Tema 4

Introducción a Matlab

Fundamentos de Informática

Grado en Ing. Química

ugr | Universidad de Granada

Jesús Alcalá y David Pelta

DECSAI

Universidad de Granada

Índice

Objetivos

- Dominar los aspectos básicos de MatLab
- Comprender los conceptos de programación
- Ser capaz de diseñar y programar algoritmos para resolver problemas generales, utilizando herramientas tipo MatLab, como paso indispensable para abordar la resolución de problemas específicos de Ing. Química.

Bibliografía

- J. Garcia Molina, F. Montoya Dato, et al., Una introducción a la Programación. Un enfoque algorítmico, Thompson, 2005
- Pérez López, César, MATLAB y sus aplicaciones en las Ciencias y la Ingeniería. Madrid : Pearson Educación, 2002
- Gilat, Amos Matlab : una introducción con ejemplos prácticos. Barcelona : Reverté, 2006
- García de Jalón J., Rodríguez J. , Vidal J.. Aprenda Matlab 7.0 como si estuviera en primero
- Stephen J. MATLAB programming for engineers. Thomson, 2008.
- ***Numerosos libros en la Biblioteca de Ciencias y el Politécnico***

¿Qué es MATLAB?

- **MATLAB = MATrix LABoratory** (laboratorio de matrices)
  - http://www.mathworks.es/products/matlab/
- Es un software muy versátil que ofrece:
  - Un **entorno de desarrollo integrado** (*Integrated Development Environment*, IDE): interprete, editor, depurador (*debugger*), asistente de ayuda, librerías, etc.
  - Un **lenguaje de programación** propio: lenguaje M
- Está disponible para los sistemas operativos Microsoft Windows, Unix y Apple Mac OS
- Entre sus prestaciones básicas se hallan las siguientes:
  - Manipulación eficiente de matrices
  - Representación de datos y funciones
  - Implementación de algoritmos
  - Creación de interfaces gráficas de usuario (*Graphical User Interface*, GUI)
  - Comunicación con dispositivos hardware

¿Qué es MATLAB?

- Las prestaciones de MATLAB se pueden ampliar incorporando al IDE las "cajas de herramientas" (*toolboxes*), aplicaciones software que implementan funcionalidades muy diversas:
  - **Cálculo técnico**
    - Cálculo numérico, análisis, visualización y desarrollo de algoritmos
  - **Diseño de control**
    - Simulación, modelización rápida de prototipos
  - **Procesamiento de señales digitales**
    - Análisis de señales, diseño de sistemas DSP
  - **Sistemas de comunicaciones**
    - Diseño y simulación de sistemas complejos de comunicaciones

¿Qué es MATLAB?

- Las prestaciones de MATLAB se pueden ampliar incorporando al IDE las "cajas de herramientas" (*toolboxes*), aplicaciones software que implementan funcionalidades muy diversas:
  - **Procesamiento de imágenes**
    - Algoritmos de adquisición, análisis y mejora de imágenes
  - **Pruebas y mediciones**
    - Análisis de datos para aplicaciones de pruebas y mediciones
  - **Bioinformática**
    - Análisis, visualización y simulación de sistemas biológicos
  - **Finanzas computacionales**
    - Análisis, simulación y desarrollo de aplicaciones financieras

Atención

- MATLAB está instalado en las Aulas de Docencia
- Existen versiones para estudiantes (hay que pagar)
- Existen versiones de prueba

Alternativa: OCTAVE (libre y gratuito)

- www.octave.org

"GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab"

- Modo Consola
- Existen algunos IDE's en desarrollo

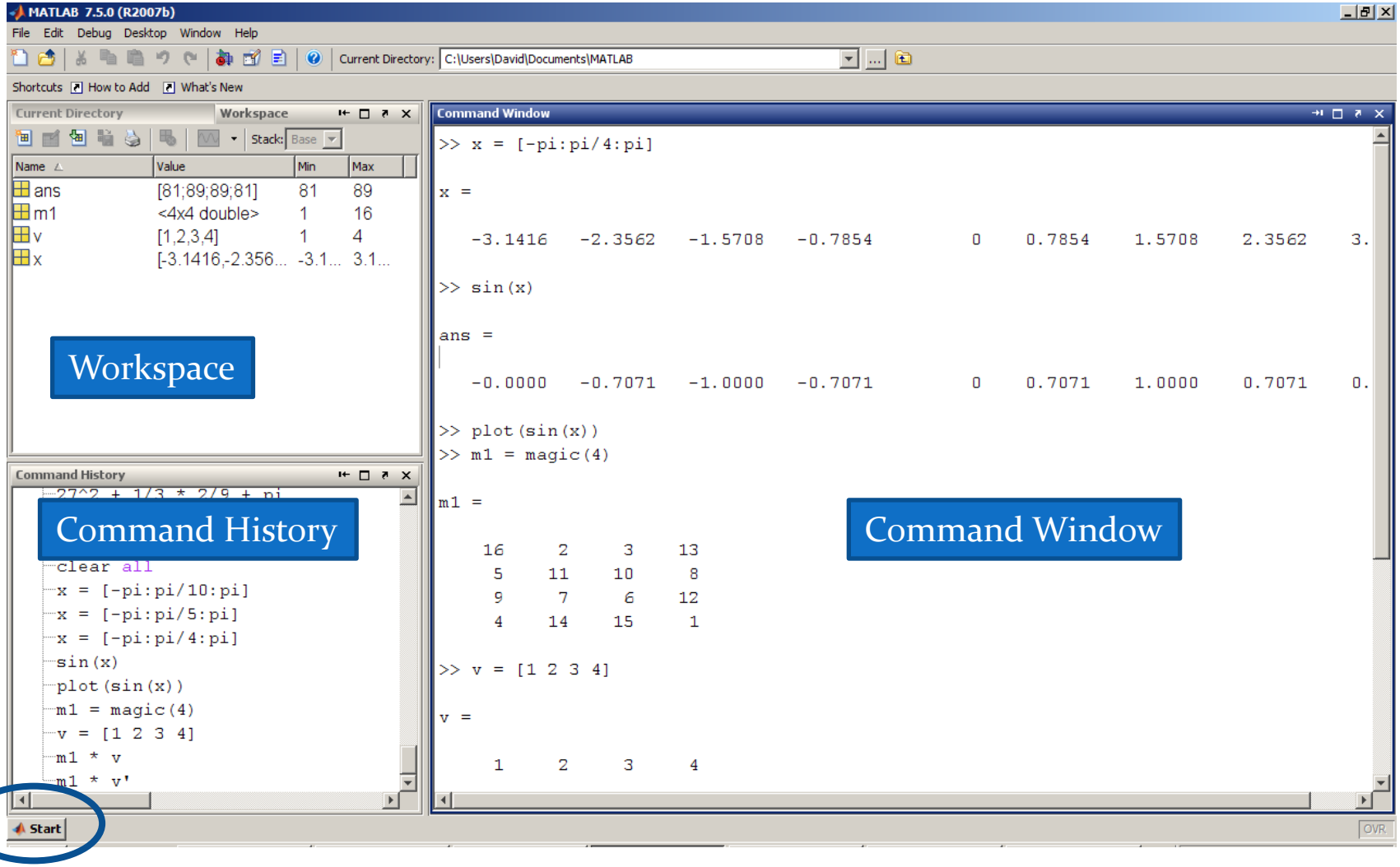
¿Qué es MATLAB?

