Capítulo 1

Manejo básico del MATLAB

1.1. Las ventanas en MATLAB

MATLAB es un programa para la realización de cálculos matemáticos y generación de gráficos. Cuenta, además, con un lenguaje de programación propio.

Según abrimos MATLAB, como en todos los programas que trabajan bajo Windows, nos aparecen en pantalla varias barras y ventanas. En la parte superior encontramos la barra de menú al que se puede acceder con el ratón. Por ejemplo si pinchamos en *Help* se accede a una información que nos ayuda en el manejo del programa. Debajo de este menú está la barra de herramientas, en la que se observan algunos iconos que indican diversas tareas que se realizan con frecuencia y, a su derecha, un recuadro con información sobre el directorio de trabajo actual (*Current Directory*), es decir, el lugar donde MATLAB buscará nuestros programas.
La distribución de ventanas en la pantalla depende de la configuración elegida por el usuario. Si en el menú pinchamos en *Desktop*, *Desktop Layout Default* aparecen en pantalla cuatro subventanas. La ventana de la derecha según miramos (*Command Window*) es la que utilizamos para escribir las instrucciones a continuación del siguiente *prompt*

```
>>
```

Por ejemplo, podemos teclear

```
>>  3+5
```

y, luego, para obtener los resultados es necesario teclear la tecla de *cambio de línea* $\hookleftarrow$, que nos devuelve

```
 ans =
     8
```

En la de arriba a la izquierda (*Workspace*) nos aparecen las variables que estamos utilizando y en la de abajo a la izquierda (*Command History*), un historial con los comandos que hemos escrito. Debajo de la ventana *Workspace* está oculta una ventana con el título *Current Directory* donde aparecen los ficheros que hay en la carpeta de trabajo.

Al trabajar en la ventana de comandos conviene tener en cuenta:

- La utilidad de las flechas situadas en el teclado entre el teclado numérico y alfanumérico: Las ordenes que vamos introduciendo en la ventana de comandos se van almacenando y pueden recuperarse con la tecla ↑ . Una vez que tenemos la línea deseada, podemos modificarla, desplazando el cursor sobre ella con las teclas ← y → y ejecutarla de nuevo. También podríamos recuperar una orden ya introducida seleccionándola con el ratón en la subventana correspondiente a *Command History*: pinchando dos veces sobre ella, con lo que se ejecuta la orden o bien arrastrándola con el ratón hasta la ventana de comandos.

  Por contra, no es posible situarse en una línea anterior, hacer una corrección y volver a ejecutarla.

- El `%` sirve para insertar comentarios. Todo lo que escribamos detrás no será tenido en cuenta por MATLAB.

```
>> 2+3 %hemos hecho una suma
ans =
     5
```

- Para finalizar o separar órdenes se utiliza *el salto de línea*, *el símbolo* "**,**" y el *símbolo* "**;**".

```
>>  5+4; 2+9; 9^2,3+2
ans =
    81
ans =
     5
```

- Observa que si inmediatamente detrás de una orden se pone un punto y coma, la orden se ejecuta, pero el resultado no aparece en pantalla.

- La orden **clc** sirve para situar el prompt

```
>>
```

  en la primera línea. Limpia la pantalla pero esta acción no borra de la memoria nada que haya sido creado anteriormente.

1.2. Aritmética Elemental

Para realizar los cálculos numéricos elementales con MATLAB es suficiente conocer la sintaxis de las distintas operaciones:

Suma	Resta	Multiplicación	División	Potenciación
`+`	`-`	`*`	`/`	`^`

Las operaciones se evalúan de izquierda a derecha, la operación potencia tiene el orden de prioridad más alto, seguida de la multiplicación y división, que se encuentran en el mismo orden de prioridad y seguidas finalmente por la suma y la resta, ambas con la misma prioridad. Para alterar la ordenación se pueden utilizar paréntesis en la forma habitual.

