Periféricos y Dispositivos de Interfaz Humana
Grado en Ingeniería Informática

PRÁCTICA 3: MANEJO DE SONIDO EN MATLAB

Objetivos

I. Identificar y representar gráficamente la forma de onda de señales de sonido.

II. Conocer la estructura de un fichero típico de sonido (ficheros WAV).

III. Entender y operar con los parámetros principales de una señal de sonido.

IV. Utilizar el entorno de programación Matlab para el manejo de sonido.

Contenido

1. Introducción a Matlab

MATLAB es una herramienta de programación para realizar cálculos numéricos con vectores y matrices. Incluye, entre otros aspectos:

- Un lenguaje de programación propio, interpretado y orientado al cálculo matricial.
- Un entorno de programación para la ejecución de código fuente Matlab.
- Herramientas de desarrollo integradas: editor, depurador, ayuda, Simulink, etc.

Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones y su facilidad de aprendizaje. Es un software muy usado en universidades y centros de investigación y desarrollo. Existe un equivalente libre de Matlab compatible con la mayoría de funciones de éste denominado Octave. Dado que la Universidad de Granada posee licencias para el uso de Matlab y que ya se encuentra instalado en los laboratorios de prácticas, será este el que usemos.

La bibliografía y recursos para aprender Matlab es inmensa, partiendo de la propia ayuda del programa, que es muy completa. Por su síntesis y brevedad, se recomienda el seguimiento del manual gratuito "Aprenda Matlab 7.0 como si estuviera en primero" (Javier García de Jalón, José Ignacio Rodríguez, Jesús Vidal, ETS de Ingenieros Industriales, Universidad Politécnica de Madrid, 2005): http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab70/matlab70primero.pdf

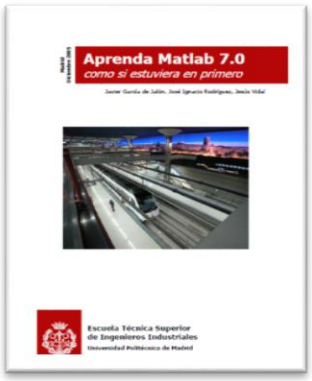


Figura 1. Portada de "Aprenda Matlab 7.0 como si estuviera en primero")

La estructura del IDE de Matlab es parecida a la que se muestra en la Figura 2, dependiendo de la versión de Matlab que se ejecute. Se pueden ejecutar instrucciones desde la ventana de comandos ("Command Window"), aunque lo más elegante es usar ficheros .m que contengan **scripts** y **funciones**. Para pedir ayudar a Matlab sobre cualquier función, se puede usar la orden "doc" seguida del nombre de la función a consultar. Como su propio nombre indica, Matlab (Laboratorio de Matrices) trata a todas las variables como si fueran una matriz. En Matlab no es necesario declarar las variables, se pueden asignar directamente. Por ejemplo: A=[1, 2; 3, 4] crea una matriz 2 x 2 con los valores especificados. Se usan comas o espacios para separar columnas y puntos y comas para separar filas.