- El IDE de MATLAB está formado por los siguientes componentes principales:
  1. **El interprete**
     - Permite al usuario la introducción de **instrucciones** (comandos)
     - Ejecuta (interpreta) las instrucciones introducidas y muestra los resultados de las mismas
     - Las instrucciones pueden ser muy diversas: declaración y asignación de variables, operaciones aritméticas y lógicas, llamadas a función, etc.
  2. **El editor**
     - Permite al usuario escribir y modificar **funciones**: bloques de instrucciones que reciben unas variables de entrada, las procesan, y devuelven otras de salida
     - La estructura de un programa, en general, consta de una función principal, que invoca a otras

¿Qué es MATLAB?

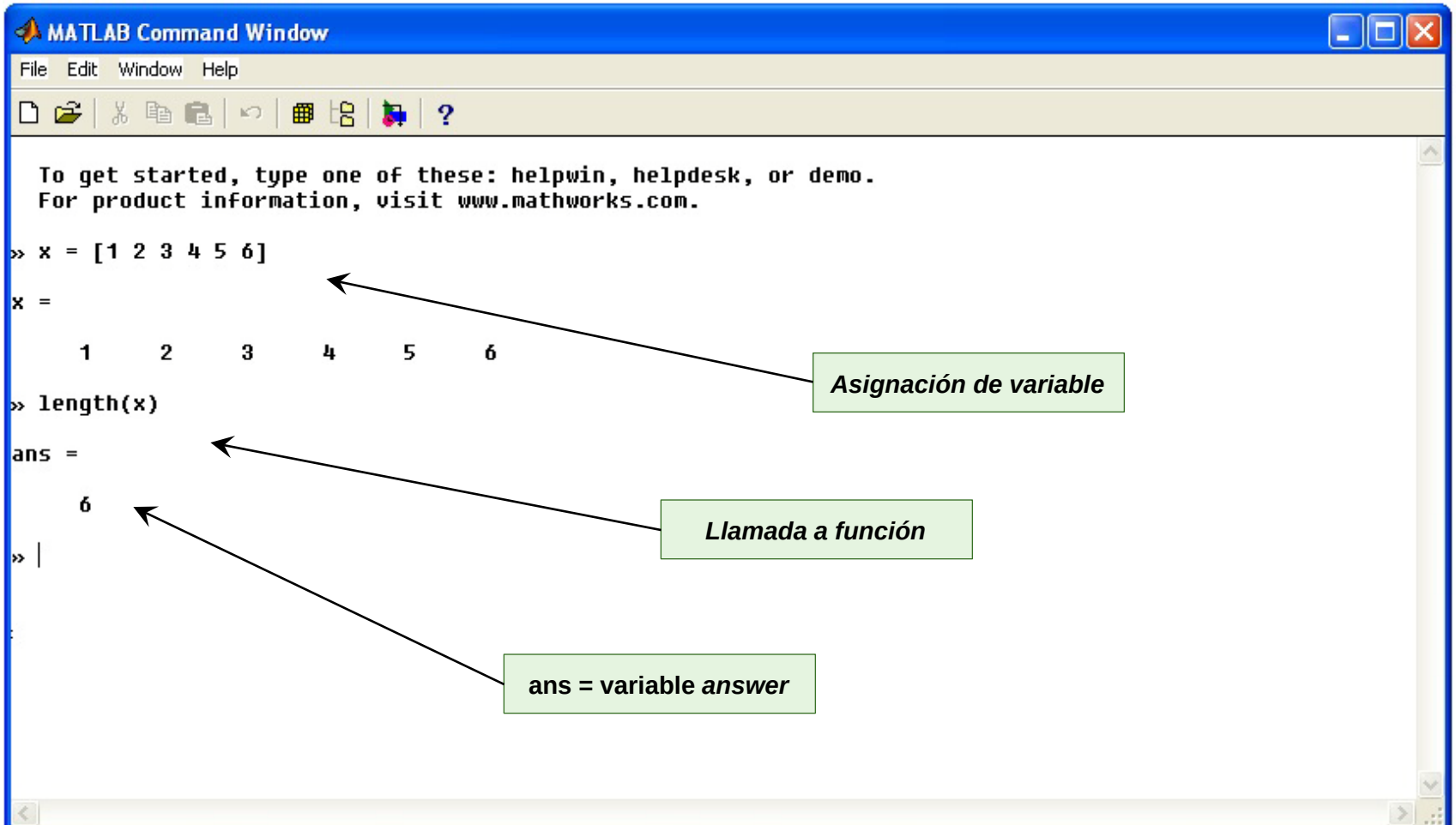
- El IDE de MATLAB está formado por los siguientes componentes principales:
  3. **El depurador**
     - Permite al usuario ejecutar instrucción a instrucción un programa, pudiendo acceder al estado de las variables empleadas por él
     - Se utiliza para detectar y corregir **errores** en la programación
  4. **El asistente de ayuda**
     - Proporciona al usuario **información** sobre la herramienta MATLAB (configuración, uso, etc.), el lenguaje M (sintaxis, ejemplos de programas, etc.), y las funciones implementadas disponibles (argumentos de entrada, propósito, retornos de salida, etc.)

