Un dibujo animado con letras

Descripción generada automáticamente con confianza bajaUniversidad Tecnológica Nacional

Facultad Regional Rosario

Especialidad: Ing. En Sistemas de Información.

Asignatura: Comunicación de datos.

Comisión: 302

Fecha: 03/06/2024

INFORME N°3 – Códigos corrector de error

|  |  |  |
| --- | --- | --- |
| **Alumno** | **Legajo** | **Firma** |
| DEL POPOLO, Chiara | 49941 |  |
| MONDINO, Juan Cruz | 51922 |  |
| MORENO, Natan | 47962 |  |
| PONTELLI, Juan Martín | 51315 |  |

Contenido

[Glosario 3](#_Toc168570925)

[Introducción 6](#_Toc168570926)

[Códigos corrector de error 7](#_Toc168570927)

[Códigos de bloque 7](#_Toc168570928)

[Código Hamming 8](#_Toc168570929)

[Códigos cíclicos 14](#_Toc168570930)

[Código de paridad 15](#_Toc168570931)

[Código Bose – Chaudhuri – Hocquenghem 17](#_Toc168570932)

[Código de repetición 24](#_Toc168570933)

[Códigos convolucionales 27](#_Toc168570934)

[Código turbo 29](#_Toc168570935)

[Código de Hadamard 32](#_Toc168570936)

[Códigos FEC 34](#_Toc168570937)

[Código Reed-Solomon 38](#_Toc168570938)

[Código LDPC 40](#_Toc168570939)

[Conclusión 44](#_Toc168570940)

[Cuestionario 45](#_Toc168570941)

[Códigos de bloque 45](#_Toc168570942)

[Código Hamming 45](#_Toc168570943)

[Códigos cíclicos 46](#_Toc168570944)

[Código de paridad 46](#_Toc168570945)

[Códigos BCH 46](#_Toc168570946)

[Código de repetición 47](#_Toc168570947)

[Códigos convolucionales 47](#_Toc168570948)

[Código turbo 48](#_Toc168570949)

[Código de Hadamard 48](#_Toc168570950)

[Códigos FEC 48](#_Toc168570951)

[Código Reed-Solomon 49](#_Toc168570952)

[Código LDPC 49](#_Toc168570953)

[Bibliografía 49](#_Toc168570954)

# Glosario

**Bits redundantes:** Teóricamente es posible corregir cualquier fragmento de código binario automáticamente. Para ello, en puesto de los códigos detectores de errores utilizando los códigos correctores de errores, de mayor complejidad matemática y mayor número de bits redundantes necesarios. La necesidad de mayor número de bits redundantes hace que a veces la corrección de múltiples bits sea inviable e ineficiente por el elevado número de bits necesarios. Por ello normalmente los códigos correctores de error se reducen a la corrección de 1,2 o 3 bits.

**Distancia Hamming:** La distancia Hamming H entre dos secuencias binarias S1 y S2 de la misma longitud, viene definida por el número de bits en que difieren.

El algoritmo de la distancia de Hamming calcula una puntuación de coincidencia para dos cadenas de datos calculando el número de posiciones en las que los caracteres difieren de una cadena de datos a otra. En el caso de que las cadenas tengan una longitud diferente, cada carácter adicional de la cadena más larga se cuenta como una diferencia de una cadena a otra.

En teoría de la información se denomina distancia de Hamming a la efectividad de los códigos de bloque y depende de la diferencia entre una palabra de código válida y otra. Cuanto mayor sea esta diferencia, menor es la posibilidad de que un código válido se transforme en otro código válido por una serie de errores. A esta diferencia se le llama distancia de Hamming, y se define como el número de bits que tienen que cambiarse para transformar una palabra de código válida en otra palabra de código válida.

Si dos palabras de código difieren en una distancia d, se necesitan d errores para convertir una en la otra.

**Campo de Galois:** Un campo de Galois es un tipo específico de campo finito en matemáticas. Para entenderlo, primero se necesita saber qué es un campo y qué significa que sea finito.

* Campo: Un campo es un conjunto de números en el que puedes realizar las operaciones de suma, resta, multiplicación y división (excepto por cero) y todas estas operaciones cumplen ciertas propiedades (asociatividad, conmutatividad, existencia de elementos neutros y de inversos).
* Campo finito: Un campo finito es un campo que contiene un número finito de elementos.

Un campo de Galois, denotado como GF(pn), es un campo finito que tiene exactamente pn elementos, donde p es un número primo y n es un número entero positivo.

Notación: GF significa "Galois Field" (Campo de Galois).

Elementos: El campo contiene pn elementos, que es una potencia de un número primo p.

Propiedades claves:

* Característica del campo: La característica del campo es p, el número primo. Esto significa que sumar p veces el número 1 te da 0. Por ejemplo, en GF(2), 1 + 1 = 0.
* Estructura algebraica: Los elementos de GF(pn) pueden ser vistos como polinomios con coeficientes en GF(p) (el campo de p elementos), y las operaciones de suma y multiplicación se realizan módulo un polinomio irreducible de grado n.

Ejemplo: GF(2): El campo de Galois con 2 elementos, 0 y 1. En este campo, la aritmética es muy simple:

* 0+0=0
* 0+1=1
* 1+1=0 (ya que se trabaja en módulo 2)
* 1×1=1

**Elemento primitivo:** es un elemento del campo que genera todos los elementos no nulos del campo a través de sus potencias sucesivas.

Para un campo finito GF(q) con q = pm, donde p es un número primo y m es un entero positivo:

1. Generador del grupo multiplicativo: El grupo multiplicativo de GF(q) (es decir, el conjunto de todos los elementos no nulos del campo, bajo la operación de multiplicación) tiene q-1 elementos. Este grupo es cíclico, lo que significa que hay un elemento Alpha (α) en el campo tal que las potencias de α generan todos los q-1 elementos no nulos del campo. Este elemento α se llama un elemento primitivo del campo.

2. Propiedad de un elemento primitivo: Un elemento α de GF(q) es primitivo si las potencias α0, α1, α2, …, αq-2 son distintas y cubren todos los elementos no nulos de GF(q). Es decir, α tiene orden q-1, lo que significa que αq-1 = 1 y no hay ningún exponente menor que q-1 tal que esta igualdad se mantenga.

Ejemplo:

Considera el campo finito GF(7). Este campo tiene 7 elementos: {0, 1, 2, 3, 4, 5, 6}. El grupo multiplicativo {1, 2, 3, 4, 5, 6} es cíclico y un elemento primitivo de este grupo es el 3, porque:

30 ≡ 1 mod{7}

31 ≡ 3 mod{7}

32 ≡ 2 mod{7}

33 ≡ 6 mod{7}

34 ≡ 4 mod{7}

35 ≡ 5 mod{7}

36 ≡ 1 mod{7}

Las potencias de 3 generan todos los elementos no nulos de GF(7), por lo que 3 es un elemento primitivo de GF(7).

**División Euclidiana:** es un procedimiento para dividir dos polinomios (o dos números) y obtener un cociente y un residuo.

**Síndrome:** Cuando se transmite una palabra de código a través de un canal de comunicación, esta puede ser afectada por errores. El receptor recibe una palabra que puede no ser la misma que la palabra de código enviada. El síndrome es una herramienta que permite determinar si ha habido errores y, en caso afirmativo, ayudar a localizar y corregir dichos errores.

# Introducción

En plena era digital la transmisión de información que se produce en diferentes ámbitos de la actividad humana es enorme, incluido el almacenamiento y lectura de datos en diferentes soportes informáticos y/o electrónicos. Esta transmisión de información se realiza mediante la utilización de códigos (mensajes), empleando para ello diferentes canales: cables de cobre, cables de fibra óptica, ondas electromagnéticas, etc. En ocasiones existen interferencias, ruido, que afectan a estos canales y pueden introducir errores en la transmisión. Por este motivo es necesario detectar cuándo se han producido errores y poder corregirlos cuando sea necesario. Precisamente, este es el papel de los códigos correctores, es decir, permitir no solo detectar un error de transmisión, sino también corregirlo sin necesidad de repetirla.

Los códigos correctores de errores no son un protocolo de comunicaciones, son una forma de codificar la información que se va a transmitir para que el equipo o centro receptor de la información pueda detectar si ha habido errores en la transmisión y corregirlos.

En la actualidad es de uso generalizado los llamados "códigos detectores", es decir, aquellos que nos avisan de que la transmisión de información ha fallado pero que no son capaces de corregir dicho error. Un ejemplo claro es en los códigos de barras adosados a los productos existentes en cualquier tipo de tiendas o comercios, produciéndose en ocasiones, en el proceso de lectura, fallos de transmisión de la información una vez se pasa por los mismos el correspondiente lector de códigos. El dispositivo nos avisa mediante un pitido, u otro medio de alarma, de que se ha producido un error de transmisión – código detector – pero no es capaz de corregirlo por la ausencia de un "código corrector", que evitase los perjuicios que ello puede suponer a los clientes que sufran dicho fallo.

