



U
N
E
X
P
O

REPÚBLICA DE VENEZUELA
UNIVERSIDAD NACIONAL EXPERIMENTAL POLITÉCNICA
"ANTONIO JOSÉ DE SUCRE"
VICE-RECTORADO DE BARQUISIMETO
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

Introducción a MATLAB¹

1. Introducción

Aunque el nombre MATLAB significa *laboratorio de matrices*, MATLAB puede utilizarse como una herramienta muy poderosa para todo tipo de cálculos y visualizaciones científicas, de ingeniería y otros campos. También es posible considerarlo como otro lenguaje de programación para el desarrollo de funciones avanzadas a partir de las utilidades estándares de MATLAB. Estas características han hecho de MATLAB una herramienta efectiva para la educación y la investigación. El objetivo de este documento es mostrarte lo fácil que es utilizar MATLAB y además darte una idea de lo que puede hacerse con él. Se asume que se utilizará MATLAB para windows.

2. Iniciando MATLAB



Para iniciar MATLAB, ubique en el escritorio el icono MATLAB  y haga doble clic sobre él. Si no aparece este icono en su escritorio, debe buscarlo en Inicio – Programas – MATLAB for windows – MATLAB. Al iniciarse MATLAB abre una ventana de comandos con un indicador de petición (prompt) ">". El indicador de petición ">" significa que MATLAB está esperando que teclees comandos para su ejecución. Por ejemplo, teclea:

```
clock
```

y deberías ver la fecha y la hora actual desplegadas en la ventana de comando. MATLAB tiene un servicio de ayuda en línea incorporado que puedes usar para obtener más información sobre los comandos. Por ejemplo, teclee

```
help clock
```

```
CLOCK Wall clock.
```

```
CLOCK returns a six element row vector containing the  
current time and date in decimal form:
```

```
CLOCK = [year month day hour minute seconds]
```

```
The first five elements are integers. The seconds element  
is accurate to several digits beyond the decimal point.
```

```
FIX(CLOCK) rounds to integer display format.
```

```
See also ETIME, TIC, TOC, CPUTIME.
```

¹ Este documento es la traducción y adaptación del original *Introduction to MATLAB* del Dr. Nishan Canagarajah, University of Bristol, Dept. of Electrical & Electronic Engineering, UK, Sept. 1998. Revisado, traducido y adaptado por Luis Tarazona, UNEXPO Barquisimeto, Dep. de Ingeniería Electrónica, Venezuela, octubre de 2001.

Para información más detallada de los comandos, puedes leer la guía de referencia de MATLAB y para obtener una lista completa de los comandos disponibles, simplemente teclea **help**. !Es importante notar que (a diferencia del C) los índices de los arreglos en MATLAB empiezan en 1 y no en 0!

Cuando quieras salir de MATLAB teclea **quit** o **exit**. Sin embargo, si quieres parar de trabajar pero te gustaría tener tus resultados guardados para la próxima sesión escribe **save** y todas tus variables se almacenarán en el archivo *matlab.mat* (el archivo por omisión – más detalles sobre esto después) Cuando quieras continuar tu trabajo, ejecuta de nuevo MATLAB y teclea **load**. El archivo se leerá y todas tus variables se restaurarán, permitiéndote continuar desde el mismo punto que paraste en la sesión previa.

Verás que los comandos **clear** y **pack** son muy útiles para optimizar el uso de la memoria.

```
clear A
```

eliminará la variable A del espacio de trabajo. Tecleando **clear** se eliminarán todas las variables del espacio de trabajo. El comando **pack** empaquetará todas las variables para optimizar el uso de la memoria. Puedes obtener más información acerca de las variables en tu espacio de trabajo con los comandos **who** y **whos**.

