Práctica 1

Introducción a MATLAB

1. Algunas observaciones preliminares

- Las órdenes se escriben en la ventana de comandos (*Command Window*), que es la ventana en la que aparece el *prompt* de MATLAB (`>>`). Se ejecutan pulsando la tecla ↵ .
- Se pueden escribir y ejecutar varias órdenes en una misma linea separándolas por comas. Por ejemplo

```
>> base=2, altura=4, area=base*altura
```

- También se pueden separar por punto y coma. En este caso se inhibe el eco que sigue a la ejecución de la instrucción correspondiente:

```
>> base=2; altura=4; area=base*altura
```

- Las teclas ↑ , ↓ permiten recuperar líneas anteriores o posteriores a la actual; ← , → permiten moverse dentro de una línea.
- Para obtener ayuda de una orden se escribe en la ventana de comandos

`>> help orden` o bien `>> helpwin orden`

Con la primera opción la información se muestra en la ventana de comandos. Con la segunda se obtiene información más completa, en formato más adecuado para imprimir, en una ventana diferente.

2. Variables

El operador de asignación

Para asignar un valor numérico a una variable se utiliza el operador de asignación `=`. Por ejemplo, para asignar a la variable `x` el valor numérico 3 se escribe, en la ventana de comandos,

```
>> x=3
```

Es importante destacar que esta expresión no indica que x sea igual 3. Respecto a la asignación de un valor a una variable es interesante tener en cuenta lo siguiente.

`3=x` Esta expresión es incorrecta porque el orden de escritura es: variable, operador de asignación, valor de la variable.

`x=x+2` Si x tiene un valor asignado previo, por ejemplo $x = 3$, la expresión es correcta produce como resultado la asignación del valor 5 a x.

Reglas para nombrar variables

- Las letras mayúsculas y minúsculas son distintas a efectos de nombrar variables. Por ejemplo, son diferentes las variables `base, Base, baSe, BASE`.
- El nombre de una variable puede tener hasta 31 caracteres; si hubiese más serían ignorados.
- El nombre de una variable debe comenzar obligatoriamente por una letra. Puede contener letras, números y el guión de subrayado `(_)`; no se permiten espacios en blanco.
- No es conveniente nombrar variables mediante expresiones que tengan un significado específico en MATLAB. Por ejemplo, no es aconsejable utilizar `log` como nombre de una variable ya que esta es la designación de la función logarítmica en MATLAB.

Algunas variables de MATLAB

`ans`	Es la variable que, por defecto, contiene los resultados de los cálculos.
`pi`	Contiene el valor del número real π.
`eps`	Es el número positivo más pequeño que sumado a 1 genera un número mayor que 1 en el ordenador.
`Inf`	Represente el valor infinito. Se obtiene, por ejemplo, en el caso de *overflow* o división por cero.
`Nan`	(Not a Number) Representa una expresión indeterminada, por ejemplo, 0/0
`i, j`	Representan la unidad imaginaria, $i = j = \sqrt{-1}$.

Información sobre las variables

Para conocer el valor que tiene asignado una variable basta escribir su nombre en la ventana de comandos. Para obtener información acerca de las variables definidas en el espacio de trabajo se pueden utilizar las órdenes:

`who`	Muestra las variables que tienen valores asignados.
`whos`	Hace lo mismo y añade información sobre el tamaño y el tipo de dato.

Borrar variables

`clear x y`	Borra las variables x e y.
`clear all`	Borra todas las variables.

3. Operaciones algebraicas y orden de prioridad

En la tabla siguiente se indican los signos que se utilizan en Matlab para indicar las operaciones algebraicas.

POTENCIACIÓN	^
MULTIPLICACIÓN Y DIVISIÓN	* /
SUMA Y RESTA	+ −

El orden de prioridad de las operaciones decrece de arriba abajo. Las operaciones situadas en la misma linea tienen el mismo orden de prioridad realizándose en ese caso de izquierda a derecha. El orden de prioridad puede modificarse mediante el uso de paréntesis redondos.

4. Ficheros m de órdenes (*scripts*)

En Matlab los cálculos se pueden realizar en la ventana de comandos (ventana el la que aparece el signo >>). Otra posibilidad es crear un fichero m que se ejecuta desde la ventana de comandos. Este último procedimiento es el aconsejable cuando los cálculos son complejos. Para crear y ejecutar un fichero m se siguen los siguientes pasos:

- Se abre el editor de Matlab, pulsando el icono "folio en blanco".
- Se escriben los cálculos que se quieran realizar.
- Se guarda el fichero pulsando el icono "disquete".
- Se ejecuta escribiendo el nombre en la ventana de comandos y pulsando la tecla "intro".

Todo lo que aparece la derecha del signo % en una línea, dentro de un fichero m, es ignorado cuando se ejecuta el fichero. Esto permite introducir comentarios.

Ejercicio 1

Se considera un cilindro de altura 15 cm y cuya base tiene un radio de 4 cm. Crear un fichero m en el que

(a) Se definen las variables radio de la base y altura del cilindro y se les asignan los valores indicados.

