PRIMERA SESIÓN: INTRODUCCIÓN A MATLAB/OCTAVE

Esta y todas las demás prácticas están pensadas para ser trabajadas delante de un ordenador con MATLAB u Octave instalado[1], y no para ser leídas como una novela. En vez de eso, cada vez que se presente un comando de MATLAB/Octave, se debe introducir el comando, pulsar la tecla `<intro>` para ejecutarlo y ver el resultado. Más aún, se desea que se verifique el resultado. Asegúrese de que se comprende perfectamente lo que se obtiene antes de continuar con la lectura.

Prerrequisitos: ninguno.

1. Primeros pasos

MATLAB y Octave, los programas, son entornos integrados para el cálculo científico y la visualización de datos. Ambos están escritos en los lenguajes C y C++.

MATLAB está distribuido por The MathWorks (`http://www.mathworks.com`). El nombre proviene de MATrix LABoratory pues originariamente fue desarrollado para el cálculo matricial. Octave, también conocido como GNU Octave (`http://www.gnu.org/software/octave/`), es un software que se distribuye libremente sujeto a los términos de la licencia GPL (GNU Public License).

A lo largo de las prácticas haremos uso frecuente de la expresión "comando de MATLAB"; en ese caso, MATLAB deberá ser entendido como el lenguaje que es el subconjunto común a ambos programas MATLAB y Octave.

Una vez llevada a cabo la instalación, la ejecución de MATLAB y Octave nos da acceso a un entorno de trabajo caracterizado por los indicadores (*prompt*)`>>` y `octave:1>`, respectivamente.

`>>`

`octave:1>`

Después de presionar la tecla `<intro>` (enter, return), todo lo que esté escrito a continuación del indicador será interpretado[2]. De hecho, lo primero que hace MATLAB es comprobar si lo que se ha escrito corresponde a las variables que ya se han definido o al nombre de uno de los programas o comandos definidos en MATLAB.

[1] Concretamente estas prácticas están realizadas para MATLAB 7.0 (R14) y Octave 3.2.3

[2] Por tanto, un programa en MATLAB no necesita ser compilado como ocurre con otros lenguajes (Fortran o C, por ejemplo).

Si todas esas comprobaciones fallan, MATLAB devuelve un aviso de error. En caso contrario, el comando es ejecutado y si procede se mostrará una salida.

Ejemplo 1. Escribe $2+2$ en el indicador de MATLAB, pulsa la tecla intro y observa el resultado:

```
>> 2 + 2
```

Escribe ahora *pepe* y pulsa intro

```
>> pepe
```

¿Qué ha ocurrido?

Finalmente, el sistema devuelve el indicador para poner de manifiesto que está preparado para un nuevo comando. En todo caso, conviene saber que cualquier comando puede abortarse pulsando simultáneamente las teclas control (`<ctrl>`) y c. Para cerrar una sesión de MATLAB se debe escribir el comando `quit` (o `exit`) y pulsar la tecla intro `<intro>`. En adelante se entenderá que para ejecutar un programa o un comando se tiene que pulsar `<intro>`. Además, los términos programa, función o comando se utilizarán de forma equivalente.

<ctrl>+c
quit
exit

Más adelante veremos que la sintaxis de MATLAB no es muy distinta de la de cualquier otro lenguaje; las diferencias serán las habituales, los símbolos de continuación, los comentarios, ... Veamos algunos de estos símbolos:

'-

Las comillas simples o apóstrofes '-' sirven para introducir un texto literal.

Ejemplo 2. Escribe *'pepe'*

```
>> 'pepe'
```

Obsérvese que MATLAB no nos ha devuelto ahora un mensaje de error, sino que ha devuelto como salida la variable por defecto `ans` (abreviatura de answer) al igual que ocurrió en el ejemplo 1 cuando escribimos $2+2$. Si ahora escribimos una cadena de caracteres (número o comando) diferente, la variable `ans` tomará este nuevo valor.

ans

Ejemplo 3.

```
>> 'Bienvenido a la UEx'
```

%

El porcentaje `%` es el símbolo usado para los comentarios. Todo lo que está por detrás de este símbolo en una línea es ignorado por el interprete.

Ejemplo 4.

```
>> 2 + 2   %Esto es un comentario
```

Si añadimos tres puntos ... al final de una línea significa que se tomará la siguiente línea como continuación.

