P0

Imagen y Vídeo por Computador

• • •

Práctica 0: Matlab

Vicente Morell
vicente.morell@ua.es

Ingeniería Multimedia

Universitat d'Alacant
Universidad de Alicante

HUMAN ROBOTICS

Profesores

Profesor responsable
Teoría y Prácticas G1 y G4
Vicente Morell
vicente.morell@ua.es

Tutorías
Miércoles 9:30 – 13:30
0014PB005

Prácticas G2 y G3
Juan Martínez
jm.macia@ua.es

Tutorías
Lunes 10:00 – 11:00
Viernes 13:00-14:00
Concertar cita 0014PB002

Vicente Morell
vicente.morell@ua.es

Ingeniería Multimedia

Universitat d'Alacant
Universidad de Alicante

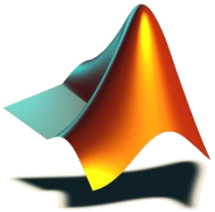
HUMAN ROBOTICS

Índice de contenidos

1. Introducción
2. Escritorio
3. Operadores escalares
4. Variables
5. Vectores y Matrices
6. Operadores y precedencias
7. Scripts
8. Funciones
9. Sentencias de selección
10. Sentencias de repetición

Vicente Morell vicente.morell@ua.es

Introducción

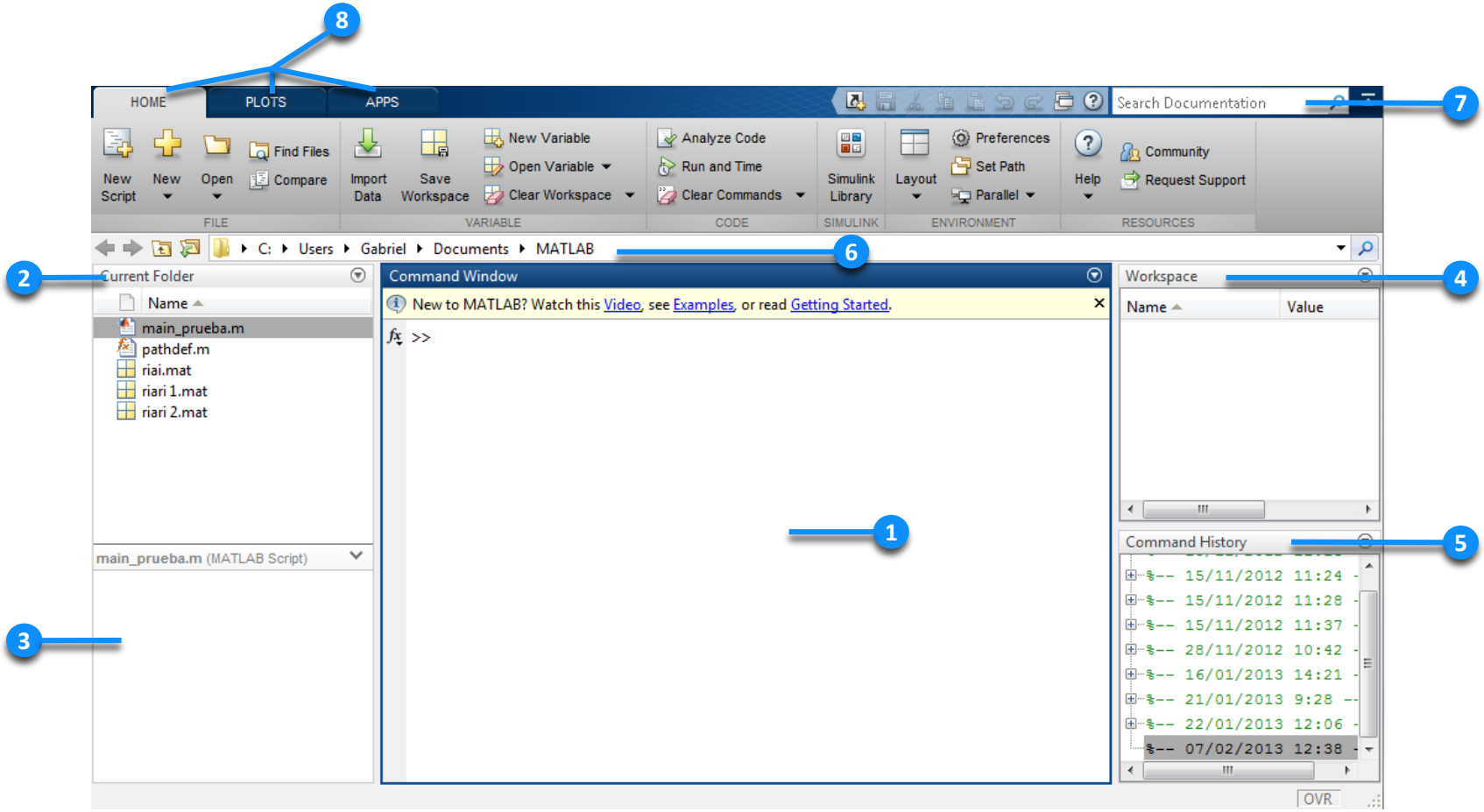


- ¿Qué es Matlab? → **MAT**rix **LAB**oratory.

- Es un programa para realizar cálculos numéricos con **vectores** y **matrices**, y por tanto se puede trabajar también con números escalares (tanto reales como complejos), con cadenas de caracteres y con otras estructuras de información más complejas.
- Matlab es un lenguaje de alto rendimiento para cálculos técnicos, es al mismo tiempo un **entorno** y un **lenguaje de programación**.
- Uno de sus puntos fuertes es que permite construir herramientas reutilizables personales. Podemos crear fácilmente **funciones propias** y programas especiales (conocidos como **M-archivos**) en código Matlab.
- Ofrece además una serie de **Toolbox** (también llamadas librerías), las cuales consisten en una colección especializada de M-archivos para trabajar en clases particulares de problemas.
- Matlab, a parte del cálculo matricial y álgebra lineal, también puede manejar polinomios, funciones, ecuaciones diferenciales ordinarias, gráficos …

