0253	0.0022
0.007308	0.005	0.1	11920	83.6	43.8	0.0070	0.0036	8195	425.5	23.1	0.0519	0.0028
0.012186	0.01	0.1	11920	146.2	121.8	0.0122	0.0102	8195	699.7	68.2	0.0853	0.0083
0.0131256	0.001	0.5	11920	145.0	263.2	0.0121	0.022	8195	497.1	128.7	0.0606	0.0157
0.017028	0.005	0.5	11920	195.8	311.2	0.0164	0.0261	8195	715.7	148.6	0.0873	0.0181
0.0219060	0.01	0.5	11920	241.5	354.2	0.0202	0.0297	8195	923.5	184.3	0.1126	0.0224
0.0228456	0.001	0.9	11920	252.5	6141.6	0.0211	0.5152	8195	629.2	203.8	0.0767	0.0248
0.026748	0.005	0.9	11920	297.9	5573.4	0.0249	0.4675	8195	833.9	247.8	0.1017	0.0302
0.031626	0.01	0.9	11920	356.4	5919.4	0.0298	0.4965	8195	1112.3	285.0	0.1357	0.0347
0.0512100	0.05	0.1	11920	602.8	1160.8	0.0505	0.0973	8195	2321.1	637.3	0.2832	0.0777
0.0609300	0.05	0.5	11920	701.3	1387.8	0.0588	0.1164	8195	2426.1	787.5	0.296	0.096
0.07065	0.05	0.9	11920	818.7	6035.0	0.0686	0.5062	8195	2543.5	879.6	0.3103	0.1073
0.0999900	0.1	0.1	11920	1187.9	2796.8	0.0996	0.2346	8195	3307.6	1610.4	0.4036	0.1965
0.10971	0.1	0.5	11920	1302.2	3052.6	0.1092	0.256	8195	3388.8	1759.3	0.4135	0.2146
0.1194300	0.1	0.9	11920	1398.5	5953.2	0.1173	0.4994	8195	3452.2	1799.4	0.4212	0.2195

Tabla 8.3.3.1. Experimentos realizados usando el código $RM(2,4)$ con un canal de Gilbert-Elliot.

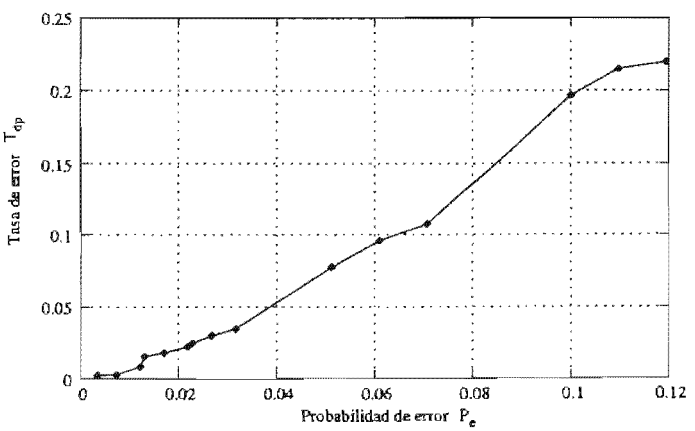


Figura 8.22: Comportamiento del código de Reed-Muller $RM(2,4)$.

Código de Reed-Muller $RM(1,3)$

Tipo de código:	$RM(1,3)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	8 bits	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01, 0.05 y 0.1
Longitud de la palabra de código:	4 bits		
Capacidad de corrección por palabra:	1 bit	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	16		
Tasa de transmisión:	0.5	**Tasa de corrección:**	0.125

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	16384	50.2	10.8	0.0030	0.0006	8192	94.9	5.9	0.0115	0.0007
0.007308	0.005	0.1	16384	112.3	14.0	0.0068	0.0008	8192	219.0	6.8	0.0267	0.0008
0.012186	0.01	0.1	16384	188.3	27.2	0.0114	0.0016	8192	364.3	14.6	0.0444	0.0017
0.0131256	0.001	0.5	16384	176.5	128.0	0.0107	0.0078	8192	236.1	63.2	0.0288	0.0077
0.017028	0.005	0.5	16384	248.6	157.2	0.0151	0.0095	8192	365.2	78.4	0.0445	0.0095
0.0219060	0.01	0.5	16384	321.7	164.0	0.0196	0.01	8192	505.5	83.2	0.0617	0.0101
0.0228456	0.001	0.9	16384	305.8	269.6	0.0186	0.0164	8192	331.0	129.9	0.0404	0.0158
0.026748	0.005	0.9	16384	375.1	285.2	0.0228	0.0174	8192	451.9	137.0	0.0551	0.0167
0.031626	0.01	0.9	16384	439.5	306.8	0.0268	0.0187	8192	573.0	147.8	0.0699	0.018
0.0512100	0.05	0.1	16384	833.0	405.6	0.0508	0.0247	8192	1407.9	211.8	0.1718	0.0258
0.0609300	0.05	0.5	16384	958.4	564.8	0.0584	0.0344	8192	1490.2	299.0	0.1819	0.0364
0.07065	0.05	0.9	16384	1109.2	728.4	0.0677	0.0444	8192	1576.2	368.7	0.1924	0.045
0.0999900	0.1	0.1	16384	1648.2	1333.6	0.1005	0.0813	8192	2388.6	713.1	0.2915	0.087
0.10971	0.1	0.5	16384	1770.5	1508.8	0.108	0.092	8192	2421.6	796.1	0.2956	0.0971
0.1194300	0.1	0.9	16384	1915.1	1650.4	0.1168	0.1007	8192	2453.3	857.4	0.2994	0.1046