# Códigos corrector de error

Se han diseñado dos estrategias diferentes para el tratamiento de los errores:

Códigos detectores de error: Consiste en incluir en los datos transmitidos, una cantidad de bits redundantes de forma que permita al receptor detectar que se ha producido un error, pero no qué tipo de error ni dónde, de forma que tiene que solicitar retransmisión.

Códigos correctores de error: Consiste en la misma filosofía que el anterior, incluir información redundante, pero, en este caso, la suficiente como para permitirle al receptor deducir cual fue el carácter que se transmitió, por lo tanto, el receptor tiene capacidad para corregir un número limitado de errores.

Los códigos correctores de errores pueden ser clasificados, principalmente, en códigos de bloque, códigos cíclicos y códigos convolucionales.

## Códigos de bloque

Un código es un código de bloque (n, m) si la información que se codificará se puede dividir en bloques de m dígitos binarios, cada uno de los cuales puede ser codificado en n dígitos binarios. Más específicamente, un código de bloque (n, m) consiste en una **función codificadora**:

y una **función decodificadora**:

El dos es porque se supone que la transmisión es de un código binario.

Este tipo de código no tiene memoria.

Un ejemplo de este tipo de código es el código Hamming.

### Código Hamming

El código de Hamming agrega tres bits adicionales de comprobación por cada cuatro bits de datos del mensaje. El algoritmo de Hamming puede corregir cualquier error de un solo bit, pero cuando hay errores en más de un bit, la palabra transmitida se confunde con otra con error en un solo bit, siendo corregida, pero de forma incorrecta, es decir que la palabra que se corrige es otra distinta a la original, y el mensaje final será incorrecto sin saberlo.

Con este nombre se conoce a los tipos de códigos correctores en k dígitos binarios. A la hora de trabajar con este tipo de códigos se puede distinguir dos operaciones:

a) **Construcción** de la palabra, que se realizará en el centro emisor.

b) **Interpretación** de la palabra, que se realizará en el centro receptor.

a) Construcción: Se parte de un código de n dígitos de distancia mínima uno. Estos n dígitos son conocidos dentro del código de Hamming como "dígitos de datos". A continuación se le añaden p (cp-1, ..., c2, c1, c0) dígitos denominados de control o paridad. Así pues, el nuevo código tendrá una longitud de palabra (l) de l = n + p. La numeración de los dígitos es la habitual (de derecha a izquierda) pero comenzando por uno: dn+p, dn+p-1, ..., d2, d1.

Cada uno de estos p dígitos que se añaden al código original va a afectar a unas determinadas posiciones de la nueva palabra de código de n + p dígitos, de forma que tomaran el valor adecuado para que se cumpla el criterio de paridad (par o impar) preestablecido en las subcombinaciones afectadas por cada uno. Se tiene, entonces, que en la construcción del código los p dígitos añadidos actúan como dígitos de paridad.

b) Interpretación: Recibida una combinación de un código de Hamming hay que comprobar si es correcta, y de no ser así habrá que detectar el dígito que varió en la transmisión.

Ahora los p dígitos añadidos actúan como dígitos de control y con ellos se forma una palabra binaria. Cada uno de los dígitos de esta palabra toma el valor 0 o 1 dependiendo de si el número de unos de las posiciones de la palabra de código por el afectadas cumplen o no el criterio de paridad establecido. Interpretando la combinación resultante en binario natural, se tendrán dos posibilidades:

* Que se corresponda con el 0. Entonces quiere decir que la transmisión ha sido correcta.
* Que se corresponda a un número distinto del 0. Entonces en la transmisión ha variado el dígito situado en la posición indicada por ese número. Quedan varias cuestiones por resolver:

1. Cómo calcular p.
2. A que posiciones afecta cada uno de los p dígitos de control o paridad.
3. Dónde se colocan estos dígitos dentro de la palabra de código.

I. Dada la forma de calcular la posición errónea, con p dígitos binarios se tiene que poder detectar el error en todas y cada una de la n + p posiciones de la palabra de código. Como la combinación formada por los p dígitos de control se interpreta en binario natural, se debe cumplir que:

2p - 1 ≥ n + p

Donde 2p -1 es el mayor número que se puede representar en binario natural con p dígitos.

II.- Se construyen todas las combinaciones posibles con p dígitos de control, y se interpreta cada una en binario natural:

Imagen que contiene Tabla

Descripción generada automáticamente

Ilustración 1

Cada dígito de control ha de afectar a aquellas posiciones en las que sea capaz de detectar error, o sea, va a afectar a las posiciones de la tabla anterior para las que ese dígito valga 1:

Texto, Carta

Descripción generada automáticamente

Ilustración 2

III.- Han de colocarse en aquellas posiciones en las que no se vean afectados por otro dígito de control, así no existirán ambigüedades a la hora de otorgarles valor en la creación del código. Estas posiciones han de ser entonces:

Tabla

Descripción generada automáticamente con confianza media

Ilustración 3

#### Ejemplo

Partiendo de un supuesto de que se quiere aplicar este método a un código binario de 4 bits.

a) Creación:

1- Se calcula el número de dígitos de control necesarios, sabiendo por el enunciado que n= 4:

Se llega a la conclusión de que p = 3, por lo que la longitud de la palabra código (l) será l = n + p = 4 + 3 = 7.

2- Ahora, hay que encontrar las posiciones de la palabra código que son afectadas por los dígitos de control, c1, c2 y c3:

**Imagen que contiene Tabla

Descripción generada automáticamente**

Ilustración 4

Por lo tanto, los dígitos de control lograrán hacer control sobre los siguientes dígitos:

C0: d1, d3, d5, d7

C1: d2, d3, d6, d7

C2: d4, d5, d6, d7

3- Los dígitos de control se posicionan en el lugar del primer dígito que controlan:

C0: d1

C1: d2

C2: d4

4- Se construye el código Hamming:

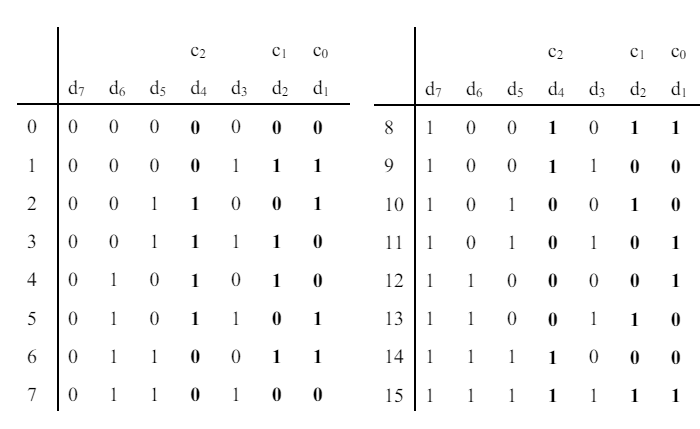


Ilustración 5

b) Interpretación:

Para analizar la interpretación del código Hamming, se supondrán 3 casos:

1. Alteración de un bit de datos:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | d7 | d6 | d5 | d4 | d3 | d2 | d1 |
| Transmitido | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Recibido | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

Hay error, pero ¿cuál es el bit que está mal? Este bit se encuentra analizando la cantidad de unos en cada set de dígitos que controlado por cada dígito de control:

* c0 🡪 d7, d5, d3, d1 = 3 unos 🡪 impar 🡪 c0 = 1.
* c1 🡪 d7, d6, d3, d2 = 3 unos 🡪 impar 🡪 c1 = 1.
* c2 🡪 d7, d6, d5, d4 = 3 unos 🡪 impar 🡪 c2 = 1.

El bit erróneo está en la posición: c0 c1 c2 = 1 1 1(2) = 7.

1. Alteración de un bit de control:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | d7 | d6 | d5 | d4 | d3 | d2 | d1 |
| Transmitido | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Recibido | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

Hay error, pero ¿cuál es el bit que está mal? Este bit se encuentra analizando la cantidad de unos en cada set de dígitos que controlado por cada dígito de control:

* c0 🡪 d7, d5, d3, d1 = 2 unos 🡪 par 🡪 c0 = 0.
* c1 🡪 d7, d6, d3, d2 = 2 unos 🡪 par 🡪 c1 = 0.
* c2 🡪 d7, d6, d5, d4 = 3 unos 🡪 impar 🡪 c2 = 1.

El bit erróneo está en la posición: c0 c1 c2 = 1 0 0(2) = 4.

1. No hay error:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | d7 | d6 | d5 | d4 | d3 | d2 | d1 |
| Transmitido | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Recibido | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

Hay error, pero ¿cuál es el bit que está mal? Este bit se encuentra analizando la cantidad de unos en cada set de dígitos que controlado por cada dígito de control:

* c0 🡪 d7, d5, d3, d1 = 2 unos 🡪 par 🡪 c0 = 0.
* c1 🡪 d7, d6, d3, d2 = 2 unos 🡪 par 🡪 c1 = 0.
* c2 🡪 d7, d6, d5, d4 = 2 unos 🡪 par 🡪 c2 = 0.

El bit erróneo está en la posición: c0 c1 c2 = 0 0 0(2) = 4. Por lo tanto, no hay error en la combinación recibida.

