



Teoría de sistema lineales
Laboratorio 02: Introducción a Matlab
Escuela de Ciencias exactas e Ingeniería

Profesor: Marco Teran

Nombre: _____

Fecha límite: 4 de agosto de 2023

Resumen

En el presente laboratorio se aplicarán las bases para desarrollar los posteriores laboratorios, talleres y trabajos escritos. Se compilarán códigos en \LaTeX y se generarán archivos **PDF**, de acuerdo al formato de presentación. Se presentan los fundamentos del uso de **MATLAB**, así como una serie de ejercicios para graficar señales y realizar operaciones matemáticas básicas. En el presente laboratorio se implementarán algunas herramientas necesarias para generar señales directamente en **MATLAB**. Se buscará apropiarse del entorno de trabajo de **MATLAB**.

1. Introducción a MATLAB

1.1. MATLAB

MATLAB es un programa de cómputo basado en operaciones matriciales y orientado al cálculo numérico por computadora. Este programa resulta muy útil para ingenieros y para el área científica ya que puede resolver problemas numéricos de una manera más sencilla que utilizando lenguajes de programación de alto nivel.

MATLAB es un sistema interactivo cuyo elemento básico son las matrices y no requiere dimensionamiento. El nombre proviene de "laboratorio de matrices". Sobre la base de **MATLAB** se han construido conjuntos de funciones específicas para diferentes problemas y campos de la ingeniería y ciencias, denominadas *toolboxes*.

También hace referencia al lenguaje de alto nivel para computación e ingeniería utilizado por este programa. Integra computación, visualización y programación.

Aplicaciones típicas de **MATLAB** son:

- Matemáticas y computación
- Desarrollo de algoritmos
- Modelado, simulación y prototipado
- Análisis de datos, exploración y visualización
- Gráficos científicos y de ingeniería.
- Desarrollo de aplicaciones

1.2. El escritorio MATLAB

La actual Interfaz Gráfica de Usuario (GUI) de **MATLAB** maneja el estilo de pestañas y *ribbons*, introducido la primera vez por **Microsoft Office 2007**. Si está familiarizado con la interfaz moderna de **Microsoft Office**, no tendrás inconvenientes cuando utilice **MATLAB**.

Una vez ejecutado **MATLAB** se presentan diferentes ventanas. El escritorio de **MATLAB** es parecido al que se aprecia en la figura 1. De forma predeterminada se muestran 5 ventanas:

Ventana de Comandos (Command Window) Se puede ingresar y ejecutar comandos **MATLAB** y se muestran las salidas. La línea de comandos que puede identificarse mediante el símbolo `>>`.

Ventana del Editor (Editor Window) En esta ventana es posible editar los archivos de Matlab, `.m`. Estos archivos pueden de tipo *script* y contengan una secuencia de instrucciones en Matlab, para su posterior ejecución. Es posible también en estos definir funciones de usuario.

Ventana del Historial de comandos (Command History Window) Contiene la lista de los últimos comandos ejecutados desde la Ventana de comandos.

Espacio de trabajo o Ventana de variables (Workspace/Variables Window) Se encuentra la lista de los objetos y variables creados durante la sesión actual. Las variables que se desean utilizar se generan y almacenan en la memoria del programa al asignarles una cierta expresión. Al hacer doble clic en un elemento en esta ventana, se abre el elemento en el Editor donde puede ser examinado o editado.

Carpeta actual o Carpeta de trabajo (Current/Work Folder Window) Al comenzar a trabajar, debe ser especificado el directorio de proyecto.

Al incluir al final de la línea el símbolo `;`, se omite el despliegue del resultado en la pantalla y solamente se asigna el resultado a la variable designada.

