TRANSMISIÓN DE DATOS 2006/07			
Práctica 6: Codificación Convolucional		**Grupo**	
		Puesto	
Apellidos, nombre		**Fecha**	
Apellidos, nombre		**18 Diciembre – 21 Diciembre**	

El objetivo de esta práctica es familiarizar al alumno con la codificación de canal mediante códigos convolucionales.

Para llevar a cabo la práctica, desarrolle cada ejercicio en un fichero MATLAB dentro del directorio P6.

Se proporcionan, accesibles desde la página de la asignatura, una serie de ficheros como guía para las funciones que se deben implementar. Todos estos ficheros están accesibles en el fichero TxDatosP6GXX.zip.

Además del código de las funciones que se piden en cada apartado se debe adjuntar los scripts que se desarrollen para poner a prueba dichas funciones y completar los ejercicios propuestos.

Justo antes de finalizar la práctica, comprima todo el directorio P6 en un fichero TxDatosP6GXX.zip (donde G indicará el grupo, A o B, y XX el puesto del laboratorio asignado), conéctese al sistema de entrega de prácticas de la Intranet y entréguelo en el grupo que corresponda (A o B). Guárdese adicionalmente una **copia personal**, para posibles futuras reutilización del código en prácticas posteriores.

1. Códigos Convolucionales

Ejercicio 1: Desarrollo de un codificador convolucional

Es este ejercicio se desarrollará un codificador convolucional no recursivo. Para ello deberá apoyarse en los ficheros suministrados en esta práctica y en los fundamentos explicados en clase de teoría.

Desarrollar una función con el siguiente interfaz:

```
function [code] = convEncoder(polGenerator,Sec)
%code:          código devuelto
%polGenerator:  matriz de 0's y 1's en la que cada fila es un polinomio
%                 generador
%Sec:           Secuencia a codificar.Cada fila un palabra mensaje
```

Nota: *Tenga en cuenta que al conocer los polinomios generadores y el número de columnas de la matriz Sec, somos capaces de inducir la memoria del codificador, L, la longitud de palabra código, n, y la longitud de la palabra mensaje, k.*

Ejercicio 2: Codificación Convolucional

Desarrolle un *script* al que llamará `codConvolucional.m`. Éste, mediante el uso de la función desarrollada en el apartado anterior, deberá codificar la secuencia contenida en el fichero `secuencia.txt`. Se utilizará para tal fin la siguiente matriz generadora:
`[1 1 1 1 1 1;1 0 1 0 1 0;1 0 0 0 1 0;1 1 0 0 0 1]`.

Nota: *Tenga en cuenta que a la salida habremos de obtener L-1 palabras código más que las palabras mensaje de la secuencia.*

Complete las siguientes tablas:

Mensaje	***Palabra código***

Codifique también, mediante este mismo *script*, la secuencia contenida en `secuencia2.txt`.

Mensaje	***Palabra código***

Comente los resultados:

Nota: *Busque analogías/diferencias entre las dos secuencias y entre sus codificaciones.*

Ejercicio 3: Desarrollo de un decodificador de *Viterbi*.

La decodificación óptima consistirá en encontrar el camino a través del diagrama de *Tellis* con la menor distancia a la secuencia recibida (`y`). Entendemos por distancia, la distancia *Hamming* (`distanciaHamming.m`).
Este cometido se podría cumplir analizando todas las posibles combinaciones de palabras código. Esto es, codificando cada una de ellas (`c`) para luego hallar sus distancias con la secuencia recibida (`y`). De esta forma, en caso de la ocurrencia de un error corregible, podríamos recuperar la secuencia original (sería la que menor distancia guardara con la recibida).
Siendo *N1* el número de palabras mensajes posibles y *N2* el número de palabras recibidas,
¿Cuántas secuencias habríamos de codificar para abordar el problema de la forma descrita?

N1^N2

El algoritmo de *Viterbi*, pese a su elevado coste computacional, supone una importante mejora con respecto a la solución descrita.

En el fichero `convDecoder.m` ofrecemos el boceto de una implementación de un decodificador *Viterbi* con 2 simplificaciones sobre el algoritmo original:

1) Solo es capaz de decodificar secuencias generadas con **L=2**, si L fuera distinto la función daría error.
2) Si en el proceso de decodificación encontramos 2 supervivientes con la misma distancia acumulada, tan solo nos quedaremos con uno de ellos.

Finalíce la implementación.

Siendo *N1* el número de palabras mensajes posibles, L el número de palabras almacenables en el *buffer* y *N2* el número de palabras recibidas, ¿Cuántas secuencias abríamos de codificar para abordar el problema de la forma descrita?

Ejercicio 4: Decodificación óptima por el algoritmo de *Viterbi*.

Desarrolle un *script* al que llamará `cod_deco_Convolucional.m`. Este, mediante el uso de la funciones desarrolladas en el apartado anterior, deberá codificar y decodificar la secuencia contenida en el fichero `secuencia3.txt`. Se utilizará para tal fin la siguiente matriz generadora:
`[0 0 1 1; 0 1 0 1; 0 1 1 1;1 0 0 0;1 0 1 1;1 1 1 1]`.

Mensaje	***Palabra código***	***Mensaje Decodificado***

Ahora haremos lo mismo, pero introduciendo una componente de ruido (`ruido.txt`) a las palabras código:

Mensaje	***Palabra código***	***Ruido***	***Codigo + Ruido***	***Mensaje Decodificado***

Comente los resultados:
