MATLAB[1] TUTORIAL:

Federico Andrés Bugni
Fbugni@udesa.edu.ar

El siguiente tutorial fue creado para introducir comandos básicos de Matlab, con la intención de permitirles entender los programas que se utilizarán en la materias Métodos Recursivos y Econometría no paramétrica del master en economía de la Universidad de San Andrés.

Lo básico - escapar y pedir ayuda:

Para salir del Matlab tenemos los comandos:

```
quit
exit
```

Para obtener índice de ayuda:

```
help
```

Para obtener ayuda sobre un comando en particular:

```
help X
```

donde X es el nombre del comando.

Para iniciar un registro de todo lo que suceda en el command window: comandos utilizados y operaciones realizadas:

```
diary X
```

que se guarda en el directorio c:/matlab/work/ y se puede leer normalmente con cualquier editor de texto, por ej. Ms Word.

Para interrumpir o reiniciar el diario se puede usar:

```
diary
diary on/off
```

Manejando Escalares:

Para asignar un valor a una variable, que llamamos por ejemplo x, se usa el signo =. Por ejemplo:

```
x=3
```

notar que ese comando se ejecuta en el command window, en forma "ruidosa".

Para que un comando se ejecute en el command window en forma silenciosa, utilizamos ; después de cada comando. Por ejemplo:

```
x=3;
```

En general, las variables en Matlab serán matrices, pero al asignarle a cualquier variable sólo un número, el programa se da cuenta de que estamos hablando de una matriz de 1x1, es decir, un escalar.

[1] Versión 5.3.0.10183, Release 11.

Las variables en Matlab pueden tener cualquier nombre largos escritos con letras. La excepción son algunas variables ya asignadas automáticamente por Matlab.
El Matlab es un programa Caps-sensitive, es decir, para el programa Matlab, es distinta de la variable si tiene mayúscula o no. Por ejemplo: X es distinto de x, o Vari es distinto de VaRi.
Ejemplos de algunas variables previamente asignadas por Matlab:

- **eps:** es el número 2.2204*10^-16, que es el número real de valor absoluto más chico que utiliza el Matlab. Todos los cálculos en Matlab están hechos con una precisión dada por eps, es decir cualquiera dos números que difieran en eps o menos serán iguales para el Matlab.
- **pi:** el número irracional 3.14159..., que como todos saben es la razón entre toda circunferencia y su diámetro.
- **inf:** infinito (en valor absoluto)
- **realmax:** es el máximo valor positivo representable en computadora, 1.7977*10^308. Mayor que eso produce overflow.
- **realmin:** es el mínimo valor positivo representable en computadora, 2.2251*10^308. Menor que eso produce overflow.

Si asignamos un valor a una variable x, se le puede asignar una función de dicha variable a dicha variable. Por ejemplo:

```
y=2*x
```

Para conocer que variables están activas, podemos utilizar:

```
who
```

o para una lista un poco más detallada sobre las características de cada variable:

```
whos
```

Para borrar la variable activa llamada x, podemos utilizar:

```
clear x
```

o para borrar todas las variables activas:

```
clear
```

Para salvar todas la variables en un archivo llamado variables.mat, en el directorio c:/matlab/work/, usamos:

```
save variables
```

y para cargarlas luego, se usa:

```
load variables
```

Para salvar un conjunto de variables, por ejemplo x e y, en un archivo algunas.mat, en el directorio c:/matlab/work/, usamos:

```
save algunas x y z
```

Para cargar un conjunto de variables de un archivo, por ejemplo x, del archivo algunas.mat:

```
load algunas x
```

También a un archivo se le pueden agregar variables, sin borrar las que ya hemos grabado. Por ejemplo para grabarle la varible h al archivo algunas:

```
save algunas h -append
```

Con escalares se puede operar normalmente con las funciones conocidas:

Operación	Comando
Suma	+
Resta	-
Producto	*
Potencia	^
Transponer	'
División	/
Seno	sin(x)
Coseno	cos(x)
Tangente	tan(x)
Arco Seno	asin(x)
Arco Coseno	acos(x)
Arco Tangente	atan(x)
Exponencial	exp(x)
Logaritmo en base e	log(x)
Logaritmo en base 10	log10(x)

Manipulando Matrices:

definiendo matrices:

Para definir una matriz se utiliza "[" y "]" para marcar principio y fin de la matriz, respectivamente. Para separar valores dentro de la matriz usamos "," para separar valores dentro de la fila (columnas) y ";" para separar las filas. Por ejemplo:

```
a=[1,2,3;4,5,6;7,8,9]
```

La misma matriz se puede producir usando "," para separar valores dentro de la fila (columnas) y enter para separar las filas, por ejemplo:

```
a=[1,2,3
4,5,6
7,8,9]
```

Para referirnos a un valor específico dentro de una matriz utilizamos "(",")", por ejemplo el valor de la matriz en la fila 2 y la columna 1, utilizamos:

```
a(2,1)
```

Un valor de la matriz puede ser cambiado en forma individual:

```
a(2,1)=10
```

De la misma manera, nos podemos referir a toda una columna (fila) de un vector utilizando ":" en la posición de las filas (columnas), indicando que hablamos de todos los valores. Una columna (fila) de un vector será un vector columna (fila).

```
a(2,:)
a(:,3)
```

A partir de matrices, se pueden generar nuevas matrices. Algunos ejemplos son:

```
b=[a;1,2,3]
```

genera una matriz de 4x3.

```
c=[a(1,:);[5,5,5];[6,6,6];a(3,:)]
```

genera una matriz de 4x3.

```
d=[a;a]
```

genera una matriz de 6x3.

```
e=[a,a]
```

genera una matriz de 3x6.

