UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
DE AREQUIPA
ESCUELA DE POSGRADO
UNIDAD DE POSGRADO DE LA FACULTAD DE CIENCIAS NATURALES Y FORMALES

MODELO MATEMÁTICO PARA LA CORRECCIÓN DE CÓDIGOS BINARIOS MEDIANTE LA MATRIZ DE HADAMARD APLICADO A LAS COTAS PARES DE PLOTKIN.

Tesis presentada por la bachiller
Jessica Andrea Núñez Carpio
Para optar el Grado Académico de
Maestra en ciencias: matemáticas, con mención en matemática universitaria superior

ASESORA: Mg. Rocío Bellido Dávila

Arequipa - Perú
2018

MODELO MATEMÁTICO PARA LA CORRECCIÓN DE CÓDIGOS BINARIOS MEDIANTE LA MATRIZ DE HADAMARD APLICADO A LAS COTAS PARES DE PLOTKIN.

Tesis Presentada por:

Bach. Jessica Andrea Núñez Carpio

JURADO:

Dra. Rosa Yolanda Carpio Barreda : _____

Dr. Adeluz Apaza Valdivia : _____

Mg. Rocío Bellido Dávila : _____

Agradecimientos

Quiero agradecer de manera muy especial a las personas que me inspiraron y compartieron sus conocimientos conmigo.
Especialmente agradezco los consejos y la confianza de mi asesora la Sra. Mg. Rocío Bellido Dávila.

Dedicatorias

A mi hermana Leslie por su infinita bondad y comprensión.
A mi padre, a mi madre por su cariño y apoyo incondicional.

INDICE

Resumen....6

Introducción....8

Capítulo I

Preliminares

1. Antecedentes....11

2. Definiciones....12

2.1. Matriz de Hadamard....12

2.1.1. Definición de la Matriz de Hadamard....13

2.1.2. Matriz de Hadamard Normalizada....14

2.1.3. Problema de Optimización Maximal....15

2.1.4. Propiedades de la Matriz de Hadamard....17

2.2. Teorema de Hadamard....18

2.3. Matriz Quasi-Hadamard....21

2.4. Alfabetos....21

2.5. Cadenas....22

2.6. Longitud de una cadena....23

2.7. Lenguajes....24

2.8. Cuerpo Finito....26

2.9. Código....27

2.10. Producto Kronecker....27

2.11. Método de Sylvester....29

Capítulo II

Códigos y Distancia de Hamming

1. Código....32

2. Tipos de códigos....33

2.1. Definición de Código Lineal....33

2.2. Códigos Bloque....35

2.3. Códigos Cíclicos....36

2.4. Código Binario....38

3. Codificación y decodificación 39
4. Matriz de Paridad 39
5. Peso de Hamming 41
6. Distancia de Hamming 42
7. Análisis de la Teoría de Códigos 46
8. Distancia de Levenshtein 53

Capítulo III

Cotas de Plotkin y Teorema de Levenshtein

1. Teorema de códigos óptimos para matriz de Hadamard 58
2. Cotas de Plotkin 61
3. Teorema de Levenshtein 64
4. Construcción de Códigos máximos-óptimos caso par 64
5. Modelación de la Codificación de Hadamard usando Matlab y Microsoft Excel 71
6. Simulación para corregir errores en el kardex de una microempresa 82
Conclusiones 94
Recomendaciones 96
Bibliografía 97
Anexos 101

Resumen

En el estudio de las Matemáticas encontramos a las matrices de Hadamard que tienen una aplicación muy importante en la Teoría de Códigos.

El problema en general de esta investigación asegura que la matriz de Hadamard es una matriz de control de paridad para generar códigos óptimos donde se corrijan los códigos binarios y así reducir la posibilidad de error. En este estudio se demuestra la relación del Teorema de Leveinshtein con el método de construcción de códigos máximos y con el método tradicional. Se distingue el tipo de código que debemos escoger para vincular el número de Hadamard que posee la matriz con la cantidad de palabras del código. Se obtienen dos tipos de resultados al relacionar parámetros como la cantidad de palabras M y la Distancia de Hamming en el caso par seleccionado de la cota de Plotkin y su correspondiente en el Teorema de Leveinshtein.

Es por esto que se realiza una modelación y simulación para construir la Codificación de Hadamard, la cual tiene uso en el kardex de una empresa como códigos de base binaria con la finalidad de obtener el método más pertinente, apropiado y práctico como data.

Palabras Clave. Matriz Casi Hadamard, códigos binarios, codificación, teorema de Leveinshtein, cotas pares de Plotkin, coreccion de errores, distancia de Hamming Leveinshtein.

Abstract

In the study of Mathematics we find the Hadamard matrices that have a very important application in the Theory of Codes.

The problem of this investigation declares the Hadamard matrix is a parity control matrix to generate optimal codes where the binary codes are corrected and thus reduce the possibility of error. In this Theory prove us the relationship between Leveinshtein's Theorem and the maximum code construction method, and traditional method too. We can see the kind of code that there are that choose for relating parameters such as Hadamard´s number that contain the matrix with the exactly number code's words. Finally obtain two kind of result for relating parameters such as M number of words and the Distance of Hamming in some case selected from bound's Plotkin and its corresponding in Leveinshtein's Theorem

That is why the need arises to perform a modeling and simulation to build the Hadamard Coding, which has use in company's kardex as binary base codes which search obtain the most appropieted method and practicle in the date.

Keywords. *Almost Hadamard matrix, binary codes, coding, Leveinshtein's theorem, Plotkin's even bounds, error correction, distance from Hamming Leveinshtein.*

Introducción

Una primera dificultad consiste en encontrar el Código con mayor capacidad de código, mientras los otros parámetros como la longitud de código y la distancia de Hamming permanecen fijos. Teniendo en cuenta que interesa también que n sea lo más pequeño posible para que el código no se alargue de más y genere un coste computacional enorme.
En cambio, se desea que M un número de palabras de código (capacidad de un código) sea lo más grande posible porque representa todas las combinaciones probables para poder detectar el error con todas las posibilidades existentes. Es así que surge una parametrización propuesta en las cotas de Plotkin.

Por otro lado cabe señalar la necesidad de una matriz generadora que propuesta aquí es la matriz de Hadamard, pues se encargará de ejercer el control de la paridad de tal código, para de esta manera generar buenos códigos de corrección de errores. Entre variadas codificaciones existentes, el generado por las matrices de Hadamard. En particular, siendo el Teorema de Leveinshtein quien otorga las condiciones para viabilizar la codificación es factible, pero no describe el método para obtener los códigos que están articulados con las cotas de Plotkin. Esto constituye un segundo problema.

Se indagará en la codificación más acorde con una modelación y simulación con códigos alternativos; que bien podrían tener utilidad en el kardex de una empresa como códigos base. Por ello los resultados que se mostrarán son de carácter numérico.

El problema en General de esta investigación envuelve esta pregunta: ¿Puede la matriz de Hadamard convertirse en una matriz de control de paridad para generar códigos óptimos donde se corrijan los códigos binarios y así reducir la posibilidad de error?

Los objetivos que constituyen esta investigación son, como objetivo general tenemos el aplicar el Teorema de Leveinshtein articulando la Cota de Plotkin seleccionada con la Matriz de Hadamard apropiada para generar Códigos Binarios Óptimos.
Como objetivos específicos para lograrlo tenemos:
Probar la existencia del número de Hadamard establecido en la Conjetura de Hadamard.
Encontrar el tipo de código que se adapte a nuestra matriz casi-Hadamard.
Aplicar el Método de Construcción de Códigos para Maximizar la cantidad de palabras, para que la longitud del código y la distancia de Hamming permanezcan fijos.
Interpretar los resultados de la implementación de las cotas pares de Plotkin a la codificación de Hadamard.
Simular Códigos Binarios Óptimos para corregir errores en el kardex de una microempresa.

La justificación de este trabajo de tesis es basada en la era de la información en que se vive, ya que es necesario conocer los mecanismos que conforman un mensaje digitalizado con respecto al envío y la recepción del código tal cual, y esto se ve en aquellos países desarrollados que poseen su información clasificada, a las que pueden acceder un grupo elit como la CIA y la KGB, entre otros. Es necesario conocer las formas de corrección de un código para así liberarlo de situaciones de error a causas de los medios de transmisión.
Por ello se quiere saber cómo es que sucede el proceso de codificación y decodificación, los parámetros que componen la codificación binaria, los métodos y las técnicas que surgen para relacionar definiciones matemáticas con la teoría de códigos a nivel computacional.
Por otro lado se sabe que se hace una simulación porque es útil al entorno, es por esto que se trabajará con lenguajes y alfabetos que controla una empresa de comercio, puede ser de retail, de procesos, de finanzas, de información; en cualquiera de éstas mandar un código por otro, sería fatal. Es así que en el proceso de ésta simulación trabajará con códigos alternativos pares, que se pueden usar en el kardex de una empresa como un sistema de ocho códigos base para emplear en

una tienda los cuales de antemano han sido agrupados en código de barras tal y como es asignado en el trabajo de tiendas o empresas.

En cuanto a la investigación de este trabajo, muy recientemente se han encontrado aplicaciones para diseñar códigos auto-correctores de errores en mensajes e imágenes trasmitidas incluso desde el espacio. Además cabe resaltar la importancia del Teorema de Leveinshtein presente en códigos de base exclusivamente decodificables, proyecciones del método con códigos abreviados y códigos de espacio-tiempo muy útiles para las comunicaciones inalámbricas, todo esto bastante aplicable en la era digital y en la generación millennials.

Capítulo I
Preliminares

1. Antecedentes.

Existe una vasta información sobre la teoría de códigos. Es por ello que surgen las matrices de Hadamard como una aplicación importante. Ésta es considerada una rama de las Matemáticas e Ingeniería surgida a mediados del siglo XIX que aunque tiene su origen en problemas de Ingeniería, el tema se ha desarrollado muy bien usando métodos matemáticos

Es así que siendo un campo tan amplio nos limitamos a la obtención de códigos lineales proponiendo aquellos que cumplen ciertas propiedades.

Los trabajos que se usan para construir tal codificación estan dados de manera implícita y es arduo descifrarlos, se fueron dando en la búsqueda de información: Primero "Hadamard matrices and their applications" de A. Hedayat y W.D. Wallis presenta el teorema de códigos óptimos para una matriz de Hadamard. En segundo lugar "Matrices de Hadamard" de T. Domínguez (2003) muestra la aplicación de la teoría de códigos con una matriz cualquiera a manera de ejemplificación, además marca la pauta para relacionar la Cota de Plotkin con el Teorema de Leveinshtein necesarios para aplicar la matriz de Hadamard. En tercer lugar esto fue confirmado y ampliado en la "Búsqueda de Matrices de Hadamard a través de Secuencias de Turyn" de Eduardo Piza (2011) el cual menciona su aplicación en teoría de códigos y muestra todos los tipos de construcción de

la Matriz de Hadamard. Apoyados en la idea de la existencia de matrices modificables llamadas Casi-Hadamard en "Error correcting codes from quasi-Hadamard matrices de V. Alvarez, J.A. Armario, M.D. Frau, E. Martin, and A. Osuna" mostrando los cuatro casos de las Cotas de Plotkin y la aplicabilidad de las Matrices Casi-Hadamard en la codificación de determinados ejemplos. Seguidamente "Coding Theory - Hadamard Codes de Massoud Malek" muestran la codificación de Hadamard desde un punto de vista más abstracto usando únicamente Matlab, pero arbitrariamente sin los parámetros definidos que buscamos. Finalmente se encontró "Codes Generated From Hadamard Matrices de Hakob SARUKHANYAN, Armen ANOYAN, Karen EGIAZARIAN, Jaakko ASTOLA, Sos AGAIAN" que ya mencionan la relación entre cada Cota de Plotkin y el Teorema de Leveinshtein de manera simplificada y organizada en una tabla".
Es por ello que la demostración, simulación y aplicación para el caso de Cotas Pares unicamente constituyen el objeto de estudio de la investigación.

2. Definiciones.

2.1. Matriz de Hadamard.

El matemático Jacques Hadamard ha contribuido en el desarrollo del análisis infinitesimal y desarrolló el teorema sobre el valor absoluto de un determinante. Ha dado su nombre a las matrices de Hadamard, al Teorema de Cauchy-Hadamard y se utiliza en criptografía la pseudo-transformación de Hadamard que son importantes en la teoría de las ecuaciones integrales, teoría de la codificación y otras áreas.

En su totalidad realizó 300 trabajos científicos y libros también escribió libros para un público más amplio. Su libro *La psicología de la invención en el campo matemático* (1945) es un trabajo maravilloso de las matemáticas, donde el Señor Jacques Hadamard usa la introspección para describir el proceso mental matemático. Describe su propio pensamiento matemático como mayormente sin palabras, acompañado a menudo de imágenes mentales que condensan la idea global de una prueba, en franca oposición a autores que identifican el lenguaje y la cognición. Además realizó una encuesta entre 100 de los físicos más relevantes del momento (aprox. 1900), preguntándoles cómo realizaban su trabajo. Muchas de las respuestas fueron idénticas a la suya; algunos informaron de que veían los conceptos matemáticos como colores. (como se cita en Math.info)

2.1.1. Definición de la Matriz de Hadamard.

Una matriz de Hadamard de orden n es una matriz H de tamaño $n \times n$ con entradas +1 y -1, tal que:

$$HH^t = nI$$

Es decir una matriz es de Hadamard si el producto interno de dos distintas filas es 0 y el producto interno de una fila consigo misma es n.

$H = \begin{pmatrix} + & + \\ + & - \end{pmatrix}_{2x2}$, $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}_{2x2}$, $n = 2$, $H^t = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

$$HH^t = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = 2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = 2I = nI$$

De la definición se sabe que cualesquiera dos columnas (o filas) de H son ortogonales. Evidentemente esta propiedad no cambia si permutamos las filas o columnas o si multiplicamos alguna fila o columna por -1: estas matrices son llamadas equivalentes.

(Piza, Eduardo 2011)

Para una matriz de orden 4 seria

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

,

$$H^t = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

$$HH^t = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

$$HH^t = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

$$HH^t = 4 \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = 4I = nI$$

2.1.2. Matriz de Hadamard Normalizada.

Sea una matriz de Hadamard, podemos encontrar otra equivalente en la cual la primera fila y la primera columna consistan enteramente de unos.
Dicha matriz de Hadamard se denomina normalizada. Claramente las filas restantes (si las hubiese) deben tener tantos más unos como menos unos. Por consiguiente, si $n \neq 1$ entonces n debe ser par.

Luego deducimos que al multiplicar cualquier fila o columna por -1 en una matriz de Hadamard se obtiene otra matriz de Hadamard, podemos cambiar la primera fila y columna para que estén formadas sólo por unos. Una matriz de este tipo se llama normalizada. Esto es, la norma de cada entrada es uno.
$|a_{ij}| = 1, |1| = 1, |-1| = 1$ (T. Domínguez 2003)

¿AHORA, PODRÁ EXISTIR UNA MATRIZ DE HADAMARD DE ORDEN 3 X 3?

Consideremos la matriz $G = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$

Se cumple que el producto de una fila con la misma columna es n, pero el producto de una fila o columna distintas son diferentes de 0. Por lo tanto no son ortogonales.

Ejemplos de Matriz de Hadamard:

Algunos ejemplos de matrices de Hadamard de ordenes pequeños (n = 1, n = 2 y n = 4) son:

Ejemplo 1:

$$H_1 = (1)$$

Ejemplo 2:

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Ejemplo 3:

$$H_4' = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Ejemplo 4:

$$H_4'' = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

(T. Domínguez 2003)

2.1.3. Problema de Optimización Maximal.

Hadamard considero el siguiente problema. Sea A = (a_{ij}) una matriz $n \times n$ con entradas reales, tales que $|a_{ij}| \leq 1$. ¿Qué tan grande puede ser el determinante de A (en valor absoluto)?

Esto es, se trata de resolver el problema de optimización

$$\max_{s.a.|a_{ij}|\leq 1} |\det A|$$

Como cada fila de A es un vector con longitud euclídea $\leq \sqrt{n}$, entonces el determinante no puede ser mayor que $n^{n/2}$, pues el valor absoluto del determinante de una matriz A es el volumen $n-$dimensional del paralelepípedo engendrado por los vectores fila de A en el espacio $n-$dimensional $\mathbb{R}^n$.

Esto es se satisface la desigualdad (Piza, Eduardo 2011)

$$|\det(A)| \leq n^{n/2}$$

Proposición:

Sea $a_i \ldots . a_n$ las filas de A. Entonces, por geometría Euclidiana, $|\ det\ (A)\ |$ es el volumen del paralelepípedo con lados $a_i \ldots . a_n$; asi que:

$$|\ det\ (A)| \leq |a_i| \ldots . |a_n|,$$

donde $|a_i|$ es la longitud Euclidiana del a; la igualdad se cumple si y sólo si $a_i \ldots . a_n$ son mutuamente perpendiculares. Por hipótesis,

$$|a_i| = (a_{i1}^2 + \cdots + a_{in}^2)^{1/2} \leq n^{1/2}\ ,$$

con igualdad si y sólo si $|a_{ij}| = 1$ para todo j. Del resultado sigue, ya que una matriz de Hadamard es una matriz real cuyas entradas todas tienen módulo 1 y cuyas filas son mutuamente perpendiculares.(ver Cameron Peter J. 4)

Ejemplo.

Haremos uso del tercer ejemplo H_4' , hallando el determinante por el método de los triángulos, entendiéndose el valor absoluto de la matriz como el determinante:

$$|A| = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix}$$

$$|A| = +1\begin{vmatrix} -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & 1 \end{vmatrix} - 1\begin{vmatrix} 1 & 1 & -1 \\ 1 & -1 & -1 \\ 1 & -1 & 1 \end{vmatrix} + 1\begin{vmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \end{vmatrix} - 1\begin{vmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & -1 \end{vmatrix}$$

$$|A| = +1[(1+1+1)-(-1+1-1)] - 1[(-1-1+1)-(+1+1+1)] + 1[(1+1+1)-(-1-1+1)] - 1[(-1+1-1)-(1+1+1)]$$

$$|A| = (3+1)-(-1-3)+(3+1)-(-1-3)$$

$$|A| = 4+4+4+4$$

$$|A| = 16$$

También podemos hallar su norma euclídea, así se tiene:

$h_1 = (1,1,1,1);$ $\quad |h_1| = \sqrt{1^2+1^2+1^2+1^2} = \sqrt{4}$

$h_2 = (1,-1,1,-1);$ $\quad |h_2| = \sqrt{1^2+(-1)^2+1^2+(-1)^2} = \sqrt{4}$

$h_3 = (1,1,-1,-1);$ $\quad |h_3| = \sqrt{1^2+1^2+(-1)^2+(-1)^2} = \sqrt{4}$

$h_4 = (1,-1,-1,1);$ $\quad |h_4| = \sqrt{1^2+(-1)^2+(-1)^2+1^2} = \sqrt{4}$

Por lo tanto, se cumple

$|\det(A)| \leq |h_1||h_2||h_3||h_4|$

$|\det(A)| \leq \sqrt{4}\sqrt{4}\sqrt{4}\sqrt{4}$

$|\det(A)| \leq 4^{\frac{1}{2}} 4^{\frac{1}{2}} 4^{\frac{1}{2}} 4^{\frac{1}{2}}$

$|\det(A)| \leq 4^{4/2}$

De lo anterior

$16 \leq 16$

2.1.4. Propiedades de la Matriz de Hadamard.

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

1) Las filas y columnas de cualquier matriz de Hadamard son ortogonales dos a dos. Esto significa que el producto escalar de dos filas o columnas distintas será cero.

Ejemplo 1: Tomando el producto escalar de la 2° y 4° columnas de H, se obtiene cero.

Las matrices de Hadamard poseen propiedades interesantes, para la verificación de las propiedades consideramos la siguiente matriz de Hadamard:

Se puede verificar que esta propiedad se cumple para todos los pares de filas o columnas de H.

2) Las filas y columnas de cualquier matriz de Hadamard son balanceadas dos a dos. Eso es, cuando la comparación de los elementos en cualquier par distinto de filas o columnas componentes de forma, será un número igual de pares de entradas y pares de entradas no idénticos.

Ejemplo 2: Con el fin de demostrar que estas columnas son balanceadas dos a dos vamos a utilizar las mismas dos columnas del ejemplo anterior y comparar las entradas en ambas columnas, así:

$$\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \begin{matrix} \leftrightarrow \\ \leftrightarrow \\ \leftrightarrow \\ \leftrightarrow \end{matrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$

Tengamos en cuenta que dos de las entradas comparadas son las mismas y dos son diferentes. Por lo que estas columnas son balanceadas dos a dos. Así podemos verificar esto para todos los pares de filas y columnas.

3) Una matriz de Hadamard normalizada es la fila y la columna balanceadas. Esto significa que cada fila y columna, a excepción de la fila y la columna inicial, tiene un número igual de unos y menos unos.
Ejemplo 3: Es obvio que H satisface la tercera propiedad por la misma definición.

(Ben Zowadab 2012)

Observación 1.

Nos preguntamos:

¿Para cualquier orden de n existen matrices de Hadamard?

La respuesta es que hay una bien conocida y necesaria condición enunciada posteriormente en el Teorema de Hadamard.

