Grado en Matemática Computacional

Estancia en Prácticas y Proyecto Final de Grado

Códigos Localmente Recuperables con Detección de Errores Locales

Autor:
Carlos J. Moreno

Supervisor:
Juan Miguel Tiscar
Tutor académico:
Fernando Hernando

Fecha de lectura: 28 de Septiembre de 2021 Curso académico 2020/2021

Resumen

Estamos inmersos en la era de la información y la comunicación. Esta comunicación debe proceder de forma segura. Asegurar esta comunicación parte de tres criterios fundamentales: integridad, disponibilidad y confidencialidad. El trabajo que se expone a continuación tiene como objetivo la integridad de los datos.

Al viajar la información por un canal, se producen unas interferencias llamadas ruidos. Esto produce ciertos errores y borrados dentro de la información que enviamos, lo que produce fallos en la comunicación. La solución que se ofrece parte de cómo generar los códigos que vamos a enviar por el canal.

La información se puede dotar de estructuras matemáticas, en concreto de estructuras algebraicas. Al hacer esa dotación, generamos isomorfismos con estructuras ya conocidas, por lo que se heredan las propiedades. Así es como se consigue poder, a través de métodos algebraicos, recuperar información perdida dentro de un proceso comunicativo.

Por otro lado, en el desarrollo en prácticas, se ve una aplicación real de la disponibilidad de la información. Se debe buscar la instantaneidad de los datos para el trabajo en tiempo real, así como la capacidad de acceso múltiple al sistema sin perdidas de calidad de servicio.

Palabras clave

Gemelo Digital, Dashboard, Codificación, Detección de errores.

Keywords

Digital Twin, Dashboard, Coding, Error detecting.

Índice general

Capítulo 1

Introducción

Cada día, aumentan el uso de tecnologías inteligentes, capaces de generar y transmitir miles y miles de datos. Estos datos viajan por las redes, ya sean estas locales (LAN) o globales (Internet), que estan llenas de ruido. En este movimiento de datos, para evitar errores y fallos en la información, se necesita de una respuesta contundente y eficaz. Este proyecto se basa completamente en el manejo de datos, su disponibilidad y su integridad.

Dentro de la informática se ha buscado siempre la velocidad, ya sea de computación de datos o de acceso a las redes (información viajando a la mayor velocidad posible). Y es que, poder manejar datos con presteza, se ha convertido en una de las tareas más significativas dentro de la informática.

En el ámbito concreto de la empresa, nos centramos en el manejo de datos en tiempo real para su impresión en pantalla. Y es que *la información es poder*[I]. Tener la disponibilidad de los datos de una planta cerámica (al igual que de cualquier otro ámbito laboral), supone un beneficio logístico muy relevante.

Cuando manejas información, esta se puede alterar para mandarla por un medio. Este tipo de alteraciones se conoce como codificación de la información. Un emisor codifica un mensaje, lo lanza por un medio, y lo recibe el receptor, este lo decodifica (si no lo hiciera, no tendría acceso a la información) y lee el mensaje. Este tipo de comunicaciones es la que se emplea hoy día. Este proceso se lleva a cabo de esta forma para que nada ni nadie interfiera entre emisor y receptor.

[I]Frase atribuida a Hobbes en su famosa obra El Leviatán.

Capítulo 2

Estancia en prácticas

2.1. Introducción

La estancia en prácticas se ha desarrollado en el Instituto de Tecnología Cerámica (ITC), un convenio nacido de la cooperación entre la Universidad Jaume I de Castelló (UJI) y la Asociación de Investigación de las Industrias Cerámicas (AICE) que, a través de dicha sinergia entre la universidad y la empresa, dan respuesta a las necesidades de las empresas cerámicas españolas.

Esta empresa tiene dos sedes diferenciadas, la primera en la propia universidad, dentro del campus; la segunda, en Almassora, en el polígono industrial SUPOI-8, siendo este segundo lugar el empleado para el desarrollo de la estancia en prácticas. Fueron realizadas en el departamento de Desarrollo de Tecnologías y Procesos Industriales.

2.2. Objetivos del proyecto formativo

Se buscaba la ingesta masiva de datos generados por un Gemelo Digital, esto es una representación de modelos físicos en una máquina virtual. Este Gemelo Digital fue creado para simular, de la forma más ajustada posible, una planta cerámica en funcionamiento.

Este sistema lanza datos en forma de *logs*, que son grandes cantidades de datos en forma de traza textual, los cuales contienen información detalla de consumos, productividad, etc. de cada una de las máquinas, en un instante de tiempo concreto, dentro del modelo generado del Gemelo Digital. Estos datos deben ser leídos y dispuestos según la forma acordada para el proyecto.

Todos los datos generados y leídos se deben almacenar en una base de datos apropiada para el uso que se les quiere dar. En este caso se empleó MongoDB como base de datos. Esta base de datos es de tipo no relacional, orientada a documentos. Por tanto, surge una necesidad, cambiar la base de datos relacional por la nueva base no relacional, es decir, se debía migrar la base de datos.

La consecución de cada uno de estos tres objetivos, ingesta de datos, transformación y preprocesamiento y, finalmente, el almacenamiento correcto en la base de datos correspondiente, es lo que dará como resultado un *Scraper* de datos. Un *Scraper* es un código, en este caso escrito en *Python*, que recolecta datos de los *logs* del Gemelo Digital y los guarda en la base de datos de *MongoDB* tal como se requiere para su posterior consulta o uso en otros programas.

Una vez se tienen disponibles los datos necesarios en la base de datos, se toman estos para generar un *Dashboard*. Un *Dashboard* es una herramienta de gestión de la información que monitoriza, analiza y muestra de manera visual los datos que hemos recopilado para hacer un seguimiento del estado de cada una de las máquinas y procesos de la representación de la planta cerámica. Este toma los datos desde la base de datos de *MongoDB* y las presenta en una web de carácter local. Un ejemplo es el de la figura 2.1

Para terminar, se buscaba lanzar varios Gemelos Digitales de forma simultánea. Una ejecución paralalela de todos ellos, donde el *Scraper* de cada uno guarde los datos sobre una misma base de datos, gestionada por un único administrador. Para esta tarea se requiere un registro previo, que permite a un usuario tener acceso únicamente a su parte de la base de datos y no a otras, dejando los privilegios al administrador.

2.3. Explicación detallada del proyecto realizado en la empresa

Dado el volumen de tareas, separaremos cada una de ellas y las estudiaremos en detalle. Así conseguiremos profundizar tanto en la metodoloía que ha llevado este proyecto como en el funcionamiento concreto de cada recurso empleado y la razón por la que se elegió.

Cabe destacar el carácter confidencial de este proyecto por lo que no se mostrarán aspectos técnicos relacionados con el mismo directamente. No aparecerá el código real, solo las ideas aproximadas, esquemas y herramientas empleadas para el desarrollo del proyecto.

Se definen ahora las herramientas que vamos a emplear. Entre estas herramientas destacan las siguientes:

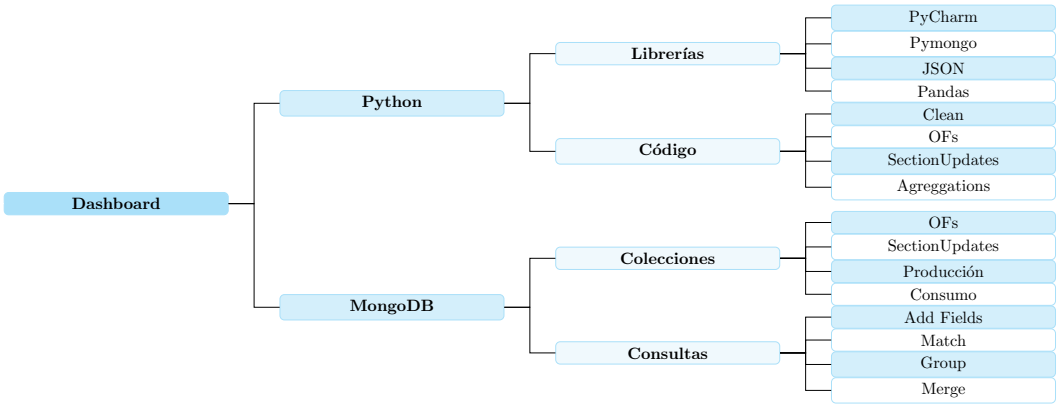
- **Gemelo Digital:** Esta herramienta es de gran importancia en el proyecto. Un Gemelo

Digital es una representación virtual de un proceso físico en tiempo real. Este Gemelo Digital representa una fábrica cerámica completa. Lanza los datos que representan cada proceso dentro de la fabricación de una planta cerámica. Los datos que lanza están relacionados con la productividad y el consumo, así como datos relacionados con los órdenes de fabricación o porcentajes de acabado de dichos órdenes. El Gemelo Digital será la fabrica que se va a analizar a través del *Dashboard*.

- ***Python*:** Este es el lenguaje de programación que se va a emplear para la resolución completa del proyecto. Se elige este lenguaje por encima de otros debido a su gran versatilidad, facilidad de uso y aprendizaje y por la gran cantidad de librerías que tiene para un uso especializado. La librería que empleamos, entre muchas otras, es *Pymongo*. Destacan otras como *JSON* o *Pandas*. El entorno de desarrollo sobre el que se trabaja es *PyCharm*, por su fácil acceso.
- ***MongoDB*:** Todo proyecto que busca trabajar con datos, se debe apoyar en una buena base de datos. La que se eligió para el proyecto fué *MongoDB*. Una base de datos orientada a documentos, lo que permite almacenar datos semi-estructurados. Esto vale para alterar libremente las bases de datos a lo largo del proceso de fabricación. Se eligió esta base de datos debido a la gran escalabilidad que posee.

Con todas las herramientas claras, se puede comenzar a detallar los pasos seguidos durante el proyecto, los tiempos de los mismos, así como las ideas que se han tenido a lo largo de dicho proyecto.

2.3.1. Metodología y definición de tareas



Este esquema representa una jerarquía completa del proyecto, donde cada parte a desarrollar necesita todas las anteriores. Procedemos ahora a la explicación completa de cada una de las

partes.

Como se ha comentado anteriormente, el lenguaje de programación utilizado a lo largo de todo el proyecto es *Python*, se ahondará ahora en las librerías.

- *PyCharm*: No es una librería, este es el entorno de desarrollo. *PyCharm* permite trabajar fácilmente, con una jerarquía bien definida. Permite un cómodo desarrollo de código con la predición de texto y métodos guiados. Un ejemplo del entorno es la figura 2.2. Otra de las ventajas, es la descarga y uso de las librerías. Para descargar una librería y poder usarla, lo primero que se hace es ir a *File* → *Settings* → *Project: (Nombre Proyecto)* → *Project Interpreter* y donde aparece el símbolo + en la figura 2.3 se añaden las librerías que se deseen.
  Para usar la librería una vez descargada, dentro del código de programa, se escribe el comando ***import (Nombre de la librería)***. Desde este punto, se podrán emplear los métodos y funciones que dicha librería tenga implementados.

Figura 2.1: Ejemplo de un *Dashboard*

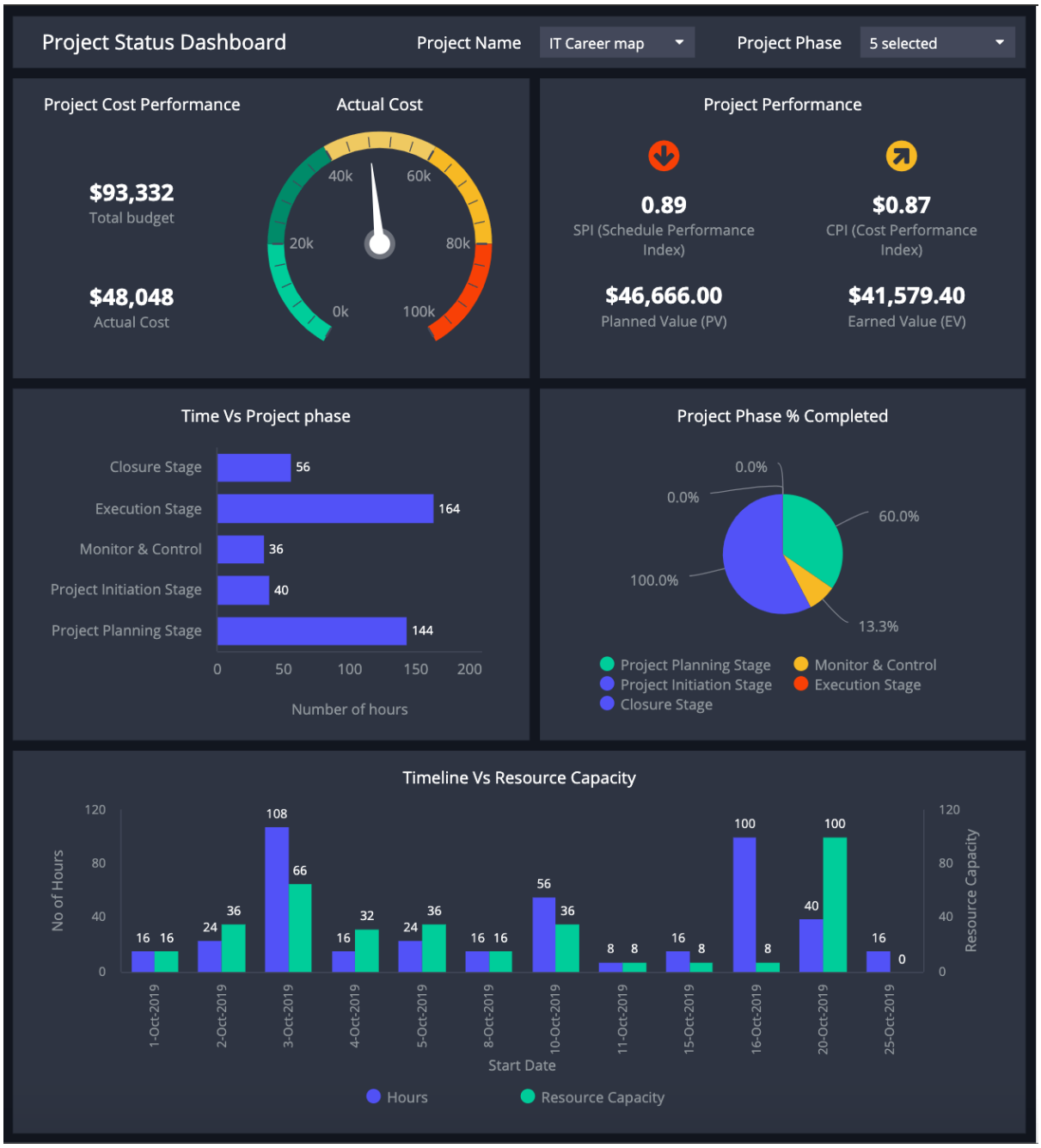
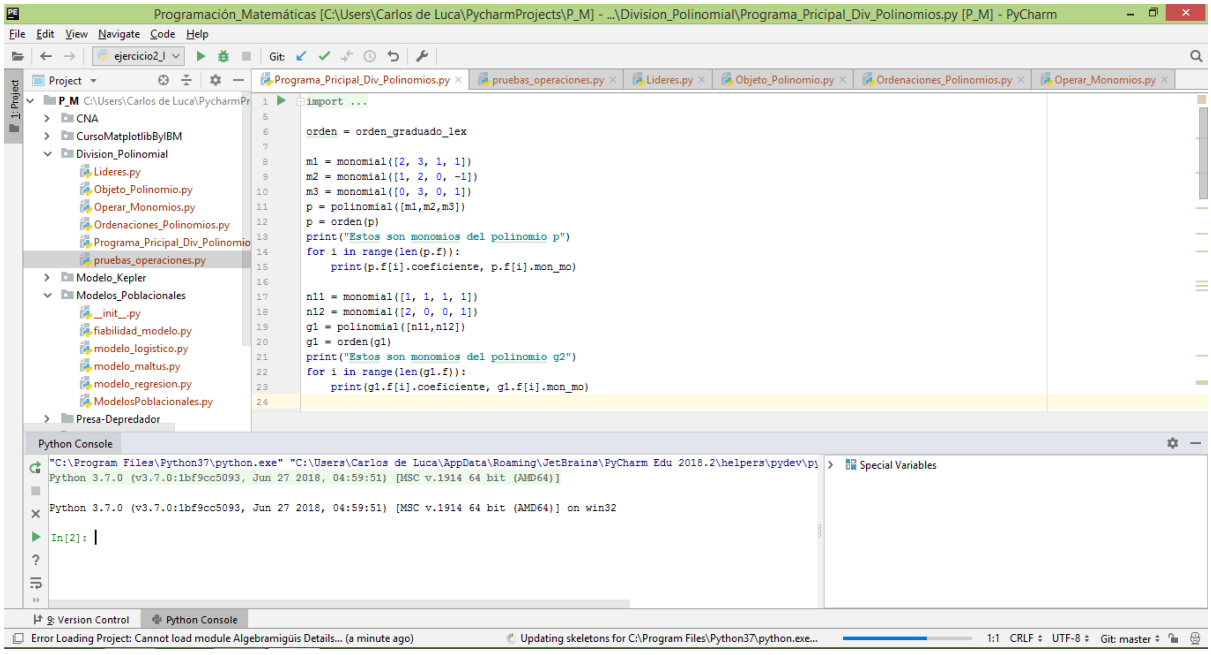


Figura 2.2: Ejemplo del IDE PyCharm



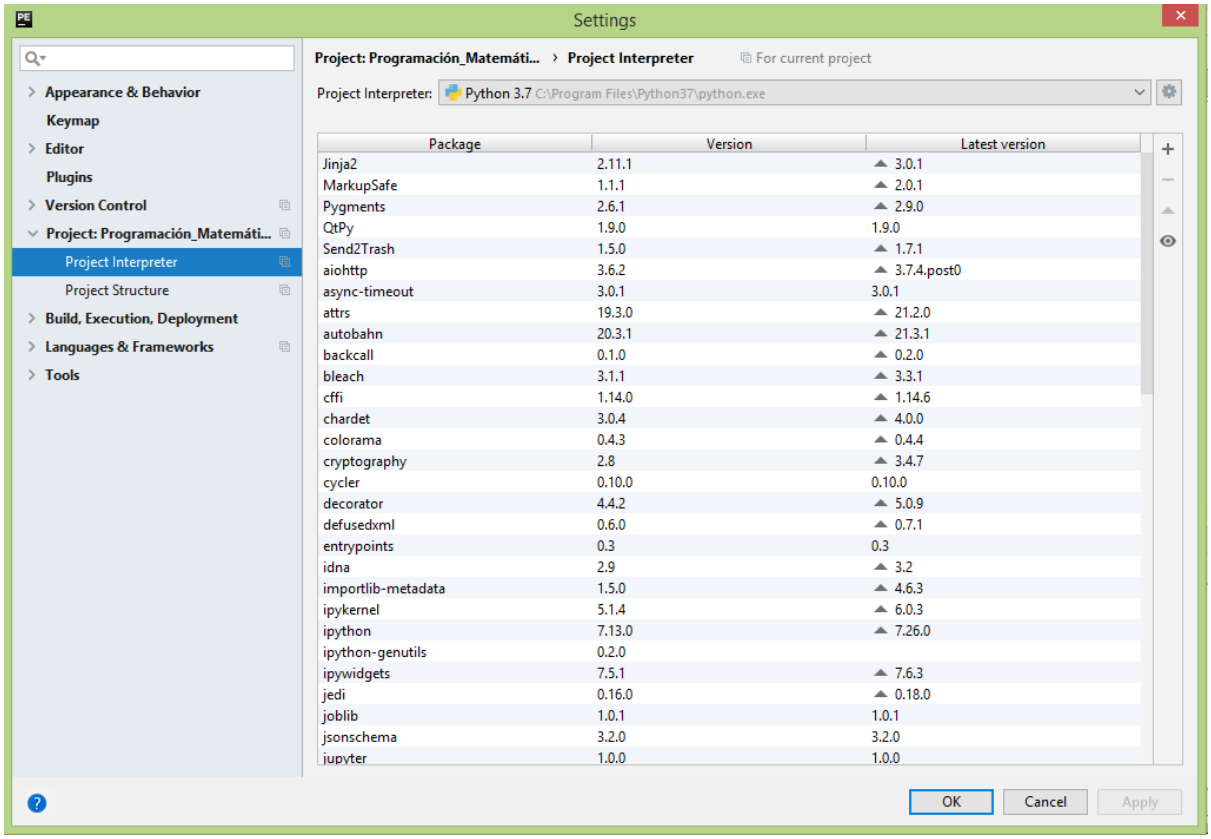


Figura 2.3: Página de configuración para insertar nuevas librerías en PyCharm

- *Pymongo*: Esta librería es la que permite trabajar con las bases de datos de *MongoDB* desde *Python*. La documentación de esta librería se puede encontrar en *Pymongo Docs* o en la url `https://pymongo.readthedocs.io/en/stable/`.
  La clave del uso de esta librería es su secuencialidad, es decir, siempre sigue unos mismos pasos. El primero de ellos es conectarte a la base de datos. Para ello se debe saber que una base de datos en MongoDB tiene un cliente. Un cliente es el host asociado a un banco de bases de datos. El usuario que tiene permiso absoluto sobre el cliente se denomina *root*. Dentro del cliente puedes tener diversas bases de datos. Los usuarios con permisos absolutos sobre las bases de datos se denominan *Administradores* o *Admins*. Los que no disponen de dichos permisos se conocen como *Usuarios* o *Users*. Para acabar, cada base de datos dispone de distintas colecciones. Estas colecciones se componen de documentos, que es la unidad mínima de información de *MongoDB*. Un ejemplo de conexión a una colección sería:

```
###

import pymongo #importamos la libreria completa

#inicializamos el cliente y lo cargamos
#como variable
client = MongoClient('mongodb://localhost:27017/')

#cargamos en una variable la base de datos
#usando la variable client
db = client['test-database']

#se aplica el mismo procedimiento para la coleccion
collection = db['test-collection']

#ejemplo de documento, diccionarios o elementos JSON
post = {"author": "Mike",
        "text": "My first blog post!",
        "tags": ["mongodb", "python", "pymongo"],
        "date": datetime.datetime.utcnow()}

collection.insert_one(post) #ejemplo de insercion
                            #de un documento

###
```

A partir de aquí, el resto de trabajo que puede realizarse a través de esta librería, se realiza con las funciones de *Pymongo*. Las funciones que pueden realizarse, se relatan en la documentación mencionada anteriormente.

