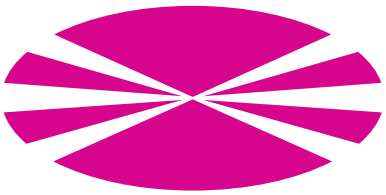

Grado en Ingeniería Informática

Software de Comunicaciones (SC)

Práctica 1

Segunda parte

Codificación de canal

Curso 2019-20

1. Sistema de transmisión codificado

El objetivo de esta segunda parte de la práctica es evaluar el rendimiento de distintos códigos de canal utilizando diferentes modulaciones sobre un canal AWGN, completando el sistema de comunicación desarrollado en la primera parte de la práctica.

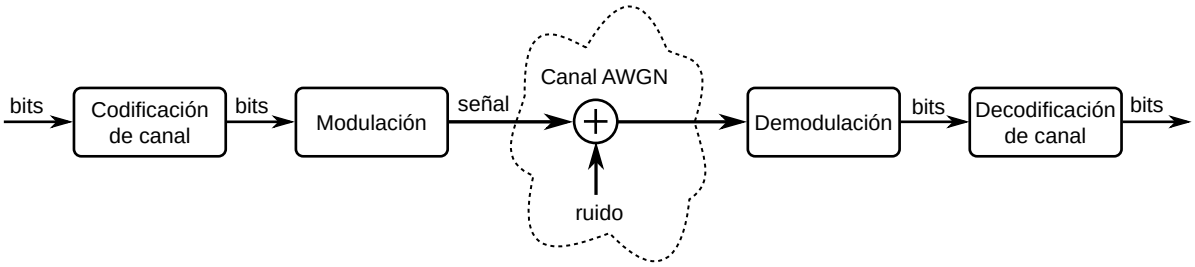

Figura 1: Sistema de comunicaciones codificado en canal AWGN.

Ahora los bits generados serán codificados por el codificador de canal antes de ser modulados. Los bits demodulados en el receptor deberán ser decodificados para intentar corregir los errores producidos por el canal y a continuación comparados con los bits fuente para evaluar el rendimiento del sistema.

En el canal AWGN el conjunto modulador-canal-demodulador es equivalente al modelo de canal BSC, con la particularidad de que la probabilidad de error p del canal BSC no se determina directamente, sino que dependerá del valor de E_b/N_0. Es importante tener en cuenta la tasa de codificación (k/n) a la hora de determinar el valor de E_b/N_0. Dado que se transmiten k bits de información por cada n bits codificados (que son a continuación modulados a símbolos de canal con energía media E_s), la energía de bit se puede calcular como

$$E_b = \frac{n}{k} \cdot \frac{E_s}{\log_2 M}.$$

Así, cuanto menor sea la tasa de codificación k/n, mayor será la energía utilizada para transmitir cada bit, y por lo tanto mayor será la varianza de ruido $N_0/2$ para un mismo valor de E_b/N_0 (es decir, los códigos de menor tasa se verán perjudicados).

1.1. Desarrollo

Una vez comprendido el modelo de comunicaciones se desarrollarán varios esquemas de codificación que permitan evaluar el rendimiento de los distintos tipos de códigos. Se debe partir del desarrollo de la primera parte de la práctica. Sobre este se deben hacer las siguientes modificaciones:

- Entre el paso 1 (generación de bits fuente) y el 2 (modulación) se debe introducir la codificación de canal. Esta dependerá del tipo de código empleado.
- En el paso 2 (modulación) se debe usar Gray Mapping.

- En el primer punto del paso 3 (generación de ruido) se debe tener en cuenta la tasa de codificación. El valor de N_0 se verá multiplicado por n/k para tener en cuenta la energía extra gastada al introducir la codificación.
- Entre el segundo y tercer punto del paso 3 se deben decodificar los bits demodulados. De nuevo el método dependerá del código empleado.

Los códigos a evaluar son los descritos en los siguientes apartados.

1.1.1. Código 1 - Hamming (Código bloque lineal)

En concreto, se debe usar el código de Hamming (7,4), de tal forma que las palabras código pueden ser obtenidas a partir de la siguiente matriz generadora **G**:

$$\mathbf{G}_{4\times 7} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Nota: la etapa de codificación se puede realizar con una sola instrucción haciendo un `reshape` de la secuencia de bits fuente. Es importante darse cuenta de que los parámetros de este código son k=4 y n=7; de esta forma, tenemos que agrupar los bits fuente en palabras de longitud k=4. De esta forma, el objetivo sería tener las palabras fuente en las filas de una matriz y multiplicar esa matriz por G. En ese caso, las palabras código se corresponderían con las filas de la matriz resultado de esa multiplicación.

CUIDADO!!! No confundir k (tamaño de la palabra fuente) con el parámetro k que se usaba en la práctica anterior y que correspondía al número de bits que se modulaban en cada símbolo.