Ejemplo 5.

```
>> 2 + 2 ...
- 3
```

Se pueden escribir varios comandos en una misma línea, separándolos por una coma , o por un punto y coma ; El punto y coma sirve además para inhibir que resultado se muestre en la pantalla.

Ejemplo 6.

```
>> 2 + 2, 2 - 1, ans + 3, 3 + 5;
```

Nótese que, aunque hemos realizado cuatro sumas, sólo se han mostrado los resultados de las tres primeras ya que hemos prohibido que se muestre la salida de la última. No obstante, MATLAB ha interpretado la orden y ha guardado resultado en la variable por defecto.

Ejemplo 7.

```
>> ans
```

2. Atajos de teclado. El comando help

La consola de MATLAB es un entorno de trabajo más potente de lo que parece gracias a una serie de atajos de teclado de gran utilidad.

Ejemplo 1. En la línea de comandos de MATLAB, pulsa la flecha hacia arriba `<up>` una vez ¿qué ha ocurrido? Pulsa ahora la flecha hacia arriba `<up>` varias veces y observa lo que sucede, repite el experimento pulsando también la flecha hacia abajo `<down>`. Puedes recuperar todas tus órdenes anteriores y volver a ejecutarlas si lo deseas.

También se pueden usar las flechas izquierda `<left>` y derecha `<right>` del teclado para mover el cursor a derecha e izquierda y editar así una línea de comandos. De este modo se pueden corregir errores, recuperando un comando con la flecha hacia

arriba y usando a continuación la edición para corregirlo. Como es natural se pueden cambiar tantos símbolos como se quiera o utilizar la tecla de retroceso `<backspace>` para borrrar.

Ejemplo 2.

```
>> sqrt(5) % Raíz cuadrada de 5
```

Utilizando la flecha arriba podemos editar el comando `sqrt(5)`, cambiar el 5 por un 4 y calcular así el valor de la raíz cuadrada de 4.

```
>> sqrt(4)
```

MATLAB posee una amplia colección de mensajes de ayuda. Si conoces el nombre de un comando, se puede encontrar la sintaxis correcta para el comando y ejemplos de su uso escribiendo `help nombre_comando`.

help

Ejemplo 3.

```
>> help sqrt
>> help help
```

Una forma alternativa de obtener ayuda consiste en utilizar la tecla de tabulación `<tab>`:

Ejemplo 4. Escribe `sq` en la línea de comandos y pulsa `<tab>` para obtener la lista funciones y comandos que comienzan por "sq". Obviamente `sqrt` es uno de ellos.

lookfor

Finalmente, el comando `lookfor` nos muestra una lista de comandos en cuyas explicaciones aparezca cadena de texto especificada.

Ejemplo 5.

```
>> lookfor 'square root'
>> help lookfor
```

3. Escalares, operaciones aritméticas y variables

A decir verdad MATLAB no distingue entre escalares y matrices. Para mayor claridad, nosotros sí haremos tal distinción.

Esencialmente podemos utilizar números enteros, reales y complejos:

Ejemplo 1.

```
>> 15, -21  %Son números enteros
>> 0.32, -1.27, .065, -5.32e+5, 4.78e-3  %Son números reales
>> 1-i, 2+3i, -3+j  %Son números complejos
```

La primera observación que debe hacerse es que usar sólo cuatro cifras decimales para representar los números reales y complejos es cuestionable. De hecho, la representación interna del número se hace con 16 cifras decimales, y lo que hemos visto es simplemente uno de los varios posibles formatos de salida de MATLAB (el utilizado por defecto). El mismo número puede tomar diferentes expresiones dependiendo de la declaración específica de formato que se haga. format

Ejemplo 2. Veamos distintas representaciones de 1/7

```
>> format long
>> 1/7
>> format short e
>> 1/7
>> format long e
>> 1/7
>> format short g
>> 1/7
>> format long g
>> 1/7
>> format rat
>> 1/7
```

El formato por defecto es el formato corto (4 cifras decimales). El comando **format** a secas nos devuelve al formato por defecto.

```
>> format
```

El número π está predefinido en MATLAB, pi

```
>> pi
```

Dos representaciones aritméticas predefinidas que pueden ser de utilidad conocer son `Inf` y `NaN`, la primera representa un número más grande que cualquier número que MATLAB pueda almacenar[3] y juega el papel de infinito, mientras que la segunda representa lo que solemos denominar una indeterminación (Not a Number).

