

# Slide Set 1

## Introduction to MATLAB

Maria Ptashkina    Damian Romero

Barcelona Graduate School of Economics

September 2020

# Outline

## Introduction

- Administrative and general view
- Fundamentals of programming

Interface: Desktop window

# Introduction

## Administrative

### Sessions

- 5 sessions of 2 hours each
- Introduction to main concepts and practice exercises

### Emails

- Maria Ptashkina: [maria.ptashkina@barcelonagse.eu](mailto:maria.ptashkina@barcelonagse.eu)
- Damian Romero: [damian.romero@barcelonagse.eu](mailto:damian.romero@barcelonagse.eu)

# Introduction

What is MATLAB?

- Interactive computational environment originally designed to solve linear algebra equations or systems of equations using matrices

What can we do with it?

- Import/export data; plots
- Matrix operations
- Numerical optimization

Some examples of applications

- Macroeconomics: solve static and dynamic equilibrium models
- Econometrics: estimation and simulation

# Contents

1. Introduction to MATLAB
  - Interface
  - Basic syntax
2. Functions, logical expressions and control structures
3. Data input/output and plots
4. Basic numerical optimization and solvers

# Goals of this course

1. Get familiar with the basic computational tools available in MATLAB and main graphics tools
2. Get the basics of logic programming: loops, if-else clauses
3. Import/export data and perform data analysis
4. Build your own functions

# Before We Start

## Fundamentals of Programming

For our purpose, we can talk about two basic “programming paradigmes” (simply, styles of programming)

- Procedural programming: writing a list of instructions to tell the computer what to do step by step
- Functional programming: writing a mathematical function, i.e. a function that takes arguments and returns a value

Silly example (not in any language):

### Functional

```
num = 1
def function_to_add_one(num):
    num += 1
    return num
```

```
function_to_add_one(num)
function_to_add_one(num)
function_to_add_one(num)
function_to_add_one(num)
function_to_add_one(num)
```

```
#Final Output: 2
```

### Procedural

```
num = 1
def procedure_to_add_one():
    global num
    num += 1
    return num
```

```
procedure_to_add_one()
procedure_to_add_one()
procedure_to_add_one()
procedure_to_add_one()
procedure_to_add_one()
```

```
#Final Output: 6
```

Most of the things you will be doing in MATLAB are procedural

# Other Languages

In Economics, many general-purpose languages are used, with the most common being:

- MATLAB
- R
- Julia
- Python

Which one to use?

- There is no definite answer: depending on the project, any of the four could be the best choice
- Generally, MATLAB and Julia are best for numerical problems and simulations, while R and Python are great at data handling
- If curious, read [this comparison](#) of the four languages

You already know Stata...

- ...but Stata is not a language, but a statistical software, so the programming logic in this course might be different



# Some Language-Agnostic Tips for Coding

Programming is more than writing code: structuring, testing, documenting and collaborating on code are central!

Main principles:

- Make your code as easy as possible
- Automate and abstract to make your code reproducible (for your future self)