El Interprete



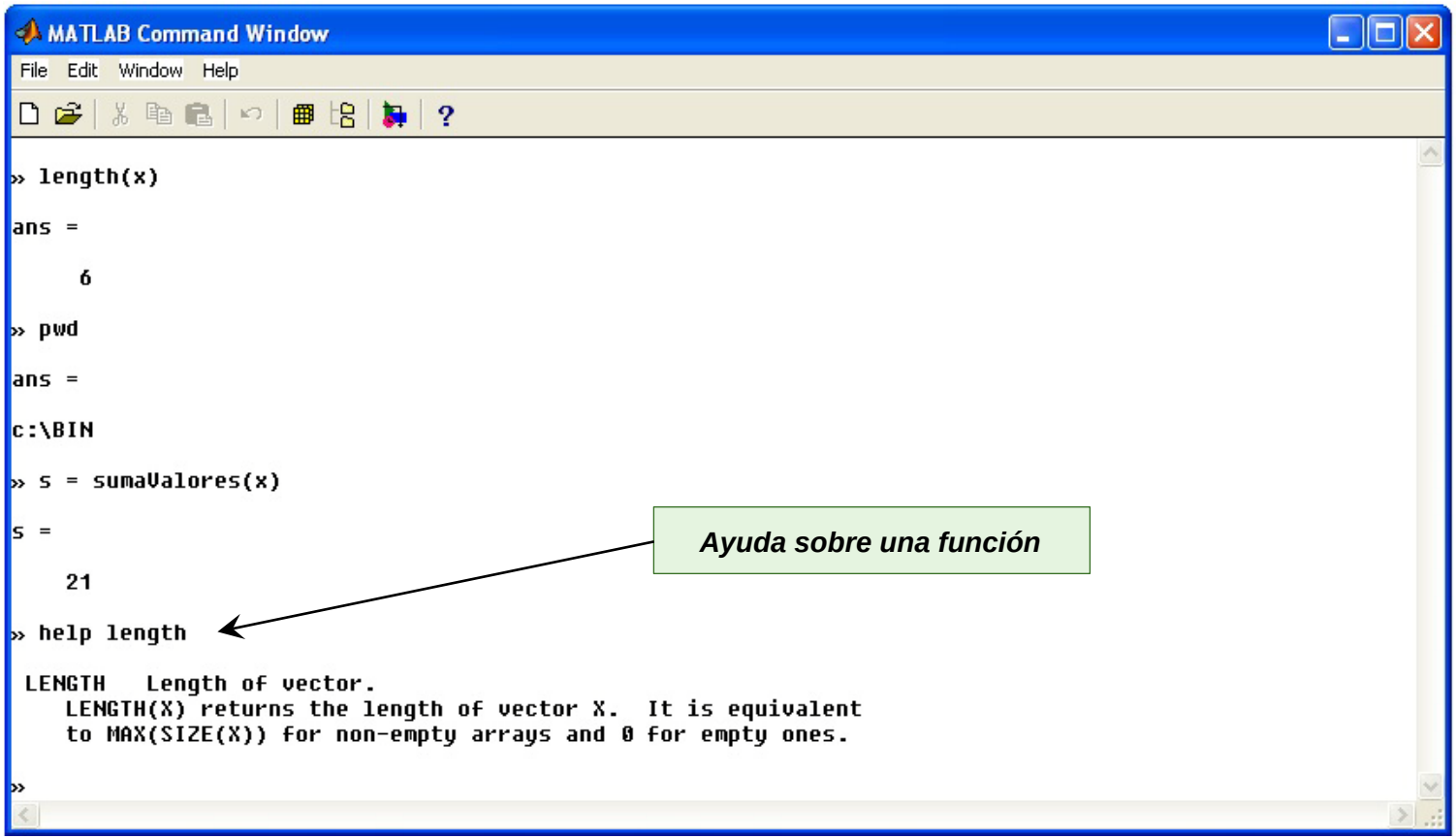
Command Window

- En el interprete se ejecutan “comandos”



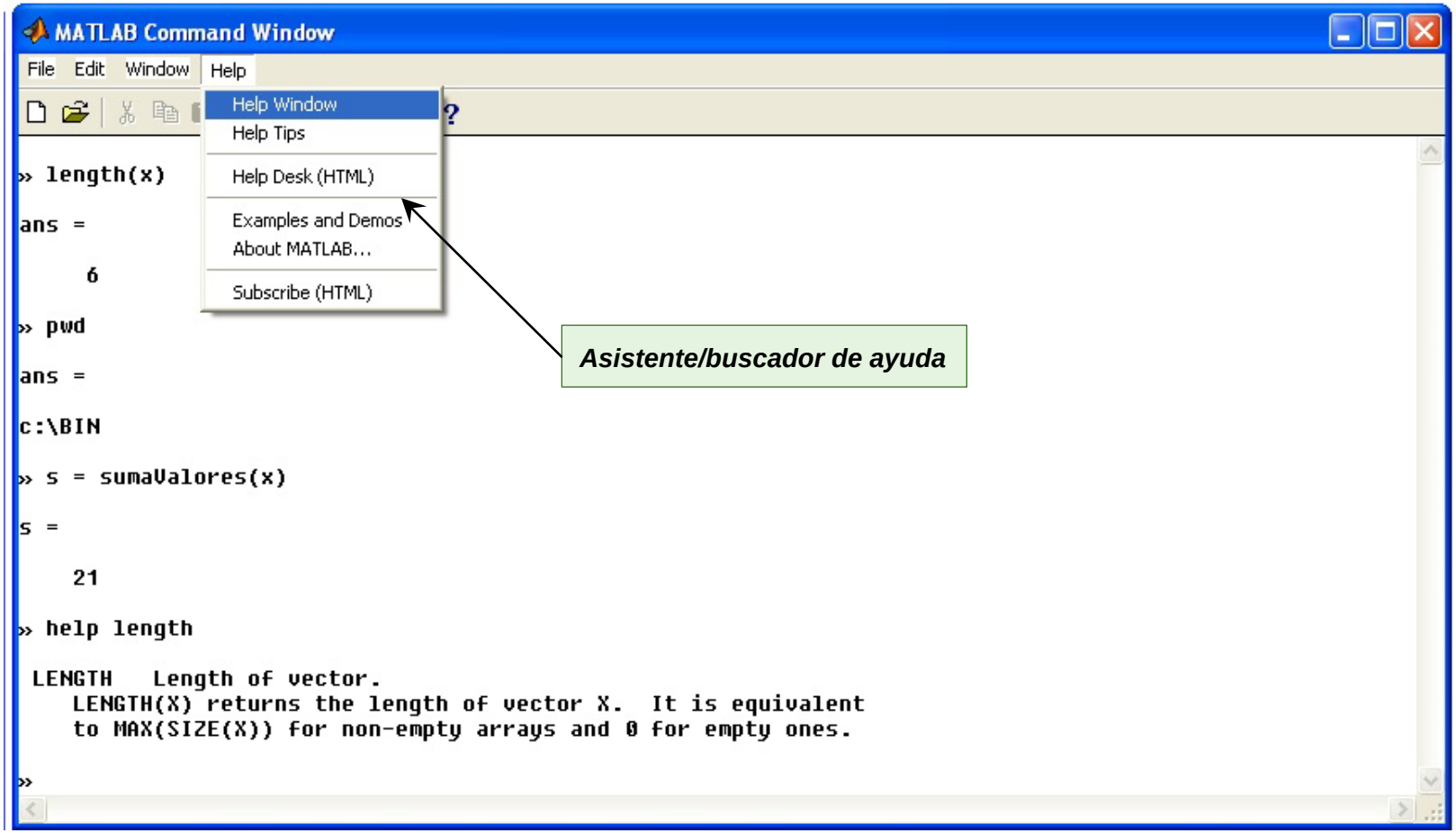
El asistente de ayuda

- El comando *help* ofrece información de ayuda en el interprete



El asistente de ayuda

- Imposible recordar todas las posibilidades de Matlab



Algunos elementos del entorno

1. Menus, botones, “layout”
2. MATLAB como calculadora
3. flechas
4. Botón “start”
5. dock / undock
6. Preferencias

Números enteros y reales

Operaciones aritméticas

- ^ exponenciación

  2 ^ 3 → 8

- * producto

  2 * 3 → 6

- / división

  2 / 3 → 0.6667

- + suma

  2 + 3 → 5

- – resta

  2 – 3 → -1

ans, "lugar" donde se guarda el resultado de la última operación

Números enteros y reales

- Orden de precedencia en expresiones
  - 1º la exponenciación, 2º los productos y divisiones, y 3º las sumas y restas
  - Si se quiere forzar un determinado orden se deben utilizar paréntesis, que siempre se evalúan primero
  - `4/4+6 = 7` `4/(4+6)= 0.4`
  - `3^5 * 2 = 486` `3^(5 * 2) = 59049`