- *JSON (JavaScript Object Notation)*: Es un formato de intercambio de datos de carácter ligero, sus documentos pesan relativamente poco. Cuando se habla de pesos, se habla de tamaños de archivo. Cuanto más grande sea un archivo, más pesado será y mayor es el coste de trabajar con tal archivo. Por este motivo se ha elegido como fuente de los documentos los archivos *JSON*.
  Además, es un tipo de documentos que son sencillos de escribir (como diccionarios en *Python*) y de leer (de forma análoga a la anterior). La estructura completa de un archivo *JSON* se describe en la figura 2.4. Como puede observarse, este tipo de elementos encajan a la perfección con la estructura de documentos de *MongoDB*. Para poder leer grandes cantidades de estos documentos, así como para poder generarlos, se debe apoyar el proyecto en la última librería descrita en nuestro esquema, *Pandas*.

- *Pandas*: Es una librería especializada en el manejo y análisis de estructuras de datos. Se basa en las estructuras de otra librería llamada *Numpy*. Se ha elegido esta librería por la

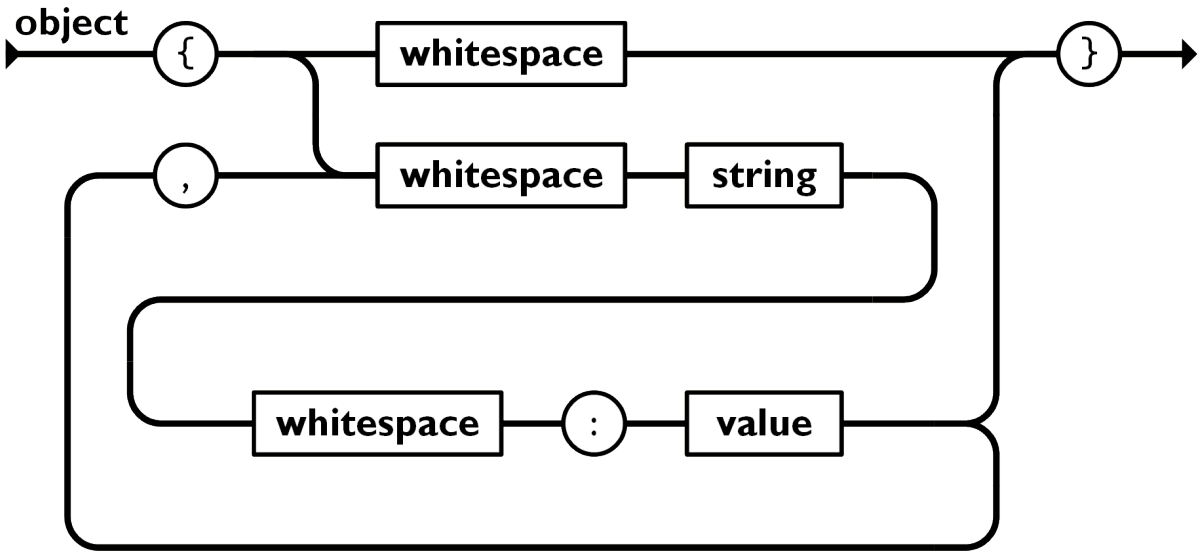


Figura 2.4: Esquema de objeto JSON

facilidad que existe a la hora de transformar *DataFrames* en objetos *JSON*. Esto se hace de una forma directa. Con una orden es suficiente para poder realizar tal transformación. A continuación se presenta un sencillo ejemplo de transformación y lectura de un *DataFrame* a *JSON*.

```
###
import pandas as pd
import json

#Ejemplo de creacion de un DataFrame
DF = pd.DataFrame(    [["a", "b"], ["c", "d"]],
                      index=["row 1", "row 2"],
                      columns=["col 1", "col 2"],
                  )

#Transformacion en una sola linea. Si cambiamos la
#orientacion, cambiamos la disposicion del resultado
result = DF.to_json(orient="index")

#Cargamos el archivo JSON para su lectura
parsed = json.loads(result)

#Y lo mostramos por pantalla
json.dumps(parsed, indent=4)
```

```
###

El resultado se sigue como:

{
    "row 1": {
        "col 1": "a",
        "col 2": "b"
    },
    "row 2": {
        "col 1": "c",
        "col 2": "d"
    }
}
```

Para ver las funcionalidades completas de *Pandas*, basta con acudir a su *documentacion* o a la url https://pandas.pydata.org/docs/.

Pues bien, se ha visto cómo funcionan las librerías de *Python*. La siguiente parte se centra en las colecciones principales de nuestra base de datos en *MongoDB*.

- *OFs (Órdenes de Fabricación)*: Dentro de una planta de industria cerámica, cada día se producen grandes cantidades de lotes de producto. Estos lotes tienen asociado un orden de fabricación. Esto es lo que representa esta colección en nuestra base de datos del cliente que tengamos en *MongoDB*.
  Estos órdenes siguen un patrón de datos concreto, como fechas de inicio de cada fase (prensado, secado...), porcentajes de acabado o cosumo electrico total empleado para ese orden concreto. Cada documento tiene un identificador único, que es un código numérico en la etiqueta 'OF', su número de orden. Esta estructura de etiquetas, salvo las que requieren que el OF haya acabado, se precargan. La precarga de etiquetas y datos, es una inicialización por medio de *Python*, con las etiquetas que deben comenzar en valores preestablecidos. En cambio, la parte que quedaría por actualizar, se realiza a lo largo de las lecturas de los *DataFrames* que nos dan los *logs* del Gemelo Digital. Estas lecturas se reflejan en las siguientes colecciones.
- *SectionUpdates*: Debemos tener en cuenta el carácter físico del proceso de producción de una planta cerámica. El tiempo es la variable sobre la que trabajamos en el modelo. Todo transcurre a través del mismo. Por lo tanto, se necesita tener el último instante de tiempo o, equivalentemente, el último dato que se toma. Para obtener este dato se tienen dos formas de lograrlo.
  La primera se haría a partir de las consultas de *MongoDB*. Esta forma es algo pesada de obtener puesto que hay que entrar en consultas en ese lenguaje (se hablará a lo largo

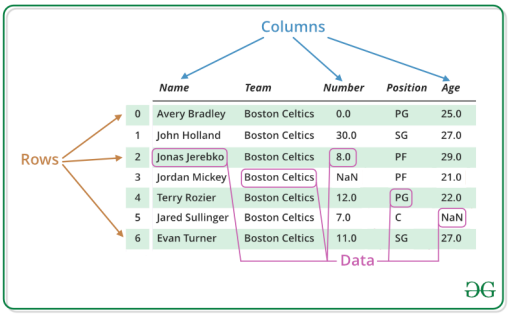


Figura 2.5: Estructura de un DataFrame en Pandas

del capítulo sobre estas consultas) y las consultas dependen del volumen de datos para su velocidad de computación.
La segunda forma es la que se ha empleado durante el proyecto. Esta forma aprovecha la estructura de datos de los *DataFrame* de *Pandas*. Este *DataFrame* sigue la estructura de la figura 2.5. Esta forma de estructurar los datos, nos permite tener el último dato leído en la ultima posición de la tabla. Así, tomamos el dato final de la tabla en cada lectura de un *DataFrame* y actualizamos la colección de *SectionUpdates*. Esta colección tiene un número fijo de documentos, cuyo identificador es la máquina. Hay un documento por máquina de la planta cerámica

- *Produción*: En esta colección se guardan todos los datos que se recopilan sobre la producción de la planta. En esta producción se mide desde el porcentaje de acabado hasta las emisiones de los procesos. Estos datos se emplean para medir el rendimiento de las máquinas. Cabe destacar el almacenamiento masivo de datos en esta colección. Es de ellos de los que se sacarán estadísticas aplicando mecanismos de agrupación de datos en *MongoDB*.

- *Consumos*: Esta colección complementa a la anterior descrita. En este caso, se guardan de forma masiva los datos referentes a consumos de planta. Estos datos van desde el consumo en energía electrica hasta los precios del gas empleado durante el orden de fabricación concreto. Estos datos se consolidarán en otras colecciones. Estas colecciones serán Consumos Diarios, Consumos Mensuales y Consumos Anuales. La relación entre los datos de Producción y Consumo nos dará la eficiencia de la planta durante un período de tiempo determinado. Esto se reflejará de forma futura en el visor de datos (*Dashboard*).

Ahora, se explican los procesos para la concentración de datos masivos. Antes se hablaba de emplear métodos de agrupación en *MongoDB*. Estos métodos se conocen como *Agreggations* y son una forma eficiente de condensar datos. Se emplean las funciones que ya vienen preestablecidas en la base de datos por motivos de sencillez y eficacia. Se explicarán ahora el funcionamiento por capas del *pipeline* y que métodos empleamos.

Dentro de la interfaz de *MongoDB* existe una pestaña llamada *Agreggations*. Esta sirve para

poder condensar datos de una o varias colecciones. Funciona por capas, es decir, cada filtro que aplicas a los datos da como resultado una nueva colección de datos con aquellos datos que pasan el filtro. Esta colección se queda en memoria y se pueden seguir aplicando más y más filtros. Todos los filtros empleados de forma consecutiva forman el *pipeline*. La interfaz se representa en la figura 2.6. Se explicarán ahora los pricipales que se han desarrollado durante el proyecto.

- *$addField*: Este filtro permite añadir nuevos campos de clave-valor a todos los documentos de la colección donde se aplica. También permite sustituir uno ya creado, basta con poner la etiqueta correspondiente con el mismo nombre que el que se busca sustituir. Para mayor información sobre este método se puede visitar https://docs.mongodb.com/manual/reference/operator/aggregation/addFields/.
  Un ejemplo de uso sería:

```
###
//Esta representacion se realiza en las agreggations de MongoDB
{
    $addFields: {
       etiqueta_1: { $op: "$referencia_1" } ,
       etiqueta_2: { $sum: "$referencia_2" },
       ...
    }
}
###
```

- *$match*: Esta opción o filtro genera una colección en memoria que solo tiene elementos coincidentes con la busqueda. Esto permite hacer un cribado de todos los datos. Para más información, la documentación de esta función está en https://docs.mongodb.com/manual/reference/operator/aggregation/match/. Un ejemplo de uso sería:

```
###
//Esta representacion se realiza en las agreggations de MongoDB
{
    $match : {
        etiqueta : "dato_concreto"
        }
}
###
```

- *$group*: Esta función es la encargada de condensar datos. Toma todos los de la colección que esté cargada en la memoria del *pipeline*. Existen diversas opciones para realizar operaciones con los datos, estas se conocen por *accumulators*. Para ampliar información, se tiene la documentación en https://docs.mongodb.com/manual/reference/operator/aggregation/group/. Un ejemplo de uso es:

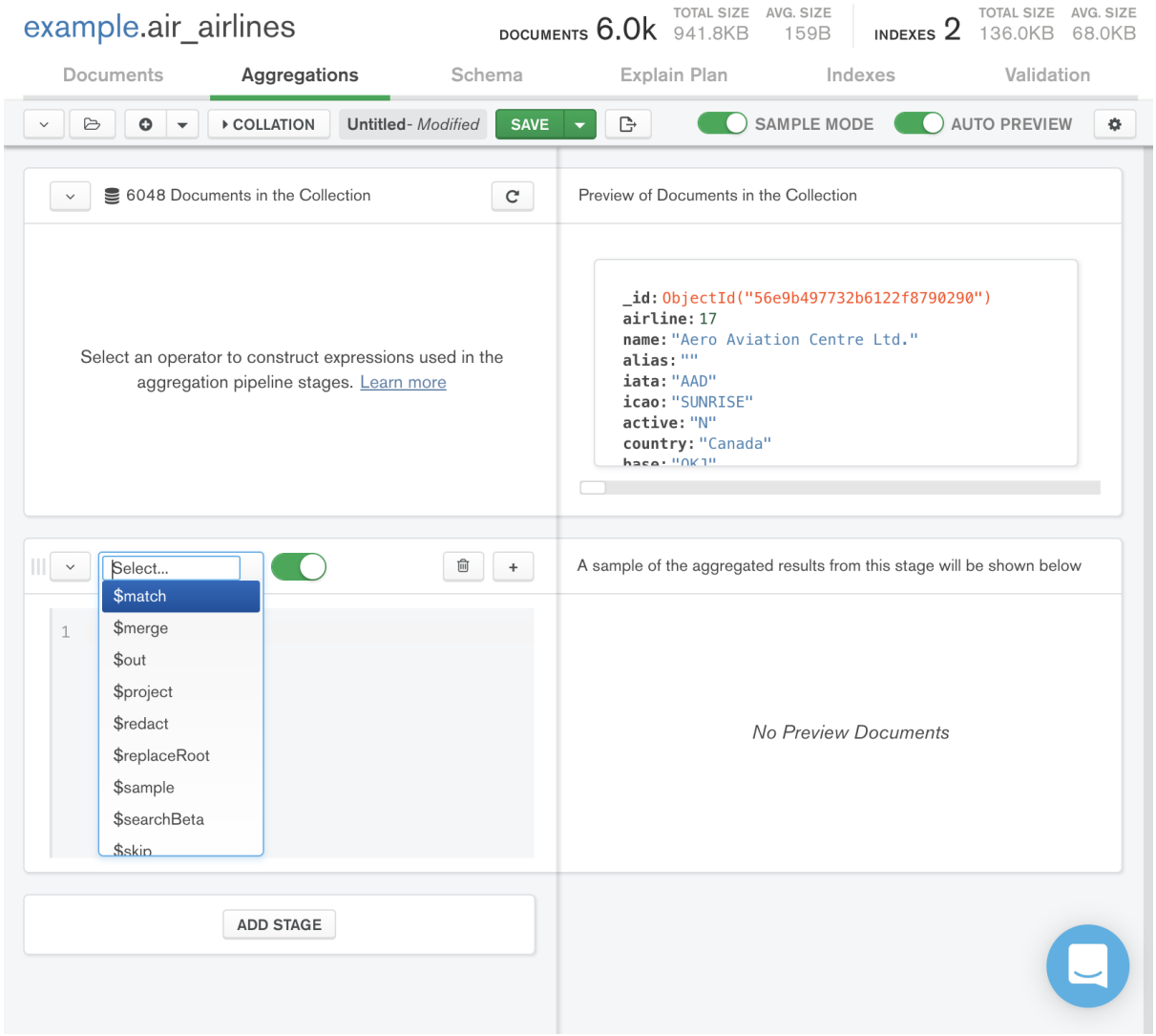


Figura 2.6: Interfaz de la opción Agreggation de MongoDB

```
###
//Esta representacion se realiza en las agreggations de MongoDB
{
    $group: {
        _id: "nombre_concreto",
        etiqueta_1: { acumulador_1: expresion sobre la que se aplica},
        etiqueta_2: { acumulador_2: expresion sobre la que se aplica},
        ...
    }
}
###
```

- *$merge*: Con esta función volcamos el resultado del *pipeline* en una nueva colección con el nombre deseado. A partir de ese momento, en la base de datos existirá una nueva colección con el resultado completo de nuestras consultas. La información ampliada se encuentra en https://docs.mongodb.com/manual/reference/operator/aggregation/merge/. Como ejemplo de uso se tiene:

```
###
//Esta representacion se realiza en las agreggations de MongoDB
{
    $merge: { into: "nombre de la nueva coleccion"}
}
###
```

La combinación de todas las anteriores son las empleadas en el proyecto. Conservando ese orden para generar los resultados concretos que se buscaban (en nuestro caso se buscaba la agrupación de datos en consumos diarios, mensuales y anuales). Para generar distintos resultados basta con jugar con diferentes métodos o funciones y con su posición en el *pipeline*. Todas las funciones se encuentran en la documentación de *Agreggations* de *MongoDB* o en la url https://docs.mongodb.com/manual/reference/operator/aggregation/.

Es el momento de ver la parte del código. Cada uno de los códigos realizados han complementado o completado otros que estaban ya hechos dentro del proyecto. Ahora que se tienen todas las piezas, se puede representar la estructura lógica del proyecto y escribir pseudocódigo sobre el mismo.

Cada vez se quiere lanzar una nueva batería de datos para probar nuestro código o cada vez que se quiere simplemente borrar información de una base de datos por el motivo que sea, se empleará un *script* que ejecuta un sencillo código, *Clean*. Este código será siempre la primera parte a ejecutar. Esto es así para que, prueba tras prueba, no se llene nunca la memoria de datos.

```
###

from pymongo import MongoClient

#El primer paso es conectarse al servidor de mongo

client = MongoClient('mongodb://localhost:27017')

#una vez realizada la conexion, hacemos lo siguiente:

client.drop_database('nombre_base_de_datos')

###
```

Tras la limpieza de la base de datos, se pueden comenzar a crear todas las partes fijas de nuestra nueva base de datos. Estas partes se corresponden con las colecciones *OFs* y *SectionUpdates*. Los identificadores de las órdenes de fabricación vienen dadas en un archivo del tipo *JSON*. Las máquinas de la planta a analizar vienen dadas en la configuración inicial. Esta se guarda en un archivo *config.ini*.

```
###

from pymongo import MongoClient
import os
import json

client = MongoClient('mongodb://localhost:27017')
db = client['base_de_datos']

script_dir = os.path.dirname(__file__) #directorio absoluto
rel_path = "config.ini" #nombre archivo buscado
abs_file_path = os.path.join(script_dir, rel_path)

machines = readFile(abs_file_path) #Metodo ya implementado anteriormente
#Devuelve un diccionario con par clave-diccionario(clave-numero)

with open('ofs.json') as f:
    ofs = json.loads(f) #Informacion de las OFs en un vector de diccionarios

#Parte de SectionUpdates

v_maquinas = []
```

```
for m in machines['Equipos']:
    for k in m:
        v_maquinas.append([{ 'equipo': k+m[k], #equipo y numero
                             'etiqueta_1': valor_por_defecto,
                             ...,
                                }])

db['SectionUpdates'].insert_many(v_maquinas)
db['OFs'].insert_many(ofs)

###
```

Cada vez que el programa comienza a ejecutarse, limpia la base de datos anterior y genera las partes básicas como hemos visto. El Gemelo Digital comienza ahora a lanzar datos, pasan por un lector sencillo de documentos *logs* que tenían previamente creado en la empresa. Esta parte transforma esos datos en un *DataFrame* de *Pandas*. Una vez transformados dichos datos, se realiza una lectura del final del *DataFrame* y se actualizan los registros de la colección de *SectionUpdates*.

```
###

/Viene en el codigo de lectura una parte
/que llama al codigo de actualizacion

def UpdateCollection(db, DF): #tiene como parametros la base de datos
                              #y el DataFrame
    last_data = DF.iloc[[0, -1]] #Extraemos la ultima fila
    final_update = {}
    for c in last_data.columns:
        final_update[c] = last_data[c]

    db['SectionUpdates'].update_many({'equipo': final_update['equipo']}
                                     ,{'$set': final_update})

###
```

Por último, queda analizar cuando se lanzan la agrupación de datos. Pues bien, una parte del código analiza el cambio de día. Esto se realiza de forma sencilla viendo si la fecha de la primera columna del *DataFrame* y la fecha de la última columna difieren. Si esto es así, se ha producido un cambio de día y se pueden agrupar los datos.

```
###
```

```
def DataAgreggationsDay(fecha, db):
    db.agreggate([
        { '$addFields': {...}    },
        { '$match': {...}    },
        { '$group': {...}    },
        { '$merge': {...}    }
    ])

/donde se ponen los datos correspondientes que se quieren agrupar
/tal como se hacia en los ejemplos

#El mismo principio se aplica a

def DataAgreggationsMonth(fecha, db):
    ...

def DataAgreggationsYear(fecha, db):
    ...

###
```

Ahora bien, para poder condensar o agrupar datos, debemos tener una extensa base de datos con la que trabajar. Esta se genera tras la transformación del *DataFrame*. Cada uno de ellos se guarda al completo, de forma similar al código representativo de *SectionUpdates*. Se disponen en dos colecciones distintas según si los datos pertenecen a *Consumos* o a *Producción*.