Ejemplo 3.

```
>> 1/0
>> 0/0
>> Inf + 1
```

Las convenciones para las operaciones aritméticas en MATLAB son similares a las de cualquier otro lenguaje de programación. El orden de asociación de las operaciones es también el mismo. Primero se operan las funciones matemáticas elementales (senos, cosenos, logaritmos, ...), las multiplicaciones y divisiones y luego las sumas y restas. Como es natural los paréntesis sirven para modificar el orden normal de las operaciones a realizar.

+ - * / ^

Ejemplo 4. Calculemos

$$\frac{1}{\frac{2}{0.1^{1/2}} - \frac{0.4}{2^{1/3}}} - 3(1-2i)$$

```
>>1/((2/0.1^(1/2))-(0.4/2^(1/3)))-3*(1-2i)
```

El comando = permite la asignación de un valor (o de una cadena de caracteres) a una variable dada.

Ejemplo 5. Para asignar la cadena 'Bienvenido a la UEx' a la variable `saludo` podemos escribir:

```
>> saludo = 'Bienvenido a la UEx'
```

Para asignar el valor de resultado de la suma $2+3$ a la variable `a` podemos escribir:

```
>> a = 2+3
```

De este modo no hay necesidad de declarar el tipo de una variable, MATLAB lo hará automática y dinámicamente. Por ejemplo, si escribimos

[3]Los números reales más grande x_{max} y más pequeño x_{min} en MATLAB son `realmax` y `realmin`, respectivamente. El número más próximo a uno dado x es $x + \text{eps}(x)$. Se puede comprobar que los números reales en MATLAB son más densos cerca de x_{min} y menos densos cuando se aproximan a x_{max}

```
>> saludo = 3
```

la variable `saludo` contendrá ahora un número y no una cadena de caracteres. Esta flexibilidad no es gratis. Si ponemos una variable de nombre `quit` igual al número 5 estamos inhibiendo el comando de MATLAB `quit`. Por consiguiente, deberíamos tratar de evitar el uso de variables que tengan el nombre de comandos de MATLAB. Sin embargo, mediante el comando `clear` seguido del nombre de una variable, es posible cancelar esta asignación y restaurar el significado original del comando `quit`.

Ejemplo 6.

```
>> quit = 5
>> quit
>> clear quit
>> quit
```

La función `clear` a secas cancela todas las asignaciones de valores a variables que hayamos realizado.

Los nombres de las variables pueden tener a lo sumo 63 caracteres alfanuméricos (no se admiten ni tildes ni la letra ñ y sí el guión bajo _). Es importante tener presente que el primer carácter siempre debe ser una letra y que para MATLAB las letras mayúsculas son distintas de las minúsculas.

Ejemplo 7.

```
>> a = 5
>> hola = 3
>> Hola
>> hola
>> a_1 = 12
>> 2a = -3
>> a2 = -3
```

El comando `clc` limpia la consola de comandos manteniendo los valores asignados a variables.

Ejemplo 8.

```
>> clc
>> a
```

Como se habrá observado para conocer el valor de una variable previamente definida basta con teclear su nombre. También se pueden conocer todas las variables definidas hasta el momento usando los comandos `who` y `whos`.

Ejemplo 9.

```
>> who
>> whos
>> clear % Cancelamos todas las asignaciones de valores a
variables
>> who
```

4. Operadores de comparación y lógicos. Algunas funciones elementales

Como cualquier lenguaje de programación, MATLAB dispone de operadores que nos permiten comparar valores.

< >

Ejemplo 1.

```
>> 2 < 3
>> 2 > 3
```

Obsérvese que la respuesta ha sido 1 para la afirmación es verdadera y 0 para la falsa. Otros operadores de comparación son <= (menor o igual que), >= (mayor o igual que), == (igual a) y ~= (distinto de). En todos los casos la salida es 1 si la afirmación es cierta y 0 si es falsa.

<= >= ==
~=

Ejemplo 2.

```
>> 2 >= 3
>> 4 >= 3
>> Inf == Inf+1
>> 3 ~= 4
```

Los operadores lógicos de MATLAB son la conjunción `&&`, la disyunción `||`, la negación `~` y la disyunción excluyente `xor`

Ejemplo 3.

```
>> 2 <= 3 && 4 < 5
>> 2 >= 3 && 4 < 5
>> 2 >= 3 || 4 < 5
>> ~(2 <= 3)
>> ~(2 >= 3)
>> xor(2 < 3, 2 > 3)
>> xor(2 < 3, 2 > 1)
```

MATLAB posee una importante biblioteca de funciones elementales cuyos argumentos pueden ser números enteros, reales o complejos, siempre que tenga sentido. El resultado se devuelve en el mismo tipo del argumento.

