Procesamiento Digital de Señales

Práctica 0. Introducción a Matlab y Simulink

1. Objetivos:

Iniciación en la utilización de la herramienta Matlab. Realizar algunos ejemplos de tratamiento de datos en Matlab.

2. Introducción

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

La figura 1 muestra la interfaz de Matlab que consta del entorno de comandos (Comand window, el espacio de trabajo (Workspace) y la historia de comandos introducidos (Command History).

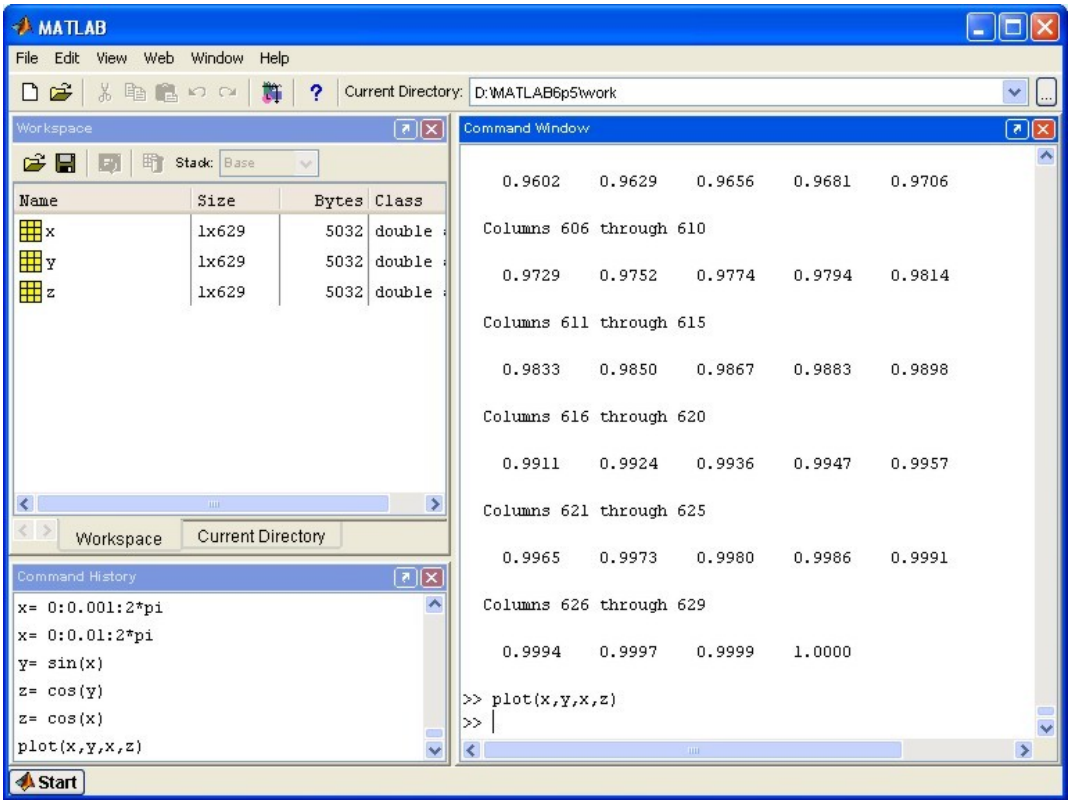

Figura 1. Interfaz de Matlab.

Realización práctica: A continuación se proporciona un tutorial sobre Matlab y Simulink. Para completar esta práctica debe ejecutar los comandos del tutorial y analizar la simulación de los modelos Simulink propocionados.

3. Tutorial sobre Matlab: comandos básicos

Ayuda de Matlab

Básicamente, existen dos formas de utilizar la ayuda de Matlab: a través de la ayuda en línea; o bien, a través del navegador de ayuda.

Para acceder a la ayuda en línea basta con teclear en la línea de comandos:

```
>> help funcion
```

donde "funcion" sería el nombre de la función sobre la que necesitamos la ayuda. Por otro lado, para acceder a la ayuda a través del navegador, es necesario seleccionar la opción "Matlab help" (Figura 2). Este segundo modo de ayuda resulta bastante más potente y eficaz que la primera añadiendo en muchos casos ejemplos de utilización.