Tabla 8.3.3.2. Experimentos realizados usando el código $RM(1,3)$ con un canal de Gilbert-Elliot.

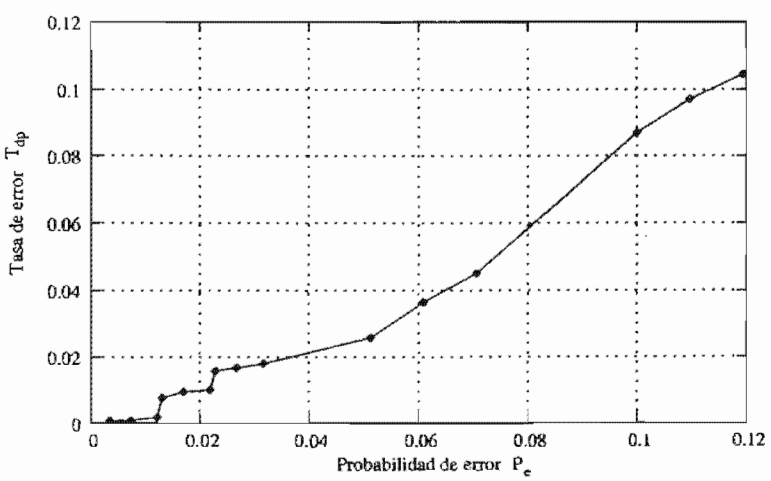


Figura 8.23: Comportamiento del código de Reed-Muller $RM(1,3)$.

Código de Reed-Muller $RM(1,4)$

Tipo de código:	$RM(1,4)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	5 bits	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	16 bits		0.05 y 0.1
Capacidad de corrección por palabra:	3 bits	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	32		
Tasa de transmisión:	0.3125	**Tasa de corrección:**	0.1875

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	26224	80.2	0.8	0.0030	0.0003	8195	184.9	0.2	0.0225	0.00002
0.007308	0.005	0.1	26224	181.8	0.0	0.0069	0.0	8195	421.3	0.0	0.0514	0.0
0.012186	0.01	0.1	26224	315.5	0.8	0.012	0.0003	8195	704.3	0.2	0.0859	0.0002
0.0131256	0.001	0.5	26224	307.2	63.2	0.0117	0.0024	8195	490.5	16.3	0.0598	0.0019
0.017028	0.005	0.5	26224	423.6	88.0	0.0161	0.0033	8195	728.0	21.3	0.0888	0.0025
0.0219060	0.01	0.5	26224	553.8	98.4	0.0211	0.0037	8195	985.0	25.6	0.1201	0.0031
0.0228456	0.001	0.9	26224	561.0	286.4	0.0213	0.0109	8195	649.7	66.1	0.0792	0.0080
0.026748	0.005	0.9	26224	643.2	292.0	0.0245	0.0111	8195	848.9	68.3	0.1035	0.0083
0.031626	0.01	0.9	26224	785.4	309.6	0.0299	0.0118	8195	1126.7	72.2	0.1374	0.0088
0.0512100	0.05	0.1	26224	1347.5	87.2	0.0513	0.0033	8195	2316.6	22.1	0.2826	0.0026
0.0609300	0.05	0.5	26224	1579.0	259.2	0.0602	0.0098	8195	2532.0	66.3	0.3089	0.0080
0.07065	0.05	0.9	26224	1791.5	510.4	0.0683	0.0194	8195	2527.6	126.3	0.3084	0.0154
0.0999900	0.1	0.1	26224	2608.1	747.2	0.0994	0.0284	8195	3338.7	200.1	0.4074	0.0244
0.10971	0.1	0.5	26224	2827.2	992.8	0.1078	0.0378	8195	3433.9	268.1	0.419	0.0327
0.1194300	0.1	0.9	26224	3106.7	1356.8	0.1184	0.0517	8195	3508.0	354.2	0.428	0.0432

Tabla 8.3.3.3. Experimentos realizados usando el código $RM(1,4)$ con un canal de Gilbert-Elliot.

