Universidad Rey Juan Carlos

ENUNCIADO PRÁCTICAS

Comunicaciones Digitales – Grado en Ingeniería en Sistemas de Telecomunicación

Descripción breve

Enunciados de las prácticas realizadas en la asignatura.
Lugar de depósito de este material: BURJC

Autores: Mihaela I. Chidean y Luis Bote Curiel

Curso 2024-2025

Índice de contenidos

Bloque I: Conceptos básicos 4
Práctica 1 — Codificación de fuente 4
Práctica 2 — Codificación de canal 9
Bloque II: Conceptos avanzados 14
Práctica 3 — Técnicas avanzadas de comunicaciones 14

Práctica 1. Codificación de Fuente

Curso 2024-2025

Parte 1. Codificación de Fuente sin Pérdidas

OBJETIVO:
Aprender a desarrollar códigos Huffman mediante MATLAB y a realizar una codificación y decodificación de fuente con dichos códigos.

Mejorar las habilidades de programación en MATLAB, siendo capaz de generar código/s optimizado/s, fácilmente generalizable/s y siempre que sea posible empleando cálculo matricial.

DESARROLLO:
Para realizar esta parte de la práctica, se recomienda repasar los conceptos de cuantificación vistos en las clases de teoría.

Asimismo, tendrá que utilizar las siguientes funciones nativas de MATLAB:

- `huffmandict`
- `huffmanenco`
- `huffmandeco`

Hito 1. Suponga una fuente que produce N símbolos con probabilidades p_i, $i = \{1,\ldots, N\}$. Diseñe una codificación binaria Huffman símbolo-a-símbolo empleando MATLAB.

¿Cuántas palabras son necesarias para codificar esta fuente con la codificación empleada?

Calcule la longitud media del código y la eficiencia de codificación.

Hito 2. Teniendo en cuenta la fuente del hito anterior, diseñe una codificación Huffman que opere sobre dos símbolos consecutivos.

¿Cuántas palabras son necesarias para codificar esta fuente con la codificación empleada?

Calcule la longitud media del código y la eficiencia de codificación.

Hito 3. Compare y comente los resultados obtenidos en los Hitos 1 y 2.

Hito 4. Suponga que en un intervalo de tiempo determinado la fuente produce la secuencia de símbolos `m1` definida como

```
m1 = randsrc(randi([95,125],1,1)*10,1,[1:N;p]);
```

¿Cuántos símbolos tiene esta secuencia?

Codifique esta secuencia empleando el código binario Huffman símbolo-a-símbolo diseñado.

¿Cuántos bits tiene la secuencia codificada?

Calcule el ratio de bits/símbolos obtenidos para esta secuencia de símbolos determinada.

Hito 5. Comente los resultados obtenidos en el Hito 4. Compárelos con obtenidos en el Hito 1.

Hito 6. Suponga que en un intervalo de tiempo determinado la fuente produce la secuencia de símbolos `m2` definida como

```
m2 = randsrc(num_simb,1,1:N);
```

Codifique esta secuencia empleando el código binario Huffman símbolo-a-símbolo diseñado.

¿Cuántos bits tiene la secuencia codificada?

Calcule el ratio de bits/símbolos obtenidos para esta secuencia de símbolos determinada.

Hito 7. Compare y comente los resultados obtenidos en los Hitos 4 y 6.

Hito 8. Considere que se modifica el diccionario obtenido en el Hito 1 según siguientes instrucciones:

```
aux = DICT{1,2};
DICT{1,2} = DICT{2,2};
DICT{2,2} = aux;
```

Realice la decodificación de la secuencia binaria obtenida en el Hito 4 con este nuevo diccionario.

Calcule el número de símbolos decodificados de manera incorrecta.

Hito 9. Comente los resultados obtenidos en el Hito 8.

CRITERIOS DE EVALUACIÓN:

Recuerda que las prácticas se evalúan mediante un examen. En dicho examen se te puede pedir que implementes código nuevo, nuevos y diferentes ejercicios relacionados con la codificación/ decodificación Huffman, comentar y comparar de forma justificada diferentes resultados, etc.. Por tanto, es importante que entiendas de verdad lo que has implementado en esta práctica. Además, es importante que dediques tiempo a aprender a implementar scripts y funciones de Matlab aprovechando lo máximo posible el cálculo matricial que proporciona Matlab.

A la hora de evaluar las preguntas de dicho examen se tendrán en cuenta:

- los resultados numéricos obtenidos,
- el código programado y sus características,
- calidad/detalle incluido en las comparaciones/justificaciones.