## Códigos cíclicos

Un código [n, k, d] lineal C sobre el cuerpo Fq se dice que es cíclico si para cualquier palabra del código c = (c0, c1, ..., cn−1) se tiene que σ(c) = (cn−1, c0, ..., cn−2) es también una palabra del código.

En otras palabras, un código lineal C es cíclico si es cerrado por el desplazamiento cíclico:

Como consecuencia, C es un código cíclico si σr(c) ∈ C para c ∈ C y r ∈ N (r ≥ 0); pero esto es lo mismo que decir que C es un c**ó**digo cíclico si:

### Código de paridad

Es un mecanismo de codificación usual, es mucho más eficiente que la simple repetición. El código ASCII (American Standard Code for Information Interchange) usa 8-tuplas binarias, dando lugar a 28 = 256 8-tuplas posibles. Pero, solo se necesitan 7 bits pues solo hay 27 = 128 caracteres ASCII. ¿Qué se puede o debe hacer con el bit restante? Usando los ocho dígitos, se puede detectar un error individual de transmisión. Por ejemplo, los códigos ASCII para A, B, y C son:

A = 6510 = 010000012

B = 6610 = 010000102

C = 6710 = 010000112

El bit de más a la izquierda siempre es 0; es decir, los 128 caracteres ASCII tienen códigos:

000000002 = 010

⋮

011111112 = 12710

El bit puede ser usado para controlar errores en los otros siete bits. Se pone como 0 o 1 de manera que el número total de bits 1 en la representación del caracter sea par. Usando este concepto de paridad, los códigos para A, B, y C se convierten en

A = 010000012

B = 010000102

C = 110000112

Se supone que se envía una A y ocurre un error de transmisión en el sexto bit de manera que se recibe (01000101). Se sabe que se produjo un error pues se recibió un número impar de unos, y se puede pedir que la palabra sea retransmitida. Cuando se usa para detectar errores, el bit de más a la izquierda se llama bit de control de paridad.

Para verificar que los códigos de paridad son detectores de error en un dígito binario basta con probar que su distancia mínima es dos, para lo que se demostrarán las siguientes dos proposiciones:

a) Un código de paridad par posee una distancia mínima estrictamente mayor que uno.

Se razona por reducción al absurdo. Se supone que existen dos combinaciones bk, bk-1, ..., b1, b0 y ak, ak-1, ..., a1, a0, con una distancia entre ellas de uno. Esto implica que existe un bit, por ejemplo el j-ésimo, que hace que:

De aquí se infiere que si la primera combinación posee m unos, la segunda tiene m + 1 o m - 1. En cualquier caso si m es par m + 1 o m - 1 son números impares, con los cual la segunda combinación no pertenecería al código. Este razonamiento lleva a un absurdo, con lo que queda demostrada la proposición.

b) En un código de paridad par existen siempre dos combinaciones que distan dos unidades.

Como el código de partida es de distancia mínima uno, existirán al menos dos subcombinaciones bk-1, ..., b1, b0 y ak-1, ..., a1, a0, en las que solamente un índice j hace que:

Si la primera combinación posee m unos, la segunda tendrá m + 1 o m - 1. Se supone que m es par, entonces los respectivos bits de paridad serán bk = 0 y ak = 1. Si m fuese impar, los bits de paridad serían bk = 1 y ak = 0. En cualquier caso se cumple que bk = ak. Por lo tanto, la distancia de estas dos combinaciones resultante es dos (bk = ak y bj = aj).

En esta demostración se ha trabajado con códigos de paridad par. El razonamiento es el mismo si fuese de paridad impar.

### Código Bose – Chaudhuri – Hocquenghem

O mejor conocido como BCH, forma una clase de códigos cíclicos de corrección de errores que se construyen usando polinomios sobre un campo finito (también llamado campo de Galois).

Una de las características clave de los códigos BCH es que, durante el diseño del código, existe un control preciso sobre la cantidad de errores de símbolo que el código puede corregir. En particular, es posible diseñar códigos BCH binarios que puedan corregir múltiples errores de bits. Otra ventaja de los códigos BCH es la facilidad con la que pueden decodificarse, es decir, a través de un método algebraico conocido como decodificación de síndromes. Esto simplifica el diseño del decodificador de estos códigos, utilizando un pequeño hardware electrónico de bajo consumo.

Claro, el código BCH (Bose-Chaudhuri-Hocquenghem) es un tipo de código corrector de errores cíclico que se utiliza para detectar y corregir múltiples errores en bloques de datos.

##### Características

Corrección de errores: Los códigos BCH pueden ser diseñados para corregir un número específico de errores t. Esto se indica en la notación BCH(n, k, t), donde:

- n es la longitud del código (número de bits en el bloque codificado).

- k es la longitud del mensaje original (número de bits de datos antes de la codificación).

- t es el número de errores que el código puede corregir.

Generador del código: El código BCH se define mediante un polinomio generador, que se utiliza para generar las palabras de código. Este polinomio es producto de ciertos polinomios mínimos en el campo de Galois.

##### Construcción

Debido a que cualquier polinomio que sea un múltiplo del polinomio generador es una palabra de código BCH válida, la codificación BCH es simplemente el proceso de encontrar algún polinomio que tenga el generador como factor.

El código BCH en sí mismo no es prescriptivo sobre el significado de los coeficientes del polinomio; conceptualmente, la única preocupación de un algoritmo de decodificación BCH es encontrar la palabra de código válida con la distancia mínima de Hamming a la palabra de código recibida. Por lo tanto, el código BCH puede implementarse como un código sistemático o no, dependiendo de cómo el implementador elija incorporar el mensaje en el polinomio codificado.

###### Codificación no sistemática

La forma más sencilla de encontrar un polinomio que sea un múltiplo del generador es calcular el producto de algún polinomio arbitrario y el generador. En este caso, el polinomio arbitrario se puede elegir utilizando los símbolos del mensaje como coeficientes.

Como ejemplo, considere el polinomio generador g(x)=x10+x9+x8+x6+x5+x3+1, elegido para su uso en el código BCH binario (31, 21). Para codificar el mensaje de 21 bits {101110111111101}, primero se lo representa como un polinomio sobre GF(2):

Entonces, resolviendo:

Por lo tanto, la palabra clave transmitida es {1100111010010111101011101110101}.

###### Codificación no sistemática

Un código sistemático es uno en el que el mensaje aparece literal en algún lugar dentro de la palabra clave. Por lo tanto, la codificación sistemática de BCH implica primero incrustar el mensaje polinomio dentro del polinomio de la palabra clave, y luego ajustar los coeficientes de los términos restantes (no mensaje) para asegurar que s(x) sea divisible por g(x).

Este método de codificación aprovecha el hecho de que restar el resto de un dividendo resulta en un múltiplo del divisor. Por lo tanto, si se toma nuestro mensaje polinomio p(x) como antes y se multiplica por xn-k (evitando que se mezcle con el resto durante el proceso de división), se puede entonces utilizar la división Euclidiana de los polinomios para producir:

Se observa que q(x).g(x) es una palabra clave válida. Como r(x) es siempre de grado menos que n-k (que es el grado de g(x)), se puede restarla de forma segura p(x)xn-k sin alterar ninguno de los coeficientes de mensaje, por lo tanto:

La ventaja a la codificación sistemática es que el receptor puede recuperar el mensaje original descartando todo después de los primeros k coeficientes, después de realizar la corrección de errores.

##### Decodificación

Hay muchos algoritmos para decodificar códigos BCH. Los más comunes siguen este esquema general:

1. Calcular los síndromes sj para el vector recibido
2. Determinar el número de errores t y el polinomio localizador de errores (x) de los síndromes
3. Calcular las raíces de la ubicación de error polinomial para encontrar las ubicaciones de error Xi
4. Calcular los valores de error Yi en esas ubicaciones de errores
5. Corregir los errores

Durante algunos de estos pasos, el algoritmo de decodificación puede determinar que el vector recibido tiene demasiados errores y no se puede corregir. Por ejemplo, si no se encuentra un valor apropiado de t, la corrección fallará. En un código truncado (no primitivo), la ubicación de un error puede estar fuera de rango. Si el vector recibido tiene más errores de los que el código puede corregir, el decodificador puede producir, sin saberlo, un mensaje aparentemente válido que no es el que se envió.

###### Cálculo de los síndromes

El vector recibido R es la suma de la palabra clave correcta C y un vector de error desconocido E. Los valores del síndrome se forman considerando R como un polinomio y evaluarlo en αc, …, αc+d-2. Así los síndromes son:

para j = c, …, c+d-2.

Desde αj son los ceros de g(x), de los cuales C(x) es un múltiple, C(αj) = 0.

Si no hay error, sj = 0 para todos j. Si los síndromes son todos cero, entonces se hace la decodificación.

###### Cálculo del polinomio de ubicación del error

Si hay síndromes distintos de cero, entonces hay errores. El decodificador necesita averiguar cuántos errores y la ubicación de esos errores.

Si hay un solo error, escriba esto como E(x)=e-xi, donde i es la ubicación del error y e es su magnitud. Entonces los primeros dos síndromes son:

Así, juntos, permiten el cálculo de e y proporcionar información sobre i (determinándolo completamente en el caso de los códigos Reed-Solomon).