Ejemplo 1.2.1 *Halle el valor de* $5^2 - 17 \cdot 2 + 7/5 - \sqrt{12}$
Solución

```
>>  5^2-17* 2 + 7/5 -12^(1/2)
```

Ejemplo 1.2.2 *Obsérvese la diferencia entre las siguientes operaciones:*

$$3^2 - 5\left(2 - \frac{3}{4} \cdot 7\right); \qquad 3^2 - 5 \cdot 2 - \frac{3}{4 \cdot 7}$$

Solución

```
>>  3^2-5*(2-3/4*7)
>>  3^2-5*2-3/(4*7)
```

Ejercicio 1.1 *Hállese el valor de* $7^2 - 17 \cdot 2 + 7/5 - \sqrt{15}$*, recuperando la orden del primer ejemplo con la tecla* $\uparrow$ *y modificando los números correspondientes.*

Ejercicio 1.2 *Pínchese dos veces con el ratón sobre* `3^2-5*(2-3/4*7)`*, en la subventana correspondiente a Command History, para ejecutarla de nuevo.*

Ejercicio 1.3 *Arrástrese con el ratón desde la subventana correspondiente a Command History hasta la ventana de Command Window la orden* `3^2-5*2-3/(4*7)`*.*

1.3. Almacenando datos

1.3.1. Variables

Una variable es un nombre al cual se le asigna un valor. *El símbolo* $=$ es el utilizado para la asignación de valores a las variables. Por ejemplo:

```
>> x=3
x =
     3
```

El nombre de la variable siempre debe ir a la izquierda del símbolo $=$ y el valor a la derecha. Es muy importante tener en cuenta que el valor que se asigna a las variables es permanente. Se guardará hasta que la variable sea borrada o se le asigne un nuevo valor.

```
>> x^2
ans =
     9
```

Asignamos un nuevo valor a **x** :

```
>> x=7
x =
     7
>> x^2
ans =
    49
```

Hasta aquí los cálculos que habíamos hecho se realizaron como en una calculadora. El hecho de poder introducir variables nos ofrece nuevas posibilidades, entre otras:

- Almacenar datos que podamos usar posteriormente:

  Ejemplo 1.3.1 *Hállese* $\sqrt[3]{32 \cdot \frac{59}{23}} + \sqrt[4]{32 \cdot \frac{59}{23}} - 2\sqrt[5]{32 \cdot \frac{59}{23}}$

  Solución

  Podríamos escribir la instrucción

  ```
  >>  a= 32*59/23
  ```

  almacenar en a el número correspondiente al resultado de $32 * \frac{59}{23}$ y después escribir

  ```
  >>  a^(1/3)+ a^(1/4)-2*a^(1/5)
  ```

- Cambiar en una fórmula larga el valor de una de las variables

  Ejemplo 1.3.2 *Si queremos calcular el espacio recorrido por un móvil en movimiento rectilíneo y uniforme de velocidad* $v_0 = 5$ *m/s, para distintos tiempos, es necesario actualizar la variable espacio para cada valor del tiempo:*

  ```
  >>  v0=5, t=1, s=v0*t
  >>  t=3         %Cambiamos el valor de t
  >>  s           %s no se ha actualizado
  >>  s=v0*t      %actualización de s
  ```

  Obsérvese por un lado que en la primera línea se han definido tres variables, sin más que separarlas por comas y por otro que hasta que no se ha actualizado la definición de la variable `s` su valor no ha cambiado.

- Usarlas como un contenedor de datos que puede ir cambiando.

  Ejemplo 1.3.3

  ```
  >>  a=1;
  >>  a=a+4        %suma 4 al contenido de a y almacena el resultado en a
  a
   =5
  ```

Observa:

- Si inmediatamente detrás de una variable se pone un punto y coma, la orden se ejecuta, la variable queda en la memoria, pero el resultado no aparece en pantalla.

```
>>  a=5; t=2; s=0.5*a*t^2
```

- La orden **clc** no borra las variables asignadas.