Vicente Morell vicente.morell@ua.es

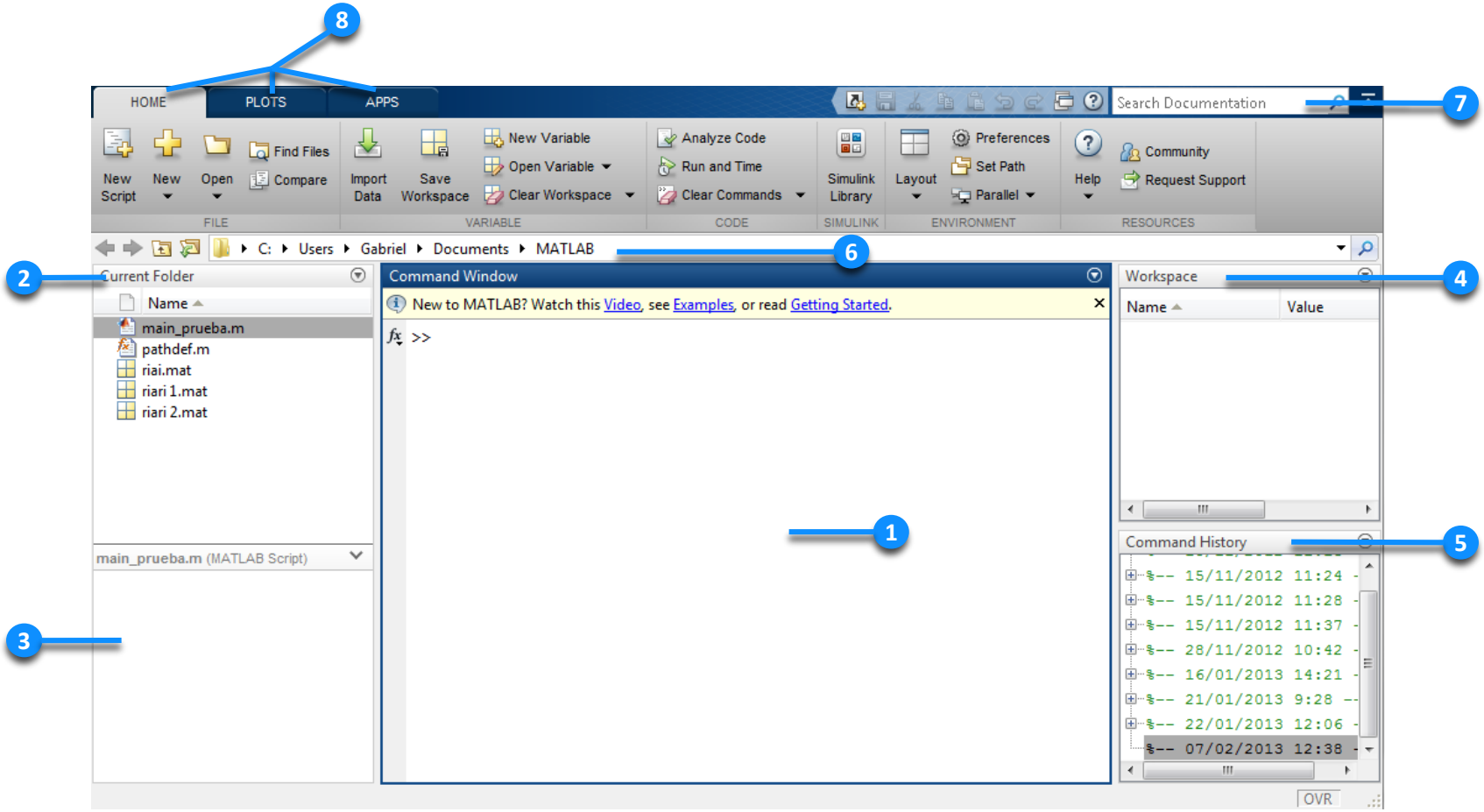
El escritorio de Matlab



1. **Command Window:** En esta ventana se ejecutan todas las instrucciones y programas. Se escribe la instrucción o el nombre del programa y se da a Enter.

Vicente Morell vicente.morell@ua.es

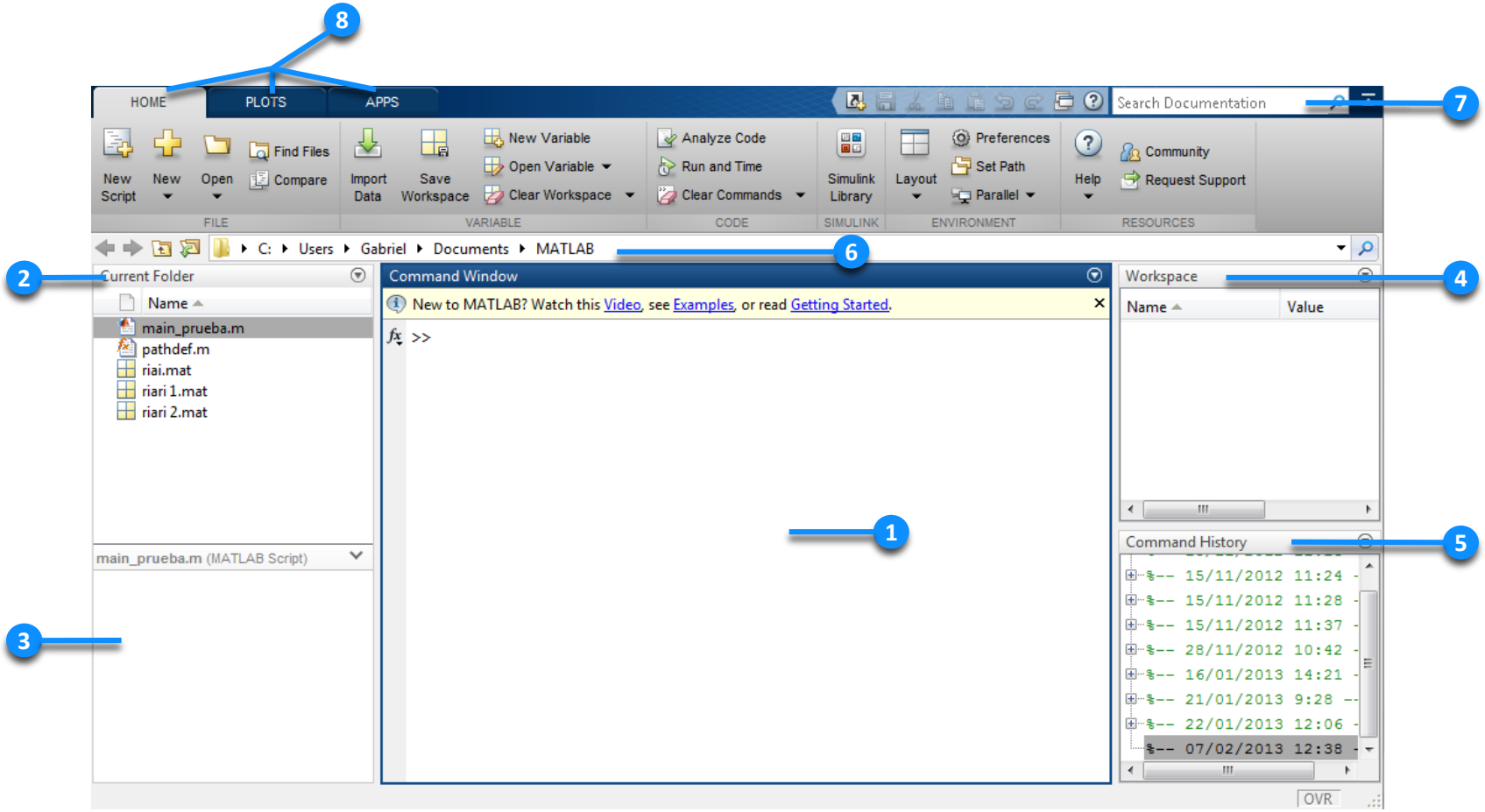
El escritorio de Matlab



2. **Current folder:** En esta ventana se muestra el directorio actual de trabajo. Matlab busca aquí y en el path el nombre de los scripts o funciones que se quieren utilizar.

