Práctica 4. Simulación de un sistema básico de comunicaciones - II

1. Introducción

El objetivo de esta segunda práctica, continuación de la anterior, es completar el simulador añadiendo un sistema TCM. El objetivo del simulador es que sea genérico, de modo que se puedan comparar los resultados vistos en teoría con las simulaciones para distintas configuraciones.
Adicionalmente, se empleará el simulador para estudiar diferentes canales. La información de los canales no estará disponible a priori, así que en primer lugar habrá que identificar las características de cada canal. Cada tipo de canal tiene asociado un ancho de banda, un tipo de ruido o de distorsión que pueden ser descubiertos analizando la salida de la función de Matlab con distintas entradas.

2. Estudio previo

- En primer lugar, calcular la **ganancia con TCM** de la constelación de 8 símbolos propuesta en la parte 1 respecto a un 4-PSK sin codificar, usando el trellis del convolucional de cuatro estados visto en los ejemplos de clase. Se debe incluir la partición que se ha hecho de la constelación y la asignación de símbolos elegida.

- Identificación de canales "ciegos". La información de los canales a emplear no está disponible a priori, así que en primer lugar hay que identificar las características de cada canal. Cada tipo de canal tiene asociado un ancho de banda, un tipo de ruido o de distorsión que pueden ser descubiertos analizando la salida de la función de Matlab con distintas entradas.

Así pues, el estudio previo consiste en, usando Matlab, caracterizar (obtener SNR, ancho de banda y capacidad) de los siguientes dos canales, de los cuales se suministra su función en código **.p** (en Moodle):

canal1a.p y ***canal1b.p***

La frecuencia de muestreo de este canal es de 8 KHz así que aceptará entradas de señal x muestreadas con esa frecuencia o superior.

3. Trabajo de laboratorio

1. Realización práctica

En este apartado continuamos los fundamentos básicos de una simulación a la que en esta práctica añadiremos la posibilidad de simular un sistema TCM. Usaremos como ejemplo el sistema TCM-8PSK visto en clase.

Sistema TCM

El sistema TCM que usaremos en esta práctica se suministra ya implementado mediante un conjunto de funciones que describimos a continuación.

1. Abrir el fichero ***p2_1.m*** que contiene la función para la creación de la estructura que almacena la información necesaria para codificar y decodificar el código TCM:

   ```
   trellis = crear_trellis(G,nbits_sin_codificar)
   ```

   donde G es una matriz que contiene los vectores de conexión del convolucional y también debemos pasar el número de bits sin codificar. En el ejemplo usaremos el trellis generado por los siguientes generadores:

   ```
   G = [1 1 1; 1 0 1];      nbits_sin_codificar = 1;
   ```

   NOTA: para observar el correcto funcionamiento de la función que crea la estructura de trellis podemos usar la función:

   ```
   ver_trellis(trellis)
   ```

2. El siguiente paso es disponer de una constelación que realice la asignación de TCM. Para ello usamos la función:

   ```
   constelacion = constelacion_tcm_8psk();
   ```

3. A continuación, codificamos una secuencia de bits mediante el uso de la función:

   ```
   bits_cod = codificar_trellis(bits, trellis)
   ```

Una vez tenemos la secuencia de bits codificados podemos seguir los mismos pasos que en el simulador de la primera parte de la práctica, ya que disponemos de una secuencia de bits, aunque en este caso está codificada. Así, el siguiente paso sería la asignación a símbolos usando el sistema de asignación de la práctica anterior y el resto de funciones para conformar, modular, añadir ruido, demodular, filtrar con el filtro adaptado, muestrear y calcular las distancias a los símbolos de la constelación.