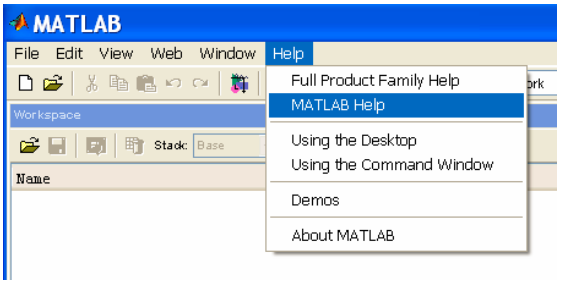

Figura 2. Menú de ayuda de Matlab.

Variables y matrices

Matlab no requiere ningún tipo de declaración de variables sino que, una vez que se utiliza una variable, Matlab crea la respectiva variable reservando el espacio de memoria necesario. Por tanto, si la variable ya existe, Matlab únicamente cambia su contenido.

En lo que se refiere a la nomenclatura de las variables. Matlab distingue entre mayúsculas y minúsculas ("Variable" es distinto de "variable") permitiendo nombres de variables que contengan al menos una letra.

Ejemplos: Introdúzcase el texto en rojo en el entorno de comandos.

Operaciones matemáticas básicas con constantes:

Entrada	Salida	Comentarios
2 + 3 7-5 34*212 1234/5786 2^5	ans = 5 ans = 2 ans = 7208 ans = 0.2173 ans = 32	Los resultados son los esperados. Nótese que al resultado se le da el nombre *ans*.
a = sqrt(2)	a = 1.4142	Se puede escoger el nombre de la variable.

Asignación de variables, definición y asignación de vectores:

b = a, pi, 2 + 3i	b = 1.4142 ans = 3.1416 ans = 2.0000 + 3.0000i	Se pueden introducir varios comandos en una sola línea. Pi, i, y j son constantes.
c = sin(pi) eps	c = 1.2246e-016 ans = 2.2204e-016	"eps" es el limite actual de precisión. No se puede operar con números inferiores a eps.
d = [1 2 3 4 5 6 7 8 9] e = [1:9] f = 1:9	d = 1 2 3 4 5 6 7 8 9 e = 1 2 3 4 5 6 7 8 9 f = 1 2 3 4 5 6 7 8 9	Definición de vectores. "d", "e", son "f" vectores. Son iguales. El operador ":" se utiliza para formar vectores; cuenta desde el número inicial al final de uno en uno.
g = 0:2:10 f(3) f(2:7) f(:)	g = 0 2 4 6 8 10 ans = 3 ans = 2 3 4 5 6 7 1 2 3 4 5 6 7 8 9	Otros usos de ":". Se utiliza para acceder a parte o la totalidad de los datos de un vector o matriz.

Trasposición de vectores, operaciones componente a componente e indexación:

`h = [1 2 3];`
`h'`
	`(nada)`
	`ans = 1`
	`2`
	`3`
		Un punto y coma ";" evita que se visualice la salida.
		Una coma simple " ' " calcula la traspuesta de una matriz, o en el caso de vectores, intercambia entre vectores fila y columna.

`h * h'`
`h .* h`
`h + h`
	`ans = 14`
	`ans = 1 4 9`
	`ans = 2 6 8`
		Operaciones con vectores. * es la multiplicación matricial. Las dimensiones deben ser las apropiadas. " .* " es la multiplicación componente a componente.

`g = [ 1 2 3;`
`4 5 6; 7 8 9]`
	`g = 1 2 3`
	`4 5 6`
	`7 8 9`
		Construcción de matrices.

`g(2,3)`
`g(3,:)`
`g(2,3) = 4`
	`ans = 6`
	`ans = 7 8 9`
	`g = 1 2 3`
	`4 5 4`
	`7 8 9`
		Accediendo a los elementos de la matriz.
		":" se utiliza para acceder a una fila completa.

Operaciones matriciales y operaciones “componente a componente”:

`g^2`
 	
`g .^ 2`
	`ans =  30  36  42`
	`66  81  96`
	`102 126 150`
	`ans =  1  4  9`
	`16 25 36`
	`49 64 81`
		Multiplica la matriz por ella misma.
		