Si hay dos o más errores:

No es inmediatamente obvio cómo comenzar a resolver los síndromes resultantes para los desconocidos ek e ik.

El primer paso es encontrar, compatible con los síndromes computados y con mínimo posible t, localizador polinomio:

Texto

Descripción generada automáticamente

Ilustración 6

Tres algoritmos populares para esta tarea son:

* Peterson–Gorenstein–Zierler algoritmo
* algoritmo de Berlekamp-Massey
* Sugiyama Euclidean algoritmo

###### Factor de polinomio localizador de errores

Ahora que se tiene el Λ(x) polinomio, sus raíces se pueden encontrar en la forma Λ(x) = (αi1 . x - 1) . (αi2 . x - 1) . … . (αiv .x - 1) por fuerza bruta, por ejemplo, usando el algoritmo de búsqueda Chien. El exponencial poder del elemento primitivo α cederá las posiciones en las que ocurren errores en la palabra recibida.

Los ceros de Λ(x) son α−i1, …, α−iv.

###### Calcular valores de error

Una vez que se conocen las ubicaciones de error, el siguiente paso es determinar los valores de error en esas ubicaciones. Los valores de error se utilizan luego para corregir los valores recibidos en esas ubicaciones para recuperar la palabra de código original.

Para el caso de BCH binario, (con todos los caracteres legibles) esto es trivial; simplemente voltear los bits para la palabra recibida en estas posiciones, y se tiene la palabra clave corregida. En el caso más general, el peso del error ej puede determinarse resolviendo el sistema lineal

Imagen que contiene Diagrama

Descripción generada automáticamente

Ilustración 7

##### Resumen

Los códigos BCH son potentes códigos cíclicos diseñados para corregir múltiples errores en bloques de datos. Utilizan la aritmética en campos de Galois y polinomios generadores específicos para proporcionar una capacidad de corrección de errores robusta. Se aplican en diversas aplicaciones, desde sistemas de comunicación hasta almacenamiento de datos, debido a su eficiencia y eficacia en la corrección de errores.

#### Códigos BCH primitivos de sentido estricto

Dado un número primo *q* y una potencia prima *q* *m* con enteros positivos *m* y *d* tal que *d* ≤ qm − 1, un código BCH primitivo de sentido estricto sobre el campo finito (o campo de Galois) GF(*q*) con longitud de código *n* = *q*m − 1 y la distancia al menos *d* se construye con el siguiente método.

Sea *α* un elemento primitivo de GF(*q*m). Para cualquier entero positivo *i*, sea *mi*(*x*) sea el polinomio mínimo con coeficientes en GF(*q*) de αi. El polinomio generador del código BCH se define como el mínimo común múltiplo (o como sus siglas en inglés: lcm) *g*(*x*) = lcm(*m*1(*x*), …, md − 1(*x*)). Se puede ver que *g*(*x*) es un polinomio con coeficientes en GF (*q*) y divide *x*n − 1. Por lo tanto, el código polinomial definido por *g*(*x*) es un código cíclico.

##### Ejemplo

Sean q = 2 y m = 4 (por lo tanto n = 15). Se considera diferentes valores de d para GF(16) = GF(24) basado en el polinomio reductor z4 + z + 1 , utilizando el elemento primitivo α(z) = z. Hay catorce polinomios mínimos mi(x) con coeficientes en GF(2) que satisfacen:

Los polinomios mínimos son:

a) Entonces, un código BCH con d = 2 (o 3), crea un polinomio generador:

Tiene una distancia mínima de Hamming de hasta 3 y corrige hasta un error. Dado que el polinomio generador es de grado 4, este código tiene 11 bits de datos y 4 bits de suma de comprobación.

b) Un código BCH con d = 8:

Este código tiene una distancia Hamming mínima de 15 y corrige 7 errores. Tiene 1 bit de datos y 14 bits de suma de comprobación.

#### Códigos BCH generales

Los códigos BCH generales difieren de los códigos BCH primitivos de sentido estricto en dos aspectos:

Primero, el requisito de que α sea un elemento primitivo GF(qm) se puede relajar. Al relajar este requisito, la longitud del código cambia de qm-1 a el orden del elemento α.

En segundo lugar, las raíces consecutivas del polinomio generador pueden correr desde αc, …, αc+d-2 en lugar de α, …, αd-1.

### Código de repetición

Una de las ideas más simple para detectar errores en una transmisión es repetir cada símbolo n veces, o repetir cada secuencia de k símbolos n veces. Si los símbolos fuente son 0 y 1, se pueden codificar de la forma siguiente:



Ilustración 8

Obteniendo un código bloque binario de longitud n llamado código de repetición de tamaño n.

Suponiendo que en la transmisión se han producido errores, recibida una palabra, basta con contar el número de 0 y 1 que contiene la palabra recibida y decidir, por un simple criterio de mayoría, su decodificación como la palabra emitida más probable. Esto solo es posible sin ambigüedades si n es impar. Aunque en el modelo presentado se ha definido un código de repetición 1 a n, se puede generalizar a uno del tipo k a kn repitiendo n veces la secuencia de longitud k.

Con esta estrategia se pueden detectar hasta n − 1 errores, es decir uno menos que la longitud de repetición. Para corregir estos errores producidos en la transmisión, suponiendo que la probabilidad de error no es muy grande, entonces es lógico pensar que la palabra enviada ha sido la que coincide con la recibida en más posiciones. Por ejemplo, con n = 7, si se recibe y = 1000100 es más lógico pensar que se ha deseado enviar la palabra x = 0000000, cometiéndose dos errores, que suponer que la palabra enviada es x = 1111111 donde se hubieran cometido 5 errores. De esta forma se puede decir que el código corrige hasta b(n − 1)/2c errores. Esta idea es la que después se generalizara para la decodificación por mínima distancia.

#### Ejemplo

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 9

La ilustración superior representa el resultado de modular directamente el flujo de bits b (n) en una señal transmitida x (t) usando un conjunto de señales BPSK de banda base. R' es el dato producido por el codificador fuente. Si ese flujo de bits pasa a través de un codificador de canal (3,1) para producir el flujo de bits c (l), la señal transmitida resultante requiere un intervalo de bits T tres veces menor que la versión no codificada. Esta reducción en el intervalo de bits significa que la energía/bit transmitido disminuye en un factor de tres, lo que da como resultado una mayor probabilidad de error en el receptor.

Se puede ver que el transmisor envía el bit de datos varias veces, un número impar de veces de hecho. Debido a que la probabilidad de error pe es siempre menor que 1/2, se sabe que más de los bits deben ser correctos en lugar de errores. La votación por mayoría simple de los bits recibidos (de ahí la razón del número impar) determina el bit transmitido con mayor precisión que enviarlo solo. Por ejemplo, se considera el código de repetición triple: por cada bit b (n) que emerge del codificador fuente, el codificador de canal produce tres. Por lo tanto, el flujo de bits que emerge del codificador de canal c (l) tiene una velocidad de datos tres veces mayor que la del flujo de bits original b (n).

La tabla de codificación ilustra cuándo se pueden corregir los errores y cuándo no pueden hacerlo mediante el decodificador de mayoría de votos.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Ilustración 10

Así, si un bit de los tres bits se recibe por error, el receptor puede corregir el error; si se produce más de un error, el decodificador de canal anuncia que el bit es 1 en lugar del valor transmitido de 0. Usando este código de repetición, la probabilidad de b(n)≠0 es igual 3p2e×(1−pe) + p3e.

Esta probabilidad de un error de decodificación es siempre menor que pe, el valor no codificado, siempre y cuando: pe < ½.

Un buen código será aquel que permita corregir la mayor cantidad posible de errores introduciendo la menor cantidad de símbolos redundantes. Como se ha comprobado con los códigos de repetición, estos dos requerimientos en general se contraponen y se debe buscar un equilibrio entre ambos.

## Códigos convolucionales

Los códigos convolucionales son códigos lineales al igual que los códigos bloque y, por tanto, se utilizan para proteger la información añadiendo redundancia a la misma, de manera que las palabras del código tengan la distancia mínima necesaria.

Sin embargo, a diferencia de los códigos bloque, las palabras de un código convolucional se generan no sólo a partir de los dígitos de información actuales sino también con la información anterior en el tiempo. Es decir, un codificador convolucional es un sistema con memoria y, en consecuencia, lleva asociada una cadena de Markov aunque ésta no es visible en la salida pero sí la condiciona.

Un código convolucional queda especificado por tres parámetros:

* Números de entradas k (Bits de información)
* Números de salidas n (Secuencia codificada de información, bits de información + bits de redundancia.)
* Redundancia (registro) m
* Memoria de código m-1

La tasa del código se representa con la siguiente fórmula:

A diagram of a algorithm

Description automatically generated

Ilustración 11

Según se puede apreciar, un codificador convolucional está formado por un registro de desplazamiento de L·k bits, donde L es la longitud limitada u obligada del codificador, que se conecta a tantos sumadores módulo 2 como bits de salida se deseen. La respuesta del codificador queda determinada por la manera en que cada uno de los n sumadores calcula los respectivos bits de salida.