Se penalizará el uso incorrecto de unidades y las faltas gramaticales y de ortografía.

Parte 2. Conversión A/D - PCM y DPCM

OBJETIVO:
Visualizar (y comprender) el proceso de cuantificación que se realiza en la conversión analógico/digital, en concreto el método basado en PCM. Se trabajará también con una codificación diferencial para estudiar el método basado en DPCM.

Mejorar las habilidades de programación en MATLAB, siendo capaz de generar código/s optimizado/s, fácilmente generalizable/s y siempre que sea posible empleando cálculo matricial.

DESARROLLO:
Para realizar esta parte de la práctica, se recomienda repasar los conceptos de cuantificación vistos en las clases de teoría.

Asimismo, tendrá que utilizar las siguientes funciones nativas de MATLAB:

- `quantiz`

Considere un sistema de comunicaciones digitales. La fuente de información tiene a la salida la señal sig_ent que se define según el siguiente código.

```
% Señal original
t = [0:0.005:1];
sig_ent = 2*sin(2*pi*randi([2,4],1,1)*t)
      - cos(2*pi*randi([6,8],1,1)*t);
```

Hito 1. Describa la fuente en este problema.

¿Qué tipo de señal es? ¿Está o no está muestreada?

Si está muestreada, ¿cuántas muestras tiene?

Si no está muestreada, ¿qué proceso habría que realizar para muestrearla y cuántos pasos serían necesarios para realizar el muestreo?

Hito 2. Defina los parámetros de entrada para que representen un cuantificador que utiliza un total de 8 niveles y que el márgen dinámico es de $X_{sc} = 3V$.

Se recomienda que programe este hito de la forma más dinámica posible, es decir, utilice siempre que es posible las fórmulas matemáticas que relacionan unos parámetros con otros.

Hito 3. Cuantifique la señal `sig_ent` empleando el cuantificador con los parámetros definidos en el Hito 2. Utilice la función `quantiz` para ello. Guarde el resultado en la variable `sig_sal`.

Hito 4. Represente en una misma gráfica `sig_ent` y `sig_sal` obtenida en el Hito 3.

Dicha gráfica ha de tener los ejes bien identificados, así como un título y una leyenda explicativa.

Comente la gráfica.

Hito 5. Represente en una misma gráfica `sig_ent` y la señal cuantificada para un cuantificador con margen dinámico $X_{sc} = 3V$ y variando el número de bits (con valores $B \in \{2,3,4,8\}$).

Todas las gráficas han de tener los ejes bien identificados, así como un título y una leyenda explicativa.

Hito 6. Comente y compare las cuatro gráficas obtenidas en el Hito 5.

Hito 7. Represente en una misma gráfica `sig_ent` y de la señal cuantificada para un cuantificador con $B = 4$ bits para cada nivel y variando el margen dinámico (con valores $X_{sc} \in \{2,3,4,8\}$).

Todas las gráficas han de tener los ejes bien identificados, así como un título y una leyenda explicativa.

Hito 8. Comente y compare las cuatro gráficas obtenidas en el Hito 7.

Hito 9. Calcule para todos los casos de los hitos 5 y 7 el error cuadrático medio de cuantificación.

¿Qué combinación de B y X_{sc} tiene el mínimo error cuadrático medio de cuantificación para el Hito 5?

¿Y para el Hito 7? ¿Qué valor tienen estos errores? Comente los resultados obtenidos.

A continuación, se decide cuantificar la señal `sig_ent` siguiendo una codificación diferencial previa. La señal de entrada al cuantificador será `sig_diferencial` y se calcula de la siguiente manera:

```
% Codificación Diferencial
sig_menos1 = [0, sig_ent(1:end - 1)];
sig_diferencial = sig_ent - sig_menos1;
sig_diferencial = sig_diferencial(2:end);

aux = max(sig_diferencial);
```

Hito 10. Describa la señal `sig_diferencial`. ¿Qué tipo de señal es? ¿Qué rango de valores tiene?

Hito 11. Cuantifique la señal `sig_diferencial` empleando un cuantificador con margen dinámico $X_{sc} =$ `aux` y variando el número de bits (con valores $B \in \{2,3,4,8\}$).