		Eleva al cuadrado cada elemento de la matriz.

Inicialización de matrices mediante funciones de Matlab:

Entrada	**Salida**	**Comentarios**

`rand(2)`
 	
`rand(2,3)`
	`ans = 0.9501 0.6068`
	`0.2311 0.4860`
	`ans = 0.8913 0.4565 0.8214`
	`0.7621 0.0185 0.4447`
		Genera una matriz de números aleatorios entre 0 y 1

`zeros(2)`
 	
`ones(2)`
	`ans = 0 0`
	`0 0`
	`ans = 1 1`
	`1 1`
		Genera una matriz 2x2 de ceros o unos.

`eye(2)`	`ans = 1 0` `0 1`	Matriz identidad I.

`hilb(3)`	`ans = 1.0000 0.5000 0.3333`
	`0.5000 0.3333 0.2500`
	`0.3333 0.2500 0.2000`
		Matriz de Hilbert 3x3.

Concatenación de matrices:

Entrada	**Salida**
`[a, a, a]`	`ans = 1 2 1 2 1 2` `3 4 3 4 3 4`

`[a; a; a]`	`ans = 1 2`
	`3 4`
	`1 2`
	`3 4`
	`1 2`
	`3 4`

`[a, zeros(2); zeros(2), a']`	`ans = 1 2 0 0`
	`3 4 0 0`
	`0 0 1 3`
	`0 0 2 4`

Operaciones entre escalares y matrices:

Entrada	Salida	Comentarios
`b=2`	`b=2`	Define b como un escalar.
`a + b`	`ans = 3 4` `5 6`	La suma se hace componente a componente.
`a * b`	`ans = 2 4` `6 8`	Igual que la multiplicación.
`a ^ b`	`ans = 7 10` `15 22`	Potencia matricial - a*a
`a .^ b`	`ans = 1 4` `9 16`	Potencia componente a componente.

Operaciones básicas con vectores: producto escalar y vectorial:

Entrada	Salida	Comentarios
`v = [1 2 3]` `u = [3 2 1]`	`v = 1 2 3` `u = 3 2 1`	Define 2 vectores.
`v * u`	`Error`	Las dimensiones no coinciden.
`v * u'`	`ans = 10`	Al tomar la traspuesta se corrige el error.
`dot(v,u)`	`ans = 10`	Producto escalar (idéntico al anterior).
`cross(v,u)`	`ans = -4 8 -4`	El producto vectorial sólo se emplea con vectores en 3 dimensiones.

Operaciones con matrices: rango, determinante, inversa y diagonalización:

Entrada	Salida	Comentarios

`k = [16 2 3;`
`5 11 10;`
`9 7 6]`
	`k = 16 2 3`
	`5 11 10`
	`9 7 6`
		Define una matriz.

`rank(k)`	`ans = 3`	El rango.
`det(k)`	`ans = -136`	El determinante.

Entrada	Salida	Comentarios

`inv(k)`	`ans =  0.0294 -0.0662  0.0956`
	`-0.4412 -0.5074  1.0662`
	`0.4706  0.6912 -1.2206`
		Inversa de una matriz

`[vec,val] = eig(k)`	`vec = -0.4712 -0.4975 -0.0621`
	`-0.6884  0.8282 -0.6379`
	`-0.5514  0.2581  0.7676`
	`val = 22.4319  0  0`
	`0  11.1136  0`
	`0  0  -0.5455`
		Vectores propios y autovalores de una matriz.
		Las columnas de "vec" contienen los vectores propios;
		las entradas de la diagonal de "val" son los autovalores.

Resolución de sistemas de ecuaciones lineales:

Una de las principales aplicaciones de las matrices es la representación de sistemas de ecuaciones lineales. Si **a** es una matriz de coeficientes, **x** es un vector columna que contiene las incógnitas y **b** los términos constantes, la ecuación:

$$\boldsymbol{a\,x = b}$$

representa el correspondiente sistema de ecuaciones.

