



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

Introductory Guide to MATLAB

Master's Degree in Mechanical Engineering for Sustainability

didatticaforli.ingstudenti@unibo.it

Table of Contents

1. Introduction	3
2. Installation Guide.....	3
2.1. Managing Toolboxes	3
3. The MATLAB Interface	5
3.1. Change Desktop Colors	6
3.2. Using the Command Window	6
4. Setting up MATLAB	7
4.1. Managing File Paths	7
4.2. Defining startup Commands and Settings	7
4.3. MATLAB Source Control Integration	8
5. Coding in MATLAB	8
5.1. Help and Documentation	8
5.2. MATLAB Scripts	8
5.3. MATLAB Live Scripts	9
5.4. MATLAB Apps.....	10
5.5. Creating and Managing Functions.....	11
5.6. Debugging in MATLAB.....	11
5.7. Calling Simulink.....	13
5.8. Evaluating Code Performance.....	13
6. Code Examples	15
6.1. Coding Fundamentals.....	15
6.2. Data Importing.....	16
6.3. Data Processing and Applications	16
6.4. Data Export and Visualization	17
6.5. Good to Know	17
7. Where to go from here... ..	17
Additional Resources	18

1. Introduction

MATLAB is a programming and numeric computing environment used by millions of engineers and scientists to analyze data, develop algorithms, and create models. MATLAB provides professionally developed toolboxes for signal and image processing, control systems, wireless communications, computational finance, robotics, deep learning and AI and more. MATLAB combines a desktop environment tuned for iterative analysis and design processes with a high-level programming language that expresses matrix and array mathematics directly. It includes the Live Editor for creating scripts that combine code, output, and formatted text in an executable notebook. Prebuilt apps allow users to interactively perform iterative tasks. Users can then automatically generate the corresponding MATLAB code to reproduce the work and add it to the script with the push of a button. Users can also extend MATLAB with thousands of packages and toolboxes shared on GitHub, MATLAB File Exchange, and elsewhere. And MATLAB code and algorithms can be deployed to run on cloud and enterprise systems, used directly in Simulink, or automatically converted to C/C++, HDL, and CUDA code to run on embedded devices¹. People can get started with MATLAB by walking through an example. This video shows the basics, and it gives an idea of what working in MATLAB is like: [Introduction to MATLAB](#)².

2. Installation Guide

The MATLAB installation is guided and straightforward. The software can be downloaded from [this page](#)³ and the shortest guide to do so is available at this [page](#)⁴. Information for students can be found at the [dedicated webpage](#)⁵. Additionally, a video guide on [How to Install MATLAB](#)⁶ is available. The full [installation guide](#)⁷ in pdf format is also offered and the [documentation](#)⁸ provides further guiding on installing.

2.1. Managing Toolboxes

MATLAB offers various add-ons and toolboxes to extend its capabilities. These include apps, libraries, functions, examples, and more. Most of the toolboxes are available for free with a student license and provide specialized libraries for specific tasks or fields, like signal

¹ <https://www.mathworks.com/videos/matlab-overview-61923.html>

² <https://www.youtube.com/watch?v=OHxR8iMHDWw>

³ <https://www.mathworks.com/academia/tah-portal/alma-mater-studiorum-universita-di-bologna-1122528.html>

⁴ <https://www.mathworks.com/help/install/ug/install-products-with-internet-connection.html>

⁵ <https://www.unibo.it/en/study/life-at-university-and-in-the-city/discounts-for-computer-tablet-and-software-1/matlab/matlab-1>

⁶ https://www.youtube.com/watch?v=f1UoHTf_Kgk

⁷ https://www.mathworks.com/help/pdf_doc/install/install_guide.pdf

⁸ <https://www.mathworks.com/help/install/install-products.html>

processing, artificial intelligence, automotive, or aerospace. The complete list of toolboxes is available at this [page](#)⁹.

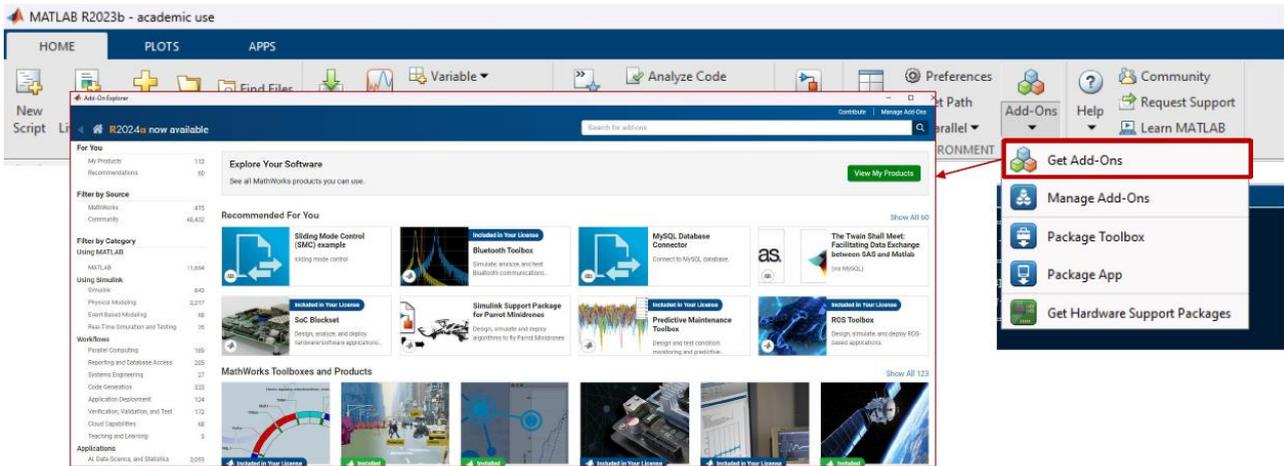


Figure 1: The Get Add-Ons interface to browse and install packages and toolboxes.

