Hands On: Introduction to MATLAB - Part 1 - Starting MATLAB

This is the first MATLAB live script of the collection ***Hands On: Using MATLAB in the 267MI "System Dynamics" course***, devoted to introduce the MATLAB/Simulink environment and tools for solving practical problems related to the topics of the 267MI course, i.e. performance analysis of dynamic systems, parametric estimation, identification of models from data, and prediction of the evolution of dynamic systems.

Use this link to go back to the main live script of the collection.

Table of Contents

Goals

The aim of this live script is to make you familiar with the MATLAB environment.

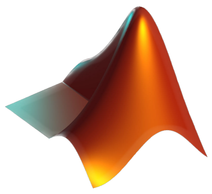
What you should know by the end of this part:

- How to start MATLAB on a computer in the University of Trieste computer laboratories in the C1 or H2bis buildings, or on your own PC;
- What is the structure of the MATLAB Desktop (i.e. the MATLAB command window and the other MATLAB windows) and how to use it proficiently;
- How to get help within MATLAB;
- How to create or open a file in the MATLAB editor.

Starting MATLAB

MATLAB is installed on the computers in the Computer Labs of the Dept. of Architecture & Engineering at the University of Trieste. Moreover, the University of Trieste has subscribed to a MATLAB TAH (Total Academic Headcount) licence that allows students (as well as professors and university staff) to download the software and install it on their PCs. For more details regarding the available products and how to download and activate the software, please visit the dedicated MathWorks web portal.

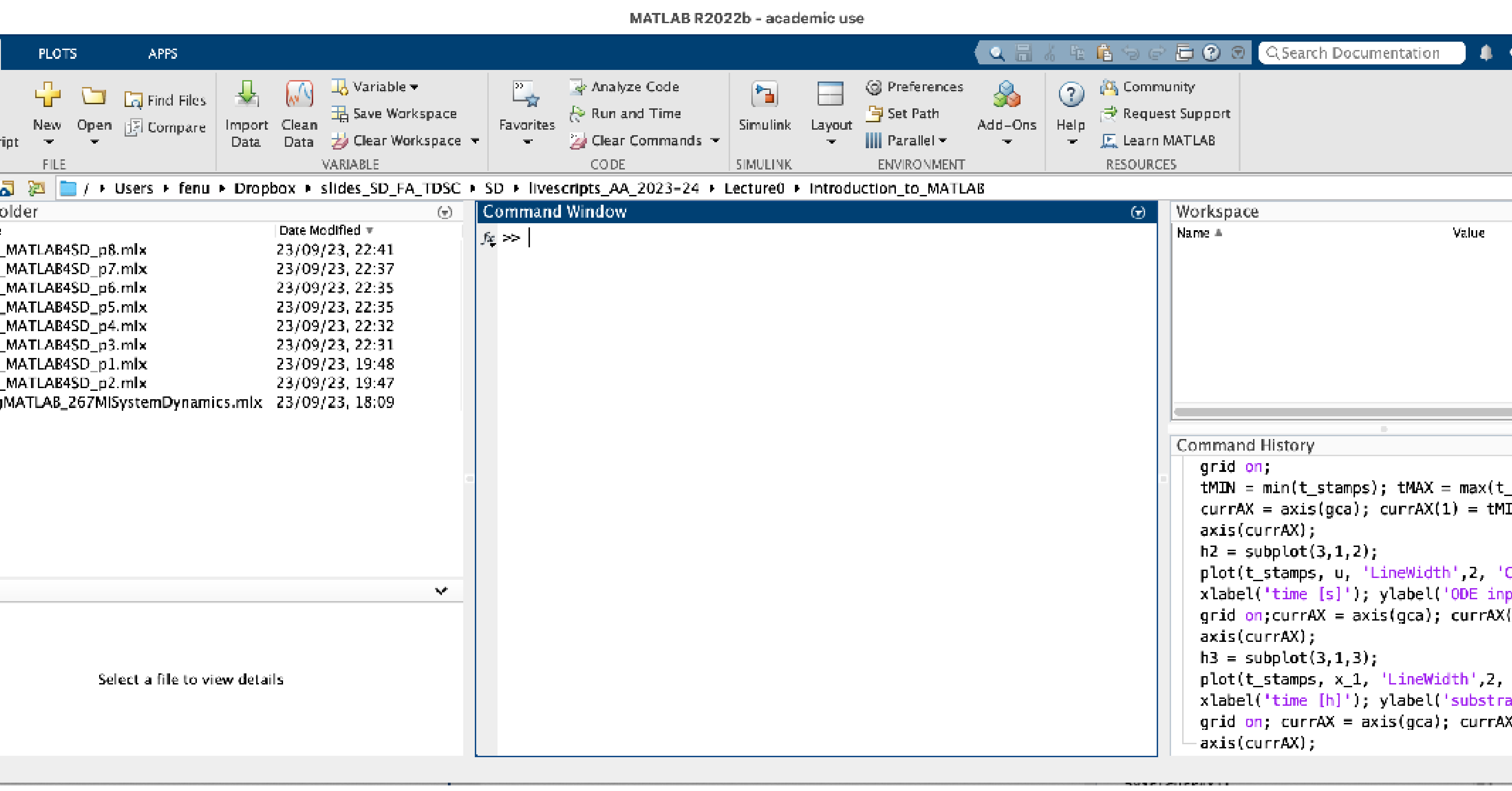
The MATLAB logo is similar to the image, shown below, of an L-shaped membrane