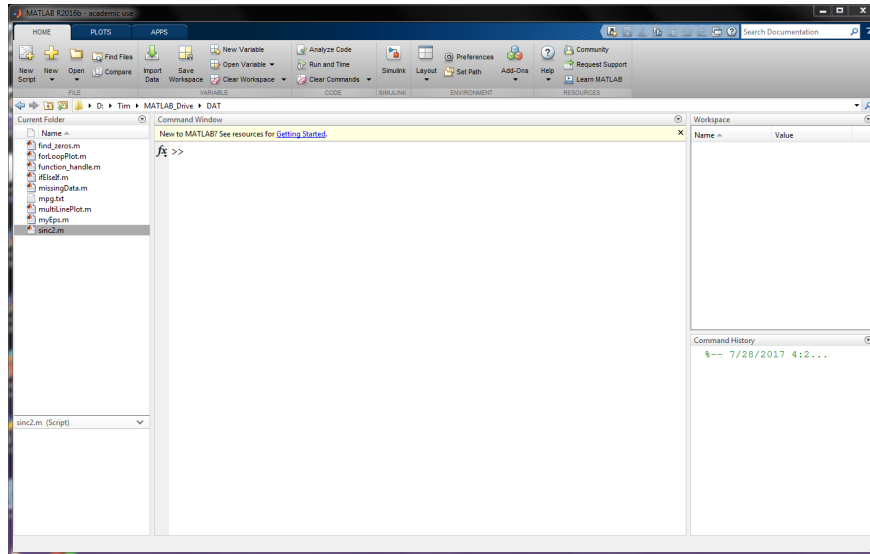


Figura 1 – Interfaz de desarrollo de MATLAB.

Comenzaremos introduciendo manualmente una matriz, para ello utilizamos las siguientes reglas:

- Separar elementos de una fila con espacios o comas.
- Usar *punto y coma* `;` para indicar final de fila.
- Incluir la lista completa de elementos dentro de corchetes, `[]`.

1.3. Formas de introducir matrices en MATLAB

Se puede introducir matrices en MATLAB:

- Lista explícita de elementos.
- Desde un fichero de datos externo.
- Utilizando funciones propias.
- Creando un fichero `.m`.

Para introducir la matriz de se escribe: Escribimos el siguiente código para borrar memoria, cerrar todas las ventanas abiertas y limpiar el *Command Window*

```
A = [ 1 6 3 2 1 3 ; 5 10 11 8 ; 9 6 7 1 2 ; 4 15 14 1 ]
```

```
A =
    16     3     2    13
     5    10    11     8
```

```
    9    6    7   12
    4   15   14    1
```

Una vez introducida la matriz, esta queda guardada en el entorno de trabajo de MATLAB. La matriz A es un cuadrado mágico: Todas sus filas, columnas y diagonales suman lo mismo. Para comprobarlo hacemos

```
sum(A)

ans =
    34    34    34    34
```

El comando `sum(A)` calcula la suma de las columnas de la matriz A, obteniéndose un vector de dimensión el número de columnas.

1.4. Expresiones en Matlab

Al igual que muchos otros lenguajes de programación, Matlab dispone de expresiones matemáticas, pero al contrario que en la mayoría de los lenguajes de programación, estas expresiones hacen referencia a matrices. Los bloques constructivos de las expresiones son

- Variables
- Números
- Operadores
- Funciones

1.4.1. Variables

Matlab no requiere ningún tipo de declaración o indicación de la dimensión. Cuando Matlab encuentra un nuevo nombre de variable la crea automáticamente y reserva la cantidad de memoria necesaria. Si la variable ya existe, Matlab cambia su contenido y si es necesario modifica la reserva de memoria. Por ejemplo, la expresión

```
num_est = 15
```

crea una matriz 1 por 1 llamada `num_est` y almacena el valor 15 en su único elemento. MATLAB no requiere declarar ni dimensionar variables.

- Las variables se auto-declaran al inicializarlas
- La memoria se reasigna dinámicamente

```
1 x=5;           % poniendo ; se realiza la asignacion pero no se muestra el ...
                  resultado
2 y=20;
3 z=x*y          % sin poner ; se puestra el resultado final

z =

    100
```

Otro ejemplo,

```
1 datos=load('datos.txt');
2 cadena='hola';
```

Realice la multiplicación de los vectores de datos: $x = [0,5,0,3,0,2,0,8,0,4,0,2]$, $y = [0,8,0,2,0,6,0,3,0,9,0,1]$. Encuentre el resultado de la multiplicación término a término de la serie de datos.

```
1 x = [0.5 0.3 0.2 0.8 0.4 -0.2];  
2 y = [0.8 0.2 -0.6 -0.3 0.9 0.1];  
3 z = x.*y
```

```
z =  
    0.4000    0.0600   -0.1200   -0.2400    0.3600   -0.0200
```

En el resultado anterior se observa que la multiplicación se realizó en una sola instrucción, además hay que destacar que la multiplicación punto a punto de dos vectores se indica mediante el punto (.) antes del símbolo de multiplicación *.

1.4.2. Números

Matlab utiliza notación decimal convencional, con punto decimal opcional y signo + ó -. Es posible utilizar notación científica. La letra *e* especifica un factor de escala de potencia de 10, ej. $1.25e-4$, que equivale a $1,25 \times 10^{-4}$. La variable compleja para expresar números imaginarios y complejos se especifica con la combinación *li* ó *lj*.