**Active learning:** to learn programming (in any language) you need to work on actual problems yourself!

[This guide](#) for Social Science research might be handy at some point in time

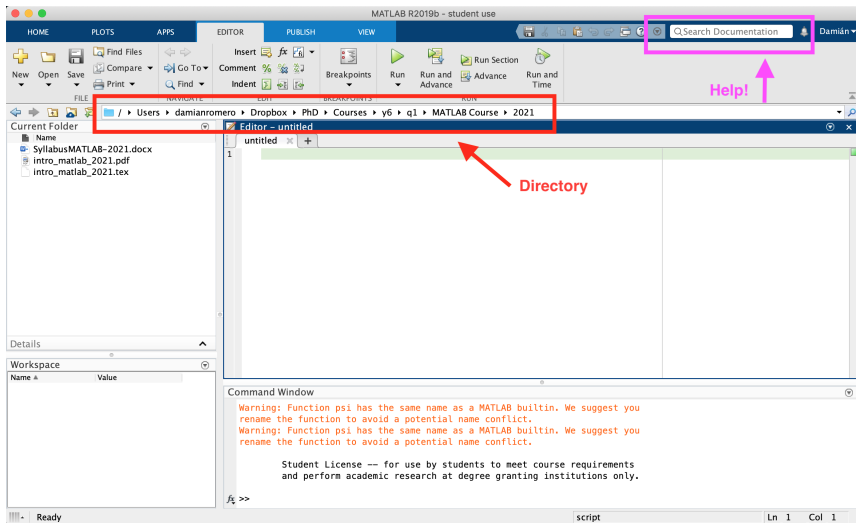
# Outline

## Introduction

- Administrative and general view
- Fundamentals of programming

## Interface: Desktop window

# MATLAB interface

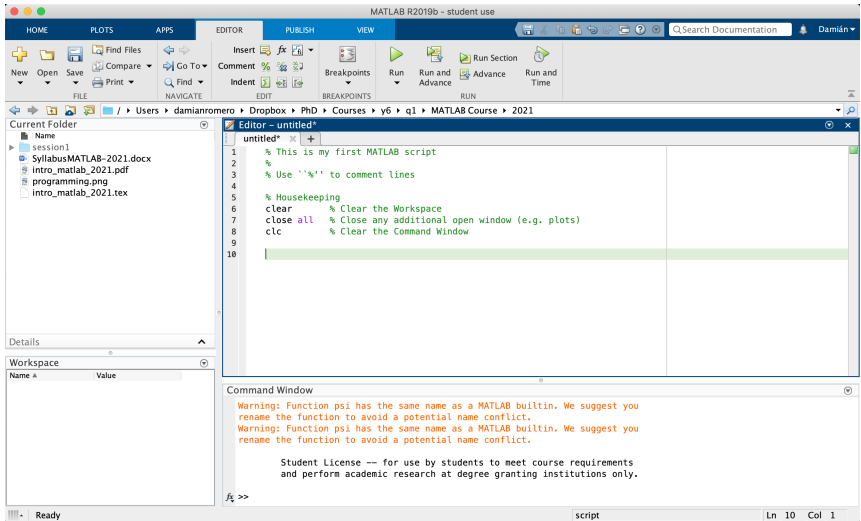


# MATLAB interface

- **Command window:** the window where to execute the commands and get basic help
- **Workspace:** contains the “objects” (e.g. variables, whole datasets etc. you are currently working with)
- **Editor:** is the environment in which you can generate Script files. A Script a file with extension “.m” in which you can save sequence of commands for future use (we will cover this extensively later on)
- **Current Folder:** The folder you are working on and on which MATLAB looks for user-created functions, MATLAB datasets, other files (e.g. Excel, txt etc..)
- **Directory:** The path of the current folder
- **Help:** provides support and documentation on different MATLAB functions