3. Demos

Una manera interesante de aprender las características y bondades de MATLAB es mediante las demostraciones. Teclea el comando **demo**, selecciona las demostraciones (demos) de interés y sigue las instrucciones. Es esencial que visites la sección básica MATLAB para que aprendas cómo introducir matrices o vectores y realizar cálculos simples con ellos. No procedas a ver otros demos sin antes hacer doble clic en el icono MATLAB  de la demostración (a menos que ya seas un experto en MATLAB). Al final de este corto tour ya deberías estar al tanto de las capacidades de MATLAB y con suerte verás lo fácil que es usarlo!

4. Ejemplos

Los siguientes ejemplos se dan como ejercicios para obtener confianza en el uso de MATLAB.

Ej. 1 Operaciones escalares

*Calcule los valores de $1.2 \cdot \sin(\pi/4 + 0.7236^2)$ y de $1.2 \cdot \cos(\pi/4 + 0.7236^2)$. Asigne los resultados a las variables **a** y **b** respectivamente. Si $\mathbf{c} = \mathbf{b} + i \cdot \mathbf{a}$, ¿cuál es la magnitud y la fase de **c**?. Pide ayuda sobre los comandos **abs** y **angle**. Debes notar que **i** y **j** pueden usarse para denotar la componente imaginaria de un número complejo siempre que **i** y **j** no tengan asignado otro valor.*

*Si $\mathbf{x} = [1 \ 2 \ 3 \ 11 \ 23 \ 45 \ 33]$ determine el valor promedio (mean) de **x**.*

Ej. 2 Operaciones con vectores

$a=[1\ 2\ 3\ 4]$ y $b=[10\ 20\ 30\ 40]$. ¿Cuál es el valor de $c=a.*b$? Calcule $d=\text{sum}(c)$ y sugiera una instrucción alternativa de MATLAB para calcular d (una sola instrucción). ¿Cuál es la diferencia entre $a*b'$ y $a'*b$?

Ej. 3 Operaciones con matrices

A es una matriz de 4×4 dada por $A=\text{randn}(4,4)$. encuentre los eigen valores y los eigen vectores usando el comando **eigen**. Encuentre la inversa de A dada por $B=\text{inv}(A)$ y comente del resultado $B*A$. Si $b=\text{randn}(4,1)$ y $Aw=b$ calcule los valores de w . La anterior es la ecuación de estimación lineal estándar que se conoce como la ecuación normal.

Ej. 4 Operaciones de control

Aunque MATLAB es muy eficiente para manipular matrices, puede ser extremadamente lento para operaciones iterativas. Por ello es recomendable evitar bucles **for** tanto como sea posible y en su lugar escribir expresiones en forma matricial. Este es un ejemplo simple para aprender como la autocorrelación, que normalmente se calcula usando bucles **for**, puede calcularse eficientemente in forma matricial en MATLAB:

```
r(p) = 0
for k=1:1001-p,
r(p)=r(p) + 1/(1001-p)*(s(k)*s(k+p-1));
end;
```

El código anterior representa $r(p) = \frac{1}{1001-p} \sum_{k=1}^{1001-p} s(k)s(k+p-1)$. Encuentre una expresión equivalente para evaluar las autocorrelaciones sin usar el bucle **for**. Si $s=\text{randn}(1000,1)$, encuentre la autocorrelación de hasta 9 retrasos. (es decir, $r(1)..r(10)$). Si la potencia de una señal está dada por $e = \frac{1}{N} \sum s(k)^2$, determine la potencia de s .

5. Salida gráfica

El sistema gráfico de MATLAB provee de una variedad de técnicas sofisticadas para visualizar datos. La sintaxis básica es **plot(x,y,'tipo de línea')**, lo cual grafica el vector y (ordenada) contra el vector x (abscisa) con el tipo de línea especificado (color y marcador). Por ejemplo para graficar los primeros 100 elementos de s que creamos en el ejemplo anterior, teclea

```
plot(s(1:100))
```

El gráfico aparece en una ventana separada. Podemos agregar etiquetas a los ejes y título a este gráfico, tecleando en la ventana de comandos

```
xlabel('Muestra #'), ylabel('Amplitud'), title('Una señal aleatoria ')
```

Puedes solapar múltiples salidas gráficas en una sola ventana usando diferentes estilos de línea luego de “congelar” la ventana actual de gráficos con el comando **hold**. **hold off** libera la ventana de gráficos actual. Para posicionar texto manualmente en el gráfico utiliza el comando **gtext**.