También se observa que un codificador convolucional memoriza las L-1 secuencias de k bits anteriores a la secuencia actual. Por tanto, L-1 representa el orden de memoria del codificador. Asimismo, al igual que en los códigos bloque, la relación k/n es la tasa o cadencia del codificador. En resumen, todo código convolucional queda definido por la terna [n, k, L].

#### Decodificación

Existen varios [algoritmos](https://en.wikipedia.org/wiki/Algorithm) para decodificar códigos convolucionales. Para valores relativamente pequeños de k , el [algoritmo de Viterbi](https://en.wikipedia.org/wiki/Viterbi_algorithm) se utiliza universalmente ya que proporciona un rendimiento [de máxima probabilidad](https://en.wikipedia.org/wiki/Maximum_likelihood) y es altamente paralelizable. Por tanto, los decodificadores Viterbi son fáciles de implementar en hardware [VLSI](https://en.wikipedia.org/wiki/VLSI) y en software en CPU con conjuntos de instrucciones [SIMD](https://en.wikipedia.org/wiki/SIMD) .

Los códigos de longitud de restricción más larga se decodifican de manera más práctica con cualquiera de varios algoritmos [de decodificación secuencial](https://en.wikipedia.org/wiki/Sequential_decoding) , de los cuales el algoritmo [de Fano](https://en.wikipedia.org/wiki/Robert_Fano) es el más conocido. A diferencia de la decodificación de Viterbi, la decodificación secuencial no es de máxima probabilidad, pero su complejidad aumenta sólo ligeramente con la longitud de la restricción, lo que permite el uso de códigos fuertes y con una longitud de restricción larga.

Tanto los algoritmos de Viterbi como los de decodificación secuencial devuelven decisiones difíciles: los bits que forman la palabra clave más probable. Se puede agregar una medida de confianza aproximada a cada bit mediante el uso del [algoritmo de Viterbi de salida suave](https://en.wikipedia.org/wiki/Soft_output_Viterbi_algorithm) . Las decisiones suaves [máximas a posteriori](https://en.wikipedia.org/wiki/Maximum_a_posteriori) (MAP) para cada bit se pueden obtener mediante el uso del [algoritmo BCJR](https://en.wikipedia.org/wiki/BCJR_algorithm) .

### Código turbo

Los turbo códigos son una clase de códigos de [corrección de errores](https://es.wikipedia.org/wiki/Correcci%C3%B3n_de_errores), que se introdujeron, junto con un algoritmo de decodificación. La importancia de los turbo códigos es que permiten una comunicación fiable y su [eficiencia energética](https://es.wikipedia.org/wiki/Eficiencia_energ%C3%A9tica) está muy cerca del límite teórico predicho por [Shannon](https://es.wikipedia.org/wiki/Teorema_de_Shannon-Hartley). Desde su introducción, los turbo códigos se han utilizado en aplicaciones de baja potencia, como las [comunicaciones por satélite](https://es.wikipedia.org/wiki/Comunicaciones_por_sat%C3%A9lite), así como para aplicaciones de interferencia limitada, como los servicios de tercera generación ([3G](https://es.wikipedia.org/wiki/3G)) de comunicaciones móviles.

#### Codificador

El codificador Turbo consiste en la concatenación de dos códigos continuos recursivos y sistemáticos, denominados codificadores constituyentes, que operan uno sobre la información a codificar (la parte sistemática) y el otro sobre una versión permutada (entre lazada) de la misma, tal y como muestra la figura

A diagram of a computer code

Description automatically generated with medium confidence

Ilustración 12

El codificador turbo consta de dos codificadores simples generalmente idénticos

– El intercalador hace que los bits de entrada al codificador 2 parezcan “diferentes” a los de la entrada del codificador 1

– La perforación y la multiplexación logran la velocidad combinada requerida

#### Ejemplos

Código convolucional sistemático recursivo de velocidad media RSC CC(2, 1, 2)

– Polinomio generador de avance 

– Polinomio generador de retroalimentación A black and blue symbol

Description automatically generated with medium confidence

– Un RSC CC(2, 1, 2) no es potente, pero con dos paralelos concatenados, ⇒ código turbo muy potente

A diagram of a block diagram

Description automatically generated

Ilustración 13

• Dos decodificadores concatenados en paralelo,

– cada uno acepta salida de canal suave e información del otro decodificador como LLR a priori

– y genera LLR a posteriori

– Los LLR están correctamente entrelazados o desintercalado

A diagram of a decoder

Description automatically generated

Ilustración 14

#### Decodificación

Todo algoritmo de decodificación turbo consiste en obtener iterativamente la siguiente métrica para cada bit de información:

A mathematical equation with numbers and symbols

Description automatically generated

Ilustración 15

Esta métrica se denomina Log-Likelihood Ratio y es novedosa en el sentido que no está basada en el criterio ML ('Maximum Likelihood') como el algoritmo de Viterbi, que es el que tradicionalmente se ha usado para la decodificación de los códigos continuos. Este cambio fue necesario debido a la complejidad de la aplicación de este criterio a los códigos Turbo (se describirían con un enrejado de tamaño prohibitivo), que emplean el algoritmo BCJR y sus modificaciones de menor complejidad, Log-MAP y max-Log-MAP, para aplicar el MAP ('Maximum A Posteriori').

La decisión a partir del LLR se basa en su signo. Cuando es positivo se decide 1 y 0 en caso contrario. Este modo de proceder constituye lo que se denomina una decodificación soft porque el valor absoluto del LLR es una medida de la confianza en la decisión. Además, cuantas más iteraciones se realicen para calcular esta métrica con más precisión se obtendrá su valor real.

#### Aplicaciones

Los turbocódigos se usan en los sistemas de telecomunicaciones, algunos ejemplos son:

* En [comunicaciones por satélite](https://es.wikipedia.org/wiki/Comunicaciones_por_sat%C3%A9lite) y espaciales
* En la [televisión digital](https://es.wikipedia.org/wiki/Televisi%C3%B3n_digital), por ejemplo en: [DVB-RCS](https://es.wikipedia.org/wiki/DVB-RCS) , [DVB-SH](https://es.wikipedia.org/wiki/DVB-SH) , [DVB-S2](https://es.wikipedia.org/wiki/DVB-S2)
* En las [comunicaciones por fibra óptica](https://es.wikipedia.org/wiki/Comunicaciones_por_fibra_%C3%B3ptica)
* En [comunicaciones inalámbricas](https://es.wikipedia.org/wiki/Comunicaciones_inal%C3%A1mbricas) (wireless)
* En sistemas de grabación ópticos
* En los módems [ADSL](https://es.wikipedia.org/wiki/ADSL)
* En [telemetría](https://es.wikipedia.org/wiki/Telemetr%C3%ADa)

### Código de Hadamard

El código de Hadamard (Reed y Solomon, 1960) es un código de corrección de errores que se utiliza para la detección y corrección de errores cuando se transmiten mensajes por canales muy ruidosos o poco ﬁables. Son un caso particular de los códigos Reed-Solomon (Wicker y Bhargava, 1999), corresponden al primer código Reed Solomon. La idea principal de estos códigos es incrementar la distancia de Hamming entre palabras. Una palabra es una cadena ﬁnita de bits que son manejados como un conjunto por una computadora.

Para construir los códigos de Hadamard en un problema de clasiﬁcación se usa la siguiente regla: Se supone la necesidad de códigos de longitud 2k, en donde hay como máximo 2k clases posibles, es decir, en caso de querer entrenar una red con el conjunto de datos de ImageNet que cuenta con 1,000 clases, se tendrá un valor de k=10, obteniendo así una longitud de 1,024. Una de las ventajas de utilizar estos códigos de Hadamard es que la distancia de Hamming entre vectores de diferentes clases es m/2, siendo m el número de clases. Esto quiere decir, que los vectores que pertenecen a una misma clase tendrán una distancia de Hamming pequeña. Mientras que vectores de clases diferentes tendrán una distancia de Hamming grande.

En la siguiente tabla se muestra un ejemplo de etiquetado y codiﬁcación Hadamard en la base de datos de imágenes de dígitos manuscritos (MNIST) (Deng, 2012). Se puede ver que el conjunto de datos MNIST se divide en 10 clases, por lo cual, el vector en codiﬁcación de Hadamard tendrá una longitud de 16 bits.

Tabla

Descripción generada automáticamente

Ilustración 16

#### Ejemplo

Si se tiene un conjunto de datos con 1,000 clases, el código de Hadamard tendrá una longitud de 1,024, por la regla de 2k. Al igual que en la obtención de las características profundas, se debe eliminar o remover la última capa que por lo general, corresponde a la etapa de clasiﬁcación. Ahora, la salida de nuestro modelo será un vector de números reales con dimensión igual al número de clases asignado. Este vector será decodiﬁcado como en un canal de transmisión de mensajes. Para decodiﬁcar el vector y convertir a codiﬁcación de Hadamard, se deberá agregar una capa al ﬁnal del modelo con la función de activación HardTanh. Esta función se encuentra deﬁnida por la Ecuación 1 y se aplica a cada elemento del vector. Sin embargo, el vector que se obtendrá como salida de nuestro modelo tendrá valores de -1 y 1. Para convertirlo a binario, se deberá recorrer el vector y cambiar por 0, todo aquel valor que sea menor o igual a 0. Con los cambios anteriores, cada vez que se procese una imagen en nuestro modelo, se obtendrá un vector binario en codiﬁcación de Hadamard.