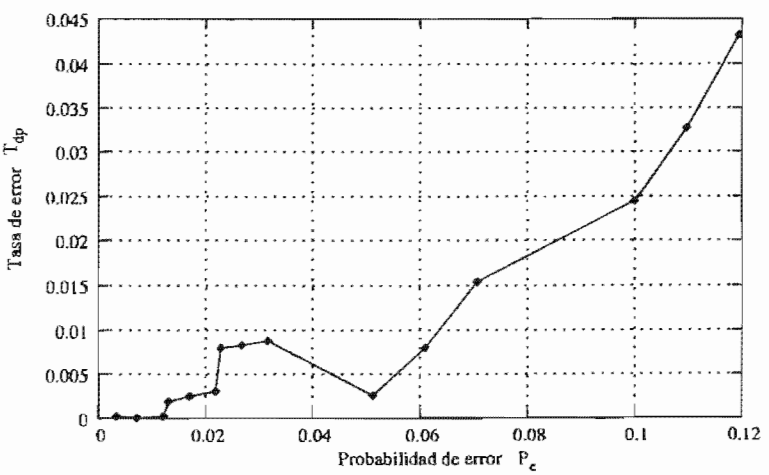


Figura 8.24: Comportamiento del código de Reed-Muller $RM(1,4)$.

Comentarios sobre los códigos de Reed-Muller

En la Figura 8.25 observamos que el código de Reed-Muller $RM(1,4)$ conserva la mayor recuperación de errores, y tiene un crecimiento moderado en comparación con los códigos $RM(1,3)$ y $RM(2,4)$. Sin embargo, hasta este momento es el código que ha presentado el mejor desempeño en el canal de Gilbert-Elliot.

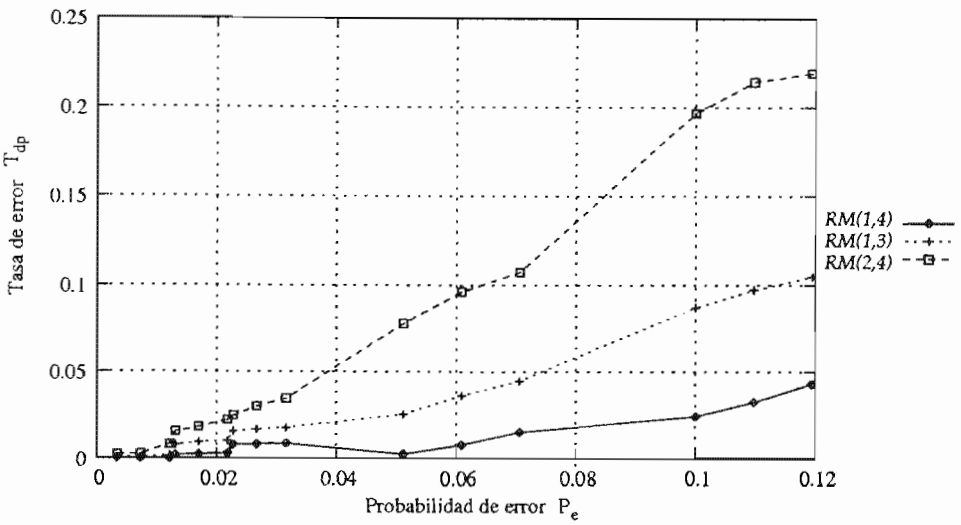


Figura 8.25: Comportamiento de los códigos de Reed-Muller.

8.4.4. Códigos de Reed-Solomon

Ahora presentamos los últimos códigos que probaremos en un canal con ruido de ráfagas, los códigos Reed-Solomon. De la misma forma que lo hemos venido haciendo, usaremos los mismos códigos que se usaron en las pruebas con ruido blanco, es decir, se usaron $RS(2^3,3)$, $RS(2^4,9)$Y $RS(2^5,17)$.

Código de Reed-Solomon $RS(2^3,3)$

Tipo de código:	$RS(2^3,3)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	5 símbolos	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	7 símbolos		0.05 y 0.1
Capacidad de corrección por palabra:	1 símbolo	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	3 bits		
Tasa de transmisión:	0.714285	**Tasa de corrección:**	0.0476 - 0.14285