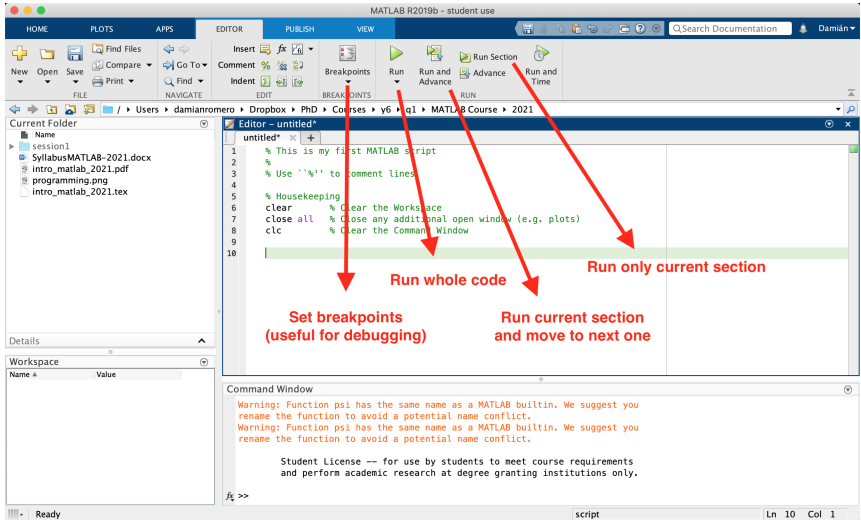
# MATLAB interface

## Editor



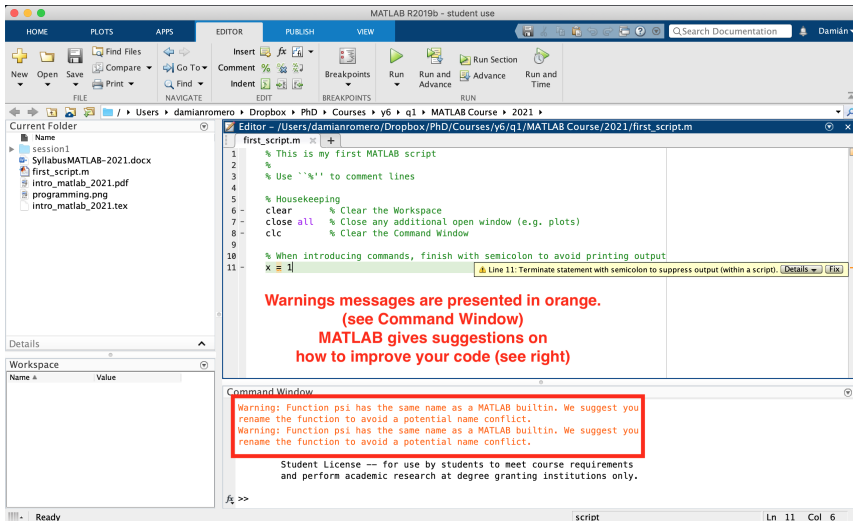
# MATLAB interface

## Editor



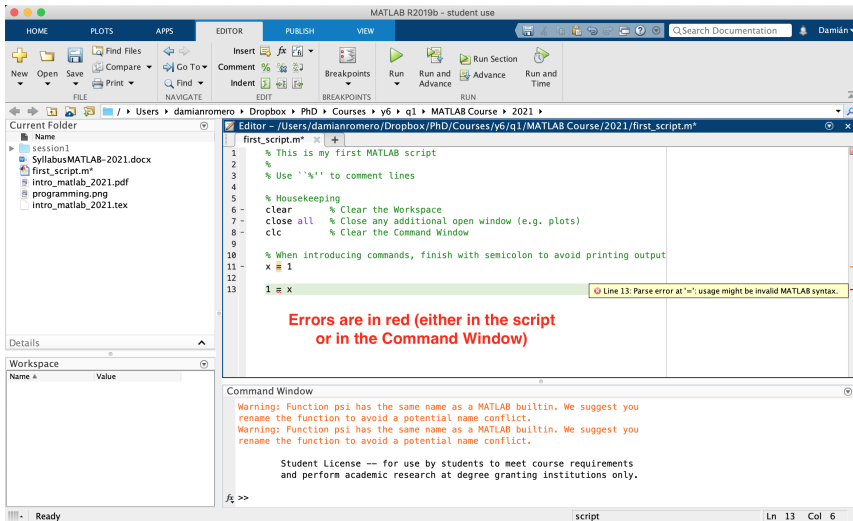
# MATLAB interface

## Editor



# MATLAB interface

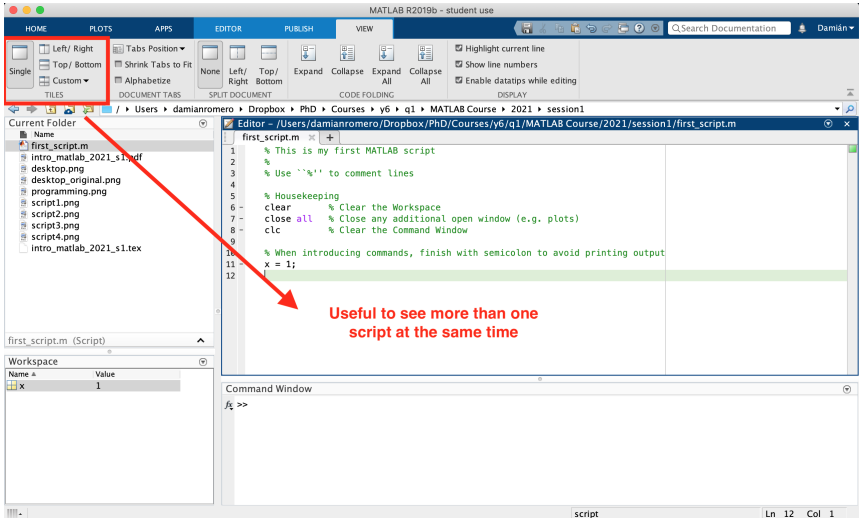
## Editor





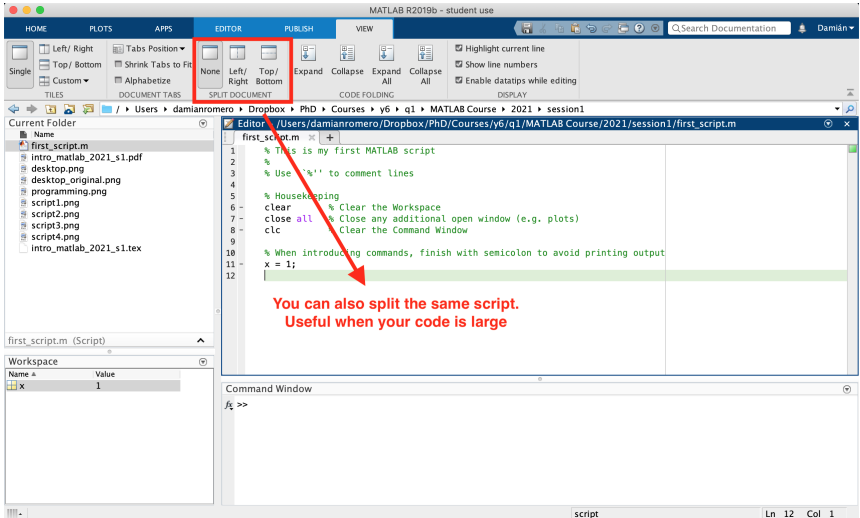
# MATLAB interface

## View



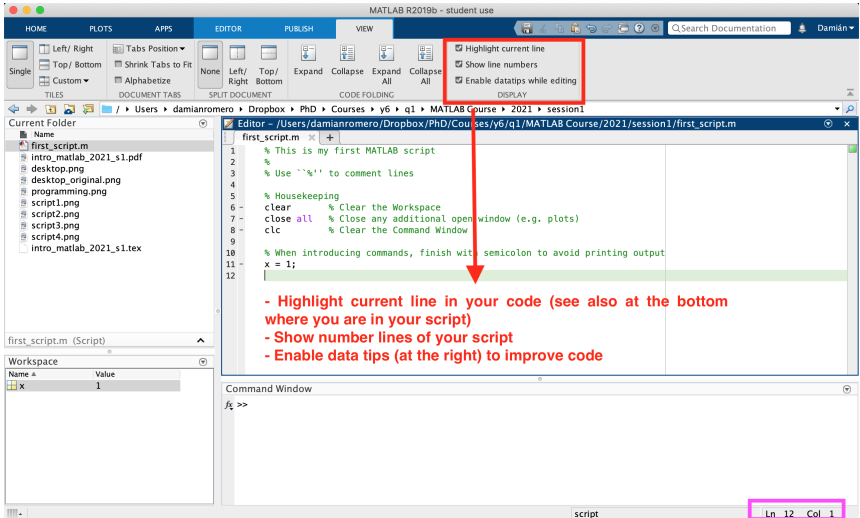
# MATLAB interface

## View



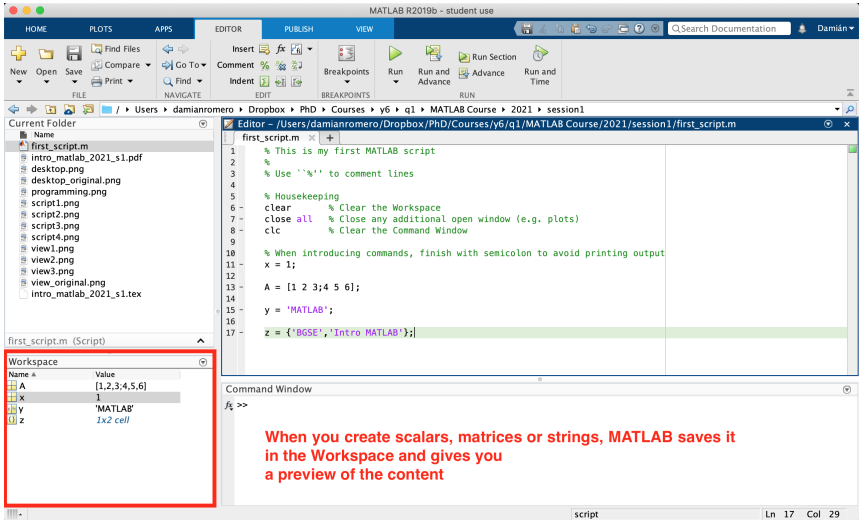