Todo este proceso deja la disposición buscada de los datos en la base de datos. Solo queda leer los datos desde el programa del *Dashboard*. Para ello, en el código dado por la empresa, basta con añadir el arranque del cliente y la base de datos. Una vez realizado, leer los datos es sencillo, se emplea el método *find*.

```
###

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017')
db = client['base de datos']
col = db['coleccion']

for doc in col.find():
    /nos da cada documento de la coleccion

###
```

Se tiene así, una idea aproximada de cómo se realiza la gestión de todos los datos de nuestro proyecto. Solo falta arreglar el arranque simultaneo de distintas bases de datos. Para ello se toma un registro de usuario y contraseña.

```
###

from getpass import getpass
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017')
client.admin.authenticate('Admin', 'AdminPass')

user = input('Nombre de usuario: ')
pwd = getpass('Password: ')

client.testdb.add_user(user, pwd,
    roles=[{'role':'readWrite','db':'base de datos'}])

###
```

Para arrancar el cliente con los permisos correspondientes, en la configuración inicial se guarda el usuario con su contraseña. Modificamos cada entrada al cliente.

```
###

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017',
                     'user',
                     'password')

###
```

Se puede ahora, arrancar múltiples simulaciones de distintos Gemelos Digitales. En este punto se da por finalizado el proyecto.

2.3.2. Planificación temporal de las tareas

Debido a la naturaleza propia del proyecto y a la imperiosa necesidad del desarrollo de partes previas para el total. Se han ido realizando las tareas en el orden descrito. Se ha asegurado el correcto funcionamiento de cada una de las partes para poder avanzar a la siguiente.

Los problemas que podían surgir en cualesquiera de las partes del proyecto al desarrollar las nuevas, se fueron subsanando de igual forma antes de continuar a las partes posteriores.

Tambíen se reajustaron, con cautela, múltiples etiquetas para nuevas entradas según avanzaba el proyecto. Estos cambios forzaron la reestructuración de las partes anteriores.

2.3.3. Estimación de recursos del proyecto

En este proyecto se ha llevado a cabo completando partes ya desarrolladas de otro anterior. Se disponía de la estructura del *Dashboard*, cuya implementación estaba completa salvo por la lectura de datos desde *MongoDB*.

Otra implementación, que también fue creada previamente antes de comenzar el proyecto, era la transformación de archivos *logs* en *DataFrames*. Con esto solo se debían migrar los datos a *MongoDB* y hacerlo de forma automática.

Los datos se lanzaban desde un simulador, un Gemelo Digital. Este era otro proyecto que había creado previamente la empresa. Lo empleabamos para las baterías de pruebas y ver como se generaban los bancos de datos.

Para realizar este trabajo, se ha requerido de un equipo informático sencillo en la oficina donde se desarrollo el proyecto. Para los días que debía trabajar telemáticamente, se empleó un programa de conexión remota.

2.3.4. Grado de consecución de los objetivos propuestos

El proyecto se completó tal y como se había previsto. Todos los fallos que se encontraron durante las baterías de pruebas se arreglaron. El proyecto es una primera versión de un programa de seguimiento de datos en tiempo real en una planta cerámica.

Conclusiones

Cuando se trabajan con grandes bancos de datos, cada mínimo detalle cuenta para la eficiencia del programa. Es de vital importancia hacer una estructura eficiente en una base de datos eficiente. Esto mejora el proceso de toma e impresión.

Esta es una de tantas lecciones que he aprendido durante la estancia en prácticas.

En la parte técnica he aprendido a buscar información veraz y de calidad, manejo con bases de datos no SQL. Esto me hizo pensar de una forma distinta a lo que había aprendido durante la carrera, ya que en esta aprendí sobre SQL.

La presentación de los datos, que infuirá directamente en la decisión final de los potenciales clientes. Todos los detalles, facilidad en uso, y, hasta la combinación de colores influye en la decisión. Es una parte donde hay que encontrar un equilibrio entre eficiencia y acabado.

Indudablemente, he mejorado en la programación en *Python*. Así como en el manejo de las librerías especificas de las que se han hablado durante todo el proyecto.

En cuanto a lo humano se refiere, puedo decir que, cuando hay un buen ambiente de trabajo, este se disfruta. Me he encontrado gente maravillosa a lo largo de la estancia en prácticas.

En definitiva, esta estancia en prácticas me han hecho grandes aportes académicos. Así como crecimiento personal.

Capítulo 3

Memoria TFG

3.1. Motivación y Objetivos

La información y la comunicación están a la orden del día. Ser capaces de asegurar que dicha comunicación sea posible, de forma íntegra y segura, es la fuente de estudio de muchos proyectos.

En nuestro caso, este proyecto centrará sus esfuerzos en desarrollar y explicar la teoría algebraica que acompaña a la codificación de dicha información.

El objetivo principal de los parrafos que a continuación se siguen, es dar una fundamentación teórica rígida y sólida sobre la dotación de estructuras algebraicas sobre la que se basa la teoría de la codificación.

3.2. Teoría de la Información

La piedra angular de este trabajo, es el estudio de codificación de la información. Para ello, deben entenderse ciertas nociones básicas para poder comprender, en un mayor grado de profundidad, toda la evolución de los códigos y su estructuración.

Definición 3.2.1 (Alfabeto y sus palabras)**.** La información a tratar, viene estrita como una sucesión de símbolos. Este conjunto de sucesiones de símbolos se conoce como alfabeto. Se representa por $\mathcal{A}$. Llamamos palabra a cualquier secuencia finita de elementos de $\mathcal{A}$. El conjunto de las palabras escritas en $\mathcal{A}$ se denota por $\mathcal{P}(\mathcal{A})$.

Definición 3.2.2 (Codificación de alfabetos)**.** Codificar el alfabeto fuente $\mathcal{A} = \{a_1, \ldots, a_m\}$ en el alfabeto $\mathcal{B} = \{b_1, \ldots, b_q\}$ es formalizar una aplicación inyectiva $c : \mathcal{A} \to \mathcal{P}(\mathcal{B})$. Para cada $a_i \in \mathcal{A}$, se tiene que $c(a_i)$ es la codificación de a_i. Al subconjunto $\mathcal{C} = Img(c)$, se le conoce como código empleado.

Las palabras que contiene un código $\mathcal{C}$ pueden tener longitud variable o idéntica. Si todas las palabras tienen idéntica longitud, se conocen como códigos en bloque. Se detallarán más adelante.

Cuando se codifica un mensaje, se codifica un mensaje escrito en el alfabeto $\mathcal{A}$. Por lo que la aplicación c se extiende a otra aplicación $c' : \mathcal{P}(\mathcal{A}) \to \mathcal{P}(\mathcal{B})$ con $c'(a_{i_1}a_{i_2}\cdots a_{i_n}) = c(a_{i_1})c(a_{i_2}))\cdots c(a_{i_n}))$.

Para que exista el proceso de decodificación única y no induzca a error la forma de codificación elegida, se debe tener que la aplicación c' sea inyectiva.

3.2.1. Códigos instantáneos

Sea $\mathcal{C}$ un código de $\mathcal{A}$ sobre $\mathcal{B}$. Para que $\mathcal{C}$ sea de interés, se busca que tenga decodificación única. Para asegurar dicha decodificación única, se impondrá la siguiente condición: ninguna de las palabras del código puede formar parte de otra como prefijo. Esto nos da la siguiente definición.

Definición 3.2.3 (Código instantáneo)**.** Diremos que un código es intantáneo si ninguna palabra código es prefijo de otra.

Proposición 3.2.1. Todo código instantáneo tiene decodificación única.

Uno de los códigos instantaneos más conocidos y estudiados, es el código Huffman o codificación Huffman. Este método de codificación instantánea se emplea en la compresión de información. Para ver cómo se crea un código instantáneo, así como para estudiar dicho código, acudir a [1].

Estos códigos se conocen así por la capacidad de decodificar el mensaje en tiempo real. Ahora se determinan bajo que condiciones exite un código instantáneo con m palabras de longitudes $l_1, \ldots, l_m$.

Teoremas de Karft y McMillan

Teorema 3.2.2 (Desigualdad de Kraft)**.** La condición necesaria y suficiente para que exista un código q-ario instantáneo con m palabras código de longitudes $l_1, \ldots, l_m$ ($l_i \geq 1$) es que se verifique la desigualdad

$$q^{-l_1} + \ldots + q^{-l_m} \leq 1.$$

Demostración. Se puede suponer que $l_1 \leq l_2 \leq \cdots \leq l_m$ (salvo reordenación de los elementos de $\mathcal{A}$). Determinemos ahora los $c(a_i)$. Tomemos $c(a_1)$ un elemento cualquiera de $\mathcal{P}(\mathcal{B})$ con longitud l_1. Elegimos ahora $c(a_2)$, de entre los q^{l_2} de $\mathcal{P}(\mathcal{B})$, uno cualquiera que no tenga como prefijo a $c(a_1)$. Como $q^{l_2-l_1} + 1 \geq q^{l_2}$, es posible hacerlo.
Supongamos tener los $c(a_1), \ldots, c(a_r)$ elegidos de esta forma con longitudes $l_1, \ldots, l_r$. Como en $\mathcal{P}(\mathcal{B})$ existen $q^{l_{r+1}}$ palabras de longitud l_{r+1}, de las cuales $q^{l_{r+1}-l_1} + \cdots + q^{l_{r+1}-l_r}$ tienen como prefijo alguna de las, ya construidas, palabras código. Ahora bien, tenemos

$$q^{-l_1} + \cdots + q^{-l_{r+1}} \leq q^{-l_1} + \cdots + q^{-l_m} \leq 1,$$

por lo que, al multiplicar por $q^{l_{r+1}}$ se obtiene la desigualdad buscada.
Recíprocamente, suponiendo la existencia de un código q-ario instantáneo de longitudes de palabra $l_1, \ldots, l_m$, entonces existe una palabra de longitud l_m en $\mathcal{P}(\mathcal{B})$ que no tiene por prefijo a ninguna de las $c(a_1), \ldots, c(a_{m-1})$, por lo que

$$q^{l_m} \geq q^{l_m-l_1} + \cdots + q^{l_m-l_{m-1}} + 1$$

y, al dividir por q^{l_m}, se da la desigualdad. □

Teorema 3.2.3 (Desigualdad de McMillan)**.** Todo código q-ario con m palabras de longitudes $l_1, \ldots, l_m$ ($l_i \geq 1$), que tenga decodificación única, verifica la desigualdad de Kraft

$$q^{-l_1} + \ldots + q^{-l_m} \leq 1.$$

Demostración. Sea $\mathcal{C}$ un código q-ario con decodificación única y sean $l_1, \ldots, l_m$ longitudes de palabras. Supongamos, al igual que antes, que $l_1 \geq \cdots \geq l_m$. Probaremos que $\lambda \leq 1$, siendo

$$\lambda = \sum_{i=1}^{m} q^{-l_i}.$$

. Operando, para cada entero positivo r, se obtiene

$$\lambda = \sum_{i_1,\ldots,i_r=1}^{m} q^{-(l_{i_1}+\cdots+l_{i_r})} = \sum_{j=rl_1}^{rl_m} \sum_{l_{i_1}+\cdots+l_{i_r}=j} q^{-j},$$

expresión que se obtiene mediante la agrupación de sumandos con igual exponente. Ahora bien, al ser $\mathcal{C}$ un código con decodificación única, el número de mensajes posibles $a_{i_1} \cdots a_{i_h}$ con codificación $c'(a_{i_1} \cdots a_{i_h})$ tiene longitud j, es menor que el número de elementos en $\mathcal{P}(\mathcal{B})$ de longitud j, o sea, que q^j. Por lo tanto $\#\{(li_1, \ldots, l_{i_r}) \mid l_{i_1} + \cdots + l_{i_r} = 1\} \leq q^j$, por lo que

$$\sum_{l_{i_1}+\cdots+l_{i_r}=j} q^{-j} \leq q^j q^{-j} = 1$$

y, en consecuencia, $\lambda^r \leq r(l_m - l_1)$. Ahora, si $\lambda > 1$ y $r \to \infty$, el infinito λ^r es de mayor orden que el infinito $r(l_m - l_1)$, lo que contradice la desigualdad, por tanto concluimos que $\lambda \leq 1$. □

3.2.2. Canales con ruido

En el traspaso de información mediante un canal, existen interferencias o alteraciones, estos se conocen como ruidos. Demos ahora unas breves nociones sobre los errores y los borrones o perdidas de información.

- Errores: Símbolos recibidos que difieren del enviado.
- Borrones: Símbolos recibidos que son ilegibles o imposibles de interpretar por el receptor (Se denotan por "␣"). Pueden ser:

  Aleatorios: Producidos de forma aislada y repartidos aleatoriamente en el mensaje.

  Ráfagas: Si se han sucedido en posiviones consecutivas en alguna parte del mensaje.

Un canal con ruido queda perfectaente determinado por:

- El conjunto de símbolos de entrada, $\mathcal{A} = \{a_1, \ldots, a_m\}$.
- El conjunto de símbolos de salida, $\mathcal{S} = \{s_1, \ldots, s_h\}$.
- Las relaciones estadísticas entre las entradas y salidas, es decir, para cada $i = 1, \ldots, h$, y cada $j = 1, \ldots, m$, la probabilidad condicionada $prob(s_i|a_j)$.

Centraremos nuestro estudio en los códigos y no en los canales. Los código sempleados para detectar y corregir errores son, usualmente, códigos en bloque. La idea radica en la introduccón de información redundante para poder realizar dicha detección y corrección.

Si se busca ampliar información relacionada con los canales, se recomienda leer [2]. Gran parte de la teoría matemática de los canales, se debe al trabajo de Shannon

Códigos en bloque

Tienen la propiedad de que todas sus palabras son de la misma longitud, n, llamada longitud del código. Sea $\mathcal{B}$ un alfabeto código, un código en bloque será un subconjunto de $\mathcal{B}^n$. Para la representación de elementos se emplea notación vectorial $\vec{x} = (b_1, \ldots, b_n), b_i \in \mathcal{B}$.

Un buen código busca maximizar la cantidad de errores que es capaz de corregir minimizando la cantidad de símbolos redundantes.

Veamos ahora unos conceptos que introducen un método de decodificación y que son de una gran relevancia a lo largo del escrito.

Definición 3.2.4 (Distancia de Hamming)**.** Si $\vec{x}, \vec{y} \in \mathcal{B}^n$, $\vec{x} = (x_1, \ldots, x_n)$, $\vec{y} = (y_1, \ldots, y_n)$, llamaremos distancia de Hamming entre $\vec{x}$ e $\vec{y}$ a

$$d(\vec{x}, \vec{y}) = \#\{i \ / \ 1 \leq i \leq n, x_i \neq y_i\}$$

Es la forma de medir la separación entre palabras. Esto quiere decir que es una distancia. Para comprobarlo, se debe tomar la aplicación d y ver que, efectivamente, es una distancia en $\mathcal{B}^n$. Basta verificar las propiedades de distancia

I d es no negativa y $d(\vec{x}, \vec{y}) = 0$ si y solo si $\vec{x} = \vec{y}$.

II $d(\vec{X}, \vec{y}) = d(\vec{y}, \vec{x})$.

III $d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) \geq d(\vec{x}, \vec{z})$.

para todo $\vec{x}, \vec{y}, \vec{z} \in \mathcal{B}^n$.

Definición 3.2.5 (Distancia mínima)**.** Llamamos distancia mínima del código $\mathcal{C}$ a

$$d = d(\mathcal{C}) = \text{mín}\{d(\vec{x}, \vec{y}) \ / \ \vec{x}, \vec{y} \in \mathcal{C}, \vec{x} \neq \vec{y}\}.$$

Para ciertos propositos es más adecuada la llamada distancia mínima relativa, que tiene en cuenta la longitud n de $\mathcal{C}$ y esta definida como

$$\delta(\mathcal{C}) = \frac{d(\mathcal{C})}{n}.$$

Nota: $1 \leq d(\mathcal{C}) \leq n$ y $0 < \delta(\mathcal{C}) \leq 1$

Algoritmo de descodificación por mínima distancia

Emitida una palabra $c \in \mathcal{C}$, recibimos una n-tupla y. Para decodificar y usamos el principio de la mínima distancia, esto significa que descodificamos y por la palabra de $\mathcal{C}$ más "parecida" a y.

Algoritmo 1: Decodificación por mínima distancia

1 Recibido $\vec{y}$;
2 Evaluar la distancia de $\vec{y}$ a todas las palabras de $\mathcal{C}$;
3 Descodificar $\vec{y}$ por la más cercana (si es única);

Proposición 3.2.4. Sea $\mathcal{C}$ un código bloque d elongitud n y distancia mínima d. El algoritmo de decodificación por mínima distancia aplicado a una n-tupla recibida,

1. Permite detectar t errores siempre que $t < d$.
2. Permite corregir cualquier configuración de t errores siempre que $2t < d$.
3. Permite corregir cualquier configuración de s borrones siempre que $s < d$.
4. Permite corregir cualquier configuracion de t errores y s borrones si $2t + s < d$.

Definición 3.2.6 (Correción de $\left\lfloor \frac{d-1}{2} \right\rfloor$ errores)**.** Si d es la distancia mínima de $\mathcal{C}$, diremos entonces que $\mathcal{C}$ corrige $\left\lfloor \frac{d-1}{2} \right\rfloor$ errores, o que $\mathcal{C}$ es un código $\left\lfloor \frac{d-1}{2} \right\rfloor$-corrector.

La importancia de esta última definición, la veremos a la hora de acotar los parámetros de un código. Nos permitirá, en función de cuantos errores queramos corregir, tomar unos u otros.

3.3. Códigos Lineales

3.3.1. Definición y estructura

Sea $\mathbb{F}_q$ el cuerpo finito con q elementos. Para ampliar la información necesaria sobre cuerpos finitos, acudir al Apéndice A.

Definición 3.3.1 (Código lineal)**.** Un código lineal de longitud n sobre $\mathbb{F}_q$ es un subespacio vectorial de $\mathbb{F}_q^n$.

$\mathcal{C}$ posee dimensión k por ser espacio vetorial. Su cardinal siempre será una potencia de q, q^k. Los parámetros n, k y la distancia mínima d, son llamados parámetros fundamentales de $\mathcal{C}$. Con los parámetros n, k y d, se dice del código que es del tipo $[n, k, d]$ o $[n, k]$. La transmisión de información $\mathcal{R}(\mathcal{C}) = \frac{k}{n}$. La redundancia del código $[n, k]$ es $r = n - k$.

Se puede interpretar todo subespacio de $\mathbb{F}_q^n$ de dimensión k como imagen de una aplicación lineal inyectiva $f : \mathbb{F}_q^k \to \mathbb{F}_q^n$. Esta aplicación no es única. Se puede entender que esta f es la aplicación de codificación y que $\mathbb{F}_q^k$ es la fuente de información que estamos codificando con $\mathcal{C}$.

Definición 3.3.2 (Matriz generatriz)**.** La matriz generatriz de $\mathcal{C}$ es la matriz de una aplicación lineal inyectiva $f : \mathbb{F}_q^k \to \mathcal{C} \subset \mathbb{F}_q^n$. Es una matriz $k \times n$ cuyas filas son una base de $\mathcal{C}$.

Una matriz generatriz G proporciona el código y la codificación. Como $\mathcal{C} = \{\vec{a}\mathrm{G} \mid \vec{a} \in \mathbb{F}_q^k\}$ un mensaje $\vec{a} \in \mathbb{F}_q^k$ se codifica por $\vec{a}\mathrm{G} \in \mathbb{F}_q^n$. Así es la codificación de los códigos lineales, únicamente requiere del almacenamiento en memoria de la matriz G. Cabe destacar que, como una base de $\mathcal{C}$ no es única, tampoco lo es una matriz generatriz. Por lo tanto, distintas matrices generatrices de un código proporcionan distintas aplicaciones f y, por tanto, distintas codificaciones.

Codificación sistemática

Cuando se codifica una palabra $\vec{a} \in \mathbb{F}_q^k$ mediante un código lineal, es interesante que la palabra codificada tenga como subpalabra a $\vec{a}$, es decir, $(\vec{a}, \vec{z})$, $\vec{z} \in \mathbb{F}_q^{n-k}$. Así los primeros k símbolos contienen la información de la palabra y los siguientes son los de control.

Este tipo de codificación es llamado sistemático y se da si y solo si G es de la forma $\mathrm{G} = (\mathrm{I}_k, \mathcal{C})$[I]. Esta es la forma estandar. El código es llamado sistemático si posee alguna matriz generatriz en forma estándar.

[I] I_k denota la matriz identidad $k \times k$

Definición 3.3.3 (Códigos equivalentes)**.** Dos códigos $\mathcal{C}_1, \mathcal{C}_2$, de igual longitud, n, sobre $\mathbb{F}_q$, son equivalentes si existe una permutación σ del conjunto $\{1, \ldots, n\}$ tal que $\mathcal{C}_2 = \{\sigma(\vec{c}) \mid \vec{c} \in \mathcal{C}_1\}$.

Cabe destacar que una permutación σ actúa sobre los índices $\{1, \ldots, n\}$ y no sobre los elementos de $\mathbb{F}_q^n$. Se escribe $\sigma(\vec{x})$ por abuso de notación para representar $(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$.