Vicente Morell vicente.morell@ua.es

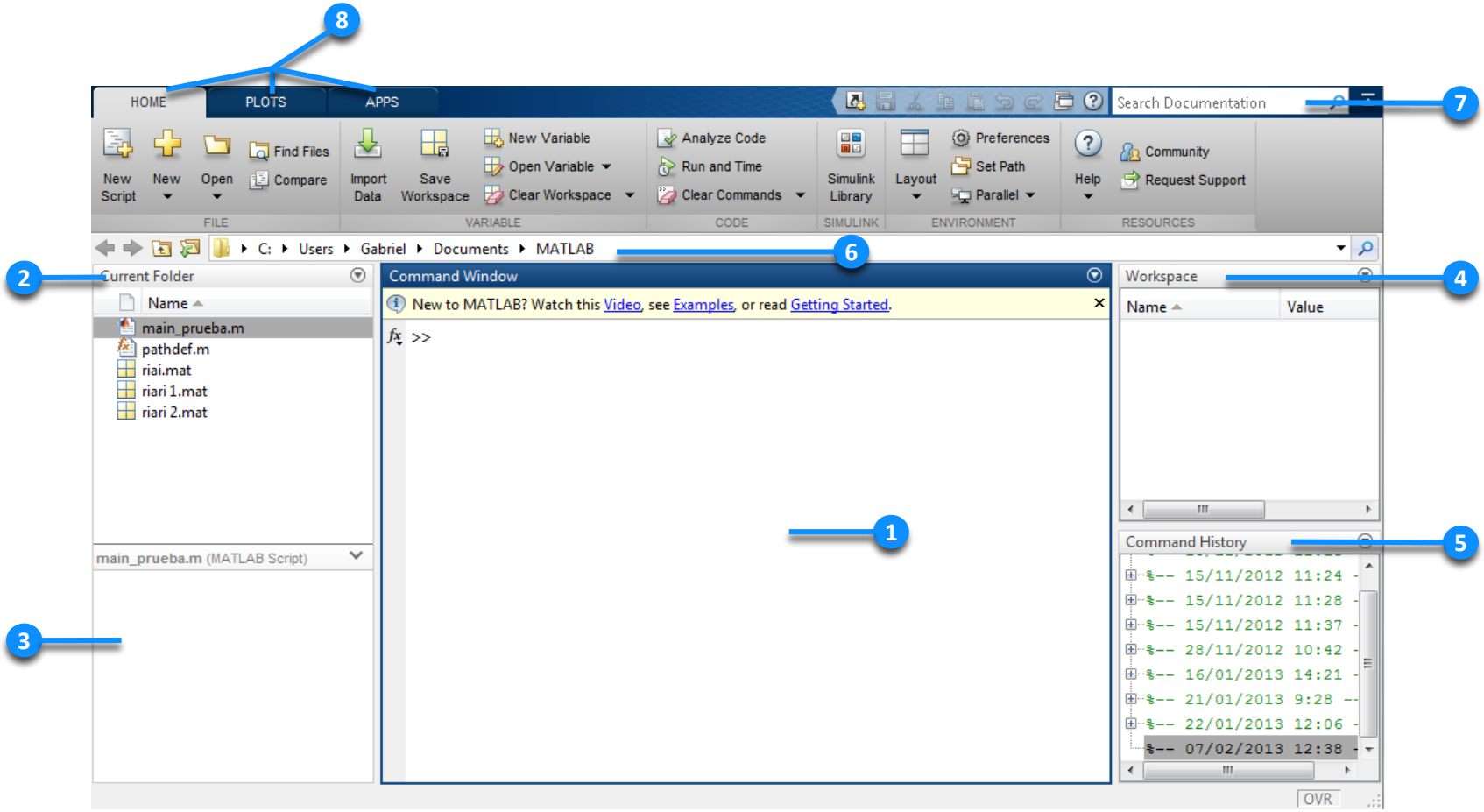
El escritorio de Matlab



3. **Details Window:** En esta ventana se muestra información adicional del archivo que se tenga seleccionado en la Carpeta actual.

Vicente Morell vicente.morell@ua.es

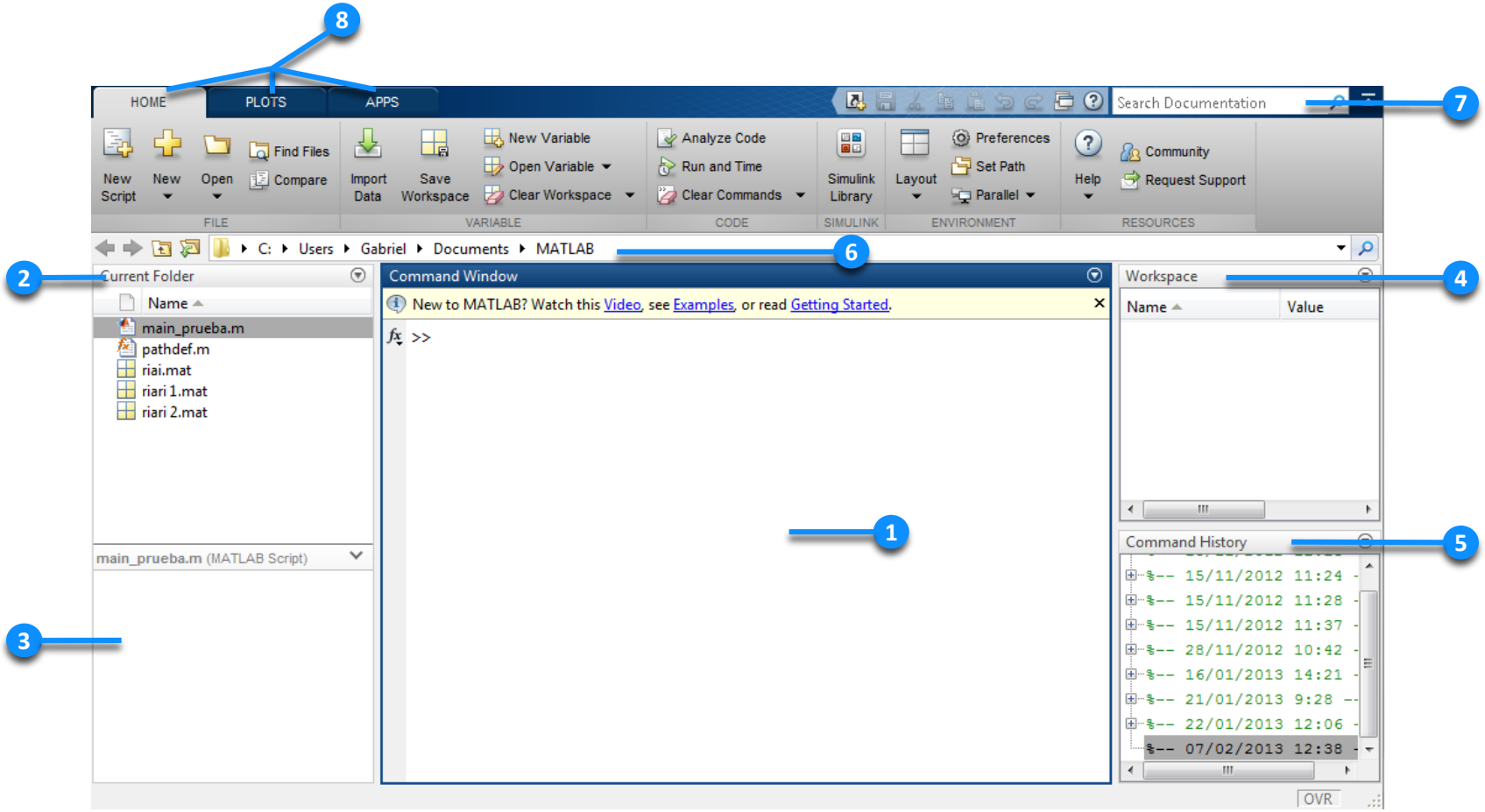
El escritorio de Matlab



4. **Workspace:** Esta ventana muestra el espacio de trabajo. En ella se puede encontrar una lista de las variables creadas durante la sesión actual de trabajo, así como su valor.

Vicente Morell vicente.morell@ua.es

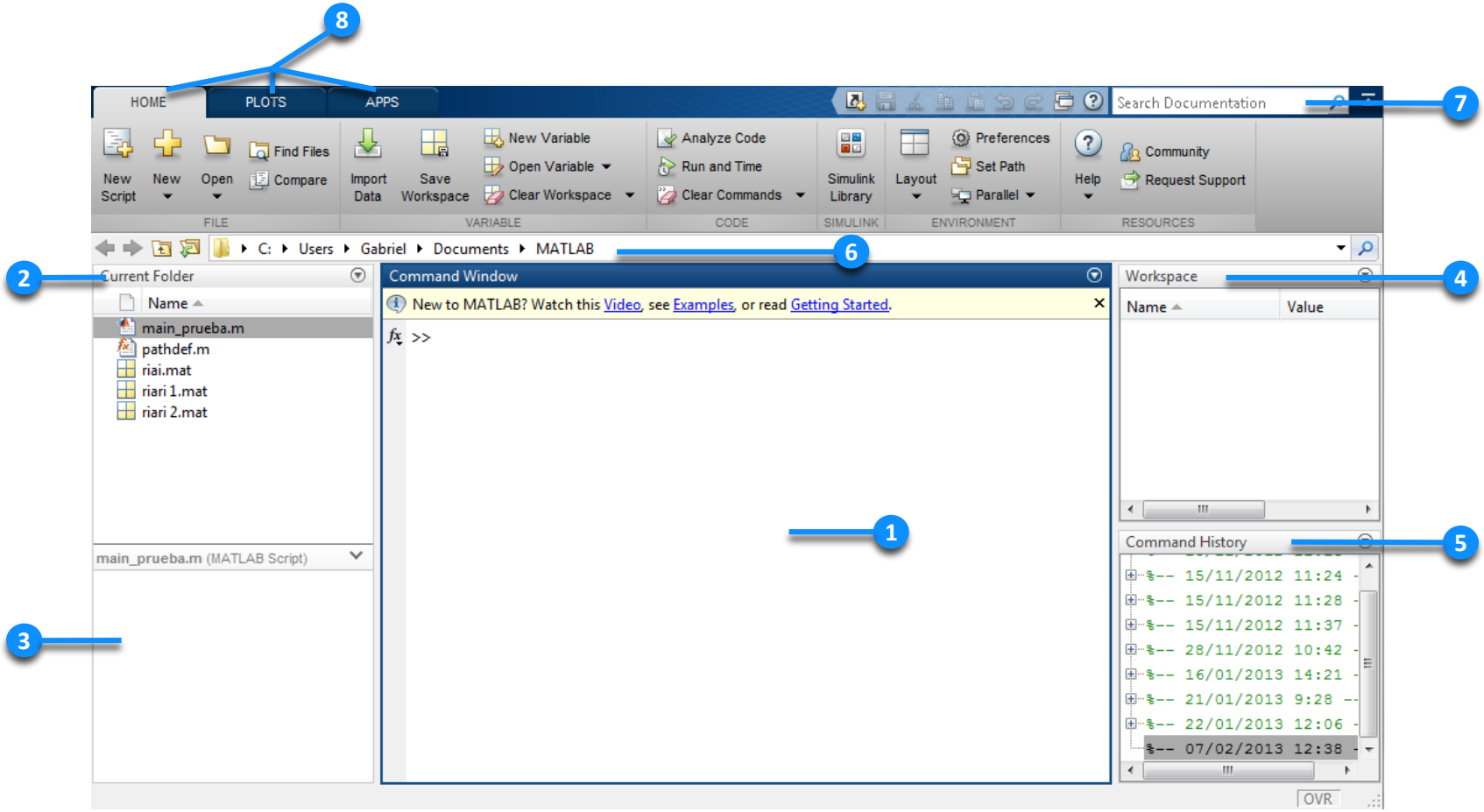
El escritorio de Matlab



5. **Command History:** Esta ventana muestra el historial de los comandos ejecutados durante las últimas sesiones.