Toolboxes can be viewed and installed from the Add-Ons manager in the HOME toolbar of the MATLAB interface, as shown in Fig. 1. In general, the installation of these libraries increases the MATLAB functionalities enabling commands that would not be available otherwise. Additionally, each toolbox has its own dedicated section in the Documentation, with examples, theory, and references on the algorithms implemented. The MATLAB Documentation provides guidance on how to [Get and Manage Add-Ons](#)¹⁰.

```

1  N = 1024;
2  n = 0:N-1;
3
4  w0 = 2*pi/5;
5  x = sin(w0*n)+10*sin(2*w0*n);
6
7  s = spectrogram(x);
8
9  spectrogram(x,'yaxis')
```

→ MATLAB Function

→ Signal Processing Toolbox Function

Figure 2: Example of commands from different libraries. MATLAB functions are always available, functions from toolbox libraries are only available if installed.

⁹ <https://www.mathworks.com/products.html>

¹⁰ https://www.mathworks.com/help/matlab/matlab_env/get-add-ons.html

3. The MATLAB Interface

Understanding the MATLAB interface is crucial to use all of the software's capabilities. At the launch, MATLAB uses the default layout consisting in different sections:

1. The **Current Folder** section: As the name suggests, the Current Folder is the active directory MATLAB is working in. Any file, MATLAB variable (*.mat) or MATLAB code (*.m/*.mlx) must be inside of this folder (exceptions can be made using the command `addpath`). Right-clicking in this section allows to interface with MS Explorer, manage the MATLAB source control, and what subfolders contain files to be accessed by MATLAB in the current programming section.
2. The **Context menu**: The Context Menu contains access buttons to managing add-ons, creating new files, importing data, and changing MATLAB preferences.
3. The **Command Window**: The Command Window is the direct interface between the user and MATLAB. Non-suppressed output of scripts will be shown here, and commands can be sent directly to the program.
4. The **Workspace**: The MATLAB Workspace contains information and details on all the currently initialized objects and variables. Clicking and right-clicking on any variable allows the user to interact with it, such as exploring its content, editing, renaming, deleting, or saving to disk.

All of these sections can be moved around and reshaped to the user's preferences. To do so, follow the MATLAB guide to [Change Desktop Layout](https://www.mathworks.com/help/matlab/matlab_env/change-the-desktop-layout.html)¹¹. Other MATLAB settings can be changed in the MATLAB Preferences window (Gear icon next in the HOME toolbar). A guide to change the [Preferences](https://www.mathworks.com/help/matlab/ref/preferences.html)¹² is provided by MathWorks.

¹¹ https://www.mathworks.com/help/matlab/matlab_env/change-the-desktop-layout.html

¹² <https://www.mathworks.com/help/matlab/ref/preferences.html>

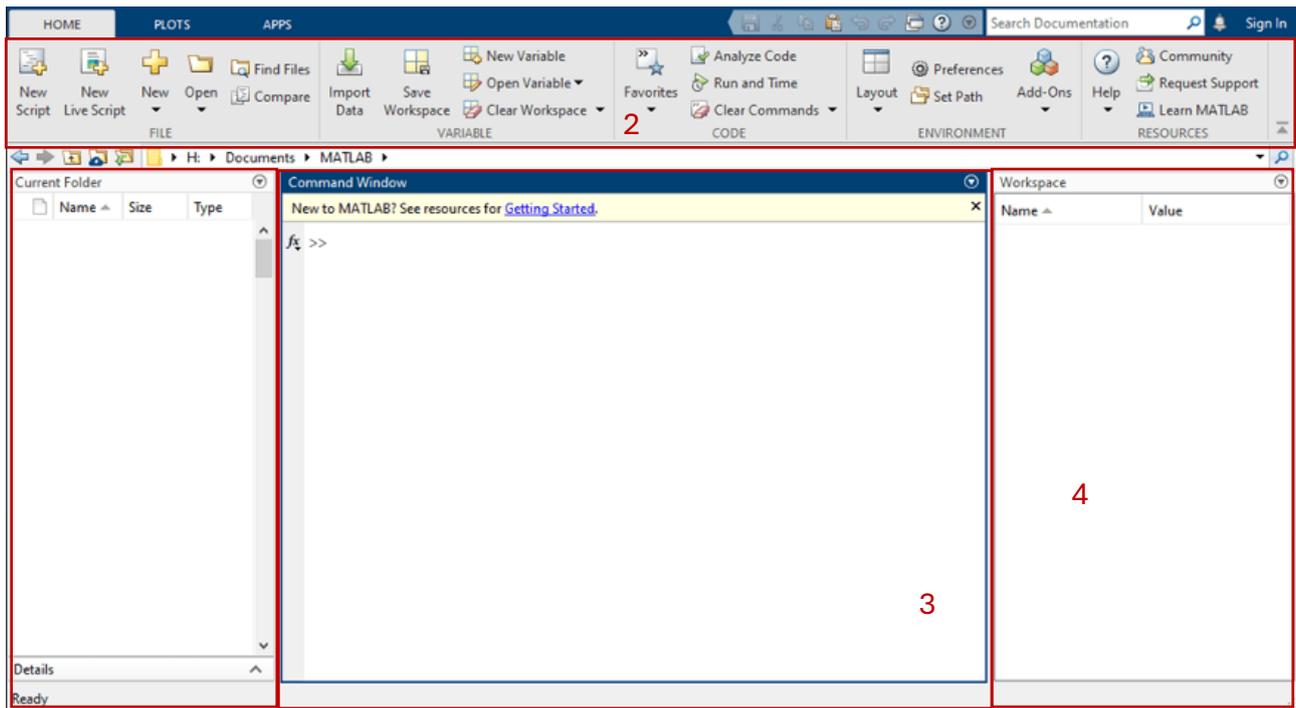


Figure 3: The differentiation of the MATLAB layout sections. 1) Current Folder; 2) Toolbars; 3) Command Window; 4) Global Workspace

3.1. Change Desktop Colors

Coding with a bright background puts strain on the users' eyes. A usual solution is using a "dark mode" to improve the code readability over long time. This can be done using the [Change Desktop Colors](#)¹³ documentation. MATLAB also [offers a set of predefined themes](#)¹⁴ to help users ([MATLAB Schemer](#)¹⁵).