Dos códigos son equivalentes si reordenando las columnas de una matriz generatriz del primero se obtiene la matriz generatriz del segundo. Dos códigos equivalentes tienen, por tanto, los mismos parámetros k y d. Dado el código $\mathcal{C}$, para cada permutación σ de $\{1, \ldots, n\}$, el conjunto

$$\sigma(\mathcal{C}) = \{\sigma(\vec{c}) \mid \vec{c} \in \mathcal{C}\}$$

es un código equivalente a $\mathcal{C}$.

Proposición 3.3.1. Todo código es equivalente a uno sistemático

Demostración. Supongamos que k es la dimensión de $\mathcal{C}$, entonces una matriz generatriz G de $\mathcal{C}$ es $k \times n$ de rango k, luego posee k columnas independientes. A través de la una permutación σ de los índices podemos conseguir que estas columnas sean las k primeras. Con esto se obtiene una matriz $\mathrm{G}' = (A, B)$ con A regular, de tamaño $k \times k$. Podemos aplicar operaciones elementales para transformar A en la matriz I_k. Si esto se aplica en la matriz G', obtenemos una matriz en forma estándar que genera el mismo espacio que las de G' y que es, por tanto, un código sistemático equivalente al que tiene a G por matriz generatriz. □

Matriz de control

A parte del sistema de generadores, mediante unas ecuaciones implicitas podemos describir también un subespacio vectorial de $\mathbb{F}_q^n$. Esta forma de caracterización origina la siguiente definición

Definición 3.3.4 (Matriz de control)**.** Diremos que una matriz H es una matriz de control de código $\mathcal{C}$ si para todo vector $\vec{x} \in \mathbb{F}_q^n$ se verifica que $\vec{x} \in \mathcal{C}$ si y solo si $\mathrm{H} \cdot \vec{x}^t = 0$.

Si $\mathcal{C}$ está definido sobre $\mathbb{F}_q$ y es de tipo $[n, k]$, entonces también H está definida sobre $\mathbb{F}_q$, es de tamaño $(n-k) \times n$ y rango $n-k$.

Proposición 3.3.2. Si G y H son matrices generatrices y de control de $\mathcal{C}$, entonces $\mathrm{GH}^t = 0$.

Demostración. Si G es una matriz generatriz de $\mathcal{C}$ dada en la forma estándar, $\mathrm{G} = (\mathrm{I}_k, C)$, la matriz $\mathrm{H} = (-C^t, \mathrm{I}_{n-k})$ tiene tamaño $(n-k) \times n$, rango $n-k$ y verifica $\mathrm{GH}^t = 0$, por tanto, es una matriz de control para $\mathcal{C}$. □

Definición 3.3.5 (Soporte, peso de Hamming y peso mínimo)**.** Sea $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$. Llamaremos soporte de $\vec{x}$ al conjunto

$$supp(\vec{x}) = \{i \mid 1 \leq i \leq n,\ x_i \neq 0\}.$$

Llamaremos peso de Hamming de $\vec{x}$ a

$$\omega(\vec{x}) = \#supp(\vec{x}) = d(\vec{x}, \vec{0}).$$

Definimos peso mínimo de un código como:

$$\omega(\mathcal{C}) = \text{mín}\{\omega(\vec{c}) \mid \vec{c} \in \mathcal{C},\ \vec{c} \neq \vec{0}\}$$

Lema 3.3.3. En un código lineal, la distancia mínima es igual al peso mínimo.

Demostración. Como un código lineal es un espacio vectorial, el resulado se sigue de la igualdad $d(\vec{x}, \vec{y}) = \omega(\vec{x} - \vec{y})$. □

Proposición 3.3.4. Sea $\mathcal{C}$ un código lineal de matriz de contro H y distancia mínima d. Entonces $d > r$ si y solo si cualesquiera r columnas de H son linealmente independientes.

Demostración. Si existen r columnas linealmente dependientes en H, entonces $\mathrm{H}\vec{x}^t = 0$ para un $\vec{x}$ (cuyas coordenadas son los coeficientes de la combinación lineal) que está en $\mathcal{C}$ y tiene peso $\leq r$; así $d \leq r$. Recíprocamente, si cualesquiera r columnas de H son independientes, entonces ningún vector de peso $\leq r$ puede pertenecer a $\mathcal{C}$, con lo que su distancia mínima es mayor que r. □

Corolario 3.3.5. La distancia mínima de un código de matriz de control H coincide con el menor cardinal de un conjunto de columnas linealmente dependientes en H.

3.3.2. Dualidad

Definición 3.3.6 (Código dual)**.** Sea $\mathcal{C}$ un código lineal y H una matriz de control de $\mathcal{C}$. Como tiene rango máximo, a su vez H puede ser interpretada como matriz generatriz de otro código sobre $\mathbb{F}_q$. Tal código es llamado código dual de $\mathcal{C}$ y se le denota por $\mathcal{C}^\perp$.

- Si $\mathcal{C}$ tiene dimensión k, entonces $\mathcal{C}^\perp$ tiene dimensión $n - k$.
- Si G es matriz generatriz de $\mathcal{C}$, como $\mathrm{G} \cdot \mathrm{H}^t = 0 \to \mathrm{H} \cdot \mathrm{G}^t = 0$ y esto implica que G es la matriz de control de $\mathcal{C}^\perp$.

Proposición 3.3.6. Si $\mathcal{C}$ es un código lineal, entonces su dual $\mathcal{C}^{\perp}$ es el ortogonal de $\mathcal{C}$. Además $\left(\mathcal{C}^{\perp}\right)^{\perp} = \mathcal{C}$.

Demostración. Sean G y H matrices generatriz y de control de $\mathcal{C}$. El resultado enunciado es una consecuencia inmediata de la igualdad $\mathrm{GH}^t = 0$, ya que $rango(\mathrm{G}) + rango(\mathrm{H}) = n$. □

Definición 3.3.7 (Código autodual)**.** Diremos que un código lineal es autodual cuando coincide con su código dual, es decir, cuando $\mathcal{C} = \mathcal{C}^{\perp}$.

Polinómios de pesos

En un código $\mathcal{C}$, mucha de la información que contiene la proporciona el conocimiento de los pesos de todas las palabras de $\mathcal{C}$. Para $1 \leq i \leq n$, tomamos los enteros

$$a_i = a_i(\mathcal{C}) = \#\{\vec{c} \in \mathcal{C} \mid \omega(\vec{c}) = i\}.$$

Destacamos que $a_0 = 1$, $a_i = 0$ para $0 < i < d$ y $\sum_{i=0}^{n} a_i = q^k$.

Definición 3.3.8 (Polinomio de pesos)**.** Se define el polinomio de pesos de $\mathcal{C}$, como

$$\mathrm{W}(\mathcal{X}) = \sum_{i=0}^{n} a_i \mathcal{X}^i = \sum_{\vec{c} \in \mathcal{C}} \mathcal{X}^{\omega(\vec{c})}.$$

Para trabajar con polinomios de dos variables tenemos:

$$\mathrm{W}(\mathcal{X}, \mathcal{Y}) = \sum_{i=0}^{n} a_i \mathcal{X}^i \mathcal{Y}^{n-i}.$$

El polinomio de pesos de código determina univocamente el polinomio de pesos de su dual a través de la identidad de McWilliams. Esta fórmula relaciona un código con su dual.

Teorema 3.3.7 (Identidad de McWilliams)**.** Sean $\mathcal{C}$ un código lineal $[n, k]$ sobre $\mathbb{F}_q$ y $\mathcal{C}^{\perp}$ su dual. Si $\mathrm{W}(\mathcal{X}, \mathcal{Y})$ y $\mathrm{W}^{\perp}(\mathcal{X}, \mathcal{Y})$ son los polinomios de pesos de $\mathcal{C}$ y $\mathcal{C}^{\perp}$ respectivamente, entonces

$$\mathrm{W}^{\perp}(\mathcal{X}, \mathcal{Y}) = q^{-k}\mathrm{W}(\mathcal{Y} - \mathcal{X}, \mathcal{Y} + (q-1)\mathcal{X}.$$

La información al respecto de este teorema, así como de mucha información sobre codificación, se puede encontrar en [3]. Para ver las equivalentes identidades, se recomienda ver [4].

3.3.3. Decodificación de códigos lineales

Sea $\mathcal{C}$ un código lineal $[n, k, d]$ sobre $\mathbb{F}_q$. Como sabemos $\mathcal{C}$ corrige $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ errores. Ahora supongamos que se envia una palabra $\vec{c} \in \mathcal{C}$. Se recibe un vector $\vec{y} \in \mathbb{F}_q^n$. El error de la transmisión es $\vec{e} = \vec{y} - \vec{c}$. Para decodificar $\vec{y}$ calculamos la distancia de $\vec{y}$ a todas las palabras de $\mathcal{C}$ y la decodificamos por la más próxima. Si durante la trasmisión se cometen, como máximo, t errores, entonces $d(\vec{c}, \vec{y}) = \omega(\vec{e}) \leq \vec{y}$ y $\vec{c}$ es la única palabra del código con esa propiedad; la decodificación es correcta. Si $t < \omega(\vec{e}) < d$, podemos detectar que se producen errores pero no corregirlos. Si $\omega(\vec{e}) \geq d$ la decodificación fallará.

Todo lo realizado hasta este momento, dota de una estructura algebraica a los códigos, esto mejora la eficiencia computacional y, además, simplifica el proceso de decodificación.

Concepto de síndrome

Supongamos que hemos enviado $\vec{c}$ y recibido $\vec{y} = \vec{c} + \vec{e}$ donde $\vec{e}$ es el error cometido en el envío.

Definición 3.3.9 (Síndrome)**.** Llamaremos síndrome de $\vec{y}$ al vector

$$s(\vec{y}) = \mathrm{H}\vec{y}^t \in \mathbb{F}_q^{n-k}.$$

Se debe tener en cuenta que $\vec{y} \in \mathcal{C}$ si y solo si $s(\vec{y}) = \vec{0}$. Así, por ser el síndrome una aplicación lineal, $s(\vec{y}) = s(\vec{c} + \vec{e}) = \cancelto{\vec{0}}{s(\vec{c})} + s(\vec{e}) = s(\vec{e})$. Una vez recibido $\vec{y}$ conocemos inmediatamente el error cometido.

Proposición 3.3.8. El síndrome del vector recibido $\vec{y}$ es una combinación lineal de las columnas de H correspondiente a las posiciones en las que han ocurrido errores.

Líder de una clase

Se considera en $\mathbb{F}_q^n$ la relación de equivalencia $\vec{u} \sim \vec{v}$. Esta relación se tiene si y solo si $\vec{u} - \vec{v} \in \mathcal{C}$. El espacio vectorial cociente obtenido con módulo de dicha relación, lo denotamos por $\mathbb{F}_q^n/\mathcal{C}$. Los elementos de $\mathbb{F}_q^n/\mathcal{C}$ son clases de equivalencia $\vec{u} + \mathcal{C} = \{\vec{u} + \vec{x} \ / \ \vec{x} \in \mathcal{C}\}$. Cada clase posee $\#\mathcal{C} = q^k$ elementos. El cardinal de $\mathbb{F}_q^n/\mathcal{C}$ es q^{n-k}, su dimensión es $n - k$.

Fijémonos que $\vec{u}$, $\vec{v}$ están en la misma clase si y solo si $\vec{u} - \vec{v} \in \mathcal{C}$, o equivalentemente, si y solo si $s(\vec{u}) = s(\vec{v})$. Recibido $\vec{y}$, como conocemos $s(\vec{y})$, conocemos la clase a la que pertenece el error cometido.

Definición 3.3.10 (Líder de una clase)**.** Si en una clase existe un único elemento de peso mínimo éste recibe el nombre de líder de la clase.

Proposición 3.3.9. Cada clase de $\mathbb{F}_q^n/\mathcal{C}$ posee a lo sumo un elemento de peso $\leq t$.

Demostración. Si existen $\vec{u}$, $\vec{v}$ en la misma clase, ambos de peso $\leq t$ entonces $\vec{u} - \vec{v} \in \mathcal{C}$ y $\omega(\vec{u} - \vec{v}) \leq \omega(\vec{u}) + \omega(\vec{v}) \leq 2t < d(\mathcal{C})$, lo cual implica que $\vec{u} - \vec{v} = \vec{0}$ y $\vec{u} = \vec{v}$. □

Esto nos introduce un algoritmo de decodificación por síndrome-lider.

Algoritmo de decodificación

Previamente se construye el diccionario síndrome-líder

Algoritmo 2: Decodificación por síndrome

1 Recibido $\vec{y}$;
2 Calculamos $s(\vec{y})$ y buscamos en la colección de síndromes;
3 Si no posee líder, falla. Fin;
4 Si posee líder, este es $\vec{e}$ y la palabra es $\vec{y} - \vec{e}$. Fin;

3.4. Cotas en los Parámetros de un Código

En esta sección estudiaremos que parámetros k, d para cierta n, son buenos para los códigos.

3.4.1. Cotas principales

Cota de Singleton

Teorema 3.4.1 (Cota de Singleton)**.** Si $\mathcal{C}$ es un código de tipo $[n, k, d]$ sobre $\mathbb{F}_q$, entonces $k + d \leq n + 1$.

Demostración. Sea H una matriz de control de $\mathcal{C}$. Por el corolario 3.3.5, la distancia mínima de $\mathcal{C}$ equivale al número mínimo de columnas linealmente dependientes de H. Puesto que el rango de H es $n - k$, este número es, como mucho, $n - k + 1$, luego $d \leq n - k + 1$ □

Los códigos dode se alcanza la igualdad $k + d = n + 1$, son llamados de máxima distancia de separación (MDS).

Cota de Plotkin

Teorema 3.4.2 (Cota de Plotkin)**.** Sea $\mathcal{C}$ un código de tipo $[n, k, d]$ sobre $\mathbb{F}_q$. Entonces

$$d \leq \frac{nq^{k-1}(q-1)}{q^k - 1}.$$

Demostración. Dados dos subespacios vectoriales, V_1, V_2 de $\mathbb{F}_q^n$. Por la fórmula de las dimensiones, se verifica

$$dim(V_1) + dim(V_2) = dim(V_1 + V_2) + dim(V_1 \cap V_2).$$

Ahora tomemos $V_1 = \mathcal{C}$ y V_2 como el hiperplano de ecuación $x_i = 0$. Se puede ver que el número de palabras código con i-ésima coordenada nula es, q^k o q^{k-1}. Procedemos ahora a sumar todos los pesos de las palabras del código. Obtenemos

$$\sum_{\vec{c} \in \mathcal{C}} \omega(\vec{c}) \leq n(q^k - q^{k-1}) = nq^{k-1}(q-1).$$

Para acabar, cada palabra del código (excepto la nula $\vec{0}$), tiene como mínimo d coordenadas no nulas. Por tanto

$$\sum_{\vec{c}\in\mathcal{C}} \omega(\vec{c}) \geq d(q^k - 1).$$

De la combinación de las desigualdades se concluye el resultado. □

Si todas las palabras tienen el mismo peso, d, el código se denomina equidistantes.

Esta cota concluye que la distancia mínima de un código es menor o igual al peso promedio de todas sus palabras no nulas.

Cota de Hamming

Sea $r = 1, \ldots, n$. Existen exactamente $\binom{n}{r}(q-1)^r$ vectores en $\mathbb{F}_q^n$ de peso r. Por lo que, una bola de radio $t \in \mathbb{N}$ centrada en $\vec{0}$ contiene

$$V_q(n,t) = 1 + \binom{n}{1}(q-1) + \cdots + \binom{n}{t}(q-1)^t$$

elementos de $\mathbb{F}_q^n$. La distancia de Hamming invariante por traslación, por tanto, una bola del mismo radio centrada en cualquier vector de $\mathbb{F}_q^n$ posee igual número de elementos, $V_q(n,t)$.

Teorema 3.4.3. Si $\mathcal{C}$ es un código de longitud n sobre $\mathbb{F}_q$ que corrige t errores, entonces

$$mV_q(n,t) \leq q^n$$

siendo m el número de palabras de $\mathcal{C}$.

Demostración. Como $\mathcal{C}$ corrige t errores, las bolas de radio t centradas en las palabras del código son disjuntas, luego la suma de los cardinales de todas estas bolas es menor que el número de elementos del espacio $\mathbb{F}_q^n$, que es q^n. □

Los códigos $\mathcal{C}$ para los que se alcanza la igualdad son llamados códigos perfectos.

Cota de Gilbert-Varshamov

Teorema 3.4.4 (Cota de Gilbert-Varshamov)**.** Si se verifica que

$$q^{n-k+1} > V_q(n, d-1)$$

entonces existe un código lineal de tipo $[n,k]$ sobre $\mathbb{F}_q$ con distancia $\geq d$.

Demostración. Construimos una base de un código con los parámetros anunciados. Como primera palabra tomemos cualquier vector $\vec{c}_1$ de peso $\geq d$. Seleccionados $\vec{c}_1, \ldots, \vec{c}_{j-1}$, independientes y generadores de un código $\mathcal{C}_{j-1}$ de parámetros $[n, j-1, \geq d]$, si $j-1 < k$, entonces, en virtud de la hipótesis del enunciado

$$q^{j-1}V_q(n, d-1) < q^n$$

luego existe un vector $\vec{c}_j \in \mathbb{F}_q^n$ a distancia al menos d de todas las palabras de $\mathcal{C}_{j-1}$. Entonces $\mathcal{C}_j = \langle \vec{c}_1, \ldots, \vec{c}_{j-1} \rangle$ es un código $[n, j]$ y su distancia mínima es $\geq d$ ya que, si $\vec{x} \in \mathcal{C}_j \setminus \mathcal{C}_{j-1}$, será $\vec{x} = \lambda\vec{c}_j + \vec{y}$ para ciertos $\lambda \in \mathbb{F}_q^*$, $\vec{y} \in \mathcal{C}_{j-1}$ y

$$\omega(\vec{x}) = \omega(\lambda^{-1}\vec{x}) = \omega(\vec{c}_j + \lambda^{-1}\vec{y}) = d(\vec{c}_j, -\lambda^{-1}\vec{y}) \geq d.$$

□

3.4.2. Códigos de Máxima Distancia de Separación (MDS)

Un código $\mathcal{C}$ del tipo $[n, k, d]$ es MDS si $d = n-k+1$. Equivalentemente, cada $n-k$ columnas de una matriz de control de $\mathcal{C}$ son linealmente independientes.

Proposición 3.4.5. Si $\mathcal{C}$ es MDS entonces también $\mathcal{C}^\perp$ es MDS.

Demostración. Si $\mathcal{C}^\perp$ no fuera MDS, debería existir una palabra $\vec{x} \in \mathcal{C}^\perp$ con peso $0 < \omega(\vec{x}) \leq k$. Si ampliamos $\vec{x}$ a una base de $\mathcal{C}^\perp$ y construimos una matriz H generatriz de $\mathcal{C}^\perp$ a partir de dicha base. Esta será una matriz de control para $\mathcal{C}$. Vamos a tomar ahora las $n-k$ columnas de H con cero en la coordenada correspondiente de $\vec{x}$. Por ser $\mathcal{C}$ MDS, estas columnas son linealmente independientes. Esto es imposible dado que son $n-k$ elementos de $\mathbb{F}_q^{n-k}$ con la primera coordenada nula. □

Sea cualquier entero n, existirán siempre códigos MDS de longitud n y dimensiones 1, $n-1$ y n; se denominan códigos MDS triviales. La importancia de los códigos MDS es que su polinomio de pesos está completamente determinado por sus parámetros.

Teorema 3.4.6. Sea $\mathcal{C}$ un código $[n, k, d]$ MDS sobre $\mathbb{F}_q$. El número de palabras de peso $n-k+r$ en $\mathcal{C}$ es

$$a_{n-k+r} = \binom{n}{k-r} \sum_{i=o}^{r-1} (-1)^i \binom{n-k+r}{i} (q^{r-i} - 1).$$

Se demostrará para $r = 1$ debido a los pesados cálculos que se requieren.

Corolario 3.4.7. Sea $\mathcal{C}$ un código $[n, k, d]$ MDS sobre $\mathbb{F}_q$. Entonces

$$a_{n-k+1} = \binom{n}{k-1}(q-1) = \binom{n}{n-k+1}(q-1).$$

Demostración. Si $\vec{x}, \vec{y} \in \mathcal{C}$ son palabras con el mismo soporte de cardinal $n-k+1$, existe $\lambda \in \mathbb{F}_q^*$ tal que $\lambda\vec{x}$ e $\vec{y}$ tienen su primera coordenada no nula igual. Entonces $\omega(\lambda\vec{x}+\vec{y}) < n-k+1 = d(\mathcal{C})$, con lo que $\vec{y} = \lambda\vec{x}$. Como consecuencia, para cada $n-k+1$ posiciones, existen exáctamente $q-1$ palabras que tienen como soporte este conjunto de posiciones. □

3.5. Códigos Cíclicos

Antes de comenzar esta parte, se recomienda la lectura del Anexo B.