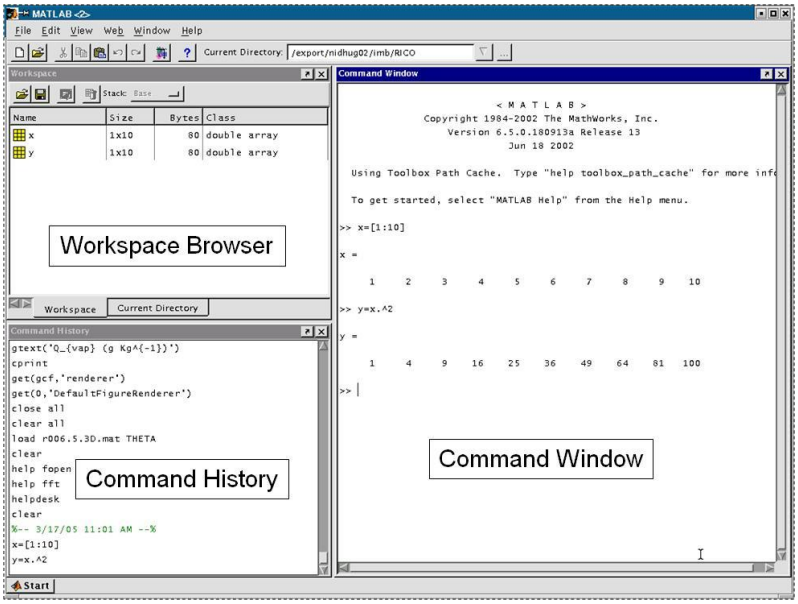


Figura 2. Aspecto habitual del IDE de Matlab 7.

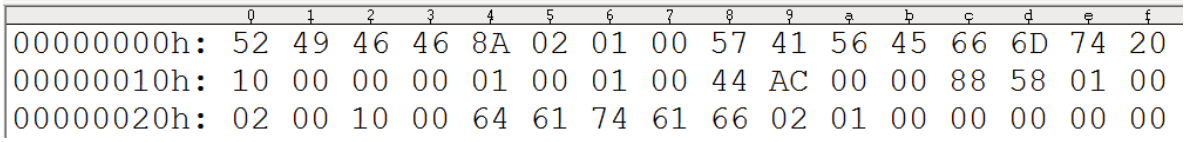
2. El formato de sonido WAV

El formato WAV o WAVE ("onda"), diseñado por Microsoft, es uno de los formatos de sonido más sencillos que existen. Consta básicamente de tres partes o bloques: el fichero comienza con un bloque de identificación, a continuación sigue un bloque que especifica los parámetros del formato y, por último, finaliza con el bloque que contiene las muestras.

La siguiente tabla refleja la estructura interna de un fichero WAV, indicando para cada uno de los bloques, los campos de que consta. El formato en que se almacenan los datos mayores que 1 bytes es "Little-endian" (primero el más pequeño). Por ejemplo, el dato binario de 16 bits H'0A10, correspondería como número entero a H'100A=D'4106.

Bloque	**Número de bytes**	**Descripción del campo**	**Contenido**
Identificación (12 bytes)	4	Caracteres de identificación	RIFF
	4	Longitud del resto del fichero	
	4	Caracteres de identificación	WAVE
Parámetros (24 bytes)	4	Caracteres de inicio del bloque de formato	fmt_
	4	Longitud del resto del bloque de formato	0x10 (para PCM)
	2	Formato de audio	0x01 (para PCM)
	2	Número de canales	0x01 (mono), 0x02 (estéreo)
	4	Frecuencia de muestreo en herzios	
	4	Bytes por segundo	(Frecuencia*Bytes_por_muestra)
	2	Bytes por muestra	(Num_canales*Resolución/8)
	2	Resolución en bits	(8 ó 16)
Muestras	4	Caracteres de inicio de bloque	data
	4	Número de bytes de muestras	
	resto	Bytes de muestras	

EJERCICIO 1: La siguiente captura de pantalla muestra el contenido en hexadecimal de un fichero WAV en sus primeros bytes. Identificar los bytes de cada campo y su valor decimal de la cabecera en este fichero.



```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 52 49 46 46 8A 02 01 00 57 41 56 45 66 6D 74 20
00000010h: 10 00 00 00 01 00 01 00 44 AC 00 00 88 58 01 00
00000020h: 02 00 10 00 64 61 74 61 66 02 01 00 00 00 00 00
```

CAMPO	BYTES	Valor decimal o caracteres
Caracteres de identificación (4 B)	52 49 46 46	R I F F
Longitud del resto del fichero (4 B)	…	…
…	…	…

3. Manejo de sonido en Matlab

Matlab trabaja siempre sobre variables de tipo vector o matriz. Un sonido digitalizado no es más que una sucesión de valores que contienen la amplitud del sonido original que se quiere reproducir. Por tanto, en Matlab se gestionaran los sonidos como matrices [m x n], donde m es el número de muestras y n el número de canales.