1.3.2. Reglas para nombrar variables

Las reglas que se utilizan para nombrar las variables son las siguientes:

- MATLAB distingue entre letras mayúsculas y minúsculas. Las variables `area`, `Area`, `AREA`, `arEa` serían variables distintas.
- El nombre de una variable puede contener un máximo de 31 caracteres ignorándose los posteriores.
- El nombre de una variable debe empezar necesariamente por una letra, aunque puede contener letras números y el guión de subrayado, nunca puede contener operadores (+,*,...), espacios en blanco ni signos de puntuación.
- No deben nombrarse variables con nombres con significado específico de funciones o variables predefinidas en MATLAB, por ejemplo `cos`=3 construiría una variable `cos` cuyo valor es `3`, y a partir de este momento no podríamos calcular el coseno de un ángulo hasta que no borrásemos la variable `cos`.

1.3.3. Información sobre las variables

Para obtener información sobre las variables definidas en una sesión de trabajo se utilizan las órdenes **who** y **whos**. La primera muestra las variables que tienen valores asignados, la segunda nos da además información sobre el tamaño y el tipo de dato.

```
>>  who
>>  whos
```

Puede observarse que MATLAB utiliza los escalares como matrices 1×1.

1.3.4. Cómo borrar variables

La orden `clear` se utiliza para borrar todas las variables definidas hasta el momento; si a la orden se le añade una lista de variables (separadas por espacios en blanco) sólo se borrarán las variables de la lista.

```
>>  clear t  % sólo borra t
>>  s=v0*t
>>  who
```

Como la variable `t` ha desaparecido MATLAB da un mensaje de error al recalcular `s` y si le pedimos información sobre las variables que hemos definido en la sesión de trabajo con la orden `who`, no nos aparece la variable `t`.

```
>>  clear a v0  % sólo borra a y v0
>>  who
```

También se pueden borrar variables pinchando con el botón derecho del ratón sobre la variable en la ventana *Workspace* y seleccionando *Delete*.

1.3.5. Algunas variables predefinidas en MATLAB

Algunas variables ya están definidas en MATLAB:

Nombre	**Significado**
`ans`	Almacena el último resultado no asignado a una variable
`eps`	Epsilon de la máquina
`pi`	π
`i` y `j`	Unidad imaginaria
`inf`	∞
`NaN`	No es un número
`date`	Fecha

El épsilon de la máquina es el número positivo más pequeño que sumado a 1 genera un número mayor que 1 en el ordenador, (en un PC `eps`=2.220446049250313e-016) y `NaN` (*Not a Number*) representa una expresión indeterminada, como puede verse en el siguiente ejemplo:

```
>>  (2-2)/(3-3)
```

1.3.6. Tipos de datos

Los tipos de datos que se almacenan en MATLAB son variados. Los más básicos son:

- **Numéricos.** Por defecto se trabaja con números reales. Si es necesario, opera con números complejos que guarda como pareja de números reales.
- **Caracteres.** Formado por cualesquiera de los caracteres del código ASCII. La forma de definir un carácter es encerrarlo entre comillas simples **'a'** (tecla situada a la derecha del 0, no confundir con el acento)
- **Lógicos.** Son los formados por los valores **verdadero (1)** y **falso(0)**. Más adelante hablaremos de ellos.

Ejercicio 1.4

1. *Defínase una variable con el nombre* ***pi*** *y almacénese en ella el valor 6.*
2. *Defínase una variable con el nombre* ***x*** *y almacénese en ella el valor* 5.
3. *Defínase una variable con el nombre* ***vo*** *y almacénese el dato de caracteres* volumen.
4. *Averíguese que variables definidas por el usuario están activas.*
5. *Elimínense las variables* ***x*** *y* ***pi****, dando las ordenes en una sola línea.*
6. *Verifíquese que efectivamente no están activas y que la variable intrínseca* ***pi*** *ha recuperado su valor*
7. *Elimínense el resto de las variables.*

Ejercicio 1.5

1. *Defínase una variable con el nombre a y almacénese en ella el valor* 5^5.
2. *Hállese el valor de:* $\dfrac{5^5}{5^5-1}$.
3. *Modifíquese la orden anterior y calcúlese:* $\dfrac{5^5}{5^5-10}$; $\dfrac{5^5}{5^5-12}$; $\dfrac{5^5}{5^5-21}$

1.4. Precisión y formatos

En MATLAB, la precisión con la que se representan los números reales internamente es siempre la misma y está entre 15 o 16 dígitos. Esta precisión se puede ver en la variable eps, nombrada anteriormente.
El formato de presentación de resultados de Matlab si es modificable por el usuario a través de la orden **format**. Puede verse un listado completo de posibilidades con **help format**. Algunos formatos habituales son: **format short**(formato por defecto), **format long** y **format rat**.