3.5.1. Definición y propiedades

Definición 3.5.1 (Códigos cíclicos)**.** Diremos que un código lineal $\mathcal{C}$ de longitud n sobre $\mathbb{F}_q$, es cíclico si verifica la propiedad siguiente: si $(c_0, c_1, \ldots, c_{n-1}) \in \mathcal{C}$, entonces $(c_{n-1}, c_0, \ldots, c_{n-2}) \in \mathcal{C}$.

Sean $\mathbb{F}_q[X]^{(n-1)}$ el espacio vectorial de todos los polinomios sobre $\mathbb{F}_q$ con grado menor que n. Sea A el anillo cociente $A = \mathbb{F}_q[X]/\langle X^n - 1\rangle$. Por los isomorfismos de espacios vectoriales

$$\mathbb{F}_q^n \cong \mathbb{F}_{q,n-1} \cong A$$

se puede identificar cada vector $(a_0, \ldots, a_{n-1})$ con el polinomio $a_0 + a_1X + \ldots + a_{n-1}X^{n-1}$ y con la clase, $a_0 + a_1X + \ldots + a_{n-1}X^{n-1} + \langle X^n - 1\rangle$. Un código lineal sobre $\mathbb{F}_q$ se puede considerar como un subconjunto de A.

Se añade ahora una restricción a la longitud de los códigos permitidos: $mcd(q, n) = 1$. Así se garantiza que el polinomio $X^n - 1$ tiene todos sus factores irreducibles distintos y sus raíces forman un grupo cíclico de orden n.

Teorema 3.5.1. Sea $\mathcal{C}$ un código lineal no nulo de longitud n sobre el cuerpo finito $\mathbb{F}_q$. $\mathcal{C}$ es cíclico si y solo si, considerando inmerso en A, es un ideal.

Para profundizar en la teoría de ideales, se recomienda acudir a [5], apartado 1.4.

Demostración. Sea $\mathcal{C}$ cíclico. Como $\mathcal{C}$ es un subgrupo abeliano de A, basta con probar que si $a(X) \in A$ y $c(X) \in \mathcal{C}$, entonces $a(X)c(X) \in \mathcal{C}$, o lo que es lo mismo, que $Xc(X) \in \mathcal{C}$. Tenemos entonces que

$$X(c_0 + c_1X + \cdots + c_{n-1}X^{n-1}) = c_{n-1} + c_0X + \cdots + C_{n-2}X^{n-1}$$

y el hecho de que este último polinomio pertenezca al código es la definición de código cíclico interpretada en lenguaje polinómico. El recíproco se demuestra de manera similar. □

Corolario 3.5.2. Dado un código cíclico no nulo $\mathcal{C}$ de longitud n, existe un único polinomio mónico $g(X) \in \mathbb{F}_q[X]$ divisor de $X^n - 1$, tal que $\mathcal{C} = \langle g(X)\rangle$. En consecuencia, los elementos de $\mathcal{C}$ pueden identificarse con los polinomios de grado menor que n múltiplos de $g(X)$.

Sea m el número de factores irreductibles de $X^n - 1$ sobre el cuerpo $\mathbb{F}_q$. Por el corolario anterior se deduce que el número de códigos cíclicos de longitud n es 2^n.

También se debe tener en cuenta que el polinomio $g(X)$, no es único generador posible del código cíclico $\mathcal{C}$.

3.5.2. Matriz generatriz y de control

Proposición 3.5.3. Si $\mathcal{C}$ es un código cíclico de longitud n sobre $\mathbb{F}_q$, con polinomio generador $g(X)$ de grado $n-k$, entonces una base de $\mathcal{C}$ es $\{g(X), Xg(X), \ldots, X^{k-1}g(X)\}$. En particular, $\mathcal{C}$ tiene dimensión k.

Demostración. Se debe probar que $\{g(X), Xg(X), \ldots, X^{k-1}g(X)\}$ es un sistema de generadores. Sea $f(X)g(X) \in \mathcal{C}$. Por el corolario 3.5.2 suponemos que $deg(f(X)) < k$. Si $f(X) = a_0 + a_1X + \cdots + a_{k-1}X^{k-1}$, nos queda

$$f(X)g(X) = a_0g(X) + a_1Xg(X) + \cdots + a_{k-1}X^{k-1}g(X)$$

una combinación lineal, por lo que $\{g(X), Xg(X), \ldots, X^{k-1}g(X)\}$ es un sistema generador. Solo falta ver que es un sistema linealmente independiente. Una relación de dependencia lineal

$$b_0g(X) + b_1Xg(X) + \cdots + b_{k-1}X^{k-1}g(X) = 0$$

puede escribirse como $b(X)g(X) = 0$, siendo $b(X) = b_0 + b_1X + \cdots + b_{k-1}X^{k-1}$. Como $deg(b(X)g(X)) < n$ y $g(X) \neq 0$, se tiene que $b(X) = 0$, luego $b_i = 0$ para $i = 0, 1, \ldots, k-1$ y el conjunto es linealmente independiente. □

Corolario 3.5.4. Sea $\mathcal{C}$ un código cíclico de longitud n y polinomio generador $g(X) = g_0 + g_1X + \cdots + g_{n-k}X^{n-k}$. La matriz

$$\mathrm{G} = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & \cdots & \cdots & g_{n-k} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & \ldots & \ldots & 0 & g_0 & g_1 & \ldots & \ldots & g_{n-k} \end{pmatrix}$$

es una matriz generatriz de $\mathcal{C}$.

Así, nos queda una matriz generatriz del código $\mathcal{C}$ a partir de un polinomio generador. Con esta misma idea, podemos construir un polinomio de control que nos permita formar una matriz de control.

Definición 3.5.2 (Polinomio de control)**.** Si $\mathcal{C}$ es un código cíclico de longitud n sobre $\mathbb{F}_q$, con un polinomio generador $g(X)$ de grado $n-k$, llamaremos polinomio de control de $\mathcal{C}$ al polinomio

$$h(X) = \frac{X^n - 1}{g(X)} = h_0 + h_1 X + \ldots + h_k X^k.$$

Proposición 3.5.5. Con las notaciones de la definición anterior, la matriz (de tamaño $(n-k)\times n$

$$H = \begin{pmatrix} 0 & 0 & \ldots & \ldots & 0 & h_k & h_{k-1} & \ldots & \ldots & h_1 & h_0 \\ 0 & 0 & \ldots & \ldots & h_k & h_{k-1} & \ldots & \ldots & h_1 & h_0 & 0 \\ 0 & \ldots & 0 & h_k & h_{k-1} & \ldots & \ldots & h_1 & h_0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_k & h_{k-1} & \ldots & \ldots & h_1 & h_0 & 0 & 0 & \ldots & \ldots & 0 \end{pmatrix}$$

es una matriz de control de $\mathcal{C}$.

Demostración. Se debe probar la identidad $\mathrm{GH}^t = 0$. Tenemos que, para todo $1 \leq i \leq k$, $1 \leq j \leq n-k$, el elemento (i,j) del producto $\mathrm{GH}^t = 0$, es el coeficiente de $X^{n-i-j+1}$ en el polinomio $g(X)h(X) = X^n - 1$. □

El polinomio de control $h(X)$, es un generador del código dual de $\mathcal{C}$. Esto demuestra que el dual de un código cíclico es también cíclico.

3.5.3. Ceros de un código cíclico

Sean $X^n - 1 = f_1(x)f2(x)\ldots f_m(x)$ la descomposición de $X^n - 1$ en factores irreducibles y sea α_i una raíz de $f_i(x)$. Para el código cíclico $\mathcal{C}_i$ engendrado por $f_i(x)$, se tiene

$$\mathcal{C}_i = \langle f_i(x) \rangle = \{c(x) \in A \ / \ c(\alpha_i) = 0\}.$$

En general, para el código cíclico engendrado por $g(X) = f_{i_1} f_{i_2} \ldots f_{i_r}$, se tendrá

$$\mathcal{C} = \langle g(X) \rangle = \{c(X) \ / \ c(\alpha_{i_1}) = c(\alpha_{i_2}) = \ldots = c(\alpha_{i_r}) = 0\}$$

lo que muestra que los códigos cíclicos pueden definirse, alternativamente, como conjuntos de polinomios con ciertas raíces n-ésimas de 1 como ceros. Esto permite el proceso inverso.

IDEA CLAVE: Podemos tomar un conjunto de elementos $\{\alpha_1, \ldots, \alpha_r\}$ en extensiones finitas $\mathbb{F}_{q^{t_1}}, \ldots, \mathbb{F}_{q^{t_r}}$ de $\mathbb{F}_q$ (los α_i estarán todos en la extensión finita $\mathbb{F}_{q^t}$, $t = mcm(t_1, \ldots, t_r)$) y definir

$$\mathcal{C} = \{c(x) \in A \ / \ c(\alpha_1) = c(\alpha_2) = \ldots = c(\alpha_r) = 0\}.$$

Tal código es automáticamente cíclico, pues si $f_i(x)$ es el polinomio irreducible de α_i se verifica que $\mathcal{C} = \langle g(X) \rangle = mcm(f_1, \ldots, f_r)$.

3.5.4. Descodificación Cíclica

Sigue siendo el esquema síndrome-líder, la diferencia es la construcción de tablas reducidas.

Construímos una tabla reducida de síndromes y líderes que contenga únicamente las entradas correspondientes a líderes con coordenada $n-1$ no nula.

Algoritmo de descodificación

Previamente se construye el diccionario síndrome-líder reducido.

Algoritmo 3: Decodificación por síndrome (códigos cíclicos)

```
Result: Posición de los fallos en el mensaje recibido
Recibido y⃗, con tamaño s;
i = 0;
v = [];
while i < s do
    if El síndrome de y⃗^(i) no aparece en la tabla reducida then
        La última coordenada es correcta;
        v[i].append(0);
    else
        La última coordenada es incorrecta;
        v[i].append(1);
    end
    i + 1;
end
```

Polinomio síndrome

Definición 3.5.3 (Polinomio síndrome)**.** Sea $\mathcal{C}$ un código cíclico generado por el polinomio $g(X)$. Se recibe un vector $\vec{y}$. Llamamos polinomio síndrome de $\vec{y}$ (representado por $s[\vec{y}](X)$), al resto de la división euclídea de $y(X)$ entre $g(X)$.

Proposición 3.5.6. Sea $\mathcal{C}$ un código cíclico que corrige t errores. Si se envia la palabra $\vec{c}$ y se recibe $\vec{y} = \vec{c} + \vec{e}$ han ocurrido, como mucho, t errores y si el polinomio síndrome de $\vec{y}$, $s[\vec{y}](X)$ tiene, como máximo, peso t, entonces $e(X) = s[\vec{y}](X)$.

Demostración. Como $s[\vec{y}](X) = y(X) - g(X)q(X) = c(X) + e(X) - g(X)q(X)$, tenemos que $s[\vec{y}](X) - e(X) = c(X) - g(X)q(X) \in \mathcal{C}$. Si $s[\vec{y}](X)$ tiene, a lo sumo, peso t y han ocurrido $\leq t$ errores, entonces $s[\vec{y}](X) - e(X)$ tiene peso, como máximo, $2t < d$, luego $s[\vec{y}](X) - e(X) = 0$. □

Este método para corregir errores fuerza a que el síndrome $s[\vec{y}](X)$ tenga peso $\leq t$, que puede ser que no sea siempre así. Pero aunque con esto no suceda, puede que con alguna de sus permutaciones cíclicas sí se verifique. Como $y^{(j)}(X) = c^{(j)}(X) + e^{(j)}(X)$ y la proposición anterior nos asegura que $e^{(j)}(X) = s[\vec{y}^{(j)}](X)$, podemos recuperar el error como $e(X) = s[\vec{y}^{(j)}]^{(n-j)}(X)$.

Este método se conoce como captura del error y es especialmente adecuado para los errores a ráfagas. Este tipo de errores se estudian a continuación.

3.5.5. Errores a ráfagas

Definición 3.5.4 (Ráfagas y su longitud)**.** Una ráfaga es un vector $\vec{x} \in \mathbb{F}_q^n$ tal que todas sus coordenadas no nulas son consecutivas. Se llama longitud de la ráfaga a $\omega(\vec{x})$.

Proposición 3.5.7. Un código cíclico $\mathcal{C}$ de parámetros $[n, k]$ no contiene ninguna ráfaga de longitud $l \leq n - k$, luego detecta cualquier error ráfaga de longitud $l \leq n - k$.

Demostración. Una ráfaga de longitud l corresponde a un polinomio de la forma $X^i l(X)$ con $deg(l(X)) < l$. En particular $l(X) \notin \mathcal{C}$ puesto que $deg(l(X)) < deg(g(X))$ y $l(X)$ no puede ser múltiplo de $g(X)$. Esto implica que $X^i l(X) \notin \mathcal{C}$. Para la segunda afirmación, si $\vec{e}$ es una ráfaga de longitud $l \leq n - k$, entonces $\vec{c} + \vec{e} \notin \mathcal{C}$, luego puede detectarse el error. □

La decodificación de ráfagas se puede hacer a través de la captira de errores. Se verifica el siguiente resultado.

Proposición 3.5.8. Sea $\mathcal{C}$ un código cíclico de parámetros $[n, k]$. Si los errores de un vector recibido $\vec{y}$ constituyen una ráfaga de longitud a lo más $n-k$, entonces existe j tal que $\vec{e}^{(j)}(X) = s[\vec{y}^{(j)}](X)$.

Demostración. Con las condiciones propuestas, algún $\vec{y}^{(j)}$ tiene errores únicamente en las coordenadas $0, \ldots, n - k - 1$, con lo que $deg(e^{(j)}(X)) < n - k$. Pero, por la unicidad de la división

euclídea tenemos

$$s[\vec{y}^{(j)}](X) = s[\vec{e}^{(j)}](X) = e^{(j)}(X)$$

obteniendo el resultado buscado. □

3.6. Códigos BCH

Sabemos ahora que los códigos cíclicos se pueden determinar como un conjunto de ceros de su polinomio generador. Si tomamos como ceros los elementos $\alpha_1, \ldots, \alpha_r$, podemos tomar la matriz

$$\mathrm{H}' = \begin{pmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_r & \cdots & \alpha_r^{n-1} \end{pmatrix}$$

como una matriz de control del código obtenido $\mathcal{C}$. Donde la distancia mínima de $\mathcal{C}$ es $\geq d$ si cualesquiera $d-1$ columnas de H' son linealmente independientes. Para determinar d no basta con una elección arbitraria de los α_i. Una posible selección la constituye el caso en que los α_i son potencias consecutivas de una raíz primitiva n-ésima de la unidad, $\alpha_i = \alpha^i$, $i = 1, \ldots, r < n$. Así todo menor de la correspondiente matriz H' se reduce a un determinante de tipo Vandermonde no nulo, y $d(C) \geq r + 1$.

Nota: Una matriz con una progresión geométrica en cada fila se conoce como matriz de Vandermonde,

$$V = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \ldots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{n-1} \end{bmatrix},$$

una matriz $n \times n$. Su determinante, se conoce como determinante de Vandermonde y su resultado es

$$|V| = \prod_{1 \leq i < j \leq n} (\alpha_j - \alpha_i).$$

3.6.1. Definición

Supongamos fijados un cuerpo finito $\mathbb{F}_q$ y números naturales n, b y δ, $2 \leq \delta \leq n$. Sea m el orden multiplicativo de q módulo n ($q^m \equiv 1 (mod\ n)$) y $\alpha \in \mathbb{F}_{q^m}$ una raíz primitiva n-ésima de la unidad.

Definición 3.6.1 (Código BCH)**.** Llamaremos código BCH sobre $\mathbb{F}_q$, de longitud n y distancia mínima prevista δ, al código cíclico (sobre $\mathbb{F}_q$ y de longitud n) cuyo polinomio generador tiene por raíces $\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+\delta-2}$.

- Si se toma $b = 1$, el código se denomina BCH en sentido estricto.

- Si la longitud n es de la forma $n = q^m - 1$, se llaman códigos BCH primitivos.
- Si, además, $m = 1$, el código se denomina Reed-Solomon.

Proposición 3.6.1. Si $\mathcal{C}$ es un código BCH de distancia prevista δ, entonces su distancia mínima, d, verifica que $d \geq \delta$.

Demostración. $\mathcal{C}$ es el espacio no nulo de la matriz

$$\mathrm{H}' = \begin{pmatrix} 1 & \alpha^b & \cdots & \alpha^{(n-1)b} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{b+\delta-2} & \cdots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix}.$$

Cualquier menor $(\delta-1)\times(\delta-1)$ de esta matriz se reduce a un determinante de Vandermonde. Por lo tanto, el resultado se sigue de la caracterización de la distancia mínima en términos de una matriz de control. □

3.6.2. Descodificación de códigos BCH

Sea el código BCH sobre $\mathbb{F}_q$ de longitud n y distancia prevista $\delta = 2t + 1$ (asÃ corrige t errores). Sea α una raíz n-ésima primitiva de la unidad. La matriz de control es

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{\delta-1} & \alpha^{2(\delta-1)} & \cdots & \alpha^{(n-1)(\delta-1)} \end{pmatrix}.$$

Supongamos enviada una palabra $\vec{c} \in \mathcal{C}$ y recibido un vector $\vec{y} = \vec{c} + \vec{e}$ con $\omega(\vec{e}) = r < t$. Sean $0 \leq i_1 < \ldots < i_r \leq n-1$, las posiciones en que han ocurrido errores y $e_{i_1}, \ldots, e_{i_r}$, las coordenadas del error $\vec{e}$ en esas posiciones.

Calculamos el síndrome del vector recibido

$$s = s(y) = s(\vec{e}) = Hy^t = (s_0, \ldots, s_{\delta-2})^t.$$

En notación polinomica nos queda:

$$s(X) = s_0 + s_1 X + \ldots + s_{\delta-2} X^{\delta-2}.$$

Se observa que para cada $h = 0, \ldots, \delta - 2$

$$s_h = y(\alpha^{h+1}) = e(\alpha^{h+1}) = \sum_{j=1}^{r} e_{ij} (\alpha^{h+1})^{ij} = \sum_{j=1}^{r} e_{ij} (\alpha^{ij})^{h+1}.$$

Nota: Simplificamos la notación: $\eta_j = \alpha^{ij}$, son llamados 'localizadores de errores'; $\epsilon_j = e_{ij}$, son llamados 'valores de error', para $j = 1, \ldots, r$, por lo que $s_h = \epsilon_1 \eta_1^{h+1} + \cdots + \epsilon_r \eta_r^{h+1}$.

Si $s(X) = 0$ entonces $y \in \mathcal{C}$, el mensaje es correcto, por lo que nos centramos en $s(X) \neq 0$.

Definición 3.6.2 (Polinomio localizador de errores)**.** Llamamos polinomio localizador de errores al polinomio

$$L(X) = \prod_{i=1}^{r} (1 - \eta_i X).$$

Definición 3.6.3 (Polinomio evaluador de errores)**.** Llamaremos polinomio evaluador de errores al polinomio

$$E(X) = \sum_{j=1}^{r} \epsilon_j \prod_{i \neq j} (1 - \eta_i X).$$

Nota: $L(X)$ y $E(X)$ son polinomios de grado r y $r-1$ respectivamente, cuyo conocimiento implica el de los η_j y ϵ_j.

Proposición 3.6.2. En las condiciones anteriores,

a) si $\rho_1, \ldots, \rho_r$ son las raíces de $L(X)$, entonces sus inversos $\rho_1^{-1}, \ldots, \rho_r^{-1}$ son los localizadores del error;

b) conocidos $\eta_1, \ldots, \eta_r$, ls valores del error son

$$\epsilon_j = \frac{-\eta_j E(\eta_j^{-1})}{L'(\eta_j^{-1})}$$

siendo $L'(X)$ la derivada (formal) de $L(X)$.

Demostración. a) Esta parte es evidente.

b) Como

$$L'(X) = \sum_{h=1}^{r} (-\eta_h) \prod_{i \neq h} (1 - \eta_i X),$$

η_j^{-1} es raíz de todos los sumandos de $L'(X)$ excepto del j-ésimo. Por lo que

$$\epsilon_j L'(\eta_j^{-1}) = (-\eta_j)\epsilon_j \prod_{i \neq j} (1 - \eta_i \eta_j^{-1}) = -\eta_j E(\eta_j^{-1})$$

donde se deduce el resultado.

□

El porlinomio evaluador se puede escribir en términos de series de potencias. Para ello necesitamos la siguiente proposición:

Proposición 3.6.3. Sea $a \in \mathbb{F}_q$. Se verifica la igualdad

$$\frac{1}{1-aX} = \sum_{i=0}^{\infty} a^i X^i.$$

Demostración. Si operamos se obtiene

$$(1-aX)\sum_{i=0}^{\infty} a^i X^i = \sum_{i=0}^{\infty} a^i X^i - \sum_{i=0}^{\infty} a^{i+1} X^{i+1} = 1$$

donde se deduce el resultado. □

Según esta proposición

$$\frac{1}{1-aX} = \sum_{i=0}^{\infty} a^i X^i$$

luego podemos poner

$$\begin{aligned} E(X) &= \sum_{i=0}^{r} \epsilon_j \frac{L(X)}{1-\eta_j X} \\ &= L(X) \sum_{j=1}^{r} \epsilon_j \sum_{i=0}^{\infty} \eta_j^i X^i \\ &= L(X) \sum_{j=1}^{r} \left(\sum_{i=0}^{\infty} \epsilon_j \eta_j^i \right) X^i \\ &= L(X) \sum_{j=1}^{r} e(\alpha^i) X^i \end{aligned}$$

Esta forma de escribir $E(X)$ nos da el teorema siguiente.