3.2. Using the Command Window

The Command Window enables users to enter individual statements at the command line, indicated by the prompt (>>). As users enter statements, the Command Window displays the results. An overview of the [Command Window](#)¹⁶ is available in the Documentation. Fig. 4 defines the most common Actions to be taken in the Command Window. In general, the Command Windows can be used as an alternative to the User Interface, as commands like >>simulink and >>preferences will do the same as clicking on the relative icon.

¹³ https://www.mathworks.com/help/matlab/matlab_env/change-desktop-colors-and-select-dark-theme.html

¹⁴ <https://blogs.mathworks.com/pick/2019/05/17/whats-your-color-scheme/>

¹⁵ https://www.mathworks.com/matlabcentral/fileexchange/53862-matlab-schemer?s_tid=prof_contriblnk

¹⁶ <https://www.mathworks.com/help/matlab/ref/commandwindow.html>

Action	How to Perform the Action
Enter multiple statements on multiple lines before running any of the statements.	Enter the multiple statements at the command line, pressing Shift+Enter between statements. This key combination is unnecessary when you enter a paired keyword statement on multiple lines, such as for and end.
Clear a statement from the command line without executing it.	Press the Escape (Esc) key.
Recall previous statements.	Press the Up arrow ↑ key. The Command History window opens and displays a log of previous statements. To recall a specific statement, type any part of the statement and then press the Up arrow key. For example, to recall the command <code>b = 2</code> , type <code>b</code> , and then press the Up arrow key.
Clear the Command Window.	Call the <code>clc</code> function. To clear the Command Window without deleting any text, call the <code>home</code> function instead. Calling the <code>home</code> function moves the cursor to the upper-left corner of the Command Window and scrolls all visible text out of view, giving the appearance of clearing the screen without deleting any text.
Evaluate a statement already in the Command Window.	Select a statement, right-click, and then select Evaluate Selection .
Execute only a portion of the code currently at the command line.	Select the code at the command line and press Enter .  <code>fx >> disp("hello"), disp("world")</code>

Figure 4: List of actions that can be taken in the Command Window.

4. Setting up MATLAB

4.1. Managing File Paths

When calling files, MATLAB has a hierarchy in where to look. This means that keeping a clean list of folders for MATLAB to use is crucial, as duplicates might be present. MATLAB has a [Function Precedence Order](#)¹⁷ when looking for functions and files which is well defined in the Documentation. The files are searched through the [MATLAB Search Path](#)¹⁸. The search [Path](#)¹⁹ can be printed using the `path` command. To manage the file paths, useful command are [addpath](#)²⁰ and [rmpath](#)²¹ to change where MATLAB should look for files and functions. For small projects, in general, remember that the Current Folder has the highest priority when looking for files.

4.2. Defining startup Commands and Settings

Managing MATLAB effectively involves setting up startup and finish scripts, configuring the user path, saving the workspace, and defining a default starting folder. These practices help streamline the workflow and ensure consistency across MATLAB sessions. MATLAB allows users to execute scripts automatically at startup and upon exit. These scripts can be used to set up the environment, define default settings, and clean up when MATLAB closes. The [startup](#)²² and [finish](#)²³ scripts allow users flexibility in managing their work. Examples include

¹⁷ https://www.mathworks.com/help/matlab/matlab_prog/function-precedence-order.html

¹⁸ https://www.mathworks.com/help/matlab/matlab_env/what-is-the-matlab-search-path.html

¹⁹ <https://www.mathworks.com/help/matlab/ref/path.html>

²⁰ <https://www.mathworks.com/help/matlab/ref/addpath.html>

²¹ <https://www.mathworks.com/help/matlab/ref/rmpath.html>

²² <https://www.mathworks.com/help/matlab/ref/startup.html>

²³ <https://www.mathworks.com/help/matlab/ref/finish.html>

adding the `addpath` command into the startup file to create a personal library to be loaded at each access, or moving to a default custom folder every time MATLAB starts up.

4.3. MATLAB Source Control Integration

Source control consists in saving snapshots of project folders. Changes in code and files can be identified and earlier versions can be recovered easily. This means any change in the code can be logged and copies of the same files become obsolete. MATLAB supports natively both [Git](#)²⁴ and [SVN](#)²⁵. Both are similar and to the same thing, so only one is usually implemented (to the user's preference). The MATLAB Documentation provides a clear guide on [Source Control Integration in MATLAB](#)²⁶, as well as a dedicated page to setting up the [Source Control Integration](#)²⁷ using scripting and commands.

5. Coding in MATLAB

5.1. Help and Documentation

All MATLAB functions have supporting documentation that includes examples and describes the function inputs, outputs, and calling syntax²⁸. There are several ways to access this information from the command line:

- Open a function Documentation in a separate window using the `doc` command (`>> doc function`).
- Display function hints (the syntax portion of the function documentation) in the Command Window by pausing after you type the open parentheses for the function input arguments (`>> function()`).
- View an abbreviated text version of the function documentation in the Command Window using the `help` command (`help function`).

5.2. MATLAB Scripts

The simplest type of MATLAB program is called a script. A script is a file that contains multiple sequential lines of MATLAB commands and function calls. Users can run a script by typing its

²⁴ <https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>

²⁵ <https://www.perforce.com/blog/vcs/what-svn>

²⁶ https://www.mathworks.com/help/matlab/matlab_prog/about-mathworks-source-control-integration.html

²⁷ https://www.mathworks.com/help/matlab/source-control.html?s_tid=CRUX_lftnav

²⁸ https://www.mathworks.com/help/matlab/learn_matlab/help.html

name at the command line²⁹. Scripts can be created by typing `edit <scriptname>` in the command window or by clicking on New Script in the HOME toolbar.