(b) Se calcular el valor del volumen del cilindro, su área lateral y su área total (área lateral más área de la base y de la tapa).

(c) Poner un comentario al comienzo del fichero explicando la finalidad del mismo, el autor y la fecha de ejecución. Poner también algún comentario explicando el significado de alguna de las variables.

5. Funciones predefinidas

En la tabla de la izquierda se muestra la notación de Matlab para las funciones elementales. En la de la derecha se indica la correspondencia con la notación habitual en Matemáticas para algunas funciones.

`exp`	`sin`	`asin`	`sinh`	`asinh`
`log`	`cos`	`acos`	`cosh`	`acosh`
`log10`	`tan`	`atan`	`tanh`	`atanh`
`log2`	`cot`	`acot`	`coth`	`acoth`
`sqrt`	`sec`	`asec`	`sech`	`asech`
	`csc`	`acsc`	`csch`	`acsch`

MATLAB	MATEMÁTICAS
`exp(x)`	e^x
`log(x)`	$\ln(x)$
`log10(x)`	$\log_{10}(x)$
`log2(x)`	$\log_2(x)$
`sqrt(x)`	$\sqrt{x}$
`sin(x)`	$\mathrm{sen}(x)$
$\vdots$	$\vdots$

Ejercicio 2

Calcular, utilizando un fichero m, las siguientes expresiones:

$$a = \mathrm{e}^{\frac{5+\ln 2}{3}}, \qquad b = \log_3(\sqrt[5]{5} + \sqrt{2}), \qquad c = \mathrm{sen}(328^\circ + \mathrm{e}).$$

6. Vectores

Para definir un vector $x = (a,\ b,\ c)$ en MATLAB basta escribir sus coordenadas entre corchetes, separadas por espacios en blanco o comas: Si las coordenadas se separan con punto y coma x queda definido como vector columna.

`x=[a b c]` Define el vector x de coordenadas $a,\ b,\ c$ como vector fila.
`x=[a, b, c]` Produce el mismo resultado.
`x=[a; b; c]` Define el vector columna x de coordenadas $a,\ b,\ c$.

Definición de vectores especiales

```
x=[a:h:b]
```

Define un vector $x = (a, a+h, a+2h, \ldots a+kh)$, donde k es el mayor entero tal que $a+kh \leq b$. Los corchetes pueden sustituirse por paréntesis o suprimirse. Cuando $h = 1$ se puede escribir `x=[x1:xn]` en lugar de `x=[x1:1:xn]`.

```
x=linspace(a,b,n)
```

Genera un vector de n coordenadas, cuya primera coordenada es a y cuya última coordenada es b. Produce el mismo efecto que `x=[x1:(xn-x1)/(n-1):xn]`.

Es posible definir un vector sin ninguna coordenada. Esto se hace con la orden

```
x=[ ]
```

Acceso a las coordenadas de un vector

Si v es un vector definido previamente, se puede acceder a sus coordenadas para conocer su valor, utilizarlo en otros cálculos o modificarlo.

`v(i)`	Devuelve la coordenada i-ésima.
`v(i:h:j)`	Devuelve las coordenada que van desde la i hasta la j, con incremento h.
`v(i:j)`	Hace los mismo que `v(i:1:j)`.
`v([i,j,k])`	Devuelve las coordenadas $i,\ j,\ k$.
`v(end)`	Devuelve la última coordenada.

Ejercicio 3

Definir los siguientes vectores:

$$x = (1, -2, 3, 7), \quad y = (2, 3, 4, 5, 6, 7, 8, 9), \quad z = (2, 3, 4, 500, 6, 7, 8, 9), \quad u = (12, 10, 8, 6, 4, 2),$$

$$v = (2, 3, 4, 5, 6, 7, 7, 9, 11, 13, 15).$$

Operaciones algebraicas

`a*x`	Producto del escalar a por el vector x.
`x+y`	Suma de los vectores x e y.
`x-y`	Diferencia de los vectores x e y.
`dot(x,y)`	Producto escalar de los vectores x e y.
`cross(x,y)`	Producto vectorial de los vectores x e y de tres coordenadas.

Operaciones elemento a elemento

Si $x = (x_1, x_2, \ldots, x_n),\ y = (y_1, y_2, \ldots, y_n)$ dos vectores y a un escalar. MATLAB permite definir las siguientes operaciones, denominadas operaciones *elemento a elemento*.

`a+x`	Devuelve el vector $(a + x_1, a + x_2, \ldots, a + x_n)$.
`x.*y`	Devuelve el vector $(x_1y_1, x_2y_2, \ldots, x_ny_n)$.
`x./y`	Devuelve el vector $(\frac{x_1}{y_1}, \frac{x_2}{y_2}, \ldots, \frac{x_n}{y_n})$.
`x.\y`	Devuelve el vector $(\frac{y_1}{x_1}, \frac{y_2}{x_2}, \ldots, \frac{y_n}{x_n})$.
`x.^a`	Devuelve el vector $(x_1^a, x_2^a, \ldots, x_n^a)$.
`x.^y`	Devuelve el vector $(x_1^{y_1}, x_2^{y_1}, \ldots, x_n^{y_n})$.