4. Por último, recuperamos la secuencia de bits utilizando el algoritmo de Viterbi. Para ello usamos la función:

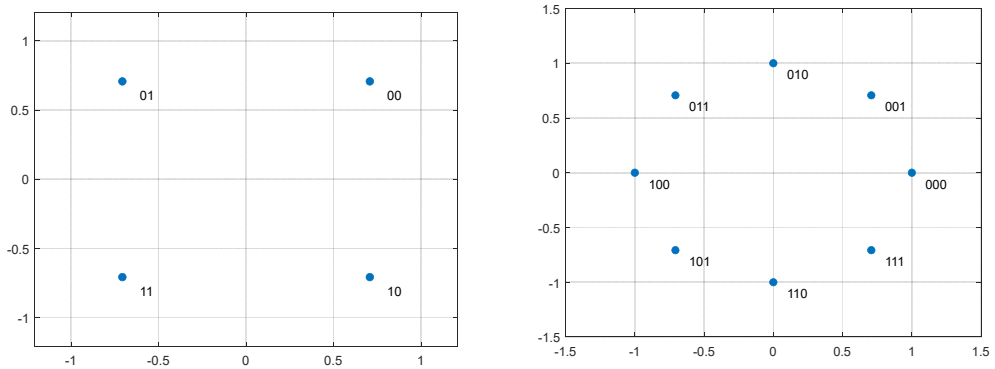
   ```
   bitsr = decodificar_trellis(trellis,D)
   ```

Compruebe que todo ha funcionado correctamente para este caso sin ruido y recuperamos exactamente los mismos bits.

Simulación sistema TCM con canal AWGN

5. Con el fichero **p2_1.m** y los de la primera parte de la práctica ya se pueden simular (con filtro Tx y Rx raíz cuadrada de coseno realzado) el sistema TCM, el de referencia 4PSK. Obtenga las curvas de simulación del sistema de referencia 4PSK, la de TCM-8PSK (la mejor posible) y la aproximación teórica a la probabilidad de error del sistema del TCM basada en la ganancia de código (4PSK+Gc). Simule un número de bits suficientemente grande como para que el resultado sea fiable.

$$P_b^{TCM-8PSK} \simeq Q\left(\sqrt{2\,G_c\,\frac{E_b}{N_0}}\right)$$



6. Obtenga la curva con la constelación TCM-8 propuesta en el estudio previo de la práctica anterior, la curva teórica aproximada con la ganancia de código que le corresponda y la de referencia 4PSK.

7. Estudio de ***'canal1b'*** del estudio previo.

El objetivo es enviar el **mayor número de bits posible** cumpliendo ciertos requisitos:

- ✓ Tiempo de transmisión acotado.
- ✓ Potencia acotada.
- ✓ Probabilidad de error máxima.

Para ello, hay que implementar los módulos necesarios para simular un sistema de comunicaciones (partes previas de la práctica) que permita la transmisión de la mayor cantidad posible de bits a través **dos** canales en los que se imponen ciertas restricciones.

- Restricciones **'canal1b'**: Tiempo de simulación: 100 seg.
  Potencia máxima: 1
  P_b máxima: 10^{-3}

Los canales son una función de Matlab con código oculto: **r = canal(x, fm)**

x: señal de entrada modulada
f_m: frecuencia de muestreo con la que se ha generado la señal
r: señal recibida en esa misma frecuencia de muestreo

Se suministra un ejemplo completo de funcionamiento para el canal, **p2_2.m**, que permite ver la mecánica de trabajo, aunque la solución no sea la óptima. En él, aparecen dos funciones que son de utilidad para comprobar que se cumplen los requisitos en todo momento:

verificar_x(x, fm, tiempo_maximo, potencia_maxima). Comprueba que la señal que se va a enviar cumple con el requisito de tiempo máximo en segundos y potencia máxima propuesta en cada situación (medida en la señal digital como `mean(x.^2)`.

verificar_error(bits, bitsr, Pb_max). Calcula la tasa de error de bit BER en la simulación y compara con el valor máximo objetivo para informar de si se cumple el requisito.

Se usan las funciones **.p** ya empleadas en la práctica anterior. Para garantizar que todos usamos la misma fuente, se suministra la función:

bits = fuente(numero_de_bits)

El trabajo consiste en la simulación de un sistema de comunicaciones que permita la transmisión de la mayor cantidad posible de bits a través canales en los que se imponen ciertas restricciones. Para conseguir dichas mejores prestraciones se pueden varias parámetros como: **frecuencia de portadora, factor de roll-off, número de taps del filtro conformador, uso de TCM, generadores del TCM, etc.**