- MATLAB can be started on an MS Windows-based computer by clicking on the MATLAB logo icon in the Start menu or double-clicking the MATLAB logo on the desktop (if the MATLAB icon is on the desktop).
- Similarly, MATLAB can be started on a Linux-based computer by clicking the MATLAB logo icon in the Application menu or simply by typing matlab and pressing the "return/enter" key in a Terminal window.
- Finally, on a Apple Mac, you may start MATLAB by clicking the MATLAB logo icon in the Dock, or in the Applications.

The Command Window

When you **first** start MATLAB you should see the following group of windows:



	Date Modified
_MATLAB4SD_p8.mlx	23/09/23, 22:41
_MATLAB4SD_p7.mlx	23/09/23, 22:37
_MATLAB4SD_p6.mlx	23/09/23, 22:35
_MATLAB4SD_p5.mlx	23/09/23, 22:35
_MATLAB4SD_p4.mlx	23/09/23, 22:32
_MATLAB4SD_p3.mlx	23/09/23, 22:31
_MATLAB4SD_p1.mlx	23/09/23, 19:48
_MATLAB4SD_p2.mlx	23/09/23, 19:47
gMATLAB_267MISystemDynamics.mlx	23/09/23, 18:09

```
>>
```

```
grid on;
tMIN = min(t_stamps); tMAX = max(t_
currAX = axis(gca); currAX(1) = tMI
axis(currAX);
h2 = subplot(3,1,2);
plot(t_stamps, u, 'LineWidth',2, 'C
xlabel('time [s]'); ylabel('ODE inp
grid on;currAX = axis(gca); currAX(
axis(currAX);
h3 = subplot(3,1,3);
plot(t_stamps, x_1, 'LineWidth',2,
xlabel('time [h]'); ylabel('substra
grid on; currAX = axis(gca); currAX
axis(currAX);
```

The most important window is the **Command Window** (the one in the centre), in which you type MATLAB commands and see the results of the commands.

There are also the **Current Folder**, **Workspace**, **Command History** and **Details** windows. The *Current Folder* window shows that MATLAB is working in a folder containing the files `usingMATLAB_267MISystemDynamics.mlx`, `Intro_MATLAB4SD_p1.mlx`, ..., `Intro_MATLAB4SD_p8.mlx` .

In particular:

- **Command Window**: this is the MATLAB command interpreter window. Commands can be directly written there, or programs in MATLAB programming language can be executed.
- **Command History** is the window that contains the list of commands entered in the Command Window.
- **Current Directory**: it displays the contents of the work folder in graphic form.
- **Workspace**: this window displays the contents of the memory area used by MATLAB (names, types and sizes of variables).
- **Details** is the window that displays detailed information on the file selected in the Current Directory window: format, preview (if possible), etc.

You can customise your settings using the **Preferences** menu in the Environment section of the Home tab. Initially it is simplest to use MATLAB's default settings. When you have decided what is the most effective way for you to work, you may close some or all of these windows. However you will always need a *Command Window*.

A First Example

To see how MATLAB works, please type the following two lines in the MATLAB Command Window, with **Enter** after each command.

```
x = 2
x = x/2 + 1/x
```

- The first command assigns the value 2 to the variable x.
- The second one uses the value stored in the variable x, evaluates the expression on the right-hand side of the =, and then stores the new value in the variable x.
- After entering each command, the result of the command is shown in the Command Window.