# MATLAB interface

## View



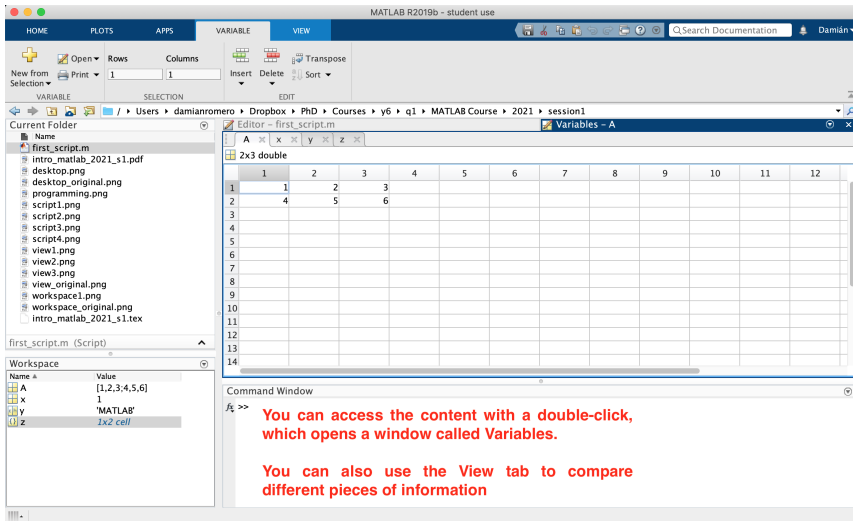
# MATLAB interface

## Workspace and Variables



# MATLAB interface

## Workspace and Variables



The image shows the MATLAB R2019b interface with the following components:

- Top Bar:** MATLAB R2019b - student use, Search Documentation, and user name Damían.
- Toolbox Bar:** HOME, PLOTS, APPS, VARIABLE, and VIEW tabs.
- Editor - first\_script.m:** A 2x3 double matrix is displayed in the Variables - A window.