Ejemplo 1.4.1 *Obténgase el valor de* π *en formato largo.*
Solución

Se escribe la instrucción

```
>>  format long
>>  pi
```

Ejemplo 1.4.2 *Obténgase el valor de* $3/4 + 4/9$ *en formato short y rat.*
Solución

Se escribe la instrucción

```
>>  format  %vuelve al formato por defecto
>>  a=3/4 +4/9
>>  format rat
>>  a
```

Conviene recordar que la aritmética que utiliza la máquina solo posee una cantidad finita de números reales. Esto hace que:

- Algunas propiedades de la suma y el producto no se cumplan siempre.

  Ejercicio 1.6 *Realícense las operaciones* $(10^{308} \times 2) \times 0{,}1$ *y* $10^{308} \times (2 \times 0{,}1)$

- Un número sea insignificante frente a otro.

  Ejercicio 1.7 *Compruébese que* $10^{50} + 1$ *y* 10^{50} *son iguales, a pesar de que para la máquina el 1 no es insignificante*

1.5. Archivos de órdenes

Escribir los comandos directamente en la ventana de trabajo de MATLAB resulta cómodo para ejecutar unas pocas órdenes. Sin embargo, cuando tenemos que ejecutar un gran número de instrucciones de forma consecutiva para lograr unos fines, necesitamos crear un *programa*. Normalmente los programas no se escriben directamente en la ventana de comandos sino que se guardan en archivos de ordenes. Para crear un programa hay que usar un *editor* de texto. MATLAB trae uno incorporado que es especialmente útil, ya que resalta en diferentes colores comandos, variables, mensajes, ... Para activarlo se puede pinchar con el ratón en el menu *File*, elegir *New* y luego *M-file* o en el icono correspondiente de la barra de herramientas. Se abre la ventana del editor de textos de MATLAB donde escribimos las instrucciones que MATLAB pueda entender, como si estuviéramos en la línea de comandos de la ventana principal. Luego se guarda el archivo con un nombre que el programa almacena con la extensión **.m**. El nombre del fichero sigue las mismas reglas que los nombres de variables. Para ejecutar un fichero basta con escribir en la *Ventana de Comandos* el nombre del archivo. También se puede ejecutar desde la *Ventana del Editor* pinchando sobre el icono *Run*.

Ejemplo 1.5.1

```
% En las primeras líneas, precedidas del símbolo % podemos poner la ayuda
% o descripción del fichero.
x=4;y=3;% Hemos definido dos variables.
% Sumaremos sus cuadrados
S=x^2+y^2
```

Lo guardamos en nuestro directorio, con un nombre válido: **prueba1**. Para ejecutarlo volvemos a la ventana de trabajo y seleccionamos en la barra de arriba (recuadro *Current Directory*) la carpeta donde está el fichero. Tecleamos el nombre del programa, sin extensión,

```
>>  prueba1
```

y MATLAB ejecuta las tres órdenes de manera consecutiva (se salta los comentarios) y nos devuelve sólo el valor de `S`.

Nota: Recuérdese que si ponemos punto y coma (`;`) al final de una línea, la orden se ejecuta, la variable queda guardada en memoria, pero no se ve nada en pantalla.

Los valores de `x`, `y` y `S` están en memoria (de hecho, deberían aparecer en la ventana de las variables).

