
Introduction to MATLAB

1.1 What is MATLAB?

1.2 Matlab Desktop

1.3 Manipulating Data in Matlab

What is MATLAB?

What is MATLAB?

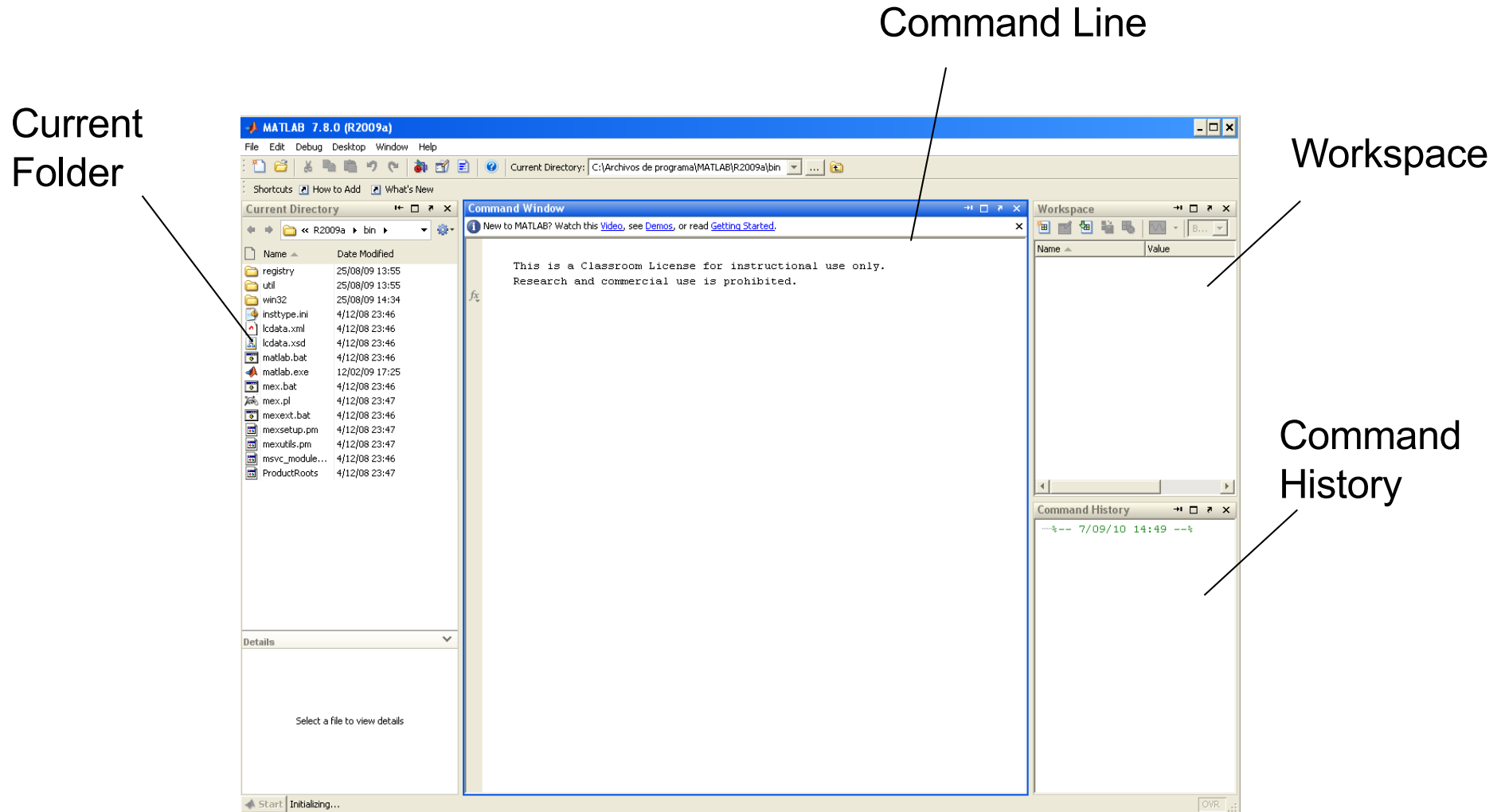
- Matlab comes from “MATrix LABoratory”
- Programming Environment
 - Technical and Scientific Computation
 - Matrix and Vector Oriented
 - Visualization
 - High-level language
 - Toolboxes

What is MATLAB?