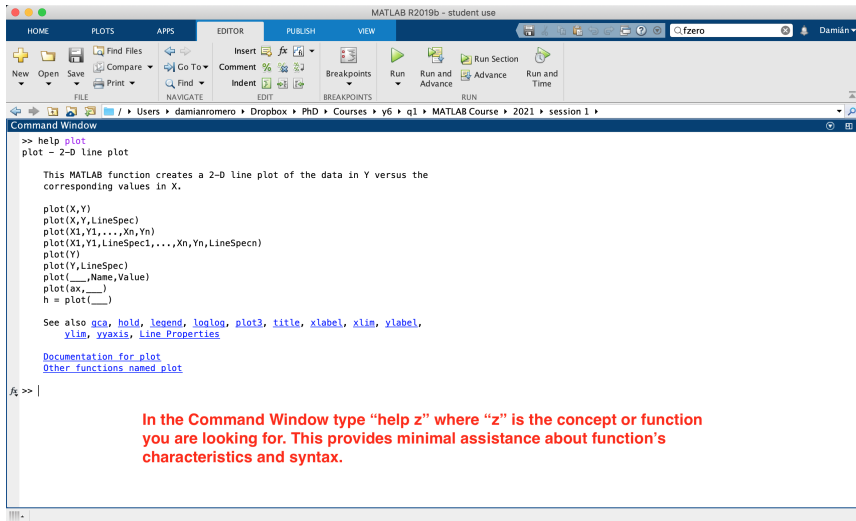
	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3									
2	4	5	6									
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
- Current Folder:** / Users / damianromero / Dropbox / PhD / Courses / y6 / q1 / MATLAB Course / 2021 / session1
- Workspace:** Lists variables: A (1x2 cell), x (1), y ('MATLAB'), and z (1x2 cell).
- Command Window:** Shows the prompt >>.