Operaciones con matrices:
Se puede operar con matrices normalmente, siempre respetando el rango de las matrices:

Operación	Operador	Ejemplo
Suma	+	b+c
Resta	-	b-c
Transponer	'	c'
Producto	*	b*c'
Potencia	^	(c'*c)^(2)

Operaciones punto a punto:
Para realizar las operaciones producto, potencia, división y división inversa en formato "punto a punto" se debe anteponer "." a cualquiera de las operaciones. Esto hace que dicha operación se realice elemento por elemento dentro de la matriz. Notar que la suma y resta son operaciones elemento por elemento en matrices y, por lo tanto, no necesitamos usar ".".
Ejemplos: llamamos v1=[1,2,3], v2=[4,5,6]:

Operación	Ejemplo
Producto	v1.*v2 =[1x4,2x5,3x6]=[4,10,18]
Potencia	v1.^2=[1^2,2^2,3^2]=[1,4,9]
División	v1./v2=[1/4,2/5,3/6]=[0.25,0.4,0.5]
División inversa	v1.\v2=[4/1,5/2,6/2]=[4,2.5,2]

El igual que los escalares, las matrices en Matlab pueden tener cualquier nombre largo compuestos por letras. Nuevamente, la excepción son algunas variables ya asignadas automáticamente por Matlab:

- **zeros(n,m):** genera una matriz de ceros de n filas y m columnas.
- **ones(n,m):** genera una matriz de unos de n filas y m columnas.
- **eye(n,m):** genera una submatriz de la matriz identidad de n filas y m columnas.
- **rand(n,m):** genera una matriz de realizaciones de una variable aleatoria uniforme en el intervalo [0,1], de n filas y m columnas.
- **randn(n,m):** genera una matriz de realizaciones de una variable aleatoria normal estándar, de n filas y m columnas

Otras operaciones que son comunes para matrices son las siguientes:

Operación	Operador
Norma	norm(a)

Rango	rank(a)
Determinante	det(a)
Traza	trace(a)
Inversa	inv(a)
Ordenar por columna	sort(a)
Promedio por columna	mean(a)
Desvío Std. por columna	std(a)

Maximizar (Minimizar) dentro de una matriz:
Para obtener un vector fila con el máximo valor de cada columna de una matriz, se utiliza el comando:

```
max(a)
```

Por lo tanto, para obtener un vector columna con el máximo valor de cada fila de una matrix, podemos usar el mismo comando de la siguiente manera:

```
max(a')'
```

Si disponemos un vector, tanto fila o columna, el comando max devuelve el máximo valor dentro de dicho vector. Entonces, para obtener el máximo valor dentro de toda una matriz usamos:

```
max(max(a))
```

En realidad, el comando max admite dos output (como vectores fila), el primero es el valor del maximizador dentro de cada columna y el segundo es la posición del maximizador de cada columna. Entonces:

```
[maxcol,poscol]=max(a)
```

Este comando aloja en el vector fila *maxcol*, los maximizadores de cada columna y en el vector fila *poscol*, la posición dentro de cada columna de dichos maximizadores de cada columna.

```
[maxfil,posfil]=max(a')
```

Este comando aloja en el vector fila *maxfil*, los maximizadores de cada fila y en el vector fila *posfil*, la posición dentro de cada columna de dichos maximizadores de cada fila.
Para minimizar en vez de maximizar, utilizamos la función min en vez de max.

Generando Series

En Matlab se pueden generar grillas de números muy fácilmente, utilizando el operador ":". Se utiliza el siguiente comando genérico:

```
serie = principio:incremento:fin
```

Este comando genera una serie de números reales, que comienza en "principio", que avanza de elemento en elemento en "incremento", hasta que la serie alcanza (o supera) el valor de "fin".

Si queremos que una serie tenga n elementos, comience en principio y termine en fin, debemos utilizar un el comando genérico:

```
serie = principio:(fin-principio)/n-1:fin
```

Graficando en Matlab:

Si tenemos tres vectores de igual dimensión x,y,z y queremos graficar z e y en función de x con un gráfico de línea. Usamos el comando:

```
plot(x,z,'-', x,y,'-')
```

El argumento '-' indica que es un gráfico de línea. Si usaramos '.' haría un gráfico de puntos.