1.5. Scripts: Ficheros .m

Los *scripts* son ficheros de texto que contienen código de MATLAB. Un *script* es una secuencia de instrucciones de Matlab guardada en un archivo con extensión .m. Se ejecuta escribiendo su nombre

```
ejem_script
```

1.6. Funciones en MATLAB

Matlab proporciona un gran numero de funciones matemáticas elementales, por ejemplo, **abs**, **sqrt**..., **exp**, **sin**, **cos**, etc. Por defecto, Matlab utiliza números complejos: La raíz cuadrada o el logaritmo de un numero negativo no producen error, sino que dan como resultado un numero complejo. Los argumentos de las funciones pueden ser números complejos MATLAB proporciona también funciones avanzadas: Funciones de *Bessel* o funciones *gamma*.

Las funciones definidas por el usuario se escriben en archivos .m que deben encontrarse en el directorio actual (o en un directorio definido en el **PATH**).

Ejemplo de una función,

```
1 function [med,des]=med_des(x)  
2 % Funciona para calcular la media y la desviacion a la vez  
3 % [med,des]=med_des(x)  
4 %  
5 % Marco Teran (nov/2018)  
6  
7 med=mean(x(:));  
8 des=std(x(:));
```

El nombre de la función debe ser igual al nombre del archivo .m.

2. Desarrollo de la práctica de laboratorio

1. (5 points) Responda brevemente en la sección de Marco Teórico de su plantilla de laboratorio las siguientes preguntas:

- (a) Definir los siguientes comandos de MATLAB y describir sus funciones: `print`, `clear`, `close`, `clc...`, `round`, `sin`, `cos`, `'LineWidth'`, `'Color'`, `hold`, `xlabel`, `ylabel`, `legend`, `xlim`, `ylim...`, `grid`.
 - (b) Investigar como abrir y guardar variables del espacio de memoria de MATLAB en archivos tipo `.dat`.
 - (c) Investigar como guardar las imágenes del plot en formato PNG y alta calidad, para exportarlas a \LaTeX .
 - (d) Investigue acerca de la generación de señales aleatorias y las características estocásticas del ruido blanco Gaussiano aditivo (AWGN, *ing.* Additive White Gaussian Noise). ¿Como se genera en MATLAB una señal contaminada con ruido especificando una relación señal ruido (SNR, *ing.* Signal to noise ratio) predeterminada?
2. (5 points) **Generar una gráfica en MATLAB y guardarla:** MATLAB es un paquete de análisis matemático que trabaja solo con funciones vectoriales y matriciales, **discretas**. Procedemos a abrir la herramienta de software matemático MATLAB.

Procedemos a crea un nuevo archivo *script* CTRL+N y lo guardamos con el nombre

LST2023II_LAB02

Escribimos el siguiente código para borrar memoria, cerrar todas las ventanas abiertas y limpiar el *Command Window*

`clear all; close all; clc;`

Luego *seteamos* los parámetros por defecto de las gráficas.

MATLAB no trabaja variables continuas en su dominio, por tanto se hace necesario realizar aproximaciones de señales discretas en el tiempo a continuas, es decir, que si se toma un intervalo de muestreo que tienda a cero (o una frecuencia de muestreo que tienda a infinito), a *simple vista* la señal parecerá *como* de tiempo continuo, es decir aparentemente *aproximadamente continua*. Esta frecuencia se llamará **frecuencia aparente de MATLAB** f_s .

```
1 settings.fs = 1e5;
2 settings.ts = 1/settings.fs;      % step in time
```

Se aconseja tomar la costumbre de guardar parámetros característicos en forma de estructuras (orientadas a objetos).

A continuación se generará el vector de tiempos. Este se puede generar:

- Con un tiempo inicial t_i y un tiempo final dado t_f :

```
3 signal.ti = 0;      % intitial time
4 signal.tf = 0.1;    % final time
5 t=signal.ti:settings.ts:signal.tf;
```

- Con una duración determinada t_m :

```
6 signal.tm = 1e-1;      % signal duration
7 L = round(signal.tm/settings.ts); % Samples
8 t = (0:L-1)*settings.ts;
```

Se utilizará la primera forma en el laboratorio.