En este caso, el resultado obtenido de la cuantificación no representa exactamente la señal original. Por tanto, para poder comparar de forma gráfica `sig_diferencial` con su versión cuantificada es necesario realizar la decodificación diferencial. A continuación se muestra el código necesario:

```
% Decodificación Diferencial
sig_decod = zeros(1,N);
sig_decod(1) = -1;
for index_sig = 2:N
      sig_decod(index_sig) = sig_decod(index_sig - 1)
            + sig_sal(index_sig - 1);
end
```

Hito 12. Represente en una misma gráfica la señal `sig_diferencial` y `sig_decod` para cada uno de los cuatro escenarios del Hito 11.

Todas las gráficas han de tener los ejes bien identificados, así como un título y una leyenda explicativa. Comente y compare las cuatro gráficas obtenidas.

Hito 13. Cuantifique la señal `sig_diferencial` empleando un cuantificador con $B = 4$ bits para cada nivel y variando el margen dinámico $X_{sc} \in \{0.75 \cdot \text{aux}, \text{aux}, 1.25 \cdot \text{aux}, 2 \cdot \text{aux}\}$.

Hito 14. Represente en una misma gráfica la señal `sig_diferencial` y `sig_decod` para cada uno de los cuatro escenarios del Hito 13.

Todas las gráficas han de tener los ejes bien identificados, así como un título y una leyenda explicativa. Comente y compare las cuatro gráficas obtenidas.

Hito 15. Calcule para todos los casos de los hitos 11 y 13 el error cuadrático medio de cuantificación.

¿Qué valor tienen estos errores? Comente los resultados obtenidos.

Hito 16. Compare los resultados obtenidos con PCM y DPCM.

CRITERIOS DE EVALUACIÓN:

Recuerda que las prácticas se evalúan mediante un examen. En dicho examen se te puede pedir que implementes código nuevo, nuevos y diferentes ejercicios relacionados con cuantificación, comentar y comparar de forma justificada diferentes resultados, etc.. Por tanto, es importante que entiendas de verdad lo que has implementado en esta práctica. Además, es importante que dediques tiempo a aprender a implementar scripts y funciones de Matlab aprovechando lo máximo posible el cálculo matricial que proporciona Matlab.

A la hora de evaluar las preguntas de dicho examen se tendrán en cuenta:

- los resultados numéricos obtenidos,
- el código programado y sus características,
- calidad/detalle incluido en las comparaciones/justificaciones.

Se penalizará el uso incorrecto de unidades y las faltas gramaticales y de ortografía.

Práctica 2. Codificación de Canal

Curso 2024-2025

Parte 1. Codificación de Canal por Bloques Lineal

OBJETIVO:
Aprender a desarrollar códigos Hamming mediante MATLAB y a realizar una codificación y decodificación de canal con dichos códigos y con otros tipos distintos.

Mejorar las habilidades de programación en MATLAB, siendo capaz de generar código/s optimizado/s, fácilmente generalizable/s y siempre que sea posible empleando cálculo matricial.

DESARROLLO:
Para realizar esta parte de la práctica, se recomienda repasar los conceptos de cuantificación vistos en las clases de teoría.

Asimismo, tendrá que utilizar las siguientes funciones nativas de MATLAB:

- `hammgen`
- `syndtable`
- `encode`
- `decode`

Considere que un sistema de comunicaciones digitales dado emplea un código de canal por bloques lineal para transmitir los bits de información almacenados en el vector `Bloque_Info`. En concreto, se empleará un código de Hamming con 3 bits de redundancia. Asimismo, considere que la transmisión de la palabra código correspondiente se ve afectada por el vector de error llamado `Error`.

Hito 1. Determine los valores de n y de k y defina el código mediante las matrices $\mathbf{G}$ y $\mathbf{H}$ utilizando la función `hammgen`.

Hito 2. Calcule la palabra código recibida utilizando la tabla de síndromes asociada a la matriz $\mathbf{H}$.

¿Se llegan a recuperar los bits de información original?

En caso negativo, ¿qué habría que hacer para, a partir de la variable "`Bloque_Rx_Correc`", recuperar dichos bits de información?

¿Por qué es necesaria la instrucción `rem` en este tipo de cálculos?

Hito 3. Determine la palabra código transmitida utilizando la función `encode` los bits de información recibidos utilizando la función `decode`, empleando en ambos casos la versión que NO hace uso de la matriz $\mathbf{G}$ o $\mathbf{H}$.

¿Cuál es la principal diferencia con respecto a la codificación/decodificación realizadas en los hitos anteriores?

¿Cuál es la principal diferencia entre el escenario de este hito y el escenario del Hito 2?