Otras órdenes

`length(v)`	Muestra el número de coordenadas de v.
`sum(v)`	Suma las coordenadas de v.
`prod(v)`	Multiplica las coordenadas de v.
`max(v)`	Devuelve la coordenada mayor.
`min(v)`	Devuelve la coordenada menor.

7. Matrices

Las matrices se definen de forma parecida a los vectores. Los elementos de una fila se separan con espacios en blanco o comas y las filas con punto y coma. De acuerdo con esto, para definir la matriz

$$a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

se escribe

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

Definición de matrices especiales

`ones(m,n)`	Genera una matriz $m \times n$ de unos.
`zeros(m,n)`	Genera una matriz $m \times n$ de ceros.
`eye(m,n)`	Genera una matriz $m \times n$ con unos en la diagonal principal y ceros en el resto.
`rand(m,n)`	Genera una matriz $m \times n$ de números aleatorios, con distribución uniforme en $(0, 1)$.

En todos los casos, si la matriz es cuadrada se puede sustituir (m, n) por (n).

Otras formas de definir matrices

`X=[A B; C D]`	Define la matriz X por cajas, a partir de matrices $A_{m\times n},\ B_{m\times p},\ C_{r\times n},\ D_{r\times p}$.
`A=diag([a,b,c])`	Define una matriz diagonal a partir del vector de los elementos de su diagonal.

Ejercicio 4

Definir las siguientes matrices:

(a) Una matriz a de ceros de, dimensión 3×4.
(b) Una matriz b de unos de dimensión 3×3.
(c) Una matriz c de números aleatorios, de dimensión 3×4.
(d) Una matriz diagonal d cuya diagonal principal es $-1,\ 2,\ 5$.
(e) Una matriz e definida por bloques a partir de las matrices $a,\ b,\ c$ y d. (a y b en la primera fila; c y d en la segunda).

□

Acceso a los elementos de una matriz

Una vez definida una matriz se puede acceder a sus elementos para conocer sus valores, utilizarlos en otros cálculos o modificarlos. Para ello se dispone de las órdenes siguientes.

`A(i,j)`	Devuelve el elemento (i, j) de la matriz A.
`A(i1:h:i2,j1:k:j2)`	Devuelve la submatriz de A formada por las filas de la i_1 a la i_2 con incremento h y las columnas j_1 a la j_2 con incremento k.
`A(i1:i2,j1:j2)`	Devuelve la submatriz de A formada por las filas de la i_1 a la i_2 y las columnas j_1 a la j_2.
`A(i1:h:i2,:)`	Devuelve la submatriz de A formada por las filas de la i_1 a la i_2 con incremento h.
`A(i1:i2,j)`	Devuelve el elemento j de las filas comprendidas entre la i_1 y la i_2.
`A(i,j1:j2)`	Devuelve el elemento i de las columnas comprendidas entre la j_1 y la j_2.
`A(:,j)`	Devuelve el elemento j de todas las filas, o sea, la columna j.
`A(i,:)`	Devuelve el elemento i de todas las columnas, o sea, la fila i.

Información sobre las dimensiones

`size(A)`	Genera un vector con las dimensión (número de filas y columnas) de la matriz A.
`[m n]=size(A)`	Asigna a m el número de filas de A y a n el número de columnas.
`size(A,1)`	Devuelve el número de filas de la matriz A.
`size(A,2)`	Devuelve el número de columnas de la matriz A.

Ejercicio 5

(a) Definir la matriz

$$a = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix}$$

(b) Extraer de a, las matrices

$$b = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix}, \qquad c = \begin{pmatrix} 2 & 3 & 4 \\ 2 & 2 & 2 \end{pmatrix}$$

(c) Determinar las dimensiones de a, b, c.

Operaciones algebraicas

Si A y B son matrices con dimensiones adecuadas y k un escalar, las operaciones algebraicas en MATLAB se efectúan con las las siguientes órdenes:

`k*A`	Multiplica por k todos los elementos de A.
`A+B`	Suma A y B.
`A-B`	Resta A y B.
`A*B`	Multiplica A por B.
`inv(A)`	Calcula la inversa de una matriz cuadrada A.
`det(A)`	Calcula el determinante de una matriz cuadrada A.
`A^n`	Eleva la matriz A al número entero n.
`A.'`	Genera la traspuesta de A.
`A'`	Genera la conjugada traspuesta de A. Si A es una matriz real A' y $A.'$ coinciden.

Operaciones elemento a elemento

Al igual que en el caso de los vectores, en MATLAB se definen otro tipo de operaciones a las que se puede denominar *operaciones elemento a elemento*. Los operadores empleados son los de las operaciones algebraicas anteponiendo un punto (.), excepto en el caso de la suma de un escalar y una matriz.

Si $A = (a_{ij})$ y $b = (b_{ij})$ son matrices de iguales dimensiones y k un escalar se tiene:

`k+A`	Suma k a cada elemento de la matriz A.
`A.*B`	Multiplica A por B elemento a elemento, es decir, define la matriz cuyo elemento (i, j) es $a_{ij}b_{ij}$.
`A./B`	Divide cada elemento de A por el correspondiente de B.
`A.\B`	Hace los mismo que `B./A`.
`A.^k`	Eleva cada elemento de A a k.
`k.^A`	Eleva k a cada elemento de A.
`A.^B`	Eleva cada elemento de A al correspondiente de B.