The First Code Example in an Interactive Environment: a MATLAB Live Script

MATLAB **live scripts** are interactive documents that combine MATLAB code with formatted text, equations and images in a single environment. In addition, live scripts store and display the output together with the code that creates it (from the MATLAB documentation). More details and examples of live scripts are available in the MATLAB documentation.

If not already done, open this document (the file named `Intro_MATLAB4SD_p1.mlx`) directly using MATLAB. Put the mouse cursor in this document section and select the command "`Run Section`" from the **Live Editor tab** or, alternatively, click the vertical striped bar to the left of the code

```
x = 2
```

```
x = 2
```

```
x = x/2+ 1/x
```

```
x = 1.5000
```

Executing this portion of code within the live script produces the same result, displayed in the same way, as code executed by entering commands directly in the Command Window.

From now on, we will use live scripts most of the time.

The MATLAB Programming Language

MATLAB is an interpreted programming language. This means that every command is evaluated when you type it into the Command Window and press the Enter key. It is possible to create sequences of commands and save them in a text file (a **script**) or create more complex programmes using **functions** and also classes. Even when you execute a MATLAB script or function, the code is interpreted.

This has the great advantage of allowing interaction with MATLAB.

There are several ways to return to a previous command, edit it to correct any errors and then execute it again, when typing the code directly in the **Command Window**. The easiest way to do this is to use the arrow keys on the keyboard: **the up arrow** allows you to return to the **previous command** (pressing several times can go back several commands), while the left and right arrow keys position the cursor within the command and the Backspace or Delete keys remove a character that can be typed again. Pressing the Enter key executes the edited command.

If you type a few characters from the the beginning of a command, for example `x =`, and then press the up arrow key, you go back to the most recent command that started with these characters. Alternatively you can find the command you want in the **Command History** sub-window, and double click on it to execute it again. You may also copy commands from the Command History sub-window using the menu opened when you right-click on them.

Example: Using the Arrow Keys in the Command Window

Use the arrow keys to go back to the previous command and execute it again, several times

```
        x = x/2+ 1/x
```

Check that every time you execute the command (i.e., every time you press the Enter key after recalling the command using the arrow key), the command itself has been stored in the Command History command list.

Example: Repeating a Command in a Live Script

Neither the up arrow key nor the Command History command list approach works when using a live script. If you want to repeat a command (or a piece of code) in a live script, you should use the following approach (by means of a so called *Control Button* - more details in the MATLAB documentation on Live Scripts and Interactive Controls):

We have already defined the variable `x` in this live script, assigning it an initial value and then performing a simple calculation that assigns the result back to the variable `x`.

Now, let's run the following piece of code clicking on the "Run Again" button more than once. What happens?

```
disp(x)
```

```
    1.5000
```

```
x = x/2 + 1/x
```

```
x = 1.4167
```

Getting Help

There are several ways of getting help within MATLAB. It is essential you become familiar with both the MATLAB `help` command and the MATLAB Help browser which has extensive documentation. MATLAB also contains a number of demonstration videos that can help you become familiar with it.

Help command

The quick way to get information about a MATLAB command is to type

```
help command_name
```

in the MATLAB Command Window. This may produce more information that will fit in one screen so you will have to scroll back to the beginning of the output.

Examples

Use the `help` command to find out more information about the `help` command

```
help help
```

```
 help - Help for functions in Command Window
    This MATLAB function displays the help text for the functionality
    specified by name, such as a function, method, class, toolbox, or
    variable.

    help name
    help

    Input Arguments
        name - Functionality name
            character vector | string scalar

    See also doc, lookfor, more, what, which, whos

    Introduced in MATLAB before R2006a
    Documentation for help
    Other uses of help
    Folders named help
```

Use the `help` command to find out information about a MATLAB toolbox (a collection of MATLAB functions and apps, with focus on a particular topic, as Signal Processing, Control Systems, System Identification, Statistics etc.):

```
help ident
```