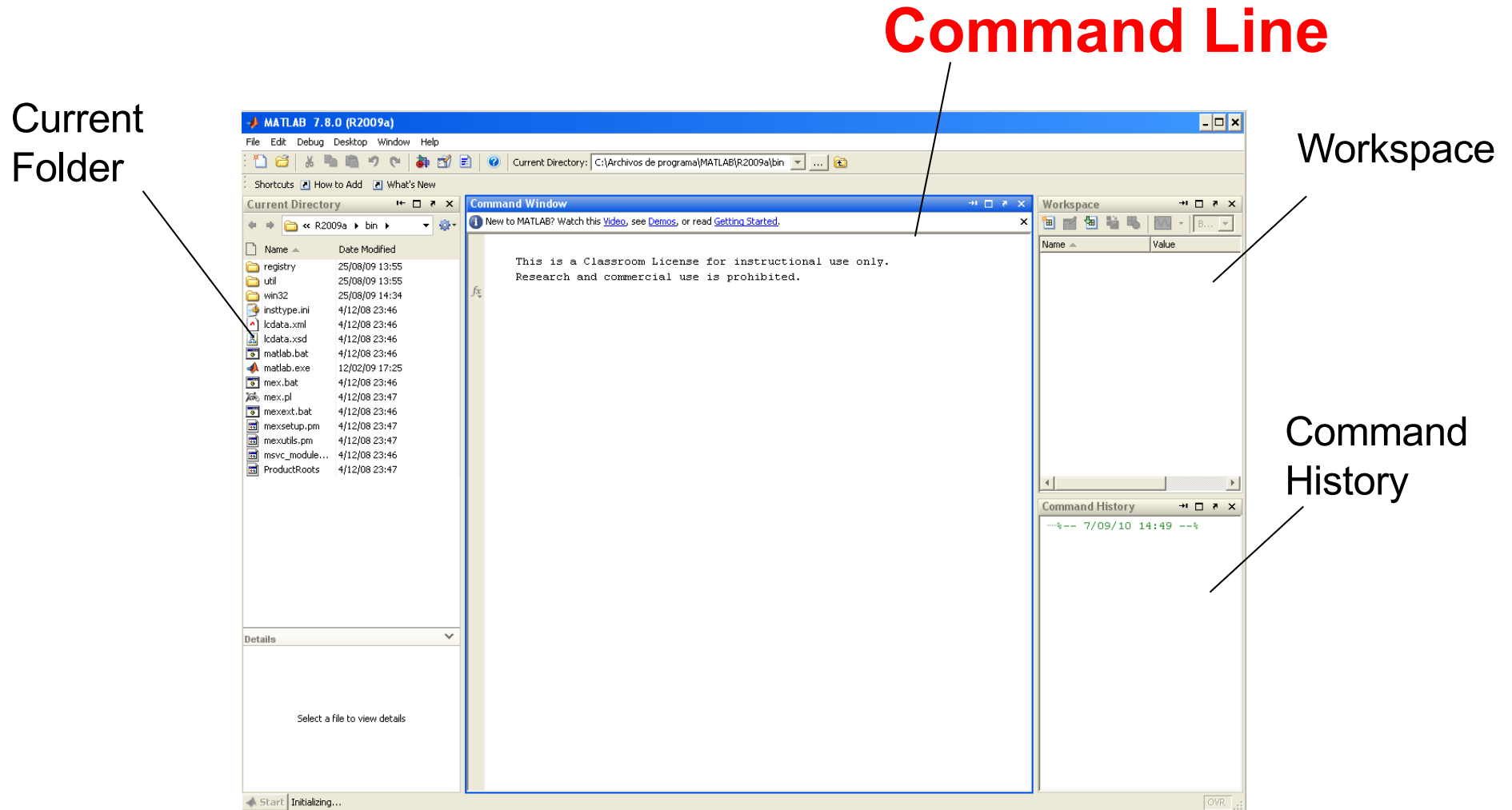
- Some toolboxes:
 - ❑ Optimization Toolbox
 - ❑ Image Processing Toolbox
 - ❑ Neural Network Toolbox
 - ❑ Non Linear Control Design Toolbox
 - ❑ Aerospace Toolbox
 - ❑ Bioinformatics Toolbox
 - ❑ more...

MATLAB Desktop

Desktop: Default Layout



Desktop: Default Layout



Desktop: Command Line Window

The command line window allows users to provide instructions to Matlab about “what to do”.

```
This is a Classroom License for instructional use only.  
Research and commercial use is prohibited.  
>>
```

- The >> symbol is called the **prompt**.
- To **enter** an instruction simply type it after the prompt and press return.

Desktop: Command Line Window

- You can use MATLAB as a calculator:

```
>> 4 + 4
```

```
ans = 8
```

- You can use predefined Math functions

- ▣ abs, acos, angle, cos, asin, complex, floor, fix, mod,...

```
>> cos(0)
```

```
ans = 1
```

- You can visualize data:

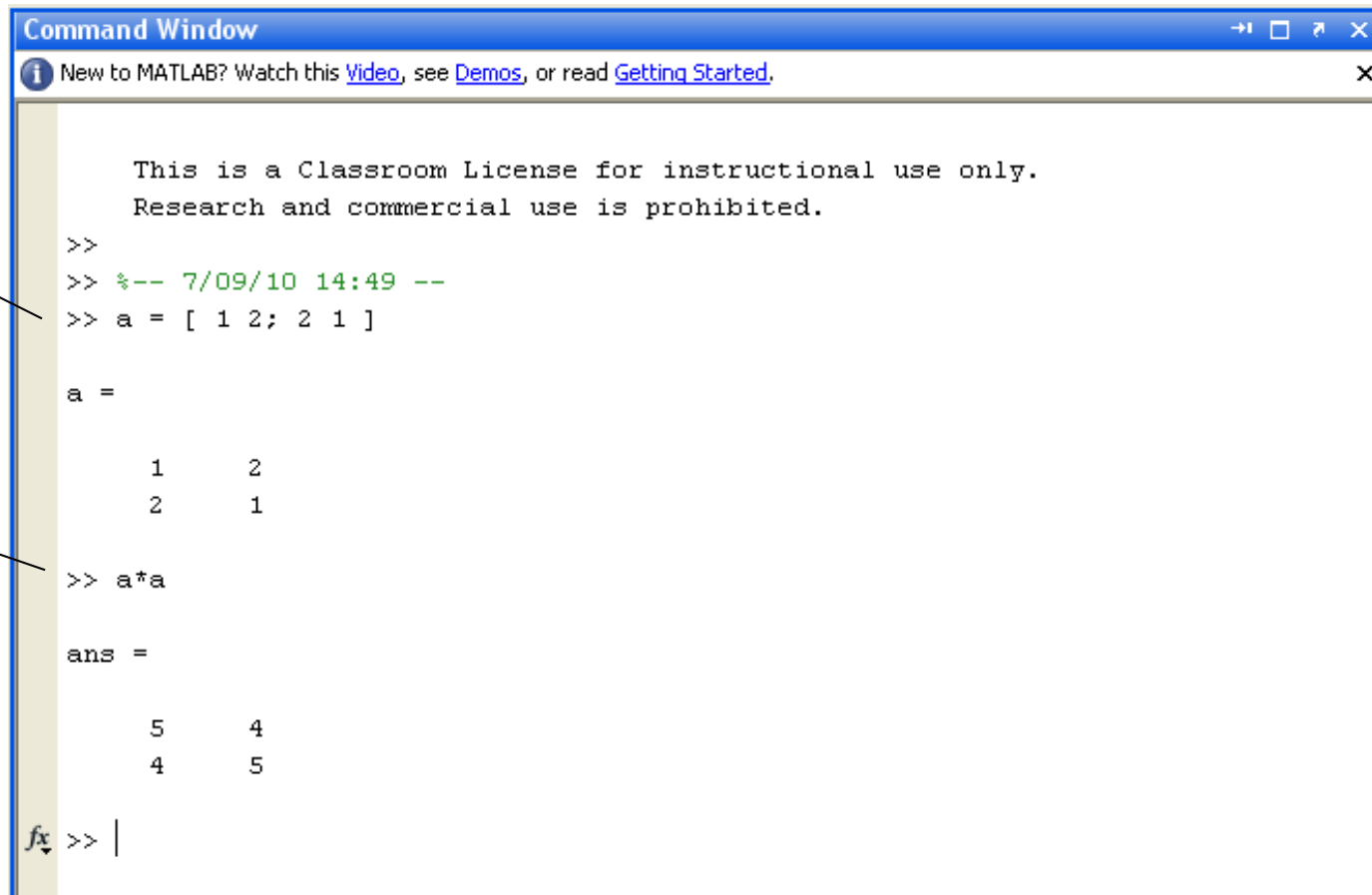
```
>> plot ([1:10])
```

- Other uses: read data from a file, write data, import data, create matrices...

Desktop: Command Line Window

Introduce
data

Introduce
commands



```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

This is a Classroom License for instructional use only.
Research and commercial use is prohibited.

>>
>> %-- 7/09/10 14:49 --
>> a = [ 1 2; 2 1 ]

a =

     1     2
     2     1

>> a*a

ans =

     5     4
     4     5

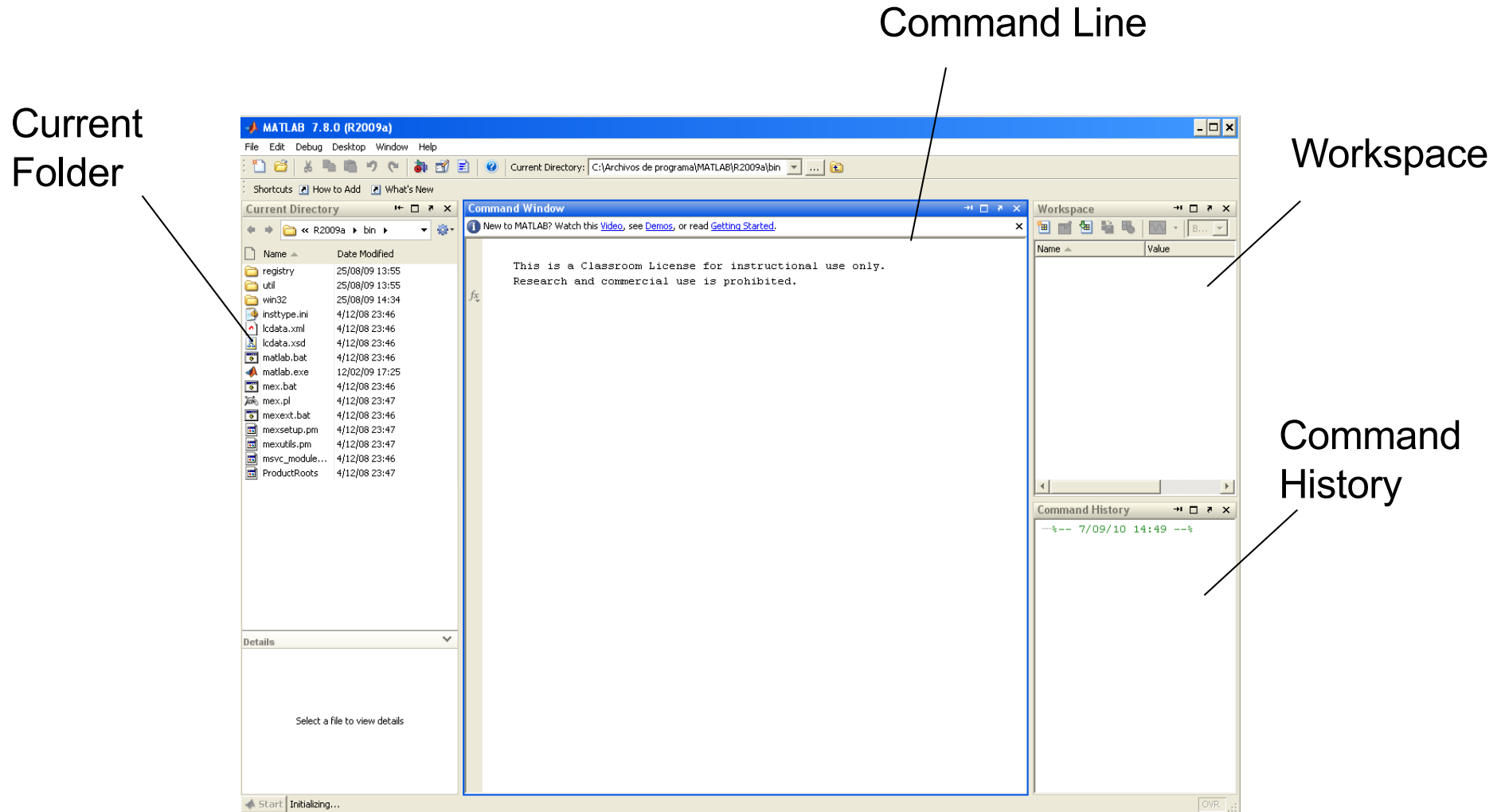
fx >> |
```