3.1. Leer fichero WAV: wavread y audioread.

`y=wavread('fichero')`: lee el archivo WAV cuyo nombre debe estar especificado dentro de los paréntesis. Esta función devuelve los datos muestreados en una matriz y. Los valores de la amplitud se escalan en el rango [-1,+1]. Sin embargo, en el fichero WAV original se almacenan como enteros con el número de bits que especifique la resolución. La función admite diferentes sintaxis para devolver otros valores aparte de las propias muestras del sonido. Más información tecleando `doc wavread` desde el intérprete de órdenes de Matlab. En las últimas versiones de Matlab, `wavread` ha sido sustituida por `audioread`.

3.2. Reproducir sonido: sound y audioplayer.

`sound(y,Fs,bits)`: reproduce el sonido contenido por la matriz y a la frecuencia de muestreo especificada en `Fs` y con la resolución especificada en `bits`. Por defecto, el valor de Fs si no se especifica es 8192 y la resolución es de 8 bits. Más información tecleando `doc sound` desde el intérprete de órdenes de Matlab.

También es posible utilizar la función `audioplayer`, cambiando ligeramente la sintaxis y el tipo de datos utilizado.

3.3. Reproducir sonido con escalado previo: soundsc.

`soundsc(y,Fs,bits)`: funciona igual que la función sound, salvo que previo a la reproducción escala todos los valores para que encajen en el intervalo [-1, 1]. De esta forma, se evita un posible

efecto de truncamiento ("clipping") de los valores fuera de esta intervalo. Más información tecleando "doc soundsc" desde el intérprete de órdenes de Matlab.

3.4. Captura de sonido: wavrecord y audiorecorder.

`wavrecord(n,Fs,Ch)`: captura una señal a través de la tarjeta de sonido del computador cuyos parámetros corresponden al número de muestras a tomar, frecuencia de muestreo (admite 8000, 11025, 22050 y 44100), el tipo de canal (1 para mono y 2 para estéreo). Más información tecleando `doc wavrecord` desde el intérprete de órdenes de Matlab.

También es posible utilizar la función `audiorecorder`, cambiando ligeramente la sintaxis y el tipo de datos utilizado.

3.5. Grabar fichero WAV: wavwrite y audiowrite.

`wavwrite(y,Fs,NBits,'Nombre.wav')`: guarda la señal almacenada en la matriz `y`, a la frecuencia de muestreo especificada en `Fs` y con la resolución `NBits` en el fichero especificado `'Nombre.wav'`. Hay que tener en cuenta que los valores de amplitud que estén fuera del rango [-1,+1] son truncados. Más información tecleando `doc wavwrite` desde el intérprete de órdenes de Matlab. En las últimas versiones de Matlab, `wavwrite` ha sido sustituida por `audiowrite`.

4. Librería propia para el manejo de sonido

En esta sección se describen las funciones que se deben de implementar en Matlab (ficheros .m) para realizar algunas tareas básicas con una señal de sonido.

4.1. Función "dibujaSonido"

Descripción: esta función debe dibujar la forma de onda de cada uno de los canales de los que consta el sonido que se le pasa como parámetro. Para dibujar en una misma ventana varios gráficos se usa la función "subplot". El título del gráfico, si se especifica como parámetro[1], figurará como título de cada gráfica con el subíndice del número de canal (ver Figura 3).

Sintaxis: `function dibujaSonido(signal,titulo);`

[1] Para saber el número de parámetros que se han pasado a la función se puede usar la orden de Matlab "nargin".