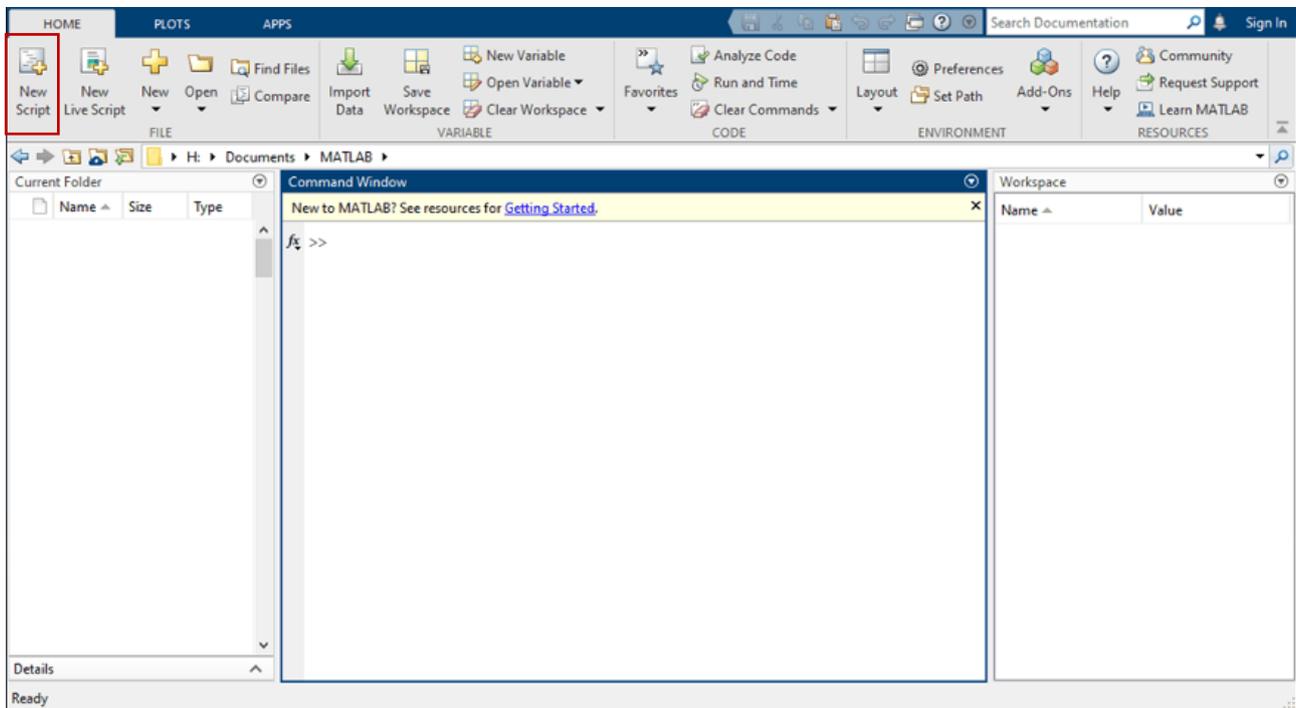


Figure 5: New Script Icon highlighted in the MATLAB interface.

Scripts contain instruction that are run by MATLAB every time the script is called. They can be used for simple instructions like initializing variables, or to do some batch processing of data. MATLAB scripts are easily opened and read using any text editor.

5.3. MATLAB Live Scripts

Live scripts are program files that contain code, output, and formatted text together in a single interactive environment called the Live Editor. In live scripts, users can write code and view the generated output and graphics along with the code that produced it. Add formatted text, images, hyperlinks, and equations to create an interactive narrative that users can share with others. The MATLAB Documentation has guides on how to [Create Live Scripts in the Live Editor](https://www.mathworks.com/help/matlab/live_editor/create-live-scripts-in-the-live-editor)³⁰. Using Live Scripts it is possible to create cells of code that can be run independently,

²⁹ https://www.mathworks.com/help/matlab/learn_matlab/scripts.html

³⁰ https://www.mathworks.com/help/matlab/matlab_prog/create-live-scripts.html

as well as [formatted text](#)³¹ to describe the code. Plots and figures can be shown directly in the [Live Editor](#)³².

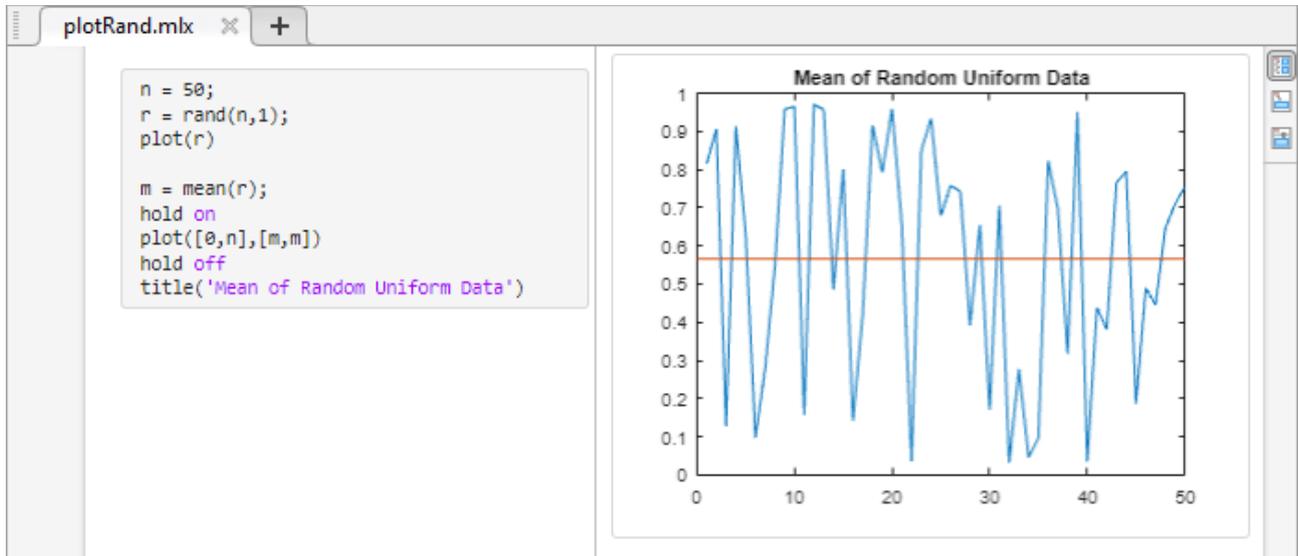


Figure 6: Example of commands inside of the Live Editor. The output of the code is shown on the right of the Live Editor window.