Funciones elementales con argumento matricial

Si el argumento de una función elemental de MATLAB es una matriz, la función se aplica a cada elemento de la matriz. Es decir:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \qquad \rightarrow \qquad f(A) = \begin{pmatrix} f(a_{11}) & \cdots & f(a_{1n}) \\ \vdots & \ddots & \vdots \\ f(a_{m1}) & \cdots & f(a_{mn}) \end{pmatrix}$$

Otras funciones de argumento matricial

`trace(A)`	Calcula la traza de la matriz A.
`max(A)`	Devuelve el mayor elemento de cada columna de A.
`min(A)`	Devuelve el menor elemento de cada columna de de A.

8. Definición de funciones

8.1. Ficheros m de función

Funciones con un argumento de entrada y uno de salida

Los ficheros m que se han comentado hasta ahora se suelen denominar *scripts* (guiones), para distinguirlos de los ficheros m que definen una función. Para definir una función $y = f(x)$ en un fichero m se procede así:

- En la primera linea se escribe la instrucción `function y=f(x)`.
- En la segunda linea se especifican las operaciones que se realizan con la `x` para obtener la `y`. Por ejemplo: $y = x^2, y = \operatorname{sen} x + x^3, \ldots$ o lo que corresponda en cada caso.
- Se da al fichero el mismo nombre que a la función, es decir `f.m`.
- En la ventana de comandos se puede evaluar la función en un punto x, escribiendo `>>f(x)`.

Las variables definidas dentro de un fichero de función son locales. Es decir, si la función es llamada desde un programa, las variables de la función no interfieren con las del programa.

Cada fichero de función define una sola función que puede ser llamada desde un programa exterior. Pero puede contener otras funciones que se llaman desde dentro de la función.

Ejercicio 6

Definir las funciones siguientes en sendos ficheros m de función:

(a) $f(x) = x^2 + \ln 2x$. Hacer las modificaciones necesarias para que el argumento x pueda ser vectorial.
(b) $f(x) = \log_3 x$.
(c) $f(x) = \operatorname{sen}(x^\circ)$, donde x° indica que el argumento está en grados sexagesimales.

Funciones con varios argumentos de entrada y de salida

Una función puede tener n argumentos de entrada y m argumentos de salida. La estructura de una función en el caso general es:

```
[y1,...,ym]=f(x1,...,xn)
  y1=f1(x1,...,xn);
          ⋮
  ym=fm(x1,...,xn);
```

También se admiten funciones sin argumentos de salida.

Ejercicio 7

(a) Escribir una función cuyos argumentos de entrada sean el radio de la base y la altura de un cilindro, y cuyo argumento de salida sea el volumen del cilindro.
(b) Modificar la función para que calcule además el área lateral y el area total. (Área lateral más el área de la base y de la tapa.)

8.2. Funciones *inline*

Una función $f(x)$ se puede definir como un objeto inline con la sintaxis:

```
f=inline('expresión')
```

donde `'expresión'` es una cadena de caracteres que define la función.

Por ejemplo, para definir la función $f(x) = \dfrac{x^2}{\operatorname{sen} x}$ como objeto inline se escribe

```
f=inline('x.^2/sin(x)')
```

Para evaluar esta función en un punto, por ejemplo en $x = 2$, basta escribir `f(2)`.

9. Gráfica de una función

Ventanas gráficas

Los gráficos de MATLAB se muestran en una ventana gráfica, en cuya barra de título aparece el nombre `Figure n`, donde n es un número natural.

Por defecto el gráfico se representa en Figure 1, si se quiere utilizar una ventana gráfica distinta hay que indicarlo explícitamente con la orden `figure(n)`, donde n es un número natural.

La orden `plot`

La orden básica para crear gráficos es

```
plot(x,y)
```

- Si x e y son números, dibuja el punto de coordenadas (x, y).
- Si x e y son vectores $x = [x_1, \ldots, x_n]$ e $y = [y_1, \ldots, y_n]$, dibuja la poligonal $(x_1, y_1), \ldots, (x_n, y_n)$.

Por defecto, la orden `plot(x,y)` utiliza el color azul, marca los puntos con un punto pequeño (.) y realiza el trazo con linea continua fina. Estos valores por defecto se pueden cambiar utilizando la orden `plot` con parámetros opcionales, cuya sintaxis es

```
plot(x,y,'cmt')
```

donde `c, m, t` son 3 parámetros opcionales que indican, respectivamente, el color, la marca de los puntos y el trazo de la línea de unión de los puntos. Sus valores se pueden consultar en la tabla que se muestra en la ayuda de la orden `plot`. El orden de los tres parámetros es arbitrario; pueden definirse solo uno o dos.