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	11529	36.7	11.7	0.0031	0.0010	8235	128.7	13.0	0.0156	0.0015
0.007308	0.005	0.1	11529	83.4	23.8	0.0072	0.0020	8235	288.3	21.9	0.035	0.0026
0.012186	0.01	0.1	11529	139.0	55.7	0.012	0.0048	8235	463.1	71.5	0.0562	0.0086
0.0131256	0.001	0.5	11529	146.5	103.4	0.0127	0.0089	8235	369.1	100.3	0.0448	0.0121
0.017028	0.005	0.5	11529	181.1	119.1	0.0157	0.0103	8235	476.8	123.8	0.0578	0.015
0.0219060	0.01	0.5	11529	249.9	169.0	0.0216	0.0146	8235	681.2	196.9	0.0827	0.0239
0.0228456	0.001	0.9	11529	242.6	206.5	0.021	0.0179	8235	453.7	181.3	0.055	0.022
0.026748	0.005	0.9	11529	297.3	240.9	0.0257	0.0208	8235	620.5	222.3	0.0753	0.0269
0.031626	0.01	0.9	11529	349.7	281.7	0.0303	0.0244	8235	745.7	280.4	0.0905	0.034
0.0512100	0.05	0.1	11529	590.0	526.0	0.0511	0.0456	8235	1629.6	760.6	0.1978	0.0923
0.0609300	0.05	0.5	11529	695.5	657.1	0.0603	0.0569	8235	1746.3	897.4	0.212	0.1089
0.07065	0.05	0.9	11529	792.9	765.9	0.0687	0.0664	8235	1840.1	981.6	0.2234	0.1191
0.0999900	0.1	0.1	11529	1146.0	1286.7	0.0994	0.1116	8235	2501.5	1815.3	0.3037	0.2204
0.10971	0.1	0.5	11529	1275.0	1436.2	0.1105	0.1245	8235	2616.7	1926.5	0.3177	0.2339
0.1194300	0.1	0.9	11529	1359.9	1526.1	0.1179	0.1323	8235	2676.9	1975.1	0.325	0.2398

Tabla 8.3.4.1. Experimentos realizados usando el código $RS(2^3, 3)$ con un canal de Gilbert-Elliot.

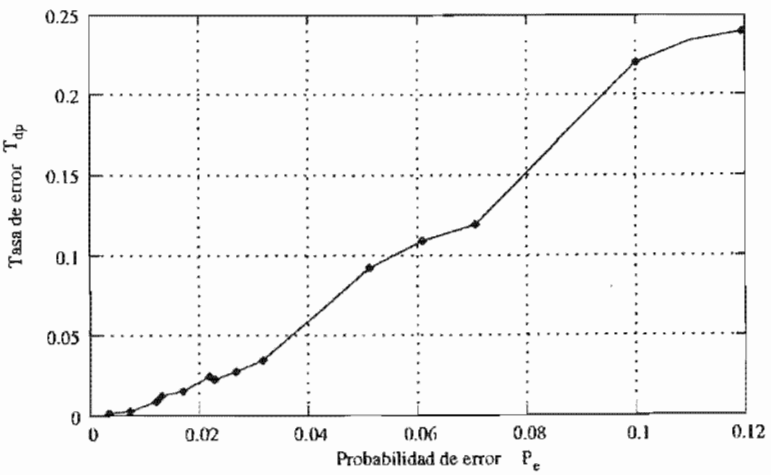


Figura 8.26: Comportamiento del código de Reed-Solomon $RS(2^3, 3)$.

Código de Reed-Solomon $RS(2^4, 9)$

Tipo de código:	$RS(2^4, 9)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	7 símbolos	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01, 0.05 y 0.1
Longitud de la palabra de código:	15 símbolos		
Capacidad de corrección por palabra:	4 símbolos	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Cardinalidad:	268435456		
Tasa de transmisión:	0.46666	**Tasa de corrección:**	0.066 - 0.266

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	17580	56.0	3.5	0.0031	0.0001	8204	164.8	0.0	0.02	0.0
0.007308	0.005	0.1	17580	132.2	9.4	0.0075	0.0005	8204	364.8	0.0	0.0444	0.0
0.012186	0.01	0.1	17580	218.4	16.6	0.0124	0.0009	8204	613.6	6.9	0.0747	0.0008
0.0131256	0.001	0.5	17580	237.6	17.7	0.0135	0.0010	8204	475.9	5.4	0.058	0.0006
0.017028	0.005	0.5	17580	280.2	22.6	0.0159	0.0012	8204	596.4	5.2	0.0726	0.0006
0.0219060	0.01	0.5	17580	390.5	52.6	0.0222	0.0029	8204	848.1	29.6	0.1033	0.0036
0.0228456	0.001	0.9	17580	393.3	43.2	0.0223	0.0024	8204	604.4	22.1	0.0736	0.0026
0.026748	0.005	0.9	17580	461.4	64.1	0.0262	0.0036	8204	769.9	34.8	0.0938	0.0042
0.031626	0.01	0.9	17580	553.9	81.5	0.0315	0.0046	8204	966.9	50.7	0.1178	0.0061
0.0512100	0.05	0.1	17580	905.0	315.8	0.0514	0.0179	8204	1882.6	395.5	0.2294	0.0482
0.0609300	0.05	0.5	17580	1070.3	513.6	0.0608	0.0292	8204	1986.0	630.7	0.242	0.0768
0.07065	0.05	0.9	17580	1240.5	661.3	0.0705	0.0376	8204	2124.7	741.1	0.2589	0.0903
0.0999900	0.1	0.1	17580	1757.2	1615.5	0.0999	0.0918	8204	2740.9	2030.4	0.334	0.2474
0.10971	0.1	0.5	17580	1910.1	1822.5	0.1086	0.1036	8204	2828.8	2177.3	0.3448	0.2653
0.1194300	0.1	0.9	17580	2101.4	2037.2	0.1195	0.1158	8204	2865.1	2292.5	0.3492	0.2794