Command Window: General Purpose Commands

■ Commands

- ❑ `clc` Clean screen
- ❑ `clear` Remove items from the workspace
- ❑ `help` Display help for MATLAB functions
- ❑ `lasterr` Last error message
- ❑ `lastwarn` Last warning message
- ❑ `ver` Version
- ❑ `path` Control MATLAB's directory search path
- ❑ `addpath` Add directories to MATLAB's search path
- ❑ `..`

Desktop: Default Layout

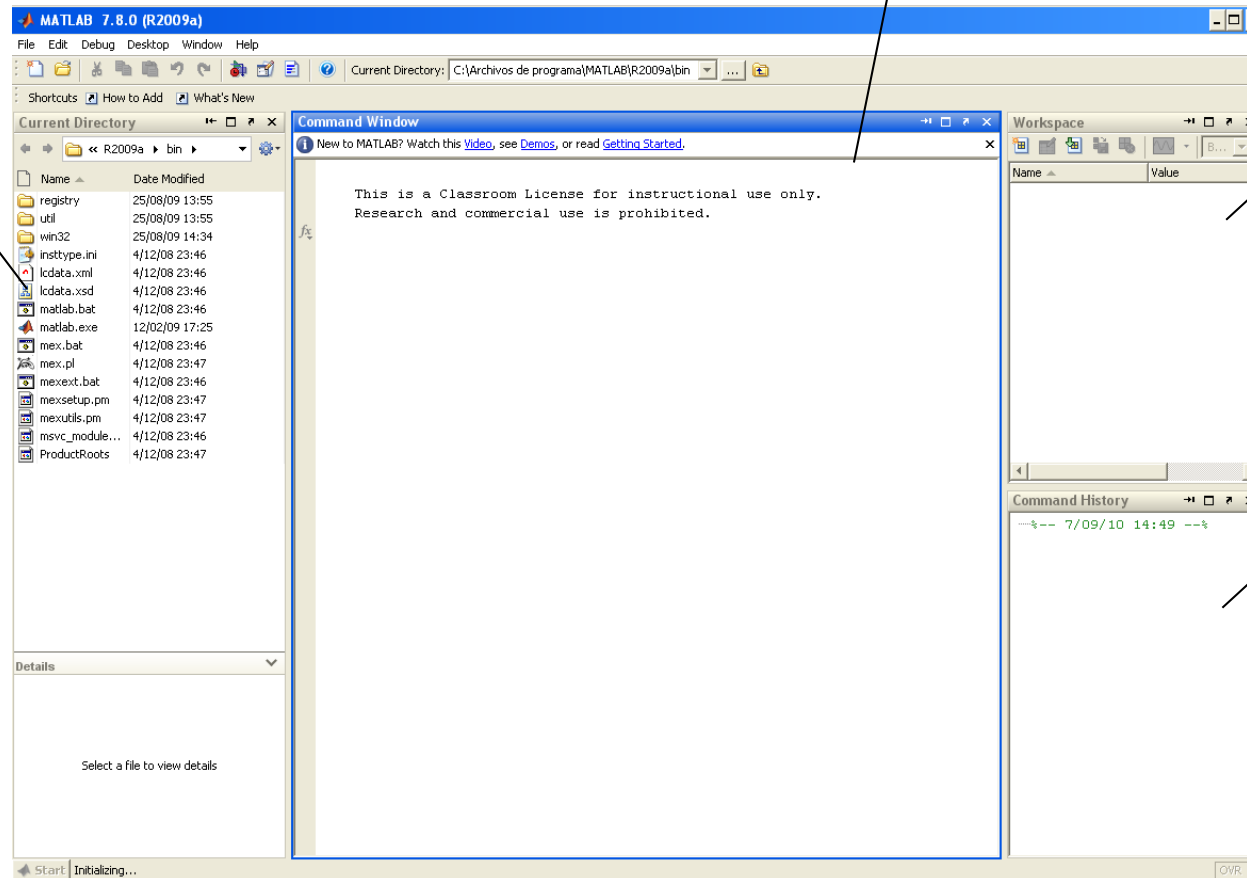


Desktop: Default Layout

Current Folder

Command Line

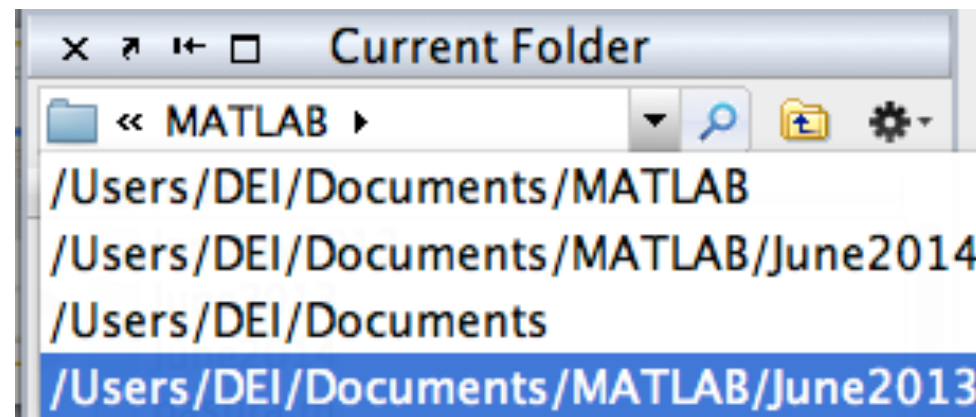
Workspace



Command History

Current Folder Window

- The current folder window shows the files in the current folder right now
- You can navigate through your folders the same way as you do it using the 'explorer' (windows) or 'finder' (mac)