De entre la gran cantidad de órdenes relacionadas con los gráficos mencionamos solo las tres que siguen:

`hold on`	Permite hacer un nuevo gráfico sobre una ya creado. La opción por defecto es `hold off` que hace que cada gráfico, en una misma figura, elimine al anterior.
`grid on`	Activa una malla. Se desactiva con `grid off`, que es la opción por defecto.
`axis[xmin,xmax,ymin,ymax]`	Impone los rangos de variación en el eje de abscisas y en el de ordenadas.

Ejercicio 8

Se consideran cuatro puntos de coordenadas $(1, 1)$, $(2, 2)$, $(3, 1)$, $(4, 2)$.

(a) Representar gráficamente utilizando la orden `plot` sin parámetros opcionales.
(b) Representar gráficamente los puntos utilizando asteriscos rojos como marcas.

La orden `fplot`

En ocasiones puede resultar más cómodo utilizar la orden

```
fplot('f',[a,b],'cmt')
```

donde

`f`	puede ser el nombre de un fichero m que define la función correspondiente, o una cadena de caracteres que define la función.
`[a,b]`	es intervalo donde se quiere representar la función.
`cmt`	son los parámetros opcionales comentados para la orden `plot`.

Ejercicio 9

Se considera la función

$$f(x) = \frac{x}{1+x^2}, \quad x \in [-10, 10]$$

(a) Representarla gráficamente en el intervalo indicado utilizando la orden `plot`.
(b) Repetir en una nueva ventana gráfica utilizando la orden `fplot`.
(c) Hacer el gráfico en en color rojo, en una nueva figura. Representar la malla y hacer que el eje OX vaya de -12 a 12 y el eje OY de $-1{,}5$ a 1,5.

Subdivisión de una ventana gráfica

Una ventana gráfica se puede subdividir en subventanas más pequeñas con la orden

```
subplot(m,n,p)
```

Con esta orden la ventana gráfica se divide en $m \times n$ subventanas distribuidas en m filas y n columnas. p indica la posición de la subventana actual, contando de izquierda a derecha y de arriba abajo.

Ejercicio 10

Se consideran las funciones

$$f_1(x) = x^2, \quad f_2(x) = x^3, \quad f_3(x) = x^4, \quad x \in [-2, 2].$$

(a) Representar cada una de las funciones f_1, f_2 y f_3 una subventana.
(b) Superponer los gráficos de las tres funciones en una cuarta subventana. Utilizar diferentes colores para las diferentes funciones.

Práctica 2

Programación con MATLAB

1. Entrada y salida de datos

En cualquier programa, por sencillo que sea, es necesario introducir y extraer datos. Las dos instrucciones que se describen a continuación permiten introducir datos desde el teclado y mostrarlos en pantalla, lo que permite construir pequeños programas interactivos.

```
x=input('mensaje')
```

Esta instrucción muestra en pantalla el 'mensaje' que hayamos escrito para que nos indique la acción que debemos hacer desde el teclado. Mediante esta acción se asigna la variable x un valor (escalar, vectorial, matricial,...) con la sintaxis adecuada a cada caso.

Para mostrar un resultado en pantalla basta invocar en el programa el nombre de la variable. Otra posibilidad, que interesa a veces, es utilizar la instrucción `disp` en alguna de las formas que se indican a continuación:

- `disp(x)` Muestra en pantalla el valor de la variable `x`.
- `disp('mensaje')` Muestra en pantalla el mensaje indicado.
- `disp([a b c])` Muestra en pantalla el vector (a, b, c).

Ejercicio 1

Escribir un programa que pida el radio de la base y la altura de un cilindro y que devuelva su volumen.

2. Estructuras repetitivas (bucles)

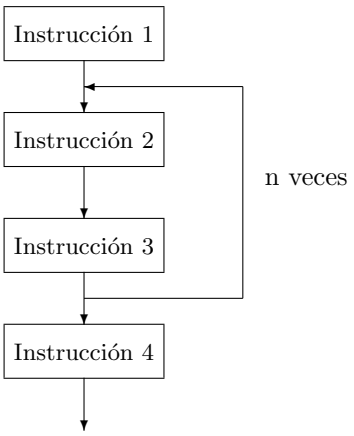
La estructura de flujo de un programa indica la forma en que se ejecutan las instrucciones. Se distinguen tres estructuras de flujo:

- Estructura secuencial: las instrucciones del programa se ejecutan una sola vez en el orden en el que están escritas, de izquierda a derecha y de arriba abajo.
- Estructura repetitiva: algunas instrucciones se ejecutan varias veces.
- Estructura condicional: algunas instrucciones se ejecutan o no dependiendo de que verifiquen ciertas condiciones.

A continuación se analizan las estructuras repetitivas. Para su estudio se dividen en bucles simples y bucles anidados

2.1. Bucles simples

Las estructuras repetitivas, o bucles, se pueden representar esquemáticamente mediante un diagrama de flujo del tipo:



En MATLAB se obtienen con la siguiente sintaxis:

```
for i = [i_1, ..., i_n]
        conjunto de instrucciones
end
```

Cuando en un programa aparece una instrucción como la anterior, el conjunto de instrucciones que aparece entre el `for` y el `end` se ejecuta n veces en la forma siguiente:

$i = i_1$
conjunto de instrucciones
$i = i_2$
conjunto de instrucciones
$i = i_3$
conjunto de instrucciones
$\vdots$
$i = i_n$
conjunto de instrucciones

Es decir, el conjunto de instrucciones se repite tantas veces como coordenadas tiene el vector i. En cada repetición la variable i toma el valor de la correspondiente coordenada del vector por lo que, si la variable se utiliza dentro del conjunto de instrucciones, las diferentes repeticiones no producen el mismo efecto.