Vicente Morell vicente.morell@ua.es

El escritorio de Matlab

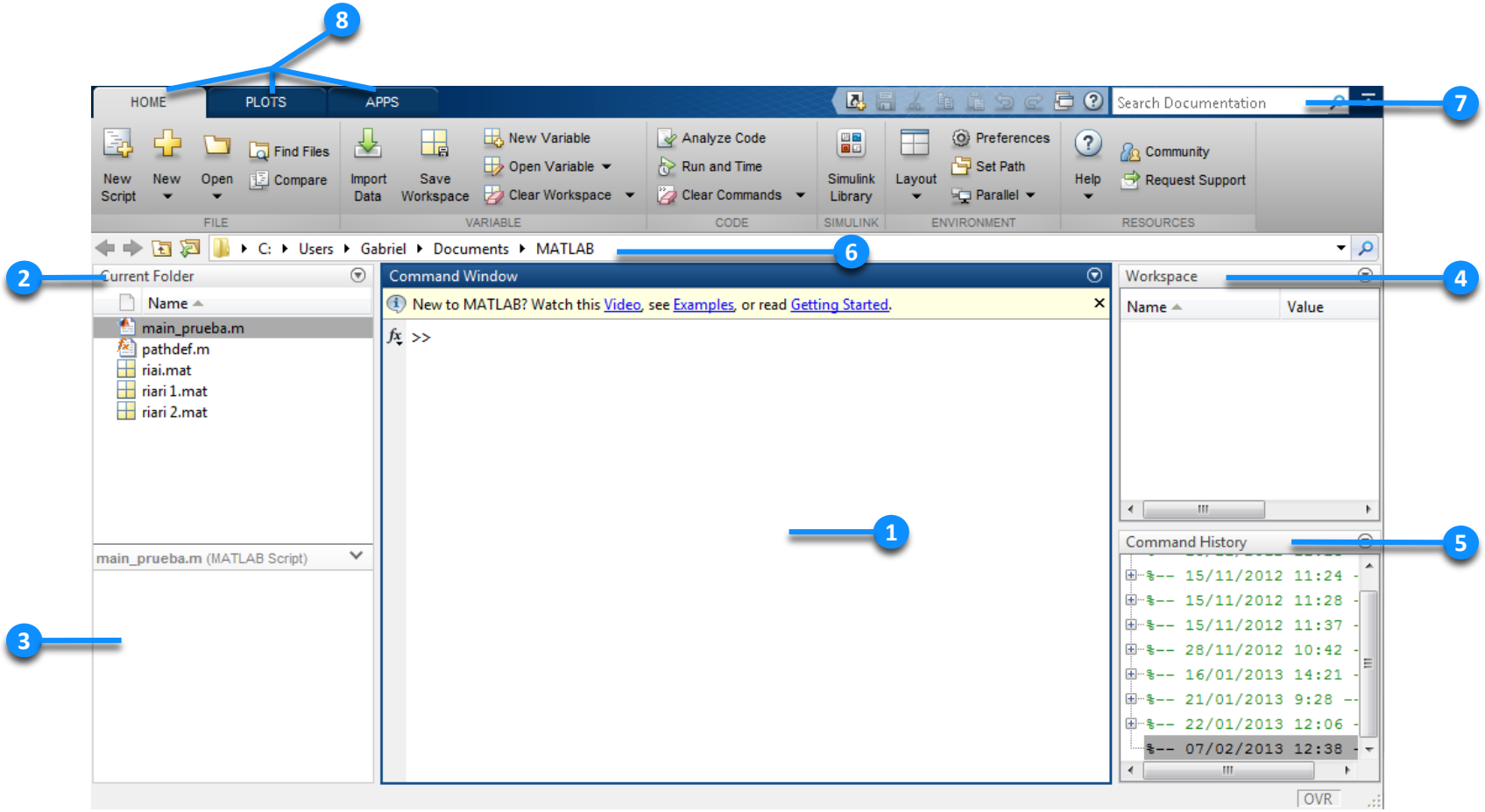


6. **Ruta a directorio actual:** Permite modificar la ruta del directorio actual.

7. **Búsqueda en la Documentación:** Permite buscar información sobre cualquier función.

Vicente Morell vicente.morell@ua.es

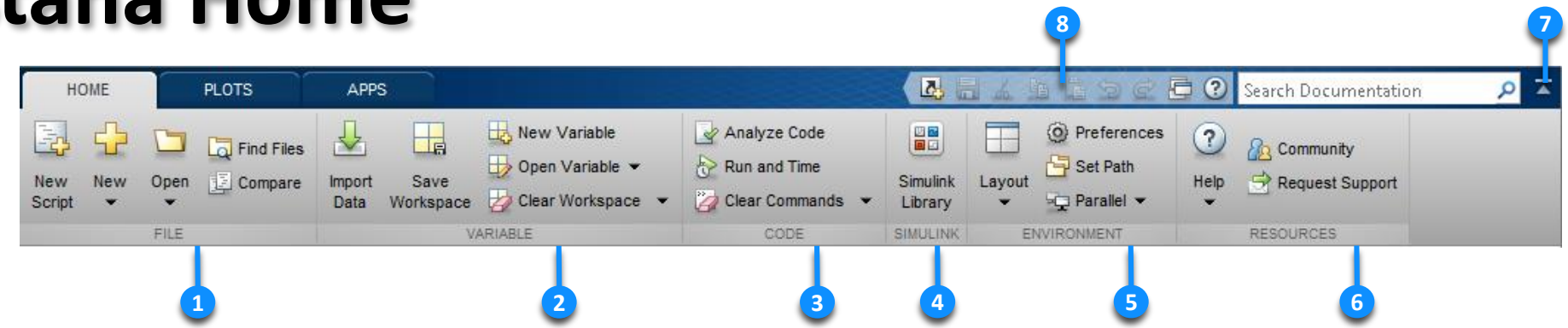
El escritorio de Matlab



8. **Pestañas:** Presentan las distintas acciones del programa agrupadas por categorías. Las pestañas principales son Home, Plots y Apps.

Vicente Morell vicente.morell@ua.es

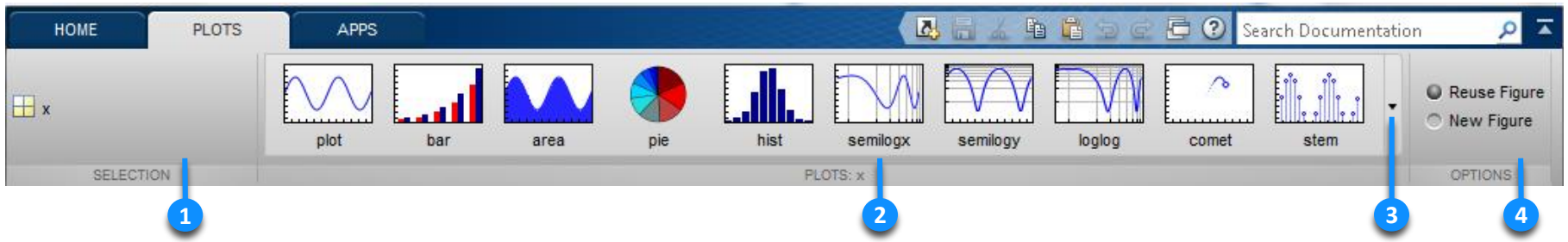
Pestaña Home



1. **File:** Este grupo permite crear un nuevo script, una nueva función, abrir scripts o funciones existentes, realizar búsquedas en disco de archivos o incluso comparar dos ficheros para encontrar duplicados.
2. **Variable:** Las acciones de este grupo permiten importar datos previamente guardados, guardar los datos de la sesión actual, crear o abrir variables, así como limpiar el espacio de trabajo.
3. **Code:** Permite analizar el código en busca de errores, lanzar el código calculando el tiempo invertido en la ejecución o limpiar la ventana de comandos o el historial.
4. **Simulink:** Lanza la librería de Simulink.
5. **Environment:** Permite acceder a las Preferencias de la aplicación, la presentación de ventanas, añadir nuevas rutas al Path por defecto o dividir trabajos para utilizar computación paralela.
6. **Resources:** La ayuda, Comunidad y petición de soporte.
7. Oculta las pestañas para ganar espacio de trabajo.
8. Barra de accesos directos.