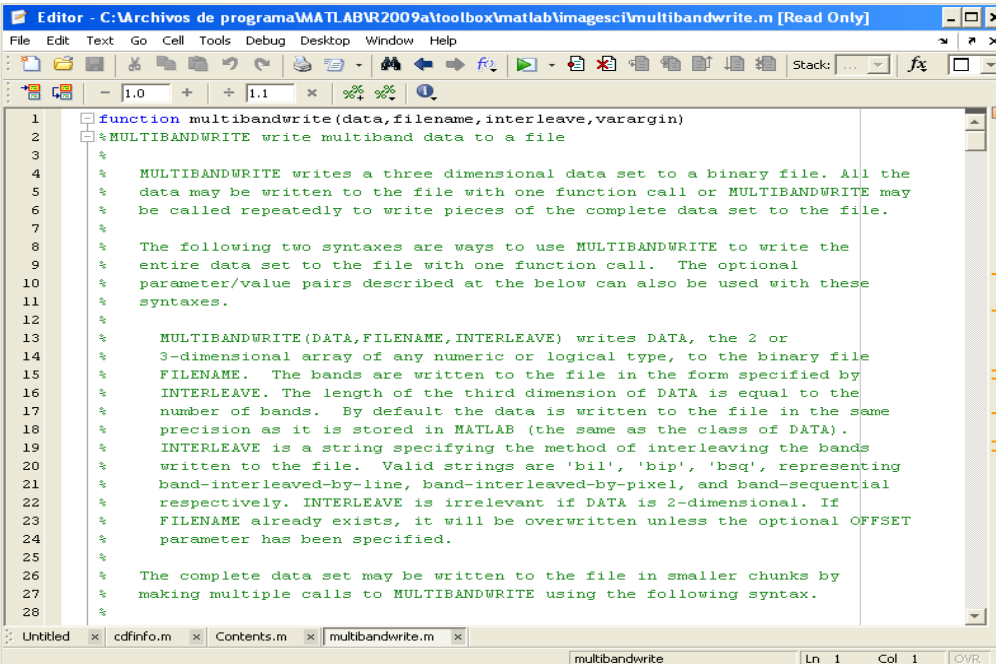


Current Folder Window

- It is important to get used to move around your hard drive using the current folder window
- **You should always know where your program files are, and which your current folder is**
 - It is a common mistake for beginners to not be able to find their programs in the computer.

Desktop: Editor

- Provides a graphical user interface for editing programs (M-files) and for debugging their execution




```
1 function multibandwrite(data,filename,interleave,varargin)
2 %MULTIBANDWRITE write multiband data to a file
3
4 % MULTIBANDWRITE writes a three dimensional data set to a binary file. All the
5 % data may be written to the file with one function call or MULTIBANDWRITE may
6 % be called repeatedly to write pieces of the complete data set to the file.
7
8 % The following two syntaxes are ways to use MULTIBANDWRITE to write the
9 % entire data set to the file with one function call. The optional
10 % parameter/value pairs described at the below can also be used with these
11 % syntaxes.
12
13 % MULTIBANDWRITE(DATA,FILENAME,INTERLEAVE) writes DATA, the 2 or
14 % 3-dimensional array of any numeric or logical type, to the binary file
15 % FILENAME. The bands are written to the file in the form specified by
16 % INTERLEAVE. The length of the third dimension of DATA is equal to the
17 % number of bands. By default the data is written to the file in the same
18 % precision as it is stored in MATLAB (the same as the class of DATA).
19 % INTERLEAVE is a string specifying the method of interleaving the bands
20 % written to the file. Valid strings are 'bil', 'bip', 'bsq', representing
21 % band-interleaved-by-line, band-interleaved-by-pixel, and band-sequential
22 % respectively. INTERLEAVE is irrelevant if DATA is 2-dimensional. If
23 % FILENAME already exists, it will be overwritten unless the optional OFFSET
24 % parameter has been specified.
25
26 % The complete data set may be written to the file in smaller chunks by
27 % making multiple calls to MULTIBANDWRITE using the following syntax.
28
```

Desktop: Editor

- Users can wrap up a sequence of commands and save them into a file for later execution
- Those command files are called **M-files** as they use the extension *.m*
- **M-files** are simply text files. They can be written and read using standard text editors (Microsoft Word is not valid) or MATLAB specific editor

Desktop: Editor

- You can open the editor
 - Typing in the command line the command
edit programname.m
 - Using the menu *File -> New -> Script*
 - Click on the new script icon on the toolbar 
 - Click on the name of a program in the *Current Folder Window*

Desktop: Editor

■ Program example

- Type the following code in your MATLAB editor

```
disp('Hello!');  
age = input ('How old are you? ');  
maxBirthYear = 2020 - age;  
minBirthYear = 2020 - age - 1;  
fprintf('\n Then you were born in %d or in %d', maxBirthYear,  
minBirthYear);
```

- Save the program with the name *myProgram.m*
 - By default Matlab will save the program with extension *.m*

Desktop: Editor

- To execute the program

- From the Editor window: click on the symbol



- From the command window: type

run myProgram

**MATLAB will look for your program in your
'current folder'**

Desktop: Editor

- To execute the program

- From the Editor window: click on the symbol



- From the command window: type

run myProgram

- From the current directory window: drag the name of the program from the current directory window and drop it in the command window

