UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Introducción a los códigos Reed Solomon

Apellidos y nombre: Flores Asenjo, Santiago J. (sflores@dcom.upv.es)[1]

Departamento/Centro: [1]Departamento de Comunicaciones
Escola Politècnica Superior de Gandia
Universitat Politècnica de València

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Índice general

1. Resumen 2

2. Introducción 2

3. Objetivos 3

4. Desarrollo 3

   4.1. Ratio y capacidad de corrección de los códigos RS 3

   4.2. Codificación y descodificación RS con MATLAB® 4

5. Cierre 6

Bibliografía 7

1 Resumen

En este artículo se van a presentar las características básicas de los códigos Reed Solomon, que quedan resumidas en la tabla 1.

Características de los códigos Reed Solomon	
1.	Son un tipo de código cíclico no binario.
2.	Especialmente útiles en presencia de errores a ráfagas.
3.	Presentan una redundancia moderada y una gran capacidad de corrección (se pueden llegar a corregir hasta t símbolos erróneos en un bloque, simplemente añadiéndole $2t$ símbolos adicionales).
4.	Muy utilizados en telecomunicación conjuntamente con otras técnicas que sean más robustas frente a errores aislados.

Tabla 1: Características de los códigos Reed Solomon

2 Introducción

Los códigos Reed Solomon (o **códigos RS**) son un tipo especial de códigos cíclicos ideados por Irving S. Reed y Gustave Solomon (de ahí su nombre) en el año 1960.

Al ser especialmente útiles en la corrección de errores a ráfagas, que afectan a bloques contiguos de bits, son muy utilizados en Telecomunicación. Entre sus múltiples aplicaciones, se encuentran las siguientes:

- Telecomunicaciones espaciales: comunicaciones vía satélite, comunicaciones de espacio profundo (*deep space*),...
- Tecnología DSL (*Digital Subscriber Line*) y sus variantes (ADSL, VDSL,...)
- Soportes digitales ópticos y magnéticos (discos duros, CD Audio, CD-ROM, DVD, Blu-Ray,...)[1]
- Códigos de barras (uni y bidimensionales)
- Radiodifusión digital: DVB (*Digital Video Broadcast*)y sus variantes (DVB-T, DVB-S, DVB-C,...)
- Radioenlaces digitales por microondas, comunicaciones móviles, ...

[1] En estos soportes, la información digital es almacenada en bloques contiguos y, para corregir cualquier error de lectura (provocado por alteración magnética localizada, ralladuras de la superficie, polvo, etc.), se utilizan los códigos RS de manera habitual desde hace décadas.

3 Objetivos

Una vez que el alumno se lea con detenimiento este artículo docente, será capaz de:

- Comprobar las prestaciones de los códigos Reed Solomon en sistemas de comunicaciones digitales sujetos a errores a ráfagas.
- Utilizar MATLAB® para implementar todo el proceso de conformación del mensaje, y su codificación, descodificación e interpretación de resultados.

4 Desarrollo

Los códigos Reed Solomon son códigos bloque lineales de tipo no binario. Así, con un código RS(n,k), a partir de un bloque de datos de entrada de k símbolos, se genera otro de salida de n símbolos que contiene a los k anteriores más $n-k$ símbolos de chequeo, y donde cada uno de todos esos símbolos contiene m bits, cumpliéndose además que:

$$n = 2^m - 1 \tag{1}$$

También pertenecen a la categoría de códigos cíclicos, por lo que cualquier permutación cíclica de una palabra código dará lugar a otra palabra código válida. Las palabras código se definen como elementos de un cuerpo finito $\mathbb{GF}(2^m)$, también llamado *Campo de Galois* (Wikipedia 2019).

4.1 Ratio y capacidad de corrección de los códigos RS

Como a partir de un bloque de k símbolos se genera uno nuevo de n símbolos, se dice que el ratio del código es k/n. El ratio da una idea de la redundancia del código: cuanto mayor sea (el límite es 1), menos símbolos de chequeo se añaden. En general, no son códigos de gran redundancia y, sin embargo, su capacidad de corrección de errores es bastante grande.

Así, como la distancia mínima entre dos palabras código válidas (definida como el número de símbolos en los cuales ambas secuencias difieren) es (Sklar 2001):

$$d_{\min} = 2t + 1 \tag{2}$$

siendo $2t = n - k$ el número de símbolos de chequeo añadidos (o redundantes), el código será capaz de corregir hasta t símbolos erróneos por cada bloque.