5.4. MATLAB Apps

MATLAB apps are interactive applications with a user-friendly interface that enable users to perform a computational task in MATLAB without writing any code. All the operations required to complete the task, such as importing data, performing calculations, and displaying results, are performed within the app. Users can use MATLAB apps to quickly produce results, visualizations, and data models without writing code³³. These apps include curve and distribution fitting, AI training, or control systems tuning. Generally, these apps provide an intuitive alternative to classic script coding. The [MATLAB File Exchange](#)³⁴ provides a list of published apps ready to be used.

³¹ https://www.mathworks.com/help/matlab/matlab_prog/format-live-scripts.html

³² https://www.mathworks.com/help/matlab/matlab_prog/modify-output-figures.html

³³ <https://www.mathworks.com/discovery/matlab-apps.html>

³⁴ <https://www.mathworks.com/matlabcentral/fileexchange/?term=type%253A%2522App%2522>

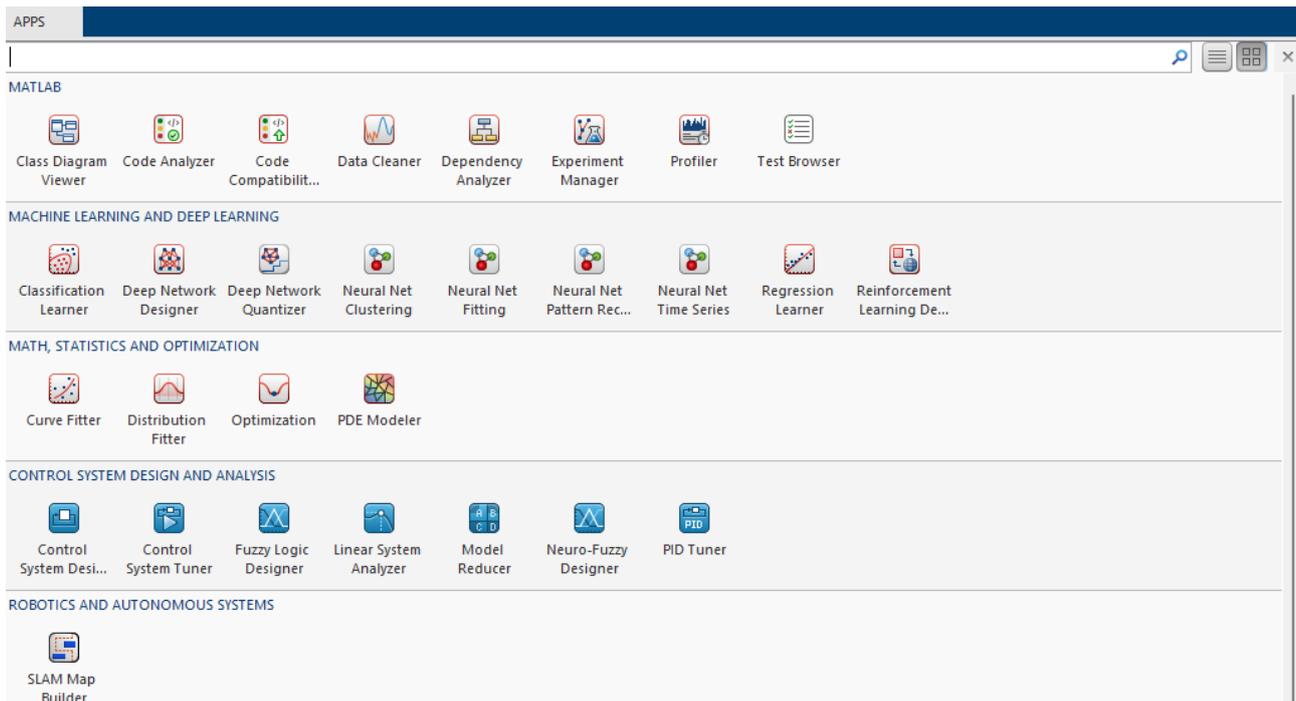


Figure 7: List (not comprehensive) of apps available in MATLAB after installing most of the toolboxes available.

5.5. Creating and Managing Functions

Functions are tasks or a set of tasks that are performed on a given set of input that transforms the input into a desired output. Usually these tasks need to be performed multiple times, so coding these task each time they are needed would be time consuming and make the code unnecessarily long. In MATLAB, functions are defined in separate files or within scripts, allowing users to reuse code efficiently. Functions have their own local variables and can accept input arguments and return output arguments. This means MATLAB functions can only see the variables passed as inputs, they work in a separate and local workspace, and everything that is computed inside of them and not passed as output is freed from memory. This allows users to programmatically execute calculations without burdening the workspace and the memory allocated to variables. The MathWorks provide an intuitive video on [How to Create a MATLAB Function](#)³⁵, as well as Documentation about [Functions](#)³⁶.

5.6. Debugging in MATLAB

Debugging consists in running and evaluating only parts of the code to find discrepancies, errors, or the source of unexpected outputs. MATLAB has its own implemented debugging functions to let the user explore what MATLAB is doing at every step. The MATLAB

³⁵ <https://www.mathworks.com/videos/functions-and-subfunctions-97413.html>

³⁶ <https://www.mathworks.com/help/matlab/functions.html>

Documentation provides guides on how to [Debug MATLAB Code Files](#)³⁷ and how to perform [Debugging and Analysis](#)³⁸. In its simplest form, this consists in adding a breakpoint inside of scripts. When run, MATLAB will pause every time it gets to the breakpoint (e.g. once every for loop if the breakpoint is inside of it). A very important feature of debugging using breakpoints is the possibility to scope the subfunctions local workspaces. While in debug mode, it is also possible to select what workspace to inspect, making this a great tool to access and evaluate variables that are usually deleted after the script is run, and cannot therefore be examined.