2.2. Teorema de Hadamard.

Si H es una matriz de Hadamard de orden n, entonces $n = 1$, o $n = 2$ o n es múltiplo de 4.

Demostración:

Sea n > 2. Normalizamos H, obteniendo una matriz de Hadamard $\widehat{H}$ equivalente.

Para ver esto, observamos en primer lugar que el cambio del signo de cada entrada en una columna de una matriz de Hadamard da otra matriz de Hadamard. Entonces el cambio de signos de todas las columnas para cuyas entradas en la primera fila es -, asumimos que todas las entradas en la primera fila son +. (Abreviamos +1 y -1 a + y -, respectivamente).

Es decir que en $\widehat{H}$ permutamos las columnas de tal forma que las primeras tres filas de la matriz resultante sean:

$$\begin{array}{cccccccccccc} + & + & \cdots & + & + & + & \cdots & + & + & + & \cdots & + \\ + & + & \cdots & + & + & + & \cdots & + & - & - & \cdots & - \\ + & + & \cdots & + & - & - & \cdots & - & + & + & \cdots & + \\ \underbrace{\qquad\qquad} & & & & \underbrace{\qquad\qquad} & & & & \underbrace{\qquad\qquad} & & & \\ a \text{ columnas} & & & & b \text{ columnas} & & & & c \text{ columnas} & & & \end{array} \quad \begin{array}{ccc} + & + & \cdots & + \\ - & - & \cdots & - \\ - & - & \cdots & - \\ \underbrace{\qquad\qquad} & & & \\ d \text{ columnas} & & & \end{array}$$

Son las tres filas de una matriz de Hadamard.

Tenemos entonces

$$a + b + c + d = n$$

Y los tres productos internos entre estas tres primeras filas arrojan las ecuaciones

Debido a que son ortogonales:

$$a + b - c - d = 0,$$
$$a - b + c - d = 0,$$
$$a - b - c + d = 0.$$

Resolviendo:

$$a + b + c + d = n$$
$$a + b - c - d = 0$$
$$a - b + c - d = 0$$
$$a - b - c + d = 0$$

Usando Gauss veremos:

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & n \\ 1 & 1 & -1 & -1 & 0 \\ 1 & -1 & 1 & -1 & 0 \\ 1 & -1 & -1 & 1 & 0 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & n \\ 0 & 0 & -2 & -2 & -n \\ 0 & -2 & 0 & -2 & -n \\ 0 & -2 & -2 & 0 & -n \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & n \\ 0 & 1 & 1 & 0 & n/2 \\ 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & -2 & -2 & -n \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & n \\ 0 & 1 & 1 & 0 & \frac{n}{2} \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -4 & -n \end{array}\right)$$

$$-4d = -n$$
$$d = \frac{n}{4}$$

$$c - d = 0$$
$$c = d$$
$$c = \frac{n}{4}$$

$$b + c = \frac{n}{2}$$
$$b = \frac{n}{2} - c$$
$$b = \frac{n}{4}$$

$$a + b + c + d = n$$
$$a = -b - c - d + n$$
$$a = -\frac{n}{4} - \frac{n}{4} - \frac{n}{4} + n$$
$$a = -\frac{3n}{4} + n$$
$$a = \frac{n}{4}$$
$$n = 4a$$

$$a = b = c = d = \frac{n}{4}$$

Sumando estas cuatro ecuaciones, obtenemos $n = 4a$, demostrando el teorema (en forma similar se demuestra también que:

$$4b = 4c = 4d = n$$

En otras palabras, puesto que de cada dos filas es ortogonal a la primera, adicionalmente vemos que cada fila tiene m entradas $+$ y m entradas $-$ donde $n = 2m$.
Por otra parte, si $n > 2$, las primeras tres filas son ahora como en la figura de arriba, con $n = 4a$. (Cameron Peter 2006)

Por lo tanto, la condición de arriba es necesaria y suficiente.
De ahí que la matriz se dé por bloques en múltiplos de 4 siguiendo el Teorema de Hadamard.

Observación 2.

La Conjetura de Hadamard.

- Debido a que cualquier n que sea múltiplo de 4 y positivo, cabe la existencia de la matriz de Hadamard de orden n, en otras palabras, que todo múltiplo de cuatro es un número de Hadamard es un problema en abierto, llamada Conjetura de Hadamard.
- Los menores órdenes n multiplos de 4 para los cuales aún no se conoce una matriz de Hadamard asociada son 668, 716, 876 y 892
- El Orden 428 fue resuelto en 2005 por H.Khagharani y B.Tayfeh-Rezaie. Hallaron una matriz de Hadamard utilizando algoritmos de Goethals-Seidel

2.3. Matriz Quasi-Hadamard.

Definición.

Definimos una matriz casi-Hadamard M de orden n como una matriz $n \times n$ de entradas $+1$ y -1 de tal manera que el producto interno de las filas de dos a dos es casi cero.

Propiedades:

- Tenemos el caso, cuando a veces es necesario precisar el número mayor q de filas en M las cuales son ortogonales uno al otro.
  Cuanto más grande es q, M es más cerca de ser una matriz de Hadamard.
  En estas circunstancias, M se denomina una Matriz cuasi-Hadamard de holgura q.

    - Podemos entender también que una matriz casi-Hadamard podría ser pensada como una Matriz Hadamard en la cual algunas filas han sido sustituidas, de modo que la condición de Hadamard generalmente se pierde a su vez.

(Álvarez, Armario, Frau, Martin y Osuna, 2007)

2.4. Alfabetos.

Definición:

Un alfabeto es un conjunto finito no vacío de elementos que llamaremos símbolos.
Si A un conjunto finito de símbolos entonces A es un alfabeto.

(Cruz Borges, 2005)

Notación:

Se suele denotar con $\sum$ a un alfabeto cualquiera. (Dr. Mata. Miguel, 2016)

2.5. Cadenas.

Una cadena o palabra sobre un alfabeto es cualquier sucesión finita de elementos de $\sum$. Admitimos la existencia de una única cadena que no tiene símbolos, la cual se denomina cadena vacía y se denota con λ. La cadena vacía desempeña, en la teoría de lenguajes formales, un papel similar al que desempeña el conjunto vacío $\emptyset$ en la teoría de conjuntos.

Ejemplo 1.

Sea $\sum = \{a, b\}$ el alfabeto finto que consta de los dos símbolos a y b. Las siguientes son cadenas sobre $\sum$:

aba

ababaaa

aaaab.

Obsérvamos que el orden aba $\neq$ aab. El orden de los símbolos en una cadena es significativo ya que las cadenas se definen como sucesiones, es decir, conjuntos secuencialmente ordenados.

Ejemplo 2.

El alfabeto $\sum = \{0, 1\}$ se conoce como alfabeto binario. Las cadenas sobre este alfabeto son secuencias finitas de ceros y unos, llamadas secuencias binarias, tales como

001

1011

001000001.

Ejemplo 3.

$\sum = \{a, b, c, \ldots, x, y, z\}$ el alfabeto del idioma castellano. Las palabras oficiales del castellano (las que aparecen en el diccionario DRA) son cadenas sobre $\sum$.

Ejemplo 4.

El alfabeto utilizado por muchos de los llamados lenguajes de programación (como Pascal o C) es el conjunto de caracteres ASCII (o un subconjunto de él) que incluye, por lo general, las letras mayúsculas y minúsculas, los símbolos de puntuación y los símbolos matemáticos disponibles en los teclados estándares.

Nota:

El conjunto de todas las cadenas sobre un alfabeto $\sum$, incluyendo la cadena vacía, se denota por $\sum*$.

Ejemplo 5.

Sea $\sum = \{a, b, c\}$, entonces

$\sum*$ = { λ, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, abc, baa, . . .}.

Es un alfabeto infinito

Observación 3.

$u, v, w, x, y, z, \ldots \alpha,\ \beta,\ \gamma,$ ... denotan cadenas, es decir, sucesiones finitas de símbolos de un alfabeto. (Prof. De Castro R. 2003)

2.6. Longitud de una cadena.

La longitud de una cadena $u \in \sum*$ se denota $|u|$ y se define como el número de símbolos de u. Contando la posición y los símbolos repetidos.

Es decir,

$$|u| = \begin{cases} 0 & , \quad si\ u = \lambda \\ n & , \quad si\ u = a_1 a_2 \ldots a_n \end{cases}$$

Deducimos que 0 cuenta como espacio o separación entre cadenas, y n es para señalar el conteo. (Universidad Tecnológica Nacional Facultad Regional Buenos Aires, 2018)

Ejemplo 1.

$$|aba| = 3, |baaa| = 4.$$

Propiedad de la Cadena:

Si $w \in \sum*$, $n, m \in \mathbb{N}$, se cumple que

$$|w^{n+m}| = |w^n| + |w^m|$$

Demostración:

Caso $n, m \geq 1$. $|w^{n+m}| = \underbrace{|ww \ldots w|}_{n+m\ veces} = (n+m)|w|$. Por otro lado,

$$|w^n| + |w^m| = \underbrace{|ww \ldots w|}_{n\ veces} + \underbrace{|ww \ldots w|}_{m\ veces} = n|w| + m|w|.$$

Caso $n = 0, m \geq 1$. $|w^{n+m}| = |w^{0+m}| = |w^m|$ Por otro lado,

$$|w^n| + |w^m| = |w^0| + |w^m| = |\lambda| + |w^m| = 0 + |w^m| = |w^m|.$$

Caso $m = 0, n \geq 1$. Similar al caso anterior.

Caso $n = 0, m = 0$. $|w^{n+m}| = |w^{0+0}| = |\lambda| = 0$. Por otro lado,

$$|w^n| + |w^m| = |w^0| + |w^0| = |\lambda| + |\lambda| = 0 + 0 = 0.$$

(Prof. De Castro R. 2003)

2.7. Lenguajes.

Un lenguaje L sobre un alfabeto $\sum$ es un subconjunto de $\sum*$, es decir $L \subseteq \sum*$.

Casos extremos:

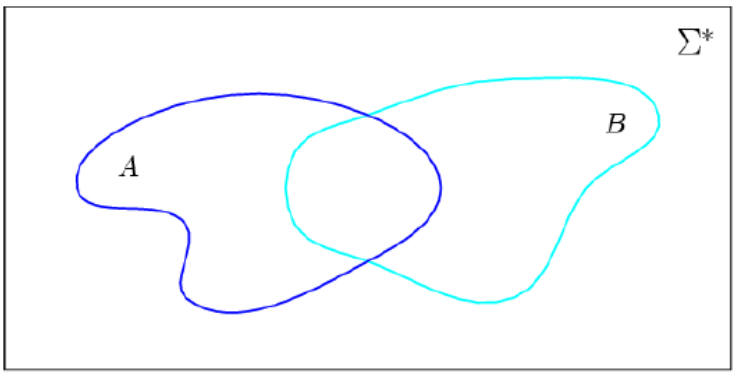
$L = \emptyset$, lenguaje vacío.

$L = \sum*$, lenguaje de todas las cadenas sobre $\sum$. Es un alfabeto completo.

Todo lenguaje L satisface $\emptyset \subseteq L \subseteq \sum*$, y puede ser finito o infinito.

Los lenguajes se denotan con letras mayúsculas $A, B, C, \ldots, L, M, N, \ldots$.

En la siguiente gráfica se visualizan dos lenguajes A y B sobre $\sum$.



Ejemplos:

Los siguientes son ejemplos de lenguajes sobre los alfabetos especificados.

$\sum = \{a, b, c\}. L = \{a, aba, aca\}.$

$\sum = \{a, b, c\}. L = \{a, aa, aaa, \dots\} = \{a^n : n \geq 1\}.$

$\sum = \{a, b, c\}. L = \{\lambda, aa, aba, ab^2a, ab^3a, \dots\} = \{ab^n a : n \geq 0\} \cup \{\lambda\}.$

$\sum = \{a, b, c, \dots, x, y, z\}\ L = \{u \in \sum * :\ u \in \text{DRA}\}$. L es un lenguaje finito.

$\sum. = \{a, b, c\}.\ L = \{u \in \sum * :\ u\ no\ contiene\ el\ símbolo\ c\}$. Por ejemplo, $abbaab \in L$ pero $abbcaa \notin L$, nos invade una cadena de otro alfabeto L.

$\sum = \{0, 1\}.\ L =$ conjunto de todas las secuencias binarias que contienen un número impar de unos.

$\sum = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. El conjunto $\mathbb{N}$ de los números naturales se puede definir como un lenguaje sobre $\sum$, en la siguiente forma:

$$N = \{u \in \sum * :\ u = 0 \text{ ó } 0\ no\ es\ un\ prefijo\ de\ u\}.$$

Aquí 01, 02 no existen; pero si es sufijo 10, 20.

Cabe señalar que el concepto abstracto de "lenguaje", tal como se ha definido, no es exactamente la misma noción utilizada en la expresión "lenguaje de programación". Para precisar la relación entre estos conceptos, consideremos el alfabeto $\sum$ de los caracteres

ASCII. Un programa en C o en Pascal, por ejemplo, es simplemente una cadena de símbolos de $\sum$ y, por lo tanto, un conjunto de programas es un lenguaje (en el sentido formal definido en esta investigación). (Prof. De Castro R. 2003)

2.8. Cuerpo Finito.

Un cuerpo finito, campo finito o campo de Galois (llamado así por Évariste Galois) es un cuerpo definido sobre un conjunto finito de elementos.

Los cuerpos finitos son importantes en teoría de números, geometría algebraica, teoría de Galois, y criptografía. Todos los cuerpos finitos tienen un número de elementos $\boldsymbol{q = p^n}$, para algún número primo $\boldsymbol{p}$ y algún entero positivo $\boldsymbol{n}$. Para cada cardinalidad q así definida hay una y sólo una manera posible de definir un campo finito, por lo que todos los campos finitos del mismo orden son isomorfos entre sí.

Sea $\boldsymbol{F}[\mathbf{7}]$ el conjunto de los enteros módulo 7 bajo la adición y multiplicación módulo 7. Es decir, los elementos de $\boldsymbol{F}[\mathbf{7}]$ son las clases de equivalencia representadas por los elementos [0], [1], [2], [3], [4], [5] y [6] donde:

[a] + [b] = [j], siendo [j] el resto de la división de (a+b)/7 (por ejemplo [5] + [6] = [4], puesto que 5+6=11, que dividido por 7, da resto 4).

[a] x [b] = [k] donde [k] es el resto de la división de (h x i)/7 (Por ejemplo, [5] x [6] = [2], puesto que 5 x 6 = 30 y 30 entre 7, da como resto 2).

Se verifica que $\boldsymbol{F}[\mathbf{7}]$ es un anillo conmutativo con elemento unitario [1]. Además se cumple:

[1] x [1] = [1] = [6] x [6].
[2] x [4] = [1] = [4] x [2].
[3] x [5] = [1] = [5] x [3].

Los elementos de $\boldsymbol{F}[\mathbf{7}]$ distintos de cero forman un grupo abeliano bajo la multiplicación. $\boldsymbol{F}[\mathbf{7}]$ es, pues, un campo. Puesto que tiene un número finito de elementos es un campo finito.

Ejemplo de cuerpo finito.

El cuerpo más pequeño tiene solamente dos elementos: 0 y 1. Se denota por $\mathbb{F}_2$ o $\mathbb{Z}_2$ y a veces puede definirse mediante las dos tablas

+	**0**	**1**
0	0	1
1	1	0

.	**0**	**1**
0	0	0
1	0	1

Tiene aplicaciones importantes en informática, especialmente en álgebra de Boole, criptografía y teoría de la codificación.

En general, para un número primo p, el conjunto de los números enteros módulo p es un cuerpo finito con los p elementos; esto se escribe a menudo como $\mathbb{Z}_p = \{0,1,\dots,p-1\}$ donde las operaciones son definidas realizando la operación en $\mathbb{Z}$, dividiendo por p y tomando el resto.

(Cuerpo (matemáticas), 2018)

2.9. Código.

Decimos que un codigo es de decodificacion unica si cada vez que recibimos una cadena de simbolos o mensaje, ésta corresponde con una única lista de palabras.

Se denota C un $[n,k]$-código sobre el cuerpo $\mathbb{F}$.

2.10. Producto Kronecker.

Sea A una matriz de tamaño $m \times n$ con entradas a_{ij} y sea B otra matriz cualquiera. La matriz

$$\begin{pmatrix} a_{11}B & a_{12}B & \dots a_{1n}B \\ a_{21}B & a_{22}B & \dots a_{2n}B \\ \vdots & \vdots & \ddots \quad \vdots \\ a_{m1}B & a_{m2}B & \dots a_{mn}B \end{pmatrix}$$

consistente en mn bloques con el tamaño de B, se llama producto de Kronecker de las matrices A y B y se denota $A \otimes B$.

Teorema.

Si H_m y H_n son matrices de Hadamard de ordenes m y n respectivamente, entonces $H_m \otimes H_n$ es una matriz de Hadamard de orden mn.

Demostración: Realizando cálculos directos podemos previamente demostrar las siguientes dos propiedades:

$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$

$(A \otimes B)^t = A^t \otimes B^t$.

Tomemos $A = C = H_m$ y $B = D = H_n$. El resultado se obtiene de la definición de matriz de Hadamard y del hecho que $I_m \otimes I_n = I_{mn}$.

(Piza Eduardo, 2011)

El producto Kronecker también denominado producto tensorial o producto directo de dos matrices A y B se define como sigue:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots a_{1n}B \\ a_{21}B & a_{22}B & \dots a_{2n}B \\ \vdots & \vdots & \ddots \quad \vdots \\ a_{m1}B & a_{m2}B & \dots a_{mn}B \end{pmatrix}$$

Por ejemplo:

$$H_4 = H_2 \otimes H_2$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Para $n > 4$, con el fin de construir una matriz de Hadamard de orden n utilizando el Producto Kronecker, n debe ser divisible por 8. Por ejemplo, H_{12} existe, pero no puede ser construido de cualquier matriz de Hadamard.

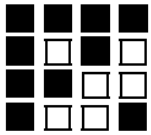
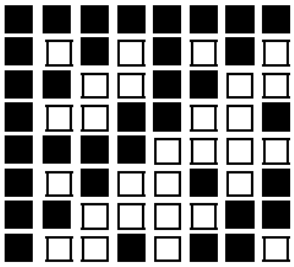
Así entendemos que el producto tensorial es un caso de transformación lineal más exactamente una transformación bilineal. Dado que posee las siguientes propiedades:

$$T: V \to W$$
$$T(u+v) = T(u) + T(v)$$
$$T(\alpha u + v) = \alpha T u + T v$$
$$T(\alpha u) = \alpha T(u)$$
$$\varphi: FxF \to G$$
$$(u, v) \to \varphi(u, v) = w$$
$$\varphi(\lambda u + v, w) = \lambda\varphi$$
$$\varphi(\lambda x_1 + x_2, y) = \lambda\varphi(x_1, y) + \varphi(x_2, y)$$
$$\varphi(x, \lambda y_1 + y_2) = \lambda\varphi(x, y_1) + \varphi(x, y_2)$$

Por lo tanto la función es Sobreyectiva (imagen-imagen), cumple las propiedades de factorización o composición de funciones; de manera muy similar es la definición de producto tensorial.

2.11. Método de Sylvester.

Una clase de matrices Hamadard $H_{2n} = H_2 \otimes H_n$ fue definido por Sylvester bajo el nombre del "Pavimento Anallagmático". En una matriz de Hadamard-Sylvester vista como pavimentos, las celdas con son de color negro y las de $-1's$ son de color blanco.

$H_1 =$, $H_2 =$, $H_4 =$  , $H_8 =$  .

Por lo tanto, la matriz Hn de Hadamard-Sylvester $n \times n$ debe tener $n\,(n-1)\,/\,2$ cuadrados blancos y $n\,(n+1)\,/\,2$ cuadrados negros.

(Massoud M., 2018)

La aplicación iterada del teorema precedente, empezando con la matriz de Hadamard de orden 2

$$H_2 = \begin{pmatrix} + & + \\ + & - \end{pmatrix}$$

nos lleva a las matrices de Sylvester S(k) de orden 2^k, que son productos iterados de Kronecker de k copias de la matriz de Hadamard H_2. Por ejemplo,

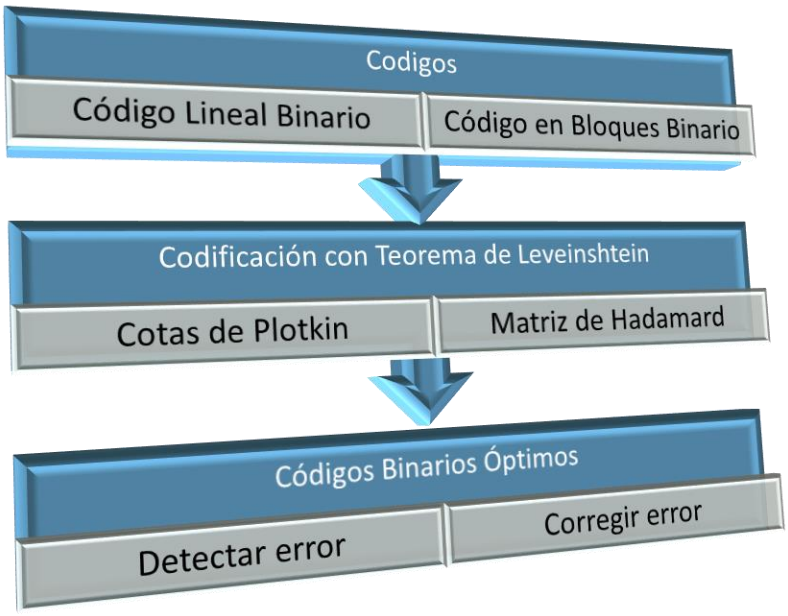
$$S(2) \coloneqq \begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix}$$

Formalmente, $S(1) \coloneqq H_2$ y $S(k\ +\ 1) \coloneqq H_2 \otimes S(k)$, para $k \in \mathbb{N}^*$.