Hay que tener en cuenta que un símbolo es considerado erróneo tanto si sólo uno de sus bits lo es, como si lo son todos. También hay que señalar que una ráfaga de $m+1$ bits erróneos consecutivos puede corromper, como máximo, 2 símbolos dentro de un bloque.

Ejemplo 1 *Supongamos un código RS(255,223). Teniendo en cuenta la ecuación 1, se trabajaría con símbolos de* $m = \log_2(255 + 1) = 8$ *bits (es decir,* 1 *byte).*

Al codificador entrarían bloques de $k = 223$ *bytes y saldrían bloques de* $n = 255$ *bytes, por lo que la redundancia añadida sería de* $2t = 255 - 223 = 32$ *bytes por bloque.*

Teniendo en cuenta la ecuación 2, el descodificador sería capaz de corregir, como máximo, un total de $t = 16$ *bytes en cada bloque.*

Si los errores no ocurren en forma de ráfagas y están dispersados de forma más aleatoria, podrían corromper un mayor número de símbolos. Ante esta situación, los códigos RS son bastante vulnerables, por lo que es esencial asegurar que sean utilizados sólo para corregir ráfagas de errores de tamaño menor a su capacidad de corrección. Si es necesario, se deberán habilitar otros códigos adicionales, conjuntamente con técnicas de entrelazado y desentrelazado, que permitan combatir simultáneamente errores aislados y a ráfagas largas.

En la elección de un determinado código RS, se ha de tener en cuenta el siguiente compromiso:

1. Al disminuir el ratio k/n, la redundancia y la complejidad de cálculo aumentan.
2. Al aumentar el ratio, también aumenta la tasa de error finalmente obtenida.

Ejemplo 2 *El código RS(255,223) presenta bastante más redundancia y más complejidad computacional que el RS(255,239), que tiene un ratio mayor. Sin embargo, su capacidad de corrección es mucho más grande.*

Compruébalo tú mismo siguiendo los cálculos vistos en el ejemplo 1.

4.2 Codificación y descodificación RS con MATLAB®

El comando que permite en MATLAB® codificar con Reed Solomon es `code = rsenc(msg,n,k)`, donde `n` es el número de símbolos del bloque de salida, `k` el de entrada y `msg` representa el mensaje a codificar, que ha de estar conformado como un array de `k` columnas conteniendo elementos de un campo de Galois (*Galois field array*). La salida `code` también se obtendrá en formato similar, pero conteniendo `n` columnas.

Para generar los arrays con los mensajes, se utiliza el comando `x_gf = gf(x,m)`, donde `m` es el número de bits por símbolo, `x` es una matriz conteniendo el mensaje con los símbolos en formato decimal, y la salida `x_gf` es un array de elementos de $\mathbb{GF}(2^m)$.

La mejor forma de ver el funcionamiento es con un ejemplo, por lo que trabajaremos con uno similar al que se sugiere en la ayuda de MATLAB® (MathWorks 2019):

Ejemplo 3 *Se considerará un código RS que trabaje con bloques de mensajes de $k = 3$ símbolos, cada uno de ellos de $m = 3$ bits. Por tanto, a la salida se obtendrán bloques de mensajes de $n = 2^m - 1 = 7$ símbolos.*

Puede comprobarse que la distancia mínima de este código RS(7,3) será $d_{min} = n - k + 1 = 5$ símbolos, por lo que, como $t = d_{min}/2$, el código sólo será capaz de corregir hasta 2 símbolos por bloque (ya que no se pueden corregir fragmentos de símbolo).

Vamos a suponer que el mensaje binario que queremos codificar es el siguiente: [101 010 011 000 001 111 011 110 001]. Por conveniencia, se han agrupado los bits de 3 en 3 (ya que $m = 3$), así que el siguiente paso es ponerlo en formato decimal: [5 2 3 0 1 7 3 6 1].