Procedemos a generar dos señales sinusoidales, una con frecuencia $f_1 = 2,42 \text{ kHz}$ y la otra con el doble de esa frecuencia $f_2 = 2f_1$

```

9 f_1 = 1.5e3;    %[Hz]
10 f_2 = 2*f_1;   %[Hz]
11 s1 = cos(2*pi*f_1*t); % Carrier based on time vector;
12 s2 = cos(2*pi*f_2*t); % Carrier based on time vector;

```

Se procede a la generación de gráficas en una primera figura 1.

```

13 figure()
14 plot(t/1e-3,s1);

```

Para dibujar las dos gráficas superpuestas se utilizan los comandos

```

15 hold on
16 plot(t/1e-3,s2,'-','LineWidth',2,'Color',[0.8500 0.3250 0.0980]);
17 hold off

```

Recreamos los parámetros del gráfico de tipo tamaño de la caja y las rejillas, etiquetas y leyendas

```

18 xmin=0; xmax=0.5; ymin=-1.1; ymax=1.1; % Box size
19 Box=[xmin xmax ymin ymax];
20 labels = {'t, [ms]', 's_n(t)'}; %labelx, labely
21 legends = {'$s_1(t)$', '$s_2(t)=-\int_0^{\pi} \{s_1(t) dt\}$'};
22 h=legend(legends);
23 set(h, 'Interpreter','latex', 'Location', 'NorthEast','FontSize',16,'...
    FontWeight','bold','Orientation','vertical');
24 xlabel(labels(1),'fontsize',16,'FontAngle','Italic');
25 ylabel(labels(2),'fontsize',16);
26 xlim([Box(1) Box(2)]); ylim([Box(3) Box(4)]);
27 grid(gca,'minor');

```

Para guardar la imagen, es preferible guardarla con los formatos .fig y .PNG automáticamente para futuras modificaciones. Se implementan los siguientes comandos

```

28 set(gcf, 'Position', get(0,'Screensize')); % Maximiza la figura
29 saveas(gca, 'example1','png');
30 print('example2', '-dpng', '-r600'); % La guarda como PNG con resolución ...
    de 600 DPI
31 saveas(gca, 'example1.fig');

```

A continuación se generará una imagen aleatoria. Vale recordar que cada vez que se compile este *script*, la línea de generación de ruido se volverá a interpretar y a generar un nuevo ruido. Muchas veces es necesario generar una sola interacción de una señal aleatoria, y debe ser utilizada más adelante sin perder esos datos, por eso guardamos los valores de la variable en un archivo tipo DAT.

```

1 noise=0.1*randn(1,length(s1));
2 save('noisesignal.mat','noise');

```

Para cargar la variable con el archivo guardado con anterioridad se utiliza el código

```
load('noisesignal.mat','noise');
```

Procedemos a compilar el siguiente resto de código, y describimos cada uno de los comandos utilizados.

```

3 subplot(2,2,1)
4 plot(t/1e-3,noise,'-','LineWidth',1,'Color',[0.3010 0.7450 0.9330]);

```

```

5 labels = {'t', [ms]', 'n(t)'}; % labelx, labely
6 xmin=0; xmax=0.5; ymin=min(noise)-0.2; ymax=max(noise)+0.2; % Box size
7 Box=[xmin xmax ymin ymax];
8 xlim([Box(1) Box(2)]); ylim([Box(3) Box(4)]);
9 grid(gca,'minor');
10 title('Ruido aleatorio')
11 subplot(2,2,2)
12 h = histogram(noise);
13 h.Normalization = 'probability';
14 h.BinWidth = 0.01;
15 h.FaceColor = [0.3010 0.7450 0.9330];
16 h.EdgeColor = 'k';
17 title('Histograma del ruido aleatorio')
18 subplot(2,2,[3 4])
19 plot(t/1e-3,s1+noise,'-','LineWidth',1,'Color',[0.3010 0.7450 0.9330]);
20 hold on
21 plot(t/1e-3,s1,'-','LineWidth',2,'Color',[0 0.4470 0.7410]);
22 hold off
23 xmin=0; xmax=0.5; ymin=min(s1+noise)-0.3; ymax=max(s1+noise)+0.3;
24 Box=[xmin xmax ymin ymax];
25 labels = {'t', [ms]', 's_n(t)'}; %labelx, labely
26 legends = {'$s_1(t)+n(t)$','$s_1(t)$'};
27 h=legend(legends);
28 set(h,'Interpreter','latex','Location','NorthEast','FontSize',16,'...
    FontWeight','bold','Orientation','vertical');
29 xlabel(labels(1),'fontsize',16,'FontAngle','Italic');
30 ylabel(labels(2),'fontsize',16);
31 xlim([Box(1) Box(2)]); ylim([Box(3) Box(4)]);
32 grid(gca,'minor');
33 title('Senal sinusoidal contaminada con ruido')

```

Reescriba este código en el documento de laboratorio comentando todas las líneas (utilice la librería **lstlisting** para escribir el código). Todas las gráficas generadas se deben guardar e incluir en el informe. Las líneas de código relevantes deben ser comentadas. Debe existir una correcta *indentación*.