1.5.1. Entrada y salida de datos por pantalla

En el archivo que acabamos de crear, los datos utilizados, $x = 4$ e $y = 3$, se introducen en el propio fichero, el cual sólo nos permite calcular la suma de éstos dos números. Si quisiéramos sumar otro par de números deberíamos modificar los datos escritos en el mismo. Sin embargo, en los ficheros de comandos también podemos escribir programas más generales que cada vez que se ejecuten nos pidan, por pantalla, los datos que debemos asignar a las variables definidas en el fichero. Para hacer esto se utiliza el comando **input**. La forma de utilizar este comando es la siguiente:

```
a=input('Mensaje pidiendo un dato en la ventana de comandos')
```

Al ejecutar el fichero, en la *Ventana de Comandos* aparecerá el mensaje pidiendo datos y el programa espera a que el usuario teclee el dato que desea introducir. Introducido éste, tras pulsar la tecla de *cambio de línea* $\hookleftarrow$, su valor se almacenará en la varible **a** y, a continuación, se ejecuta el resto del programa.

Ejemplo 1.5.2

```
% Este fichero permite sumar los cuadrados de dos números reales cualesquiera
x=input('Introduce el primer número ');
y=input('Introduce el segundo número ');
S=x^2+y^2
```

Lo guardamos en nuestro directorio, con el nombre : **prueba2**.

Ejercicio 1.8

Escríbase un programa en un archivo, de nombre ***producto****, que dados tres números reales cualesquiera calcule su producto.*

A menudo nos interesará mostrar en la pantalla un mensaje de texto o el valor de una variable sin que aparezca su nombre. Para ello MATLAB dispone del comando **disp**, cuyo funcionamiento es el siguiente:

`disp` ('Mensaje de texto')

o bien

`disp`(x)

siendo x el nombre de una variable.

Ejemplo 1.5.3

```
% Voy a crear ahora un programa que me sirva para sumar dos
% números cualesquiera pidiéndolos por pantalla con la orden input.
x=input('Introduce el primer número, x= ');
y=input('Introduce el segundo número, y= ');
% Voy a escribir un mensaje, con la orden disp
disp('Su suma es:')
S=x+y;
% Pido que muestre en pantalla la variable numérica S
disp(S) %Sin comillas por ser una variable
```

Lo guardamos en nuestro directorio, con el nombre : **prueba3**.

Ejemplo 1.5.4 *Según el algoritmo que da las dos posibles soluciones de una ecuación de segundo grado, escribe un programa en un archivo, de nombre* ***sol2grado****, que calcule las soluciones x1 y x2. Utilícese para calcular las soluciones de la ecuación* $2x^2 - 3x + 1 = 0$.
Solución

```
a=input('Introduce el coeficiente de x^2 de la ecuación de segundo grado: a=');
% orden para introducir un dato por pantalla
b=input('Introduce el coeficiente de x de la ecuación de segundo grado: b=');
c=input('Introduce el término independiente de la ecuación de segundo grado:c=');
disp('Las soluciones son:') % orden para presentar un mensaje en pantalla
x1=( -b+sqrt(b^2-4*a*c))/(2*a), x2 =( -b-sqrt(b^2-4*a*c))/(2*a) % La orden sqrt(x)
% devuelve la raiz cuadrada de x.
```

Una vez creado el archivo **sol2grado** se ejecuta con la orden

```
>> sol2grado
Introduce el coeficiente de x^2 de la ecuación de segundo grado: a=2
Introduce el coeficiente de x de la ecuación de segundo grado: b=-3
Introduce el término independiente de la ecuación de segundo grado:c=1
Las soluciones son:
x1 =
    1
x2 =
   0.5000
```

Obsérvese que no se introduce la extensión .m

Recuerda:

Función	**Salida**
`a=input('Mensaje')`	Almacena en a un dato introducido por teclado, mostrando en pantalla la cadena mensaje.
`disp('mensaje')`	Presenta en pantalla la cadena mensaje.
`disp(x)`	Presenta la variable x

Ejercicio 1.9 *Constrúyase un fichero de nombre* ***ejercicio1_9*** *que pida por pantalla dos números reales y devuelva su suma y su producto. Utilícese para calcular la suma y el producto de los números 4 y 9.*

Ejercicio 1.10 *Escríbase un programa en un archivo, de nombre* ***Areacorona*** *que calcule el área de una corona circular, introducidos por pantalla los radios. Pruébese para el caso de una corona de radios r1=12, r2=21.*