Vicente Morell vicente.morell@ua.es

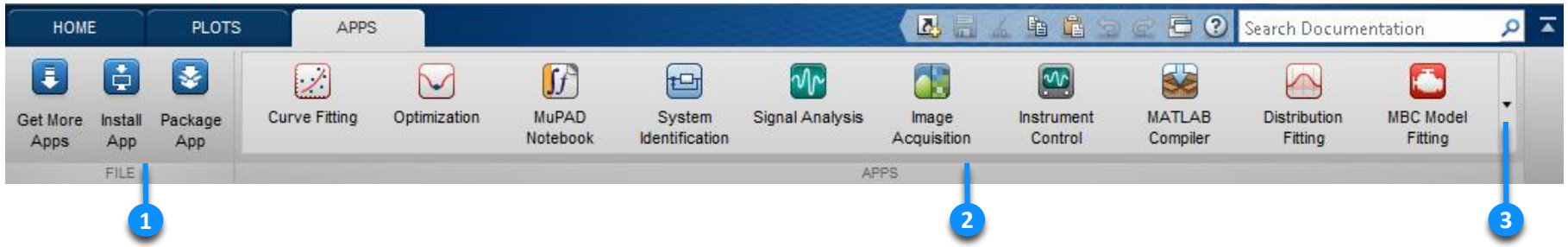
Pestaña Plots



1. **Selection:** Muestra la variable seleccionada sobre la que se quiere realizar el dibujo de alguna de las gráficas disponibles.
2. **Plots:** Presenta toda una serie de gráficas disponibles en Matlab para mostrar visualmente los datos.
3. Permite acceder a todas las opciones de gráficos presentes en Matlab para elegir el que más le convenga a los datos seleccionados.
4. **Options:** Permite elegir si queremos nuestro gráfico en una nueva figura o que realice el plot sobre la misma figura (equivale a la instrucción en ventana de comandos **hold off/on**).

Vicente Morell vicente.morell@ua.es

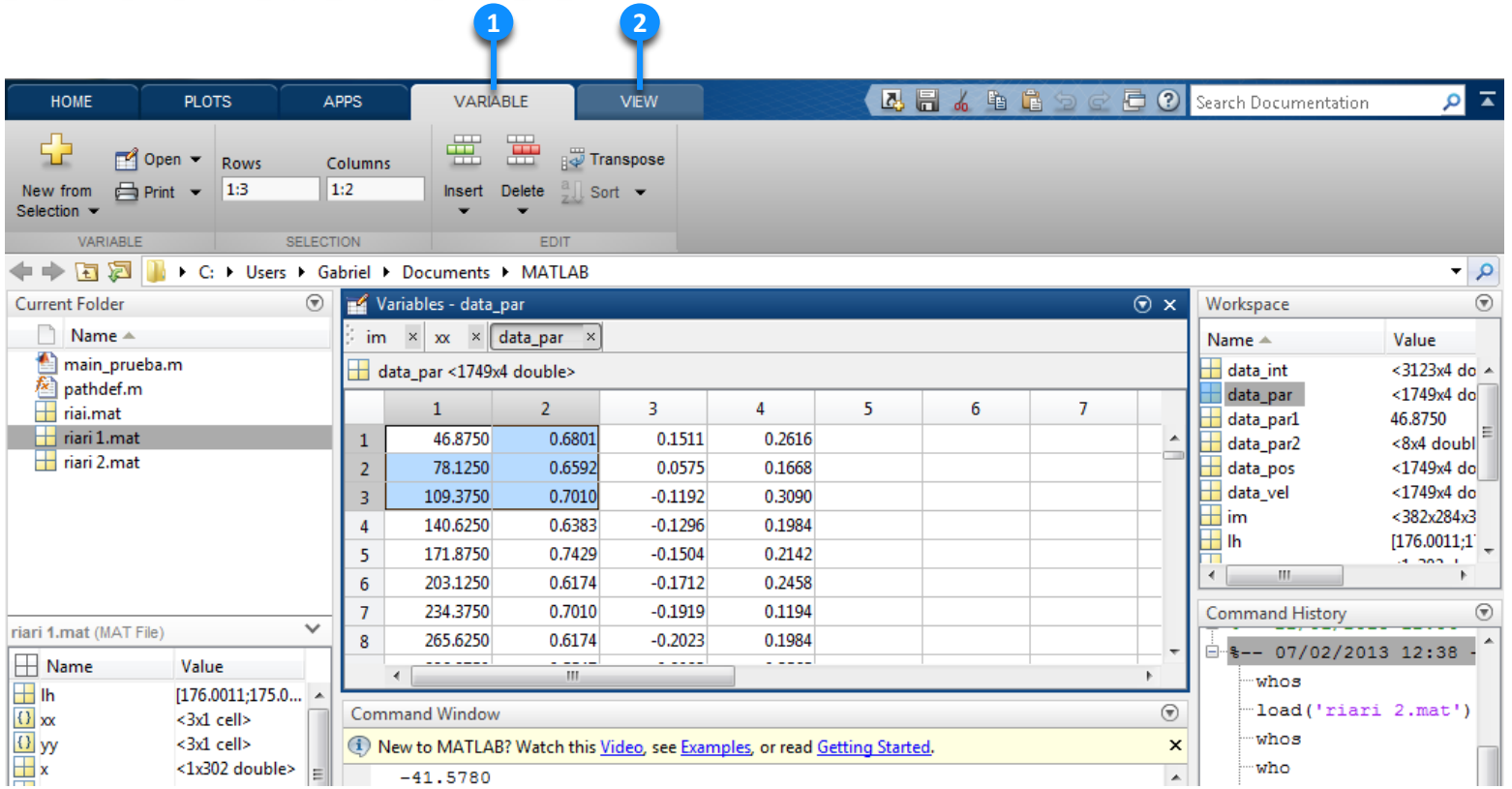
Pestaña Apps



1. **File:** Muesta las acciones para obtener más toolboxes, instalar una toolbox en concreto o comprimirla para instalarla en otra copia de Matlab.
2. **Apps:** Las distintas Toolbox presentes en la edición de Matlab.
3. Permite acceder a todas las Toolbox presentes.

Vicente Morell vicente.morell@ua.es

Pestañas adicionales



Escritorio

1. **Variable:** Pestaña que aparece cuando seleccionamos una variable.
2. **View:** Pestaña adicional que aparece al seleccionar una variable.

Expresiones básicas

Se puede usar MATLAB para realizar **cálculos numéricos**:

```
>> 2+4

ans =

    6
```

```
>> 2*4

ans =

    8
```

```
>> 2-4+5-3

ans =

    0
```

```
>> 2+4

ans =

    6
```

```
>> 2+3*3

ans =

    11
```

```
>> (2+3)*3

ans =

    15
```

```
>> 24/2+4

ans =

    16
```

```
>> 24/(2+4)

ans =

    4
```

```
>> 2^4

ans =

    16
```

```
>> 2^4+3

ans =

    19
```

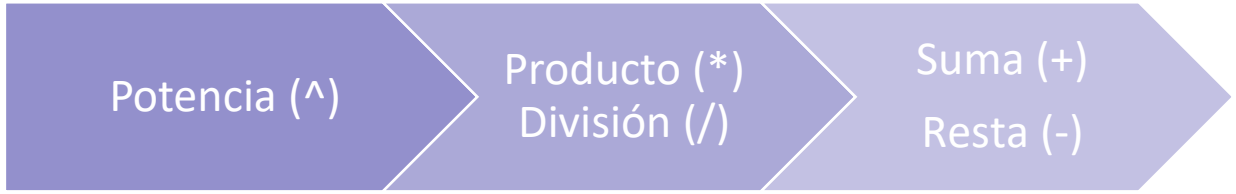
```
>> 2^(4+3)

ans =

    128
```

```
>> 2-4^5*4-3

ans =

   -4097
```