Algunas funciones elementales soportadas por MATLAB son[4]:

`sqrt`	raíz cuadrada	`sin`	seno
`abs`	valor absoluto o módulo	`cos`	cos
`exp`	exponencial	`tan`	tangente
`log`	logaritmo natural	`asin`	arcoseno
`log10`	logaritmo decimal	`acos`	arcocoseno
`sign`	signo	`atan`	arcotangente

```
sqrt abs
exp sign
log log10
sin asin
cos acos
tan atan
```

Algunas funciones relacionadas con los números complejos son `conj`, `real`, `imag`, `angle` y `compass`.

```
conj
real
imag
angle
compass
```

Ejemplo 4.

```
>> z = 2+3i
>> conj(z)
>> real(z)
>> imag(z)
>> angle(z)
>> compass(z)
```

Ahora podemos utilizar las operaciones y las funciones elementales introducidas arriba para construir nuevas funciones. Para ello definimos lo que se conoce como *funciones anónimas*. Las funciones anónimas responden a la siguiente sintaxis:

[4]Todas las funciones trigonométricas de la tabla operan en radianes.

```
nombre_funcion = @(argumentos) expresión
```

Veamos algunos ejemplos:

Ejemplo 5.

```
>> fun1 = @(x) x^2-1+exp(x)
>> fun1(0)
>> fun2 = @(x,y) cos(x)*sin(y)
>> fun2(pi,pi/3)
>> fun2(pi/3,pi/2)
```

Crea fichero fun.m

```
function y=fun2(x)
y = x^2-1+ex;
```

fichero fun.m

Un ejemplo de una función a trozos podría ser

Ejemplo 6.

```
>> fun3 = @(x) exp(x)*(x<=0) + x*(x>0)
```

5. Gestión de ficheros

Todo trabajo que se realiza durante una sesión MATLAB, se puede escribir en un archivo a modo de *diario*. Este archivo contendrá una transcripción completa y literal de todos los comandos que ha interpretado MATLAB y sus correspondientes respuestas. El contenido del archivo se puede editar con cualquier procesador de textos.

diary

Para iniciar un archivo de diario llamado, por ejemplo, `sesion01.txt` en C:\ debemos escribir

```
>> diary c:\sesion01.txt
```

en la línea de comandos de MATLAB y, como siempre, pulsar `<intro>`. Probablemente no se percibirá ningún cambio, pero el ordenador ha comenzado a registrar todo lo que ocurre en la consola de MATLAB en el fichero `sesion01.txt`

```
>> %Esto se incluirá en nuestro diario
```

En cualquier momento se puede suspender esta orden escribiendo

```
>> diary off
```

Escribiendo ahora `diary on` (o `diary c:\sesion01.txt`) se volverá a guardar toda la sesión en el archivo de diario que previamente se había creado, añadiendo la información nueva a continuación de la ya registrada. Si hay un diario activo al salir de MATLAB, el archivo de diario será cerrado automáticamente.

Al salir de una sesión de MATLAB, los valores de todas las variables se pierden. El archivo de diario sólo mantiene un registro de lo que ha aparecido en la pantalla, no se almacenan los valores de las variables. Si se desea, antes de salir de MATLAB, se pueden guardar los valores actuales de las variables en un archivo para su uso en una sesión futura. El siguiente comando guarda los valores de todas las variables existentes en un archivo que llamaremos `mi_archivo.mat`; por defecto, MATLAB (pero no Octave) añade la extensión `.mat` si no ha sido especificada.

```
>> save mi_archivo.mat
```

save

Los valores de todas las variables se guardarán en forma binaria (ilegible, por tanto, para los seres humanos) en un archivo llamado `mi_archivo.mat` dentro del directorio de trabajo. Para cargar estos valores de las variables en una futura sesión de MATLAB se escribe

```
>> load mi_archivo
```

load

Es posible salvar el valor de una serie de variables determinadas sin necesidad de guardar el resto.