Desktop: Editor

- Result of the execution

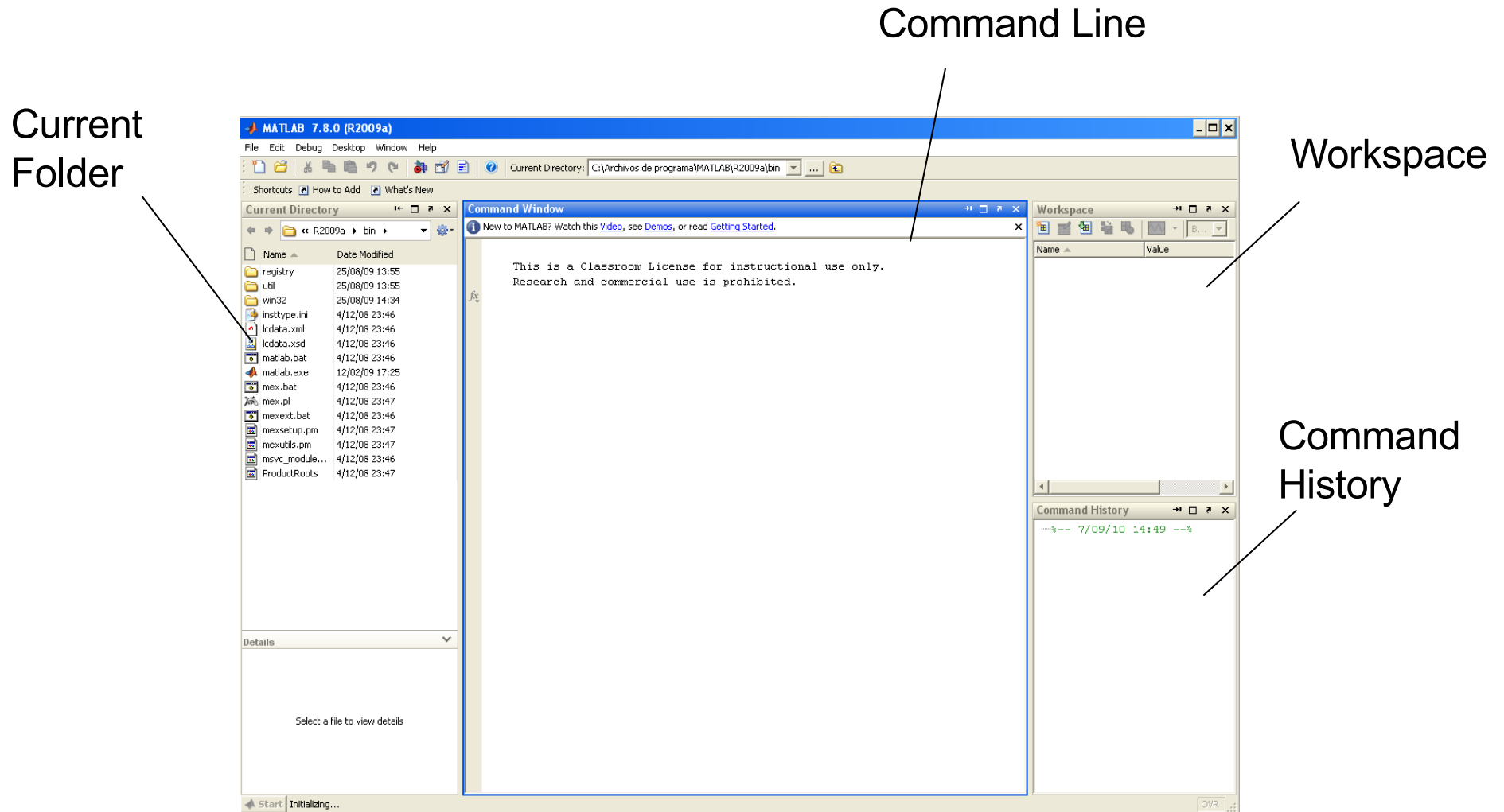
```
>> run myProgram
```

```
    Hello!
```

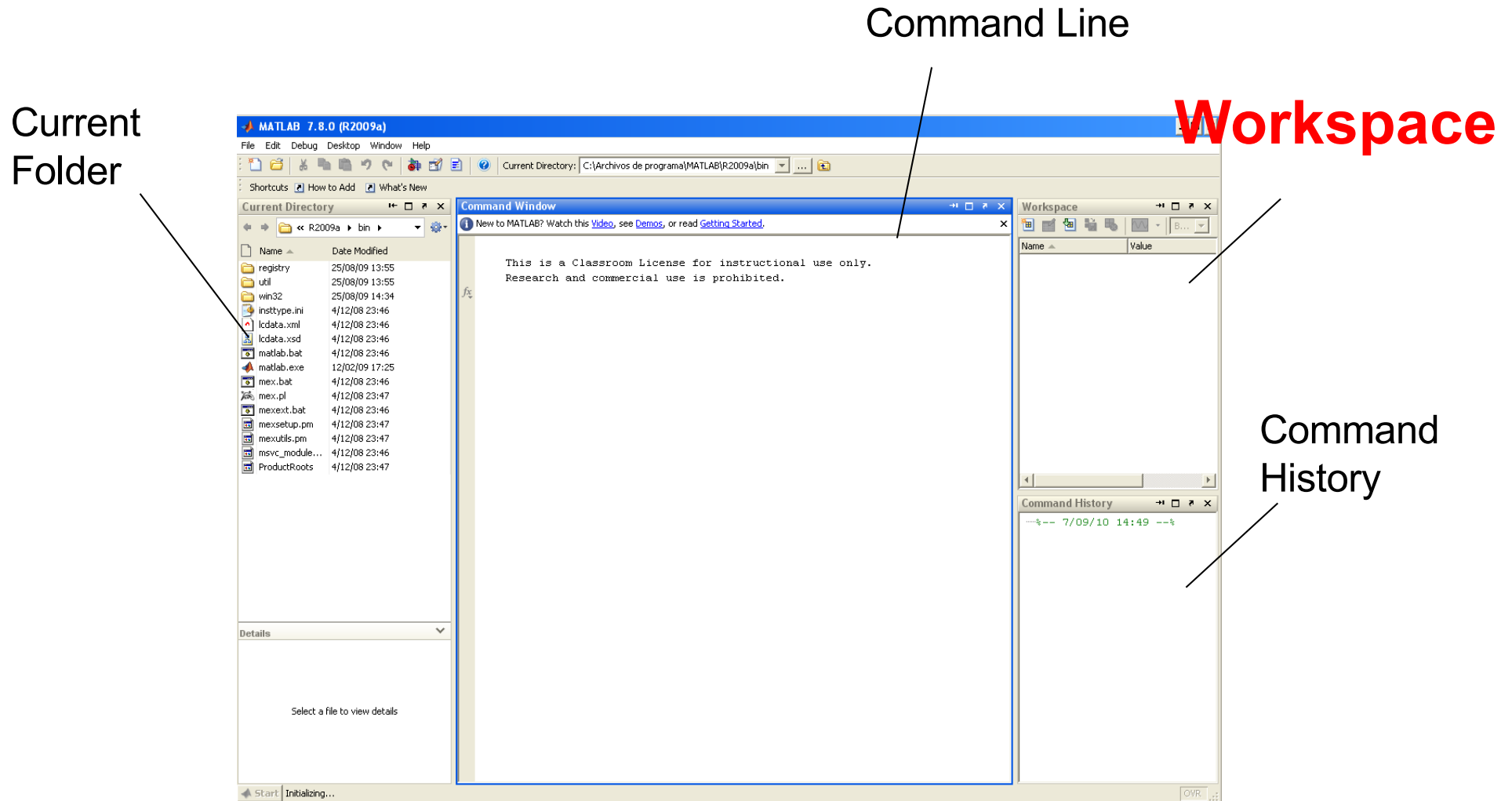
```
    How old are you? 18
```

```
    Then you were born in 2001 or in 2000
```