Las matrices de Sylvester fueron descubiertas en 1887 por J. J. Sylvester y tienen algunas propiedades especiales: son normalizadas, simétricas, tienen traza nula y sus elementos en las restantes filas y columnas distintas a la primera tienen la misma cantidad de unos que menos unos. (Piza E. 2011)

Capítulo II

Códigos y Distancia de Hamming



Esquema 1: Representación gráfica de la relación de las variables

1. Código.

Sea *A, B* dos alfabetos. Llamaremos código o diccionario a la aplicación inyectiva:

$$\boldsymbol{C}: \bigcup_{\boldsymbol{k}\in\mathbb{N}} \boldsymbol{A^k} \rightarrow \bigcup_{\boldsymbol{l}\in\mathbb{N}} \boldsymbol{B^l}$$

Usualmente también se denomina código a la imagen de la aplicación $\boldsymbol{C}$.

Llamamos código a la imagen de $\boldsymbol{C}$ y se llama palabra código o simplemente palabra a cada uno de sus elementos.

Recordemos que una función $\boldsymbol{f}: \boldsymbol{X} \rightarrow \boldsymbol{Y}$ es inyectiva si a elementos distintos del conjunto $\boldsymbol{X}$ (dominio) les corresponden elementos distintos en el conjunto $\boldsymbol{Y}$ (codominio) de $\boldsymbol{f}$. Es decir, cada elemento del conjunto Y tiene a lo sumo una preimagen en $\boldsymbol{X}$, o, lo que es lo mismo, en el conjunto $\boldsymbol{X}$ no puede haber dos o más elementos que tengan la misma imagen.

De manera más precisa, la función $\boldsymbol{f}: \boldsymbol{X} \rightarrow \boldsymbol{Y}$ es inyectiva si, sólo si a, b son elementos de X tales que $f(a) = f(b)$, entonces $a = b$.
O equivalentemente , la función $\boldsymbol{f}: \boldsymbol{X} \rightarrow \boldsymbol{Y}$ es inyectiva si, sólo si a, b son elementos diferentes de X, entonces $f(a) \neq f(b)$. Simbólicamente

$$\forall\, a, b \in X, f(a) = f(b) \Longrightarrow\ a = b$$

que es equivalente a su contrarrecíproco

$$\forall\, a, b \in X, a \neq b \Longrightarrow\ f(a) \neq f(b)\ .$$

Ejemplo.

Los primeros códigos un poco más elaborados son los códigos en bloque. El método es bien simple, se basa en "bloques" de información que se transforman en otros "bloques" mediante una aplicación llamada diccionario.

Considerando el alfabeto binario $\sum = \{\mathbf{0}, \mathbf{1}\}$
Tenemos el diccionario o lenguaje:

$$\mathbf{0 \rightarrow\ 0000}$$
$$\mathbf{1\ \rightarrow\ 1001}$$
$$\mathbf{2\ \rightarrow 1010}$$
$$\mathbf{3 \rightarrow\ 0011}$$
$$\mathbf{4\ \rightarrow\ 1100}$$
$$\mathbf{5\ \rightarrow\ 0101}$$
$$\mathbf{6\ \rightarrow\ 0110}$$
$$\mathbf{7\ \rightarrow 1111}$$

Que responde a una secuencia determinada.

Y si la cadena es el mensaje: "35724", el código que se forma a enviar es: "00110101111110101100". Entonces $|u| = 20$ es la longitud del código o cadena.

De aquí deducimos que si está erróneo algún carácter o dígito, entonces ya se envía otro número, por lo tanto otra cadena es recibida.

Aquí se encuentra el problema, en la detección del error y cómo lo corregiremos.

El coste computacional de la decodificación es excesivo. Se debe guardar el diccionario con el que se codifica la información, el cual tiende a ser relativamente grande (en este caso es mayor que el mensaje a codificar).

Es por eso que surgen varios tipos de códigos con sus diferentes decodificaciones.

(Cruz B. H., 2005)

2. Tipos de códigos.

2.1. Definición de Código Lineal.

Sea $\mathbb{F}$ un cuerpo finito de $\boldsymbol{q}$ elementos. Un código lineal, $\boldsymbol{C}$, es un subespacio del espacio vectorial $\mathbb{F}^{\boldsymbol{n}}$, que es $\boldsymbol{n}$- dimensional de orden $\boldsymbol{q^n}$.

Entonces escribimos $[\boldsymbol{n}, \boldsymbol{k}, \boldsymbol{d}]$ -código para un código lineal de $\mathbb{F}^{\boldsymbol{n}}$ con $\boldsymbol{dim_F\ C = k}$ y distancia mínima d. Escribiremos sólo $[\boldsymbol{n}, \boldsymbol{k}]$-código cuando no se hace énfasis en la distancia mínima.

(Pinero Jacobo, 2011)

Observación 4.

Cabe mencionar que el peso de una palabra es el número de componentes no cero (o su distancia a la palabra 0).

Ahora para la presentación de los códigos lineales se usan dos tipos de matriz a saber, la matriz generadora y la matriz de control.

(Iranzo y Pérez,2004)

Matriz Generadora:

Una matriz generadora de C es cualquier matriz cuyas filas formen una base para C. Se denota $G(C)$.

Sea M una matriz de orden $n \times n$. Denotamos $M\ (i_1, \dots, i_r; j_1, \dots, j_s)$ la submatriz de M que proviene de la intersección de las Filas $i_1, \dots, i_r$ y las columnas $j_1, \dots, j_s$.

Entonces, si $G = (I_k|A)_{k\times n}$ es la matriz generadora en forma típica de $un\ [n,k]$- código lineal entonces $H = (-A^t|I_{n-k})$ es una matriz de control de C.

Teorema.

Todo código lineal $\boldsymbol{C}$ es permutación equivalente a un código cuya matriz generadora esta en forma típica.

Prueba:

Sea G la matriz generadora. Por álgebra lineal (método de Gauss- Jordán) sabemos que existen matrices invertibles (de operaciones elementales),E, F, tales que $EGF = (I_k|A)$, donde I_k es la identidad y A alguna matriz adecuada.

También sabemos que podemos elegir F como sólo producto de permutación de los pivotes (o cualquier permutación que pusiera en las primeras columnas un conjunto de información).

Entonces EG genera el mismo código C y por tanto CF es permutación equivalente y tiene la matriz de la forma deseada. (Pinero J., 2011)

Ejemplo.

Sea

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \in M_{2\times 3}(\mathbb{Z}_2)$$

Como las filas son linealmente independientes, Rango de G es 2 y genera un código binario L con parámetros [3; 2]. Como

$$(x_1 \quad x_2)\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = (x_1 \quad x_1 + x_2 \quad x_2) \sim (x_1, x_1 + x_2, x_2)$$

Codificamos

$$00 \rightarrow 000;\ 01 \rightarrow 011;\ 10 \rightarrow 110;\ 11 \rightarrow 101.$$

Luego

$$\mathrm{L} = \{000;\ 011;\ 101;\ 110\} = E(3)$$

y vemos que la distancia es $d = 2$.

(A. Podesta R., 2006)

2.2. Códigos Bloque.

$$\boldsymbol{C}: \bigcup_{\boldsymbol{k}\in\mathbb{N}} \boldsymbol{A^k} \rightarrow \boldsymbol{B^n}$$

Es una aplicación inyectiva, entonces $\boldsymbol{C}$ es un código en bloque de longitud $\boldsymbol{n}$.

(Borges, C. E., 2005)

Dicho de otra forma: Un (n, M) −código bloque C sobre un alfabeto A, es un subconjunto de A^n, con $|C| = M$.

Normalmente, hablaremos de n como la longitud de C y de M como el tamaño o el número de palabras código de C. (Peralta, J., 2018)

Ejemplo de Código Bloque. Sea el diccionario:

0	0000	4	1100
1	1001	5	0101
2	1010	6	0110
3	0011	7	1111

Mensaje	Código
3572 $_{(8)} \Rightarrow$	**0011 0101 1111 1010**

De aquí deducimos que éste código es:

- Excesivamente costoso.
- Y desconocemos la distancia mínima.

(Cruz E Borges H. 2005)

Cabe resaltar que los códigos bloque son una extensión de los códigos lineales.

2.3. Códigos Cíclicos.

Recordemos que n y q son dos números enteros son coprimos o primos entre sí si su máximo común divisor (mcd) es 1. En tal caso los únicos divisores comunes son -1 y 1.
Esto equivale a decir que la fracción a/b es irreductible (no se puede simplificar).

Ejemplo.
36 y 91 son coprimos porque en la descomposición en factores primos de cada uno, no aparece ningún factor común: $36 = 2^2 \times 3^2$ y $91 = 7 \times 13$.
En particular, nos interesa el caso si $q = 2$ entonces $n\ es\ impar$.

Definición del Código Cíclico.
Un código lineal $C \subset \mathbb{F}_q^n$ es cíclico si

$$c_0 c_1 \dots c_n \in C \Rightarrow c_{n-1} c_0 \dots c_{n-2} \in C$$

Por definición un código lineal C es cíclico si es cerrado por el desplazamiento cíclico $c_0 c_1 \dots c_{n-1} \mapsto c_{n-1} c_0 \dots c_{n-2}$. En este caso, C es cerrado por todos los desplazamientos cíclicos $c_0 c_1 \dots c_{n-1} \mapsto c_k \dots c_{n-1} c_0 \dots c_{k-1}$.

Si $C \subset \mathbb{F}_q^n$ es un código lineal $q - ario$, a cada palabra código podemos asignarle un polinomio como sigue:

$$\phi: C \to \mathbb{F}_q|x|,\ c_0 c_1 \dots c_{n-1} \mapsto c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}.$$

ϕ es un isomorfismo de espacio vectorial de C sobre $\phi(C)$. Luego, de ahora en adelante, ignoraremos ϕ y pensaremos a las palabras códigos como polinomios, y recíprocamente.

Se entiende por mapeo referirse a una función o una relación matemática, cuando se trata de campos y / o contradominios no numéricos. Otros sinónimos son de aplicación matemática y la transformación.

Mientras que un isomorfismo pretende captar la idea de tener la misma estructura. Dos estructuras matemáticas entre las que existe una relación de isomorfismo se llaman isomorfas

El cociente

$$R_n = \frac{\mathbb{F}_q|x|}{\langle x^n - 1\rangle}$$

es el álgebra de polinomios de grado menor que n, con la suma usual de polinomios y el producto de polinomios seguido de reducción módulo $x^n - 1$.

Aquí deducimos que se trata de una grupo cociente.

Observación 4.

Un código C es cíclico si y solo si $\phi(C)$ es un *ideal* en $\mathbb{F}_q|x| = \langle x^n - 1\rangle$. En efecto, si $c_0c_1 \ldots c_{n-1} \in C$, entonces

$$x(c_0 + c_1x + \cdots + c_{n-1}x^{n-1}) = c_0x + c_1x^2 + \cdots + c_{n-1}x^n \; mod\; (x^n - 1)$$
$$= c_{n-1} + c_0x + \cdots + c_{n-2}x^{n-1}$$

(A. Podesta Ricardo 2006, Cruz E. Borges 2005, Peralta Justo 2018)

Ejemplo de Código Cíclico.

Veamos, por ejemplo, el código $C_3(7) = \langle x^3 + x + 1\rangle$. Para obtener este código más fácilmente, armamos la siguiente tabla, en donde en la primera columna ponemos los factores $p(x)$ y en la segunda los múltiplos. $p(x)(x^3 + x + 1)$ del polinomio generador. Como multiplicar por x es hacer un desplazamiento cíclico en las palabras de $\mathbb{F}_2^7$, las palabras códigos de $C_3(7)$ se obtienen haciendo los desplazamientos del polinomio generador por $1, x, x^2$y x^3, y luego haciendo todas las posibles sumas de estas cuatro palabras.

$p(x)$	$p(x)(x^3+x+1)$	palabra código
0	0	0000000
1	x^3+x+1	1101000
x	x^4+x^2+x	0110100
x^2	$x^5+x^3+x^2$	0011010
x^3	$x^6+x^4+x^3$	0001101
$1+x$	$x^4+x^3+x^2+1$	1011100
$1+x^2$	x^5+x^2+x+1	1110010
$1+x^3$	x^6+x^4+x+1	1100101
$x+x^2$	$x^5+x^4+x^3+x$	0101110
$x+x^3$	$x^6+x^3+x^2+x$	0111001
x^2+x^3	$x^6+x^5+x^4+x^2$	0010111
$1+x+x^2$	x^5+x^4+1	1000110
$1+x+x^3$	x^6+x^2+1	1010001
$1+x^2+x^3$	$x^6+x^5+x^4+x^3+x^2+x+1$	1111111
$x+x^2+x^3$	$x^6+x^5+x^2$	0100011
$1+x+x^2+x^3$	$x^6+x^5+x^3+1$	1001011

Cabe notar que el código $C_3(7)$ coincide con el código de Hamming binario $\mathcal{H}_2(3)$. Es decir, $\mathcal{H}_2(3)$ es cíclico.

Funcionando el ultimo codigo de $1x^6+1x^5+0x^4+1x^3+0x^2+0x\quad+1x^0$

(A. Podesta Ricardo 2006)

2.4. Código Binario.

Sea C un código $q-$ario. Se dice que C es un código binario, ternario o cuaternario según sea $q=2, q=3$ ó $q=4$, respectivamente. Los códigos binarios son los más comunes y los más antiguos. Últimamente, hay gran interés en ciertos códigos cuaternarios ya que poseen algunas mejoras respecto de los binarios.

Existe también la tasa de información de un $(n,M)-$código $q-$ario se define por

$$R = R_q(C) = \frac{log_q(M)}{n}$$

Esta tasa da una idea del porcentaje de dígitos que guardan la información del mensaje original sobre el total de dígitos transmitidos. Se buscan códigos con tasa de información alta, digamos

$$R > \frac{2}{3} \text{ ó } R > \frac{3}{4}$$

Ejemplo de Código Binario y tasas de información.

Sean $A_2 = \{0, 1\}$ y $A_3 = \{0, 1, 2\}$. Los códigos binarios (sobre A_2)

$$C_1 = \{0, 1\}, C_2 = \{00, 01, 10\},$$
$$C_3 = \{000, 111\}, C_4 = \{000, 011, 101, 110\},$$

tienen parámetros $(1, 2), (2, 3), (3, 2)$ y $(3, 4)$, respectivamente. Las correspondientes tasas de información están dadas por

$$R(C_1) = log_2(2) = 1;\ R(C_2) = \frac{log_2(3)}{2} \approx 0{,}7925;$$
$$R(C_3) = log_2(2)/3 = 1/3;\ R(C_4) = log_2(4)/3 = 2/3$$

(A. Podesta Ricardo 2006)

3. Codificación y decodificación

Sigue este proceso:

$$00 \xrightarrow[\text{codificamos}]{} 000000 \xrightarrow[+1\ \text{error}]{} \left\{ \begin{array}{c} 100000 \\ 010000 \\ 001000 \\ 000100 \\ 000010 \\ 000001 \end{array} \right\} \xrightarrow[\text{corregimos}]{} 000000 \xrightarrow[\text{decodificamos}]{} 00$$

4. Matriz de Paridad

Para relacionar una Distancia de un código lineal y algunas cotas, debemos tener en cuenta que no hay una manera efectiva de calcular la distancia dL de un código lineal L a partir de una matriz generadora. Sin embargo, esto es posible a partir de una matriz de paridad.

Entonces:

4.1. Definición (Matriz generadora).

Los códigos lineales, al ser subespacios, se pueden tomar como la imagen de una aplicación lineal inyectiva. Podemos considerar entonces la matriz G asociada a dicha aplicación lineal. Definiremos dicha matriz como la matriz generadora del código.

(Cruz E. Borges, 2005)

Y al ser la matriz generatriz la asociada a una aplicación lineal inyectiva tiene rango máximo.

4.2. Definición (Matriz de control).

Llamamos matriz de control H de un código lineal a toda matriz tal que $GH^t = 0$ donde G es una matriz generatriz del código.

(Cruz E. Borges, 2005)

Nota:

El espacio vectorial $\mathbb{F}_q^n$ tiene un producto interno natural dado por:

$x \cdot y = x_1 y_1 + \cdots + x_n y_n \quad x, y \in \mathbb{F}_q^n$

4.3. Definición (Matriz de control de Paridad).

Primero tengamos en cuenta la siguiente definición:

Si L es un $[n, k]_q$ -código, el conjunto

$$L^{\perp} = \{ x \in \mathbb{F}_q^n : x \cdot c = 0 \; para \; todo \; c \in L \}$$

es el código dual de L.

Entonces la definición de matriz de control de paridad dice:

Sea L un $[n, k]_q$ -código. Una matriz H se dice matriz de paridad de L si es una matriz generadora de $L^{\perp}$.

Si se agrega un digito de control de paridad a un $[n, k]_q$ - código obtenemos un $[n+1, k]_q$ - código. Es posible agregar varios dígitos de control de paridad, de modo

que cada uno de estos realice un control de paridad entre un cierto conjunto de coordenadas.

Ejemplo.

Tomemos las palabras de $\mathbb{Z}_2^3$ y agreguemos 3 dígitos de control de paridad de la siguiente manera: el primero chequea las dos primeras coordenadas, el segundo chequea la primera y la tercera y el último chequea la segunda y la tercera. Es decir, si $x_1, x_2, x_3 \in \mathbb{Z}_2^3$, tomamos $x_1, x_2, x_3, y_1, y_2, y_3, \in \mathbb{Z}_2^6$ donde

$$x_1 + x_2 = y_1, \qquad x_1 + x_3 = y_2 \quad x_2 + x_3 = y_3 .$$

Luego, el código es

$$L = \{000000, 001011, 010101, 011110, 100110, 101101, 110011, 111000\}$$

Las ecuaciones anteriores son equivalentes al sistema

$$\begin{cases} x_1 + x_2 + y_1 = 0 \\ x_1 + x_3 + y_2 = 0 \\ x_2 + x_3 + y_3 = 0 \end{cases}$$

cuya matriz es

$$H = \left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \Longrightarrow Hx^T = 0$$

Luego el equivalente, L es el conjunto solución del sistema anterior, o sea

$$L = \{x \in \mathbb{Z}_2^6 : Hx^T = 0\}$$

En general, las ecuaciones como en este sistema se llaman ecuaciones de control de paridad y H es la matriz de paridad. Veremos que todo código lineal tiene una matriz de paridad, es decir, puede ser obtenido como el espacio solución de un sistema lineal homogéneo formado por ciertas ecuaciones de control de paridad. (A. Podesta Ricardo 2006)

5. Peso de Hamming.

Sea $x \in B^n$ con $x = (x_1, \ldots, x_n)$ entonces:

$$w(x) = |\{i \,/\, 1 \leq i \leq n \ \wedge \ x_i \neq 0\}|$$

es el peso de Hamming de x.

(Cruz E. Borges 2005)

6. Distancia de Hamming.

Sea $x, y \in B^n$ con $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$ entonces:

$$d(x, y) = |\{i \,/\, 1 \leq i \leq n \ \wedge \ x_i \neq y_i\}| = w(x - y)$$

es la distancia de Hamming entre x e y.

Nota 1: $w(x) = d(x, 0)$

Nota 2: Se puede comprobar que w es una norma y efectivamente d es la distancia inducida por la norma w.

Si los atributos son categóricos, entonces la distancia Hamming propone
una distancia entre dos cadenas:
La distancia es 1 por cada elemento diferente
0 por cada elemento idéntico en la cadena.

La distancia de toned a roses es 3. Se necesitan tres errores para transformar una palabra en otra.
La distancia de 1011101 a 1001001 es 2. Se necesitan dos errores para transformar un código en otro.
La distancia de 2173896 a 2233796 es 3. Se necesitan tres errores para transformar una cadena en otra.

Dicho de otra forma, define el mínimo número de sustituciones para convertir una cadena en otra. Solo es válida si ambas secuencias tienen la misma longitud.

(Cárdenas-Montes M.,2018)

6.1. Distancia de Hamming de un código.

Es la distancia mínima entre las palabras que componen el código
Ejemplo: {100, 111, 011}
mín{d(100, 111), d(100, 011), d(111, 011)} = mín{2, 3, 1} = 1
esto se refiere a la combinatoria.