Números enteros y reales

- **Funciones elementales**
  - **sqrt(x)** calcula la raíz cuadrada de x
    sqrt(16) 4
  - **abs(x)** devuelve el valor absoluto de x
    abs(-16) 16
  - **mod(x, y)** devuelve el resto de dividir x entre y
    mod(16, 7) 2
  - **round(x)** devuelve el número entero más cercano a x ("redondeo" de x)
    round(16.7) 17
  - **floor(x)** devuelve el número entero más cercano e inferior a x ("suelo" de x)
    floor(16.7) 16
  - **ceil(x)** devuelve el número entero más cercano y superior a x ("techo" de x)
    ceil(16.7) 17

Números enteros y reales

- **Funciones trigonométricas**
  - `sin(x), cos(x), tan(x)`
  - `csc(x), sec(x), cot(x)`
  - `sinh(x), cosh(x), tanh(x)`
  - `asin(x), acos(x), atan(x)`
- **Funciones exponencial y logarítmica**
  - **`exp(x)`** calcula la exponencial ex
  - **`log(x)`** calcula el logaritmo neperiano de x
  - **`log10(x)`** calcula el logaritmo en base 10 de x

¿ Como se usa esto ?

Números enteros y reales

"**Números especiales**"

- pi % número Pi
- exp(1) % número de Euler
- i % número imaginario i
- j % número imaginario i
- intmin % número entero más pequeño con que se puede trabajar
- intmax % número entero más grande con que se puede trabajar
- realmin % nro más pequeño con que se puede trabajar: 2.2251e-308
- realmax % nro. más grande con que se puede trabajar: 1.7977e+308
- inf % infinito
- -inf % – infinito
- NaN % not-a-number (puede darse por ejemplo al dividir por 0)

Definición de variable

- Una variable es una estructura de datos que permite almacenar un valor o conjunto de valores.
- Es equivalente a una "variable" en el contexto matemático.
- Una variable corresponde con un **área reservada de la memoria principal** del ordenador
- El tamaño del área reservada dependerá del tipo del dato que se vaya a almacenar

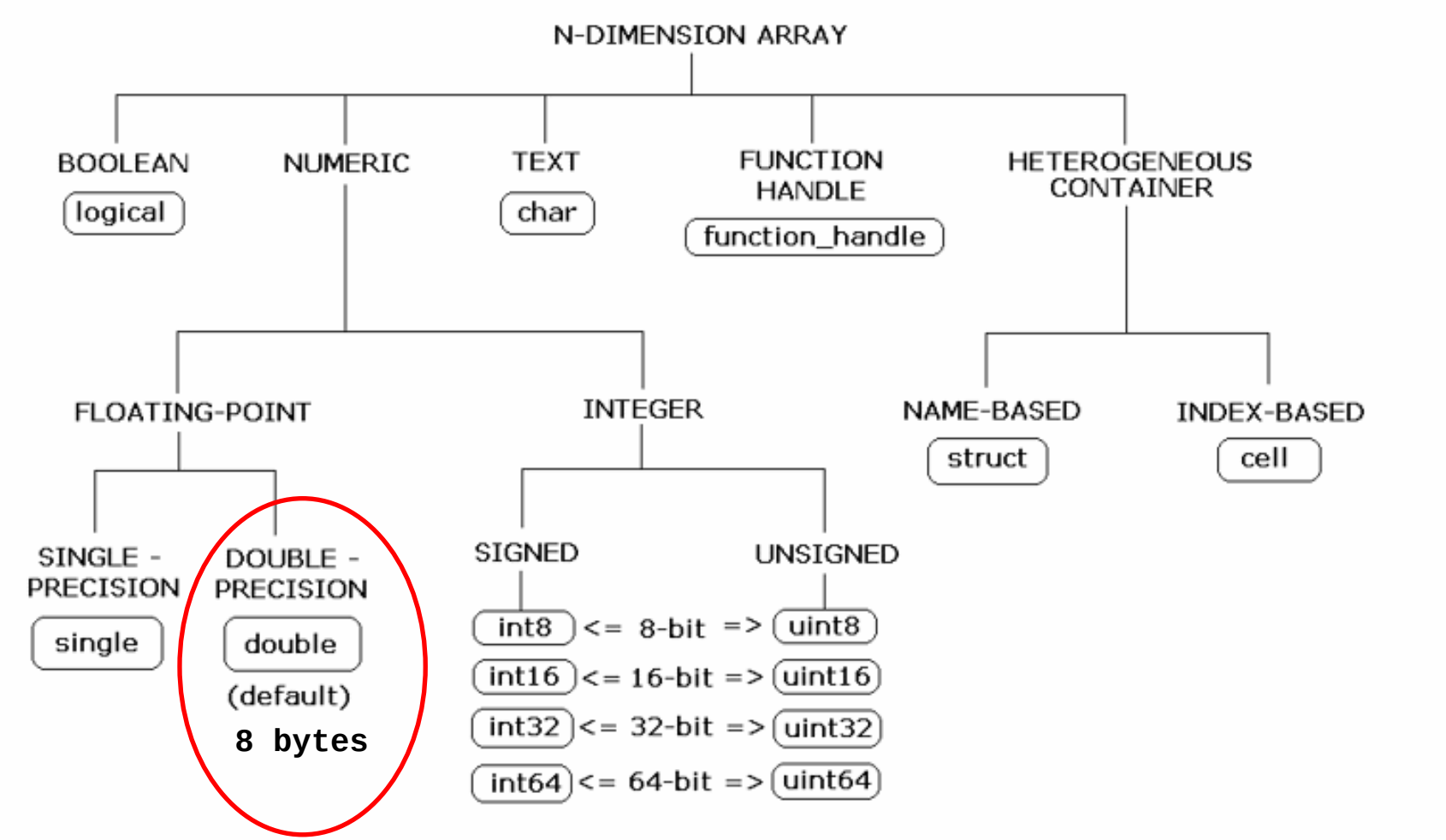
Definición de variable

- Una variable tiene asociado un **nombre o identificador**
  - El nombre es único dentro del "ámbito" o "alcance" de la variable
  - combinación de letras, numeros y "_".
  - No empieza con número
  - Limite de 63 simbolos.
  - Sensible a mayúsculas / minúsculas
  - ***Existen "palabras reservadas"***
- El **ámbito** de una variable puede ser:
  - *Local*: cuando la variable sólo puede ser accedida por un subconjunto de instrucciones del programa:
    - Un bloque condicional
    - Un bucle
    - Una función…
  - *Global*: cuando la variable puede ser accedida por cualquier instrucción del programa