Vicente Morell vicente.morell@ua.es

Orden de precedencia de los operadores escalares

Mayor precedencia



Menor precedencia

Orden de evaluación de los operadores aritméticos:

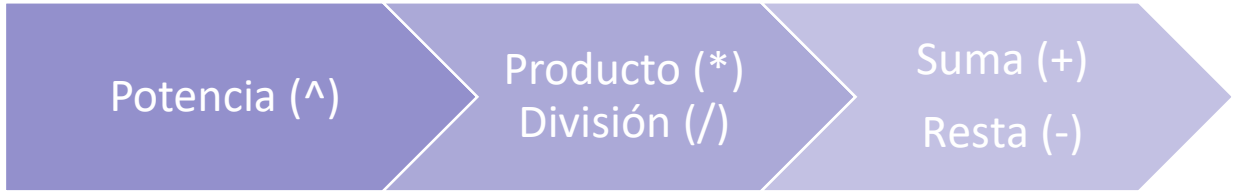
1. Primero los de más precedencia.
2. En caso de igual precedencia, de izquierda a derecha

5 + 2 ^ 3 – 4 * 8 / 2	1º Potencia 2^3
5 + 8 – 4 * 8 / 2	2º Producto 4*8
5 + 8 – 32 / 2	3º División 32/2
5 + 8 – 16	4º Suma 5+8
13 – 16	5º Resta 13-16
-3	6º Resultado

Vicente Morell vicente.morell@ua.es

Orden de precedencia de los operadores escalares

Mayor precedencia



Menor precedencia

Uso de paréntesis para modificar el orden de evaluación:

- Se evalúa primero la expresión del **paréntesis más interno**.

((5 + 2 ^ 3) – 4) * (8 / 2) → 1º Potencia 2^3

((5 + 8) – 4) * (8 / 2) → 2º Suma 5+8

(13 – 4) * (8 / 2) → 3º Resta 13-4

9 * (8 / 2) → 4º División 8/2

9 * 4 → 5º Producto 9*4

36 → 6º Resultado

Vicente Morell vicente.morell@ua.es

Variables

- Los datos se almacenan en **variables**.
- La atribución de un valor a una variable se denomina **asignación**.
- Declaración de variables en Matlab:
  - No es necesario declararlas antes de asignarles un valor
  - Es necesario asignarles un valor antes de usarlas en una expresión.
- Reglas para construir **identificadores** o **nombres** de variables:
  - Letras **a-z**, **A-Z** (ASCII), **números** y "_"
  - Máximo número de caracteres: **namelengthmax**
  - El **primer carácter** ha de ser alfabético
  - Se distingue entre **mayúsculas y minúsculas**
  - No se pueden usar las palabras reservadas
- Las **palabras reservadas** son identificadores utilizados por el lenguaje.

Asignación de una **constante**

```
>> X=16

X =

    16
```

Asignación de una **variable**

```
>> Y=X

Y =

    16
```

Asignación de una **expresión**

```
>> Z = X^2 + 3*Y + 1

Z =

   305
```


Vicente Morell vicente.morell@ua.es

Variables

- Si se intenta utilizar una variable no declarada, se produce un error:

```
>> 3-s*45^t
??? Undefined function or
variable 's'.
```

- Matlab dispone de una serie de **variables predefinidas**:
  - **ans** Si el último valor calculado no se asigna a una variable, entonces se asigna automáticamente a **ans**.
  - **eps** Precisión de los cálculos en coma flotante. Tolerancia utilizada en los cálculos.
  - **realmax** Mayor número en coma flotante que puede ser representado en el ordenador.
  - **realmin** Menor número positivo en coma flotante que puede ser representado en el ordenador.
  - **pi** 3.1415926535897...
  - **i**, **j** Unidad imaginaria.
  - **inf** Infinito. Resultado de cálculos como **2/0**.

Vicente Morell vicente.morell@ua.es

Vectores y Matrices

- Para crear una matriz se introduce una lista de valores de la siguiente manera:
  - Los elementos de cada fila se separa mediante **blancos** o **comas**
  - Las filas se separan mediante punto y coma, "**;**"
  - Toda la lista de elementos se rodea mediante corchetes, **[ ]**

Matriz

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

x =

   1   2   3
   4   5   6
   7   8   9
```

Vector fila

```
>> f = [2 6 1]

f =

   2   6   1
```

Vector columna

```
>> c = [4; 1; 9]

c =

   4
   1
   9
```

Matriz de 1x1: escalar

```
>> e = [7]

e =

   7
```

```
>> e = 7

e =

   7
```

Matriz lógica

```
>> a = logical( [1 0 1; 0 1 1; 1 0 0] )

a =

   1   0   1
   0   1   1
   1   0   0
```


Vicente Morell vicente.morell@ua.es

Vectores y Matrices

- Operador "**:**". Este operador genera un vector fila con una secuencia de números según una ley simple.
- Usando **dos** elementos (inicio y límite del intervalo):
  - 1:5 equivale a [1 2 3 4 5]
  - 1.5:4 equivale a [1.5 2.5 3.5]
- Usando **tres** elementos (inicio, incremento, límite):
  - 1:3:10 equivale a [1 4 7 10]
  - 1:1:5 equivale a [1 2 3 4 5]
  - 5:–1:1 equivale a [5 4 3 2 1]
  - 0:pi/4:pi equivale a [0 0.7854 1.5708 2.3562 3.1416]
- Los índices de las matrices y vectores comienzan en **1** y el último elemento es accesible mediante el índice **end**.

Vicente Morell vicente.morell@ua.es

Operaciones con matrices

- Operaciones elemento a elemento. Los operandos deben ser del **mismo tamaño** o uno de ellos ha de ser **escalar**.
  - **A + B** Suma
  - **A – B** Sustracción
  - **A .* B** {A(i,j) * B(i,j)}
  - **A ./ B** {A(i,j) / B(i,j)}
  - **A .\ B** {B(i,j) / A(i,j)}
  - **A .^ B** {A(i,j) ^ B(i,j)}
  - **A .'** Traspuesta
- Operaciones matriciales. Los operandos deben ser de **tamaño compatible** o uno de ellos ha de ser **escalar**.
  - **A * B** Producto matricial
  - **A / B** A * B^–1
  - **A \ B** A^–1 * B
  - **A ^ B** Potencia de matrices
  - **A '** Traspuesta (conjugada)

Vicente Morell vicente.morell@ua.es

Selección de elementos

- Es posible seleccionar un elemento individual de una matriz:
  - **A(i, j)** es el elemento que se encuentra en la fila ***i*** y columna ***j*** de A.
- Los elementos de una matriz se pueden utilizar en expresiones:
  - W = **A(2,3)** * x / 2
- Si se hace referencia a un elemento fuera de la matriz, se produce un **error**.
- Se puede asignar un valor a un elemento de una matriz:
  - **A(2,3)** = 2 * pi * r
  - Si se asigna un valor a un elemento fuera de la matriz, ésta **crece** lo necesario para dar cabida al elemento. El resto de los elementos creados se rellena a 0.
- Selección múltiple utilizando **vector fila**, rangos o toda una dimensión con el operador "**:**" y **matrices lógicas**.
- Ejemplos con un vector fila A, que vale [3 7 2 1 7 8 3 5 6]
  - A(1, [3 7 5]) vale [2 3 7]
  - A(1, 3:7) vale [2 1 7 8 3]
  - A(1, 1:2:9) vale [3 2 7 3 6]
  - A(1, 1:2:9) = 0 produce [0 7 0 1 0 8 0 5 0]
  - A(1, 12) = 3 produce [3 7 2 1 7 8 3 5 6 0 0 3]

Vicente Morell vicente.morell@ua.es

Concatenación de matrices

```
>> a = [1 2; 3 4]

a =

   1   2
   3   4
```

- Concatenación de una **fila**:

```
>> b = [a ; 7 8]

b =

   1   2
   3   4
   7   8
```

- Concatenación de una **columna** :