6.2. Propiedades para la detección de errores.

Para detectar d errores de un bit entre dos palabras, es necesario un código con una distancia de Hamming de al menos $d + 1$

Observaciones:

De otra forma: Con una distancia de Hamming " d" se pueden detectar $d - 1$ errores
Por ejemplo $C = \{001, 010, 100\}$, d. Hamming $= 2$

Es importante además saber que:

• Un error aislado siempre se detecta porque no pertenecen al alfabeto del código
Un error en $001 \Rightarrow 101, 011, 000, \notin C$

• Dos errores aislados no se detectan porque pertenecerían al alfabeto del código
Dos errores en $001 \Rightarrow 111, 010, 100$. Dos pertenecen a C

6.3. Propiedades para la corrección de errores.

Para corregir d errores de un bit entre dos palabras es necesario un código con una distancia de Hamming de al menos $2d + 1$.
Es decir que con una distancia de Hamming de d se pueden corregir. $(d - 1)/2$ errores

Ejemplo:
C = {0000000000, 0000011111, 1111100000,1111111111},
d. Hamming = 5
Se pueden detectar d-1 = 5-1 = 4 errores
Se pueden corregir (d-1)/2 = 4/2 = 2 errores

Estas dos formas también se pueden analizar de los supuestos del ejemplo anterior con uno o dos errores por la combinatoria, y realizar asi las interpretaciones correctas.

6.4. Enfoque Informático.

Desde un enfoque geométrico tenemos que una palabra código de n bits puede ser representarse como un vector en un espacio de n dimensiones.
Así el código 010 puede representarse como $X = (\,0\;1\;0\,)$.

Por ejemplo, si se considera un código de repetición triple $(3, 1)$ (1 bit de mensaje, dos bits de repetición, palabra código de tres bits), las palabras código serían 000 y 111.
Cualquier mensaje recibido que no coincida con estas palabras código evidencia la presencia de errores.
Pero si los tres bits están errados (Se transmite 000 pero se recibe 111) entonces será imposible detectar el error.
Si se quiere corregir el error, asumimos entonces que al menos dos de los tres bits son correctos.

Entonces, 001 se decodifica como 000 y 101 se decodifica como 111. Esto corrige palabras con un solo error, pero para dos o tres errores la probabilidad (de ocurrir i errores en una palabra código de n bits) de que una palabra esté errada resulta aumentada siendo mayor.

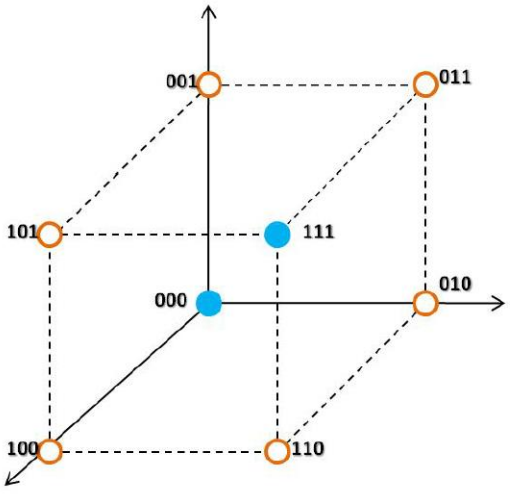


Figura que representa los vectores que representan palabras código de tres bits.

En total se grafican las 8 posibles combinaciones de palabras código y los puntos azules representan el código de repetición.

La distancia que separa dos puntos cualesquiera se reconoce como la distancia Hamming, que está relacionada con el poder de control de errores del código (fortaleza del código) y se define como el número de elementos diferentes entre dos puntos. Por ejemplo:

Si consideramos los vectores:

$$X = (\,0\;1\;1\,)$$

$$Y = (\,1\;1\;0\,)$$

Entonces la distancia Hamming=$d(X,Y) = 2$, porque el primer y tercer elemento son distintos.

La distancia mínima de un código en particular es la distancia Hamming más pequeña entre dos vectores de código válidos.

De esta forma, la detección de errores es posible siempre que se cumpla que el número de errores de transmisión en una palabra código sea menor a $d_{mín}$, por lo que la palabra errada no es un vector válido.

Si es mayor o igual, la palabra errada puede corresponder a un vector válido y el error no podría ser detectado.

La capacidad de corrección y detección de un código se define como:

Corrección:

$$t \leq \left[\frac{d_{mín} - 1}{2}\right]$$

Detección:

$$l \leq d_{mín} - 1$$

Para el ejemplo desarrollado, $d_{mín} = 3$, por lo que el código de repetición es capaz de corregir hasta 1 error por palabra y detectar hasta 2 errores. Evidentemente, la fortaleza del código depende del número de bits que se le agregan al mensaje original.

La distancia mínima de una codificación por bloque (repetición) se define como:

$$d_{mín} \leq n - k + 1$$

Donde:

n =Longitud de la palabra código.

k =Longitud del Mensaje original

De (n, k) −código. (Mezoa M., 2018)

6.5. Definición de la Distancia mínima.

Sea c un código, llamaremos distancia mínima d del código C a:

(Cruz E. Borges, 2005)

$$d = d(C) = \min\{d(x, y)/x, y \in C \wedge x \neq y\}$$
$$d = d(C) = min\{w(x) / x \in C \wedge x \neq 0\}$$

Se hace hincapié en que la definición de la distancia mínima es equivalente al peso mínimo.

7. Análisis de la Teoría de Códigos.

Una aplicación interesante de las matrices de Hadamard se refiere a la Teoría de Códigos. Esta es una rama de las Matemáticas e Ingeniería que tiene su origen en problemas de Ingeniería dada la aparición de las computadoras, es asi que por la cantidad de información que se almacena es que se ha desarrollado usando métodos matemáticos cada vez más sofisticados.

Los códigos se inventaron para corregir errores en canales de comunicación debidos al ruido. Supongamos que se envía un mensaje digitalizado (esto es, formado por 0's y 1's). Normalmente al enviar un 0 se recibe un 0, pero en algún caso por problemas de ruido como copiar un texto de internet y cuando lo pegamos obtenemos un espacio o un dígito modificado. Entonces esto hace que el 0 se recibe como un 1, esto representa un error de código.

Es importante para el receptor advertir la existencia del error y corregirlo. ¿Cómo puede hacerse esto? Podemos "alargar" el mensaje introduciendo signos de control. El ejemplo más usual para nosotros es el NIF con su letra de control al final de los números del DNI en España, pero en nuestro país al costado del Documento Nacional de Identidad DNI está un código separado por un guion para indicar cuántos homónimos tenemos, pero otro código que nos salva de la doble identidad es el Ubigeo necesario en grupos de votación.

Si se produce un error al introducir estos números (salvo que haya dos errores que se compensen, lo cual es altamente improbable) la letra de control lo marca, que en nuestro ejemplo lineal está al costado. Los códigos más sencillos son los lineales.

Veamos el siguiente ejemplo. Suponemos que estamos mandando una "palabra" digital de tres letras $u_1u_2u_3$, por ejemplo la palabra "UNO", que en la codificación se genera un código binario de alfabeto binario, donde cada letra puede tomar los valores 0 y 1, es decir, que los $u_i = 0$ ó 1.

Ahora tomamos una matriz de dimensión $n \times 3$ formada por $0's$ y $1's$, que será nuestra Matriz de control de Paridad. Por ejemplo, para $n = 3$, consideramos

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Alargamos el código poniendo

$$(u_1u_2u_3) \rightarrow \begin{bmatrix} I_{3\times3} \\ A \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \dots (1)$$

en base 2, donde $x_i = u_i$ para $i = 1, 2, 3$. De esta manera se obtiene la siguiente correspondencia entre los códigos que queremos enviar y los códigos alargados con tres cifras de control.

Como estamos realizando una multiplicación de matrices no cualquieras, sino una de ellas, es la matriz binaria, asociada con su matriz de control de paridad, que en su conjunto hace recurrencia a la combinatoria que se genera.

Esto significa, analizar donde cae el error en una letra, por ejemplo "U@O" se recibe "010". Entonces analizando todas las posibilidades, como las improbables x' y x'', consideremos:

(T. Domínguez, 2003)

000

100

010

001

110

101

011

111

Resolviendo de (1):

- Para

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Tenemos:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = x'$$

- Para

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Tenemos:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = x_1$$

- Para

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Tenemos:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = x_2$$

- Para

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Tenemos:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = x_3$$

- Para

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Tenemos:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} = x_4{}'$$

Entonces en la entrada a61 tenemos 2 y por las propiedades de cuerpo finito le corresponde 0, porque 2:2=1, con residuo 0.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = x_4$$

- Para

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

Tenemos:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} = x_5{}'$$

Lo mismo ocurre

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&1&1\\1&0&1\\1&1&0\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}=\begin{bmatrix}1\\0\\1\\1\\0\\1\end{bmatrix}=x_5$$

- Para

$$\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix}=\begin{bmatrix}0\\1\\1\end{bmatrix}$$

Tenemos:

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&1&1\\1&0&1\\1&1&0\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix}=\begin{bmatrix}0\\1\\1\\2\\1\\1\end{bmatrix}=x_6{}'$$

De forma similar

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&1&1\\1&0&1\\1&1&0\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix}=\begin{bmatrix}0\\1\\1\\0\\1\\1\end{bmatrix}=x_6{}'$$

- Para

$$\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix}=\begin{bmatrix}1\\1\\1\end{bmatrix}$$

Tenemos:

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&1&1\\1&0&1\\1&1&0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}=\begin{bmatrix}1\\1\\1\\2\\2\\2\end{bmatrix}=x''{}'$$

Lo cual se convertería a:

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}=\begin{bmatrix}1\\1\\1\\0\\0\\0\end{bmatrix}=x''$$

Obteniendo en resumen:

$$000 \to 000 \;\; 000 \to x'$$
$$100 \to 100 \;\; 011 \to x_1$$
$$010 \to 010 \;\; 101 \to x_2$$
$$001 \to 001 \;\; 110 \to x_3$$
$$110 \to 110 \;\; 110 \to x_4$$
$$101 \to 101 \;\; 101 \to x_5$$
$$011 \to 011 \;\; 011 \to x_6$$
$$111 \to 111 \;\; 000 \to x''$$

Es propicio señalar que por la combinatoria se generan 8 casos pero la matriz admite solo seis, quedando relegados la primera combinatoria 000000, y la última 111000 ya que es improbable que en 000, la palabra como fue enviada fue recepcionada, en este caso no analizaríamos nada, y también es improbable que todas las letras de la palabra estén erróneas y ocurra 111 y esto no admite en la teoría de códigos.

Así se obtiene una séxtupla en los 8 vectores fila con la siguiente correspondencia entre los códigos que queremos enviar y los códigos alargados con tres cifras de control para la detección o ubicación, estando éstos invertidos en cada x_i, es decir estando primero la terna de control y luego la terna de letras que queremos enviar ya corregida.

$$\perp\perp\perp\to \text{UNO UNO} \to x'$$
$$\text{e}\perp\perp\to \%\text{NO } \mathbf{U}\text{NO} \to x_1$$
$$\perp \text{e}\perp\to \text{U}\%\text{O U}\mathbf{N}\text{O} \to x_2$$
$$\perp\perp \text{e} \to \text{UN}\% \text{ UN}\mathbf{O} \to x_3$$
$$\perp\perp\perp\to \text{UNO UNO} \to x_4$$
$$\perp\perp\perp\to \text{UNO UNO} \to x_5$$
$$\perp\perp\perp\to \text{UNO UNO} \to x_6$$
$$\text{e e e} \to \%\%\%\mathbf{UNO} \to x''$$

Donde $\perp$ señala el dígito correcto, se dá la repetición, se cumple la paridad y no hay señal de error.

e para señalar la ubicación del error detectado.

% para corregir el error.

Vemos así que, como la palabra tiene 3 dígitos y el error está en la primera letra, al tercer lugar del dígito de control, se encontrará la letra correcta.

También podemos observar que los números de control pueden coincidir, pero lo hacen para palabras que se diferencian en los tres dígitos como los x_1, x_2, x_3 .Es decir donde hay un error, dos aciertos. De esta forma, si suponemos que sólo se produce un error (o dos), la secuencia de control lo advertirá, detectando y ubicando el error para luego corregirlo.
Pero también es importante que en caso de error el receptor sea capaz de regenerar la palabra enviada.

En efecto, en la séxtupla, la distancia mínima entre dos cadenas de este código es 3, se ubican en los vectores fila x_1, x_2, x_3, esto es conocido como la distancia de Hamming que aquí lo entenderemos como el número de cifras diferentes entre dos palabras o cadenas.

$$\text{e} \perp \perp \rightarrow \%\text{NO}\mathbf{U}\text{NO} \rightarrow x_1 \,; d = NOU = 3$$
$$\perp \text{e} \perp \rightarrow \text{U}\%\text{OU}\mathbf{N}\text{O} \rightarrow x_2; d = OUN = 3$$
$$\perp\perp \text{e} \rightarrow \text{UN}\%\text{UN}\mathbf{O} \rightarrow x_3; d = UNO = 3$$

De esta manera, si hay un error, sólo una de las palabras correctas es posible, por ejemplo si fuera en el x_2 se señala el error que se produce, la detección y reconstrucción de la palabra.

Los dos problemas que evidenciamos son la detección de errores y la reconstrucción de la palabra correcta, constituyen la clave de la teoría de códigos. Elegir la matriz que es la de comprobación de la paridad, que resulte la más apropiada en este caso, que también es útil para diseñar otros tipos de códigos. Esto constituye un problema de gran interés y dificultad.

Para la reconstrucción de palabras es importante que la distancia de Hamming d sea lo mayor posible, para corregir muchos errores a la vez.

Es fácil ver que un código con distancia mínima d puede corregir hasta $[(d-1)/2]$ errores. Para nuestro caso $\left[\frac{3-1}{2}\right] = 1$, en lo cual corregimos un solo error.

Es por eso que en el ejemplo anterior se puede corregir un error, pero no hay garantía para corregir dos errores.

En general entonces, para este estudio y en lo posterior:

Un (n, M, d) −código es un conjunto de:

M: Cantidad de vectores de dimensión n, donde $M = 8$ vectores que representan la combinatoria.

n : Dimensión del vector, donde $n = 6$, son las séxtupla que se generan, donde n se refiere a la longitud de la cadena codificada

d :Distancia de Hamming, donde $d = 3$, tal que dos vectores cualesquiera difieren en, al menos, d coordenadas, 3 coordenadas.

Notemos además que se produce una composición de vectores en la séxtupla, donde existen dos vectores o ternas, una es el vector de control y la otra es el vector recibido. Quedando el código (6, 8, 3)

Lógicamente interesa que n sea lo más pequeño posible para que el código no se alargue de más, lo cual es una gasto computacional excesivo.

Y en cambio M y d sean lo más grande posible; M porque representa todas las combinaciones probables para poder detectar el error con toda las posibilidades existentes, y d, la distancia de Hamming, para contar todos los errores que pueden producirse.

8. Distancia de Levenshtein.

La distancia de Levenshtein es también llamada distancia de edición o distancia entre palabras.

Esta distancia mide el número mínimo de operaciones para transformar una cadena de caracteres en otra cadena de caracteres.

Las operaciones válidas para esta medida son la inserción, la eliminación y la sustitución de un carácter.

La distancia de Levenshtein es una generalización de la distancia de Hamming.

En la distancia de Hamming ambas cadenas tienen que tener la misma longitud y por lo tanto solo admite la operación de sustitución.

Existen otras generalizaciones de la distancia de Hamming como la distancia de Damerau-Levenshtein. (Cárdenas-Montes M., 2018)

Esta distancia de edición consta de:

- Medida similaridad entre dos hileras:

Hilera fuente (f)

Hilera meta (m).

- La similaridad entre dos palabras se calcula como el mínimo número de operaciones que transforma a una de las palabras en la otra (transformar f en m).
- Las tres operaciones son:

  Eliminación(E)

  Inserción(I)

  Substitución(S)
- A más distancia Levenshtein ~ Mayor diferencia en las hileras.
- Entre más cerca de cero más parecidas son las hileras.

Algoritmo de Leveinshtein:

1. El tamaño de la hilera f es $\boldsymbol{x}$, y el tamaño de la hilera m es y. Si $x = 0$, retornar y; y si $y = 0$, retornar x.
2. Construir una matriz con $y + 1$ filas y $x + 1$ columnas. Inicializar la primer fila de la matriz con la secuencia $0, 1, 2, \dots, x$; y la primer columna de la matriz con la secuencia $0, 1, 2, \dots, y$.
3. Colocar cada carácter de la hilera f en su correspondiente celda i (i va de 1 a x).
4. Colocar cada carácter de la hilera m en su correspondiente celda j (j va de 1 a y).
5. Si $f(i)$ es igual a $m(j)$ el costo de la celda es 0.
6. Si $f(i)$ es diferente de $m(j)$ el costo de la celda es 1.
7. El valor de la celda $d(i, j)$ es el mínimo de:
8. Valor de la celda $(i-1, j) + 1$. (eliminación)
9. Valor de la celda $(i, j-1) + 1$. (inserción)
10. Valor de la celda $(i-1, j-1)$ + costo de celda (i, j). (sustitución)
11. La distancia es la celda $d(x, y)$.

    Tendríamos que aplicar las estrategias así, para no pasar de los elementos que pueden contener:

	0	1	2	3	4
0		c	o	m	a
1	c	S	E	E	E
2	e	I	S	E	E
3	n	I	I	S	E
4	a	I	I	I	**S**

E: Por eliminación

I: Por inserción

S: Por sustitución

Resolvemos según el algoritmo:

1. CENA=4→y

   COMA=4→x
2. Como

   X+1=5

   Y+1=5

   Entonces 0,1,2,3,4
3. Colocar j,…→en1,2,3,4

   Entonces CENA y=m→j
4. Colocar i…→en1,2,3,4

   Y COMA x=f→i
5. Si f(i)=m(j) Entonces costo 0. Cada vez que encontremos la misma letra
6. Si f(i)≠m(j) Entonces costo 1. Cada vez que encontremos diferente letra
7. Valor de Celda d(i,j) mín:
8. Siguiendo las pautas del punto 7 combinadas con 5 y 6

   Para la primera fila j=1

- (1,1)→ sustitución

  (0,0)+(1,1)=0+0=0
- (2,1) → eliminación

  (1,1)+1=0+1=1
- (3,1) → eliminación

  (2,1)+1=1+1=2
- (4,1) → eliminación

  (3,1)+1=2+1=3

  Para la segunda fila j=2
- (1,2)→ inserción

  (1,1)+1=0+1=1
- (2,2) → sustitución

  (1,1)+1=0+1=1
- (3,2) → eliminación

(2,2)+1=1+1=2

- (4,2) → eliminación

  (3,2)+1=2+1=3

  Para la tercera fila j=3

- (1,3)→ inserción

  (1,2)+1=1+1=2

- (2,3) → inserción

  (2,2)+1=1+1=2

- (3,3) → sustitución

  (2,2)+1=1+1=2

- (4,3) → eliminación

  (3,3)+1=2+1=3

  Para la cuarta fila j=4

- (1,4)→ inserción

  (1,3)+1=2+1=3

- (2,4) → inserción

  (2,3)+1=2+1=3

- (3,4) → inserción

  (3,3)+1=2+1=3

- (4,4) → sustitución

  (3,3)+0=2+0=2

9. Por tanto aplicando el algoritmo resulta:

   Siendo así 2, la Distancia de Leveinshtein entre estas dos palabras.

	0	1	2	3	4
0		c	o	m	a
1	c	0	1	2	3
2	e	1	1	2	3
3	n	2	2	2	3
4	a	3	3	3	**2**

Intuitivamente:

Se tienen las hileras: f = coma y m = coma. $DL(f, m) = 0$, porque no hay que realizar transformaciones.

Se tienen las hileras: f = coma y m = cena. $DL(f, m) = 2$, porque con dos sustituciones se transforma f en m.

Siendo DL: Distancia de Leveinshtein

Es decir la palabra COMA se convierte en CEMA, donde corregimos la O por la E y CEMA se convierte en CENA, donde cambiamos la M por la N.

(Prof. M.Sc. Ramírez Kryscia, 2017)

Capítulo III

Cotas de Plotkin y Teorema de Levenshtein

1. Teorema de códigos óptimos para matriz de Hadamard.

Para establecer las implicancias de este teorema haremos una breve introducción y que será importante para la Construcción de Leveinshtein.

Tomamos como código bloque:

$$A = 111\ 110$$

$$B = 111\ 001$$

Donde se observa claramente que $n = 6$; que:

$$d_{AB} = 3$$

Y por ello la distancia mínima entre los códigos A y B es $d = 3$, y entonces $(d - 1)$ es el error detectado de código, es decir 2 posibles errores ubicar a la vez con un solo envío.

Primero debemos entender que como toda comparación se hace cada dos códigos para ver si se recibe lo que se envió así falle en un dígito, pero se puede corregir. Además una distancia d entre códigos sugiere la posibilidad de corrección con la fórmula $\frac{1}{2}(d - 1)$. Así:

$(n, M_1, d_1; 2)$ código $C_1 = A$

$(n, M_2, d_2; 2)$ código $C_2 = B$

$A = 111\ 110; n_1 = 6; M_1 = 2; d_1 = 3; q = 2$

$B = 111\ 001; n_2 = 6; M_2 = 2; d_2 = 3;\ q = 2$

La teoría dice $M_2 \geq M_1$. Y es lógico pensarlo puesto que estamos analizando dos palabras, dos códigos $M_2 = M_1$.

Segundo, de lo que se trata es de construir un nuevo código que será encerrado en la computadora: Un nuevo código C $(n, M_1, d; 2)$ mediante álgebra, así:

Entonces del código A:

$a_1 = 1$

$a_2 = 1$

$a_3 = 1$

$a_4 = 1$

$a_5 = 1$

$a_6 = 0$

Lo mismo para el código B.