Para ponerle títulos al gráfico y sus ejes, luego a hacer el comando plot, usamos:

```
xlabel('años')
ylabel('PBI')
title('Evolución del PBI')
```

Inclusive podemos usar el valor de una variable previamente definida, por ejemplo h, para utilizar en el título de un gráfico, con el comando num2str:

```
title(['Gráfico n°  ',num2str(h)]);)
```

Para que la serie tenga una leyenda para distinguir las series graficadas, ubicada abajo a la derecha del gráfico:

```
legend('Arg','Bra',4)
```

Para incluirle al gráfico una grilla y ejes, usamos:

```
grid on
axis
```

Ejemplo: graficando el seno y el coseno

```
x=-4:0.1:4
y=sin(x)
z=cos(x)
plot(x,z,'-', x,y,'.')
grid on
axis
legend('sin','cos',4)
xlabel('x')
ylabel('f(x)')
title('Graficando con el Matlab')
```

M-Files:

Las M-Files son programas que el usuario puede crear (y salvar) en Matlab que engloban un conjunto de comandos. Una vez salvado el M-File en el directorio c:/matlab/work/, el usuario puede ejecutar todos los comandos del M-file (si es que el M-file esta bien escrito) en forma instantánea.
Los M-Files son creados y editados en el Matlab Editor/Debugger. Para acceder a él, usamos el menú:

File → New → M-File

Dentro del Matlab Editor/Debugger, tipeamos la primera línea para identificar el programa:

```
function [y,z] =nombre(x,h)
```

Esto indica que generaremos una función llamada nombre.m, que ante los imputs x,h devuelve un vector y,z.

Dentro de un programa, es útil saber que dentro de una línea, todo lo escrito que siga al signo % será ignorado por el programa. Esto será de mucha utilidad para hacer comentarios al programa.

Ejemplo: generando estadísticos "a mano"

```
function [media, varianza]=estadisticos(x)
% calcula la media muestral y varianza muestral de un vector x;
media=sum(x)/length(x);
varianza=(sum(x.^2) /length(x)-media.^2);
```

Una vez salvado el archivo con el nombre ejemplo.m, y si disponemos de un vector x, podemos encontrar su media y varianza muestrales con el comando:

```
[a,b] = ejemplo(x)
```

Loops:

Muchas veces será útil repetir una cierta operación un número determinado de veces, siempre que se cumpla una determinada condición. A esta repetición la llamamos loop. Antes de analizar los tipos de loops existentes, será útil entender los operadores lógicos que podemos utilizar sobre variables de Matlab. Los siguientes son algunos ejemplos:

Operador	Descripción
<	menor que
>	mayor que
< =	menor o igual que
> =	mayor o igual que
= =	igual a
~ =	distinto de
&	y
|	ó
~	no

Para generar loops, existen tres tipos de comandos:

- **If:** repite una determinada operación una vez si se cumple una determinada condición lógica.
- **For:** repite una determinada operación un número finito y predeterminado de veces.
- **While:** repite una determinada operación en forma continua mientras se cumpla una determinada condición lógica.

Los loops pueden ser encadenados uno dentro de otro, es decir, por ejemplo, que dentro un if puede haber un while.
El siguiente cuadro esquematiza el formato de cada uno de los loops:

Tipo de loop		Esquema del loop:
if	if - end	if <condición> <programa> end

	if - else- end	if <condición> <programa 1> else <programa 2> end
	if - elseif- end	if <condición> <programa 1> elseif <programa 2> end
	if - elseif- else- end	if <condición> <programa 1> elseif <programa 2> else <programa 3> end
For		for i=1:n, <programa> end
While		while <condición> <programa> end

Para entender los loops, lo mejor en ver algunos ejemplos simples. Antes de ejecutar los ejemplos, predefinir x = 0, y = 1, z = 2.

Ejemplo: if - else - end.

```
if (x==0 & y==1);
        h=1;
        else;
        h=0;
end;
```

Ejemplo: if - elseif - end.

```
if (x==30 | y==1);
        h=1;
        elseif (z==2);
                h=0;
end;
```

Ejemplo: if - elseif - else - end.

```
if (x==30 | y==1);
        h=1;
                elseif (z~=2);
                        h=0;
         else;
        h=20;
end;
```

Ejemplo: for - end.

```
for i=1:10;
        h(i)=2*i;
end;
```

Ejemplo: for - end.

```
for i=[1,2,5,10];
        m(i)=2*i;
end;
```

Ejemplo: while - end.

```
i=10;
j=10;
while (i>3 |j<=22);
        prod(i)=i*j;
        j=j+1;
        i=i-1;
end;
```

Potpurrí de cosas útiles:

Si tenemos una función f(x) y queremos encontrar un cero de la misma alrededor de algún punto x_0, podemos utilizar la función fzero:

```
fzero(f,x0)
```

Para definir funciones utilizamos el comando inline:

```
f=inline('f(X)')
```

donde f es una función conocida y X es un vector de variables independientes en la función que queremos definir.

Para indicarle al Matlab que debe asignarle el valor de la variable x a un dato que el usuario le debe ingresar a través del teclado, usamos:

```
x=input('¿Cual es el valor de x?')
```

Para indicarle al Matlab que espere hasta que el usuario toque cualquier tecla:

```
waitforbuttonpress
```