Ejemplo 1.

```
>> a = 1; b = sqrt(2); c = 'no se guarda';
>> save var_ab.mat a b
>> clear
>> a
>> b
>> c
>> load var_ab
>> a, b
>> c
```

Finalmente, podemos pedir a MATLAB que nos indique cuál es el directorio de trabajo actual con la orden `pwd` y su contenido con la orden `dir`. El contenido de un fichero concreto se muestra con el comando `type` seguido del nombre del fichero. También podemos borrar ficheros del directorio de trabajo usando el comando `delete` y cambiar de directorio con el comando `cd` de forma similar a como se hacía en MS-DOS.

pwd
dir
type
delete
cd

Ejemplo 2.

```
>> dir_trab=pwd
>> dir
>> type var_ab.mat
>> delete var_ab.mat
>> dir
>> cd('C:\')
>> dir
>> pwd
>> cd(dir_trab)
>> pwd
>> dir
```

En MATLAB existen funciones especificas para importar datos de hojas de cálculo generadas con otros programas. El abanico de formatos que reconoce MATLAB es mucho más amplio que en Octave. Esencialmente, Octave reconoce bases de datos separados por delimitadores, por ejemplo comas (ficheros `.csv`). Mientras que MATLAB puede leer los datos de hojas de cálculo más sofisticadas, por ejemplo de Excel (ficheros `.xls`).

En la siguiente sesión mostraremos brevemente el uso de las funciones `csvread` y `dlmread` (válidas en MATLAB y Octave) y de `xlsread` (exclusiva de MATLAB).

De forma análoga, los datos numéricos generados en MATLAB se pueden exportar a hojas de cálculo para su lectura y manipulación con otros programas. En este caso, la funciones que veremos serán `csvwrite` y `dlmwrite` (para MATLAB y Octave) y `xlswrite` (sólo para MATLAB).

6. Ejercicios

Antes de comenzar escribe lo siguiente:

```
>> clear all
>> diary a:\sesion01.txt
>> % Sesión 01.  NOMBRE APELLIDOS
```

Ejercicio 1. Asigna a una variable a el número uno. Sobre esa variable calcula $a = 2a$. Repite esta operación hasta alcanzar el valor de 128.

Ejercicio 2.

1. Escribe un número entero menor que 777
2. Divídelo entre 7 (si la división es exacta elige otro número y vuelve dividirlo).
3. Suma los valores de los seis primeros decimales.

Si no nos hemos equivocado te ha salido 27.

Ejercicio 3. Calcula el valor de $((1 + x) - 1)/x$ para $x = 10^{-15}$ (obviamente el resultado es 1 para todo $x \neq 0$).

Ejercicio 4. Define tres variables a, b y c con valores 10^{308}, $1.1 \cdot 10^{308}$ y $-1.001 \cdot 10^{308}$, respectivamente. Compara, usando los operadores de comparación, el valor de $a + (b + c)$ con el de $(a + b) + c$ (evidentemente, por la propiedad asociativa de la suma, son iguales para todo a, b y c).

Ejercicio 5. Define una variable z cuyo valor sea un número complejo no real. Calcula su módulo usando la conocida fórmula $|z| = \sqrt{z \cdot \bar{z}}$ (la barra superior es la conjugación). El módulo de un número complejo se puede calcular de forma directa usando la función `abs`. Compara el valor obtenido antes con el valor que te da la función `abs`.

Ejercicio 6. Usando la ayuda de MATLAB (comando `help`), explica la diferencia entre los comandos `floor, ceil, round` y `fix`. Escribe tus explicaciones con comentarios, %, en la ventada de comandos e ilústrarla con ejemplos en `MATLAB`.

Ejercicio 7. Calcula el coseno de 43 grados [Pista: usa el comando `help` de MATLAB para encontrar la función apropiada]

Ejercicio 8. Define la siguiente función en MATLAB $f(x, y) = x^2 - y^2 - 1$. A continuación usando el comando `help` aprende como se usa la función `ezplot` y dibuja la gráfica de f.

```
>> diary off
```

[1] G. Borrell i Nogueras *Introducción Informal a Matlab y Octave* `http://iimyo.forja.rediris.es/`

[2] R. Echevarria Libano *Breves apuntes para comenzar con MATLAB.* `http://personal.us.es/echevarria/documentos/APUNTESMATLAB.pdf`

[3] J. García de Jalón, J.I. Rodríguez y J. Vidal *Aprenda Matlab como si estuviera en primero* `http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab70/matlab70primero.pdf`

[4] A. Quarteroni y F. Saleri. *Cálculo científico con MATLAB y Octave.* Springer-Verlag Italia, 2006