Tabla 8.3.4.2. Experimentos realizados usando el código $RS(2^4, 9)$ con un canal de Gilbert-Elliot.

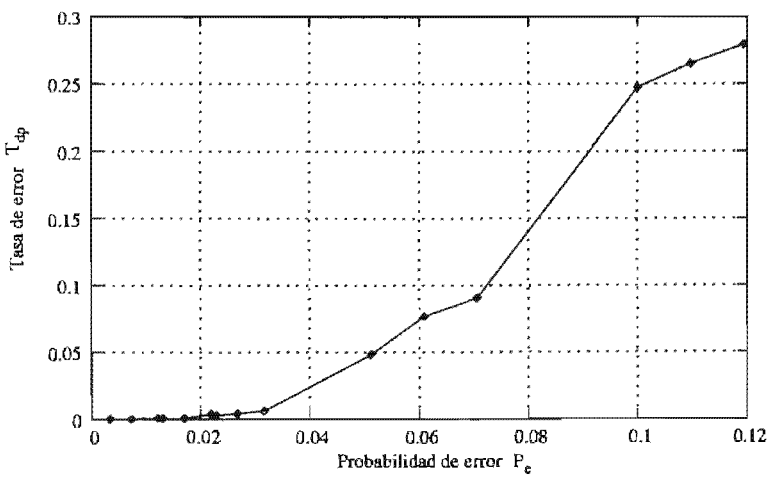


Figura 8.27: Comportamiento del código de Reed-Solomon $RS(2^4, 9)$.

Código de Reed-Solomon $RS(2^5, 17)$

Tipo de código:	$RS(2^5, 17)$	**Tipo de canal:**	Canal de Gilbert-Elliot.
Longitud del mensaje:	15 símbolos	1a. probabilidad (P_{c_1}):	0.001, 0.005, 0.01,
Longitud de la palabra de código:	31 símbolos		0.05 y 0.1
Capacidad de corrección por palabra:	8 símbolos	2a. probabilidad (P_{c_2}):	0.1, 0.5 y 0.9
Longitud de símbolos	5 bits		
Cardinalidad (m):	3.77×10^{22}		
Tasa de transmisión:	0.4838	**Tasa de corrección:**	0.05161 - 0.2580

P_e	P_{c_1}	P_{c_2}	B_b	B_e	B_{ep}	T_e	T_{ep}	D_b	D_e	D_{ep}	T_d	T_{dp}
0.0034056	0.001	0.1	17360	57.5	1.3	0.0033	0.00007	8400	467.5	0.0	0.0556	0.0
0.007308	0.005	0.1	17360	124.8	3.4	0.0071	0.0001	8400	1018.9	0.0	0.1212	0.0
0.012186	0.01	0.1	17360	206.3	6.3	0.0118	0.0003	8400	1482.5	0.0	0.1764	0.0
0.0131256	0.001	0.5	17360	239.8	4.8	0.0138	0.0002	8400	1151.0	0.0	0.137	0.0
0.017028	0.005	0.5	17360	298.4	6.9	0.0171	0.0003	8400	1519.2	0.0	0.1808	0.0
0.0219060	0.01	0.5	17360	385.6	12.6	0.0222	0.0007	8400	1963.4	11.5	0.2337	0.0013
0.0228456	0.001	0.9	17360	391.4	8.3	0.0225	0.0004	8400	1421.7	0.0	0.1692	0.0
0.026748	0.005	0.9	17360	443.9	14.3	0.0255	0.0008	8400	1653.2	2.5	0.1968	0.0029
0.031626	0.01	0.9	17360	552.6	47.0	0.0318	0.0027	8400	2222.9	44.7	0.2646	0.0053
0.0512100	0.05	0.1	17360	890.4	422.4	0.0512	0.0243	8400	3312.4	1003.6	0.3943	0.1194
0.0609300	0.05	0.5	17360	1056.5	637.3	0.0608	0.0367	8400	3383.3	1461.6	0.4027	0.174
0.07065	0.05	0.9	17360	1223.6	830.7	0.0704	0.0478	8400	3387.4	1657.5	0.4032	0.1973
0.099918	0.08	0.9	17360	1736.4	1813.5	0.1	0.1044	8400	3655.3	3339.1	0.4351	0.3975
0.0999900	0.1	0.1	17360	1735.5	1872.6	0.0999	0.1078	8400	3751.8	3550.9	0.4466	0.4227
0.10971	0.1	0.5	17360	1910.0	2077.5	0.11	0.1196	8400	3740.4	3660.1	0.4452	0.4357
0.1194300	0.1	0.9	17360	2061.2	2227.4	0.1187	0.1283	8400	3793.5	3745.4	0.4516	0.4458

Tabla 8.3.4.3. Experimentos realizados usando el código $RS(2^5, 17)$ con un canal de Gilbert-Elliot.