Imagen que contiene Diagrama

Descripción generada automáticamente

Ilustración 17

En general, los cambios que se deben realizar en un modelo para obtener vectores binarios en codiﬁcación de Hadamard son dos, los cuales son: (1) reemplazar el número de clases correspondiente con el valor resultante al calcular 2k, y (2) agregar una capa con la función de activación HardTanh. Además, fuera del modelo, será necesario recorrer el vector obtenido como salida del modelo para cambiar los valores menores a cero por ceros. Así se obtendrá vectores binarios en codiﬁcación de Hadamard de una imagen.

#### Aplicación

La sonda Mariner 4 fue la primera en tomar fotografías de otro planeta. En 1965 tomo 22 fotografías completas de Marte en blanco y negro, cada una de 200 × 200 pixeles. A cada píxel se le asignó una 6-upla representando uno de 64 tonos de gris según el brillo, desde el blanco 000000 hasta el negro 111111. Se utilizó un código binario de Hadamard con parámetros [32, 6, 7] y la transformada rápida de Fourier (FFT) para la decodificación. El número total de bits (dígitos binarios) por fotografía es de 240.000. La transmisión se llevó a cabo a un ritmo de 8 1 3 bits por segundo, por lo que llevó exactamente 8 horas transmitir cada foto.

## Códigos FEC

FEC son siglas que en inglés significan Forward Error Correction. Los códigos pertenecientes a esta clasificación son un método eficaz de procesamiento de señales digitales que mejora la tasa de error de bits de los enlaces de comunicación agregando información redundante (bits de paridad) a los datos en el lado del transmisor para que el lado del receptor utilice la información redundante para detectar y corregir errores que puedan tener. Se lo introduce en el enlace de transmisión. La codificación de la señal que tiene lugar en el transmisor debe ser decodificada correctamente por el receptor para poder extraer la información de la señal original.

Los códigos FEC no requieren protocolo de enlace entre el origen y el destino, lo que significa que los dos sistemas no necesitan establecer una conexión antes de que se puedan transmitir los datos. Esto hace posible transmitir datos a múltiples destinos simultáneamente desde una única fuente. Además, la capacidad de estos códigos para corregir errores presenta la ventaja de que es necesario retransmitir menos datos, lo que reduce el uso del ancho de banda y el consumo de energía.

Debido a sus capacidades de corrección de errores, FEC se implementa comúnmente cuando se transmiten datos a través de canales de comunicación ruidosos o poco confiables, lo que lo hace adecuado tanto para comunicaciones por cable como inalámbricas. Por ejemplo, FEC proporciona una solución eficiente para transmitir contenido de video o admitir redes celulares. También puede ser utilizado por módems y enrutadores, admitir comunicaciones ópticas y por cable y facilitar transmisiones de datos unidireccionales.

##### Funcionamiento

En la forma más simple de FEC, el transmisor envía cada carácter varias veces para brindar protección contra la pérdida o corrupción de datos. El receptor compara las instancias de los caracteres y, si hay discrepancias, utiliza un sistema de "reglas mayoritarias" para recuperar los datos. Por ejemplo, los datos transmitidos podrían incluir la letra mayúscula W, que tiene un valor binario de 01010111. Si el transmisor enviara el byte W tres veces, el receptor compararía los bits en cada instancia de byte para determinar qué bits son precisos. Como se muestra en la siguiente ilustración.

Diagrama

Descripción generada automáticamente

Ilustración 18

En este caso, hay dos errores en los datos cuando llegan al receptor. El quinto bit en la segunda instancia de carácter es diferente de los otros dos bits en esa posición, y el segundo bit en la tercera instancia es diferente de los otros dos en su posición. Para cada posición de bit, el receptor compara los bits. Si los tres son iguales, ese valor se considera el bit correcto. Si un valor es diferente de los otros dos, los dos valores mayoritarios se consideran el bit correcto.

El proceso exacto utilizado para las comunicaciones basadas en FEC varía de un sistema a otro. Sin embargo, a un nivel alto, generalmente siguen un enfoque similar. El transmisor usa algún tipo de codificador para agregar los datos de paridad necesarios a los datos originales, y el receptor usa algún tipo de decodificador para extraer los datos originales de los datos transmitidos mientras corrige errores en el proceso. La ilustración siguiente proporciona una descripción general simplificada de cómo funciona.

Diagrama

Descripción generada automáticamente

Ilustración 19

Las organizaciones que implementan FEC deben utilizar el enfoque que mejor se adapte a sus circunstancias. Sin embargo, deben tener en cuenta una pauta importante: el transmisor y el receptor deben utilizar el mismo tipo de FEC y seguir las mismas reglas al codificar y decodificar datos. Sólo entonces podrán estar seguros de que el decodificador está identificando y corrigiendo correctamente los errores.

Se requiere una definición e implementación precisas de las reglas de codificación para evitar una mala interpretación de la información por parte del receptor que decodifica la señal. La interoperabilidad exitosa sólo se producirá cuando tanto el transmisor como el receptor sigan e implementen las mismas reglas de codificación y decodificación.

Diagrama

Descripción generada automáticamente

Ilustración 20

Como se puede ver, FEC es el elemento esencial que debe definirse para permitir el desarrollo de transceptores interoperables utilizando tecnología óptica sobre enlaces punto a punto.

Al considerar qué FEC elegir para una nueva especificación, es necesario considerar algunas métricas clave, incluidas las siguientes:

* Tasa de sobrecarga de codificación: la relación entre el número de bits redundantes y los bits de información.
* Ganancia de codificación neta (NCG): la mejora de la sensibilidad óptica recibida con y sin uso de FEC asociada con el aumento de la velocidad de bits.
* Umbral BER previo a la FEC: un umbral predefinido para una transmisión posterior a la FEC sin errores determinado por NCG.

A continuación se observarán dos tipos de códigos, el Reed-Solomon y el LDPC, que son códigos FEC, pero con modificaciones que los vuelven cíclico y convolucional respectivamente.

### Código Reed-Solomon

Un código Reed-Solomon se especifica como RS(n, k) con símbolos de s bits. Lo anterior significa que el codificador toma k símbolos de los s bit y añade símbolos de paridad para hacer una palabra de código de n símbolos. Existen n - k símbolos de paridad de s bits cada uno. Un decodificador puede corregir hasta t símbolos que contienen errores en una palabra de código, donde 2t = n - k.

El siguiente diagrama muestra una típica palabra de código Reed-Solomon (este se conoce como un código sistemático puesto que los datos se dejan inalterados y los símbolos de paridad se anexan):

#### Ejemplo

Un código popular Reed-Solomon es RS(255, 223) con símbolos de 8 bits. Cada palabra de código contiene 255 bytes de palabra de código, de los cuales 223 bytes son datos y 32 bytes son paridad. Para este código se tiene:

* N = 255, k = 223, s = 8
* 2t = 32, t = 16

El decodificador puede corregir cualquier error de 16 símbolos en la palabra de código, es decir, errores de hasta 16 [bytes](https://es.wikipedia.org/wiki/Byte) en cualquier lugar de la palabra pueden ser automáticamente corregidos.

Un error de símbolo ocurre cuando al menos un bit de un símbolo es erróneo, un RS(255, 223) puede corregir 16 errores de símbolos. En el peor caso, errores de 16 bits pueden ocurrir, cada uno en un símbolo distinto (byte) de forma que el decodificador corrige errores de 16 bits. En el mejor caso, 16 errores de byte completos ocurren de tal forma que el decodificador corrige 16x8 errores de bit. Los códigos Reed-Solomon son particularmente útiles para corregir burst error (cuando una serie de bits en el código de palabra se reciben con error).

Dado un tamaño de símbolo s, la máxima longitud de la palabra de código (n) para un código Reed-Solomon es n = 2s - 1. Por ejemplo, la máxima longitud de un código con símbolos de 8 bits (s = 8) es de 255 bytes.

Los códigos Reed-Solomon pueden ser acortados haciendo un número de símbolos de datos igual a cero en el codificador, no transmitiendo estos, y reinsertando éstos en el decodificador. Por ejemplo, el código (255, 223) descrito anteriormente puede ser acortado a (200, 168). El codificador toma un bloque de 168 bytes de datos añade 55 bytes cero, crea una palabra de código de (255, 223) y transmite solo los 168 bytes de datos y 32 bytes de paridad.

La cantidad de poder de procesamiento para codificar y decodificar códigos Reed-Solomon se relaciona con el número de símbolos de paridad por palabra de código. Un valor grande de t significa que un gran número de errores pueden ser corregidos pero requiere mayor poder computacional que un valor pequeño de t.

#### Decodificación

Los procedimientos algebraicos de decodificación de Reed-Solomon pueden corregir errores y datos perdidos. Un "borrado" ocurre cuando la posición de un símbolo errado es conocido. Un decodificador puede corregir hasta t errores o hasta 2t "borrados".