Para resolver el sistema en MatLab

```
>> x = a \ b          o          >> x=inv(a)*b
```

x es igual a la inversa de **a** por **b**

Ejemplo

```
a = [1 2 3; 4 5 6; 7 8 10]; b = [1 1 1]';
x= a\b
```

Solución:

```
      x                                                    =
    -1
     1
     0
```

Mostrar variables definidas en el espacio de trabajo:

El comando `whos` se emplea para mostrar la lista de las variables definidas en el entorno de comandos.

```
>> whos
  Name       Size                    Bytes  Class
  a        100x1                       800  double array
  b        100x100                   80000  double array
  c          1x1                         8  double array
Grand total is 10101 elements using 80808 bytes
```

Borrar y almacenar en disco variables en el espacio de trabajo:

El comando `clear` borra variables del entorno.

El comando `save` se utiliza para guardar en archivos variables del entorno.

```
>> save datos.mat          (guarda todas las variables)
>> save datos.mat x        (sólo guarda x)
```

Ejercicio: Generación de señales en Matlab:

Señales periódicas sinusoidales:

Supongamos que se desea generar una señal discreta mediante el muestreo de la señal continua $x(t)= A \sin(2\pi F_0 t+\theta)$ con una frecuencia de muestreo $F_s= 1/T_s$ obteniéndose $x(n)= x(nT_s)= A \sin(2\pi F_0 nT_s +\theta)$.

```
>> F0= 262;
>> A= 2;
>> phi= pi/4;
>> fSampling= 8000;
>> tSampling= 1/fSampling;
>> t= -0.005:tSampling:0.005;
>> x= A*sin(2*pi*F0*t+phi);
>> stem(t,x);
>> hold on;
>> plot(t,x);
>> xlabel('t sec');
>> ylabel('x(t)');
```

Señales no sinusoidales periódicas:

A continuación se generarán muestras de una señal diente de sierra.

```
>> F0= 262;
>> A= 2;
>> phi= pi/4;
>> fSampling= 8000;
>> tSampling= 1/fSampling;
>> t= -0.005:tSampling:0.005;
>> y= A*sawtooth(2*pi*F0*t);
>> figure;
>> stem(t,y);
>> hold on;
>> plot(t,y);
>> xlabel('t sec');
>> ylabel('y(t)');
```

Señales no periódicas:

```
>> F0= 262;
>> A= 2;
>> phi= pi/4;
```

```
>> fSampling= 8000;
>> tSampling= 1/fSampling;
>> t= -0.005:tSampling:0.005;
>> z= A*sinc(2*F0*t);
>> figure;
>> stem(t,z);
```

El paquete "Signal Processing Toolbox" de Matlab proporciona numerosas funciones para generación de señales. Utilice la función `pulstran` y `square` para crear un tren de pulsos y una señal cuadrada. Recuerde que puede consultar la documentación con los comandos:

```
>> doc pulstran
>> doc square

>> F0= 262;
>> A= 2;
>> phi= pi/4;
>> fSampling= 8000;
>> tSampling= 1/fSampling;
>> t= -0.005:tSampling:0.005;
>> w= A*square(2*pi*F0*t);
```

Señales aleatorias:

```
>> gNoise= randn(1,1e6);
>> hist(gNoise,100);
```

Ejercicio: Generación de una secuencia de notas musicales.

En la siguiente tabla se muestras las frecuencias de las siete notas de la segunda octava.