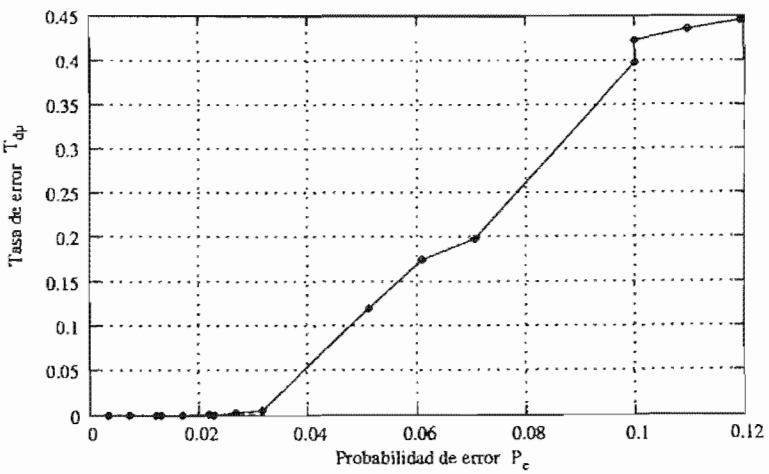


Figura 8.28: Comportamiento del código de Reed-Solomon $RS(2^5, 17)$.

Comentarios sobre los códigos de Reed-Solomon

Son los códigos de Reed-Solomon los que mejor responden frente al ruido del canal de Gilbert-Elliot. En este caso, fue el único código que pudo realizar una recuperación *completa* en los diez experimentos cuando $P_{c_2} = 0.9$ fué $RS(2^5, 17)$. Sin embargo, obsérvese que en la Figura 8.29, los códigos $RS(2^4, 9)$ y $RS(2^5, 17)$ mantienen un comportamiento muy similar en un inicio, pero aproximadamente a partir de la probabilidad de error 0.032 el código $RS(2^5, 17)$ tiene una menor recuperación de errores.

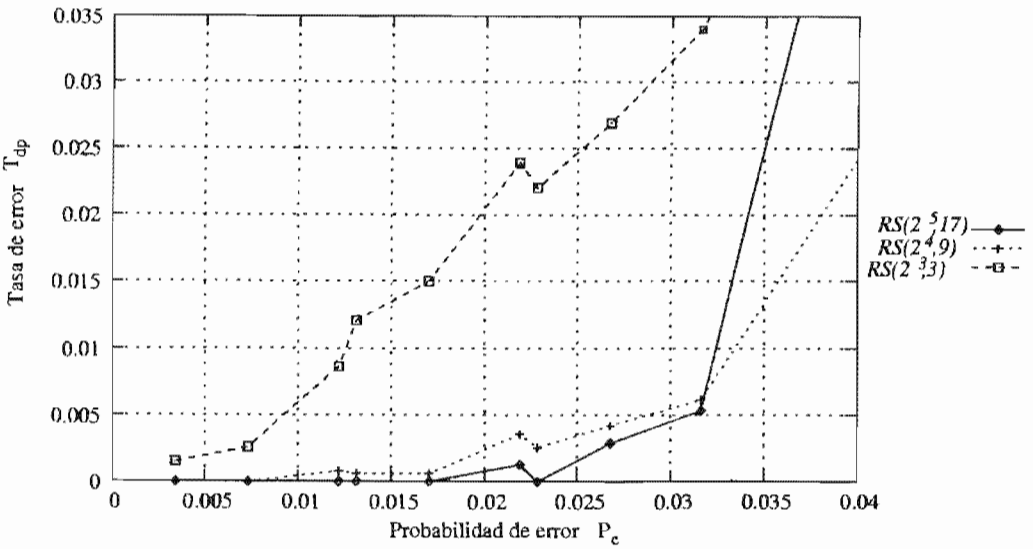


Figura 8.29: Comportamiento de los código de Reed-Solomon.