Ejercicio 2

(a) Escribir un programa que, al ejecutarlo, escriba en pantalla la frase: "Primer programa"

(b) Modificar el programa para que la escriba 20 veces.

(c) Modificar el programa para que numere cada una de las 20 copias[1].

(d) Modificar el programa para que, en cada ejecución, pida el número de copias.

Ejercicio 3

(a) Escribir un programa que calcule la suma $1 + 2 + 3 + \cdots + 100$.

(b) Modificar el programa para que calcule la suma de los números pares hasta 100.

(c) Modificar el programa para que calcule la suma de los cubos de los números pares hasta 100.

[1]Para conseguir que la numeración aparezca en la misma linea que la frase, se puede utilizar la orden `num2str(i)` que convierte la variable numérica `i` en cadena de caracteres.

Ejercicio 4

(a) Escribir un programa que calcule el factorial de 5.

(b) Modificar el programa para que pida un número y calcule su factorial.

Ejercicio 5

Escribir un programa que calcule el valor del número e, base de los logaritmos neperianos, a partir de la expresión:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{20!}.$$

Funciones que puede realizar la variable del bucle

Tal como se ha visto en la práctica anterior, las estructuras repetitivas en MATLAB se obtienen con la siguiente sintaxis:

```
for i = [i_1, ..., i_n]
        conjunto de instrucciones
end
```

Se puede considerar i como una variable interna del bucle que en cada repetición toma distintos valores. Dentro de un bucle la variable puede intervenir de varias formas:

- Como contador.
- Como variable con la que se realizan operaciones en las instrucciones del bucle.
- Como índice de un vector o de una matriz.

Los aspectos 1 y 2 ya se han sido analizados. A continuación se estudia el caso en el que la variable i es el índice de un vector.

Vector definido por sus coordenadas

Recordemos que cuando se define un vector, por ejemplo `x=[3 1 4]`, se están definiendo implícitamente 3 variables *indexadas* (que dependen de un índice), que en este caso serían $x(1) = 3,\ x(2) = 1,\ x(3) = 4$. Recíprocamente, se puede definir un vector a partir de sus coordenadas. Este procedimiento puede utilizarse para construir un vector mediante un bucle, mediante un esquema del tipo

```
for i=1:n
   x(i)
end
```

Ejercicio 6

En la línea de comandos:

(a) Definir el vector `x=[3 1 4]` y comprobar que quedan definidas `x(1), x(2)` y `x(3)`.

(b) Escribir `y(1)=6; y(2)=3; y(3)=9;` y comprobar que queda definido el vector `y=[6 3 9]`.

Ejercicio 7

(a) Escribir un programa que genere, mediante un bucle, el vector `x=[1 1 1 1 1]`.

(b) Escribir un programa que pida un número n y que genere un vector de n coordenadas, las de índice impar iguales a uno y las de índice par iguales a dos.

Ejercicio 8

Se denomina sucesión de Fibonacci a la definida por la la ley de recurrencia

$$\begin{cases} x_1 = 1,\ x_2 = 1 \\ x_n = x_{n-1} + x_{n-2}, \qquad n \geq 3 \end{cases}$$

Escribir un programa que pida un número natural n y genere los n primeros términos de la sucesión de Fibonacci.

Ejercicio 9

(a) Escribir un programa que, mediante un bucle, sume las coordenadas del vector `x=[3 2 5 4]`.

(b) Modificar el programa para que pida un vector y sume sus coordenadas.

Ejercicio 10

La media y la varianza de un conjunto de n datos, $x_1,\ x_2, \ldots, x_n$, están dadas por

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad s^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \bar{x}^2.$$

Escribir un programa que, al suministrarle n datos, calcule su media y su varianza.

2.2. Bucles anidados

Se dice que dos bucles están anidados cuando uno de ellos (el bucle interno) forma parte del conjunto de instrucciones del otro (el bucle externo). De acuerdo con esto su sintaxis es la siguiente:

```
for i = [i_1 ··· i_m]
    for j = [j_1 ··· j_n]
        conjunto de instrucciones
    end
end
```

y equivale a

```
i = i_1
    j = j_1
        conjunto de instrucciones
    j = j_2
        conjunto de instrucciones
      ⋮
    j = j_n
        conjunto de instrucciones

i = i_2
    j = j_1
        conjunto de instrucciones
    j = j_2
        conjunto de instrucciones
      ⋮
    j = j_n
        conjunto de instrucciones
  ⋮
```

Al final, el conjunto de instrucciones se ejecuta $m \times n$ veces.