Definición de variable

- Una variable es de un **tipo de dato** concreto
  - Un número
  - Un carácter
  - Una cadena de caracteres
  - Un dato Booleano
  - Una estructura
  - …
- El tipo de dato de una variable puede ser complejo, i.e., puede almacenar varios valores de diferentes tipos de datos
  - Ejemplo: una lista de objetos, donde cada objeto es un dato "primitivo" (número, carácter, dato Booleano)
- Por defecto, Matlab trabaja con números en doble precisión

Es posible declarar el tipo de las variables para que utilicen menos espacio de almacentamiento



Variables: dar valores

Antes de usar una variable, es necesario darle algún valor. Es decir ***"inicializarla".***

La forma más directa es mediante una sentencia de asignación.

```
>> nro = 45
```

LADO IZQUIERDO:
una variable

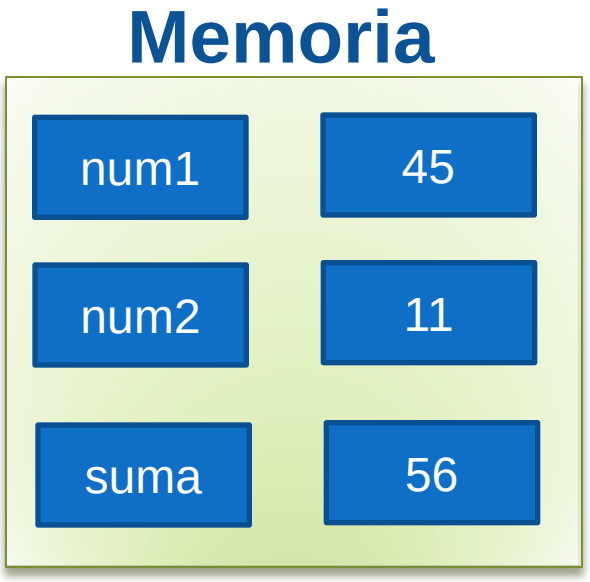
Operador de Asignación

LADO DERECHO:
una variable, un literal o una expresión compleja.

Variables: dar valores

Cuando se ejecuta una operación de asignación, primero se evalúa la expresión del lado derecho y luego se almacena el valor resultante en la variable indicada en el lado izquierdo.

```
>> num1 = 45;
>> num2 = 11;
>> suma = num1 + num2;
>> suma
suma =
     56
>>
```



Regla de Asignación

Una variable en el lado derecho de una sentencia de asignación debe tener un valor antes de que la sentencia de asignación se ejecute. Hasta que un programa le da un valor a una variable, esa variable no tiene valor.

Ejemplo:

```
y = x + 1
```

ERROR LÓGICO: *la variable* ***x*** *no tiene ningún valor. El valor que toma la variable* ***y*** *es impredecible!!!*

En la izquierda de una de asignación solo pueden existir variables. La siguiente expresión no es valida:

```
Valor_Neto - Tasas =  34015;
```

Regla de Asignación

La operación de asignación es una operación destructiva: el valor almacenado en una variable se pierde o se destruye y se sustituye por el nuevo valor en la sentencia de asignación.

Variables

- Operaciones aritméticas con variables

```
>> x = 2^3
x =  8
>> a = 2 * 3
a =  6
>> y = a /x
y =  0.7500
```

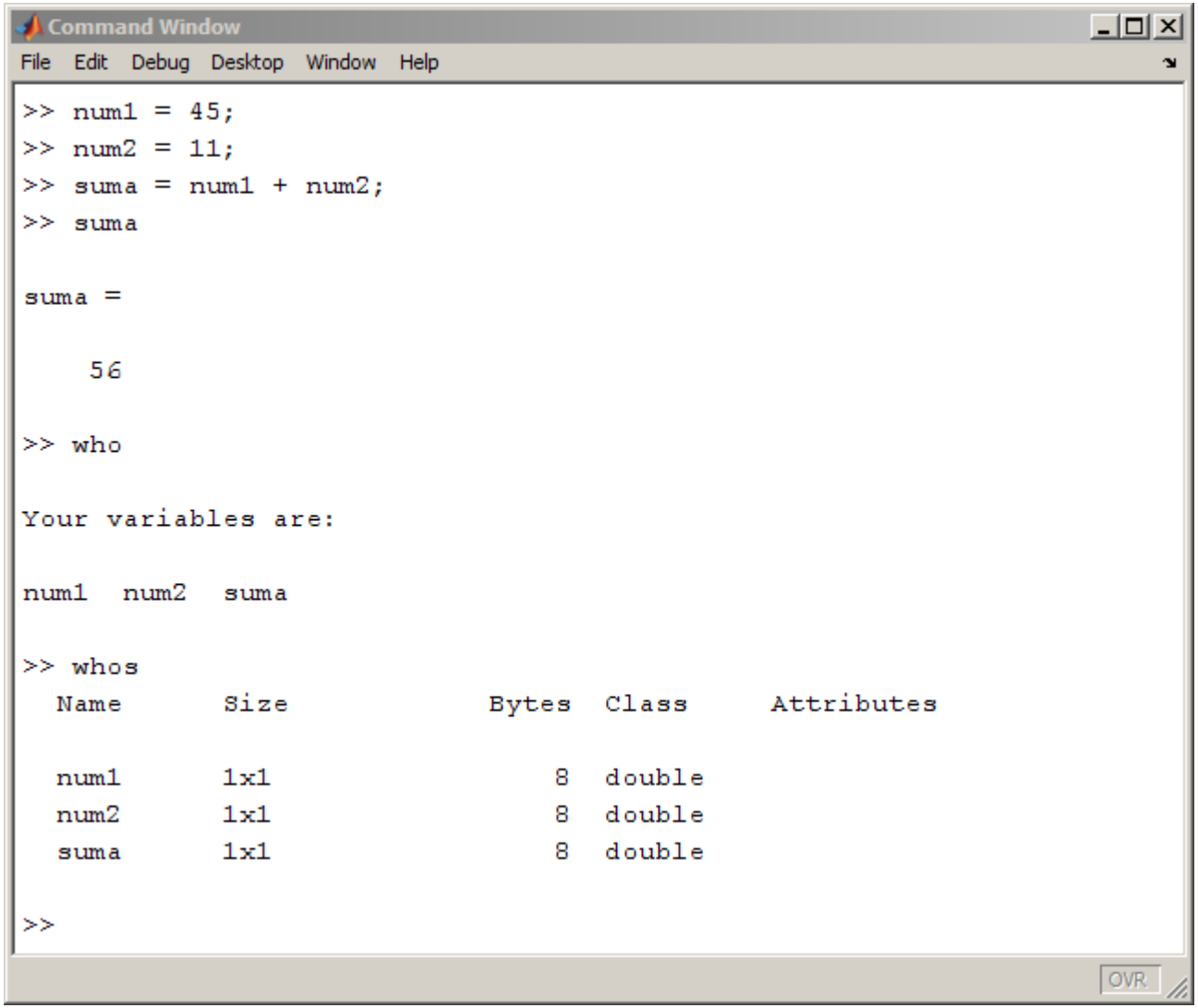
```
>> x = 10;
>> a = 3.25;
>> b = 3;
>> y = a*x + b
y = 35.5000
```

```
>> x = 2.18;
>> y = sin(x) * cos(pi * x)
y =  0.6924
```