**You can access the content with a double-click, which opens a window called Variables.**

**You can also use the View tab to compare different pieces of information**

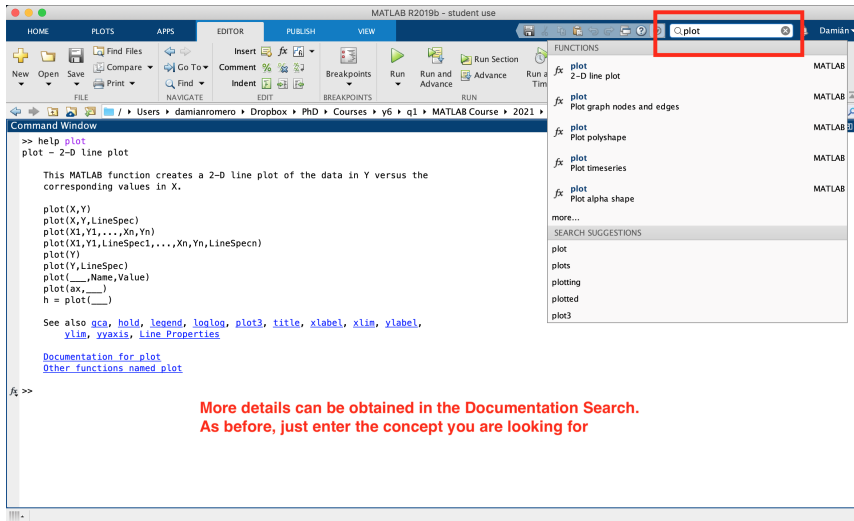
# MATLAB interface

## Getting help



# MATLAB interface

## Getting help



The image shows the MATLAB R2019b interface. The top toolbar includes a search icon and a search bar containing the text "plot", which is highlighted with a red rectangle. The "Command Window" on the left displays the help text for the `plot` function, including its syntax and usage examples. The "FUNCTIONS" pane on the right lists various plotting functions, with `plot` selected. Below the functions list, the "SEARCH SUGGESTIONS" section shows related terms like `plots`, `plotting`, and `plotted`.

Command Window

```
>> help plot
plot - 2-D line plot

This MATLAB function creates a 2-D line plot of the data in Y versus the
corresponding values in X.

plot(X,Y)
plot(X,Y,LineSpec)
plot(X1,Y1,...,Xn,Yn)
plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)
plot(Y)
plot(Y,LineSpec)
plot(__,Name,Value)
plot(ax,__)
h = plot(__)
```

See also [gca](#), [hold](#), [legend](#), [loglog](#), [plot3](#), [title](#), [xlabel](#), [xlim](#), [ylabel](#), [ylim](#), [yyaxis](#), [Line Properties](#).

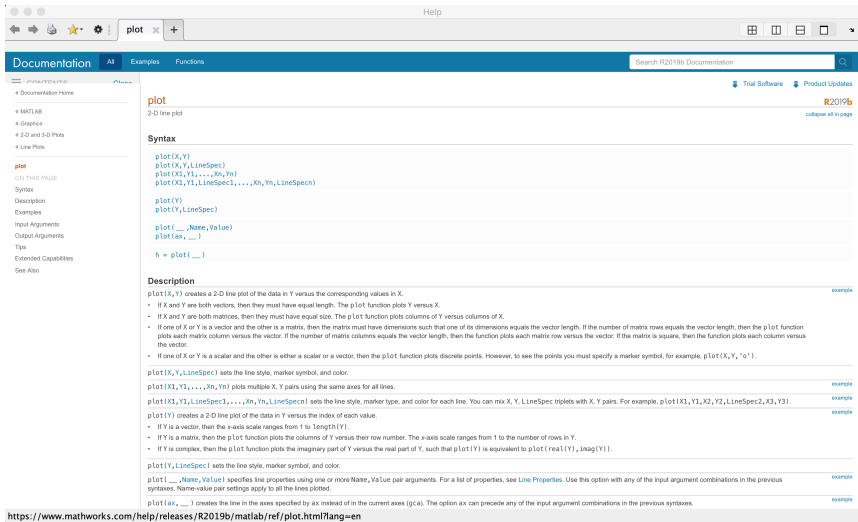
[Documentation for plot](#)  
[Other functions named plot](#)