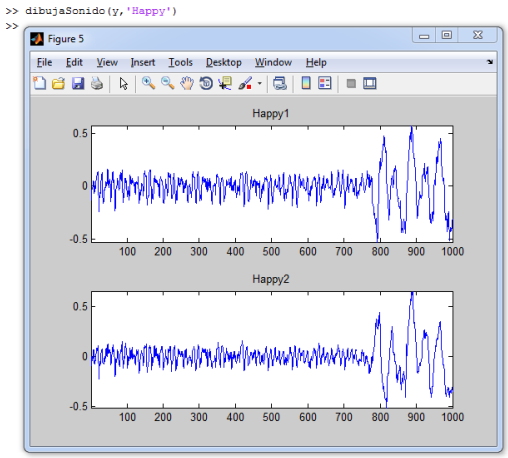


Figura 3. Ejemplo de gráfica obtenida con "dibujaSonido" para una señal estereofónica

4.2. Función "getCabeceraWav"

Descripción: obtiene los valores principales contenidos en la cabecera del fichero WAV cuyo nombre se le pasa como parámetro. Para leer un fichero cualquiera a nivel de bytes, en Matlab se puede usar la función `fread`. En este caso, se deben leer con esa función los 44 primeros bytes del fichero WAV que contienen la cabecera. A partir de ahí, cada uno de los parámetros requeridos se extrae de los bytes correspondientes. Para esta función, no se puede usar wavread con las opciones extendidas, aunque puede servir para comprobar que los valores son correctos.

Sintaxis:
```
function [canales, frecMuestreo, resolución, numeroBytesMuestras]
        =getCabeceraWav('fichero');
```

Ejemplo:

```
[ch, freq, resol, numBytes]   =getCabeceraWav('sound.wav');
      % devuelve ch=1, freq=44100, resol=16, numBytes=66150
```

4.3. Función "reproduceSonido"

Descripción: esta función reproduce el sonido que se le pase como parámetro, usando los valores de frecuencia de muestreo y resolución que también recibe como parámetros. Si se especifica un tiempo de inicio y un tiempo de parada, el sonido se reproducirá en ese intervalo. En caso contrario, el sonido se reproduce íntegro. La función devuelve la señal correspondiente al sonido reproducido.

Sintaxis:
```
function[y]=reproduceSonido(signal,fMuest, resol, t_ini, t_fin);
```

Ejemplo:

```
reproduceSonido(y, 8192, 8); % reproduce el sonido y íntegro a 8192 Hz y 8 bits
segmento=reproduceSonido(y, 8192, 8, 1.2, 2.85); % reproduce el sonido y, 8192 Hz
      % y 8 bits, entre 1.2s y 2.85s.
```

4.4. Función "grabaSonido"

Descripción: esta función captura sonido desde la línea de entrada de la tarjeta de sonido y lo graba en el fichero que se le pase como parámetro. Se emplean para el muestreo los valores de frecuencia de muestreo, resolución, número de canales y duración especificados como parámetros. La función devuelve la señal grabada (y).

Sintaxis: `function [y]=grabaSonido(fichero, frecMuestreo, resolucion, canales, duracion);`

Ejemplo:

```
grabacion=grabaSonido('sonido.wav', 8192, 16, 1, 10.3);
```

4.5. Función "insertaSonido"

Descripción: dadas las señales de sonido s1, s2 (las cuales deben tener el mismo número de canales y frecuencia de muestreo), inserta la señal s2 en el momento indicado como parámetro. A continuación de la señal s2, se continúa con el resto de la señal de s1. Si el momento indicado es mayor que la duración de s1, se inserta la señal s2 al final de s1. La función devuelve la señal generada (y).

Sintaxis: `function [y]=insertaSonido(s1, s2, frecMuestreo, tiempo);`

Ejemplo:

```
mezcla=insertaSonido(sound1, sound2, 44100, 23.7);
```

4.6. Función "fadeOut"

Descripción: produce un efecto de desvanecimiento lineal en la señal que se reciba unos segundos antes del final de dicha señal. Si la duración del efecto es mayor que el de la señal completa, se realizará un desvanecimiento desde el principio. La función devuelve la señal generada (y).

Sintaxis: `function [y]=fadeOut(signal, frecMuestreo, duracion);`

Ayuda: se puede aprovechar la naturaleza matricial de Matlab para realizar en pocos pasos esta función. Se trata de definir un vector "rampa" decreciente de 1 a 0 con tantos valores como muestras vaya a tener el efecto de desvanecimiento (la función linspace puede servir para esto). Después se multiplica escalarmente el tramo de señal que se quiere desvanecer por este vector rampa, obteniendo la señal modificada. En Matlab prácticamente no se usan los bucles for y while (si se puede hacer todo de una vez, ¿por qué hacerlo elemento a elemento?).