Comandos sobre Variables

- ***who:*** Cuando se quiere tener una relación de las variables que se han utilizado en una sesión de trabajo
- ***whos:*** similar a who pero proporciona además información sobre el tamaño, la cantidad de memoria ocupada y el carácter real o complejo de cada
- ***Comando clear***
  - ***clear*** : elimina todas las variables creadas previamente (excepto las variables globales).
  - ***clear A, b***: borra las variables indicadas A y b.
  - ***clear global:*** borra las variables globales.
  - ***clear functions:*** borra las funciones.
  - ***clear all:*** borra todas las variables, incluyendo las globales, y las funciones.

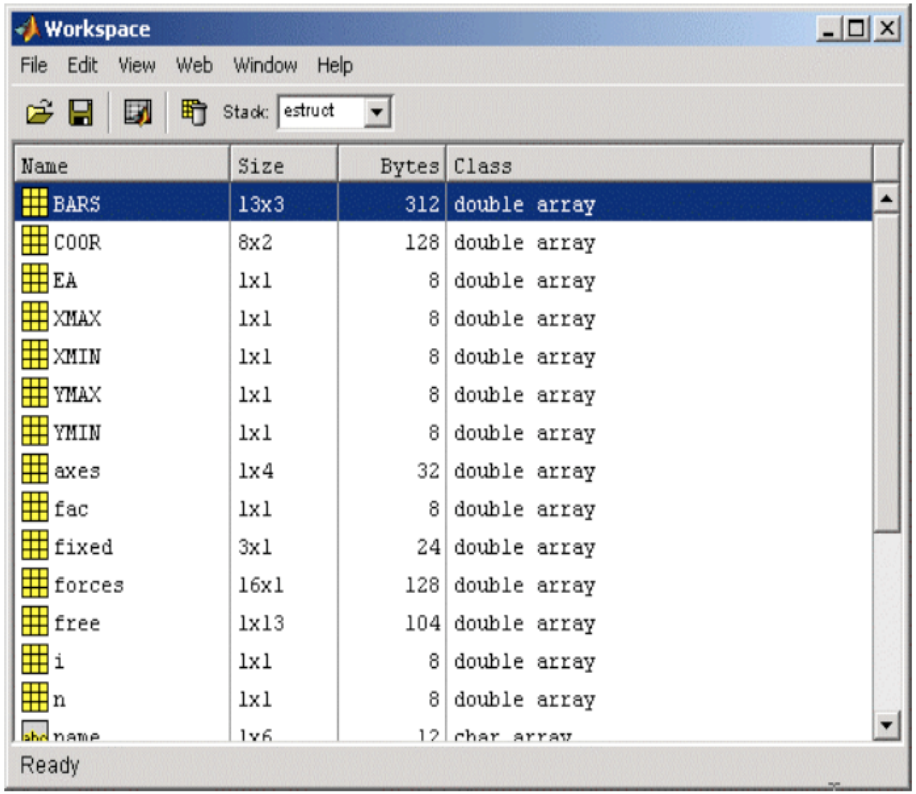
Ejemplo

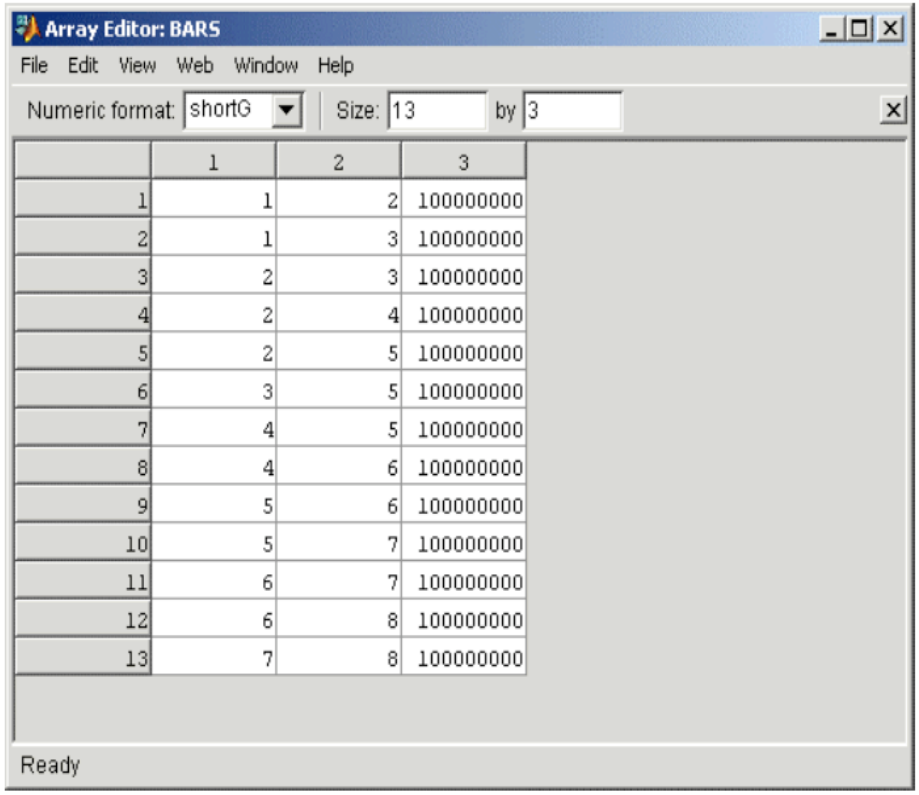

Command Window
File Edit Debug Desktop Window Help
>> num1 = 45;
>> num2 = 11;
>> suma = num1 + num2;
>> suma
suma =
56
>> who
Your variables are:
num1 num2 suma
>> whos
Name Size Bytes Class Attributes
num1 1x1 8 double
num2 1x1 8 double
suma 1x1 8 double
>>
OVR

El espacio de trabajo

- La ventana *Workspace Browser* constituye un entorno gráfico para ver las variables definidas en el espacio de trabajo
  - Se activa a través del menú *View/Workspace*
  - Haciendo doble clic en una variable aparece una nueva ventana donde poder modificar el contenido de dicha variable
  - *Workspace Browser en acción*

El espacio de trabajo





El espacio de trabajo

- Es posible guardar las variables del *workspace* en un fichero para poder recuperarlas más tarde: comandos **save** y **load**

```
>> save sesion.mat    % guarda el workspace actual en sesion.mat
>> load sesion.mat    % carga en memoria el wokspace guardado en
                      % sesion.mat
```