Desktop: Default Layout



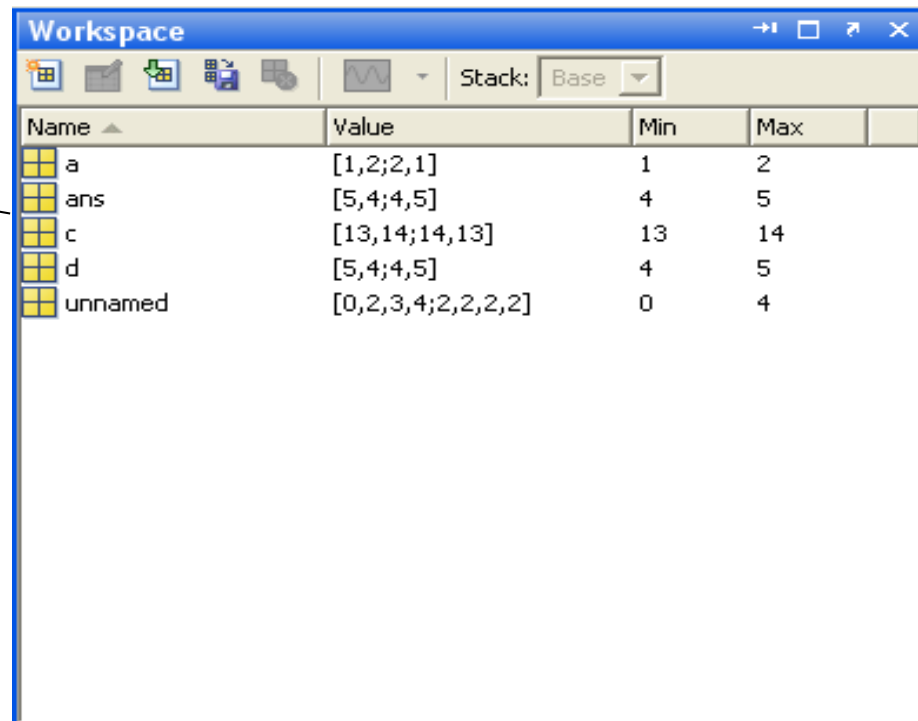
Desktop: Default Layout



Desktop: Workspace Browser

- The Workspace Browser provide access to the variables built up during a Matlab session.

Double click on the variable name to change its data



The screenshot shows the Matlab Workspace Browser window. It has a title bar 'Workspace' and a toolbar with icons for workspace, command window, and other tools. Below the toolbar is a 'Stack' dropdown menu set to 'Base'. The main area contains a table with columns 'Name', 'Value', 'Min', and 'Max'. The table lists five variables: 'a', 'ans', 'c', 'd', and 'unnamed'. Each variable has a small grid icon to its left. A line from the text 'Double click on the variable name to change its data' points to the 'a' variable in the table.

Name	Value	Min	Max
a	[1,2;2,1]	1	2
ans	[5,4;4,5]	4	5
c	[13,14;14,13]	13	14
d	[5,4;4,5]	4	5
unnamed	[0,2,3,4;2,2,2,2]	0	4

Desktop: Workspace Browser

■ Examples

```
>> 4 + 4
```

```
ans = 8
```



The variable 'ans' is stored in the workspace with the value 8

```
>> 10 * 2
```

```
ans = 20
```



Now the value of the variable 'ans' is updated to 20

```
>> a = 20 * 5
```

```
a = 100
```



This time the result of the operation will be stored in the workspace in a variable called 'a'

```
>> b = a * 100
```

```
b = 2000
```



You can create as many variables in the workspace as you want

Desktop: Workspace Browser




- Note that...
 - Variables defined in a M file are added to the workspace, in the same way as when they are created from the command window

Manipulating data in Matlab: **Matrices**

Define and Manipulate Data

- Everything in Matlab is a Matrix

- To create...

- a Vector  `A = [1 2 3]`
- a Matrix  `B = [1 2 3; 4 5 6; 7 8 9]`
- a Matrix  `B = [1 2 3
4 5 6
7 8 9]`

Semicolons and newlines separate **rows**

Spaces and commas separate **columns**

Define and Manipulate Data

- Everything in Matlab is a Matrix

A

1	2	3
---	---	---

B

1	2	3
4	5	6

- To access...

- an element of a vector: $vector_name(position)$
- an element of a matrix: $matrix_name(row, column)$
- a entire row of a matrix: $matrix_name(row, :)$
- a entire column of a matrix: $matrix_name(:, column)$
- a part of the matrix $B(row_ini:row_end, col_ini:col_end)$

```
>>A(2)
```

```
ans = 2
```

The element in the 2nd position of the vector

```
>> B(2,2)
```

```
ans = 5
```

The element in the second row and second column of the matrix

```
>>B(1,:)
```

```
ans = [ 1 2 3]
```

The first row

```
>>B(1:2,2:3)
```

```
ans = [2 3; 5 6]
```

The range of values between the first row and second column and the element in the second row and secon column

Define and Manipulate Data

- Everything in Matlab is a Matrix

A

1	2	3
---	---	---

B

1	2	3
4	5	6

- To access...

