PRÁCTICA 1
Señales, Cuantificación y Convolución

El objetivo de esta práctica del laboratorio es que el alumno se familiarice con **MATLAB**, aplicando sencillos comandos de procesado de señal. Los comandos necesarios vienen dados durante la práctica, para más ayuda hay que consultar los *tutorials* proporcionados y la ayuda que incluye **MATLAB**.

Antes de empezar:

1. Crear en la carpeta **Mis Documentos* un directorio de trabajo, llamado p.ej. **Laboratorio_PDS*** y en ello cinco subdirectorios correspondientes a las prácticas.
2. Copiar los ficheros **`trama.mat`**, **`hn_1.mat`** y **`v1.wav`** desde el directorio **`\Mi PC\Documentos compartidos\Laboratorio PDS\ Ficheros de prueba`** al directorio de la práctica 1.
3. Abrir **MATLAB.**
4. Cambiar el "C*urrent Directory*" al directorio de la practica 1.

A. Señales

1. Cargue la señal `trama` usando el comando `load`. Se trata de 250 muestras de señal de voz, muestreadas a 8 kHz con 16 bits y guardadas en formato `.mat` de **MATLAB**.

2. Muestre en pantalla el contenido de 10 muestras cualesquiera; ahora visualízelas con ayuda de la función "`stem`". Finalmente, compare el resultado de dibujar la señal completa con "`stem`" y "`plot`".

3. En lugar de pintar el eje x en muestras (en este caso, de 1 a 250), estamos interesados en verlo en tiempo (milisegundos). Recordando que la frecuencia de muestreo original era 8kHz, genere en un vector el eje de tiempos correspondiente (`eje_t`) y muéstrelos conjuntamente con:

   ```
   >> plot(eje_t, trama)
   ```

   Puede añadirle una etiqueta con `xlabel`, por ejemplo:

   ```
   >> xlabel('t (ms.)')
   ```

4. Cargue el fichero `v1.wav` (fichero de voz) con la función `wavread`, y muestre nuevamente con el eje x en tiempo su forma de onda gráficamente con `plot`, pero esta vez en segundos, sabiendo que está digitalizada de la misma forma que `trama`.

5. Para escuchar el contenido, use la función `sound` con la frecuencia de muestreo correcta.

B. Cuantificación

6. **MATLAB** trabaja internamente con precisión numérica muy elevada (64 bits), necesaria para realizar cálculos numéricos de forma fiable. A efectos de estudiar el fenómeno de la cuantificación, será equivalente a disponer de números reales (precisión infinita). Por tanto, cada vez que se carga una señal `.wav`, ésta queda representada con "precisión infinita" entre –1 y 1.

   Se puede simular perfectamente el proceso de cuantificación trabajando con números enteros, siendo el rango disponible el número de escalones de cuantificación. Así, si se

cuantifica con 8 bits, hay $2^8 = 256$ niveles disponibles, y por tanto es como trabajar entre –128 y 128 (exactamente sería entre –127 y 128, pero ya nos cuidaremos de no saturar).

Para realizar "físicamente" el proceso de cuantificación, o asignación al escalón de cuantificación más próximo, se usa la función `round`, que devuelve el entero más próximo. Así, la cuantificación con 8 bits de la información contenida en `v1` la se obtiene con

```
>> v1q = round(v1*128)/128;
```

7. Calcule, represente y escuche las señales cuantificadas con 8, 6, 4 y 2 bits y la señal error de cuantificación. ¿Puede deducir alguna relación entre las amplitudes de dicho error y los tamaños de los escalones de cuantificación?

8. A partir del teorema de Parseval, se puede calcular la relación señal a ruido en cada uno de los casos mediante:

$$\mathrm{SNR} = 10 * \log_{10} \frac{\sum_{n=0}^{L-1} (x[n])^2}{\sum_{n=0}^{L-1} (x[n] - \hat{x}[n])^2}$$

Implemente una función de **MATLAB** (`.m`) llamada `f_snr` que calcule dicho valor a partir de las señales original y cuantificada y que responda al siguiente prototipo:

```
function snr = f_snr(x, xq);
```

Consejo: el sumatorio puede implementarse con un "`for`", pero en **MATLAB** es mucho más eficiente "vectorizar" la operación mediante la función "`sum`".

9. Haciendo uso de la función anterior, calcule la SNR de cuantificación en cada uno de los 4 casos (N = 8, 6, 4 y 2). Compare el valor obtenido con la predicción teórica suponiendo distribución gaussiana de los datos. Compruebe la validez de dicha suposición mediante la función "`histfit`", la cual representa la distribución estadística experimental de los datos y ajusta sobre ellos una gaussiana (por ejemplo, `histfit(v1,100)`).

C. Convolución

10. Para realizar la convolución de una de las señales de voz con un filtro, nos inventamos una h[n] causal de 10 puntos decreciente de la siguiente forma:

```
>> h = [10 9 8 7 6 5 4 3 2 1]/10;
```

Ojo: el primer valor, `h[n]=1`, se corresponde con el "instante" n=0, pero en **MATLAB** ocupa la posición 1, es decir, h(1), ya que los vectores en **MATLAB** se definen de 1 a N (no de 0 a N-1).

11. Realice una función `.m` que haga la convolución de dos señales cualesquiera (en este caso de la señal de voz y `h`). Puede seguir el esquema dado en la página Q&Q.6 de las transparencias de clase, pero teniendo en cuenta ahora que los vectores son finitos y **MATLAB** indexa de 1 a N y no de 0 a N-1. Compruebe su función comparando el resultado con el que da la función `conv` de **MATLAB**.

12. Realice ahora la convolución con la respuesta impulsiva simulada de un recinto acústico en el fichero `hn_1.mat`. Escuche comparativamente ambas (original y convolucionada) y explique el efecto que se produce.