- El comando **diary** también permite guardar y recuperar todas las variables utilizadas durante una sesión

```
>> diary sesion.txt   % el workspace se guarda a partir de este punto
                      % en sesion.txt
>> ...
>> diary off          % deja de guardarse el workspace
>> ...
>> diary on           % se reanuda el proceso de guardar el workspace
```

¿ Donde se guardan ? En el "Current Directory"

Números complejos

- En MATLAB, un número complejo está compuesto de:
  - **Parte real**: que es un número real
  - **Parte imaginaria**: que es un número real multiplicado por **i** o por **j**
- Creación

```
>> c = 2 + 3i
c = 2.000 + 3.000i
>> c = 2 + 3j
c = 2.000 + 3.000i
>> c = complex(2, 3)
c = 2.000 + 3.000i
```

- Cuidado si se tiene declarada una variable con nombre *i* o *j*

```
>> i = 3;
>> c = 1 + i
c = 4          % c no es 1 + i
```

Números complejos

- Operaciones aritméticas que crean números complejos

```
>> (-1)^0.5
ans = 0.000 + 1.000i
>> (-3)^0.25
ans = 0.9306 + 0.9306i

>> log(-1)
ans = 0.000 + 3.1416i

>> roots(p) % las raíces de un polinomio pueden
ser complejas
```

Números complejos

Funciones de manipulación

- **`real(c)`** parte real del número complejo c
- **`imag(c)`** parte imaginaria del número complejo c
- **`isreal(c)`** devuelve uno si c es un número real, cero en cc

```
>> A = complex(2,3);
>> isreal(A)
ans = 0
>> real(A)
ans = 2
 >> imag(A)
 ans = 3
```

Operaciones con números complejos

```
>> a = complex(2,3);      % 2 + 3i
>> b = complex(3,8);      % 3 + 8i

>> a+b
ans = 5.0000 +11.0000i

>> a-b
ans = -1.0000 - 5.0000i

>> a*b
ans = -18.0000 +25.0000i

>> a / b
ans = 0.4110 - 0.0959i

>> a ^ 2
ans = -5.0000 +12.0000i
```

Ejercicios

1) Cuales de los siguientes son nombres válidos de variables?

x_1 x1 12342 _hh %valor prog.cpp

2) ¿ Que hace el siguiente ejemplo?

```
>> valor = 0;
>> valor = valor + 1;
>> valor
```

3)¿ Como haría para intercambiar el valor de dos variables?

4) Convierta las siguientes fórmulas a expresiones en Matlab

3x 3x+y $\frac{x+y}{7}$ $\frac{x+y}{z+2}$ $area = \frac{base \cdot altura}{2}$

$F = \frac{9}{5}C - 32$ Celsius a Farenheit

Ejercicios

- Dados cuatro valores que representan dos puntos en el plano, *(x1, y1) , (x2, y2)*, calcula los coeficientes (a, b, c) de la ecuación general de la recta $ax+by+c=0$ que los une. El cálculo de los coeficientes se realiza mediante las expresiones: $a=y2-y1$, $b=x1-x2$, $c=y1\,x2-y2\,x1$
- Convierta las siguientes fórmulas a expresiones en Matlab

$$z=\frac{5x-4y}{3x^2+6y^4}-\frac{\sqrt{3x^2-28}}{123-y} \qquad x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$ (son dos)

- ¿ Como hago para hacer cosas mas complejas ?
  - Pedirle datos al usuario
  - Mostrar mensajes
  - Repetir varias veces el mismo conjunto de instrucciones
  - agrupar un conjunto de comandos y utilizarlo cuando quiera
  - etc, etc, etc.

Lectura / escritura de datos desde teclado

- Función input

```
>> n = input(‘ Ingrese nro de variables: ‘)
```

- Función disp

```
>> disp(‘ este es un script de prueba’)
```

- Control de formato
  - format
  - sprintf

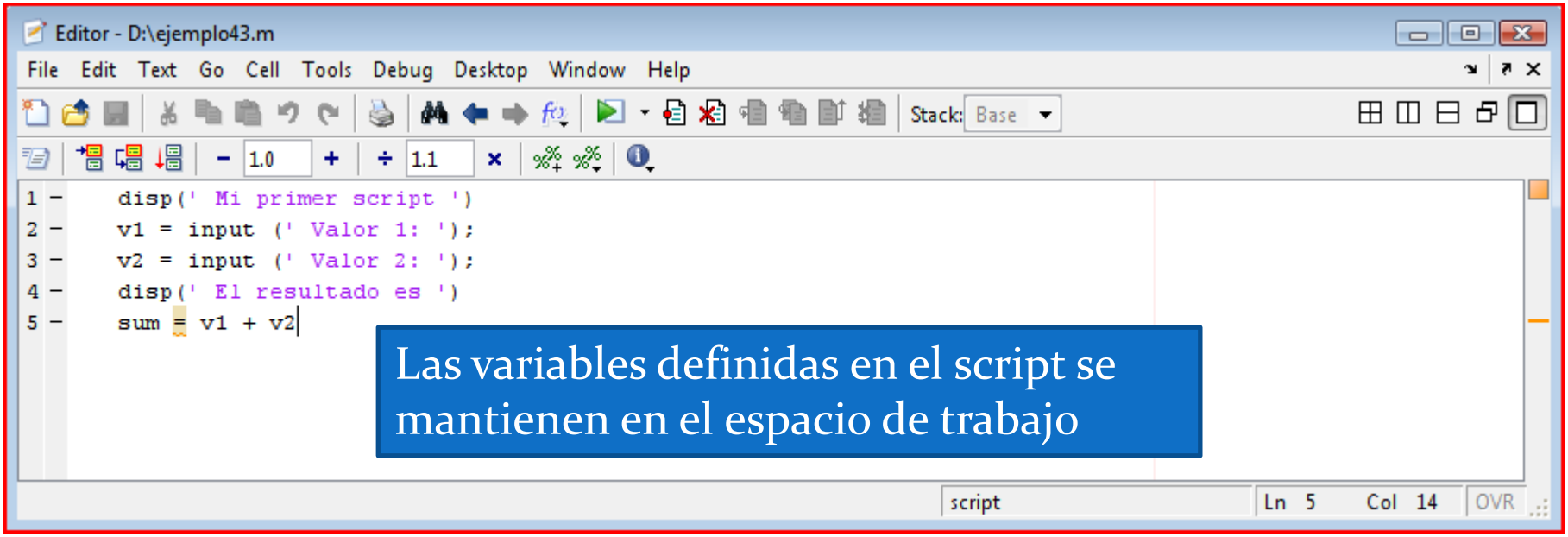
Investiguelos

Ficheros de Comandos: scripts

- Ficheros de texto con extensión .m
- Contienen un conjunto de instrucciones que se ejecutan sucesivamente cuando se teclea el nombre del fichero en la línea de comandos de MATLAB
- “Editor” de MATLAB (o cualquier otro)
- Un fichero de comandos puede llamar a otros ficheros de comandos.
- Si un fichero de comandos se llama desde de la línea de comandos de MATLAB, las variables que crea pertenecen al espacio de trabajo base de MATLAB y permanecen en él cuando se termina la ejecución de dicho fichero.