```
>> c = [a [1;1]]

c =

   1   2   1
   3   4   1

>> a(:,3) = [1; 1]

a =

   1   2   1
   3   4   1
```


Vicente Morell vicente.morell@ua.es

Otras operaciones con matrices

- Borrar una fila: **A(1,:)=[ ]**
- Borrar una columna: **A(:,1)=[ ]**
- Generación de matrices:
  - Matriz de ceros, **zeros(n,m)**
  - Matriz de unos, **ones(n,m)**
  - Matriz identidad **eye(n,m)**
  - Matriz de elementos aleatorios **rand(n,m)**
- Funciones para vectores y matrices (aplicadas algunas de estas funciones a matrices, realizan dichas operaciones por columnas):
  - **sum(v)** suma los elementos de un vector
  - **prod(v)** producto de los elementos de un vector
  - **dot(v,w)** producto escalar de vectores
  - **cross(v,w)** producto vectorial de vectores
  - **mean(v)** (hace la media)
  - **diff(v)** (vector cuyos elementos son la resta de los elementos de v)
  - **[y,k]=max(v)** valor máximo de las componentes de un vector (k indica la posición), **min(v)** (valor mínimo). El valor máximo de una matriz M se obtendría como **max(max(M))** y el mínimo **min(min(v))**
  - **[n,m]=size(A)** devuelve el número de filas y columnas
  - Matriz inversa: **B=inv(A)**, rango: **rank(A)**

Vicente Morell vicente.morell@ua.es

Operadores relacionales

- Los operadores relacionales realizan una comparación entre los operandos, devolviendo un valor lógico (0 o 1).
  - A < B menor que
  - A > B mayor que
  - A <= B menor o igual que
  - A >= B mayor o igual que
  - A == B igual a
  - A ~= B distinto de

```
>> a = [-1 6 5 6];
>> b = [-3 4 5 4];
>> a>b

ans =

   1   1   0   1

>> a>=b

ans =

   1   1   1   1
```

```
>> a==b

ans =

   0   0   1   0

>> a~=b

ans =

   1   1   0   1

>> a>4

ans =

   0   1   1   1
```


Vicente Morell vicente.morell@ua.es

Operadores lógicos

- Los operadores lógicos relacionan valores lógicos entre si, para obtener otro resultado lógico.
  - Negación lógica (**~**)
  - Y elemento a elemento (**&**)
  - O elemento a elemento (**|**)
- Los operadores lógicos escalares se denominan "**vagos**" porque no evalúan el operando de la derecha si no es necesario.
  - Y escalar vago (**&&**)
  - O escalar vago (**||**)

```
>> a=1;
>> b=0;
>> x=(b~=0)&&(a/b > 18.5)

x =

   0
```

```
>> a = [0 0 1 1];
>> b = [1 1 1 0];
>> a|b

ans =

   1   1   1   1

>> a&b

ans =

   0   0   1   0

>> ~ans

ans =

   1   1   0   1
```

- Si el resultado de **(b ~= 0)** es falso, ya no es necesario evaluar el operando de la derecha. De este modo se evitan cálculos innecesarios e incluso errores.

Vicente Morell vicente.morell@ua.es

Orden de precedencia de los operadores

Paréntesis

Operadores matriciales
- Traspuesta (.')
- Potencia (.^)
- Traspuesta compleja conjugada (')
- Potencia matricial (^)

Operadores unarios

Operadores aritméticos
- Multiplicación (.*)
- División (./) (. \)
- Producto matrices (*)
- División matrices (/) (\)
- Suma (+)
- Resta (-)

Operador dos puntos (:)

Operadores relacionales
- Menor (<) (<=)
- Mayor (>) (>=)
- Igual (==)
- Distinto (~=)

Operadores lógicos
- Y elemento a elemento (&)
- O elemento a elemento (|)
- Y vago (&&)
- O vago (||)

Vicente Morell vicente.morell@ua.es

Scripts

- Compuestos por cualquier **secuencia de instrucciones** válidas en Matlab.
- El script se almacena en un fichero con la extensión **.m**
- Para ejecutarlos se teclea su nombre en la línea de comandos de Matlab.
- Variables utilizadas por los scripts:
  - Pueden utilizar cualquiera de las **variables del workspace**.
  - Las variables que crea también se **almacenan en el workspace**.
  - Una vez finalizada la ejecución, las variables que haya creado, **permanecen en el workspace**.
- Los scripts han de ser usados cuando se ha de **repetir muchas veces una misma secuencia** de operaciones.

```
% ivc_script1: declara dos matrices (a y b) y obtiene
%el producto matricial y la suma a+b

%Datos
a=[3 4 1;-2 -1 -9;-7 4 4];
b=[1 2 1;3 -1 2;8 -4 0];

%Cálculo y salida por pantalla
disp('El producto matricial es '); disp(a*b);
disp('La suma matricial es '); disp(a+b);
```

```
>> help ivc_script1
 ivc_script1: declara dos matrices (a y b) y obtiene
el producto matricial y la suma a+b
```


Vicente Morell vicente.morell@ua.es

Funciones

- Reciben **parámetros de entrada**.
- Devuelven **valores de salida**.
- Operan sobre **variables locales**, sólo conocidas por la función. Aunque tengan el mismo nombre que otras existentes en el workspace.
- Para ejecutarlos se utilizan como cualquier otra función de Matlab.
- La primera línea (no comentario) comienza por la palabra clave **function**.
  - La función se llama igual que el fichero (sin la extensión).
  - La variable **resultado** recibirá el resultado de la función.
  - Los ***parámetros formales*** a y b sólo existen dentro de la función.
  - Al llamar a la función, los argumentos 3 y 4 se asignan a los argumentos formales a y b.
- La variable **resultado** sólo existe durante la ejecución de la función.

```
% ivc_funcion1: Suma al cuadrado de dos números
function resultado = ivc_funcion1(a, b)
resultado = (a+b)^2;
```

```
>> help ivc_funcion1
 ivc_funcion1: Suma al cuadrado de dos números
```

```
>> ivc_funcion1(3,4)

ans =

   49
```


Vicente Morell vicente.morell@ua.es

Funciones

- Los parámetros proporcionados a las funciones pueden ser **constantes**, **variables** o **expresiones.**
- Una función escalar aplicada a una matriz opera sobre cada elemento, devolviendo otra matriz del mismo tamaño.
- Ciertas funciones pueden devolver valores **simples** o **múltiples**.

```
>> cos(0.54)

ans =

   0.8577
```

```
>> cos(a^3-0.5)

ans =

   0.8928
```

```
>> a=0.32

a =

   0.3200

>> cos(a)

ans =

   0.9492
```

```
>> a=0:0.5:1

a =

         0    0.5000    1.0000

>> cos(a)

ans =

    1.0000    0.8776    0.5403
```

```
>> a= [-5 5 4 8 29 -58 3 0];
>> maximo = max(a)

maximo =

   29

>> [maximo posicion] = max(a)

maximo =

   29


posicion =

    5
```

Vicente Morell vicente.morell@ua.es

Funciones

- Es posible definir funciones que devuelvan **más de un resultado**. Para ello es necesario:
  - Definir la estructura del Resultado: **[may, men]**
  - Asignar un valor a cada una de las variables que componen el resultado (**may** y **men**).
- Se pueden crear **subfunciones**:
  - Esto puede hacerse utilizando distintas funciones, cada una en su propio fichero **.m**.
  - Si se añaden las funciones dentro del mismo archivo **.m** dónde se define la función principal, estas funciones solo pueden usarse desde otra función de este mismo fichero.

```
function [may, men] = sumaymen(vec, min)
% SUMAYMEN(V, M)
% Suma los numeros de un vector que sean mayores o iguales que uno de referencia
% y los menores que este
% Parametros:
% V: vector de numeros a sumar
% M: minimo valor que puede sumarse
% Resultado: [MAYORES MENORES]
% MAYORES: suma de los mayores o iguales
% MENORES: suma de los menores
% Calculo
elementos = vec>=min;
may = sum(vec(elementos));
men = sum(vec(~elementos));
```

Vicente Morell vicente.morell@ua.es

Sentencias de selección if

- Sintaxis: La estructura de este bloque es:

```
if expresión lógica
   secuencia de instrucciones 1
else
   Secuencia de instrucciones 2
end
```

- Semántica:
  1. Se evalúa **expresión lógica**. El resultado puede ser verdadero (1) o falso (0) .
  2. Si es verdadero (1), se ejecuta **secuencia de instrucciones 1.**
  3. Si es falso (0) , se ejecuta **secuencia de instrucciones 2.**
  4. Después se ejecutan la instrucciones que siguen al **end**.