¿Se llegan a recuperar los bits de información original en este hito? ¿Por qué?

Hito 4. Determine la palabra código transmitida utilizando la función `encode` los bits de información recibidos utilizando la función decode, empleando en ambos casos la versión que SI hace uso de la matriz **G** o **H**.

¿Cuál es la principal diferencia con respecto a la codificación/decodificación realizadas en los hitos anteriores?

¿Cuál de las alternativas es más versátil?

¿Qué otras opciones de codificación/decodificación tienen disponibles las funciones `encode` y `decode`?

¿Qué diferencias/similitudes hay en cuanto a los resultados obtenidos en los Hitos 1 a 3?

¿Por qué en algunos casos sí se recuperan los bits de información y en otros casos no? ¿Por qué?

A continuación, considere los siguientes tres vectores de error:

```
Error_5{1} = [0 0 0 0 0 0 0];
Error_5{2} = [0 0 1 0 0 0 1];
Error_5{3} = [0 1 0 1 0 1 0];
```

Hito 5. Compare los bits de información recibidos.

¿En qué casos es el receptor capaz de detectar el error en el canal?

¿En qué casos es el receptor capaz de corregir correctamente el error ocurrido en el canal?

¿Qué ocurre cuando detecta un error, tiene certeza de que lo corrige correctamente? ¿Por qué?

Considere a continuación que el sistema de comunicaciones digitales modifica el código de canal por uno cuya matriz generadora es:

```
G = [1 1 1];
```

Hito 6. Determine la secuencia de bits transmitida utilizando la alternativa de programación que más adecuada considere.

¿Qué tipo de codificación se está realizando?

¿Cuántos bits (en total) tendrá la secuencia de bits transmitida? ¿Se llegan a recuperar todos los bits de información original en este hito? ¿Por qué?

CRITERIOS DE EVALUACIÓN:

Recuerda que las prácticas se evalúan mediante un examen. En dicho examen se te puede pedir que implementes código nuevo, nuevos y diferentes ejercicios relacionados con la codificación bloque lineal, comentar y comparar de forma justificada diferentes resultados, etc.. Por tanto, es importante que entiendas de verdad lo que has implementado en esta práctica. Además, es importante que dediques tiempo a aprender a implementar scripts y funciones de Matlab aprovechando lo máximo posible el cálculo matricial que proporciona Matlab.

A la hora de evaluar las preguntas de dicho examen se tendrán en cuenta:

- los resultados numéricos obtenidos,
- el código programado y sus características,
- calidad/detalle incluido en las comparaciones/justificaciones.

Se penalizará el uso incorrecto de unidades y las faltas gramaticales y de ortografía.

Parte 2. Simulación de un Sistema Paso Banda con Codificación de Canal y utilizando el Equivalente Paso Bajo Discreto

OBJETIVO:

Visualizar (y comprender) la Ganancia de Codificación obtenida mediante la utilización de la codificación de canal.

Mejorar las habilidades de programación en MATLAB, siendo capaz de generar código/s optimizado/s, fácilmente generalizable/s y siempre que sea posible empleando cálculo matricial.

DESARROLLO:

Para realizar esta parte de la práctica, se recomienda repasar los conceptos de codificación de canal vistos en las clases de teoría, así como diversos conceptos vistos en la asignatura de Teoría de la Comunicación.

A la hora de simular un sistema de comunicaciones, no es necesario emular la transmisión en tiempo continuo. Para simplificar los cálculos, se puede recurrir al concepto de constelación y al hecho de que el ruido se puede proyectar sobre la misma base que la señal, de forma que ambos puedan ser representados sobre un número limitado de ejes, que conforman un espacio vectorial.

En nuestro caso, utilizaremos una base de dos componentes, por lo que dicho espacio vectorial es de dos dimensiones. Por tanto, podemos ver la señal transmitida y la recibida como un array de símbolos complejos (un símbolo complejo representará la fase y la amplitud que la portadora lleva en cada momento). Como recordará, cualquier señal paso banda se podía representar con su equivalente paso bajo, que es precisamente lo que estamos utilizando aquí. Además, este equivalente es discreto porque cada símbolo transmitido se corresponde con un número complejo, por lo que la señal transmitida será una secuencia de números complejos. Obviamente, es posible proyectar el ruido sobre este espacio, por lo que el ruido también será una secuencia de números complejos.