Para la descodificación existen diversos métodos, pero en este caso seguiremos la siguiente estrategia:

1. Calcular el síndrome multiplicando los bits recibidos por $\mathbf{H}^T$. (De nuevo un `reshape` apropiado puede calcular todos los síndromes sin necesidad de iterar.)
2. Si el síndrome es igual a 0, extraer los bits fuente de la palabra código (nótese que las columnas 3, 5, 6 y 7 de **G** coinciden con las de una matriz identidad, por lo que dichas posiciones de la palabra codificada contendrán los bits fuente).
3. Si el síndrome es distinto de cero, determinar la posición del bit erróneo y corregirlo. Gracias a usar un código Hamming esta posición se puede extraer directamente de la conversión binario $\rightarrow$ decimal del síndrome.

Utilizando modulaciones QPSK y 16-QAM con Gray Mapping, comparar sobre la misma figura las curvas de BER frente a E_b/N_0 (en decibelios) de los sistemas sin decodificación con las de los sistemas codificados usando Hamming.

1.1.2. Código 2 - Convolucional

En este nuevo apartado se modificará el sistema de transmisión para hacer uso de códigos convolucionales.

- **Codificador**: para implementarlo de manera sencilla en Matlab se puede usar la representación mediante un trellis que nos da la función `poly2trellis`. A dicha función le pasaremos dos parámetros: la *constraint length*, que es el número de posiciones del registro (3 en este apartado), y un array de enteros que representen la estructura del codificador. Inicialmente se implementará el codificador representado en la Figura 2. El array de enteros que lo representa es [7 5 6]. Por ejemplo, el 7 de este array es "1 1 1" en binario, lo cual indica que el primer bit de salida se formará mediante la combinación de los valores de los tres registros. Una vez creado el trellis, la codificación requiere únicamente de utilizar la función `convenc`.

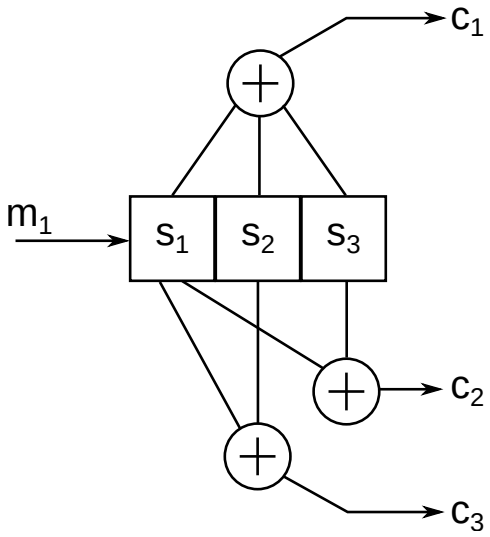

Figura 2: Codificador convolucional incial.

- **Descodificador**: se basará en la función de Matlab `vitdec` que implementa el algoritmo de Viterbi. Además de los datos a descodificar y el trellis, se indicarán los siguientes parámetros:

  - *Traceback depth*: viene a representar la memoria consumida por el descodificador de Viterbi. Asumiremos que su valor es cinco veces la *constraint length*. (Nota: si este valor es mayor que el número de bits a decodificar, la función devolverá un error).
  - *Opmode*: seleccionaremos el modo "trunc".
  - *Dectype*: indicaremos el modo de descodificación "hard".

Se pide:

1. Comparar una vez más sobre una misma figura las curvas de BER frente a E_b/N_0 para QPSK y 16-QAM con Gray Mapping del sistema con y sin codificación. ¿Mejora los resultados obtenidos con el código de Hamming?

2. Sustituir el código convolucional por el representado en la Figura 3. Obtener la curva de BER y compararla con la obtenida por el otro código. ¿Cómo influye el cambio realizado en el codificador en el rendimiento del sistema?

3. Diseñar un código convolucional que supere en ganancia de codificación a todos los anteriores. Para ello, puede ser muy interesante utilizar la funcion de Matlab `distspec`, que calcula la distancia libre de un código convolucional creado con `poly2trellis`.

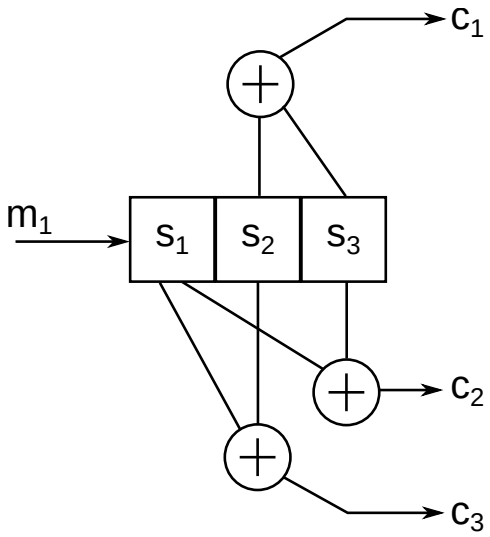

Figura 3: Segundo convolucional a implementar.

2. Defensa de la práctica

El plazo para la realización de esta práctica será hasta el 31 de Mayo de 2020 (incluido). Los alumnos deberán entregar todo su código en la tarea de Moodle correspondiente antes de las 23:55 del 31 de Mayo. La defensa de la práctica será individual y se hará de forma telemática mediante videoconferencia con Microsoft Teams.