Y como se va a trabajar con bloques de entrada de $k = 3$ símbolos, lo conformamos como una matriz de tres columnas, es decir, con cada fila conteniendo un bloque de entrada al codificador RS:

$$\text{mensaje} = \begin{bmatrix} 5 & 2 & 3 \\ 0 & 1 & 7 \\ 3 & 6 & 1 \end{bmatrix} \tag{3}$$

Así, para codificar en MATLAB® dicho mensaje con el código RS(7,3) del ejemplo 3, se utilizará la siguiente secuencia de comandos:

```
k=3;      % longitud de los bloques de entrada, en símbolos
m=3;      % número de bits por símbolo
n=2^m-1;  % longitud de los bloques codificados, en símbolos

% El mensaje se definen sobre un GF(2^m)
% La matriz se conforma con k columnas
% Cada elemento puede tomar valores entre 0 y 2^m-1
msg = gf([5 2 3; 0 1 7; 3 6 1],m);

% mensaje codificado:
codedMsg = rsenc(msg,n,k)
```

Podemos comprobar que a la salida se obtiene un Galois field array de 3 filas (una por cada bloque de salida) y 7 columnas (que es el número de símbolos por bloque de salida), con los siguientes elementos:

$$\text{codedMsg} = \begin{bmatrix} 5 & 2 & 3 & 5 & 4 & 4 & 2 \\ 0 & 1 & 7 & 6 & 6 & 0 & 7 \\ 3 & 6 & 1 & 7 & 4 & 0 & 2 \end{bmatrix} \tag{4}$$

Para verificar las propiedades correctoras del código, vamos a simular un ruido que provoque dos errores en dos símbolos consecutivos del primer bloque, dos errores separados en el segundo bloque y tres errores aislados en el tercer bloque. Para ello generaremos la siguiente matriz de errores:

$$\text{noise} = \begin{bmatrix} 0 & 0 & 0 & 2 & 3 & 0 & 0 \\ 6 & 0 & 1 & 0 & 0 & 0 & 0 \\ 5 & 0 & 6 & 0 & 0 & 4 & 0 \end{bmatrix} \tag{5}$$

El array generado lo sumaremos al mensaje codificado de la ecuación 4. Y el resultado lo trataremos de descodificar con el comando `rsdec` (MathWorks 2019), tal y como se muestra en la siguiente secuencia de comandos:

```
% Galois field array con ruido:
noise=gf([0 0 0 2 3 0 0 ;6 0 1 0 0 0 0 ;5 0 6 0 0 4 0],m);

% mensaje recibido con ruido:
receivedMsg = noise+codedMsg;

% decodedMsg contiene el mensaje descodificado
% cnumerr contiene el número de errores corregidos por fila
% Si cnumerr(i) = -1 hay un error no recuperable en la fila i
[decodedMsg,cnumerr] = rsdec(receivedMsg,n,k)
```

Al ejecutar la anterior secuencia, el mensaje descodificado que se obtiene a la salida resulta ser:

$$\text{decodedMsg} = \begin{bmatrix} 5 & 2 & 3 \\ 0 & 1 & 7 \\ 6 & 6 & 7 \end{bmatrix} \tag{6}$$

Si lo comparamos con lo visto en la ecuación 3, comprobamos que se han podido corregir todos los errores, excepto los del último bloque (última fila), tal y como se indica también en el vector `cnumerr` que también se obtiene a la salida:

$$\text{cnumerr} = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix} \tag{7}$$

Este vector señala 2 errores corregidos en la primera fila, otros 2 en la segunda, y la presencia de errores irrecuperables en la tercera (indicado con un -1).

5 Cierre

A lo largo de este artículo docente, hemos visto las características principales de los códigos Reed Solomon, tan utilizados desde hace tiempo en multitud de sistemas de comunicaciones digitales.

Lo más importante es su capacidad para corregir errores a ráfagas, presentes en símbolos consecutivos de una secuencia. Esta capacidad será mayor o menor dependiendo de las características del código RS finalmente elegido.

Se han proporcionado las herramientas para poder implementar de manera básica este tipo de códigos en MATLAB®, por lo que, para comprobar que realmente has aprendido los objetivos marcados, ahora es el momento de que te pongas manos a la obra e intentes realizar tus propias pruebas, variando el número de bits por símbolo, el tamaño de los bloques de entrada, etc.

Bibliografía

MathWorks (2019). *Reed-Solomon decoder - MATLAB rsdec*. MathWorks España. [Internet; descargado 1-mayo-2019]. URL: https://es.mathworks.com/help/comm/ref/rsdec.html (vid. págs. 4, 6).

Sklar, Bernard (2001). *Digital Communications. Fundamental and Applications*. Prentice Hall. ISBN: 978-0-13-084788-1 (vid. pág. 3).

Wikipedia, Colaboradores de (2019). *Cuerpo finito*. Wikipedia, La enciclopedia libre. [Internet; descargado 1-mayo-2019]. URL: https://es.wikipedia.org/wiki/Cuerpo_finito (vid. pág. 3).