Ejemplo:

```
atenuada=fadeOut(sound, 8192, 3.0); % realiza un efecto de fade-out durante los 3
                                    % últimos segundos de la señal
```

4.7. Función "eco"

Descripción: produce un efecto de eco de la que se reciba con el retraso y el factor de potencia que se especifiquen como parámetros. El retraso especifica el tiempo que pasa hasta que se produce el eco, mientras que el factor de potencia especifica el tanto por uno de la muestra que se añade como eco. La función devuelve la señal generada (y).

Sintaxis: `function [y]=eco(signal, frecMuestreo, retraso, potencia);`

Ayuda: nuevamente, se puede aprovechar la naturaleza matricial de Matlab para calcular un vector de eco de la señal original (desplazado el retraso y atenuado según el factor de potencia). Al principio, se rellenaría con 0s durante el tiempo de retraso especificado. Finalmente se suman los dos vectores en un tercero que contendría la señal con el eco incorporado.

Ejemplo:

```
ecoSignal=eco(sound, 8192, 0.5, 0.8); % crea un eco del 80% de la señal con 1/2
                                      % segundo de retraso
```

Nota: si se añade al valor de las muestras de la señal otro valor, es posible que se sobrepasen los valores [-1, 1]. Para evitar que se trunquen todos esos valores, se recomienda usar la función “soundsc”.

4.8. Función "reverse"

Descripción: invierte la señal que recibe como parámetro, intercambiando la primera muestra por la última, la segunda por la penúltima, etc. La función devuelve la señal generada (y).

Sintaxis: `function [y]=reverse (signal);`

Ejemplo:

```
alreves=reverse(sound); % alreves es el sonido invertido de sound
```

Ayuda: se pueden usar funciones implementadas en Matlab para revertir vectores. Si no, siempre se puede recurrir a la programación tradicional iterativa, aunque es mucho menos eficiente.

5. Script para probar las funciones implementadas

Para comprobar el correcto funcionamiento, se debe crear un *script* de Matlab denominado “pruebaSonido.m” que realice las siguientes acciones:

1. Leer un fichero .WAV con wavread (no muy largo, de menos de 20 segundos de duración).
2. Dibujar la forma de onda de los canales con “dibujaSonido”.
3. Obtener la información de la cabecera del fichero con “getCabeceraWav”.

4. Grabar un fichero de 2 segundos desde la tarjeta de sonido.
5. Insertar la señal grabada a los 3 segundos de la primera señal.
6. Aplicar un eco del 40% con 0.5 segundos de retraso.
7. Aplicar un efecto de *fadeout* en el último segundo.
8. Dibujar la forma de onda de la señal resultante.
9. Almacenar la señal obtenida como un fichero WAV denominado “mezcla.wav”.
10. Reproducir al revés la señal obtenida.

La diferencia fundamental entre un *script* y una función es el alcance de las variables declaradas. En una función, para que actúen de forma modular, el alcance de las variables es local a la propia función, mientras que en un script, las variables se declaran en el “*Workspace*” de Matlab. Para identificar rápidamente si un fichero .m es una función o un script basta con mirar si en la primera línea se utiliza la palabra reservada *function*.

6. Evaluación y entrega

1. Completar y rellenar la tabla del Ejercicio 1, guardarla en un fichero PDF (ejercicio1.pdf)
2. Código fuente debidamente documentado de todas las funciones implementadas (4.1-4.8)
3. Código fuente debidamente documentado del script de prueba (pruebaSonido.m).

- Se puede realizar individualmente o por parejas.
- Se entregarán estos ficheros a través de la actividad correspondiente en SWAD comprimidos como un único zip (Practica3.zip).
- Número de sesiones para esta práctica: 3 (12 y 19 de mayo, 2 de junio)
- Fecha límite de entrega: 9 de junio de 2016.