Bucles anidados y matrices

Cuando se define una matriz, por ejemplo

$$a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix},$$

se están definiendo implícitamente 6 variables doblemente indexadas (que dependen de dos índices). Para este caso

```
a(1,1)=1; a(1,2)=2; a(1,3)=3; a(2,1)=4; a(2,2)=5; a(2,3)=6.
```

Recíprocamente, es posible definir una matriz a partir de sus elementos. Esto permite definir una matriz mediante dos bucles anidados cuyas variables sean los índices de los elementos de la matriz, de acuerdo con uno de los esquemas siguientes:

```
for i = 1 : m
    for j = 1 : n
        a(i,j)
    end
end
```

```
for i = 1 : m
    for j = 1 : n
        a(j,i)
    end
end
```

El primer esquema genera una matriz $m \times n$ por filas. El segundo esquema genera una matriz $n \times m$ por columnas.

Ejercicio 11

(a) Escribir un programa que genere una matriz de ceros de dimensión 2×3, utilizando dos bucles andidados.

(b) Modificar el programa para que pida dos números naturales m y n y que genere una matriz de ceros de dimensión $m \times n$.

Ejercicio 12

Se considera la matriz $a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$.

(a) Escribir un programa que, partiendo de una matriz a, genere la matriz b cuyos elementos son los de a elevados al cuadrado.

(b) Modificar el programa para que genere la matriz c igual a la traspuesta de a.

Ejercicio 13

(a) Escribir un programa que, partiendo de la matriz a del ejercicio anterior, genere las matrices b y c simultáneamente, utilizando dos bucles anidados.

(b) Modificar el programa para que pida la matriz a y calcule a partir de ella las matrices b y c.

Ejercicio 14

Indicar las matrices a y b que se generan al ejecutar el siguiente programa:

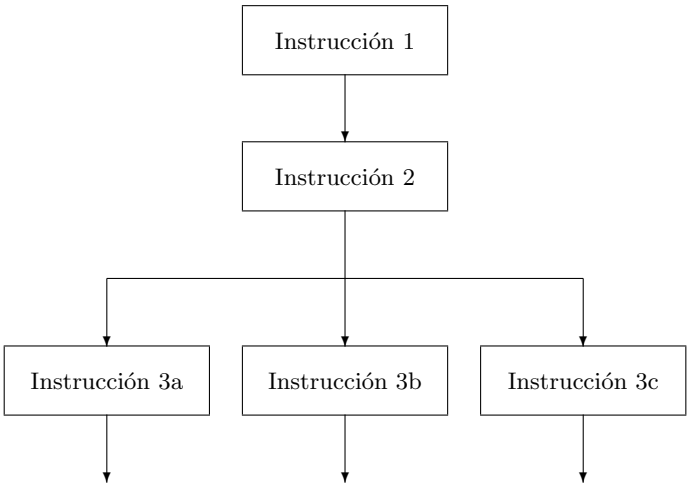
```
for i=1:2
   for j=1:3
      a(i,j)=i+j;
      b(j,i)=i-j;
   end
end
a, b
```

Ejercicio 15

Escribir un programa que genere la matriz $a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, utilizando dos bucles anidados.

3. Estructuras condicionales

Las estructuras condicionales permiten que en un punto del programa se presenten varias alternativas, de las cuales solo será tenida en cuenta una en cada ejecución. El diagrama de flujo adopta la forma:



Dependiendo de alguna condición especificada en el algoritmo, después de la instrucción 2 se ejecuta la instrucción 3a, la 3b o la 3c (pero solo una de ellas).

3.1. Expresiones lógicas

Una expresión lógica se caracteriza por su valor de verdad: 1, en el caso de que sea verdadera y 0, en el caso de que sea falsa.

Las expresiones lógicas se construyen a partir de expresiones numéricas utilizando operadores relacionales, y a partir de otras expresiones lógicas mediante operadores lógicos.

Operadores relacionales

En la tabla siguiente se describen los diferentes operadores relacionales junto a su simbolización en MATLAB.

SÍMBOLO	SIGNIFICADO
$<$	menor
$<=$	menor o igual
$>$	mayor
$>=$	mayor o igual
$==$	igual
$\sim=$	distinto[2]

Con estos operadores, a partir de dos expresiones numéricas x e y, se construyen las expresiones lógicas

$$x < y, \quad x <= y, \quad x > y, \quad x >= y, \quad x == y, \quad x \sim= y,$$

que pueden ser verdaderas o falsas dependiendo de los valores de x e y. Por ejemplo, si escribimos en la linea de comandos:

$$>> 2 >= 3,\ 2 \sim= 3,\ 2 == 2$$

obtenemos como resultado 0, 1, 1 ya que la primera expresión es falsa y las dos últimas verdaderas.

Operadores lógicos

Dos expresiones lógicas se pueden combinar mediante operadores lógicos para dar lugar a una nueva expresión lógica. En la siguiente tabla se indican los más utilizados junto con su representación en MATLAB.

[2] El símbolo $\sim$ se obtiene con las combinaciones de teclas Alt Gr 4 (4 en el teclado alfabético), o Alt 126 (126 en el teclado numérico).

