Señales y Sistemas: Lab 1

En esta práctica se va a diseñar un sistema de comunicaciones básico para transmitir información empleando señales audibles. En esta primera práctica se va a asumir que no existe ni ruido ni problemas de sincronización. Se pide realizar las siguientes actividades:

1. Fije una duración de $T = 5$ segundos para las señales y una frecuencia de muestreo f_s. Se recomienda que la frecuencia de muestreo coincida con una de las habituales para señales de audio. Por ejemplo: 8 kHz, 22,050 kHz o 44,1 kHz. Tenga en cuenta que, cuanto mayor sea la frecuencia de muestreo, más potencia de procesamiento requerirá. Su eje (discreto) de tiempos será, por tanto,

   ```
   t=0:1/fs:T
   ```

2. Construya una base ortonormal de n señales de energía finita (de duración T segundos), de manera que las señales de la base ortonormal sean audibles. Una posibilidad es pensar en tonos de distintas frecuencias. n debe ser al menos 200. Llamaremos $\phi_1(t), \ldots, \phi_n(t)$ a las señales de la base ortonormal.

   NOTA: Tenga cuidado con la relación entre la distribución frecuencial de las señales $\phi_1(t), \ldots, \phi_n(t)$ y la frecuencia de muestreo f_s.

3. Fuente: Genere un vector binario de longitud n. Estos n bits pueden ser generados aleatoriamente, o pueden provenir de una representación binaria de cierta información. Por ejemplo, el siguiente código convierte texto en un vector binario (8 bits por caracter):

   ```
   text='Hello world';
   data = reshape(dec2bin(text, 8).'-'0',1,[])';
   ```

4. Modulación: Convierta el vector de bits en un vector de ± 1. Por ejemplo, puede asignar $+1$ al valor binario '1' y asignar -1 al valor binario '0'. Llamaremos $\alpha_1, \ldots, \alpha_n$ a los elementos de este vector. Por último, genere la señal

   $$x(t) = \sum_{k=1}^{n} \alpha_k \phi_k(t)$$

5. Escuche la señal $x(t)$ y obtenga su espectrograma. Para ello, se recomienda emplear las funciones 'sound' y 'spectrogram' de Matlab.

6. Detector: A partir de la señal $x(t)$, extraiga las componentes $\alpha_1, \ldots, \alpha_n$ mediante integración numérica.

7. Demodulación: Vuelva a convertir el vector con ± 1 en un vector binario. Si el vector binario representa texto, puede recuperar el texto con la siguiente instrucción:

   ```
   text = char(bin2dec(reshape(char(data+'0'), 8,[]).'));
   ```

8. Modifique el valor de T y de n y extraiga conclusiones.

9. Cambie la modulación y demodulación: En lugar de convertir cada bit en un valor ± 1, agrupe los bits y modúlelos usando valores distintos. Por ejemplo, si toma grupos de 2 bits, puede realizar la siguiente conversión

bits	α_k
'00'	-3
'01'	-1
'10'	+1
'11'	+3