Luego este nuevo código tendrá una extensión de: $n = a_1 n_1 + a_2 n_2$

Reemplazando $n = 1n_1 + 1n_2; n = 6 + 6; n = 12.$

Para d será: $d \geq a_1 d_1 + a_2 d_2$

Reemplazando : $d \geq 1 \cdot 3 + 1 \cdot 3; d \geq 6$

Tercero pegamos a_1 copias de C_1, sitio por sitio; seguimos con a_2 copias de C_2

$C = (111\ 110)(111\ 001) = C = 111\ 110111\ 001$……..(*)

Donde: $n = 12; M = 2; d = 6; q = 2$

Nótese en (*) que si hubiésemos comparado el mismo código A con A era el correcto sin error porque esta duplicado y la $d = 0$ sin error.

Generalizando así para códigos q-arios:

C_1	C_1	$\cdots$	C_1	C_2	C_2	$\cdots$	C_2
a_1				a_2			

Así para otros códigos q-arios diferentes de los binarios se repetirá q veces el primer código con q veces el siguiente para generar toda la combinatoria y abarcar todos los casos de posible error. Entonces en el código binario dos veces, dos veces el primer codígo y así con el siguiente.

Recordemos que para un código lineal existen códigos alargados para la detección y corrección de errores porque un código en bloques comprende códigos lineales.
Cuarto paso, no olvidemos por último restar $M_2 - M_1$ filas de C_2 para obtener el código deseado que en nuestro caso sería cero tal que el código como fue enviado fue recibido la palabra.

Finalmente denotamos este código como $a_1C_1 \oplus a_2C_2$ reemplazando $1(111\ 110) \oplus 1(111\ 001) = 111\ 110 \oplus 111\ 001$
Se opera tomando en cuenta que $\oplus$, el operador más en un círculo N-ario se usa uno a uno similar al producto Knocker pero en este caso en su linealidad solo se usa para juntar los posibles códigos que van acompañados de otro dígito y allí se detectaran los errores, por no ser exactamente iguales al patrón en la cadena generada, guardando relación con el tercer paso.
Y así todos estos elementos conforman el cuerpo o la familia del código en cuestión.

1.1. Teorema de códigos óptimos de un caso par.

La existencia de una matriz de Hadamard de orden 4t implica la existencia del código óptimo siguiente: (Hedayat y W.D. Wallis 1978)

$$(4t, 8t, 2t; 2)$$

Con esto construiremos códigos binarios que confluyen o se entrelazan con los Límites de Plotkin.

Demostración:
Primero entendamos la necesidad de establecer un parámetro t que relacione todos los elementos del código, y que no dependa de n.

Entonces para el caso A. Hedayat y W.D.Walis.

$$(4t, 8t, 2t; 2); (12{,}24{,}6; 2); M(2d, d) = 4d$$

Donde el parámetro $t = 3$

Se realiza construyendo la matriz mediante el método Sylvester y se trata del caso de la igualdad.

2. Cotas de Plotkin

Para (no necesariamente) códigos binarios (n, M, d), Plotkin descubrió las siguientes cotas para el número M de palabras de código:

(Álvarez J, A. Armario, M. D. Frau, E. Martin, y A. Osuna 2007)

$M \leq 2\left[\frac{d}{2d-n}\right]$ si d es par y $d \leq n < 2d$,……***(A)***

$M \leq 2n$ si d es par y $n = 2d$,……***(B)***

Demostración:

Para M par, es decir la cantidad de palabras par vamos a calcular

$$S = \sum_{u,v \in C} d(u, v)$$

Siendo $u\ y\ v$ los dígitos del código como $a_1; a_2$ que aplicando la derivada de la sumatoria se hace por la regla de la cadena (derivada de lo de afuera por derivada de lo de adentro) sacando así las cadenas de códigos alargados como en los códigos lineales, similar a los códigos cíclicos que se comportan de manera algebraica. Generando así las combinaciones.

Por una parte hay $M(M - 1)$ parejas debido a la comparación de códigos cuando obtenemos un nuevo código y la distancia entre ellas es al menos d; porque se van generando copias, entonces se tiene $S \geq M(M - 1)d$.

Y claro está esto no debe excedes a dicha sumatoria porque es el conjunto de códigos o lenguaje.

Por otra parte, pongamos las M palabras como filas de una matriz $M \times n$. Como se hace en la matriz generadora del código

Supongamos que la columna i contiene x_i $0's$ y $M - x_i$ $1's$. Completándose así para la comparación esta vez matricialmente.

Entonces esta columna contribuye $2x_i(M - x_i)$ a la suma. Y esto se observa en cada fila.

Por tanto $S = \sum_{i=1}^{n} 2x_i(M - x_i)$. Y de esta manera completamos todas las posibles combinatorias para detectar los errores.

Por cálculo elemental se prueba que el máximo de esta función de n variables se alcanza para $x_i = M/2$, o sea el máximo es la mitad de las palabras para separar y realizar la comparación una a una de los dígitos del código; obteniéndose de esta forma
$S \leq nM^2/2$, esto para generalizar en cada fila de código y hacer el mismo proceso.

De esta manera comparando la primera relación de M con d y luego la de M con n tenemos entonces:

$$M(M - 1)d \leq \frac{nM^2}{2}$$

Desarrollando:

$$\frac{M(M - 1)}{M^2} \leq \frac{n}{2d}$$

$$\frac{(M - 1)}{M} \leq \frac{n}{2d}$$

$$1 - \frac{1}{M} \leq \frac{n}{2d}$$

$$1 - \frac{n}{2d} \leq \frac{1}{M}$$

$$\frac{2d - n}{2d} \leq \frac{1}{M}$$

$$(2d - n)M \leq 2d$$

lo que implica:

$$M \leq \frac{2d}{2d - n}$$

$$\frac{M}{2} \leq \frac{d}{2d - n}$$

$$\frac{M}{2} \leq \left[\frac{d}{2d - n}\right]$$

Separando el factor:

$$M \leq 2\left[\frac{d}{2d-n}\right]$$

Ejemplo, si $n = 6$ obtenemos $d = 4$, el número máximo de palabras es $2[4/2] = 4$.

2.1. PRIMER CASO: Usando la teoría de Domínguez: De longitud y distancia considerados en un Intervalo.

Sea el código $(12,4,8;2)$ donde $n = 12; M = 4; d = 8; q = 2$ cuya condición para la cota es $8 \leq 12 < 2(8)$ entonces se cumple el intervalo establecido $8 \leq 12 < 16$.

Seguidamente comprobaremos para M:

$M \leq 2\left[\frac{8}{2(8)-12}\right]$; $M \leq 2\left[\frac{8}{16-12}\right]$; $M \leq 2\left[\frac{8}{4}\right]$; $M \leq 4$.

Lo cual en el código propuesto se cumple para la igualdad $M = 4$.

2.2. SEGUNDO CASO: Usando la teoría de A. Hedayat y W.D. Walis: De longitud y distancia Exacta.

Sea el código $(12,24,6;2)$ donde $n = 12; M = 24; d = 6; q = 2$ cuya condición para la cota es que $n = 2(6)$ entonces se cumple para la igualdad $n = 12$.

Seguidamente comprobaremos para M:

$M \leq 2[12]$; $M \leq 24$.

Lo cual en el código propuesto se cumple para la igualdad $M = 24$.

2.3. Análisis de la Cota de Plotkin y sus implicancias.

A simple vista la cota de Plotkin es importante para conseguir códigos en los que se alcance cada cota. Así para un n y d fijados se obtiene un código con el número máximo de palabras posibles.

Para conseguir este código trabajamos en ambos casos con las igualdades.

Por lo tanto todas las palabras tienen que estar a distancia d y en cada columna debe haber el mismo número de ceros que de unos. Una herramienta fundamental para conseguirlo son las matrices de Hadamard.

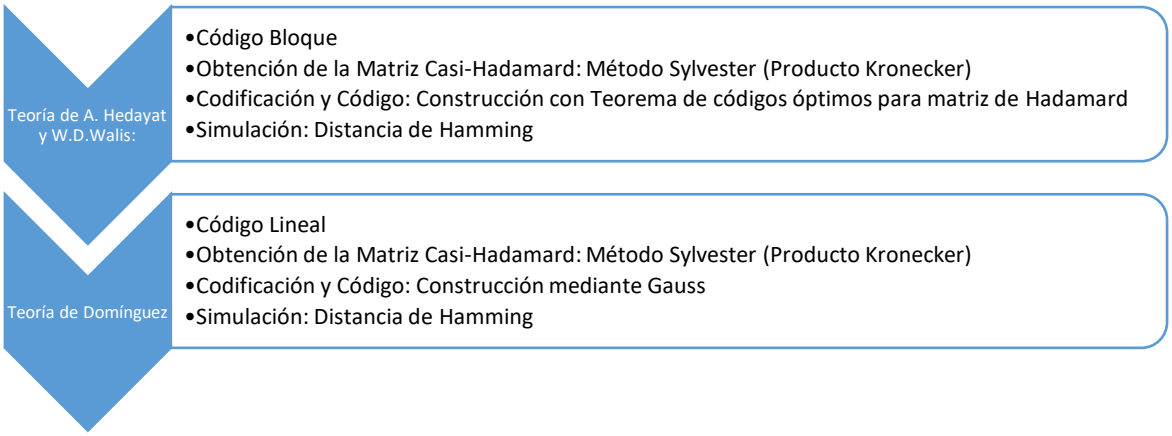
3. Teorema de Levenshtein.

Si existen las matrices de Hadamard apropiadas, cuando d es un número par, entonces las cotas de Plotkin, tienen la forma: (T. Domínguez, 2003)

$$M(n,d) = 2\left[\frac{d}{2d-n}\right], d \leq n < 2d,$$
$$M(2d,d) = 4d;$$

Levenshtein demostró que si la conjetura de Hadamard es verdadera, entonces el límite de Plotkin es sostenido. Debido a que las matrices de Hadamard generan códigos de Hadamard esto ha asegurado la teoría de códigos.

Entonces resulta inmediato preguntarse cómo deben de ser las matrices de Hadamard, como unimos con las cotas; falta conocer el proceso de construcción dado que la Matriz de Hadamard tiene muchos métodos para obtenerse, siendo para esta investigación el más adecuado el Método Sylvester relacionado con el producto Kronecker. Cabe destacar que esto se llevará a cabo con la definición de matriz de Hadamard y el procedimiento para obtener matrices casi-Hadamard.



Esquema 2: Modelos matemáticos de corrección de códigos binarios

4. Construcción de Códigos máximos-óptimos caso par.

4.1. Construcción de matriz de código usando Teoría de Domínguez.

Consideramos la matriz de Hadamard de orden 8 normalizada con los -1's convertidos en 1's y los 1's en 0's, esto es,

H_8=

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

Eliminamos la primera columna

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

Quitamos la primera columna y posteriormente las filas que comienzan por uno. Es decir la segunda, tercera, cuarta y sexta fila, para quedamos con la primera, quinta, séptima y octava fila. Seguidamente volvemos a quitar la primera columna

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

Y así obtenemos una matriz casi-Hadamard.

Ahora pegamos la matriz resultante consigo misma, igual como dice la construcción de un nuevo código M, para obtener el código (12, 4, 8)

$$\left(\begin{array}{cccccc|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}\right)$$

que puede corregir hasta tres errores lo cual será demostrado más adelante.

4.2. Construcción de matriz de código usando Teoría de A. Hedayat y W.D.Walis.

Si H es una matriz normalizada de orden $4t$.

Cuando $t = 1$

$$H' = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Cambiando la segunda por la tercera fila, tenemos:

$$H'' = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Cuando $t = 2$

$$n = 4t = 8$$

$H_8 =$

1	1	1	1	1	1	1	1
1	-1	1	-1	1	-1	1	-1
1	1	-1	-1	1	1	-1	-1
1	-1	-1	1	1	-1	-1	1
1	1	1	1	-1	-1	-1	-1
1	-1	1	-1	-1	1	-1	1
1	1	-1	-1	-1	-1	1	1
1	-1	-1	1	-1	1	1	-1

Y cuando $t = 3$

$$n = 4t = 4(3) = 12$$

Observándose en la matriz la primera fila y la primera columna de 1´s, así como la mitad de entradas positivas y la otra mitad negativas.

Luego las $8t$ filas de dos matrices:

$$W_{4t}^{(1)} = \frac{1}{2}(J + H)$$

$$W_{4t}^{(2)} = \frac{1}{2}(J - H)$$

Forman un código $(4t, 8t, 2t; 2)$

Cuando $t = 1$

$$(4,8,2;2) \approx (n, M, d; q)$$

Entonces

$$J = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Y si $t = 1$

$$W_{4(1)}^{(1)} = \frac{1}{2}\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}\right)$$

$$W_{4(1)}^{(1)} = \frac{1}{2}\begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 0 & 2 & 0 \\ 2 & 2 & 0 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

$$W_4^{(1)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

De la misma forma se construye $W_4^{(2)}$.

Entonces construyamos nuestra matriz del código, teniendo en cuenta que el parámetro $t = 3$:

El orden de H es $4t$ osea $4(3) = 12$. Y necesitaremos:

$H_{12} =$

1	1	1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	1	1	1	-1	-1	-1	1	-1
1	-1	-1	1	-1	1	1	1	-1	-1	-1	1
1	1	-1	-1	1	-1	1	1	1	-1	-1	-1
1	-1	1	-1	-1	1	-1	1	1	1	-1	-1
1	-1	-1	1	-1	-1	1	-1	1	1	1	-1
1	-1	-1	-1	1	-1	-1	1	-1	1	1	1
1	1	-1	-1	-1	1	-1	-1	1	-1	1	1
1	1	1	-1	-1	-1	1	-1	-1	1	-1	1
1	1	1	1	-1	-1	-1	1	-1	-1	1	-1
1	-1	1	1	1	-1	-1	-1	1	-1	-1	1
1	1	-1	1	1	1	-1	-1	-1	1	-1	-1

$J_{12} =$

1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1

$-H_{12} =$

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	1	-1	-1	-1	1	1	1	-1	1
-1	1	1	-1	1	-1	-1	-1	1	1	1	-1
-1	-1	1	1	-1	1	-1	-1	-1	1	1	1
-1	1	-1	1	1	-1	1	-1	-1	-1	1	1
-1	1	1	-1	1	1	-1	1	-1	-1	-1	1
-1	1	1	1	-1	1	1	-1	1	-1	-1	-1
-1	-1	1	1	1	-1	1	1	-1	1	-1	-1
-1	-1	-1	1	1	1	-1	1	1	-1	1	-1
-1	-1	-1	-1	1	1	1	-1	1	1	-1	1
-1	1	-1	-1	-1	1	1	1	-1	1	1	-1
-1	-1	1	-1	-1	-1	1	1	1	-1	1	1

Formaremos el código de $(12,24,6,2)$ y la matriz nos saldrá de 12 columnas y 24 filas (12 lo componen $W_{12}^{(1)}$ y 12 de $W_{12}^{(2)}$, esto induce el Teorema anterior de códigos óptimos).

$$W_{4(3)}^{(1)} = \frac{1}{2}(J_{12} + H_{12})$$

$$W_{4(3)}^{(1)} = \frac{1}{2}\begin{pmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 0 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 2 \\ 2 & 2 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 0 \\ 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 2 & 2 & 2 \\ 2 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 2 & 2 \\ 2 & 2 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 2 \\ 2 & 2 & 2 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 \\ 2 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 2 & 0 & 0 \end{pmatrix}$$

Construimos una parte del código:

$$W_{12}^{(1)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Ahora:

$$W_{4(3)}^{(2)} = \frac{1}{2}(J_{12} - H_{12})$$

$$W_{4(3)}^{(2)} = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 2 \\ 0 & 2 & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 2 & 2 \\ 0 & 2 & 0 & 2 & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 2 \\ 0 & 2 & 2 & 0 & 2 & 2 & 0 & 2 & 0 & 0 & 0 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 2 & 0 & 2 & 2 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 2 & 0 & 2 & 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 2 & 2 & 0 & 2 \\ 0 & 2 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 2 & 2 & 2 & 0 & 2 & 2 \end{pmatrix}$$

Construimos la segunda parte del código:

$$W_{12}^{(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Entonces:

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

Queda formado el código $(12, 24, 6;\ 2)$. De longitud de palabra 12, 24 el número de palabras la distancia de Hamming 6.

5. Modelación de la Codificación de Hadamard usando Matlab y Microsoft Excel.

5.1. Modelación usando la Teoría de Domínguez.

En este caso el que ingresa es un código formado por cuatro palabras.

Sea (1 00 1) este código, entonces de operar:

$$(\,1\;00\;1\,)_{1\times M}\left(\begin{array}{cccccc|cccccc} 0 & 0\,0 & 0\,0 & 0 & 0\,0 & 0 & 0 & 0\,0 & 0\,0 & 0 & 0\,0 & 0 \\ 1 & 1\,1 & 0\,0 & 1 & 1\,1 & 0 & 1 & 1\,1 & 0\,0 & 1 & 1\,1 & 0 \\ 1 & 0\,1 & 1\,1 & 0 & 0\,1 & 1 & 1 & 0\,1 & 1\,1 & 0 & 0\,1 & 1 \\ 0 & 1\,0 & 1\,1 & 1 & 1\,0 & 0 & 0 & 1\,0 & 1\,1 & 1 & 1\,0 & 0 \end{array}\right)_{M\times n}$$

Queda:

$$(\,1\;00\;1\,)_{1\times 4}\left(\begin{array}{cccccc|cccccc} 0 & 0\,0 & 0\,0 & 0 & 0\,0 & 0 & 0 & 0\,0 & 0\,0 & 0 & 0\,0 & 0 \\ 1 & 1\,1 & 0\,0 & 1 & 1\,1 & 0 & 1 & 1\,1 & 0\,0 & 1 & 1\,1 & 0 \\ 1 & 0\,1 & 1\,1 & 0 & 0\,1 & 1 & 1 & 0\,1 & 1\,1 & 0 & 0\,1 & 1 \\ 0 & 1\,0 & 1\,1 & 1 & 1\,0 & 0 & 0 & 1\,0 & 1\,1 & 1 & 1\,0 & 0 \end{array}\right)_{4\times 12} =$$

$$0000 = 0$$
$$0001 = 1$$
$$0000 = 0$$
$$0001 = 1$$
$$0001 = 1$$
$$0001 = 1$$
$$0000 = 0$$
$$0001 = 1$$
$$0000 = 0$$
$$0001 = 1$$
$$0001 = 1$$
$$0001 = 1$$

$$= (0\;1\;0\;1\;\;1\;1\;0\;1\;\;0\;1\;1\;1\,)_{1\times n}$$

Osea:

$$= (0\;1\;0\;1\;\;1\;1\;0\;1\;\;0\;1\;1\;1\,)_{1\times 12}$$

Observándose otras palabras códigos M diferentes de 1 0 0 1 que comparando uno a uno con el código original:

Suponiendo un ideal:

$$= (\mathbf{1}\,\mathbf{0}\,0\,1 \;\; 1\,\mathbf{0}\,0\,1 \;\; \mathbf{1}\,\mathbf{0}\,\mathbf{0}\,1)_{1\times 12}$$

Tenemos:

$$= (\mathbf{0}\,\mathbf{1}\,0\,1 \;\; 1\,\mathbf{1}\,0\,1 \;\; \mathbf{0}\,\mathbf{1}\,\mathbf{1}\,1)_{1\times 12}$$

$$F\;F\;V\;V \;\; V\;F\;V\;V \;\; F\;F\;F\;V$$

Vemos que en el primer código se corrigen 2 errores, en el segundo código se corrige hasta un error y en el último código se corrige a lo más tres errores.

Entonces para calcular el valor de d exacto vamos a la ecuación del caso A de la desigualdad en la cota:

$4 \leq 2\left[\frac{d}{2d-12}\right]$ si d es par

$8d - 48 \leq 2d$

$6d \leq 48$

$d \leq 8$

Cumple $d \leq 12 < 2d$

Ahora corroboramos nuestro análisis con la fórmula para corrección de errores de la distancia de Hamming $\frac{(d-1)}{2}$

$\frac{(8-1)}{2} = \frac{7}{2} = 3{,}5.$

Esto quiere decir que corrige a lo más tres errores.

Luego, dá como resultado las posibilidades de corregir el error según sea el caso, así:

$$= (0\,1\,0\,1 \;\; 1\,1\,0\,1 \;\; 0\,1\,1\,1)_{1\times 12}$$

Donde el primer código indica posibilidad intermedia de error, el segundo representa una posibilidad mínima de error, casi correcto; y el último código es la posibilidad máxima de error.