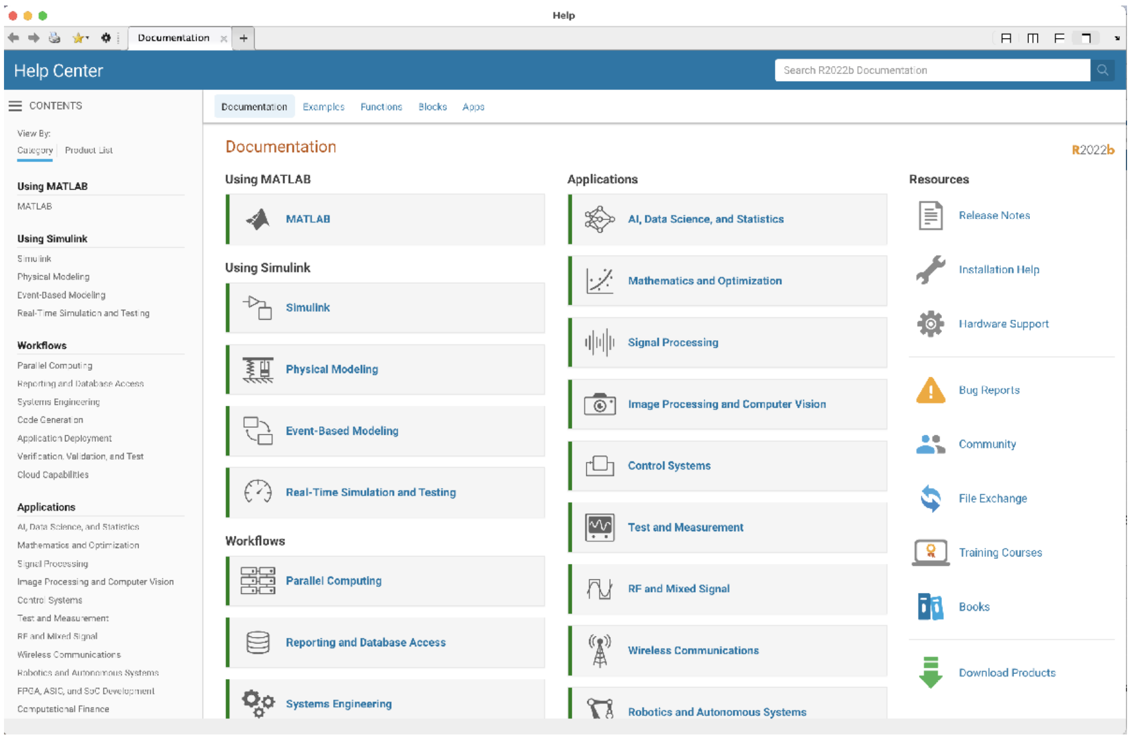
```
  System Identification Toolbox
  Version 10.0 (R2022b) 13-May-2022

  General.
    identpref        - Set System Identification Toolbox preferences.
    InputOutputModel - Overview of input/output model objects.
    DynamicSystem    - Overview of dynamic system objects.
    lti              - Overview of linear time-invariant system objects.
    idlti            - Overview of identified linear models.
    idnlmodel        - Overview of identified, nonlinear dynamic models.

  Apps.
    systemIdentification - App for model identification and analysis

  Data analysis and processing.
    iddata           - Construct an input-output data object.
    idfrd            - Construct a frequency response data object.
    iddata/detrend   - Remove trends from data sets.
    delayest         - Estimate the time delay (dead time) from data.
    iddata/feedback  - Investigate feedback effects in data sets.
    getTrend         - Offsets and linear trends of data sets.
    iddata/fft       - Transform data from time to frequency domain.
    iddata/ifft      - Transform data from frequency to time domain.
    iddata/getexp    - Retrieve separate experiment(s) from multiple-experiment
                       iddata objects.
    iddata/advice    - Advice about a data set.
    iddata/merge     - Merge several data experiments.
    nkshift          - Shift data sequences for delays.
    iddata/plot      - Plot input-output data.
    iddata/resample  - Resample data by decimation and interpolation.
    iddata/isnlarx   - Test if a nonlinear ARX model is better than a linear.
    misdata          - Estimate and replace missing input and output data.
    idfilt           - Filter data through Butterworth filters.
    idinput          - Generates input signals for identification.
    iddata/isreal    - Check if a data set contains real data.

  Nonparametric estimation.
    cra              - Compute impulse response by correlation analysis.
    etfe             - Empirical Transfer Function Estimate and Periodogram.
    impulseest       - Direct estimation of impulse response as FIR filter.
    spa              - Spectral analysis.
    spafdr           - Spectral analysis with frequency dependent resolution.

  Linear model identification.
    tfest            - Transfer function model estimation.
    ssest            - State-space model estimation.
    procest          - Process model estimation.
    ar               - AR-models of signals using various approaches.
    spectrumest      - Minimum-phase transfer function for power spectrum data.
    era              - Eigensystem Realization Algorithm to fit impulse response data.
    armax            - Prediction error estimate of an ARMAX model.
    arx              - LS-estimate of ARX-models.
    arxRegul         - ARX regularization scaling and weighting data.
    oe               - Prediction error estimate of an output-error model.
    bj               - Prediction error estimate of a Box-Jenkins model.
    polyest          - Prediction error estimate of a generic polynomial model.
    greyest          - Parameter estimation for a linear grey-box model.
    ivar             - IV-estimates for the AR-part of a scalar time series.
    iv4              - Approximately optimal IV-estimates for ARX-models.
    n4sid            - State-space model estimation using a subspace method.
    ssregest         - State-space model estimation by reduction of regularized ARX models.
    pem              - Update parameters of a model to fit estimation data.
    init             - Initialize (randomize) the parameters of a model.
    getpar           - Get parameter attributes of linear models.
    setpar           - Set parameter attributes of linear models.
    idpar            - Parameter configuration for structured estimation
                       of initial states and input levels.

  Nonlinear model identification.
    nlarx            - Identify a Nonlinear ARX model.
    nlhw             - Identify a Hammerstein-Wiener model.
    nlgreyest        - Estimate nonlinear grey-box model parameters.
    nlssest          - Train a nonlinear state space model.
    pem              - Update parameters of a model to fit estimation data.