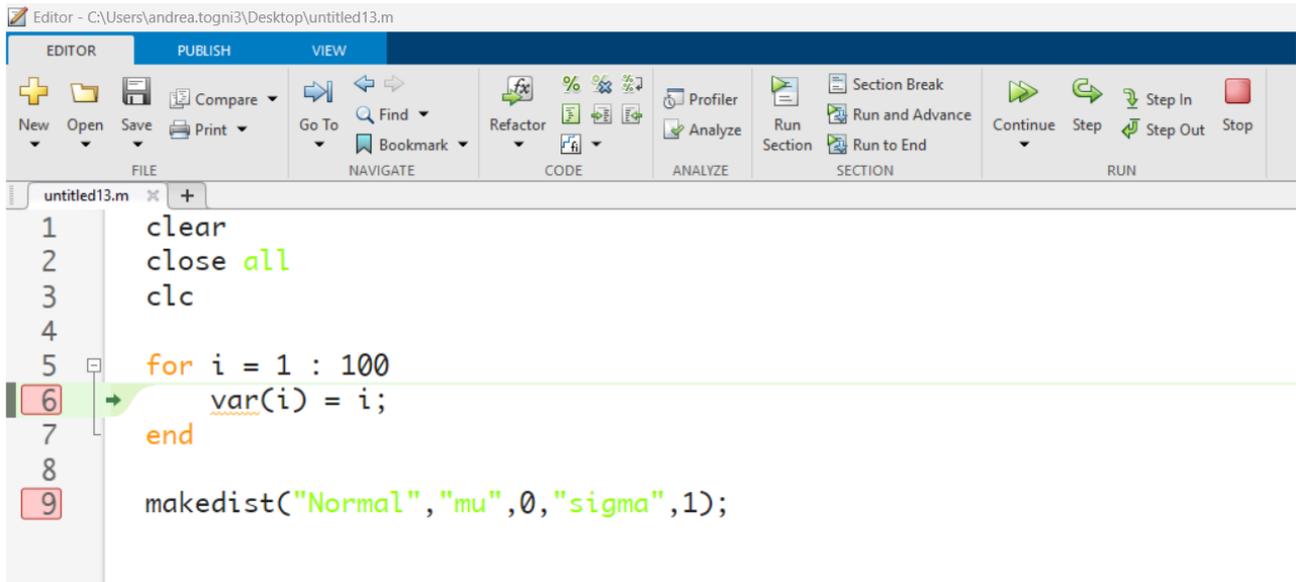


Figure 8: Example of a script in debugging mode. Lines 6 and 9 contain breakpoints which are respectively reached 100 and 1 times.

³⁷ https://www.mathworks.com/help/matlab/matlab_prog/debugging-process-and-features.html

³⁸ <https://www.mathworks.com/help/matlab/debugging-code.html>

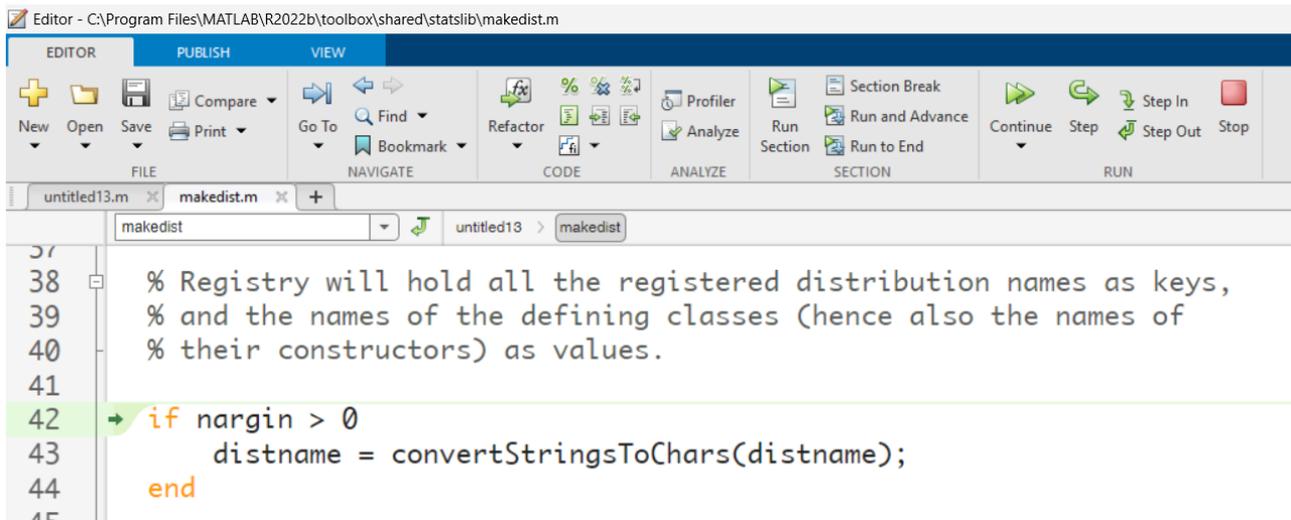


Figure 9: Example of a script in debugging mode when inside of a subfunction. The local workspace is highlighted above the code (current workspace from makedist).

5.7. Calling Simulink

Simulink is the companion software of MATLAB that allows user to simulate dynamical models with continuous or discrete variables. This means that it is possible to simulate the effect of physical phenomena (continuous) on digital systems (discrete) like controllers and plants. While Simulink has its own interface and commands, models can be programmatically launched through scripts using the `sim` command. The Documentation provides guidance on how to [Run Simulations Programmatically](https://www.mathworks.com/help/simulink/ug/using-the-sim-command.html)³⁹ and make MATLAB and Simulink work in synergy.

5.8. Evaluating Code Performance

The MATLAB Documentation provides guidance on how to [Measure the Performance of Your Code](https://www.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html)⁴⁰. The easiest way to do so is using the `tic-toc` commands. These are called sequentially and print to the command window the elapsed time between their call.