Codificación usando Dominguez:

Usamos "MATLAB" y hacemos corer el programa con los códigos que van saliendo, asi:

```
>> Z=[0 0 0 0 0 0 0 0 0 0 0 0;1 1 1 0 0 1 1 1 1 0 0 1;1 0 1 1 1 0 1 0 1 1 1 0;0 1 0 1 1 1 0 1
0 1 1 1]
Z =
    0    0    0    0    0    0    0    0    0    0    0    0
    1    1    1    0    0    1    1    1    1    0    0    1
    1    0    1    1    1    0    1    0    1    1    1    0
    0    1    0    1    1    1    0    1    0    1    1    1
>> A=[1 0 0 1]
A =
    1    0    0    1
>> W=A*Z
W =
    0    1    0    1    1    1    0    1    0    1    1    1
>> B=[0 1 0 1]
B =
    0    1    0    1
>> Y=B*Z
Y =
    1    2    1    1    1    2    1    2    1    1    1    2
>> C=[1 1 0 1]
C =
    1    1    0    1
>> X=C*Z
X =
    1    2    1    1    1    2    1    2    1    1    1    2
>> D=[0 1 1 1]

D =
    0    1    1    1
>> V=D*Z
V =
    2    2    2    2    2    2    2    2    2    2    2    2
>> E=[1 0 1 1]
E =
```

```
    1     0     1     1
>> U=E*Z
U =
    1     1     1     2     2     1     1     1     1     2     2     1
>> F=[1 0 1 0]
F =
    1     0     1     0
>> T=F*Z
T =
    1     0     1     1     1     0     1     0     1     1     1     0
>> G=[1 1 1 0]
G =
    1     1     1     0
>> S=G*Z
S =
    2     1     2     1     1     1     2     1     2     1     1     1
>> H=[0 0 0 0]
H =
    0     0     0     0
>> R=H*Z
R = 0     0     0     0     0     0     0     0     0     0     0     0
```

5.2. Modelación usando la Teoría de A. Hedayat y W.D.Walis.

Este es el caso de un código bloque conformado por 24 caracteres o 24 palabras. Este metodo es algo diferente por ser mde mucha data.

Sea un bloque de cuatro códigos cada uno de 6 dígitos:

$$A = 1\,1\,1\,1\,1\,0$$
$$B = 1\,1\,1\,0\,0\,1$$

La distancia de Hamming es seis que coincide con la longitud de la cadena de cada código lineal, esto siempre en códigos bloque, entonces compararemos dos códigos lineales a la vez por estar en bloque. Es decir tenemos seis códigos lineales que juegan cada par.

La cadena lineal final de n=24 es una cadena de copias, que resultan de trabajarse dos a dos, A y B cadenas originales para encontrar dos errores. En el punto 4.1 del presente capítulo se explica cómo es que se generan las copias de un código, luego el código obedece a:

$$a_1C_1 \oplus a_2C_2$$

Reemplazando

$$1(111\ 110) \oplus 1(111\ 001) = 111\ 110 \oplus 111\ 001$$

$$AB = (1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)$$

Extendiendo el código haciendo tantas copias como longitud de la cadena original lo permitan hasta llegar a 24, es decir:

$a_1C_1 \oplus a_2C_2 \oplus a_3C_1 \oplus a_4C_2$

Obteniéndose:

$$1(111\ 110) \oplus 1(111\ 001) \oplus 1(111\ 110) \oplus 1(111\ 001)$$
$$= 111\ 110 \oplus 111\ 001 \oplus 111\ 110 \oplus 111\ 001$$

$$AB = (1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)$$

Que es lo mismo decir que el código alargado es el ABAB porque el operador $\oplus$ trabaja de manera asociada conformando un código bloque.

Haciendo esto quedaría para el código A y B

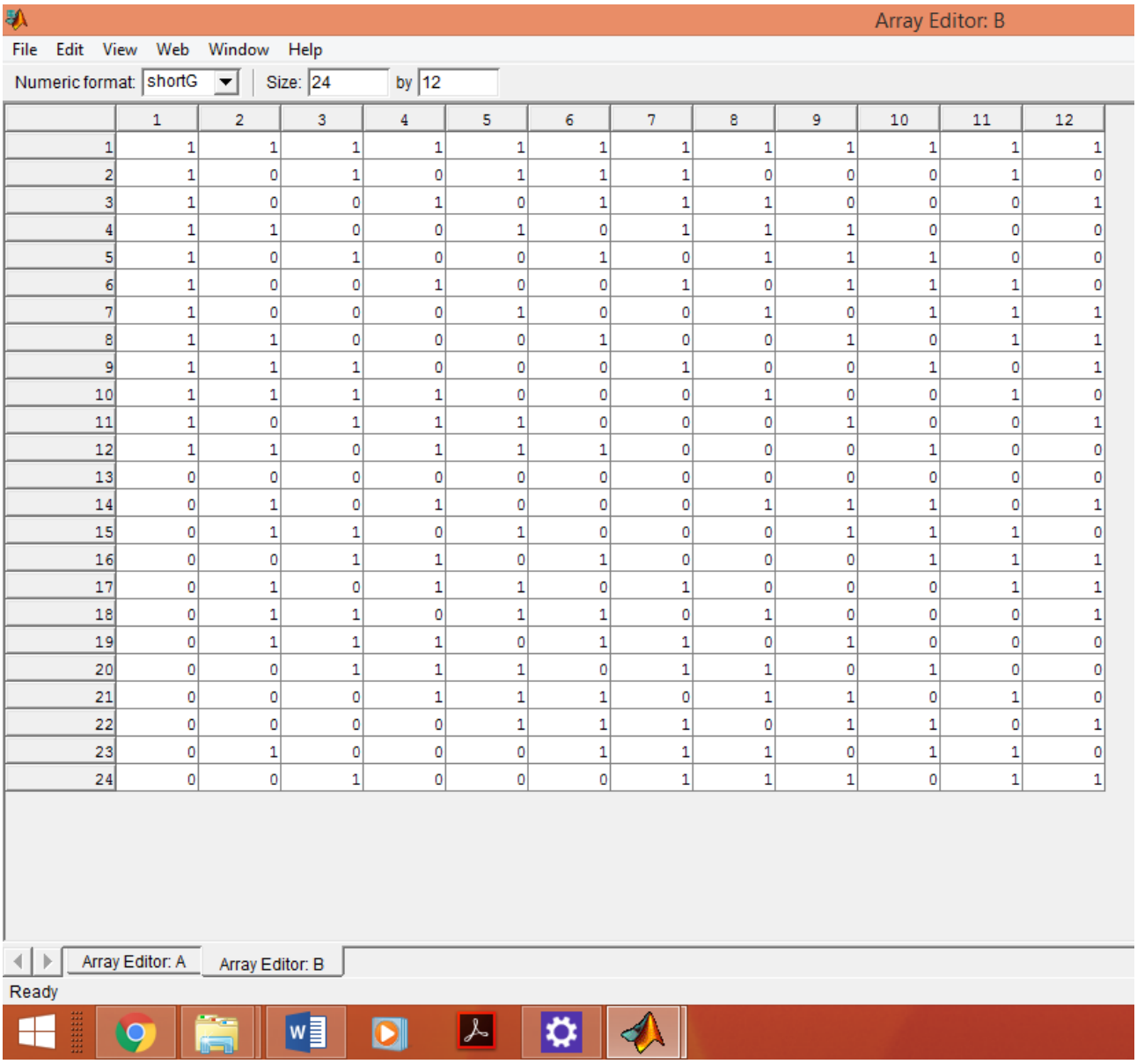
$$(1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)_{1\times 24}$$

Luego la matriz:

$$\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1
\end{pmatrix}_{24\times 12}$$

Seguidamente, ingresando en Matlab para calcular el producto tenemos:

Imagen 1: Captura de matlab del código lineal alargado ABAB de 1x24



	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	0	1	0	1	1	1	0	0	0	1	0
3	1	0	0	1	0	1	1	1	0	0	0	1
4	1	1	0	0	1	0	1	1	1	0	0	0
5	1	0	1	0	0	1	0	1	1	1	0	0
6	1	0	0	1	0	0	1	0	1	1	1	0
7	1	0	0	0	1	0	0	1	0	1	1	1
8	1	1	0	0	0	1	0	0	1	0	1	1
9	1	1	1	0	0	0	1	0	0	1	0	1
10	1	1	1	1	0	0	0	1	0	0	1	0
11	1	0	1	1	1	0	0	0	1	0	0	1
12	1	1	0	1	1	1	0	0	0	1	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0
14	0	1	0	1	0	0	0	1	1	1	0	1
15	0	1	1	0	1	0	0	0	1	1	1	0
16	0	0	1	1	0	1	0	0	0	1	1	1
17	0	1	0	1	1	0	1	0	0	0	1	1
18	0	1	1	0	1	1	0	1	0	0	0	1
19	0	1	1	1	0	1	1	0	1	0	0	0
20	0	0	1	1	1	0	1	1	0	1	0	0
21	0	0	0	1	1	1	0	1	1	0	1	0
22	0	0	0	0	1	1	1	0	1	1	0	1
23	0	1	0	0	0	1	1	1	0	1	1	0
24	0	0	1	0	0	0	1	1	1	0	1	1

Imagen 2: Captura de matlab de la matriz de control de la paridad de 24x12

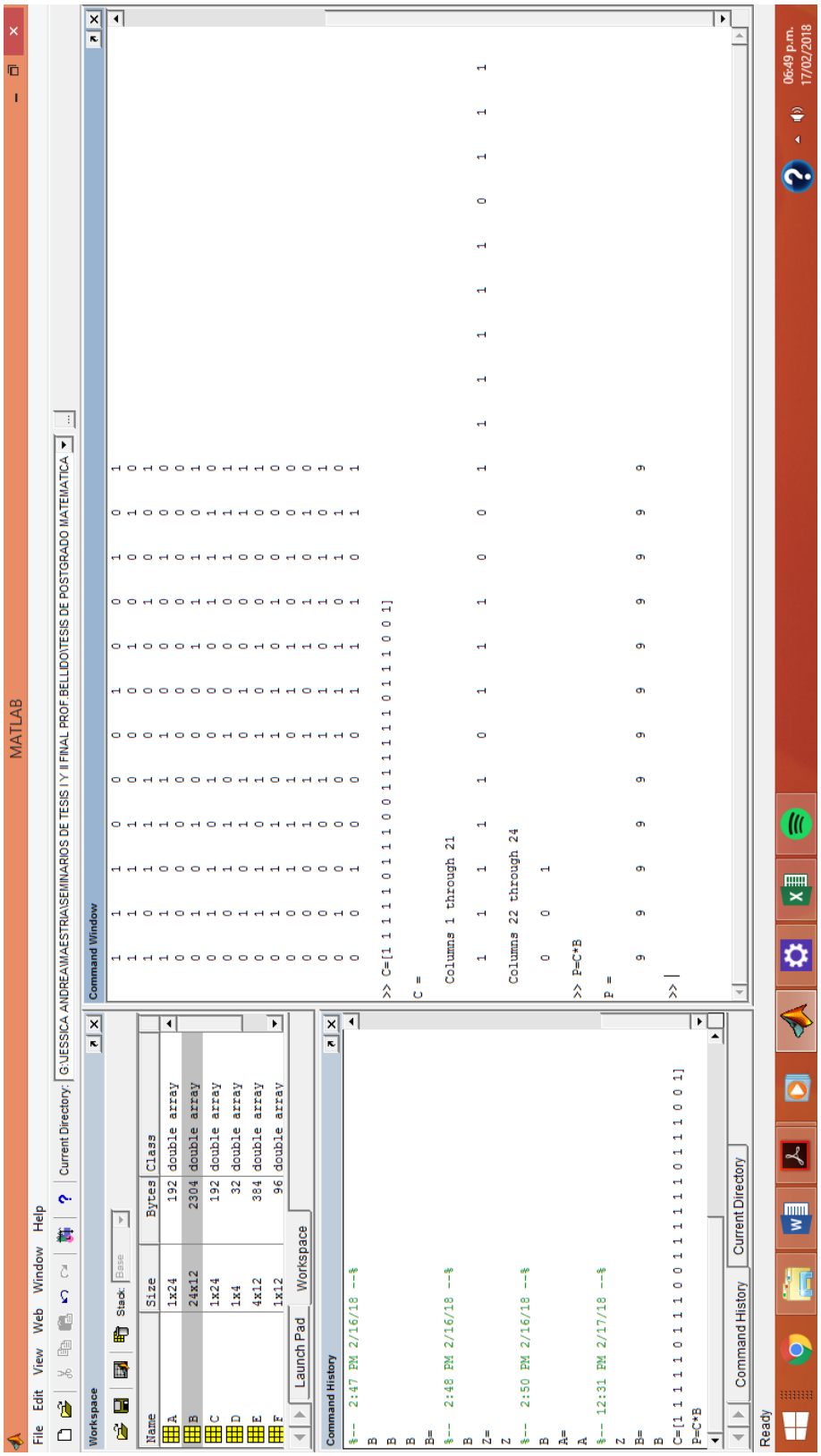


Imagen 3: Captura de la puesta en marcha en matlab del producto interno entre el código y la matriz.

Finalmente convirtiendo cada palabra a su representación en binario como cuerpo tenemos:

$$(1\ 1\ 1\ 1\ 1\ 1\ \ 1\ 1\ 1\ 1\ 1\ 1)$$

$$(1\ 1\ 1\ 1\ 1\ 1\ \ 1\ 1\ 1\ 1\ 1\ 1)_{1\times n}$$

$$(1\ 1\ 1\ 1\ 1\ 1\ \ 1\ 1\ 1\ 1\ 1\ 1)_{1\times 12}$$

Como en la segunda cota $M \leq 2(12)$, $M \leq 24$ si d es par y $12 = 2d$, entonces d=6; esto lo articula por consiguiente al Teorema de Leveinstein $M(2d, d) = 4d$ donde la distancia de Hamming es seis también.

Y siendo así en cada código se aplica la fórmula para corrección de errores de la distancia de Hamming $\frac{(d-1)}{2}$; $\frac{(6-1)}{2} = \frac{5}{2} = 2{,}5$.

Lo que se entiende como que en el bloque se pueden corregir por lo menos dos errores en cada código lineal como se puede observar si comparamos con el código AB

$$(1\ 1\ 1\ 1\ 1\ \mathbf{0}\ \ 1\ 1\ 1\ \mathbf{0}\ \mathbf{0}\ 1)$$

$$(1\ 1\ 1\ 1\ 1\ \mathbf{1}\ \ 1\ 1\ 1\ \mathbf{1}\ \mathbf{1}\ 1)_{1\times 12}$$

$$V\ V\ V\ V\ V\ \boldsymbol{F}\ \ V\ V\ V\ \boldsymbol{F}\ \boldsymbol{F}\ V$$

Así en bloque se puede observar que hay un error en el primer y segundo componente del código hay dos errores tal como lo admite la distancia de Hamming.

En bloque nótese que se trata de códigos opuestos, A es opuesto a B en las tres últimas palabras, por otro lado se puede observar el 0 por el 1, el 1 por el 0 donde indica incorrecto, es decir que hay error en el primer y segundo componente del código porque la comparación se hace de dos a dos analizando en bloque. Entonces cuando enviamos AB y recibimos CC. No olvidemos además que la matriz de control de paridad interviene en nuestra codificación de Hadamard para analizar por bloques, así tenemos:

Codificación usando Teoría de A. Hedayat y W.D.Walis:

To get started, select "MATLAB Help" from the Help menu.

>> B =

1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	1	1	1	0	0	0	1	0
1	0	0	1	0	1	1	1	0	0	0	1

```
    1    1    0    0    1    0    1    1    1    0    0    0
    1    0    1    0    0    1    0    1    1    1    0    0
    1    0    0    1    0    0    1    0    1    1    1    0
    1    0    0    0    1    0    0    1    0    1    1    1
    1    1    0    0    0    1    0    0    1    0    1    1
    1    1    1    0    0    0    1    0    0    1    0    1
    1    1    1    1    0    0    0    1    0    0    1    0
    1    0    1    1    1    0    0    0    1    0    0    1
    1    1    0    1    1    1    0    0    0    1    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    1    0    1    0    0    0    1    1    1    0    1
    0    1    1    0    1    0    0    0    1    1    1    0
    0    0    1    1    0    1    0    0    0    1    1    1
    0    1    0    1    1    0    1    0    0    0    1    1
    0    1    1    0    1    1    0    1    0    0    0    1
    0    1    1    1    0    1    1    0    1    0    0    0
    0    0    1    1    1    0    1    1    0    1    0    0
    0    0    0    1    1    1    0    1    1    0    1    0
    0    0    0    0    1    1    1    0    1    1    0    1
    0    1    0    0    0    1    1    1    0    1    1    0
    0    0    1    0    0    0    1    1    1    0    1    1

>> C=[1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 0 1]
C =
  Columns 1 through 14
    1    1    1    1    1    0    1    1    1    0    0    1    1    1
  Columns 15 through 24
    1    1    1    0    1    1    1    0    0    1

>> P=C*B
P =
    9    9    9    9    9    9    9    9    9    9    9    9

>> D=[1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0]
```

```
D =
  Columns 1 through 14
    1    1    1    1    1    0    1    1    1    0    0    1    1    1
  Columns 15 through 24
    1    1    1    0    0    0    0    0    0    0

>> Q=D*B
Q =
    9    8    6    6    7    7    6    6    6    8    7    8

>> E=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
E =
  Columns 1 through 14
    1    1    1    1    1    1    1    1    1    1    1    1    1    1
  Columns 15 through 24
    1    1    1    1    1    1    1    1    1    1

>> R=E*B
R = 12    12    12    12    12    12    12    12    12    12    12    12
```

6. Simulación para corregir errores en el kardex de una microempresa.

6.1. Simulación usando Teoría de Domínguez.

En Matlab: Se realizó la multiplicación de una cadena de cuatro palabras llamada A que en nuestro kardex de microempresa son objetos de tecnología con la matriz generadora "nuestro código de Hadamard" la matriz de 4×12 llamada Z, obteniendo como producto una cadena de longitud doce:

$$W = (0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1)$$

Como se puede apreciar:

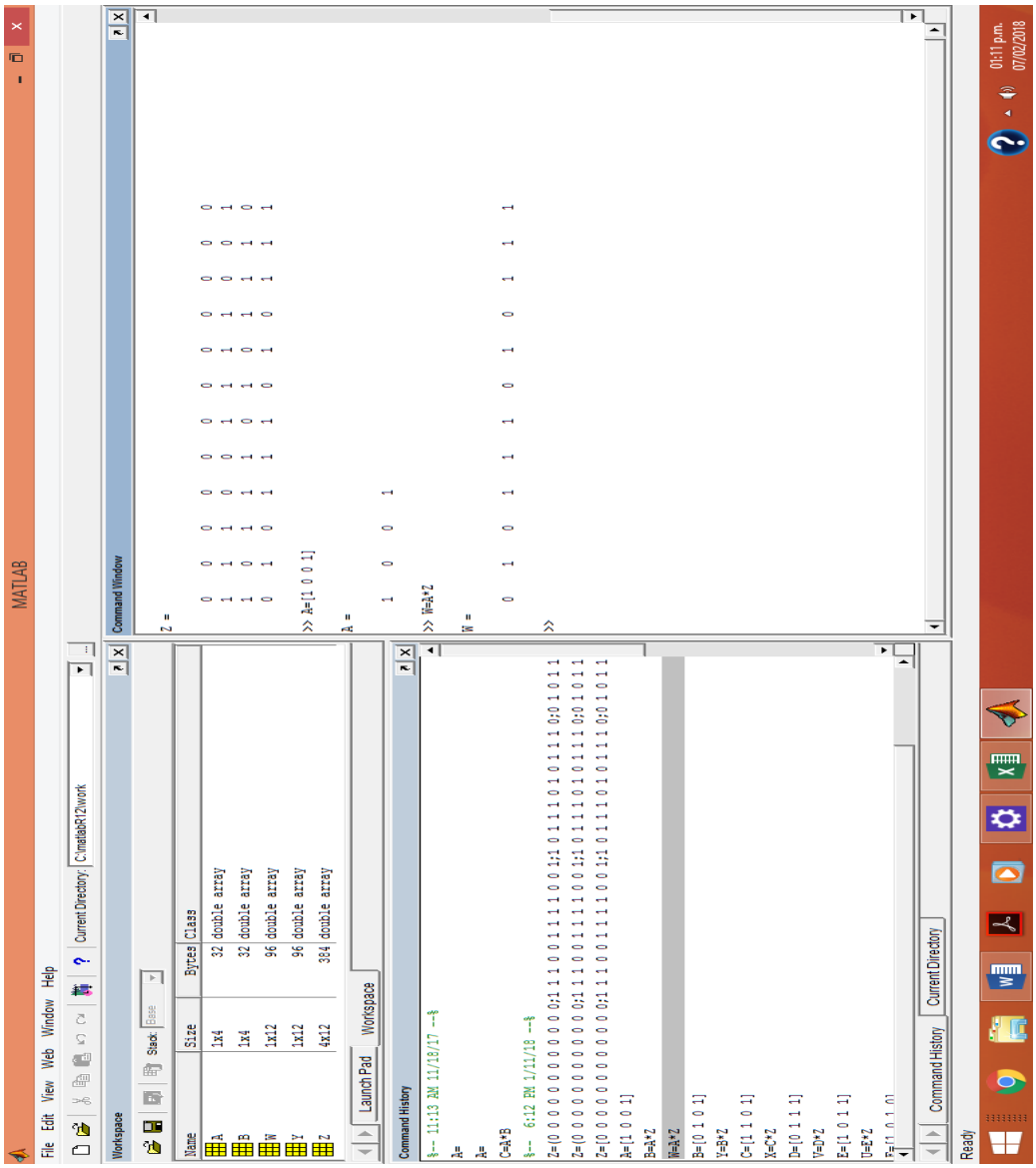


Imagen 4: Captura de Matlab donde se observa la primera palabra código de la codificación de hadamard para organizar en Excel según las iteraciones.