Existen otras posibilidades para controlar la salida gráfica. Es posible obtener gráficos log-log o semi-log, gráficos 3D, gráficos de malla (mesh), etc. Para imprimir un gráfico elige Print desde el menú FILE de la ventana del gráfico. También es posible copiar el gráfico y pegarlo en otro documento de windows: revisa el menú EDIT.

6. Herramientas de Procesamiento de señales

En los ejemplos anteriores aprendiste a utilizar algunas funciones estándares de MATLAB para realizar diferentes operaciones con matrices. El propósito de esta sección es presentar la Caja de Herramientas (Toolbox) de Procesamiento Digital de Señales, la cual contiene operaciones que se encuentran comúnmente en procesamiento digital de señales. Estas funciones están escritas a partir de las rutinas básicas de MATLAB, lo cual te permite utilizarlas sin tener que escribirlas por ti mismo. De nuevo, el demo de MATLAB te ayudará a entender las características y funcionalidad de esta caja de herramientas. Ejecuta de nuevo el demo, visita el icono de Toolbox y sigue las instrucciones en el icono de procesamiento de señales.

Generación de señales

En MATLAB, las señales se representan bien sea como vectores (unidimensionales) o como matrices (bidimensionales). Para generar una secuencia de señal podemos especificar cada elemento de la señal o utilizar las funciones matemáticas estándar. Con mucha frecuencia nos interesa generar señales sinusoidales o señales aleatorias. Generemos una señal que tenga componentes sinusoidales a 25Hz y a 35Hz, muestreada a 100Hz

```
t=0:0.01:1;
y=sin(2*pi*25*t) + sin(2*pi*35*t);
```

Hemos generado una señal **y** de 101 elementos de longitud (101 valores o muestras). Agreguemos algo de ruido aleatorio con una desviación estándar de 0.1 a la señal anterior. Los comandos **rand** y **randn** son las rutinas de MATLAB usadas para generar secuencias aleatorias con distribuciones uniforme o normal, respectivamente. Así,

```
yn= y + 0.1*randn(1, length(t));
```

Ahora grafica esta nueva señal y observa como se ve. Infortunadamente, este gráfico no nos da ninguna información útil sobre la señal. Una forma alternativa de ver esta señal es en el dominio de la frecuencia mediante Transformadas de Fourier. Obtengamos una transformada de Fourier de 128 puntos de la señal **yn**. Esto se puede hacer muy fácil en MATLAB usando el siguiente comando

```
Yn=fft(yn,128)
```

Dado que Y_n normalmente es un número complejo, no podemos graficarlo directamente para visualizar la transformada de Fourier. Necesitamos realizar otras dos operaciones simples para poder hacerlo. Primero necesitamos definir el eje de frecuencia. Dado que la transformada de Fourier de una señal real es simétrica respecto a la frecuencia de Nyquist (50Hz en este caso), solamente necesitamos los primeros 65 elementos de Y_n . Por lo tanto, la variable frecuencia f está dada por

```
f = 50*(0:64)/64;
```

y ahora grafica la transformada de Fourier usando

```
plot(f, abs(Yn(1:65)));
```

Resulta obvio a partir de este gráfico que la señal está compuesta de dos sinusoides a las frecuencias especificadas más algo de ruido.

También podemos manipular y procesar las señales para analizar diferentes características. Por ejemplo, podemos extraer el componente de 25Hz de la señal que acabamos de generar con solo usar la información de la transformada de Fourier. La primera tarea para lograr esto es diseñar un filtro con una respuesta en frecuencia deseada tal que la salida del filtro contenga solamente el componente de 25 Hz.