fx >>

More details can be obtained in the Documentation Search.  
As before, just enter the concept you are looking for

# MATLAB interface

## Getting help



The screenshot shows the MATLAB documentation page for the `plot` function. The browser window has a title bar with "Help" and standard navigation buttons. The page header includes "Documentation", "AI", "Examples", and "Functions" tabs, along with a search bar containing "Search R2019b Documentation". On the left, a navigation pane lists "MATLAB", "Graphics", "2-D and 3-D Plots", and "Line Plots", with "Line Plots" expanded to show "plot". The main content area is titled "plot" and "2-D line plot". It includes a "Syntax" section with the following code snippets:

```
plot(X,Y)
plot(X,Y,LineStyle)
plot(X1,Y1,...,Xn,Yn)
plot(X1,Y1,LineStyle1,...,Xn,Yn,LineStylen)

plot(Y)
plot(Y,LineStyle)

plot(__,Name,Value)
plot(ax, __)

h = plot(__)
```

The "Description" section explains the function's behavior:

- `plot(X,Y)` creates a 2-D line plot of the data in Y versus the corresponding values in X.
- If X and Y are both vectors, then they must have equal length. The `plot` function plots Y versus X.
- If X and Y are both matrices, then they must have equal size. The `plot` function plots columns of Y versus columns of X.
- If one of X or Y is a vector and the other is a matrix, then the matrix must have dimensions such that one of its dimensions equals the vector length. If the number of matrix rows equals the vector length, then the `plot` function plots each matrix column versus the vector. If the number of matrix columns equals the vector length, then the function plots each matrix row versus the vector. If the matrix is square, then the function plots each column versus the vector.
- If one of X or Y is a scalar and the other is either a scalar or a vector, then the `plot` function plots discrete points. However, to see the points you must specify a marker symbol, for example, `plot(X,Y,'o')`.

The "Syntax" section continues with:

- `plot(X,Y,LineStyle)` sets the line style, marker symbol, and color.
- `plot(X1,Y1,...,Xn,Yn)` plots multiple X, Y pairs using the same axes for all lines.
- `plot(X1,Y1,LineStyle1,...,Xn,Yn,LineStylen)` sets the line style, marker type, and color for each line. You can mix X, Y, `LineStyle` triplets with X, Y pairs. For example, `plot(X1,Y1,X2,Y2,LineStyle2,X3,Y3)`.
- `plot(Y)` creates a 2-D line plot of the data in Y versus the index of each value.
- If Y is a vector, then the x-axis scale ranges from 1 to `length(Y)`.
- If Y is a matrix, then the `plot` function plots the columns of Y versus their row number. The x-axis scale ranges from 1 to the number of rows in Y.
- If Y is complex, then the `plot` function plots the imaginary part of Y versus the real part of Y, such that `plot(Y)` is equivalent to `plot(real(Y), imag(Y))`.

The "Syntax" section continues with:

- `plot(Y,LineStyle)` sets the line style, marker symbol, and color.
- `plot(__,Name,Value)` specifies line properties using one or more Name, Value pair arguments. For a list of properties, see [Line Properties](#). Use this option with any of the input argument combinations in the previous syntaxes. Name-value pair settings apply to all the lines plotted.
- `plot(ax, __)` creates the line in the axes specified by `ax` instead of in the current axes (gca). The option `ax` can precede any of the input argument combinations in the previous syntaxes.

At the bottom of the page, the URL <https://www.mathworks.com/help/releases/R2019b/matlab/ref/plot.html?lang=en> is displayed.