³⁹ <https://www.mathworks.com/help/simulink/ug/using-the-sim-command.html>

⁴⁰ https://www.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html

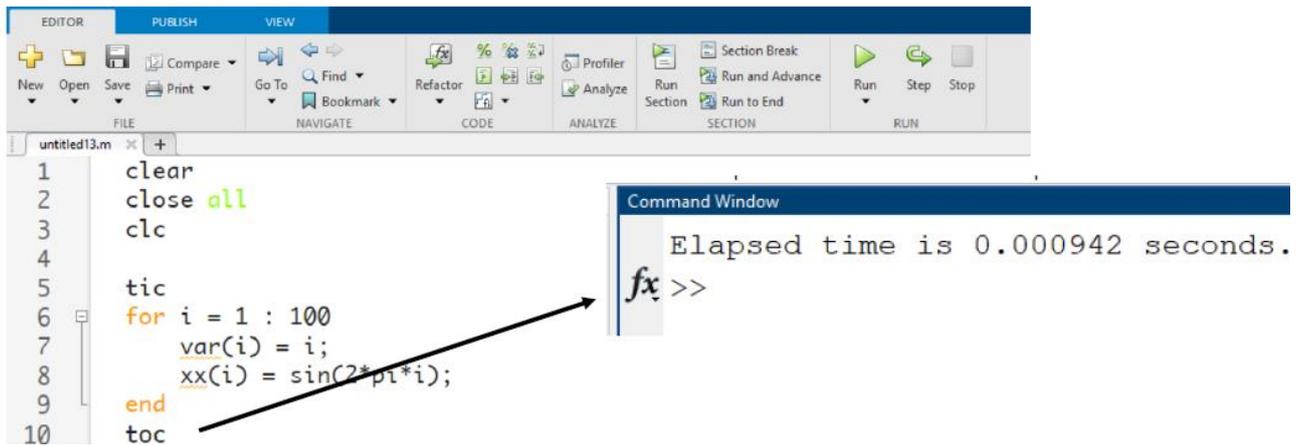


Figure 10: Example and output in the Command Window of running the tic-toc commands.

A most comprehensive analysis can be performed using the [Code Profiler](#)⁴¹. This tool can be called by selecting the “Run and time” option in the Toolbar, and allows users to profile every line of code. This means that the execution of every command is analysed by MATLAB, which produces a report on the number of function calls, and the elapsed time of each line. Additionally, the Profiler works on every subfunction in the script, making it a general and powerful tool for code revision and performance improvement.

⁴¹ https://www.mathworks.com/help/matlab/matlab_prog/profiling-for-improving-performance.html

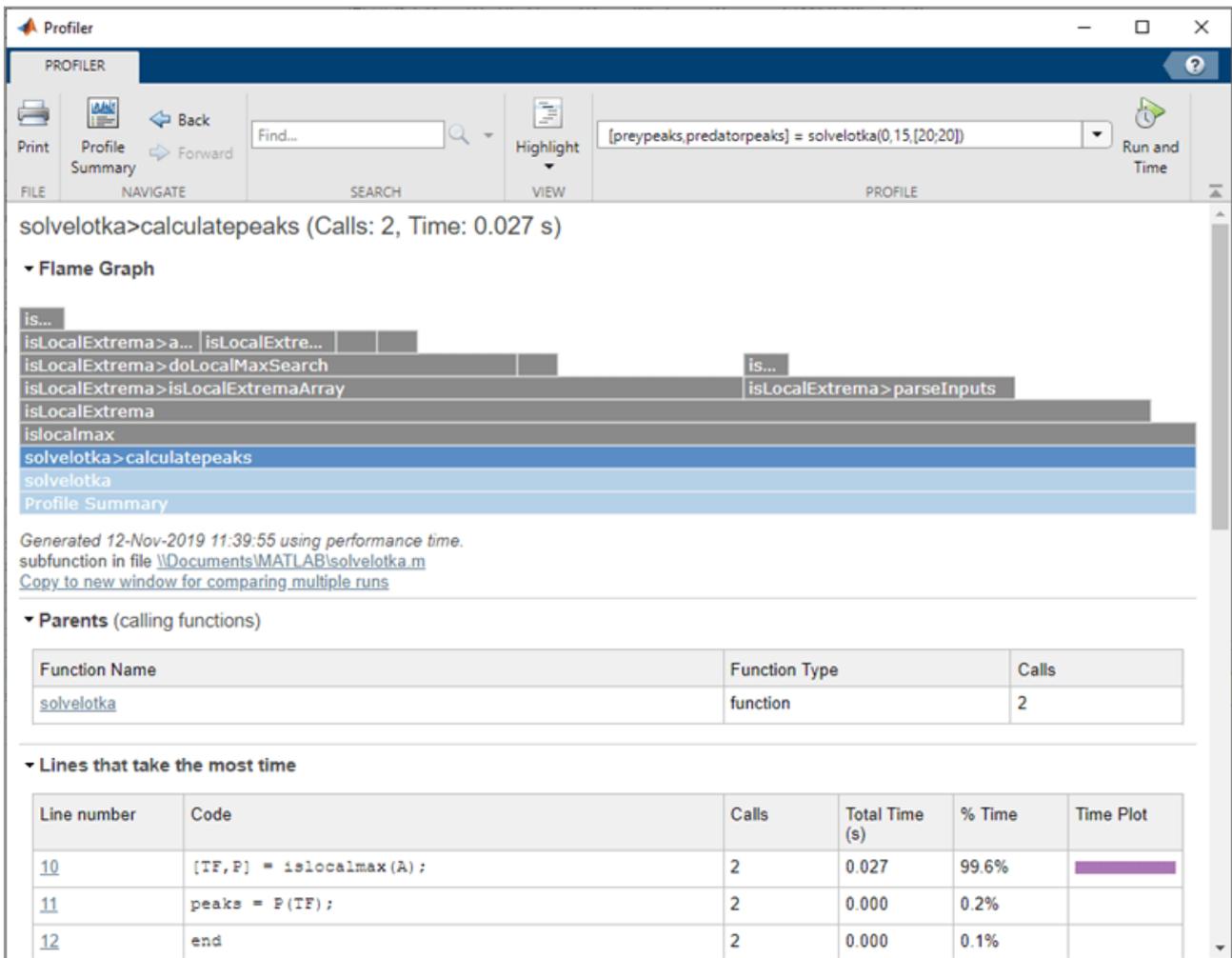


Figure 11: Profiler window. Calls, subfunction and single lines are highlighted as contributors to the total computing time (0.027 s).

6. Code Examples

The following links give access to a series of MATLAB live scripts created to provide further context and examples on how to use MATLAB. These can be opened using MATLAB and executed in the Live Editor to show applications of the concepts presented.