Diseño de filtros

Una manera muy conveniente de caracterizar cualquier filtro arbitrario es mediante funciones de transferencia. La siguiente función de transferencia

$$H(z) = \frac{3 + 2z^{-1} + 4z^{-3}}{1 + 3z^{-1} + 5z^{-2}}$$

será introducida en MATLAB con

```
b = [ 3 2 0 4 ];  
a = [ 1 3 5 ];
```

La respuesta en frecuencia de este sistema se puede calcular muy fácilmente usando el comando **freqz**. Para calcular la respuesta en frecuencia en 64 puntos igualmente espaciados dentro del intervalo $[0 \pi]$ tecleamos

```
[h, w] = freqz(b, a, 64);
```

Como ejercicio, grafique la magnitud de la respuesta en frecuencia de este filtro asumiendo que la frecuencia de Nyquist es 50Hz,

Para extraer la componente de 25Hz, necesitamos un filtro con una respuesta unitaria en 25Hz y cero en cualquier otra frecuencia. En este ejemplo consideramos una de las rutinas, **fir2**, para ilustrar las capacidades de diseño de filtros. Esta rutina nos permite diseñar un filtro FIR a partir de una característica magnitud-frecuencia dada. Teclea **help fir2** para aprender la sintaxis.

Para extraer la componente de 25Hz, requerimos que la respuesta en magnitud de nuestro filtro sea 0 en todas las frecuencias menos en 25Hz. Esto lo podemos especificar como sigue

```
f=[0 0.1 0.4 0.45 0.5 0.55 0.8 1];
m=[0 0 0 0 1 0 0 0];
```

Podemos graficar esto para ver la respuesta en frecuencia del filtro.

```
plot(f*50, m)
```

y podemos notar que la respuesta es cero excepto en 25Hz. Ahora,

```
b1=fir2(100,f,m);
```

generará un filtro FIR de 100 secciones ajustado a la respuesta en frecuencia anterior. *Grafique la respuesta en frecuencia de este filtro usando el comando **freqz***. Este filtro puede ahora ser usado para extraer la componente deseada, al filtrar la señal original **yn**.

Filtrado

El filtrado está en el corazón de la mayoría de los algoritmos de procesamiento de señales. El filtrado en el dominio del tiempo se ejecuta fácilmente en MATLAB mediante el comando **filter**. Luego de haber obtenido la respuesta al impulso de la función de transferencia de nuestro filtro, podemos ahora filtrar la señal **yn** para obtener nuestra componente de 25Hz con un solo comando, como se muestra mas abajo.

```
ynf = filter(b1, 1, yn);
```

*Grafica la fft de **ynf** y verifica que la señal filtrada contiene solo la componente de 25Hz. Grafica la señal **ynf**. ¿Puedes explicar porqué esta señal no es una senoide perfecta?*

Los ejemplos anteriores ilustran claramente las características de la caja de herramientas (toolbox) de procesamiento de señales para el diseño de filtros y el análisis espectral. *Diseña un filtro para extraer la componente de 35Hz de la señal **yn***.

7. Programación en MATLAB

En MATLAB se pueden agrupar los comandos básicos para crear archivos guión (scripts) y archivos de función. Estos archivos deben tener una extensión **.m** para indicar que son un *archivo-M* de MATLAB. Los archivos guión son normalmente una colección de comandos de MATLAB que simplemente se ejecutan en forma secuencial. Los archivos de función permiten que se pasen argumentos y las variables se manipulan localmente para entregar los resultados especificados. Esto es útil cuando nos interesa sólo el resultado y no las operaciones intermedias ya que todas las variables de la función son locales y se eliminan del espacio de trabajo. La escritura de estos archivos es muy sencilla porque un archivo guión es solo una colección de comandos estándar de MATLAB almacenados en un *archivo-M*.