Teorema 3.6.4 (Ecuación clave)**.** Los polinomios $L(X)$, $E(X)$ y $s(X)$ están relacionados mediante la ecuación

$$E(X) \equiv L(X)s(X)(mod\ X^{\delta-1}).$$

Demostración. Tras la computación anterior tenemos

$$E(X) = L(X)\sum_{i=0}^{\delta-2} e(\alpha^{i+1})X^i = L(X)\sum_{i=0}^{\delta-2} s_i X^i = L(X)s(X)(mod\ X^{\delta-1})$$

como queríamos demostrar. □

Nótese que la ecuación clave muestra que, una vez conocidos $s(X)$ y $L(X)$, el polinomio evaluador de errores, $E(X)$, se obtiene inmediatamente. Es suficiente multiplicar $L(X)$ y $s(X)$ y reducir el resultado $mod\ X^{\delta-1}$.

La descodificación se resume a encontrar $L(X)$ y $E(X)$.

Método euclídeo

Se basa en el algoritmo de Euclides en $\mathbb{F}_q[X]$. Empecemos dando algunos resultados sobre $L(X)$ y $E(X)$.

Lema 3.6.5. $mcd(L(X), E(X)) = 1$

Demostración. No existen raíces coincidentes entre $L(X)$ y $E(X)$. □

Proposición 3.6.6. Si $\widetilde{L}(X)$ y $\widetilde{E}(X)$ son dos polinomios que verifican

1) $deg(\widetilde{L}(X)) \leq t$, $deg(\widetilde{E}(X)) \leq t$;

2) $\widetilde{E}(X) \equiv s(X)\widetilde{L}(X)(mod\ X^{\delta-1})$,

entonces existe un polinomio $\lambda(X)$ tal que $\widetilde{L}(X) = \lambda(X)L(X)$ y $\widetilde{E}(X) = \lambda(X)E(X)$.

Demostración. Tomamos la congruencia 2) y la ecuación clave tenemos que $E(X)\widetilde{L} \equiv L(X)\widetilde{E}(X)(mod\ X^{\delta-1})$. Al ser $deg(E(X)\widetilde{L}(X)) < \delta - 1$ y $deg(L(X)\widetilde{E}(X)) < \delta - 1$, esta congruencia implica que $E(X)\widetilde{L}(X) = L(X)\widetilde{E}(X)$. Siendo los polinomios $L(X)$ y $E(X)$ primos entre sí, forzosamente $L(X)|\widetilde{L}(X)$. Por lo que, si ponemos $\widetilde{L}(X) = \lambda(X)L(X)$, la igualdad anterior implica que $\widetilde{E}(X) = \lambda(X)E(X)$. □

Basta entonces encontrar polinomios $\widetilde{L}(X)$ y $\widetilde{E}(X)$ que satisfagan las condiciones 1) y 2) de la proposición para determnar $L(X)$ y $E(X)$. Por lo que, salvo multiplicación de elementos de $\mathbb{F}_q$, tenemos

$$L(X) = \frac{\widetilde{L}(X)}{mcd(\widetilde{L}(X), \widetilde{E}(X))} \text{ y } E(X) = \frac{\widetilde{E}(X)}{mcd(\widetilde{L}(X), \widetilde{E}(X))}.$$

A partir de este instante, se empleará la teoría del Apéndice B, por lo que se recomienda su lectura previa.

El proceso a seguir es el siguiente: se realiza el algoritmo de Euclides con los polinomios $f_0(X) = X^{\delta-1}$ y $f_1(X) = s(X)$. Sea j el menor índice para el que $deg(f_j(X)) < t$. Pongamos

$$\widetilde{L}(X) = u_j(X);\ \widetilde{E}(X) = (-1)^{j+1} f_j(X).$$

Proposición 3.6.7. Si $r \leq t$, los polinomios $\widetilde{L}(X)$ y $\widetilde{E}(X)$ verifican las condiciones 1) y 2) de la proposición anterior.

Demostración. Por definición, $deg(\widetilde{E}(X)) < t$. Según B.4.3

$$deg(u_j(X)) = deg(f_0(X)) - deg(f_{j-1}(X))$$

luego, por ser $deg(f_0(X)) < \delta - 1$ y $deg(f_{j-1}(X)) \geq t$, se tiene que $deg(u_j(X)) \leq t$. Para la segunda parte se emplea B.4.2

$$(-1)^{j+1} f_j(X) = (-1)^{j+1}(-1)^j (v_j(X) X^{\delta-1} - u_j(X) s(X))$$

de donde

$$(-1)^{j+1} f_j(X) - u_j(X) s(X) = -v_j(X) X^{\delta-1} \equiv (mod\ X^{\delta-1})$$

quedando demostrada la segunda parte. □

Proposición 3.6.8. $mcd(\widetilde{L}(X), \widetilde{E}(X)) = 1$.

Demostración. Se sabe que $\widetilde{L}(X) = \lambda(X) L(X)$ y que $\widetilde{E}(X) = \lambda(X) E(X)$, siendo $\lambda(X) = mdc(\widetilde{L}(X), \widetilde{E}(X))$. Probemos que $\lambda(X) \in \mathbb{F}_q$. Según B.4.3,b), $mcd(u_j(X), v_j(X)) = 1$, es suficiente probar que $\lambda(X)$ divide a ambos polinomios $u_j(X)$ y $v_j(X)$ para demostrar la proposición. Esto es claro para $u_j(X)$ ya que $\lambda(X) L(X) = \widetilde{L}(X) = u_j(X)$. Veamos que ocurre con $v_j(X)$, en la proposición anterior se probó que

$$v_j(X) X^{\delta-1} = \widetilde{L}(X) s(X) - \widetilde{E}(X) = \lambda(X)(L(X) s(X) - E(X)).$$

Por otra parte, como $E(X) \equiv L(X) s(X) (mod\ X^{\delta-1})$, existe un polinomio $\mu(X)$ tal que $E(X) = L(X) s(X) + \mu(X) X^{\delta-1}$. Sustituyendo este valor en la igualdad anterior

$$v_j(X) X^{\delta-1} = -\lambda(X) \mu(X) X^{\delta-1}$$

luego $v_j(X) = -\lambda(X)\mu(X)$ y $\lambda(X) | v_j(X)$. □

Ahora se tiene que

$$L(X) = \lambda u_j(X) \ y \ E(X) = (-1)^{j+1}\lambda f_j(X).$$

Determinemos la constante $\lambda \in \mathbb{F}_q$. El auténtico polinomio localizador verifica que $L(0) = 1$,

$$1 = L(0) = \lambda u_j(0).$$

Esto induce el siguiente resultado.

Teorema 3.6.9. Si $r \leq t$, los polinomios

$$\widetilde{L}(X) = \frac{u_j(X)}{u_j(0)}; \ \widetilde{E}(X) = \frac{(-1)^{j+1}f_j(X)}{u_j(0)}$$

son los auténticos polinomios localizador y evaluador de errores.

El algoritmo queda tal que:

Algoritmo 4: Decodificación por Método Euclídeo

---

input: Se recibe un vector $\vec{y}$.

1 Se determina su sindrome $s(X) = s_1 + \cdots + s_{\delta-2}X^{\delta-2}$ con $s_i = u(\alpha^i)$;

2 Se aplica el algoritmo de Euclides modificando a los polinomios $f_0(X) = X^{\delta-1}$ y $f_1(X) = s(X)$, hasta obtener un índice j tal que $deg(f_j(X)) < t$;

3 En este punto se computan los polinomios localizador y evaluador de errores

$$L(X) = \frac{u_j(X)}{u_j(0)}, \ E(X) = \frac{(-1)^{j+1}f_j(X)}{u_j(0)};$$

4 Ahora se encuentran todas las raíces de $L(X)$ evaluando este polinomio en todos los $1, \alpha, \ldots, \alpha^{n-1}$. Si $\alpha^{h_1}, \ldots, \alpha^{h_r}$ son tales raíces, entonces sus inversos son localizadores del error, $\eta_1 = \alpha^{n-h_1}, \ldots, \eta_r = \alpha^{n-h_r}$. El valor del error asociado al localizador η_j es

$$\epsilon_j = \frac{-\alpha^{n-h_j}E(\alpha^{h_j})}{L'(\alpha^{h_r})}.$$

5 Corregir el mensaje recibido: para $i = 1, \ldots, n-1$, la i-ésima coordenada del mensaje corregido es

$$\begin{cases} y_i & \text{si} \quad i \neq n-h_1, \ldots, n-h_r; \\ y_i - \epsilon_j & \text{si} \quad i = n-h_j \end{cases}$$

IDEA CLAVE: Es evidente que el proceso de descodificación comporta únicamente operaciones aritméticas elementales en $\mathbb{F}_q$, con lo que, teniendo en cuenta el coste de tales operaciones sobre un cuerpo finito, la complejidad del proceso es polinómica en los datos.

3.6.3. Códigos Reed-Solomon

Como ampliación de información, [8] es un buen artículo donde se representan estos códigos. Así como su codigicación y decodificación.

Definición 3.6.4 (Códigos Reed-Solomon)**.** Un código Reed-Solomon sobre $\mathbb{F}_q$ es un código BCH primitivo de longitud $n = q - 1$.

Proposición 3.6.10. Los códigos Reed-Solomon sobre $\mathbb{F}_q$ son MDS (Máxima distancia de separación).

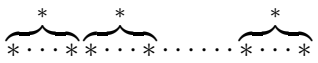
Demostración. En todo código BCH, fijada la distancia prevista δ y la raíz n-ésima α, su distancia mínima y dimensión, d y k, verifican $d \geq \delta$ y $k = n - deg(g(X))$, siendo $g(X) = mcm\{\text{Irr}(\alpha^i, \mathbb{F}_q) \mid i = 1, \ldots, n-1\}$. Puesto que $\alpha^i \in \mathbb{F}_q$ para todo i, se tendrá $deg(g(X)) = \delta - 1$ luego, teniendo en cuenta la cota de Singleton, $k = n - \delta - 1$, de donde $d = n - k + 1$. □

La característica más importante de estos códigos viene dada por la raíz n-ésima, α, que es un elemento de $\mathbb{F}_q$, así, todo el trabajo de código impican únicamente operaciones en $\mathbb{F}_q$. La problemática de estos códigos es que queda limitada su longitud a $q - 1$ sobre $\mathbb{F}_q$. No existen códigos Reed-Solomon sobre $\mathbb{F}_2$. Se plantean dos soluciones posibles ante dicho problema, la primera solución es el descenso de cuerpo y la segunda es la concatenación.

Descenso de cuerpo

Dado un cuerpo finito $\mathbb{F}_{q^r}$, extensión de $\mathbb{F}_q$, fijemos una base $\{1, \alpha, \ldots, \alpha^{r-1}\}$ de $\mathbb{F}_{q^r}$ sobre $\mathbb{F}_q$. Cada elemento de $\mathbb{F}_{q^r}$ puede identificarse con el vector de $\mathbb{F}_q^r$ de sus coordenadas de la base anterior. Aplicando este procedimiento a cada componente de un vector $\mathbb{F}_{q^r}^n$, obtendremos un vector de $\mathbb{F}_q^{rn}$.

vector original sobre $\mathbb{F}_{q^r}$



vector obtenido sobre $\mathbb{F}_q$

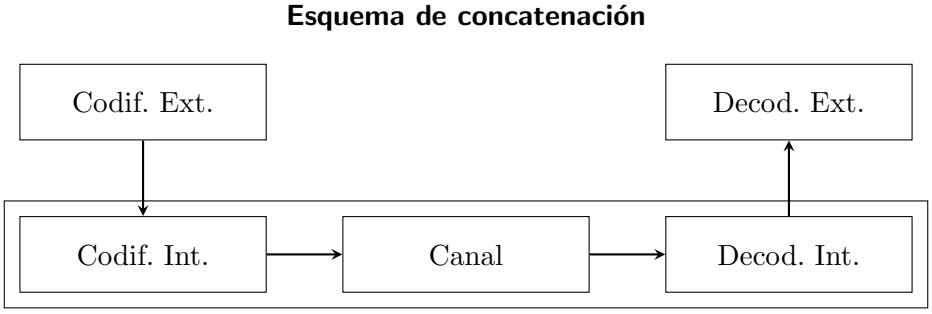
Si $\mathcal{C}$ es un código de longitud n sobre $\mathbb{F}_{q^r}$, a partir de él podemos conseguir un código sobre $\mathbb{F}_q$ de longitud rn, se dice que este cuerpo se obtiene del original por descenso de cuerpo.

Proposición 3.6.11. Sea $\mathcal{C}$ un código de Reed-Solomon sobre $\mathbb{F}_{2^r}$ con distancia $d = 2t + 1$. Entonces, el código binario obtenido por descenso de cuerpo sobre $\mathbb{F}_2$ corrige todos los errores ráfaga de longitud $l \leq (t-1)r + 1$.

Demostración. Recibida una palabra binaria, a través del método contrario descrito anteriormente (ascendiendo el cuerpo), podemos transformarla en un vector de $\mathbb{F}_{2^r}$. Si los errores forman una ráfaga de longitud $l \leq (t-1)r + 1$, como cada r símbolos binarios se colapsan en uno de $\mathbb{F}_{2^r}$, la palabra transformada contiene a lo más t coordenadas erróneas, que es la capacidad correctora del código. □

Concatenación

Como bien indica el nombre, esta técnica concatena dos códigos, uno $\mathcal{C}_e$, llamado externo, y otro $\mathcal{C}_i$, llamado interno. El proceso de codificación es el siguiente:



Si suponemos que los mensajes transmitidos son binarios y $\mathcal{C}_i$ es $[n, k, d]$, puede entenderse la parte interna del proceso como un nuevo canal (a veces llamado supercanal) que transmite k-tuplas binarias. Estas tuplas pueden identificarse con elementos de $\mathbb{F}_{2^k}$ y el código $\mathcal{C}_\urcorner$ tomarse $[N, K, D]$ sobre $\mathbb{F}_{2^k}$.

El código interno permite corregir los errores aislados, pero no las ráfagas, de estas se ocupará el código externo. Los códigos Reed-Solomon se emplean habitualmente como código externo por su capacidad de corrección de ráfagas.

3.7. Códigos Localmente Recuperables con Detección de Errores Locales

Esta parte se basa en la traducción del artículo [7].

3.7.1. Introducción

Supongamos que cae un nodo con información, recuperaremos la información con códigos correctores de errores. Normalmente se emplean los Reed-Solomon, veamos como estos se emplean para dicha tarea.

La información se almacena en una larga secuencia de b símbolos, elementos de $\mathbb{F}_l$ (cuerpo finito). Esta secuencia se particiona en bloques, $b = b_1, \ldots,$ de la misma longitud m. De acuerdo al isomorfismo $\mathbb{F}_l^m \cong \mathbb{F}_{l^m}$, cada una de estos bloques se puede entender como un elemento del cuerpo finito $\mathbb{F}_q$, $q = l^m$. Fijamos un entero $k < q$. El vector $b = (b_1, \ldots, b_k) \in \mathbb{F}_q^k$ se codifica usando un código Reed-Solomon de dimensión k sobre $\mathbb{F}_q$, cuya longitud n, $k < n \leq q$, es igual al número de nodos que emplearemos en el almacenamiento. Entonces escogemos $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ y enviamos $b_1 + b_2\alpha_i + \ldots + b_k\alpha_i^{k-1}$ al i-ésimo nodo. Cuando un nodo falla, podemos recuperar la información que almacena usando la interpolación Lagrangiana de la información de cualesquiera otros k nodos disponibles.

Se pueden emplear otros tipos de códigos, por tanto, pongamos el problema en términos apropiados.

3.7.2. Códigos localmente recuperables

Definición 3.7.1 (Coordenadas localmente recuperables)**.** Sea $\mathcal{C}$ un código lineal de longitud n y dimensión k sobre $\mathbb{F}_q$. Una coordenada $i \in \{1, \ldots, n\}$ es localmente recuperable con localidad r si hay un conjunto recuperador $\mathcal{R}$, $\mathcal{R} \subseteq \{1, \ldots, n\}$ con $i \notin \mathcal{R}$ y $\#\mathcal{R} = r$, tal que para cada palabra del código $x \in \mathcal{C}$, una errata en la coordenada x_i de x se pueda recuperar usando la información dada por las coordenadas de x con índices en $\mathcal{R}$

Definición 3.7.2 (Códigos localmente recuperables)**.** Dado un código $\mathcal{C}$, se dice que es localmente recuperable (LRC) con localidad $\leq r$ si cada coordenada lo es. La localidad de $\mathcal{C}$ es la menor r que verifica dicha condición.

Definición 3.7.3 (Conjunto recuperador)**.** Sea $\mathcal{C}$ un código $[n, k, d]$. Sea G una matriz generatriz de $\mathcal{C}$ con columnas $c_1, \ldots, c_n$. Dado un conjunto $\mathcal{R} \subset \{1, \ldots, n\}$ y una coordenada $i \notin \mathcal{R}$, decimos que $\mathcal{R}$ es un conjunto recuperador para i si $c_i \in \langle c_j : j \in \mathcal{R} \rangle$, el espacio lineal abarcado por $\{c_j : j \in \mathcal{R}\}$.

Sea $\pi_{\mathcal{R}} : \mathbb{F}_q^n \to \mathbb{F}_q^r$ la proyección en las coordenadas de $\mathcal{R}$, donde $r = \#\mathcal{R}$. Para $x \in \mathbb{F}_q^n$, escribimos $x_{\mathcal{R}} = \pi_{\mathcal{R}}(x)$. Consideramos los códigos perforados y acortados $\mathcal{C}[\mathcal{R}] = \{x_{\mathcal{R}} : x \in \mathcal{C}\}$ y $\mathcal{C}[[\mathcal{R}]] = \{x_{\mathcal{R}} : x \in \mathcal{C},\ supp(x) \subseteq \mathcal{R}\}$. Empleamos la nocación $\mathcal{C}^{\perp}$ para el dual de $\mathcal{C}$.

Lema 3.7.1. $\mathcal{C}[\mathcal{R}]^{\perp} = \mathcal{C}^{\perp}[[\mathcal{R}]]$.

Nota: $c_i \in \langle c_j : j \in \mathcal{R} \rangle$ si y solo si $dim(\mathcal{C}[\mathcal{R}]) = dim(\mathcal{C}[\overline{\mathcal{R}}])$, donde $\overline{\mathcal{R}} = \mathcal{R} \bigcup \{i\}$, así la noción de conjunto recuperador no depende de la matriz generatriz escogida. En este caso, existen $w_1, \ldots, w_n \in \mathbb{F}_q$ tales que $\sum_{j \in \mathcal{R}} w_j c_j = 0$ con $w_i \neq 0$ y $w_j = 0$ si $j \notin \overline{\mathcal{R}}$. Entonces $w = (w_1, \ldots, w_n) \in \mathcal{C}^{\perp}$ y $w_{\overline{\mathcal{R}}} \in \mathcal{C}^{\perp}[[\overline{\mathcal{R}}]]$. Así obtenemos el siguiente resultado.

Lema 3.7.2. $\mathcal{R}$ es un conjunto recuperador para una coordenada i si y solo si existe una palabra $w_{\overline{\mathcal{R}}} \in \mathcal{C}^{\perp}[[\overline{\mathcal{R}}]]$ con $w_i \neq 0$. En este caso $\#\mathcal{R} \geq d(\mathcal{C}^{\perp}) - 1$.

La menor cardinalidad de un conjunto de recuperación $\mathcal{R}$ para una coordenada i es la localidad de i. La localidad de $\mathcal{C}$ es la mayor localidad de entre sus coordenadas.

Una palabra $w_{\overline{\mathcal{R}}} \in \mathcal{C}^{\perp}[[\overline{\mathcal{R}}]]$ con $w_i \neq 0$ no solo nos da un conjunto recuperador, si no también, un método recuperador: para cada $x \in \mathcal{C}$ tenemos $w \cdot x = w_{\overline{\mathcal{R}}} \cdot x_{\overline{\mathcal{R}}} = 0$, donde $\cdot$ denota el producto usual interno, $w \cdot x = w_1 x_1 + \ldots + w_n x_n$, así

$$x_i = -w_i^{-1}(w_{\mathcal{R}} \cdot x_{\mathcal{R}})$$

3.7.3. Códigos Localmente Recuperables con Detección de Errores Locales

Nuestro punto de partida es el siguiente

Lema 3.7.3. La distancia mínima de $\mathcal{C}$ es $\geq d$ si y solo si para todo $\mathcal{S} \subseteq \{1, \ldots, n\}$ con $\#\mathcal{S} > n - d$ tenemos $dim(\mathcal{C}[\mathcal{S}]) = dim(\mathcal{C})$.

La siguiente proposición es consecuencia directa del lema anterior y nos dirige a la siguiente definición de conjunto recuperador detector de errores.

Proposición 3.7.4. Un conjunto $\overline{\mathcal{R}} \backslash \{i\}$ es un conjunto recuperador para cada coordenada $i \in \overline{\mathcal{R}}$ si y solo si $d(\mathcal{C}[\overline{\mathcal{R}}]) > 1$.