  Model structure creation.
    idtf                - Create transfer functions with identifiable parameters.
    idproc              - Create simple continuous-time process models with identifiable parameters.
    idss                - Create state-space models with identifiable parameters.
    idpoly              - Create linear polynomial models with identifiable parameters.
    idfrd               - Create frequency response data models.
    idgrey              - Create user-parameterized (grey-box) linear models.
    idnlarx             - Create nonlinear ARX models.
    idnlgrey            - Create nonlinear user-parameterized models.
    idnlhw              - Create nonlinear Hammerstein-Wiener type models.
    idNeuralStateSpace  - Nonlinear state-space models based on deep networks.
                          (requires Deep Learning Toolbox)

  Model data extraction.
    tfdata           - Numerators and denominators and their standard deviations.
    zpkdata          - Zero/pole/gain and their standard deviations.
    idssdata         - State-space matrices and their standard deviations.
    polydata         - Polynomials corresponding to idpoly structure and their
                       standard deviations.
    frdata           - Frequency response data and its covariance.
    get              - Properties of model object.
    getpvec          - Model parameters and their standard deviations.
    getcov           - Parameter covariance of an identified model.

  Model conversion.
    c2d              - Continuous to discrete conversion.
    d2c              - Discrete to continuous conversion.
    d2d              - Resample discrete-time model.
    ss2ss            - State coordinate transformation.
    canon            - Canonical forms of state-space models.
    ssform           - State space model structure configuration.
    chgFreqUnit      - Change frequency units in IDFRD model.
    addMinPhase      - Add minimum phase to frequency response magnitude.
    chgTimeUnit      - Change time units of models.
    noise2meas       - Represent noise component as input-output model.
    noisecnv         - Append noise inputs to measured inputs.
    balred           - State-space model order reduction.
    idnlarx/findop   - Find idnlarx model's operating point.
    idnlhw/findop    - Find idnlhw model's operating point.
    data2state       - Map past input-output values to states of model.
    linapp           - Linear approximation of nonlinear model for a given input.
    linearize        - Linearization of nonlinear models about an operating point.
    ss,tf,zpk,frd    - Transformations to numeric LTI models of Control System Toolbox.
    Most Control System Toolbox conversion routines also apply to the
    identified linear models of System Identification Toolbox.

  Simulation and prediction.
    idmodel/forecast - Forecast the response of identified model.
    predict          - Prediction over the observed sample range.
    pe               - Compute prediction errors.
    idmodel/sim      - Simulate a model with user-defined inputs.
    idParametric/simsd    - Monte Carlo simulations of identified linear models.
    slident               - Simulink library for recursive estimation and simulation.
    initialCondition - Initial condition representation for linear time-invariant systems.

  Uncertainty analysis.
    simsd            - Monte Carlo simulations of linear models.
    rsample          - Random sampling of linear models.
    getcov           - Parameter covariance of an identified model.
    setcov           - Modify parameter covariance data of an identified model.
    translatecov     - Translate covariance data across model transformations.
    present          - Model display with parameter standard deviations.
    showConfidence   - View confidence regions on bode, step, impulse, pole-zero
                       map and Nyquist plots of linear models.

  Model analysis and validation.
    compare          - Compare simulated/predicted output with measured output.
    pe               - Prediction errors.
    predict          - N-step ahead prediction over observed sample range.
    forecast         - Forecast the response of identified model N steps.