Ejemplo

- Utilizaremos el editor interno de Matlab



- ¿ como ejecutar ?

La ruta de trabajo

- En MATLAB, las funciones/scripts se encuentran en ficheros con extensión .m
- En principio, no todos los ficheros .m almacenados en el disco duro pueden ser accesibles desde MATLAB, i.e., pueden ser invocados en el interprete
- MATLAB tiene registrada las rutas (***paths***) de las carpetas/directorios en los que "buscar" funciones que se invocan en el interprete

La ruta de trabajo

- El comando **path** muestra las rutas de trabajo registradas en MATLAB

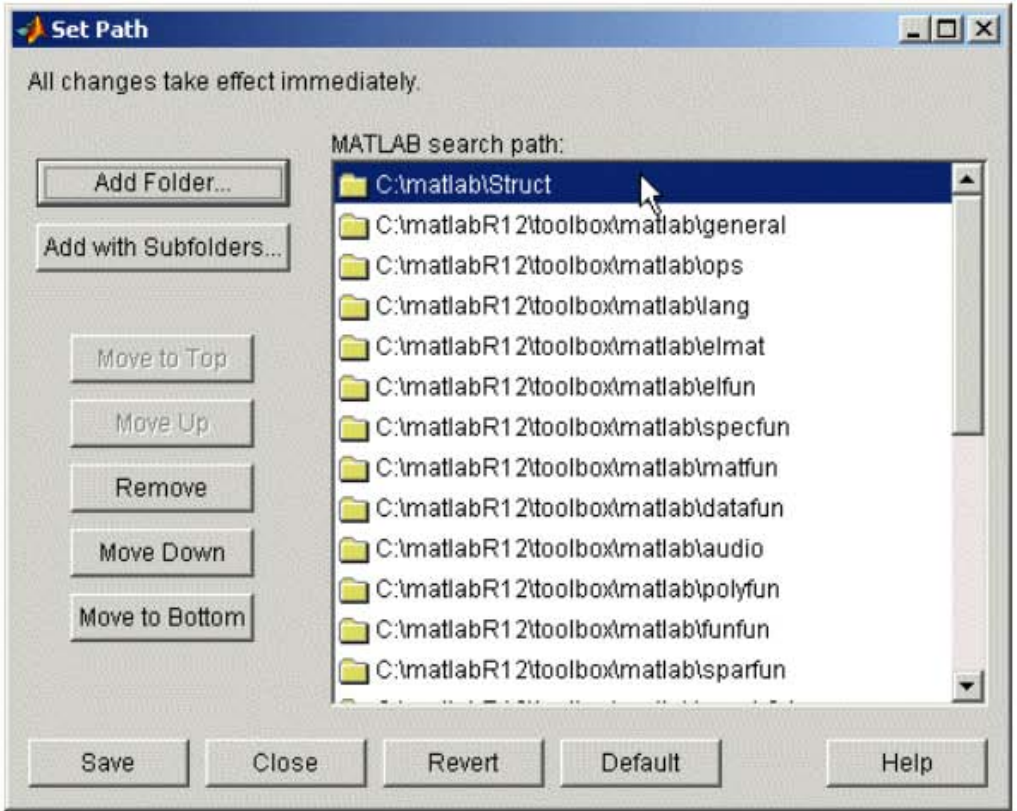
```
>> path
            MATLABPATH
  C:\matlabR12\toolbox\matlab\general
  C:\matlabR12\toolbox\matlab\lang
  ...
  C:\matlabR12\toolbox\matlab\winfun
  C:\matlabR12\toolbox\matlab\demos
  C:\matlabR12\toolbox\local
```

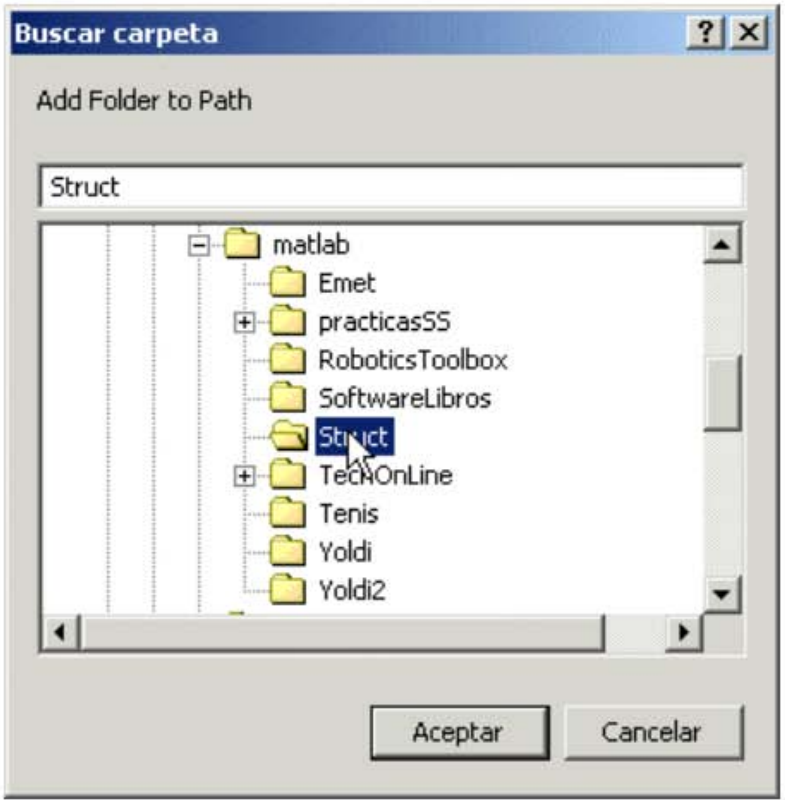
- El comando **addpath** permite añadir una o más rutas de trabajo al path de MATLAB, mientras que el comando **rmpath** elimina rutas

```
>> addpath 'C:\Informatica\Practicas\P1'
```

La ruta de trabajo

- Se pueden añadir y eliminar rutas de trabajo al/del path de MATLAB mediante la ventana de diálogo que se abre con el menú *File/Set Path*





La ruta de trabajo

- Una de las rutas de trabajo es siempre el **directorio actual**, i.e., aquel en el que uno se encuentra cuando está usando el interprete
- Para conocer y cambiar de directorio actual se emplean comandos similares a MS-DOS y Linux

```
>> pwd              % muestra la ruta del directorio actual
ans = C:\MATLAB\bin
>> cd ..            % sube un nivel en la jerarquía de directorios
>> pwd
ans = C:\MATLAB
>> cd ..
>> pwd
ans = C:\
>> cd Informatica  % baja un nivel en la jerarquía yendo a un
   directorio existente en el actual
>> pwd
ans = C:\Informatica
>> ls               % lista el contenido  del directorio actual
.            ..      ejercicio1.m        ejercicio2.m
```