Definición 3.7.4. Un conjunto $\mathcal{R} \subseteq \{1, \ldots, n\}$ se llama conjunto recuperador detector de $t \geq 0$ errores (t-edr) para una coordenada $i \notin \mathcal{R}$ si $d(\mathcal{C}[\overline{\mathcal{R}}]) > t+1$, donde $\overline{\mathcal{R}} = \mathcal{R} \bigcup \{i\}$.

Entonces, un conjunto t-edr $\mathcal{R}$ para una coordenada i es un conjunto recuperador para i; además $\mathcal{R} \bigcup \{i\} \backslash \{j\}$ es un conjunto t-edr para todo $j \in \mathcal{R}$. Si $\#\mathcal{R} = r$ de acuerdo al lema 3.7.3, $\mathcal{R}$ es un conjunto t-edr para i si y solo si $dim(\mathcal{C}[\mathcal{S}]) = dim(\mathcal{C}[\overline{\mathcal{R}}])$ para todo $\mathcal{S} \subseteq \overline{\mathcal{R}}$ con $\#\mathcal{S} \geq r - t$, donde $\overline{\mathcal{R}} = \mathcal{R} \bigcup \{i\}$. En particular, $dim(\mathcal{C}[\overline{\mathcal{R}}]) \leq r - t$.

Por otra parte, como $dim(\mathcal{C}[\overline{\mathcal{R}}]) > t+1$, implica que $d(\mathcal{C}[\mathcal{R}]) \geq t+1$, hasta t errores en cualquier palabra $x_{\mathcal{R}}$. $x \in \mathcal{C}$ pueden ser detectados. Así, cuando ocurren a lo máximo t errores en $x_{\mathcal{R}}$, un conjunto t-edr $\mathcal{R}$ detecta que errores ocurrieron así como el correcto valor de x_i. Para comprobar los errores y recuperar los borrados podemos emplear métodos como el descrito anteriormente

$$x_i = -w_i^{-1}(w_{\mathcal{R}} \cdot x_{\mathcal{R}}).$$

La mínima cardinalidad de un conjunto t-edr $\mathcal{R}$ para una coordenada i es la t-localidad de i. El código $\mathcal{C}$ se llama código t-detector de errores localmente recuperables (t-LREDC) si para cada coordenada, un conjunto recuperador que detecta t errores existe. Cada código con distancia mínma $d > t+1$ es un t-LREDC (simplemente tomamos $\overline{\mathcal{R}} = \{1, \ldots, n\}$). El máximo sobre las t-localidades es la t-localidad de $\mathcal{C}$, denotada por $r_t = r_t(\mathcal{C})$.

Ahora daremos dos cotas de $r_t(\mathcal{C})$. La primera generaliza la cota del lema 3.7.2 para $r = r_0$, usando la generalización de los pesos de Hamming.

Proposición 3.7.5. Sea $\mathcal{C}$ un código $[n, k, d]$, t-LREDC. La t-localidad r_t de $\mathcal{C}$ verifica que $r_t(\mathcal{C}) \geq d_{t+1}(\mathcal{C}^{\perp}) - 1$, donde $d_{t+1}(\mathcal{C}^{\perp})$ es el $(t+1)$-ésimo peso de Hamming generalizado de $\mathcal{C}^{\perp}$.

La segunda cota de r_t generaliza la cota de Singleton

$$n + 2 \geq k + d + \left\lceil \frac{k}{r_0} \right\rceil$$

Proposición 3.7.6. Sea $\mathcal{C}$ un código $[n, k, d]$, t-LREDC. La t-localidad de $\mathcal{C}$ verifica

$$n + t + 2 \geq k + d + \left\lceil \frac{k}{r_t - t} \right\rceil (t+1)$$

Por similaridad al caso $t = 0$, diremos que el código $\mathcal{C}$ es t-óptimo si su t-localidad alcanza la igualdad en $n + t + 2 \geq k + d + \left\lceil \frac{k}{r_t - t} \right\rceil (t+1)$, es decir,

$$n + t + 2 = k + d + \left\lceil \frac{k}{r_t - t} \right\rceil (t+1).$$

Capítulo 4

Conclusiones

A lo largo del desarrollo teórico hemos visto muchas formas de generar códigos o como codificar mensajes. El aspecto más destacable es la forma de estructurar estos códigos. Las restricciones a las que se someten, ya sean por parámetros o por cuerpos donde trabajar, permiten dotar de propiedades algebraicas a dichos códigos. Estas propiedades, a su vez, simplifican los procesos de codificación y decodificación. Permiten dar pautas o algoritmos para resolver, de una forma computacionalmente ligera, los problemas que puedan causar los ruidos en un canal de paso de mensajes. Todo unido, nos da una solución factible al problema de los errores y borrones.

Anexo A

Cuerpos Finitos

La ampliación de los cuerpos finitos, así como del Anexo B, polinomios en cuerpos finitos, viene ampliada en [6], [9], [10] [11].

El propósito de este Anexo es dar una noción sobre cuerpos finitos, su existencia, estructura o manipulación de los elementos. En el Anexo B se abordarán los polinomios sobre estos cuerpos.

Teorema A.0.1 (de Wedderburn)**.** Todo cuerpo finito es conmutativo.

A.1. Cardinal, Característica y Unicidad de los Cuerpos Finitos

A.1.1. Cardinal de un cuerpo finito

Proposición A.1.1. Para todo número natural primo p, existe un cuerpo finito con p elementos.

Demostración. Si p es un número primo, el anillo cociente $\mathbb{Z}/p\mathbb{Z}$ es un cuerpo con p elementos. □

El conjunto $\{0, 1, \ldots, p-1\}$ es un sistema completo de representantes de $\mathbb{Z}/p\mathbb{Z}$. Por lo tanto, se puede identificar dicho conjunto con $\mathbb{Z}/p\mathbb{Z}$, con sus correspondientes operaciones $+, \cdot$.

Proposición A.1.2. Si q es el cardinal de un cuerpo finito, existen un primo p y un número natural r tales que $q = p^r$.

Demostración. Sea K un cuerpo de cardinal q y denotemos por 1_K su elemento unidad. Consideramos ahora la aplicación lineal:

$$\varphi : \mathbb{Z} \to K;\ \varphi(n) = n \cdot 1_K$$

donde, si n es un número natural, $n \cdot 1_K = 1_K + \cdots + 1_K$ y, si n es negativo, $n \cdot 1_K$ es el opuesto de la anterior suma. El núcleo de φ es no nulo y fácilmente se comprueba que consiste en un ideal de la forma $p\mathbb{Z}$, siendo p un número primo. Así, K debe contener un subcuerpo de cardinal p, $k = Img(\varphi) \simeq \mathbb{Z}/p\mathbb{Z}$. Entonces K tiene estructura de espacio vectorial sobre k con dimensión finita. Si r es tal dimensión, puesto que $K \simeq k^r$, el enunciado es trivial. □

De esta demostración se puede deducir el siguiente resultado.

Corolario A.1.3. Salvo isomorfismo, $\mathbb{Z}/p\mathbb{Z}$ es el único cuerpo posible con un número primo p de elementos. Se denotará como $\mathbb{F}_p$ a lo largo de los capítulos de este trabajo.

Se emplean para estos cuerpos la notación $\mathbb{F}_p$, el único cuerpo (salvo isomorfismo) con p elementos. Estos cuerpos son llamados cuerpos primos.

Teorema A.1.4. Para todo $q = p^r$, existe un cuerpo finito con q elementos.

Demostración. Sea $\mathbb{F}_p[X]$ el anillo de polinomios sobre el cuerpo primo con p elementos y sea $f(X)$ un polinomio irreducible de grado r. El anillo cociente $A = \mathbb{F}_p[X]/\langle f(X)\rangle$ es un cuerpo. Empleando la división euclídea se prueba que todo polinomio de $\mathbb{F}_p[X]$ posee un único representante en A de grado $< r$. Por tanto, A puede identificarse con el conjunto de polinomos de $\mathbb{F}_p[X]$ de grado $< r$ y su cardinal sería p^r. □

A.1.2. Característica de un cuerpo finito

Dado un cuerpo finito $\mathbb{F}_q$, si $q = p^r$, como $\mathbb{F}_q$ contiene a $\mathbb{F}_p$, para todo $a \in \mathbb{F}_q$ se tiene que

$$p \cdot a = \overbrace{a + \ldots + a}^{p \text{ veces}} = (p \cdot 1) \cdot a = 0 \cdot a = 0$$

y p es el mínimo con tal propiedad.

Definición A.1.1 (Característica)**.** Si $q = p^r$ diremos que el cuerpo tiene característica p.

Proposición A.1.5. Si $\mathbb{F}_q$ es un cuerpo de característica p, entonces para cada par de elementos $a, b \in \mathbb{F}_q$ y cada entero positivo s, se verifica que

$$(a + b)^{p^s} = a^{p^s} + b^{p^s}.$$

Demostración. Basta con desarrollat el binomio de Newton y aplicar las propiedades de los cuerpos finitos. □

A.1.3. Unicidad de los cuerpos finitos

Lema A.1.6. Sea $\mathbb{F}_q$ un cuerpo finito contenido, como subcuerpo , en otro cuerpo finito $\mathbb{F}_s$. Sea α un elemento no nulo de $\mathbb{F}_s$. Entonces el subconjunto de $\mathbb{F}_q[X]$ de polinomios que tienen a α por raíz, no se reduce al polinomio cero y coincide con el de múltiplos de un polinomio mónico irreducible.

Demostración. Si $\mathbb{F}_s$ es un espacio vectorial de dimensión finita sobre $\mathbb{F}_q$, r es la dimensión, por lo que el conjunto $\{1, \alpha, \ldots, \alpha^r\}$ es linealmente dependiente sobre $\mathbb{F}_q$. Sea $f(X)$ el polinomio mónico no nulo de grado mínimo, esto es con $deg(f(X)) \leq r$, que admite α como raíz. Al ser de grado mínimo, $f(X)$ es irreducible. Esto es así porque si $f(X) = f_1(X)f_2(X)$, entonces α es raíz de $f_1(X)$ o de $f_2(X)$, contradiciendo que el grado de $f(X)$ sea mínimo. Sea ahora $g(X)$ un polinomio que tiene a α como raíz. Por la división euclídea se puede escribir $g(X) = f(X)c(X) + h(X)$. Esto deja a $h(X)$ como un polinomio de grado menor al de $f(X)$, pero si evaluamos en α, obtenemos que $g(\alpha) = 0 = h(\alpha)$, luego $h(X) = 0$ y $g(X)$ es múltiplo de $f(X)$. □

Definición A.1.2 (Polinomio mínimo)**.** El polinomio $f(X)$, cuya existencia y unicidad asegura el lema anterior, se denomina el polinomio mínimo o irreducible de α sobre $\mathbb{F}_q$.

Nota: Lo denotamos por $Irr(\alpha, \mathbb{F}_q)$ o $Irr(\alpha)$ si no hay posibilidad de confusión.

Proposición A.1.7. En un cuerpo finito $\mathbb{F}_q$, cada $a \in \mathbb{F}_q$ verifica que $a^q = a$.

Demostración. Tenemos dos casos posibles:

1) Si $a = 0$, el resultado enunciado está claro.

2) Si $a \neq 0$, por el teorema de Lagrange, el orden de a divide al orden del grupo multiplicativo $\mathbb{F}_q^*$, luego $a^{q-1} = 1$ y $a^q = a$.

□

Como consecuencia inmediata de esta proposición, tenemos el siguiente resu

Corolario A.1.8. El polinomio $X^q - X$ factoriza completamente sobre el cuerpo $\mathbb{F}_q$, es decir,

$$X^q - X = \prod_{x \in \mathbb{F}_q} (X - a).$$

Se ha mostrado que los elementos de $\mathbb{F}_q$ coinciden con las raíces de $X^q - X$ en dicho cuerpo. Por ello, se puede tomar una definición alternativa de $\mathbb{F}_q$ como el conjunto de raíces de $X^q - X \in \mathbb{F}_p[X]$. Esta definición es válida siempre que se pueda asegurar la existencia de esas raíces en algún cuerpo de característica p. Esto se conoce como clausura algebraica de $\mathbb{F}_p$.

Nota: La clausura algebraica de un cuerpo K es el cuerpo más pequeño que contiene a K y a las raíces de $K[X]$.

Proposición A.1.9. Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio irreducible de grado r y sea $q = p^r$. El polinomio se factoriza completamente sobre $\mathbb{F}_q$ (es decir, tiene r raíces en $\mathbb{F}_q$).

Demostración. Sea el cuerpo con q elementos $\mathbb{F}_p[X]/\langle f(X)\rangle$ y sea x la clase de X en este cuerpo. Como x es una raíz de $f(X)$, tenemos que $f(X) = Irr(x, \mathbb{F}_p)$. Como x es también raíz de $X^q - X$, por el corolario anterior y por el lema A.1.6, se tiene que $f(X)|X^q - X$. Puesto que $X^q - X$ factoriza completamente en $\mathbb{F}_q$, también ocurre para $f(X)$. □

Corolario A.1.10. Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio irreducible de grado r y sea $q = p^r$. Como espacio vectorial, $\mathbb{F}_q$ es isomorfo al conjunto de expresiones polinómicas $a_0 + a_1\alpha + \ldots + a_{r-1}\alpha^{r-1}$, siendo α una raíz cualquiera de $f(X)$ en $\mathbb{F}_q$ y $a_i \in \mathbb{F}_p$.

Demostración. Basta on probar que el conjunto $\{1, \alpha, \ldots, \alpha^{r-1}\}$ constituye una base de $\mathbb{F}_q$ como espacio vercorial sobre $\mathbb{F}_p$. En este caso, forma un conjunto linealmente independiente (al contrario α sería una raíz de un polinomio de grado menor que r), por lo que es un base. □

Teorema A.1.11 (Unicidad de un cuerpo finito)**.** Para toda potencia q de un número primo existe, salvo isomorfismo, un solo cuerpo finito con q elementos.

Demostración. Sea $q = p^r$ y $f(X) \in \mathbb{F}_p[X]$ un polinomio irreducible de grado r. Probemos que F_q es isomorfo a $F_p[X]/\langle f(X)\rangle$. Por el corolario alterior, los elementos de $\mathbb{F}_q$ pueden expresarse en la forma $a_0 + a_1\alpha + \cdots + a_{r-1}\alpha^{r-1}$, siendo α una raíz de $f(X)$. El isomorfismo buscado se otiene asignando a tal elemento el $a_0 + a_1X + \cdots + a_{r-1}X^{r-1}(mod\ f(X))$. □

A.2. Estructuras de los Cuerpos Finitos

El cuerpo finito $\mathbb{F}_q$ contiene dos grupos abelianos, $(\mathbb{F}_q, +)$ y $(\mathbb{F}_q, \cdot)$.

Teorema A.2.1 (Estructura aditiva)**.** Si $q = p^r$, el grupo aditivo $(\mathbb{F}_q, +)$ es un producto directo de r grupos cíclicos de orden p:

$$(\mathbb{F}_q, +) \simeq \mathbb{Z}/p\mathbb{Z} \times \ldots \times \mathbb{Z}/p\mathbb{Z}.$$

Demostración. Puesto que $\mathbb{F}_q$ es un espacio vectorial sobre el cuerpo primo $\mathbb{F}_p$ cualquier base induce el isomorfismo anterior. □

Definición A.2.1 (Orden)**.** Dado un grupo abeliano finito $(G, \cdot)$, llamamos orden de $x \in G$ al orden del subgrupo engendrado por x, es decir,

$$ord(x) = \text{mín}\{n \ / \ x^n = 1\}.$$

Definición A.2.2 (Exponente del grupo)**.** Sea $(G, \cdot)$ un grupo abeliano finito. Llamaremos exponente de G a

$$exp(G) = m.c.m\{ord(x) \ / \ x \in G\}.$$

Lema A.2.2. Si $(G, \cdot)$ es un grupo abeliano finito de exponente n, entonces existe un elemento $x \in G$ de orden n.

Demostración. Sea $n = p_1^{e_1} \cdots p_m^{e_m}$ la descomposición de n en factores primos. Como $p_i^{e_i}$ aparece en la factorización, existe $x_i \in G$ de orden $k_i p_i^{e_i}$ para un cierto natural k_i. Entonces el elemento $x_i^{k_i}$ tendrá orden exáctamente $p_i^{e_i}$ y, por tanto, $x = \prod_{i=1}^m x_i^{k_i}$ tendrá orden exáctamente n. □

Teorema A.2.3 (Estructura multiplicativa)**.** El grupo multiplicativo $(\mathbb{F}_q^*, \cdot)$ es cíclico de orden $q - 1$.

Demostración. Sea n el exponente de $\mathbb{F}_q^*$. Por el lema anterior existe un elemento de orden n. Por tanto $n \leq q - 1 = \#(\mathbb{F}_q^*)$. Po otro lado, al ser n múltiplo del orden de todo elemento, los $q - 1$ elementos de $\mathbb{F}_q^*$ satisfacen la ecuación $X^n - 1 = 0$, con lo que $q - 1 \leq n$ y, finalmente, $n = q - 1$. Al existir un elemento de orden $q - 1$, el grupo es cíclico. □

Definición A.2.3 (Elemento primitivo)**.** Llamaremos elemento primitivo de $\mathbb{F}_q$ a un generador del grupo cíclico $(\mathbb{F}^*, \cdot)$.

Así, si α es un generador de $\mathbb{F}_q$, entonces $\mathbb{F}_q = \{0, \alpha, \alpha^2, \ldots, \alpha^{q-1} = 1\}$

Teorema A.2.4. Si $q = p^r$ y α es un elemento primitivo de $\mathbb{F}_q$, entonces $\mathbb{F}_q = \mathbb{F}_q[\alpha]$.

Demostración. El polinomio irreducible de α debe tener grado r, pues en caso contraro, por la proposición A.1.9, α pertenecería a un cuerpo con menos de q elementos y su orden sería inferior a $q - 1$. Así $\mathbb{F}_p[\alpha]$ tiene al menos q elementos y, como $\mathbb{F}_p[\alpha] \subseteq \mathbb{F}_q$, el resultado está claro. □

A.3. El Retículo de Subcuerpo

Sabemos que todo cuerpo finito $\mathbb{F}_q$ contiene al cuerpo primo $\mathbb{F}_p$, siendo p la característica de $\mathbb{F}_q$. Veamos qué otros subcuerpos contiene $\mathbb{F}_q$

Teorema A.3.1. Sea $q = p^r$. Todo subcuerpo de $\mathbb{F}_q$ tiene cardinal p^s con $s|r$. Recíprocamente, para todo s tal que $s|r$, existe un único subcuerpo de $\mathbb{F}_q$ con cardinal p^s. El conjunto de subcuerpos de $\mathbb{F}_q$ tiene pues una estructura de retículo, donde las relaciones de contención se corresponden con las relaciones de divisibilidad entre los divisores de r.

Demostración.

- Un subcuerpo k de $\mathbb{F}_q$ tendrá cardinal p^s, $s \leq r$, ya que k contiene al subcuerpo primo $\mathbb{F}_p$. Además $\mathbb{F}_q$ es un espacio vectorial sobre k, su cardinal, $q = p^r$, será una potencia de p^s y, por tanto, $s|r$.
- Si $s|r$, se comprueba que $p^s-1|p^r-1$, por lo que $X^{p^s-1}-1|X^{p^r-1}-1$ y $X^{p^s}-X|X^{p^r}-X$. Por la factorización en A.1.8, existe un subcuerpo de $\mathbb{F}_q$ con p^s elementos. Si $\mathbb{F}_q$ contuviera dos subcuerpos distintos con dicho cardinal, el conjunto de los elementos de la unión contendría más de p^s elementos. Todos ellos deberían ser raíces de $X^{p^s} - X$, polinomio que no admite más de p^s raíces.

□

Corolario A.3.2. Sean K_i, $i = 1, \ldots, s$, cuerpos finitos con p^{r_i} elementos y sea $r = mcm(r_1, \ldots, r_s)$. El cuerpo $\mathbb{F}_{p^r}$ contiene cada cuerpo K_i, o algún subcuerpo isomorfo, y es el mínimo con tal propiedad.

A.4. Conjugación

Sea $\mathbb{F}_{q^m}$ el cuerpo finito con q^m elementos. Por le teorema A.3.1, este cuerpo contiene a $\mathbb{F}_q$. Tomemos $\alpha \in \mathbb{F}_{q^m}^*$, y sea $f(X)$ el polinomio irreducible de α sobre $\mathbb{F}_q$. Si $\mathbb{F}_{q^r}$ es el cuerpo más pequeño que contiene a α, entonces $f(X)$ tiene grado r y, según A.1.9, se factoriza completamente en $\mathbb{F}_{q^r}$, luego también en $\mathbb{F}_{q^m}$.

Definición A.4.1 (Elemento conjugado)**.** Llamaremos elemento conjugado de α sobre $\mathbb{F}_q$ a cualquier raíz (en $\mathbb{F}_{q^m}$) de $f(X)$.

El siguiente resultado caracteriza los conjugados de α.

Proposición A.4.1. Sea r el grado de $f(X) = Irr(\alpha, \mathbb{F}_q)$. Entonces α posee r elementos conjugados distintos (incluido él mismo). Explícitamente tales conjugados son $\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{r-1}}$.