Luego en este resultado ya tenemos la secuencia W cadena de longitud doce donde entiéndase que a lo más ocho errores se pueden reconocer en ella, esto es la comprobación de la cota de Plotkin pues hay seis errores en total en la cadena lo cual es admitido y no pasa el límite, esto se demuestra anteriormente en la modelación.

Luego se deduce que son palabras códigos M diferentes de 1 0 0 1, por ello se subdividirán en cadenas de cuatro para comparar con la cadena de cuatro palabras original, teniendo:

$B = 0101$, asignado a artículos de deporte

$errores = 2$; posibilidad intermedia de error

$C = 1101$, asignado a artículos de salud

$errores = 1$; posibilidad mínima de error, casi correcto

$D = 0111$, asignado a artículos de belleza

$errores = 3$; posibilidad máxima de error

Quedando así en un reporte en Excel:

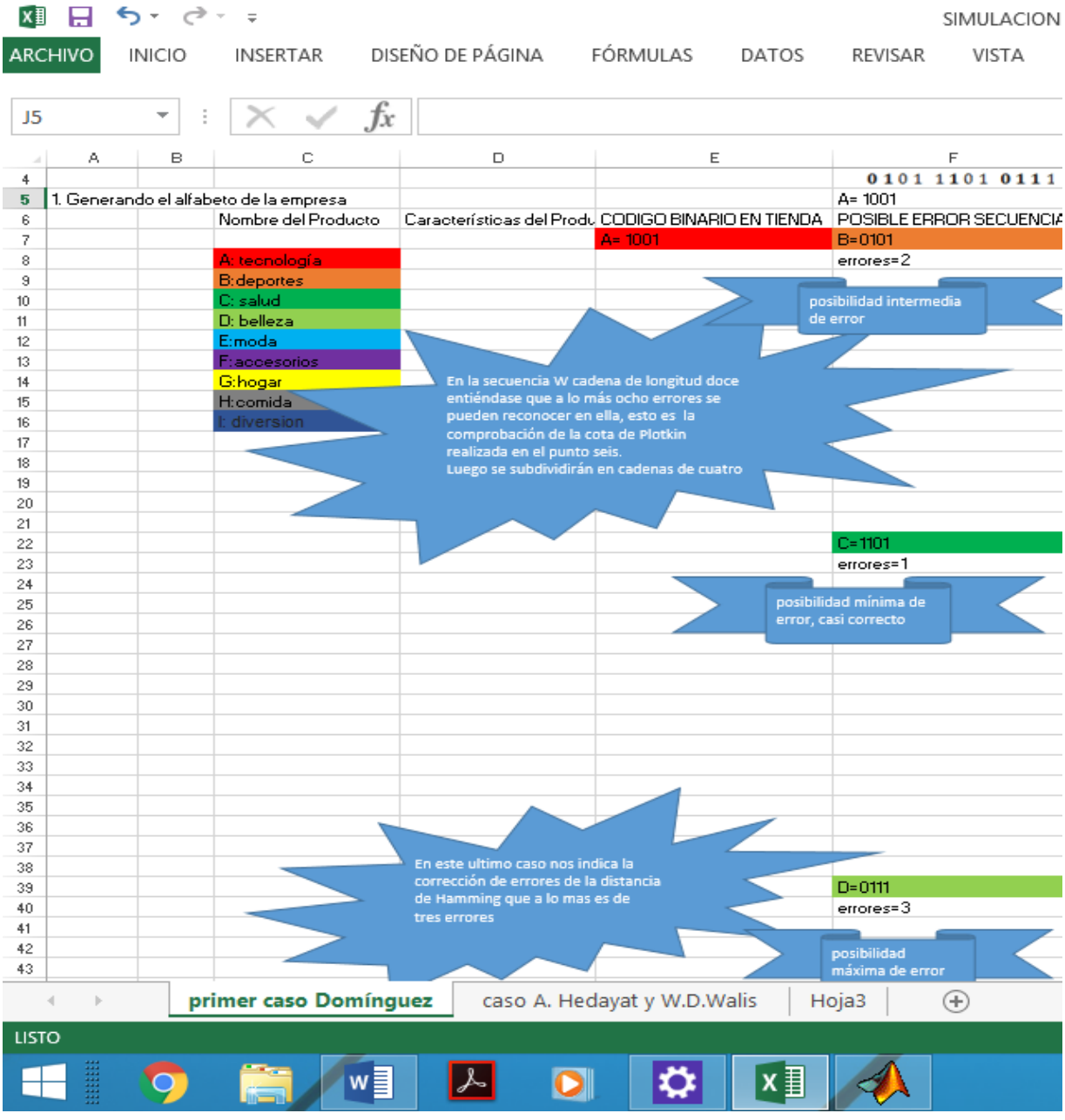


Imagen 5: Captura de Excel donde se observa el análisis de los errores en la primera iteración.

Por lo que un error en la cadena da lugar a otra palabra apoyados en la definición de la distancia de Leveinshtein que se traduce en confundir un artículo por otro si éste perteneciese a otro lenguaje idiomático por ejemplo o si el empresario tendría que tomar cuenta los productos de cada categoría a un abastecedor.

Ahora, entramos a un proceso de iteración tal como lo asevera el Prof. Víctor Álvarez Solano en "Internacional Workshop on Hadamard and Cocyclic Matrices and Applications-IWHCMA": (De Solano, Víctor Álvarez, 2018)

Imagen 6: Captura de Excel donde se observa el análisis de todos los errores en la trece iteraciones hasta finalizar.

Donde se observa que de la anterior cadena W y generándose las cadenas de palabra B, C, y D se la multiplica nuevamente por nuestra matriz casi Hadamard a cada una respectivamente iterándose así:

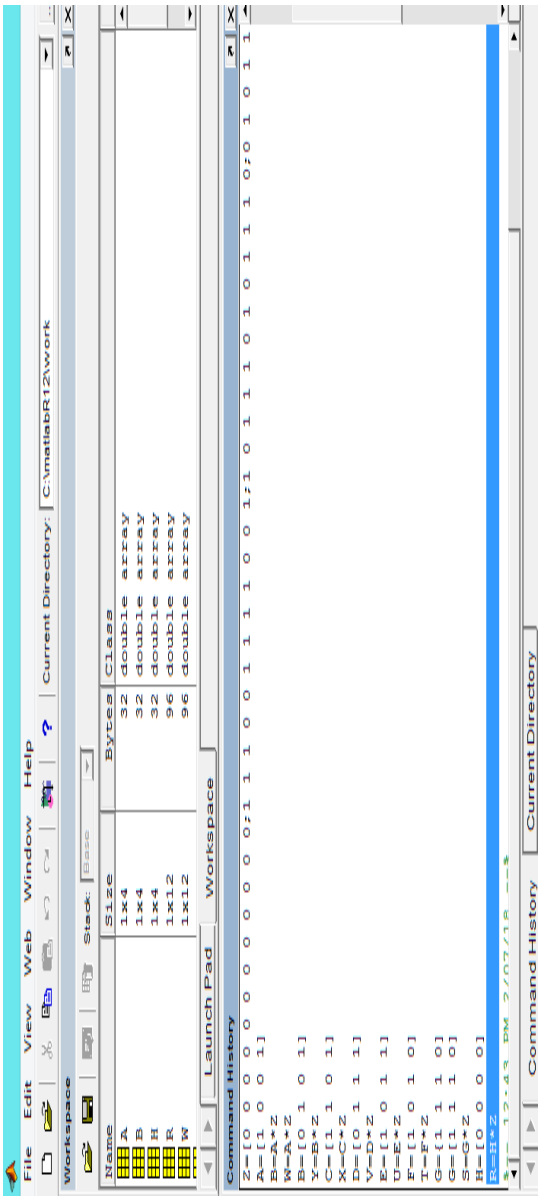


Imagen 7: Captura de Matlab donde se observa cada código que ingresará dando lugar a la cadena de códigos alargados que se fragmentaron como en la vista anterior.

Y así sucesivamente repitiéndose el proceso anteriormente señalado tenemos la **Simulación para la Teoría de Domínguez:**

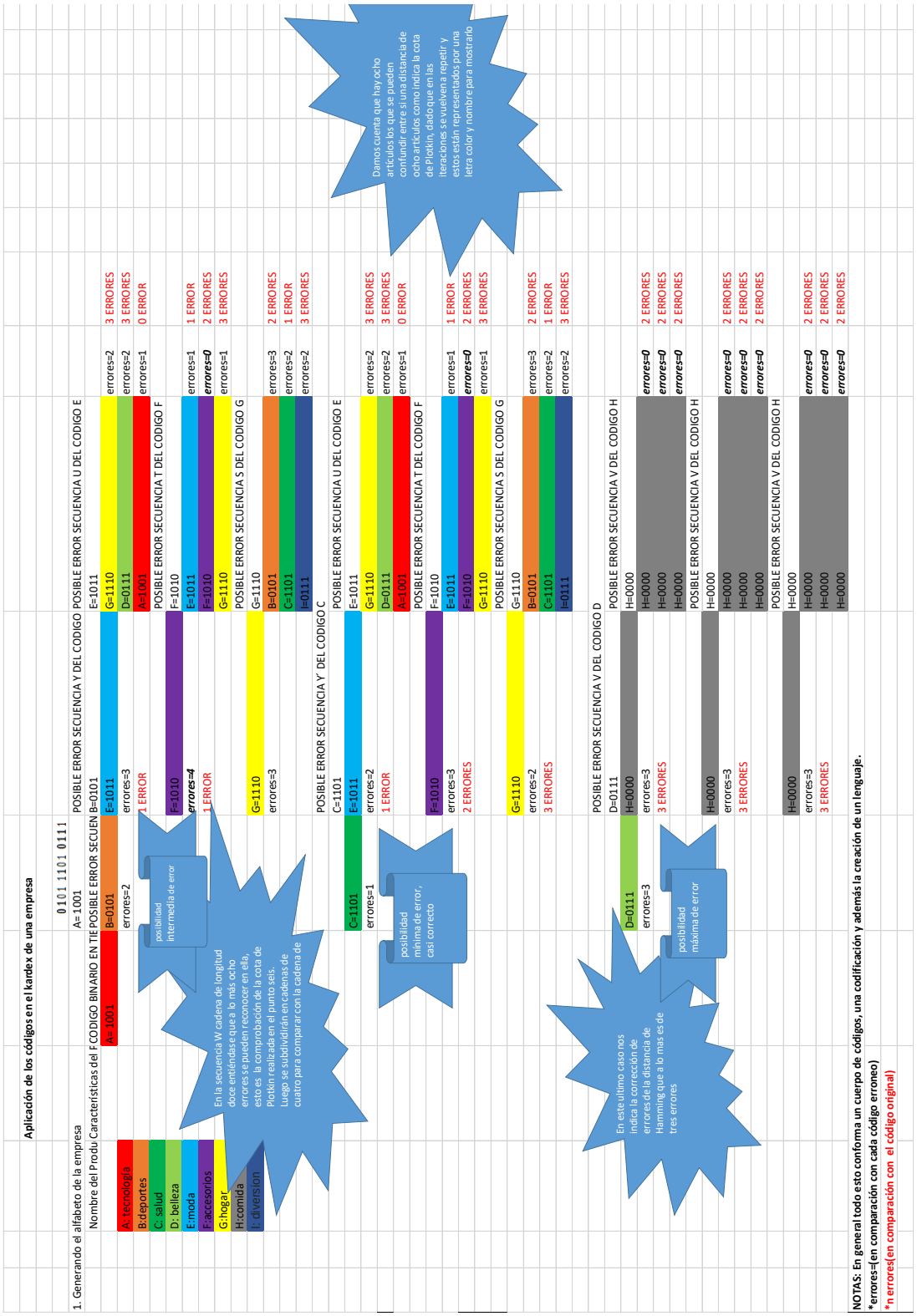


Imagen 8: Captura de Excel donde se observa que los códigos alargados se han fragmentado en palabras y de allí la comparación con la palabra código original inicial para resaltar errores dando lugar a un algoritmo y base de datos tal como se muestra listo para aplicar.

Interpretación de los resultados de aplicación de la Teoría de Domínguez:

1° Corrige a lo más tres errores en cada terna de cadena, y además como máximo tres códigos en alguna cadena, como queda demostrado en el último código.

2° Se han corregido seis errores en total del código lo cual está en el rango de la distancia de Hamming $d \leq 8$.

3° Cuando realizamos las iteraciones salen números diferentes de 1 y 0, como 2, entonces lo reemplazamos por cero debido al residuo, esto por la teoría de cuerpo, así colocamos cero en número par y uno en número impar.

4° Damos cuenta que hay ocho tipos de artículos los que se pueden confundir entre sí, una distancia de ocho artículos como indica la cota de Plotkin, dado que en las iteraciones se vuelven a repetir están representados por una letra color y nombre para mostrarlo didácticamente en esta simulación.

5° Entramos en la disyuntiva de la comparación para deducir el error ¿con qué cadena de palabra tenemos que comparar?, ¿será con cada cadena E, F, G, H que multiplicó a la matriz en cada iteración? o ¿será con la cadena primigenia la original A que multiplicó a la matriz en el primer proceso?

Analizamos: no podemos comparar entre errores no es la anterior a la cadena de cada iteración puesto que tendríamos errores que son igual a cuatro y después también errores igual a cero, lo cual no admite la teoría de códigos, no se puede corregir todo $e = 4$, si esta todo correcto $e = 0$ no hay error.

Entonces deducimos que se compara con la cadena primigenia la original A que multiplicó a la matriz en el primer proceso, dado que permiten errores de uno, dos y hasta tres totalmente admitido en la teoría de códigos y además refrendado con la corrección de códigos que nos indica la distancia de Hamming. Además porque guarda consistencia y es sustentable con el proceso primigenio para después iterar de la misma forma.

6° En general todo esto conforma un cuerpo de códigos y una codificación es decir la creación de un lenguaje.

7°Este modelo matemático se puede usar para el manejo de la logística kardex de una empresa donde la codificación es importante por categorías para manejar precios, costos, stock con un determinado abastecedor y la propia contabilidad. Luego, para validar los códigos de diferentes productos se registran en una APP, así con la ayuda de un código de barras y la categoría dada por nuestra propuesta de codificación de códigos binarios y la matriz casi Hadamard quedarán almacenados.

6.1. Simulación usando Teoría de A. Hedayat y W.D.Walis.

Recordemos que para este caso la distancia de Hamming es seis que coincide con la longitud de la cadena de cada código lineal, entonces compararemos dos códigos lineales a la vez por estar en bloque. Tenemos seis códigos lineales que juegan cada par.

La cadena lineal final de $n = 24$ es una cadena de copias que resultan de trabajarse dos a dos, siendo A y B cadenas originales para encontrar dos errores, extendiendo el código haciendo tantas copias como longitud de la cadena original lo permitan hasta llegar a 24.

Veremos las simulaciones que derivan:

Probemos para

$a_3C_1 \oplus a_4C_2 \oplus a_5C_1 \oplus a_6C_2$

Reemplazando

$$1(111\ 110) \oplus 1(111\ 001) \oplus 1(111\ 110) \oplus 0(111\ 001)$$

$$= 111\ 110 \oplus 111\ 001 \oplus 111\ 110 \oplus 000\ 000$$

$$AB' = (1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$

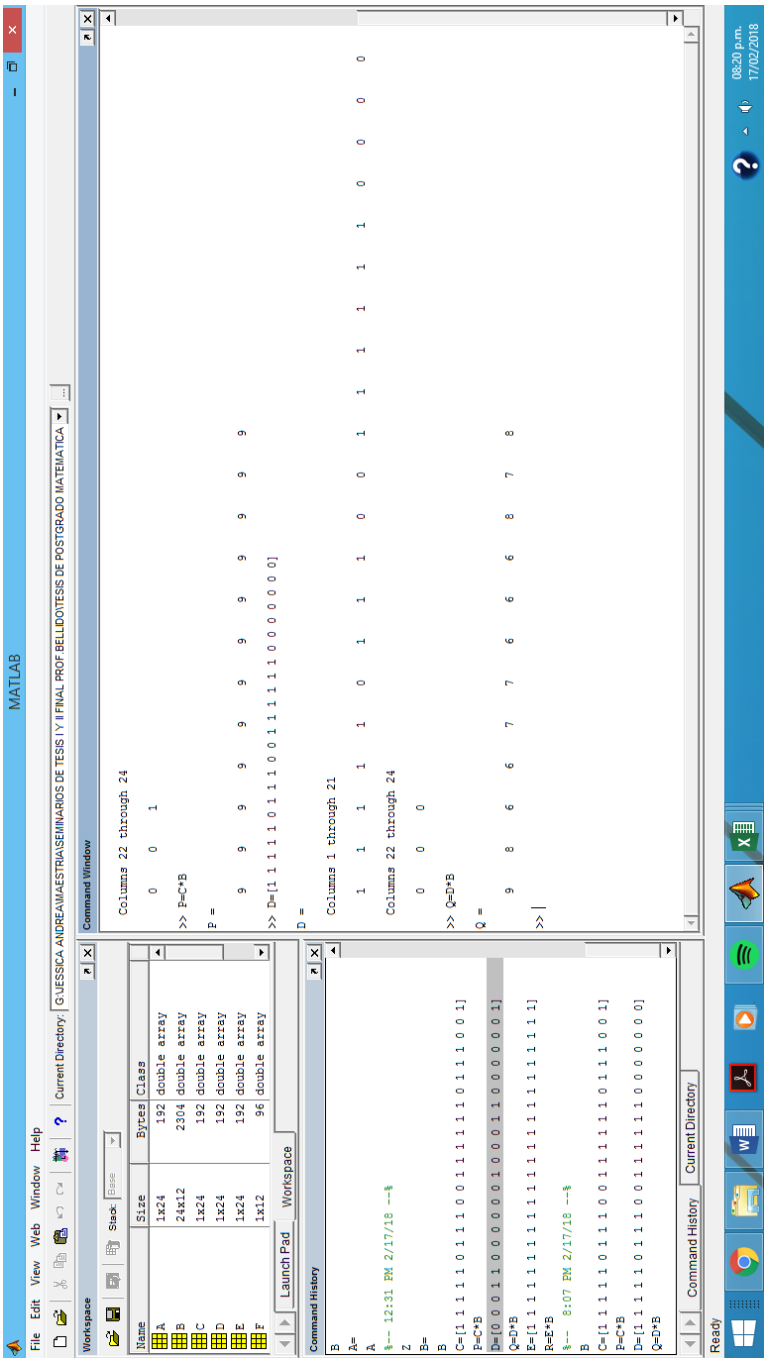


Imagen 9: Captura de Matlab con la codificación de hadamard usando la teoría de de A. Hedayat y W.D.Walis.

Convirtiendo a binario tenemos:

1 0 0 0 1 1 0 0 0 0 1 0

Obtenemos la **Simulación para la Teoría de A. Hedayat y W.D.Walis:**

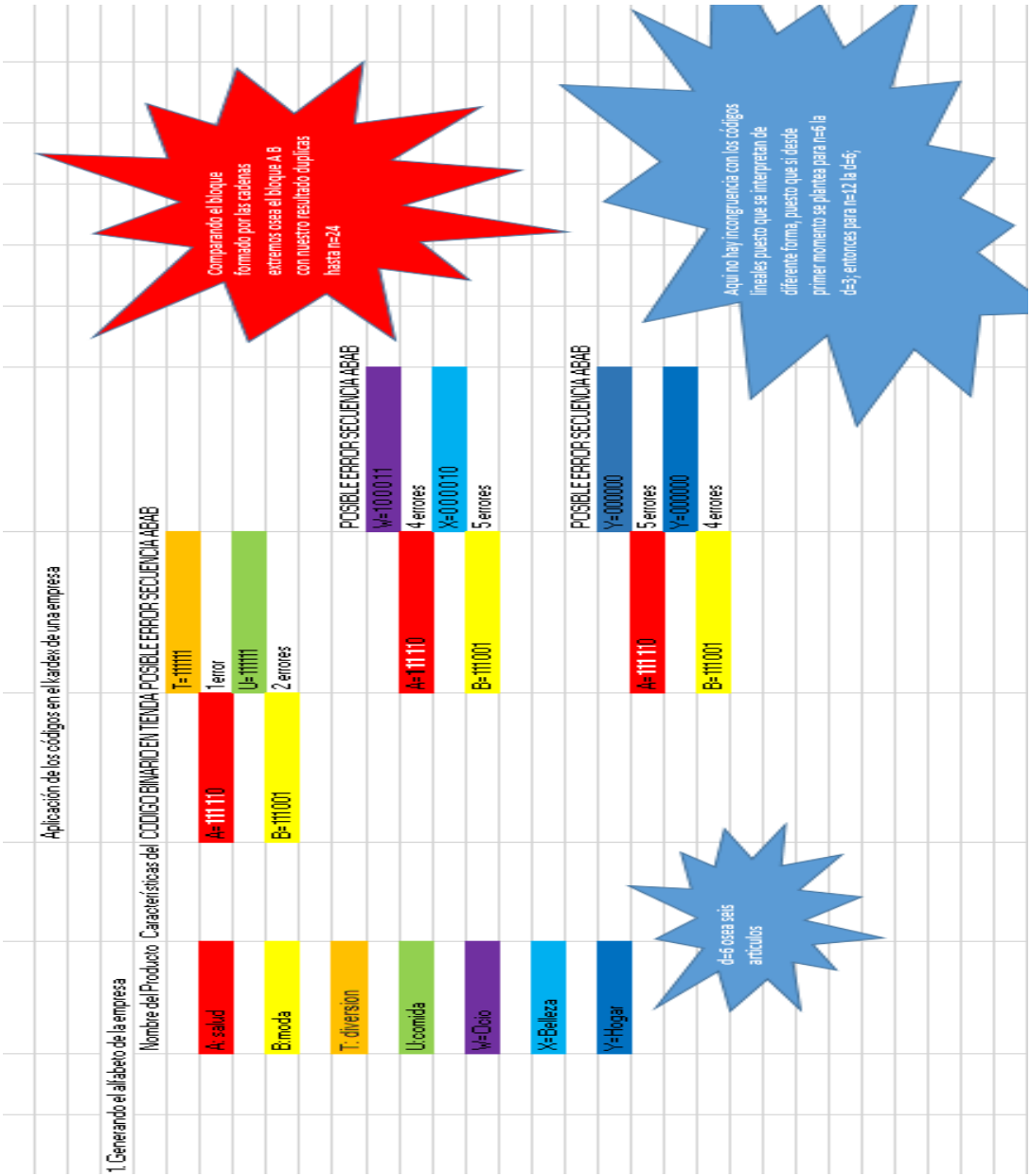


Imagen 10: Captura de Excel con el modelo usando la teoría de de A. Hedayat y W.D.Walis. Códigos muy alargados fragmentados que señalan error en cierto lugar de la palabra código.