La información sobre los "borrados" puede ser frecuentemente otorgada por el demodulador en un sistema de comunicación digital, es decir, el demodulador "marca" los símbolos recibidos que con probabilidad contienen errores.

Cuando una palabra de código es decodificada, existen tres posibilidades: si 2s + r < 2t (s errores, r "borrados") entonces la palabra de código original transmitida puede ser siempre recuperada; sino el decodificador detectará que no puede recuperar la palabra de código original e indicará este hecho, o l decodificador decodificará erróneamente y recuperará una palabra de código incorrecta sin indicación.

### Código LDPC

Los códigos LDPC (Low-Density Parity-Check) son los códigos correctores cuyas prestaciones son las que más se acercan a los límites teóricos que estableció Shannon. Debido a esta propiedad, se emplean en una gran variedad de sistemas de comunicaciones. La característica más importante de estos códigos radica en que sus matrices de chequeo de paridad son de baja densidad, es decir, tienen pocos elementos distintos de 0 y por esta razón, tanto la codificación como la decodificación mediante ellos es muy eficiente en términos de complejidad computacional. Existen diversos métodos de decodificación para estos códigos, pero el más utilizado es el algoritmo belief propagation, también conocido como suma-producto, que se basa en el intercambio de información probabilística estimada para cada bit en cada fila de chequeo de paridad de H.

#### Funcionamiento

Los Códigos LDPC utilizan una matriz de paridad de baja densidad para la codificación y decodificación. Esta matriz se caracteriza por tener un número significativamente bajo de unos en comparación con los ceros. Al usar una matriz de baja densidad, los códigos LDPC pueden minimizar la cantidad de redundancia en los datos transmitidos, permitiendo así una mayor eficiencia de la transmisión.

1. **Creación de Matriz:** La matriz de paridad se genera utilizando una serie de algoritmos, que garantizan que sea de baja densidad y tenga ciertas propiedades que permiten la corrección de errores.
2. **Transmisión de Datos:** Los datos se transmiten junto con la información de paridad generada por la matriz. Esta redundancia permite al receptor detectar y corregir errores en los datos transmitidos.
3. **Decodificación:** Al recibir los datos, el receptor utiliza la matriz de paridad y los algoritmos de decodificación para detectar y corregir cualquier error en la transmisión.

Los códigos LDPC son particularmente útiles en entornos con altas tasas de error, como las comunicaciones inalámbricas y por satélite. Son capaces de corregir muchos errores sin necesidad de retransmitir los datos, lo que los hace especialmente valiosos en situaciones donde la retransmisión no es práctica o posible.

#### Ejemplo

A continuación se muestra un fragmento de gráfico de un código LDPC de ejemplo que utiliza [la notación de gráfico de factores de Forney](https://en.wikipedia.org/wiki/Factor_graph) . En este gráfico, *n* nodos variables en la parte superior del gráfico están conectados a ( *n* − *k* ) nodos de restricción en la parte inferior del gráfico.

Esta es una forma popular de representar gráficamente un código LDPC (*n*, *k* ). Los bits de un mensaje válido, cuando se colocan en las T en la parte superior del gráfico, satisfacen las restricciones gráficas. Específicamente, todas las líneas que se conectan a un nodo variable (cuadro con un signo '=') tienen el mismo valor, y todos los valores que se conectan a un nodo factor (cuadro con un signo '+') deben sumar, [módulo](https://en.wikipedia.org/wiki/Modular_arithmetic) dos, cero (en otras palabras, deben sumar un número par o debe haber un número par de valores impares).

[A black and white background with white squares and a plus

Description automatically generated](https://en.wikipedia.org/wiki/File:Ldpc_code_fragment_factor_graph.svg)

Ilustración 21

Ignorando cualquier línea que salga de la imagen, hay ocho posibles cadenas de seis bits correspondientes a palabras de código válidas: (es decir, 000000, 011001, 110010, 101011, 111100, 100101, 001110, 010111). Este fragmento de código LDPC representa un mensaje de tres bits codificado como seis bits. Aquí se utiliza la redundancia para aumentar las posibilidades de recuperación de errores de canal. Este es un [código lineal](https://en.wikipedia.org/wiki/Linear_code) (6, 3) , con *n* = 6 y *k* = 3.

Nuevamente, ignorando las líneas que salen de la imagen, la matriz de verificación de paridad que representa este fragmento de gráfico es

A number of black numbers

Description automatically generated with medium confidence

Ilustración 22

En esta matriz, cada fila representa una de las tres restricciones de verificación de paridad, mientras que cada columna representa uno de los seis bits de la palabra de código recibida.

En este ejemplo, las ocho palabras de código se pueden obtener colocando la [matriz de verificación de paridad](https://en.wikipedia.org/wiki/Parity-check_matrix)H en esta forma [a través de operaciones](https://en.wikipedia.org/wiki/Row_operations) básicas de fila en [GF(2)](https://en.wikipedia.org/wiki/GF(2)) :

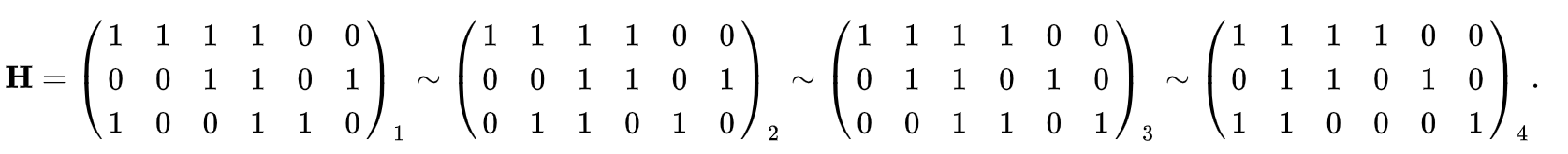


Ilustración 23

* Paso 1: h.
* Paso 2: La fila 1 se suma a la fila 3.
* Paso 3: Se intercambian las filas 2 y 3.
* Paso 4: La fila 1 se suma a la fila 3.

A partir de esto, la [matriz generadora](https://en.wikipedia.org/wiki/Generator_matrix)G se puede obtener como 𝐼𝑘 (observando que en el caso especial de que se trate de un código binario), o específicamente:

A number of numbers on a white background

Description automatically generated

Ilustración 24

Finalmente, multiplicando las ocho posibles cadenas de 3 bits por G , se obtienen las ocho palabras de código válidas. Por ejemplo, la palabra clave para la cadena de bits '101' se obtiene mediante:

A number on a white background

Description automatically generated

Ilustración 25

dónde  es el símbolo de la multiplicación mod 2.

Como verificación, el espacio de filas de G es ortogonal a H tal que 

La cadena de bits '101' se encuentra en los primeros 3 bits de la palabra clave '101011'.

#### Aplicaciones

Los Códigos LDPC se han vuelto cada vez más importantes en el mundo de las comunicaciones digitales. Una de las áreas en las que se utilizan ampliamente es en las comunicaciones por satélite, donde el ruido y la interferencia pueden causar errores significativos en los datos transmitidos. Además, los códigos LDPC también se utilizan en tecnologías de banda ancha como WiMAX y en el estándar de televisión digital DVB-S2.

En el campo de la informática, los códigos LDPC se utilizan en sistemas de almacenamiento de datos para garantizar la integridad de los datos. Esto incluye tanto el almacenamiento de disco duro como el almacenamiento de datos en la nube. Además, los códigos LDPC también se utilizan en algunas formas de memoria flash, donde pueden ayudar a prolongar la vida útil del dispositivo.

#### Beneficios y desafíos

Los códigos LDPC ofrecen varios beneficios significativos. En primer lugar, proporcionan una alta eficiencia de corrección de errores, lo que permite una comunicación más confiable en entornos ruidosos. Además, su diseño de baja densidad permite una mayor eficiencia en la transmisión de datos, lo que puede ser beneficioso en aplicaciones donde el ancho de banda es limitado.

Por otro lado, los códigos LDPC también presentan algunos desafíos. Su implementación puede ser más compleja en comparación con otros tipos de códigos correctores de errores, y la decodificación puede ser computacionalmente intensiva, lo que puede ser un problema para los dispositivos con recursos limitados.

# Conclusión

Los códigos de corrección de error son fundamentales para garantizar la fiabilidad de la transmisión y almacenamiento de datos en una amplia gama de aplicaciones tecnológicas. La elección del código adecuado depende de varios factores, como la naturaleza del canal de comunicación, la probabilidad y tipo de errores esperados, la complejidad computacional y los requisitos de eficiencia.

Aplicaciones críticas como las comunicaciones espaciales y almacenamiento de datos optan por códigos con alta capacidad de corrección de errores como Reed-Solomon y BCH.

Sistemas móviles y de alta velocidad prefieren códigos como LDPC y Turbo.

Sistemas simples y de bajo costo pueden usar códigos de paridad o de repetición para una implementación rápida y sencilla.

El uso correcto de estos códigos asegura que los datos se mantengan íntegros y confiables.