3. (10 points) Generar señales continuas y discretas en Matlab:

Los siguientes comandos aparecen en versión de estándar, pero usted deberá aplicar el estilo y guardar las gráficas tal cual como se mostró anteriormente. Recuerde comentar cada línea de código.

En el siguiente *script* se generará una señal sinusoidal *aparentemente continua* (no se volverá a utilizar esta denominación, se debe sobreentender) con una frecuencia de $f_m = 10 \text{ kHz}$. Para que la señal discreta sea *aparentemente continua* se utilizará una frecuencia de muestreo de $f_s = 500 \text{ kHz}$. En el siguiente código representaremos el osciloscopio de la señal (comportamiento de la señal en el dominio del tiempo) y la magnitud de su espectro.

```

1 fs=1e6; % Frecuencia de muestreo se la señal de Matlab 1 MHz
2 f=10e3; % Frecuencia de la señal senoide
3 ti=0; % Tiempo incial de la señal [s]
4 nCyl=5; % Numero de ciclos (periodos) a generar de la sinusoidal
5 t=ti:1/fs:nCyl*1/f; % Vector de tiempo conntinuo
6 x=cos(2*pi*f*t); % Señal senoide
7
8 figure()
9 plot(t,x) % Dibujar la señal
10 title('Señal sinusoidal continua');
11 xlabel('Tiempo, [s]'); ylabel('Amplitud');
12 figure

```

```

13 [f,X]=spectrumGen(x,fs,2);           % Espectro bidireccional de la señal y ...
    vector de frecuencia
14 plot(f,X)                           % Dibujar la señal
15 title('Espectro de la señal sinusoidal');
16 xlabel('Frecuencia, [Hz]');
17 ylabel('Amplitud');

```

Código 1 – Representación de una señal continua en Matlab

Para encontrar el espectro de la señal utilizamos la función

`[freq,X]=spectrumGen(Signal,fs,side)`.

La cual tiene como entradas la señal *Signal*, *fs* la frecuencia de muestreo aparente y *side* para establecer si el espectro es de un lado o dos lados (*side* = 1 o *side* = 2). Descargue `spectrumGen.m` en la carpeta `/laboratory/introtomatlab/code/` del repositorio del repositorio del curso [aquí](#).

Para trabajar con secuencias de tiempo discreto en MATLAB, en la practica es mucho más sencillo. Para gráficas de señales de tipo discreto, se utiliza el comando `stem` en lugar de `plot`, ambos son análogos.

A continuación, se representará en MATLAB la siguiente secuencia de tiempo discreto:

$$x[n] = \dots, 0, 0, 5, 1, 1, 5, 2, 2, 5, 3, 2, 5, 2, 1, 5, 1, 0, 5, 0, \dots$$

Para ello primero debemos definir el vector tiempo n . En este caso, para tener un panorama completo de la señal, lo ideal es crear un vector temporal entre $[-8, 8]$, de la siguiente forma: `n=-8:1:8`, lo cual significa que generará un vector de -8 a 8 , con un paso (*step*) de una unidad. El siguiente código se utilizara para representar de forma gráfica la señal, el espectro no presenta ningún tipo de interés.

```

1 n=-8:1:8; % Vector de tiempo discreto
2 x_n=[0,0,0,0.5,1,1.5,2,2.5,3,2.5,2, 1.5,1,0.5,0,0,0]; % señal x_n
3 stem(n,x_n); % Dibujar la señal
4 title('Señal de tiempo discreto');
5 xlabel('n, [sample]');
6 ylabel('x[n]');

```

Código 2 – Señal de tiempo discreto en MATLAB

Dibuje de forma discreta, y para un intervalo adecuado donde se puedan apreciar todas las características, las siguientes funciones `sign`, `sawtooth`, `pulstran`, `rectpuls`, `tripuls` y `gauspuls`.

3. Recursos adicionales

- Curso de Matlab completo en YouTube [enlace](#).
- CheatSheet (hoja de lista de comandos útiles) de Matlab [descargar](#).
- Libro **MATLAB for Engineers** [descargar](#).

NOTA: Es obligatorio que se realice un *rigurosa* presentación de resultados obtenidos para cada sección del informe de laboratorio y se realice su respectivo análisis. Adicionalmente, no olvide incluir las conclusiones apropiadas del laboratorio.