Nota	Do	Re	Mi	Fa	So	La	Si
Frecuencia (Hz)	524	588	660	698	784	880	988

Utilice una frecuencia de muestreo Fs= 4000 Hz y fases aleatorias.

```
>> clear all
>> close all
>> fSampling = 4000
>> tSampling = 1/fSampling;
>> t = 0:tSampling:0.5;
>> fNote = [524 588 660 698 784 880 988];
>> Do = sin(2*pi*fNote(1)*t+2*pi*rand);
>> Re = sin(2*pi*fNote(2)*t+2*pi*rand);
>> Mi = sin(2*pi*fNote(3)*t+2*pi*rand);
>> Fa = sin(2*pi*fNote(4)*t+2*pi*rand);
>> So = sin(2*pi*fNote(5)*t+2*pi*rand);
>> La = sin(2*pi*fNote(6)*t+2*pi*rand);
>> Si = sin(2*pi*fNote(7)*t+2*pi*rand);


>> expWtCnst = 6;
>> expWt = exp(-abs(expWtCnst*t));
>> Do = Do.*expWt;
```

```
>> Re = Re.*expWt;
>> Mi = Mi.*expWt;
>> Fa = Fa.*expWt;
>> So = So.*expWt;
>> La = La.*expWt;
>> Si = Si.*expWt;

>> noteSequence = [Do Re Mi Fa So La Ti];

>> soundsc(noteSequence,fSampling);
```

Ejercicio: Análisis espectral de secuencias de notas musicales.

```
>> nFFT = 2^14;
>> tuneF = fft(noteSequence,nFFT);
>> magTune = abs(tuneF);
>> phaseTune = angle(tuneF);
>> phaseTune = unwrap(phaseTune);
>> fSpacing = fSampling/nFFT;
>> fAxis = -fSampling/2:fSpacing:fSampling/2-fSpacing;
>> magTune = fftshift(magTune);
>> phaseTune = fftshift(phaseTune);

>> plot(fAxis,20*log10(magTune));
>> xlabel('Frequency F(Hz)')
>> ylabel('Magnitude X(F)')

>> figure
>> plot(fAxis,phaseTune);
>> xlabel('Frequency F(Hz)')
>> ylabel('Phase X(F)')

>> figure
>> spectrogram(noteSequence,256,0,[],fSampling)
```

4. Introducción a Simulink

Simulink es una herramienta basada en la construcción de diagramas de bloques para modelado, simulación y análisis de sistemas dinámicos. Soporta tanto sistemas lineales como no lineales: en tiempo continuo, muestreados, híbridos y sistemas multifrecuencia (contienen sistemas muestreados a diferentes frecuencias).

La figura 3 muestra la interfaz de trabajo de Simulink que consta de la librería de módulos o bloques que componen el sistema (Simulink Library Browser), la ventana de diseño del sistema que incluye bloques de la librería, así como ventanas de visualización de datos y resultados del análisis.

Para ejecutar Simulink en Matlab teclee `simulink` en el entorno de comandos.

```
>> simulink
```