```
% RAIZ: calcula la raìz cuadrada si el
% numero es positivo. Si no, no hace nada
x = input('Introduce un numero: ');
if x > 0
   x = sqrt(x);
end
disp(x);
```

```
>> raiz
Introduce un número: 16
    4

>> raiz
Introduce un número: -16
  -16
```


Vicente Morell vicente.morell@ua.es

Sentencias de selección if

- Si la expresión lógica es matricial:
  - el resultado será verdadero cuando lo sea **cada elemento** del vector lógico resultante.
  - Si la matriz lógica resultante es **vacía**, el resultado se trata como falso.

```
>> raiz
Introduce un número: [1 2 3 4]
   1.0000    1.4142    1.7321
2.0000

>> raiz
Introduce un número: [-1 2 3 4]
   -1     2     3     4
```

```
% EsPar: indica si un n˙mero es par
n = input('Introduce un n˙mero: ');
if mod(n, 2) == 0
   disp('par');
else
   disp('impar');
end
```

```
>> parimpar
Introduce un número: 23
impar
>> parimpar
Introduce un número: 54
par
```


Vicente Morell vicente.morell@ua.es

Sentencias de selección switch

- Se utiliza para determinar la ejecución de una secuencia de instrucciones entre **múltiples alternativas**.
- La secuencia de instrucciones a ejecutar **se selecciona** en función del resultado de evaluar una expresión **criterio** que puede tomar una colección predefinida de valores numéricos o cadenas de caracteres.

```
switch criterio
   case valor1
     Secuencia sentencias 1
   case valor2
     Secuencia sentencias 2
   ...
   case valorN
     Secuencia sentencias N
   otherwise
     Secuencia sentencias N+1
end
```


Vicente Morell vicente.morell@ua.es

Sentencias de selección switch

- **Criterio** puede ser una expresión de tipo numérico o cadena de caracteres
  - Conviene **evitar** expresiones que produzcan **números reales** porque el error de redondeo puede hacer que el resultado nunca coincida con un valor especificado en un case.
- Cada uno de los **valor i** es uno o más valores posibles que puede tomar el **criterio**.
- La forma de estos valores puede ser:
  - Un **único** valor.
  - Una lista de valores entre llaves **{v1, v2, v3…}**.

```
switch criterio
   case valor1
     Secuencia sentencias 1
   case valor2
     Secuencia sentencias 2
   …
   case valorN
     Secuencia sentencias N
   otherwise
     Secuencia sentencias N+1
end
```


Vicente Morell vicente.morell@ua.es

Sentencias de selección switch

- Se evalúa la expresión **criterio**.
- Si el resultado de evaluar **criterio** coincide con el valor o lista de valores de **valor1** entonces se ejecuta el conjunto de instrucciones **Secuencia sentencias 1**
- Si no, si el resultado de evaluar **criterio** coincide con el valor o lista de valores de **valor2** entonces se el conjunto de instrucciones **Secuencia sentencias 2**
- ... y así sucesivamente con cada CASE
- En caso de que la expresión no satisfaga ningún valor de los especificados en alguno de los CASE, se ejecuta el conjunto de instrucciones **Secuencia sentencias n+1**

```
switch criterio
   case valor1
      Secuencia sentencias 1
   case valor2
      Secuencia sentencias 2
   ...
   case valorN
      Secuencia sentencias N
   otherwise
      Secuencia sentencias N+1
end
```


Vicente Morell vicente.morell@ua.es

Sentencias de repetición

- Se utilizan para ejecutar una misma actividad **repetidas veces**.
- Las sentencias que se realizan en cada repetición (o iteración) son las mismas, y lo que puede variar en cada iteración son los datos.
- La repetición de un bloque de sentencias un determinado número de veces también se denomina **bucle**.
- Estructuras generales de iteración según control por:
  - **Contador**: número **definido** de veces
    - **PARA** cada valor del *contador* **DESDE** *inicio* **HASTA** *fin* **CADA** *incremento* ejecutar bloque de instrucciones
  - **Expresión lógica**: número **indeterminado** de veces.
    - **MIENTRAS** se cumple la *expresión lógica* ejecutar bloque de instrucciones
    - Ejecutar
      bloque de instrucciones
      HASTA que llegue a cumplirse la *expresión lógica*

Vicente Morell vicente.morell@ua.es

Sentencias de repetición for

- **Sintaxis**: La estructura es:

```
for contador = vector
    secuencia de instrucciones
end
```

- **Semántica**:
  - **Contador**: variable de control de bucle.
  - **Vector**: secuencia de valores que tomará el contador.
    - Puede ser cualquier **vector** o **expresión** que genere un vector
    - Generalmente tiene las formas
      - ***inicio : fin***
      - ***inicio : incremento : fin***
    - Si es una matriz, se considera un vector de columnas, es decir, se asigna al contador cada una de las columnas

Vicente Morell vicente.morell@ua.es

Sentencias de repetición for

```
for contador = vector
    secuencia de instrucciones
end
```

- Funcionamiento:
  1. Al entrar en el bucle se evalúa la expresión que define el **vector**.
  2. Si el **vector** es vacío, se pasa al punto 7.
  3. Se asigna al **contador** el primer elemento del **vector**.
  4. Se ejecuta la **secuencia de instrucciones**.
  5. Si no quedan elementos en el **vector**, se pasa al punto 7.
  6. Se asigna a **contador** el siguiente elemento del **vector**.
  7. Se vuelve al punto 4.
  8. Se ejecuta la instrucción que sigue al **end**.

Vicente Morell vicente.morell@ua.es

Sentencias de repetición for

```
% Suma de los primeros n naturales
n = input('Introduzca N: ');
suma = 0;
for numero = 1:n
    suma = suma + numero;
end
disp('La suma es:'); disp(suma);
```

- Prevención de errores comunes con los bucles for:
  - Es preciso asegurarse de que las variables de las expresiones que definen el **vector** están **inicializadas**.
  - No ha de olvidarse delimitar el cuerpo del bucle con **end**.
  - Es muy conveniente utilizar el **sangrado** para reconocer fácilmente el conjunto de instrucciones que pertenecen al bucle.
  - Los valores del **vector** que se asignarán al contador son los que se establecen al entrar en el **for**. Si el valor del vector se modifica dentro del bucle, esto no tendrá ningún efecto en los sucesivos valores asignados al **contador**.

Vicente Morell vicente.morell@ua.es

Sentencias de repetición while

- **Sintaxis**: La estructura es:

```
while ExprLog
    secuencia de instrucciones
end
```

- **Semántica**:
  - ***ExprLog***: es una expresión de tipo lógico. El bucle se ejecuta **mientras** que ***ExprLog*** tiene valor verdadero (distinto de 0).
  - Debe existir alguna instrucción en la ***secuencia de instrucciones*** que modifique el valor de ***ExprLog***
  - La ***secuencia de instrucciones*** podría no llegar a ejecutarse
  - Si ***ExprLog*** es una matriz, la condición se considerará verdadera si lo son todos los elementos de la matriz.
  - Si ***ExprLog*** es una matriz vacía, se interpreta como falsa.

Vicente Morell vicente.morell@ua.es

Sentencias de repetición while

- Funcionamiento:
  1. Se evalúa ***ExprLog***
  2. Si es falsa, continúa la ejecución en la instrucción siguiente al **end**.
  3. Si es verdadera, se ejecuta la **secuencia de instrucciones**
  4. Se vuelve al punto 1.

```
while ExprLog
    secuencia de instrucciones
end
```

```
% Dos datos positivos
a = input('Introduzca un vector con dos datos positivos:');
while length(a) ~= 2 || any(a <= 0)
   a = input('Asegurese de que son dos datos positivos:');
end
% Ahora se procesan los datos % ...
```

- Prevención de errores comunes con los bucles while:
  - Es preciso asegurarse de que las variables de las expresiones que definen el **vector** están **inicializadas antes** de evaluarlas por primera vez.
  - **Delimitar** siempre el cuerpo del bucle con **while ... end**
  - Utilizar el **sangrado** para reconocer fácilmente el conjunto de instrucciones que pertenecen al bucle
  - En la secuencia de instrucciones debe **modificarse** alguna variable que forme parte de **ExprLog** para que alguna vez se evalúe a verdadero, y evitar la ejecución infinita del bucle.

Vicente Morell vicente.morell@ua.es

P0

Imagen y Vídeo por Computador

• • •

Práctica 0: Matlab

Vicente Morell
vicente.morell@ua.es

Ingeniería Multimedia

Universitat d'Alacant
Universidad de Alicante

HUMAN ROBOTICS