Observe el código proporcionado en el fichero auxiliar. Lea cada línea e intente comprender qué función realiza. Tenga en cuenta que dicho script incluye una función auxiliar llamada `calculaBER_caso_concreto` que se emplea para minimizar la cantidad de líneas totales, dado que esta parte de código se ejecuta múltiples veces en la simulación.

Observe que, al comienzo del script, se define el modulador (indicando el tipo de modulación, el orden, la utilización de una codificación Gray y el tipo de entrada al modulador). En recepción se utilizará el correspondiente demodulador. Las instrucciones que definen el modulador y el demodulador son:

```
Modulador = comm.PSKModulator(M,0,'BitInput',true);
Demodulador = comm.PSKDemodulator(M,0,'BitOutput',true);
```

Observe también que la SNR deseada la obtenemos haciendo que la potencia del ruido sea inversamente proporcional a dicha SNR en unidades naturales, ya que la potencia de señal es 1 en este caso (es conveniente comprobar la potencia de los símbolos transmitidos porque puede ser distinta de 1 según el tipo de modulación utilizada). Las instrucciones utilizadas para generar el ruido son:

```
% Generación del ruido
snr = 10^(SNR/10);
n = (randn(N,1) + j.*randn(N,1)) * (1/sqrt(2)) * sqrt(1/snr);
```

Para realizar la simulación del sistema de comunicaciones deberá trabajar con un número de bits a transmitir (modelado con la variable `Nb`) y realizar las representaciones para SNR de 0dB a 10dB (en pasos de 1dB). Debe modificar el fichero auxiliar para incluir además las codificaciones de los diferentes escenarios considerados:

- Modulación: (a) BPSK y (b) QPSK
- Codificación de canal: (a) Sin aplicar codificación de canal; (b) Aplicando un código de repetición (3,1); (c) Aplicando un código Hamming (7,4).

Hito 1. Añada el código necesario para incluir los parámetros que configuran las tres opciones disponibles para la codificación de canal.

Hito 2. Añada el código necesario para incluir las instrucciones que realizan la codificación y decodificación de canal.

Hito 3. Añada el código necesario para calcular la BER.

Hito 4. Represente, en una misma figura, las curvas de BER frente a la SNR utilizando la modulación BPSK y las tres opciones disponibles para la codificación de canal. Incluya título, ejes y leyenda representativa en dicha figura.

Hito 5. Calcule la SNR mínima para obtener una BER $\leq 10^{-3}$.

Deberá programa este cálculo para que sea automático a partir de los datos y resultados disponibles.

Tenga en cuenta obtendrá unos resultados dependientes de la precisión del `vector_SNR`.

Compare el valor obtenido mediante este cálculo automático con el valor aproximado que obtendría a partir de la gráfica.

Hito 6. Determine la Ganancia de Codificación.

Asegúrese de que entiende perfectamente cuál es el proceso realizado para calcular dicha ganancia.

¿Le parecen razonables los valores obtenidos?

Hito 7. Represente, en una misma figura, las curvas de BER frente a la SNR utilizando la modulación QPSK y las tres opciones disponibles para la codificación de canal.

Incluya título, ejes y leyenda representativa en dicha figura.

Hito 9. Repita los hitos 5 y 6 para el caso de la modulación QPSK.

Hito 10. Compare los resultados obtenidos en los Hitos 5, 6 y 9.

¿Deberían ser iguales las ganancias de codificación obtenidas?

Hito 11. ¿Cuáles son las ventajas e inconvenientes de las codificaciones de canal empleadas?

CRITERIOS DE EVALUACIÓN:

Recuerda que las prácticas se evalúan mediante un examen. En dicho examen se te puede pedir que implementes código nuevo, nuevos y diferentes ejercicios relacionados con la codificación de canal y simulación de sistemas de comunicación, comentar y comparar de forma justificada diferentes resultados, etc.. Por tanto, es importante que entiendas de verdad lo que has implementado en esta práctica. Además, es importante que dediques tiempo a aprender a implementar scripts y funciones de Matlab aprovechando lo máximo posible el cálculo matricial que proporciona Matlab.

A la hora de evaluar las preguntas de dicho examen se tendrán en cuenta:

- los resultados numéricos obtenidos,
- el código programado y sus características,
- calidad/detalle incluido en las comparaciones/justificaciones.

Se penalizará el uso incorrecto de unidades y las faltas gramaticales y de ortografía.

Universidad Rey Juan Carlos

Práctica 3. Técnicas Avanzadas de Comunicaciones

Curso 2023-2024

Parte 1. OFDM

Objetivo:

Visualizar (y comprender) los bloques de un sistema OFDM implementado en MATLAB.