- an element of a vector: *vector_name(position)*
- an element of a matrix: *matrix_name(row, column)*
- a entire row of a matrix: *matrix_name(row, :)*
- a entire column of a matrix: *matrix_name(:, column)*
- a part of the matrix B(*row_ini:row_end* , *col_ini:col_end*)

What happen when you try to access a position out of the range of the Vector/Matrix?

Define and Manipulate Data

- Everything in Matlab is a Matrix

- To access...

- an element of a vector: *vector_name(position)*
 - an element of a matrix: *matrix_name(row, column)*
 - a entire row of a matrix: *matrix_name(row, :)*
 - a entire column of a matrix: *matrix_name(:, column)*
 - a part of the matrix B(*row_ini:row_end* , *col_ini:col_end*)

A

1	2	3
---	---	---

B

1	2	3
4	5	6

What happen when you try to access a position out of the range of the Vector/Matrix?

Error: Index exceeds matrix dimensions.

Define and Manipulate Data

- Everything in Matlab is a Matrix

A			B
	1	2	3
	4	5	6

- To modify ...

- an element of a vector $vector_name(position) = \underline{new\ value}$
- an element of a matrix $matrix_name(row, column) = \underline{new\ value}$

```
>> A = [1 2 3]
```

```
A = 1 2 3
```

```
>> A(2) = 4
```

```
A = 1 4 3
```

```
>> B = [1 2 3; 4 5 6]
```

```
B = 1 2 3
```

```
4 5 6
```

```
>> B(2,2) = 6
```

```
B = 1 2 3
```

```
4 6 6
```

Define and Manipulate Data

- Everything in Matlab is a Matrix

A				B			
	1	2	3		1	2	3
					4	5	6

- To modify ...

- an element of a vector $vector_name(position) = \underline{new\ value}$
- an element of a matrix $matrix_name(row, column) = \underline{new\ value}$

What happens when you try to assign a value to a position out of the range? Example $B(1,4) = 5$

```
>> B = [1 2 3; 4 5 6]
```

```
B = 1 2 3
```

```
4 5 6
```

```
>> B(2,2) = 6
```

```
B = 1 2 3
```

```
4 6 6
```

Define and Manipulate Data

- Everything in Matlab is a Matrix

A				B			
	1	2	3		1	2	3
					4	5	6

- To modify ...

- an element of a vector $vector_name(position) = \underline{new\ value}$
- an element of a matrix $matrix_name(row, column) = \underline{new\ value}$

What happens when you try to assign a value to a position out of the range? Example $B(1,4) = 5$
Matlab fill the rest of the matrix with zeros

```
>> B = [1 2 3; 4 5 6]
```

```
B = 1 2 3  
    4 5 6
```

```
>> B(2,2) = 6
```

```
B = 1 2 3 5  
    4 6 6 0
```

Define and Manipulate Data

- Everything in Matlab is a Matrix

- To delete an element:

- Matlab deletes an element and reorganizes the matrix/vector
- Delete an element of a vector: $vector_name(position) = []$;
- Delete the row of a matrix: $matrix_name(row, :) = []$;
- Delete the column of a matrix: $matrix_name(:, column) = []$;

- Example:

```
>> A = [1 2 3 4 5 6];
```

```
>> A(3) = [];
```

```
>> A
```

```
A = [1 2 4 5 6]
```

```
>> B = [1 2 3; 4 5 6];
```

```
>> B(:,2) = [];
```

```
>> B
```

```
B = [1 3  
4 6]
```

Define and Manipulate Data

- Everything in Matlab is a Matrix
 - Specials function to define vectors and matrices: *zeros*, *ones*
 - `zeros(number of rows, number of columns)`

- Example:

`A = zeros(2,2)`

`A = [0 0
0 0]`

- `ones(number of rows, number of columns)`
- Example:

`A = ones(1,3)`

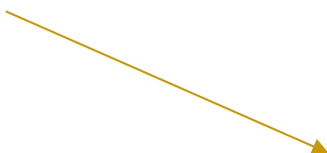
`A = [1 1 1]`

These two functions are very useful to give an initial value to all the elements in a matrix or vector

Define and Manipulate Data

- Everything in Matlab is a Matrix
 - `size (matrix)` returns the size of the matrix
 - Example:

```
size(A)  
ans = 2  2
```



This means that the matrix A has 2 rows and 2 columns

Aritmetic Operators

- Aritmetic operators

- + Matrix Addition
- - Matrix Subtraction
- * Matrix multiplication
- / Matrix right division
- \ Matrix left division
- ^ Matrix power
- ' Matrix transpose

- .* Array multiplication
- ./ Array right division
- .\ Array left division
- .^ Array power
- .' Array transpose

- Matrix arithmetic operations are defined by the rules of linear algebra.

- Array arithmetic operations are carried out element by element

Other Operators

This is a very important operator you are going to use very often in this course

■ :

- It generates a sequence of values
- Usefull to create matrices and vectors
- Use:

initial value : step : final value

OR

initial value : final value

- Examples:

- $A = 4:7$

- $A = [4 \ 5 \ 6 \ 7]$

- $B = 1:2:10$

- $B = [1 \ 3 \ 5 \ 7 \ 9]$

- $C = 4 : 0.1: 4.5$

- $C = [4 \ 4.1 \ 4.2 \ 4.3 \ 4.4 \ 4.5]$

- $D = 4: -1: 0$

- $D = [4 \ 3 \ 2 \ 1 \ 0]$

Bibliography

- “MATLAB: An introduction with Applications”, John Wiley & Sons, Inc., Hoboken (NJ), USA