8.4.5. Comportamiento de los códigos en el canal de Gilbert-Elliot

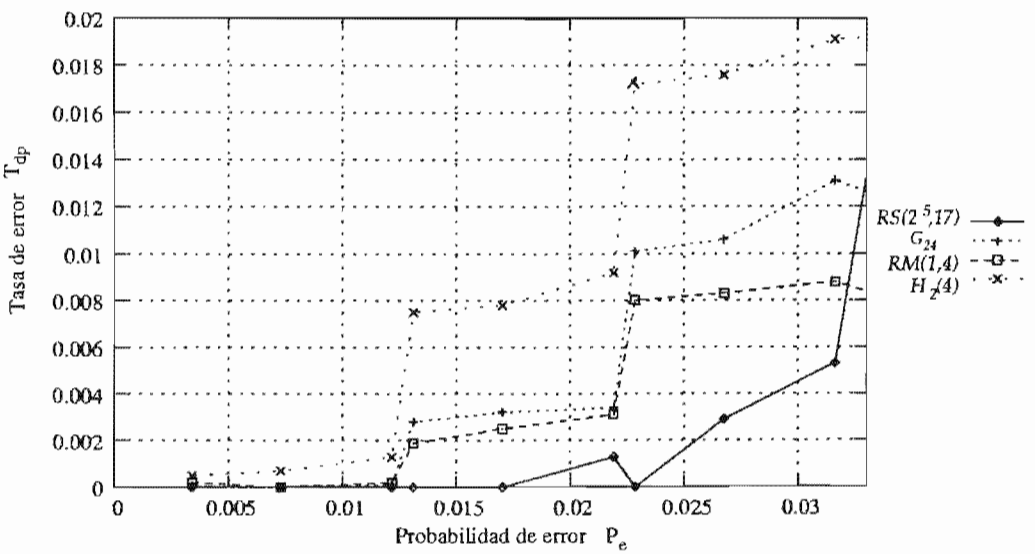


Figura 8.30: Comportamiento de los códigos en un canal simétrico binario.

Como el lector puede observar en la Figura 8.30, el código de $RS(2^5, 17)$ es el que mejor responde al ruido de ráfagas cuando se trata de probabilidades de error inferiores a 0.03. Después

de este punto el rendimiento del código va disminuyendo a un ritmo acelerado. No obstante, como vimos en el Cuadro 8.2, las probabilidades que presenta un canal de transmisión son, en general, inferiores a 0.03, es decir, su desempeño es favorable siempre y cuando consideremos probabilidades de error usuales en los distintos canales de transmisión.

Podemos observar que a diferencia de lo que ocurrió con el canal simétrico binario, donde el código de menor rendimiento fué Reed-Solomon, bajo este tipo de ruido es el código más apropiado.

Es interesante observar que los códigos G_{24}, $RM(1,4)$ y $RS(2^5,17)$ inician con rendimientos muy parecidos, solamente el código $H_2(4)$ tiene un menor rendimiento. No obstante, debe recordarse que los valores más pequeños de la probabilidad de error P_e simulan un comportamiento más cercano a un canal simétrico binario que a un canal de Gilbert-Elliot. De tal suerte, que es hasta que la segunda probabilidad de cruzamiento P_{c_2} aumenta, cuando se simula mejor el ruido de ráfagas.

Capítulo 9

Conclusiones

Cuando iniciamos la comparación de los códigos consideramos que no existirían muchos elementos inesperados por la información que habíamos reunido hasta ese momento. Sin embargo, pudimos observar que nuestra hipótesis no fue del todo corroborada. La hipótesis menciona dos puntos esencialmente, el primero relacionado con la capacidad de corrección de un código frente al canal y el segundo con respecto a una suposición del mejor código corrector y detector de errores.

Empezaremos con lo relacionado con el primer punto de la hipótesis, según la cual, si teníamos un código con capacidad de corrección de 1 bit por cada 10 bits entonces se recuperarían la mayor parte de los mensajes que originalmente fueron enviados usando un canal simétrico binario con $P_c = 0.1$. Sin embargo, el resultado de los experimentos mostró un rendimiento inferior. Una observación más detallada del comportamiento de los errores sobre las palabras de código nos muestra que aun manteniendo la probabilidad de cruzamiento $P_c = 0.1$, es muy frecuente que ocurran más de 1 error en un bloque de 10 bits, porque hay muchos otros bloques que no tienen ningún bit alterado, de tal forma que la capacidad de corrección del código se rebasa con facilidad.

En lo referente al segundo punto, habíamos supuesto que el código que presentaría un mejor desempeño sería el código de Reed-Solomon. Pero, en el canal simétrico binario el desempeño de estos códigos fué inferior al de los tres códigos restantes. Por el contrario, durante la simulación de la transmisión usando el canal de Gilbert-Elliot, los códigos Reed-Solomon obtuvieron el mejor desempeño. Podemos explicarnos este comportamiento al hacer una observación más cuidadosa. Como los códigos de Reed-Solomon corrigen a nivel de símbolos y no de bits entonces conviene que los errores estén agrupados en un bloque, de tal suerte, que los bits erróneos involucren a la menor cantidad de símbolos. Esto último ocurre en el canal de Gilbert-Elliot, pero no así en el canal simétrico binario, donde los errores están distribuidos a lo largo de la transmisión.