Desarrollo:

Para realizar esta parte de la práctica, dispone de la función `OFDM_Pe` definida en el propio *script* .

Para los siguientes apartados, cuando lo considere oportuno, puede variar el número de bits a transmitir (indicando el valor que ha escogido) y, cuando deba representar resultados variando la SNR, realizar las representaciones para SNR de 0dB a 20dB (en pasos de 5dB).

Hito 1.1. Comente los pasos que se realizan en la función `OFDM_Pe` y responda a las siguientes cuestiones:

- ¿Qué parámetros toma como entrada?
- ¿Qué parámetros ofrece como salida?

Hito 1.2. Represente la curva de BER frente a SNR para una OFDM con 16 portadoras y un prefijo cíclico de 4, variando la SNR. Comente los resultados.

Hito 1.3. Cree una nueva función `OFDM_Pe2` (al final del script) basada en la función `OFDM_Pe` que implemente una modulación 8PSK. Represente, en una misma figura, las curvas de BER frente a la SNR del apartado anterior y de éste. Comente los resultados.

Criterios de evaluación:
Recuerda que las prácticas se evalúan mediante un examen. En dicho examen se te puede pedir que implementes código nuevo, nuevos y diferentes ejercicios relacionados con OFDM, comentar y comparar de forma justificada diferentes resultados, etc. Por tanto, es importante que entiendas de verdad lo que has implementado en esta práctica. Además, es importante que dediques tiempo a aprender a implementar *scripts* y funciones de Matlab aprovechando lo máximo posible el cálculo matricial que proporciona Matlab.

A la hora de evaluar las preguntas de dicho examen se tendrán en cuenta:
- los resultados numéricos obtenidos,
- el código programado y sus características,
- calidad/detalle incluido en las comparaciones/justificaciones.

Se penalizará el uso incorrecto de unidades y las faltas gramaticales y de ortografía.

Parte 2. Espectro Ensanchado

Objetivo:

Visualizar (y comprender) la técnica de espectro ensanchado demostrando su efectividad para suprimir interferencias de tipo sinusoidal.

Desarrollo:

Para realizar esta parte de la práctica, dispone de la función `SS_Pe` definida en el propio script .

Para los siguientes apartados, cuando lo considere oportuno, puede variar el número de bits a transmitir (indicando el valor que ha escogido) y, cuando deba representar resultados variando la SNR, realizar las representaciones para SNR de 0dB a 10dB (en pasos de 1dB).

Hito 2.1. Comente los pasos que se realizan en la función `SS_Pe` y responda a las siguientes cuestiones:

- ¿Qué parámetros toma como entrada?
- ¿Qué parámetros ofrece como salida?
- ¿Qué tipo de técnica de espectro ensanchado se está realizando?
- ¿Cuál es la instrucción en la que se realiza el ensanchado de la señal a transmitir?
- ¿Cuáles son las instrucciones en la que se obtiene la correlación de la señal recibida con el código de ensanchado?

Hito 2.2. Represente, en una misma figura, las señales transmitidas y las recibidas para un bit con los siguientes valores de los parámetros: `snr_in_dB = 8`, `Lc = 20`, `w0 = 1` y diferentes amplitudes de la señal interferente A = 0, 1, 2, y 3. Obtenga cuatro figuras, una para cada valor de A, y comente los resultados.

Hito 2.3. Represente, en una misma figura, las curvas de BER frente a la SNR utilizando diferentes amplitudes de la señal interferente A = 0, 1, 2, y 3. Comente los resultados.

Criterios de evaluación:

Recuerda que las prácticas se evalúan mediante un examen. En dicho examen se te puede pedir que implementes código nuevo, nuevos y diferentes ejercicios relacionados con espectro ensanchado, comentar y comparar de forma justificada diferentes resultados, etc. Por tanto, es importante que entiendas de verdad lo que has implementado en esta práctica. Además, es importante que dediques tiempo a aprender a implementar *scripts* y funciones de Matlab aprovechando lo máximo posible el cálculo matricial que proporciona Matlab.

A la hora de evaluar las preguntas de dicho examen se tendrán en cuenta:

- los resultados numéricos obtenidos,
- el código programado y sus características,
- calidad/detalle incluido en las comparaciones/justificaciones.

Se penalizará el uso incorrecto de unidades y las faltas gramaticales y de ortografía.