6.1. Coding Fundamentals

[Arithmetic Operations in MATLAB](#)

[MATLAB Data Types](#)

[Creating, Managing, and Indexing Variables in MATLAB](#)

[The Command Window](#)

[MATLAB Arrays and Matrices Tutorial](#)

[Using Loops and Control Flow in MATLAB](#)

[Introduction to Function Handles in MATLAB](#)

6.2. Data Importing

[Introduction to Importing Data in MATLAB](#)

[Extracting Points from Images](#)

[Import Data Tool](#)

[Importing and Reading Images in MATLAB](#)

[Cleaning Data](#)

6.3. Data Processing and Applications

[Linear Least-Square Fit in MATLAB](#)

[Non-Linear Least-Square Fit in MATLAB](#)

[Introduction to Fitting and Creating Statistical Distributions in MATLAB](#)

[Introduction to Image Processing in MATLAB](#)

[Introduction to Interpolation in MATLAB](#)

[Numerical Integration of a Function using trapz](#)

[Discrete Integration of a Function using Trapezoidal Rule](#)

[Numerical Integration of a Function using integral](#)

[Integration of a Function using the Symbolic Toolbox](#)

[Introduction to Fast Fourier Transform \(FFT\) in MATLAB](#)

[Finding Roots of Nonlinear Functions in MATLAB](#)

[Solving Ordinary Differential Equations \(ODEs\) in MATLAB](#)

[Using Transfer Functions in MATLAB](#)

[Using Units of Measurement in MATLAB](#)

[Implementing and Visualizing Time-Frequency Distributions in MATLAB](#)

[Differentiation in MATLAB](#)

6.4. Data Export and Visualization

[Introduction to 2D Plotting in MATLAB](#)

[Introduction to 3D Plotting in MATLAB](#)

[2D Plotting in MATLAB and Exporting](#)

6.5. Good to Know

[Creating and Managing Functions in MATLAB](#)

[Best Practices for Managing MATLAB](#)

[Managing File Paths and Custom Folders](#)

[Adding Folders and Subfolders to MATLAB Path](#)

[Parallel Computing](#)

7. Where to go from here...

Novice users can improve their ability with MATLAB by going through the the [MATLAB Onramp](#)⁴² course to make sure the fundamentals are well established. A more comprehensive course can be found in the [MATLAB Fundamentals](#)⁴³. These provide a solid background in MATLAB's general usage. Intermediate users can refer to the documentation to discover specific libraries to facilitate completing their projects. As an example, the documentation related to the Automotive Toolbox is available at this [page](#)⁴⁴, the Aerospace Toolbox at this [page](#)⁴⁵, and the Signal Processing Toolbox at this [one](#)⁴⁶. For a complete and comprehensive documentation, MATLAB also provides multiple [PDF Documentation Files for MATLAB](#)⁴⁷. The documentation files elaborate on the topics seen in this introductory guide. For example, the [MATLAB Graphics Documentation](#)⁴⁸ provides complete information on how to create 2D plots, at the cost of having a verbose document (700+ pages). Still, knowing where to look for detailed information is crucial. Advanced or independent users who wish to learn MATLAB more in-depth can read

⁴² <https://matlabacademy.mathworks.com/details/matlab-onramp/gettingstarted>

⁴³ <https://matlabacademy.mathworks.com/details/matlab-fundamentals/mlbe>

⁴⁴ https://www.mathworks.com/help/overview/automotive.html?s_tid=CRUX_lftnav

⁴⁵ https://www.mathworks.com/help/overview/aerospace.html?s_tid=CRUX_lftnav

⁴⁶ https://www.mathworks.com/help/overview/signal-processing.html?s_tid=CRUX_lftnav

⁴⁷ https://www.mathworks.com/help/pdf_doc/matlab/index.html

⁴⁸ https://www.mathworks.com/help/pdf_doc/matlab/creating_plots.pdf

the [PDF Documentation Files for MATLAB](#)⁴⁹ to assess advanced programming techniques, or explore the MATLAB [Application Products](#)⁵⁰ to see what libraries are available to use. Additionally, MATLAB provides a list of affiliated [MATLAB and Simulink Based Books](#)⁵¹. These are generally written by experts of the field and include specialized theory and numerical examples that can be downloaded and run on MATLAB, making for a great support to develop custom applications.

Additional Resources

Examples Library:

[MATLAB Documentation Examples](#)

[MATLAB Documentation Videos](#)

[MATLAB Documentation Functions List](#)

[MATLAB Live Script Gallery](#)

MathWorks Resources:

[MATLAB Official Website](#): The official website from MathWorks.

[MATLAB Academy](#): Learn MATLAB for free with MATLAB Onramp and access interactive self-paced online courses and tutorials on Deep Learning, Machine Learning and more.

[MATLAB Documentation Website](#): Documentation, examples, videos, and other support resources for MathWorks products including MATLAB and Simulink.

[MATLAB Cody](#): A fun community coding game to challenge your skills and learn MATLAB.

[MATLAB Central](#): An open exchange for the MATLAB and Simulink user community.

[MATLAB File Exchange](#): Download and share free MATLAB code, including functions, models, apps, support packages and toolboxes.

[MATLAB Online](#): MATLAB Online provides access to MATLAB from any standard web browser wherever you have Internet access.

Interactive Courses:

[MATLAB Onramp](#)

[Core MATLAB Skills \(Series of courses\)](#)

⁴⁹ https://www.mathworks.com/help/pdf_doc/matlab/index.html

⁵⁰ <https://www.mathworks.com/products.html>

⁵¹ https://www.mathworks.com/academia/books.html?s_tid=hc_resources

[Build MATLAB Proficiency \(Series of courses\)](#)

[MATLAB Skills for Simulink Modeling](#)

Documentation:

[PDF Documentation Files for MATLAB](#)

Video:

[Official Youtube Channel](#): Examples, applications and other support resources for MathWorks products including MATLAB and Simulink in video form.

[Introduction to MATLAB Playlist](#)

Other Resources:

[Unibo Student Ambassadors](#): MATLAB Student Ambassadors are important links between MathWorks and their universities. Located all over the world, ambassadors inform students about the availability of MATLAB and Simulink on campus and showcase the products' functionality.