En los experimentos realizados usando el canal simétrico binario, el mejor desempeño se presentó con el código Reed-Muller $RM(1,4)$. Sin embargo también este código presenta la menor capacidad de transmisión.

Por último, el código más adecuado está en función del tipo de canal que se usa durante la transmisión y el tipo de ruido al que se ve afectado. Cada código ofrece características distintas

que pueden hacerlo más adecuado o menos adecuado a una situación dada. También debe de tomarse en cuenta que aun cuando los códigos Reed-Muller y Reed-Solomon mantuvieron el mejor rendimiento, también tienen un proceso de decodificación más complicado que los códigos de Golay y Hamming, aunque tanto Reed-Muller como Reed-Solomon ofrecen la flexibilidad de crear un código que se adapte en la medida de lo posible a las características del canal. Esta flexibilidad consiste en que con estos códigos existe la posibilidad de cambiar los parámetros que los generan, de tal forma que puedan corregir una mayor cantidad de errores. Esta última ventaja es inexistente en los códigos de Hamming, dado que estos códigos, siempre corrigen un solo error por palabra de código. No obstante, el que un código, independientemente de cual se trate, corrija más errores ocasionará que se agreguen más bits de redundancia, trayendo como consecuencia que la tasa de transmisión disminuya. Aquí regresamos a la disyuntiva característica de la teoría de códigos *encontrar un equilibrio entre la capacidad de corrección y la cantidad de bits de información que se desea transmitir.*

Bibliografía

[1] Arazi, Benjamin. *A Commonsense Approach to the Theory of Error Correcting Codes.* Estados Unidos, Ed. MIT Press, 1987.

[2] Bartur, Meir. *Single Fiber, Single wavelength. Dual Rate GbE / FE transceiver.* IEEE 802.3. Mayo 2002.

[3] Cooke, Ben. *Reed-Muller Error Correcting Codes.* The MIT Undergraduate Journal of Mathematics. 1999.

[4] Chang, Edward S. y Richard Taborek. *Recommendation of 10-13 Bit Error Rate for 10 Gigabit Ethernet.* Unisys Corporation y Transcendata, Inc. Julio 1999.

[5] Galavíz Casas, José. *Introducción a la Teoría de Códigos, Teoría de la Información y Criptografía.* Notas de clase. Departamento de Matemáticas. Universidad Nacional Autónoma de México. 2000.

[6] Gilbert, William J. *Modern Algebra with Applications.* Estados Unidos, Ed. Jon Wiley & Sons, 1976.

[7] Hamming, R. W. *Coding and Information Theory.* Estados Unidos, Ed. Prentice-Hall, 1980.

[8] Hoffman, D. G. et. al. *Coding Theory: The Essentials.* Estados Unidos, Ed. Marcel Dekker, 1991.

[9] Kish, Paul *The Effect of Network Cabling on Bit Error Rate Performance.* NORDX/CDT. 2002.

[10] Lint, J. H. Van. *Introduction to Coding Theory.* Alemania, Ed. Springer-Verlag, 1999.

[11] Lluis-Puebla, Emilio. *Álgebra Lineal. Álgebra Multilineal y k-teoría algebraica clásica.* México, Ed. Sistemas Técnicos de Edición, 1997.

[12] Mushkin, Mordechai e Israel Bar-David. *Capacity and Coding for the Gilbert-Elliot Channels.* IEEE Transactions of Information Theory, Vol. 35, No. 6. Noviembre 1989.

[13] Reed, Irving S. y Xuemin Chen. *Error-Control Coding for Data Networks.* Estados Unidos, Ed. Kluwer Academic Publishers, 1999.

[14] Roman, Steve. *Coding and Information Theory.* Estados Unidos, Ed. Springer-Verlag, 1992.

[15] Roman, Steve. *Introduction to Coding and Information Theory.* Estados Unidos, Ed. Springer, 1997.

[16] Shannon, C. E. *A Mathematical Theory of Communication.* The Bell System Technical Journal. Vol. 27, pp 379-423, 623-666. July, October, 1948.

[17] Sun, Pen-Ting. *Similarity of Discrete Gilbert-Elliot and Polya Channel Models to Continuos Rayleigh Fading Channel Model.* Departamento de Ingeniería en Comunicaciones de la Universidad Nacional de Chiao Tung, Taiwan. Junio 2002.

[18] Togneri, Roberto y Christopher J. S. de Silva. *Fundamentals of Information Theory and Coding Design.* Estados Unidos, Ed. Chapman & Hall, 2002.