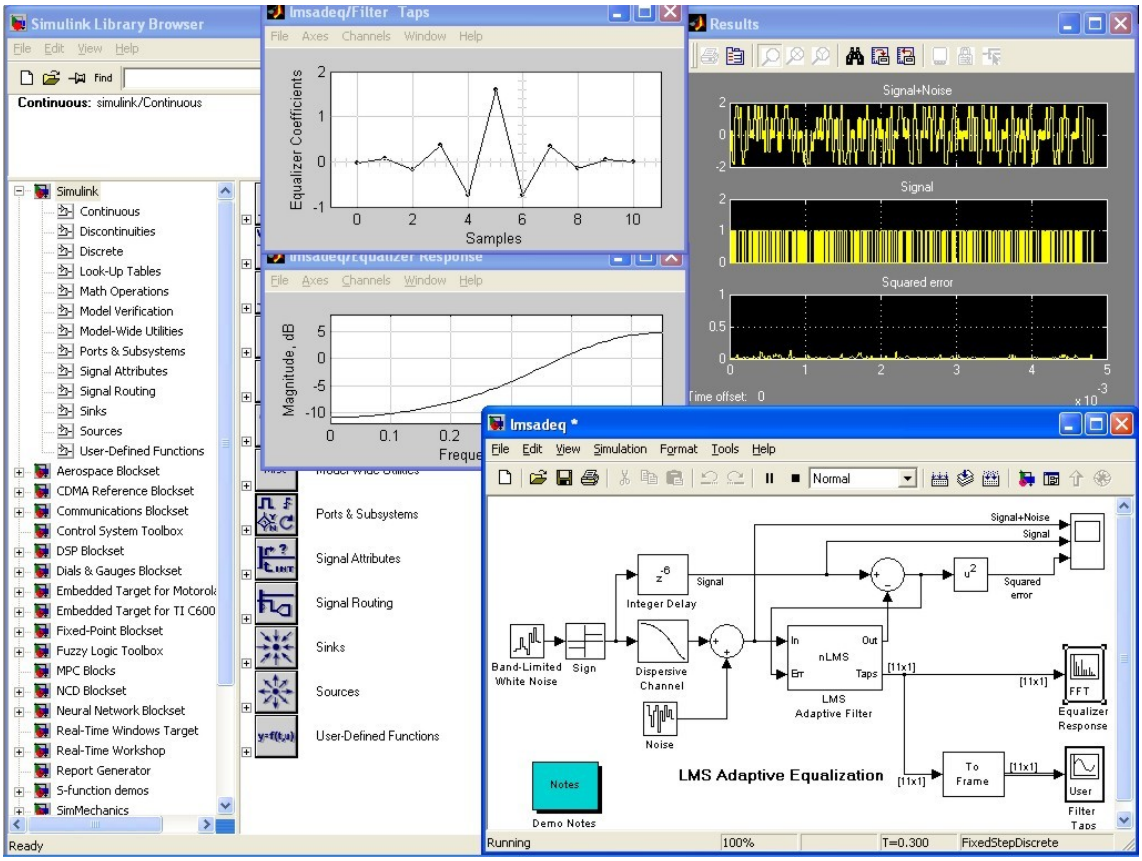

Figura 3. Interfaz de Simulink.

Realización práctica: Analice e interprete el ecualizador de audio de 5 bandas mostrado en la figura 4.

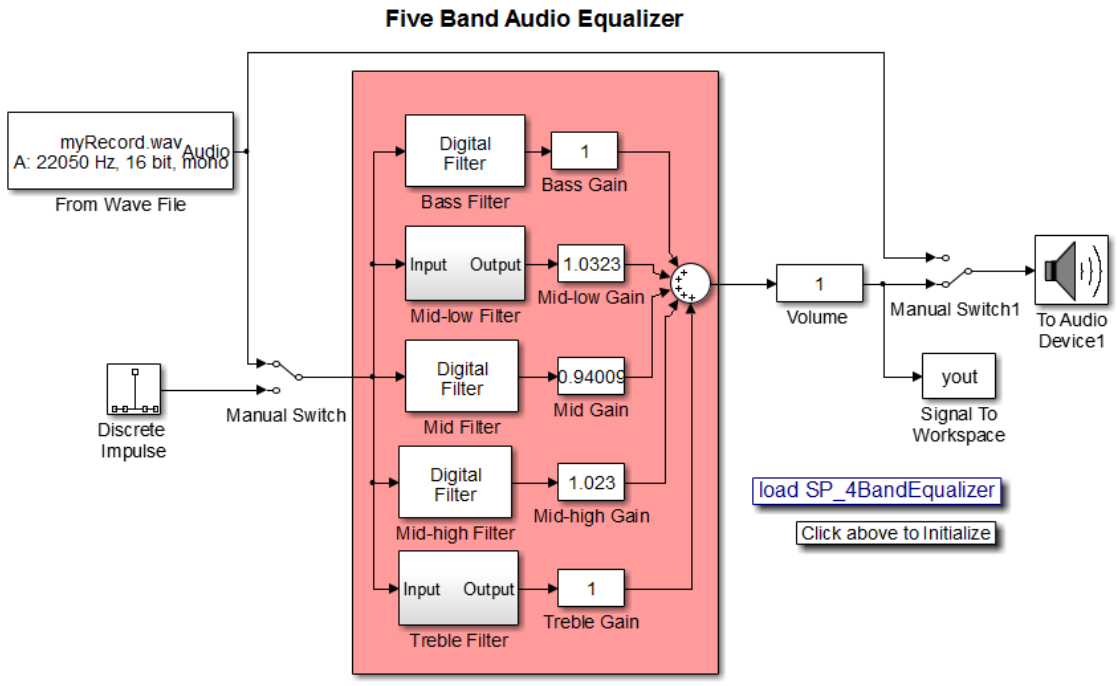

Figura 4. Ecualizador de audio de cinco bandas.