Sin embargo, los archivos de función requieren de un comando en el inicio del archivo para indicar los argumentos y un nombre de archivo que debe ser igual al nombre de la función. Se presenta un ejemplo simple de una función que calcula la autocorrelación de un vector columna, por lo cual se le ha nombrado como **autocorr.m** (Note que el nombre de la función es idéntico al nombre del archivo).

```
function y=autocorr(x,p)
%
% Puedes insertar comentarios aquí para dar explicaciones.
% Todos los comentarios deben ir precedidos del signo %.
% A diferencia de C, se necesita un signo % por cada nueva línea
% y la línea completa es tratada como un comentario.
%
% En este caso, x es la señal y p el número de retardos.

n = length(x);
for k = 1:p+1,
    y(k) = 1/(n-k+1)*(x(1:(n-k+1))'*x(k:n))
end;
```

ahora al teclear

```
z = autocorr(s, 10)
```

Se guardarán en **z** los estimados de la autocorrelación de **s**.

ADVERTENCIA: Asegúrate que el nombre de tu función no esté en conflicto con una función MATLAB existente.

En este instante es útil notar que es posible invocar comandos del sistema operativo desde el ambiente MATLAB. En particular, es útil para ejecutar el editor de texto, cambiar el directorio, etc. El carácter de exclamación (!) pasa cualquier comando que lo sigue al sistema operativo. Por lo tanto, para invocar el editor de notas (notepad) desde MATLAB, sólo teclea

```
!notepad test.m &
```

lo cual iniciará notepad como una nueva tarea de windows.

*Como ejercicio, escriba la función **[m1, f1] = fft_mag(n, N)** para calcular la magnitud del espectro **m1** de **N** puntos y las frecuencias correspondientes **f1**, de una señal real **s**. Asuma que la frecuencia de muestreo es 1 kHz. Pruebe su función con **s=sin(2*pi*200*t)** y comente de los resultados obtenidos al variar **N**.*

8. Entrada y salida mediante archivos

Es casi seguro que en ocasiones querrás guardar los resultados de una sesión en particular de MATLAB para un uso futuro. Los comandos **save** y **load** sin argumentos usan el archivo por omisión *matlab.m*. También podemos especificar el nombre de un archivo y el nombre de la variable para guardar diferentes variables en archivos de datos diferentes. Por ejemplo,

```
save data1 A
```

almacenará la variable **A** en el archivo de datos *data1.mat* con formato MATLAB. Sin embargo, el formato de datos de MATLAB puede no ser conveniente cuando deseas exportar tus resultados a otras aplicaciones o software. Por esto el comando **save** también permite que se almacenen los datos en formato ascii, como se muestra a continuación (la extensión *.dat* del archivo es importante para identificarlo como un archivo ascii).

```
save data1.dat A -ascii
```

Para recuperar la variable **A** utiliza

```
load data1.dat
```

Lo cual leerá el archivo *data1* y recuperará las variables almacenadas en el mismo.

MATLAB también provee rutinas para analizar datos contenidos en archivos de texto y en archivos binarios de datos. Las capacidades de E/S por archivos de MATLAB están basadas en el lenguaje de programación C. De este modo, **fread**, **fwrite**, **fscanf** y **fprintf** son algunos de los comandos que permiten un manejo poderoso de archivos externos desde MATLAB.

9. Conclusiones

Este ejercicio fue diseñado para resaltar las características esenciales de MATLAB para el análisis de datos y procesamiento de señales. Hemos visto que MATLAB es muy útil para muchos tipos de cálculos numéricos. De hecho, existen muchas cajas de herramientas de MATLAB para diferentes aplicaciones tales como identificación de sistemas, control y redes neuronales, lo cual ha hecho de MATLAB una herramienta muy usada en investigación de ingeniería. Sin embargo, la incapacidad para realizar manipulaciones simbólicas es probablemente una de las pocas limitaciones de MATLAB.

Dr. Nishan Canagarajah
Septiembre 1998

Traducido y revisado por:
Luis Tarazona.
Octubre 2001.