Interpretación de los resultados de aplicación de la Teoría de A. Hedayat y W.D.Walis:

1° Entonces en A aparecen 4 errores y en B hay 5 errores. Lo cual es admitido por la distancia de Haming que es igual a 6 y como se comparan dos a dos puede darse el caso que uno este completamente bien y el otro totalmente mal como el peor de los escenarios, así

un código no descarta al otro y por eso se interpreta de manera independiente, entonces es válido que la distancia de Hamming es igual a seis, así se interpreta como una doble comparación simultánea para el caso de códigos bloques.

2° Sin embargo se generan siete artículos distintos, lo cual incrementa la familia de códigos pero tiende a muchas posibilidades que se desenmarcan de lo finito de un lenguaje en la teoría de errores de códigos.

3° Luego entendemos que el uso de esta comparación de dos cadenas a la vez conlleva a mayor posibilidad de error entonces esta simulación y modelamiento quedan descartados, asumiendo el método lineal de Domínguez como el más práctico, preciso y aplicable.

Conclusiones

1. La existencia del numero de Hadamard queda demostrado en el teorema de Hadamad, que está presente para ambas simulaciones en la matriz de control de la paridad matriz de Hadamard, de esta manera el orden de la matriz es el número de Hadamard. Luego ambas son modificadas para obtener matrices casi Hadamard permaneciendo el número de Hadamard. Finalmente en el resultado de la modelación se obtienen códigos lineales cuya longitud de palabra n certifica el número de Hadamard, para ambos casos 12.

2. Se debe trabajar con códigos lineales y códigos bloque ambos binarios ya que es fácil corregir algo falso corrigiéndo con lo opuesto, así si hay error en uno se corrige con cero y viceversa. Éstos poseen parámetros (n, M, d) corrigiendo así la mayor cantidad de palabras n, sin extender a más palabras M y ubicándose mejor el error através de la distancia de hamming d. Estos se articulan bien con las matrices casi-Hadamard ya que su producto interno dos a dos es casi 0 que con la teoría de cuerpos admite residuos que conservan el lenguaje binario. Por lo tanto después del modelamiento con la Teoría de Domínguez, los códigos lineales constituyen la codificación más estable.

3. Se aplicaron dos métodos distintos de Construcción de Código para obtener la matriz de control de la paridad los cuales son el de Gauss y el del teorema de códigos óptimos para la

matriz de Hadamard articulandose con el producto Kronecker y el Método de Sylvester. Siendo la distancia de Hamming d un elemento de control porque define el número de elementos con error. El parámetro d resalta en el método de la investigación como variable interviniente, el cual hizo posible los ajustes entre el Teorema de Leveinstein, las Matrices de Hadamard y las Cotas de Plotkin para lograr articularlos.

4. Interpretando los resultados tenemos que La matriz generadora y conversora de códigos representa los vectores de control conformando la matriz de control de paridad que por ser matriz ortogonal, equivalente e invertible y debido a que las filas y columnas distribuidas dos a dos generan pares hacen posible alargar los códigos y detectar el error a través de la distancia de Hamming d por ser balanceadas y luego corregidos.

5. De las simulaciones, la mas optima es la que considera la teoría de Domínguez constituyendo un lenguaje completo formado por ocho códigos prácticos, precisos y aplicables que cumplen con la distancia de Leveinshtein que consiste en confundir un artículo por otro afinando así la incompleta modelación de la teoría de Hedayat. Conformando un cuerpo de códigos y la creación de un lenguaje.

6. Concluimos que para obtener los codigos binarios optimos se deben usar codigos lineales, aplicar la distancia de Leveinstein y ademas realizar de manera pertinente la codificacion con la matriz de Hadamard y asi corregirlos algorítmicamente y automáticamente a manera de un lenguaje codificado. Todo ello se ha implementado en el Modelo Matemático Para La Corrección De Códigos Binarios Mediante La Matriz De Hadamard Aplicado a Las Cotas Pares De Plotkin.

Recomendaciones

1) Tantos métodos de construcción para la matriz de Hadamard y casi-Hadamard existan, tantos algoritmos o codificación existirán. Por ello se sugiere complementar con futuros trabajos de investigación.

2) Para las cotas de Plotkin impares la construcción de la matriz casi hadamard se realiza por otros métodos más complejos que articulan el Teorema de Leveinshtein, éste será otro problema que se deja para posteriores investigaciones.

Bibliografía

1. Álvarez, J. A. Armario, M. D. Frau, E. Martin, y A. Osuna. (2007). Error correcting codes from quasi-Hadamard matrices. *Lecture Notes in Comput. Sci. 4547, 294 Dpto. Matemática Aplicada I, Universidad de Sevilla, Avda. Reina Mercedes s/n 41012 Sevilla, Spain.*
2. Graña Otero Beatriz. (2008). Teoría de códigos.
3. Ben Lantza Michael Zowadab (2012) An Overview of Complex Hadamard Cubes. *Rose-Hulman Undergraduate Mathematics Journal, 13*(2).
4. Cameron Peter J. (2006). Hadamard matrices. Encyclopaedia of DesignTheory: Topics. *The Encyclopaedia of Design Theory. Europa.* Recuperado de http://www.maths.qmul.ac.uk/~leonard/designtheory.org/library/encyc/topics/had.pdf
5. Borges Hernández, Cruz Enrique. (2005). Codificación y decodificación de los códigos cíclicos. Comparativa con otros códigos. *Universidad de Cantabria.*
6. Borges Hernández, Cruz Enrique. (2005). Códigos cíclicos.
7. Dr. Mata. Miguel (2016). Matemáticas discretas. *FIME-UANL*
8. SARUKHANYAN Hakob, ANOYAN Armen. Institute for Informatics and Automation Problems of NAS of Armenia, EGIAZARIAN Karen, ASTOLA Jaakko. Tampere International Center on Signal Processing, Tampere University of Technology, Tampere, Finland, AGAIAN Sos, University of Texas at San Antonio, The University of the New York City. Codes Generated From Hadamard Matrices.

9. Hedayat y W.D. Wallis. Hadamard matrices and their applications. *The Annals os Statistics.* University of Illinois at Chicago Circle and University of Newcastle.
10. Pinero Simón, Jacobo Juan. (2011-2012). *Apuntes de Códigos Correctores de Errores.*
11. Peralta López Justo. Ampliación Matemática Discreta. *Universidad De Almeria. Departamento De Agebra Y Analisis Matematico.*
12. Lay, David C. (2007), *Álgebra Lineal Y Sus Aplicaciones*. México, S.A. de C.V. Pearson Educación.
13. Universidad Tecnológica Nacional Facultad Regional Buenos Aires. Secretaría de Gestión Académica. *Lenguajes, Grámaticas Y Autómatas.* Unidad 7.
14. Tarazona Luis (2001). Redes de Computadores. DIP UNEXPO Barquisimeto ELT-51123.
15. Iranzo Aznar María Jesús, Monasor Francisco Pérez. (2004-2005). Lecciones de Elementos de Algebra. Aplicaciones. *Curso Departamento de Algebra. Facultad de Matemáticas.* Universidad de Valencia.
16. Mezoa Mariangela. Codificación de Canal: Código Hamming y Código Convolucional. *Creative Commons Attribution License 3.0. OpenStax-CNX module: m36906.*
17. Massoud Malek. Coding Theory. Hadamard Codes. California State University, East Bay.
18. Cárdenas-Montes Miguel. Medidas de Distancia.
19. Morad Mohammed Ali, Mahmood Adawiyah Ali, Hameed Khalid Hadi. (2009). Construction of Sylvester-Hadamard Matrices by Using Binary Code. *J. of al-anbar university for pure science.* University of Diyala-College of Basic Education.University of Diyala- College of science, *3*(1).
20. Fernández Gallardo Pablo. Profesor Asociado de Matemáticas. Códigos Correctores De Errores, Quinielas Y Empaquetamientos. Universidad Politécnica de Madrid
21. Piza, Eduardo (2011). Búsqueda de matrices de Hadamard a través de secuencias de Turyn. *Revista de Matemática: Teoría y Aplicaciones*, Universidad de Costa Rica. San José, Costa Rica, *18*(2), 193-214.
22. Prof. M.Sc. Ramírez Benavides Kryscia Daviana. Algoritmos de Similaridad y Distancia. UCR –ECCI. CI-2414. Recuperación de Información.
23. Prof. De Castro Rodrigo. (2003). Teoría de la Computación -II. Capítulo 1 Alfabetos, cadenas y lenguajes
24. A. Podesta. Ricardo (2006). Introducción A La Teoría De Códigos Autocorrectores. Mathematics Subject Classification. 94-01, 94-06, 94B05, 94B15. CONICET y SecytUNC.
25. Stanley I. Grossman S (2008). *Álgebra lineal.* Mcgraw-Hill/Interamericana Editores, S.A. De C.V. Sexta edición. México.

26. T. Domínguez (2003). Matrices de Hadamard. *Departamento de Análisis Matemático, Facultad de Matemáticas, Universidad de Sevilla.* Bol. Soc. Esp. Mat. Apl. no23 (2003), 91–100.
27. De Solano, Víctor Álvarez. Recuperado de https://investigacion.us.es/sisius/sis_showpub.php?idpers=4505
28. De Math.info. Recuperado de http://www.learn-math.info/spanish/historyDetail.htm?id=Hadamard
29. De Cuerpo (matemáticas). Recuperado de https://es.wikipedia.org/wiki/Cuerpo_(matem%C3%A1ticas)

Anexos

Anexo 1.

Matriz de Hadamard y Número de Hadamard en Matlab.

A continuación comprobamos las características de las Matrices de Hadamard en Matlab:

```
>> G=Hadamard(2)
G =
    1     1
    1    -1
>> H=Hadamard(4)
H =
    1     1     1     1
    1    -1     1    -1
    1     1    -1    -1
    1    -1    -1     1


>> J= H'
J =
    1     1     1     1
    1    -1     1    -1
    1     1    -1    -1
    1    -1    -1     1
>> P= H*J
P =
    4     0     0     0
    0     4     0     0
    0     0     4     0
    0     0     0     4
```

```
>> H=Hadamard(12)
H =
     1     1     1     1     1     1     1     1     1     1     1     1
     1    -1     1    -1     1     1     1    -1    -1    -1     1    -1
     1    -1    -1     1    -1     1     1     1    -1    -1    -1     1
     1     1    -1    -1     1    -1     1     1     1    -1    -1    -1
     1    -1     1    -1    -1     1    -1     1     1     1    -1    -1
     1    -1    -1     1    -1    -1     1    -1     1     1     1    -1
     1    -1    -1    -1     1    -1    -1     1    -1     1     1     1
     1     1    -1    -1    -1     1    -1    -1     1    -1     1     1
     1     1     1    -1    -1    -1     1    -1    -1     1    -1     1
     1     1     1     1    -1    -1    -1     1    -1    -1     1    -1
     1    -1     1     1     1    -1    -1    -1     1    -1    -1     1
     1     1    -1     1     1     1    -1    -1    -1     1    -1    -1

>> B=H'
B =
     1     1     1     1     1     1     1     1     1     1     1     1
     1    -1    -1     1    -1    -1    -1     1     1     1    -1     1
     1     1    -1    -1     1    -1    -1    -1     1     1     1    -1
     1    -1     1    -1    -1     1    -1    -1    -1     1     1     1
     1     1    -1     1    -1    -1     1    -1    -1    -1     1     1
     1     1     1    -1     1    -1    -1     1    -1    -1    -1     1
     1     1     1     1    -1     1    -1    -1     1    -1    -1    -1
     1    -1     1     1     1    -1     1    -1    -1     1    -1    -1
     1    -1    -1     1     1     1    -1     1    -1    -1     1    -1
     1    -1    -1    -1     1     1     1    -1     1    -1    -1     1
     1     1    -1    -1    -1     1     1     1    -1     1    -1    -1
     1    -1     1    -1    -1    -1     1     1     1    -1     1    -1
```

```
>> A=H*B

A =

    12     0     0     0     0     0     0     0     0     0     0     0
     0    12     0     0     0     0     0     0     0     0     0     0
     0     0    12     0     0     0     0     0     0     0     0     0
     0     0     0    12     0     0     0     0     0     0     0     0
     0     0     0     0    12     0     0     0     0     0     0     0
     0     0     0     0     0    12     0     0     0     0     0     0
     0     0     0     0     0     0    12     0     0     0     0     0
     0     0     0     0     0     0     0    12     0     0     0     0
     0     0     0     0     0     0     0     0    12     0     0     0
     0     0     0     0     0     0     0     0     0    12     0     0
     0     0     0     0     0     0     0     0     0     0    12     0
     0     0     0     0     0     0     0     0     0     0     0    12

>> F=Hadamard(3)

??? Error using ==> hadamard

n must be an integer and n, n/12 or n/20 must be a power of 2.

>> F=Hadamard(24)

F =

  Columns 1 through 12

     1     1     1     1     1     1     1     1     1     1     1     1
     1    -1     1    -1     1     1     1    -1    -1    -1     1    -1
     1    -1    -1     1    -1     1     1     1    -1    -1    -1     1
     1     1    -1    -1     1    -1     1     1     1    -1    -1    -1
     1    -1     1    -1    -1     1    -1     1     1     1    -1    -1
     1    -1    -1     1    -1    -1     1    -1     1     1     1    -1
     1    -1    -1    -1     1    -1    -1     1    -1     1     1     1
     1     1    -1    -1    -1     1    -1    -1     1    -1     1     1
     1     1     1    -1    -1    -1     1    -1    -1     1    -1     1
     1     1     1     1    -1    -1    -1     1    -1    -1     1    -1
     1    -1     1     1     1    -1    -1    -1     1    -1    -1     1
     1     1    -1     1     1     1    -1    -1    -1     1    -1    -1
     1     1     1     1     1     1     1     1     1     1     1     1
     1    -1     1    -1     1     1     1    -1    -1    -1     1    -1
```

1 1 -1 -1 1 -1 1 1 1 -1 -1 -1
1 -1 -1 1 -1 1 1 1 -1 -1 -1 1
1 -1 1 -1 -1 1 -1 1 1 1 -1 -1
1 -1 -1 1 -1 -1 1 -1 1 1 1 -1
1 -1 -1 -1 1 -1 -1 1 -1 1 1 1
1 1 -1 -1 -1 1 -1 -1 1 -1 1 1
1 1 1 -1 -1 -1 1 -1 -1 1 -1 1
1 1 1 1 -1 -1 -1 1 -1 -1 1 -1
1 -1 1 1 1 -1 -1 -1 1 -1 -1 1
1 1 -1 1 1 1 -1 -1 -1 1 -1 -1

Columns 13 through 24

1 1 1 1 1 1 1 1 1 1 1 1
1 -1 1 -1 1 1 1 -1 -1 -1 1 -1
1 -1 -1 1 -1 1 1 1 -1 -1 -1 1
1 1 -1 -1 1 -1 1 1 1 -1 -1 -1
1 -1 1 -1 -1 1 -1 1 1 1 -1 -1
1 -1 -1 1 -1 -1 1 -1 1 1 1 -1
1 -1 -1 -1 1 -1 -1 1 -1 1 1 1
1 1 -1 -1 -1 1 -1 -1 1 -1 1 1
1 1 1 -1 -1 -1 1 -1 -1 1 -1 1
1 1 1 1 -1 -1 -1 1 -1 -1 1 -1
1 -1 1 1 1 -1 -1 -1 1 -1 -1 1
1 1 -1 1 1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 1 -1 -1 -1 1 1 1 -1 1
-1 1 1 -1 1 -1 -1 -1 1 1 1 -1
-1 -1 1 1 -1 1 -1 -1 -1 1 1 1
-1 1 -1 1 1 -1 1 -1 -1 -1 1 1
-1 1 1 -1 1 1 -1 1 -1 -1 -1 1
-1 1 1 1 -1 1 1 -1 1 -1 -1 -1
-1 -1 1 1 1 -1 1 1 -1 1 -1 -1
-1 -1 -1 1 1 1 -1 1 1 -1 1 -1
-1 -1 -1 -1 1 1 1 -1 1 1 -1 1
-1 1 -1 -1 -1 1 1 1 -1 1 1 -1

```
   -1   -1    1   -1   -1   -1    1    1    1   -1    1    1
>> F=Hadamard(20)
F =
  Columns 1 through 12
    1    1    1    1    1    1    1    1    1    1    1    1
    1   -1   -1    1    1   -1   -1   -1   -1    1   -1    1
    1   -1    1    1   -1   -1   -1   -1    1   -1    1   -1
    1    1    1   -1   -1   -1   -1    1   -1    1   -1    1
    1    1   -1   -1   -1   -1    1   -1    1   -1    1    1
    1   -1   -1   -1   -1    1   -1    1   -1    1    1    1
    1   -1   -1   -1    1   -1    1   -1    1    1    1    1
    1   -1   -1    1   -1    1   -1    1    1    1    1   -1
    1   -1    1   -1    1   -1    1    1    1    1   -1   -1
    1    1   -1    1   -1    1    1    1    1   -1   -1    1
    1   -1    1   -1    1    1    1    1   -1   -1    1   -1
    1    1   -1    1    1    1    1   -1   -1    1   -1   -1
    1   -1    1    1    1    1   -1   -1    1   -1   -1    1
    1    1    1    1    1   -1   -1    1   -1   -1    1    1
    1    1    1    1   -1   -1    1   -1   -1    1    1   -1
    1    1    1   -1   -1    1   -1   -1    1    1   -1   -1
    1    1   -1   -1    1   -1   -1    1    1   -1   -1   -1
    1   -1   -1    1   -1   -1    1    1   -1   -1   -1   -1
    1   -1    1   -1   -1    1    1   -1   -1   -1   -1    1
    1    1   -1   -1    1    1   -1   -1   -1   -1    1   -1
  Columns 13 through 20
    1    1    1    1    1    1    1    1
   -1    1    1    1    1   -1   -1    1
    1    1    1    1   -1   -1    1   -1
    1    1    1   -1   -1    1   -1   -1
    1    1   -1   -1    1   -1   -1    1
    1   -1   -1    1   -1   -1    1    1
   -1   -1    1   -1   -1    1    1   -1
   -1    1   -1   -1    1    1   -1   -1
    1   -1   -1    1    1   -1   -1   -1
```

```
    -1    -1     1     1    -1    -1    -1    -1
    -1     1     1    -1    -1    -1    -1     1
     1     1    -1    -1    -1    -1     1    -1
     1    -1    -1    -1    -1     1    -1     1
    -1    -1    -1    -1     1    -1     1    -1
    -1    -1    -1     1    -1     1    -1     1
    -1    -1     1    -1     1    -1     1     1
    -1     1    -1     1    -1     1     1     1
     1    -1     1    -1     1     1     1     1
    -1     1    -1     1     1     1     1    -1
     1    -1     1     1     1     1    -1    -1
>> F=Hadamard(30)
??? Error using ==> hadamard
n must be an integer and n, n/12 or n/20 must be a power of 2.
```

Anexo 2.

Captura de la app "Stock Controller" para validar códigos.

A continuación capturas para realizar la validación de códigos en la modelación con la Teoría de Dominguez:

72% 7:40 PM
salud talco
0
INVENTARIO
0
TRANSACCIONES
Nombre salud talco
Código 0000
Código de barras 7752902000901
Tipo de unidad unid.
Descripción
Precio unitario de ref. (excl. impuestos) 1 €
Impuesto aplicado 1 %
Tipo hikkkm
Alerta cuando el inventario es inferior a 10 unid.
Creado hace 4 semanas por User
Última actualización hace 4 semanas por User
Imagen de producto

72% 7:40 PM
camisa
0
INVENTARIO
0
TRANSACCIONES
Nombre camisa
Código 1234
Código de barras 2012186522694
Tipo de unidad unid.
Descripción util
Precio unitario de ref. (excl. impuestos) 89.90 €
Impuesto aplicado 18 %
Tipo hikkkm
Alerta cuando el inventario es inferior a 10 unid.
Creado hace 1 mes por User
Última actualización hace 1 mes por User
Imagen de producto