Demostración. Si $\alpha \in \mathbb{F}_{q^m}$ también $\alpha^q, \ldots, \alpha^{q^{r-1}} \in \mathbb{F}_{q^m}$. Sea $f(X) = a_0 + a_1X + \cdots + a_{r-1}X^{r-1} + X^r$, $a \in \mathbb{F}_q$. Por A.1.5 y A.1.7

$$\begin{aligned} f(X^q) &= a_0 + \cdots + a_{r-1}X^{q(r-1)} + X^{qr} \\ &= (a_0 + \cdots + a_{r-1}X^{(r-1)} + X^r)^q \\ &= f(X)^q \end{aligned}$$

se demuestra por recurrencia que $\alpha^q, \alpha^{q^2}, \ldots$ son también raíces de $f(X)$. Sólo falta probar que los elementos $\alpha, \alpha^q, \ldots, \alpha^{q^{r-1}}$, son todos distintos. Si $\alpha^{q^i} = \alpha^{q^j}$ para algunos i, j, $0 \leq i < j \leq r-1$, se tendría entonces que $\alpha^{q^{j-i}} = 1$ con $j - i < r$. Por lo que $\alpha \in \mathbb{F}_{q^{j-i}}$ con lo que sería la raíz de un polinomio de grado menor que r. □

Corolario A.4.2. Sea $\alpha \in \mathbb{F}_{q^m}$. Si r es el menor entero positivo tal que $\alpha^{q^r} = \alpha$, entonces $Irr(\alpha, \mathbb{F}_q) = (X - \alpha) \cdot (X - \alpha^q) \cdot \ldots \cdot (X - \alpha^{q^{r-1}})$.

Corolario A.4.3. Los elementos conjugados de α tienen todos igual orden (como elementos de $\mathbb{F}^*_{q^m}$). En particular, si α es un elemento primitivo. entonces lo son también todos sus conjugados.

A.4.1. Automorfismos de un cuerpo finito

Tomamos la aplicacion

$$\sigma : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m};\ \sigma(X) = X^q.$$

Es un automorfismo de $\mathbb{F}_{q^m}$ por A.1.5, que deja invariantes los elementos de $\mathbb{F}_q$ por A.1.7. Un automorfismo de $\mathbb{F}_{q^m}$ con tal propiedad, se llama $\mathbb{F}_q$-automorfismo.

Definición A.4.2 (Automorfismo de Frobenius)**.** El $\mathbb{F}_q$-automorfismo σ de $\mathbb{F}_q$,

$$\sigma : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m};\ \sigma(X) = X^q$$

se denomina automorfismo de Frobenius de $\mathbb{F}_{q^m}$ respecto de $\mathbb{F}_q$.

Teorema A.4.4. Los $\mathbb{F}_q$-automorfismos de $\mathbb{F}_{q^m}$ constituyen un grupo cíclico de orden m engendrado por el automorfismo de Frobenius.

Demostración. Por la proposición A.4.1 , $\langle\sigma\rangle$ es un grupo de $\mathbb{F}_q$-automorfismos de $\mathbb{F}_{q^m}$ de orden m. Veamos que todo $\mathbb{F}_q$-automorfismo φ es de la forma σ^i para algún i. Sea β un elemento primitivo de $\mathbb{F}_{q^m}$, de orden $q^m - 1$, y sea $g(X) = b_0 + \cdots + b_{m-1}X^{m-1} + X^m$ su polinomio irreducible sobre $\mathbb{F}_q$. Entonces

$$0 = \varphi(b_0 + \cdots + b_{m-1}\beta^{m-1} + \beta^m) = b_0 + \cdots + b_{m-1}\varphi(\beta)^{m-1} + \varphi(\beta)^m,$$

lo que muestra que $\varphi(\beta)$ es un conjugado de β. Por lo que $\varphi(\beta) = \beta^{q^i} = \sigma^i(\beta)$ para algún i. Así queda probado el resultado, pues es β un generador de $\mathbb{F}^*_{q^m}$. □

Anexo B

Polinomios en Cuerpos Finitos

B.1. Polinomios Ciclotómicos sobre Cuerpos Finitos

B.1.1. Raíces n-ésimas de la unidad

Definición B.1.1 (Derivada formal)**.** Sea K un cuerpo de característica p. Dado un polinomio $f(X) \in K[X]$, $f(X) = a_0 + a_1X + \ldots + a_mX^m$, se define su derivada formal como $f'(X) = a_1 + 2a_2X + \ldots + ma_mX^{m-1}$.

Lema B.1.1. α es una raíz múltiple de $f(X)$ si y solo si es raíz de $f(X)$ y de $f'(X)$.

Demostración. Supongamos que α es una raíz múltiple de $f(X)$. Sea $\lambda > 1$ la multiplicidad de α. Podemos expresar $f(X) = (X-\alpha)^\lambda g(X)$ para algún $g(X) \in K[X]$ donde $g(X)$ no es divisible por $(X-\alpha)$. Derivamos $f(X)$ y nos queda

$$\begin{aligned} f'(X) &= \lambda(X-\alpha)^{\lambda-1}g(X) + (X-\alpha)^\lambda g'(X) \\ &= (X-\alpha)^{\lambda-1}(\lambda g(X) + (X-\alpha)g'(X)) \end{aligned}$$

por lo que α es raíz de $f'(X)$. Además, $(X-\alpha)$ no divide a $(\lambda g(X) + (X-\alpha)g'(X)$. Si lo dividiera, como $(X-\alpha)g'(X)$ si es divisible, obliga a $\lambda g(X)$ a serlo, contradiciendo nuestra elección de $g(X)$.
Por otro lado, sabemos que α es una raíz de $f(X)$ y de $f'(X)$. Supongamos ahora que $\lambda = 1$, es decir, que α no es una raíz multiple de $f(X)$. Si $\lambda = 1$, entonces $(X-\alpha)$ no divide a $f'(X)$, por lo que α no es raíz de $f'(X)$, contradiciendo nuestra hipótesis, por lo que $\lambda > 1$ y, por tanto, α una raíz múltiple $f(X)$. □

Definición B.1.2 (Raíces n-ésimas de la unidad)**.** Si n es un número natural no divisible por p, el polinomio $X^n - 1$ tiene, en la clausura algebráica de K, n raíces distintas a las que se conoce como raíces n-ésimas de la unidad. Si α es una de dichas raíces, entonces $\alpha^n = 1$.

Si se multiplican dos raíces n-ésimas de la unidad se produce de nuevo una raíz n-ésima de la unidad, así el conjunto de todas ellas forma un grupo. Sus generadores son llamados raíces n-ésimas primitivas de la unidad. Si α es una raíz n-ésima primitiva de la unidad, las demás raíces son $\alpha^2, \ldots, \alpha^n = 1$.

Sea α^i una de estas raíces, es primitiva si es también generadora del grupo, es decir, si existe $j \in \mathbb{N}$ tal que $(\alpha^i)^j = \alpha$. Ocurrirá si y solo si $mcd(i, n) = 1$. Por lo que hay exactamente $\phi(n)$ raíces primitivas de la unidad, siendo ϕ la función de Euler.

Nota: La función de Euler es la siguiente: Sea n un número entero positivo, $\phi(n)$ se define como el número de enteros positivos menores o iguales a n y que son primos entre sí. Esto último hace que si $m \leq n$ es primo con n, $mcd(m, n) = 1$. La función queda

$$\phi(n) = \{m \in \mathbb{N} \mid m \leq n,\ mcd(m, n) = 1\}.$$

B.1.2. Polinomios ciclotómicos

Sea α una raíz primitiva n-ésima de la unidad.

Definición B.1.3 (Polinomio ciclotómico)**.** Llamaremos n-ésimo polinomio ciclotómico sobre K a

$$\lambda(X) = \prod_{mcd(i,n)=1} (X - a^i).$$

Como toda raíz n-ésima de la unidad es raíz primitiva de algún orden $d|n$, se obtiene la descomposición

$$X^n - 1 = \lambda_n(X) \prod_{d|n, d<n} \lambda_d(X).$$

El polinomio $\lambda_n(X)$ tiene grado $\phi(n)$.

Lema B.1.2. Sea n un número natural primo con q y sea d el orden multiplicativo de q en $\mathbb{Z}/n\mathbb{Z}$. El polinomio ciclotómico $\lambda_n(X)$ factoriza sobre el cuerpo $\mathbb{F}_q$ un $\phi(n)/d$ polinomios mónicos irreducibles de grado d

Demostración. Tomemos β raíz del polinomio $\lambda_n(X)$. El grado de $Irr(\beta, \mathbb{F}_q)$ es exactamente la dimensión de $\mathbb{F}_q[\beta]$ como $\mathbb{F}_q$-espacio vectorial. Como hemos supuesto que d es el menor número

natural tal que $q^d \equiv 1(mod\ n)$ por lo que, como β es una raíz primitiva n-ésima, también el menor tal que $\beta^{q^d-1} = 1$. De este razonamiento se deduce que la dimensión del espacio vectorial $\mathbb{F}_q[\beta]$ es d, lo que nos da $deg(Irr(\beta, \mathbb{F}_q)) = d$. Este planteamiento se emplea en cada raíz β de $\lambda_n(X)$. □

Este lema nos permite conocer el número y grado de los factores irreducibles de $X^n - 1$.

B.2. El Orden de un Polinomio

Todo polinomio $f(X) \in \mathbb{F}_q[X]$ con $f(0) \neq 0$ es divisor de $X^n - 1$ para algún entero positivo n. Esto es asi puesto que toda raíz α de $f(X)$ es de orden finito, debido a que es elemneto de un cuerpo finito, por lo que existe algún entero positivo m que cumple que $\alpha^m = 1$ y $(X - \alpha)|(X^m - 1)$. Entonces se deduce un n tal que $f(X)|X^n - 1$. Se da a continuación una demostración que acota n.

Lema B.2.1. Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio de grado $d \geq 1$ con $f(0) \neq 0$. Existe un entero natural $e \leq q^d - 1$ tal que $f(X)|X^e - 1$.

Demostración. Sea el anillo cociente $\mathbb{F}_q/\langle f(X)\rangle$. Este anillo posee $q^d - 1$ elementos no nulos. Por lo que, dos elementos o más del conjunto $\{X^i + \langle f(X)\rangle \mid i = 0, 1, \ldots, q^d - 1\}$ deben ser iguales. Tomemos $X^r + \langle f(X)\rangle = X^s + \langle f(X)\rangle$ con $r > s \geq 0$, o lo que es lo mismo, $X^r \equiv X^s(mod\ f(X))$. Ahora, como por hipótesis, $mcd(X, f(X)) = 1$, es $X^{r-s} \equiv 1(mod\ f(X))$ y $f(X)|X^{r-s} - 1$. □

Definición B.2.1 (Orden de un polinomio)**.** Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio no constante con $f(0) \neq 0$. Llamaremos orden de $f(X)$, y lo denotaremos $ord(f(X))$, al menor entero natural e tal que $f(X)|X^e - 1$. Si $f(X)$ es constante, $f(X) = \lambda$, pondremos $ord(f(X)) = 1$. Si $f(0) = 0$, será $f(X) = X^t g(X)$ con $g(X) \neq 0$, y pondremos $ord(f(X)) = ord(g(X))$.

Proposición B.2.2. Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio irreducible de grado $d \geq 1$ con $f(0) \neq 0$. El orden de $f(X)$ coincide con el orden de cualquiera de sus raíces (en el grupo multiplicativo $\mathbb{F}^*_{q^d}$).

Demostración. Todas las raíces del polinomio $f(X)$ son conjugadas, así que tienen el mismo orden. Sea α una raíz y e el orden de la misma. Tenemos que $\alpha^e = 1$, por lo que α es raíz de $X^e - 1$ y, como $f(X)$ es el polinomio mínimo de α, entonces $f(X)|X^e - 1$. Concluimos que $ord(f(X)) \leq e$. Ahora bien, si $ord(f(X)) < e$, entonces α sería una raíz de la unidad de orden menor que e, pero esto contradice nuestra hipótesis, por lo que $ord(f(X)) = e$. □

Corolario B.2.3. Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio irreducible de grado $d \geq 1$ con $f(0) \neq 0$. Entonces $ord(f(X))|q^d - 1$.

Demostración. El orden de α en el grupo $\mathbb{F}_{q^d}^*$ divide al orden del grupo. □

B.3. Número de Polinomios Irreducibles

Calcularemos el número de polinomios irreducibles de grado r sobre $\mathbb{F}_q$.

Proposición B.3.1. $X^{q^r} - X$ es el producto de todos los polinomios irreducibles sobre $\mathbb{F}_q$ cuyo grado divide a r.

Demostración. $X^{q^r} - X$ no posee raíces múltiples, pues su polinomio derivado es -1. Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio irreducible de grado s tal que $s|r$. Por la proposición A.1.9, $\mathbb{F}_{q^s}$ contiene las s raíces de $f(X)$. Tomemos α una de dichas raíces. Puesto que $s|r$, α está en $\mathbb{F}_{q^r}$, luego $\alpha^{q^r} = \alpha$ y $f(X)|X^{q^r} - X$. Por otra parte, si $f(X)$ es irreducible y divide a $X^{q^r} - X$, razonando de manera similar, se muestra que su grado es un divisor de r. □

Corolario B.3.2. Si denotamos por $N_q(d)$ el número de polinomios irreducibles de grado d sobre $\mathbb{F}_q$, entonces se tiene que

$$q^r = \sum_{s|r} sN_q(s).$$

Definición B.3.1 (Función de Möbius)**.** Llamaremos función de Möbius a la función de variable natural, $\mu : \mathbb{N} \to \{-1, 0, 1\}$ definida del modo siguiente: si $n \in \mathbb{N}$ y $n = \prod_{i=0}^{s} p_i^{e_i}$ es su descomposición en factores primos, entonces

$$\mu(n) = \begin{cases} 1 & \text{si } n = 1 \\ 0 & \text{si } e_i \geq 2 \text{ para algún valor de } i \\ (-1)^s & \text{si } e_i = 1 \text{ para todo valor de } i \end{cases}$$

Lema B.3.3. Si $n \in \mathbb{N}$, entonces se verifica que

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & \text{si } n = 1 \\ 0 & \text{si } n > 1 \end{cases}.$$

Demostración.

- Si $n = 1$, el resultado es trivial.
- Si $n > 1$. Supongamos $p_1, p_2, \ldots, p_s$ los divisores primos de n, donde $p_i \neq p_j$ con $i \neq j$

para cada $i, j = 1, \ldots, s$. Por definición de la función de Möbius tenemos

$$\begin{aligned}\sum_{d|n} \mu(d) &= \mu(1) + \sum_{i=1}^{s} \mu(p_i) + \sum_{1\leq i<j\leq s} \mu(p_i p_j) + \cdots + \mu(p_1 p_2 \cdots p_s) \\ &= 1 + \binom{s}{1}(-1) + \binom{s}{2}(-1)^2 + \cdots + \binom{s}{s}(-1)^s \\ &= (1-1)^s = 0.\end{aligned}$$

□

Lema B.3.4 (Fórmula de inversión de Möbius)**.** Sea f una función de variable natural con valores en un grupo abeliano. Para $n \in \mathbb{N}$ definamos $g(n)$ mediante

$$g(n) = \sum_{d|n} f(d).$$

Entonces se verifica que

$$f(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) = \sum_{d|n} \mu\left(\frac{n}{d}\right) g(d).$$

Demostración.

$$\begin{aligned}\sum_{d|n} \mu(d) g(\frac{n}{d}) &= \sum_{d|n} \mu(d) \sum_{e|(n/d)} f(e) \\ &= \sum_{e|n} \sum_{d|(n/e)} \mu(d) f(e) \\ &= \sum_{e|n} f(e) \sum_{d|(n/e)} \mu(d) \\ &= f(n).\end{aligned}$$

□

Teorema B.3.5. El número de polinomios irreducibles de grado r sobre $\mathbb{F}_q$ es

$$N_q(r) = \frac{1}{r} \sum_{s|r} \mu(s) q^{\frac{r}{s}} = \frac{1}{r} \sum_{s|r} \mu\left(\frac{r}{s}\right) q^s.$$

Demostración. Se aplica la fórmula de inversión de Möbius sobre la función $f(r) = rN_q(r)$ □

Corolario B.3.6. Para todo cuerpo finito $\mathbb{F}_q$ y todo entero natural r, existe un polinomio irreducible de grado r sobre $\mathbb{F}_q$.

Demostración. Esto ocurre por que la suma de la expresión $N_q(r)$, mediante el teorema anterior, es siempre positiva. □

B.4. Algoritmo de Euclides

Existe una división con resto conocida como euclídea. Dados $f_0, f_1 \in \mathbb{F}_q[X]$ con $def(f_0(X)) \geq deg(f_1(X))$, existen polinomios $q(X)$ y $r(X)$ tales que $deg(r(X)) < deg(f_1(X))$ que cumplen

$$f_0(X) = f_1(X)q(X) + r(X).$$

Sean $f_0(X), f_1(X) \in \mathbb{F}_q[X]$ con $deg(f_0(X)) \geq deg(f_1(X))$, veamos el proceso de determinación de $m(X) = mcd(f_0(X), f_1(X))$, realizamos lo siguiente:

$$\begin{aligned} f_0(X) &= f_1(X)q_1(X) + f_2(X) \\ f_1(X) &= f_2(X)q_2(X) + f_3(X) \\ f_2(X) &= f_3(X)q_3(X) + f_4(X) \\ &\vdots \end{aligned}$$

Con $deg(f_1(X)) > deg(f_2(X)) > \cdots$, por lo que, en un número finito de pasos, se tiene un resto tal que $deg(f_k(X)) = 0$

Proposición B.4.1. Con las notaciones anteriores, si $f_k(X) = 0$, entonces

$$mcd(f_0(X), f_1(X)) = f_{k-1}(X).$$

Demostración. Tenemos que $f_k(X) = 0$, por lo tanto $f_{k-1}(X)|f_{k-2}(X)$. Si iteramos el proceso, nos queda $f_{k-1}(X)|f_{k-3}(X), \ldots, f_{k-1}(X)|f_1(X), f_{k-1}(X)|f_0(X)$. Así pues, $f_{k-1}(X)|mcd(f_0(X), f_1(X)) = m(X)$. Por el otro lado, tenemos que $m(X)|f_0(X)$ y $m(X)|f_1(X)$, por lo que verificamos que $m(X)|f_2(X) = f_0(X) - f_1(X)q_1(X)$, e, al igual que antes, iterando, llegamos a que $m(X)|f_{k-1}(X)$. Si aunamos ambos resultados, y sabiendo que la factorización en $\mathbb{F}_q[X]$ es única, obtenemos que $m(X) = f_{k_1}(X)$ (salvo multiplicación por constante). □

Se puede escribir $m(X) = mcd(f_0(X), f_1(X))$ como combinación lineal de f_0 y $f_1(X)$ con coeficientes en $\mathbb{F}_q[X]$. Definamos estos polinomios $u_i(X), v_i(X)$ como

$$u_0(X) = 0 \quad u_1(X) = 1; v_0(X) = 1 \quad v_1(X) = 0;$$

y, por el proceso de iteración para $1 < i < k$,

$$u_{i+1}(X) = u_i(X)q_i(X) + u_{i-1}(X);$$
$$v_{i+1}(X) = v_i(X)q_i(X) + v_{i-1}(X);$$

Proposición B.4.2. Con las notaciones anteriores, para cada $i = 0, \ldots, k-1$, se verifica que

$$f_i(X) = (-1)^i(f_0(X)v_i(X) - f_1(X)u_i(X)).$$

Proposición B.4.3. Los polinomios $u_i(X)$, $v_i(X)$ verifican que,

a) si $i \geq 1$, entonces $deg(u_i(X)) = deg(f_0(X)) - deg(f_{i-1}(X))$; y

b) $u_i(X)v_{i+1}(X) - v_i(X)u_{i+1}(X) = (-1)^{i+1}$

Ambas dos demostraciones, se realizan por ejercicio de inducción sobre i.

Bibliografía

[1] D.A. Huffman, *'A method for the construction of minimum-redundancy codes'*, 1952.

[2] C.E. Shannon, *'A mathematical theory of communication'*, 1948.

[3] F.J. MacWilliams, N.J.A. Sloane, *'The theory of error-correcting codes'*

[4] Xiaomin Bao, *'The equivalent identities of the MacWilliams identity for linear codes'*, 2013.

[5] David Cox, John Little, Donal O'shea, *'Ideals, Varieties, and Algorithms'*, Third Edition, 2007.

[6] Carlos Munuera, Juan Tena, *'Codificación de la información'*, 1997.

[7] Carlos Munuera, *'Locally recoverable codes with local error detection'*, 2018.

[8] Cecilia E. Sandoval Ruiz, *Antonio Fendón 'Codificador y decodificador digital Reed-Solomon programados para hardware reconfigurable'*, 2007.

[9] Llorenç Huguet, Josep Rifà, Juan Tena, *'Cuerpos Finitos'*, Apartados [1-3].

[10] Thomas W. Judson, Stephen F. Austin, *'Abstract algebra: Theory and Applications'*, 2012.

[11] María Jesús Iranzo, Francisco Pérez, *Elementos de álgebra. Aplicaciones.*