# Cuestionario

## Códigos de bloque

### Código Hamming

**¿Cómo funcionan los códigos de Hamming en términos de construcción y detección de errores, y qué limitaciones tienen en la corrección de errores múltiples?**

El código de Hamming agrega tres bits de comprobación por cada cuatro bits de datos, permitiendo la corrección de errores de un solo bit. Durante la construcción en el emisor, se añaden p bits de paridad a n bits de datos, formando una palabra de n + p bits que cumple un criterio de paridad. En el receptor, los bits de paridad se usan para detectar y localizar errores. Si la paridad es incorrecta, se identifica y corrige el bit erróneo. Sin embargo, si hay errores en más de un bit, el código puede corregir de manera incorrecta, resultando en un mensaje final erróneo. Las cuestiones clave incluyen cómo calcular p, las posiciones que afectan los bits de control, y dónde colocar estos bits en la palabra de código.

## Códigos cíclicos

### Código de paridad

**¿Cómo funciona el código de paridad brevemente?**

El código de paridad es una técnica simple de detección de errores que se utiliza para identificar errores de un solo bit en la transmisión de datos. Funciona añadiendo un bit adicional (bit de paridad) a una cadena de bits de datos para asegurar que el número total de bits '1' en la cadena sea par (paridad par) o impar (paridad impar).

### Códigos BCH

**Brevemente, ¿cómo se codifica un código BCH?**

Resumidamente, la codificación se logra realizando los siguientes pasos:

1. Seleccionar el campo de Galois: Elige un campo de Galois GF(2m), donde m determina el tamaño del campo.
2. Elegir los polinomios mínimos: Selecciona los polinomios mínimos que corresponden a las raíces del polinomio generador. Estos polinomios mínimos deben ser irreducibles y se eligen de tal manera que el polinomio generador pueda corregir hasta t errores.
3. Formar el polinomio generador: El polinomio generador g(x) es el producto de los polinomios mínimos seleccionados. La longitud del código n es generalmente 2m - 1, pero puede ser acortada según sea necesario.
4. Codificación: Para codificar un mensaje, el mensaje original se multiplica por el polinomio generador g(x) para producir la palabra de código. Esto asegura que todas las palabras de código estén dentro del conjunto de códigos válidos definidos por el polinomio generador.

**Brevemente, ¿cómo se codifica un código BCH?**

Los pasos para decodificar son:

1. Recibido el mensaje: Cuando un mensaje codificado se recibe, puede contener errores debido a ruido en el canal de comunicación.
2. Cálculo del síndrome: Se calcula un síndrome usando el mensaje recibido. El síndrome es una secuencia de valores que indica la presencia de errores y proporciona información sobre su ubicación.
3. Algoritmo de Berlekamp - Massey: Este algoritmo se utiliza para encontrar el polinomio localizador de errores, que identifica las posiciones de los errores en el mensaje recibido.
4. Corrección de errores: Una vez identificadas las posiciones de los errores, se puede corregir el mensaje recibido modificando los bits correspondientes.

### Código de repetición

**¿Puede dar un ejemplo de un código de repetición con su detección y corrección?**

Por ejemplo, con n = 7, si se recibe y = 1000100 es más lógico pensar que se ha deseado enviar la palabra x = 0000000, cometiéndose dos errores, que suponer que la palabra enviada es x = 1111111 donde se hubieran cometido 5 errores.

**¿Como se detectan los errores y se corrigen en un código de repetición?**

Suponiendo que en la transmisión se han producido errores, recibida una palabra, basta con contar el número de 0 y 1 que contiene la palabra recibida y decidir, por un simple criterio de mayoría, su decodificación como la palabra emitida más probable.

## Códigos convolucionales

**¿En qué se diferencian los códigos convulsionantes de los códigos bloques?**

Los códigos convolucionales son códigos lineales al igual que los códigos bloque y, por tanto, se utilizan para proteger la información añadiendo redundancia a la misma. Sin embargo, a diferencia de los códigos bloque, las palabras de un código convolucional se generan no sólo a partir de los dígitos de información actuales sino también con la información anterior en el tiempo. Es decir, un codificador convolucional es un sistema con memoria y, en consecuencia, lleva asociada una cadena de Markov aunque ésta no es visible en la salida pero sí la condiciona.

### Código turbo

**¿En qué consiste el código turbo?**

El codificador Turbo consiste en la concatenación de dos códigos continuos recursivos y sistemáticos, denominados codificadores constituyentes, que operan uno sobre la información a codificar (la parte sistemática) y el otro sobre una versión permutada (entre lazada) de la misma.

### Código de Hadamard

**¿A qué tipo de código es parecido el código de Hadamard?**

El código de Hadamard es un código de corrección de errores que se utiliza para la detección y corrección de errores cuando se transmiten mensajes por canales muy ruidosos o poco ﬁables y son un caso particular de los códigos Reed-Solomon, corresponden al primer código Reed Solomon.

## Códigos FEC

**¿Cuándo se implementan los códigos FEC?**

Debido a sus capacidades de corrección de errores, FEC se implementa comúnmente cuando se transmiten datos a través de canales de comunicación ruidosos o poco confiables, lo que lo hace adecuado tanto para comunicaciones por cable como inalámbricas. Por ejemplo, FEC proporciona una solución eficiente para transmitir contenido de video o admitir redes celulares. También puede ser utilizado por módems y enrutadores, admitir comunicaciones ópticas y por cable y facilitar transmisiones de datos unidireccionales.

### Código Reed-Solomon

**¿Qué ocurre cuando una palabra es decodificada con Reed-Solomon?**

Cuando una palabra de código es decodificada, existen tres posibilidades: si 2s + r < 2t (s errores, r "borrados") entonces la palabra de código original transmitida puede ser siempre recuperada; sino el decodificador detectará que no puede recuperar la palabra de código original e indicará este hecho, o l decodificador decodificará erróneamente y recuperará una palabra de código incorrecta sin indicación.

### Código LDPC

**¿En qué se destacan los códigos LDPC?**

La característica más importante de estos códigos radica en que sus matrices de chequeo de paridad son de baja densidad, es decir, tienen pocos elementos distintos de 0 y por esta razón, tanto la codificación como la decodificación mediante ellos es muy eficiente en términos de complejidad computacional.

Los códigos LDPC son particularmente útiles en entornos con altas tasas de error, como las comunicaciones inalámbricas y por satélite. Son capaces de corregir muchos errores sin necesidad de retransmitir los datos, lo que los hace especialmente valiosos en situaciones donde la retransmisión no es práctica o posible.

# Bibliografía

* <https://www.um.es/web/otri/-/desarrollo-de-codigos-correctores-aplicados-a-la-resolucion-de-problemas-de-transmision-de-informacion-en-empres-1/1.0#:~:text=Los%20c%C3%B3digos%20correctores%20de%20errores,en%20la%20transmisi%C3%B3n%20y%20corregirlos>.
* <https://es.wikipedia.org/wiki/C%C3%B3digos_detectores_y_correctores_de_error>
* <https://www.infor.uva.es/~cevp/FI_I/fichs_pdf_teo/FI_I_Tema8_CodCorr.pdf>
* <https://docs.informatica.com/es_es/data-quality-and-governance/data-quality/10-4-0/guia-de-transformaciones-de-developer/transformacion-de-comparacion/estrategias-para-la-coincidencia-de-campos/distancia-de-hamming.html>
* <https://es.wikipedia.org/wiki/Distancia_de_Hamming>
* <https://www.geocities.ws/abianchi04/textoredes/c3.pdf>
* <http://abstract.ups.edu/aata-es/section-error-detecting-correcting-codes.html>
* <https://academia-lab.com/enciclopedia/codigo-bch/>
* <https://riull.ull.es/xmlui/bitstream/915/223/1/Introduccion+a+la+teoria+de+codigos%3A+Codigos+ciclicos.+.pdf>
* <https://e-archivo.uc3m.es/entities/publication/76a009dd-78ed-4325-b218-281d85d93943>
* <https://www.electricity-magnetism.org/es/decodificador-ldpc/>
* <https://en.wikipedia.org/wiki/Low-density_parity-check_code>
* <file:///C:/Users/kiki2/Downloads/178866-Text%20de%20l'article-240506-1-10-20100503.pdf>
* <https://www.southampton.ac.uk/~sqc/ELEC6214/AWCNS-L16.pdf>
* <https://es.wikipedia.org/wiki/Turbo_c%C3%B3digo>
* <http://www.lucamartino.altervista.org/TOPIC6.pdf>
* <https://en.wikipedia.org/wiki/Convolutional_code>
* <https://oa.upm.es/4642/2/ESTRAN_MONO_2009_01.pdf>
* INTRODUCCION A LA TEORIA DE CODIGOS - RICARDO A. PODESTA
* Hoyos, A., Ruiz, U., y Chavez, E. (2021). Hadamard’s Defense Against Adversarial Examples.
* Reed, I. S. y Solomon, G. (1960). Polynomial Codes Over Certain Finite Fields. Journal of The Society for Industrial and Applied Mathematics
* Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L. (2009). ImageNet: A largescale hierarchical image database. En: 2009 IEEE Conference on Computer Vision and Pattern Recognition.
* Recuperación de información utilizando códigos de Hadamard y el grafo de semi-espacios proximales (2021) - Bryan Rodrigo Quiroz Palominos - Ensenada, Baja California, México.