SÍMBOLO	SIGNIFICADO
&	y
|	ó
∼	no

Con estos operadores, a partir de las expresiones lógicas **p** y **q**, se definen las expresiones lógicas:

$$\mathbf{p} \ \& \ \mathbf{q}, \quad \mathbf{p} \mid \mathbf{q}, \quad \sim \mathbf{p},$$

que se leen, respectivamente,

$$\mathbf{p} \text{ y } \mathbf{q}, \quad \mathbf{p} \text{ ó } \mathbf{q}, \quad \text{no } \mathbf{p}.$$

Por tratarse de expresiones lógicas, lo único que es necesario saber acerca de ellas es su verdad o falsedad. Esta solo depende de los valores de verdad de las expresiones **p** y **q** a partir de las que se definen, de acuerdo con la siguientes tablas:

p	**q**	**p** & **q**	**p** | **q**
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

p	**∼p**
1	0
0	1

Teniendo esto en cuenta se puede comprobar, por ejemplo, que son verdaderas las siguientes expresiones lógicas:

$$(3 >= 3) \ \& \ \sim (2 == 5), \qquad (1 > 6 \mid 1 <= 6) \ \& \ \sim (3 < -2).$$

Los operadores relacionales solo permiten comparar dos expresiones numéricas. Por ejemplo, la expresión $2 < x < 5$ tiene sentido en Matemáticas, pero en MATLAB habría que escribirla en la forma $2 < x \quad \& \quad x < 5$.

Prioridad de los operadores

En este momento se conocen los operadores aritméticos (+, −, ∗, /, ^), los que realizan operaciones matriciales elemento a elemento (.∗, ./, .^) y los operadores relacionales y lógicos que se acaban de definir. En una misma expresión pueden aparecer varios tipos de operadores. El orden de actuación es que se establece en la siguiente tabla, teniendo en cuenta que: (1) La prioridad disminuye de arriba abajo. (2) Los operadores situados en una misma linea tienen la misma prioridad. (3) Si se presentan varios operadores con igual prioridad en una misma expresión, se ejecutan de izquierda a derecha. (4) El orden de prioridad se puede modificar utilizando paréntesis.

ORDEN DE PRECEDENCIA DE OPERADORES					
^	.^				
*	/	.*	./		
+	−	∼	+ (unario)	− (unario)	
<	<=	>	>=	==	∼=
&	|				

3.2. Sintaxis de las estructuras condicionales en MATLAB

En MATLAB las estructuras condicionales se construyen con la sintaxis `if-elseif-else-end`. Dependiendo del número de alternativas, esta sintaxis adopta diferentes formas.

Una alternativa

```
if expresión lógica
    conjunto de instrucciones
end
```

El conjunto de instrucciones se ejecuta solo en el caso de que la expresión lógica sea verdadera.

Dos alternativas

```
if expresión lógica
    conjunto de instrucciones 1
else
    conjunto de instrucciones 2
end
```

Cuando la expresión lógica es verdadera, se ejecuta el conjunto de instrucciones 1; cuando es falsa se ejecuta el conjunto de instrucciones 2.

Varias alternativas

```
if expresión lógica 1
    conjunto de instrucciones 1
elseif expresión lógica 2
    conjunto de instrucciones 2
elseif expresión lógica 3
    conjunto de instrucciones 3
else
    conjunto de instrucciones
end
```

Se ejecuta la instrucción correspondiente a la primera expresión lógica verdadera.

Si son todas falsas se ejecuta la instrucción correspondiente a la alternativa que no está asociada a ninguna expresión lógica (la que viene después de `else`). Si todas las expresiones lógicas son falsas y esta alternativa no esta presente (pues no es obligatoria), no se ejecuta ninguna instrucción.

Conviene insistir en que en una estructura de este tipo solo se ejecuta, a lo sumo, un conjunto de instrucciones.

En MATLAB las estructuras condicionales también se pueden construir con la sintaxis `switch-case` que no se estudia aquí.

Ejercicio 16

Escribir un programa que pida un número y diga si es par o impar y si es mayor, menor o igual que cincuenta.

Ejercicio 17

Escribir un programa que pida la edad x de una persona, expresada en años, y la clasifique como niño ($x \leq 12$), adolescente ($12 < x \leq 17$), joven ($17 < x \leq 25$) o adulto ($x > 25$).

Ejercicio 18

Escribir un programa que pide los tres coeficientes $a,\ b,\ c$ de una ecuación de segundo grado $ax^2 + bx + c = 0$, y muestra si las raíces son complejas o reales y, en este último caso, si son iguales o distintas. En todos los casos devuelve los valores numéricos de las raíces.

Ejercicio 19

Escribir un programa que pida los tres lados de un triángulo, estudie si es posible construirlo y, en caso afirmativo, diga si es equilátero, isósceles o escaleno.

Ejercicios con bucles y estructuras condicionales

Ejercicio 20

Escribir un programa que pide un número n y genera una matriz unidad de dimensión $n \times n$.

Ejercicio 21

Escribir un programa que pide dos matrices, analiza si es posible sumarlas y, en caso afirmativo, calcula su suma.

Ejercicio 22

Escribir un programa que determina los números de tres cifras que son iguales a la suma de los cubos de sus cifras.