    resid            - Compute and test the residuals associated with a model.
    sim              - Model response and its uncertainty to user-defined input signal.
    bode             - Bode plot of linear models (with uncertainty region).
    step             - Step response of linear and nonlinear models (with uncertainty region).
    impulse          - Impulse response of linear models (with uncertainty region).
    nyquist          - Nyquist diagram of linear models (with uncertainty region).
    iopzmap          - Zeros and poles of a linear model (with uncertainty regions).
    freqresp         - Frequency response over a frequency grid.
    evalfr           - Evaluate frequency response at given frequency.
    findstates       - Initial state estimation.
    goodnessOfFit    - Goodness of fit measures for comparing data.
    absorbDelay      - Replace delays by poles at z=0 or phase shift.
    nparams          - Number of model parameters.
    isstable         - True for linear models with stable dynamics.
    idParametric/advice   - Advice about an estimated model.
    idParametric/spectrum - Noise spectrum of time series models (with uncertainty regions).
    DynamicSystem/view    - View linear model responses (requires Control System Toolbox)

  Model structure selection.
    aic              - Compute Akaike's information criterion.
    fpe              - Compute final prediction criterion.
    arxstruc         - Loss functions for families of ARX-models.
    selstruc         - Select model structures according to various criteria.
    struc            - Generate typical structure matrices for ARXSTRUC.

  Recursive parameter and state estimation.
    recursiveARX     - Recursively estimate an ARX model.
    recursiveARMAX   - Recursively estimate an ARMAX model.
    recursiveOE      - Recursively estimate an Output-Error model.
    recursiveBJ      - Recursively estimate a  Box-Jenkins model.
    recursiveAR      - Recursively estimate an AR model.
    recursiveARMA    - Recursively estimate an ARMA model.
    segment          - Segment data and track abruptly changing systems.
    extendedKalmanFilter  - Extended  Kalman filter state estimator for discrete nonlinear models.
    unscentedKalmanFilter - Unscented Kalman filter state estimator for discrete nonlinear models.
    particleFilter        - Particle filter state estimator for discrete nonlinear models.

  Nonlinear mapping functions.
    idCustomNetwork        - Nonlinear function with user-specified unit function.
    idDeadZone             - Dead zone.
    idLinear               - Linear function with offset.
    idFeedforwardNetwork   - Multi-layer neural network
                            (requires Deep Learning Toolbox)
    idPolynomial1D         - One-dimensional polynomial function.
    idPiecewiseLinear      - Piecewise linear function.
    idSaturation           - Saturation.
    idSigmoidNetwork       - Sigmoid network.
    idTreePartition        - Tree partition function.
    idUnitGain             - Unit gain function.
    idWaveletNetwork       - Wavelet network.
    idGaussianProcess      - Gaussian Process (GP) function.
                             (requires Statistics and Machine Learning Toolbox)
    idTreeEnsemble         - Regression tree ensemble.
```

```
                           (requires Statistics and Machine Learning Toolbox)
    idSupportVectorMachine - Support Vector Machine (SVM) function.
                           (requires Statistics and Machine Learning Toolbox)
    idnlfun/evaluate       - Evaluate nonlinear mapping function value.
    idnlfun/reset          - Reset parameters of nonlinear mapping function.

  Regressor and parameter management for nonlinear models.
    linearRegressor     - Linear regressor set specification.
    polynomialRegressor - Polynomial regressor set specification.
    periodicRegressor   - Periodic regressor set specification.
    customRegressor     - Custom formula regressor set specification.
    idnlarx/getreg      - Get regressors of a nonlinear ARX model.
    idnlgrey/getinit    - Get initial states of a nonlinear grey-box model.
    idnlgrey/setinit    - Set initial states of a nonlinear grey-box model.
    idnlgrey/getpar     - Get parameters of a nonlinear grey-box model.
    idnlgrey/setpar     - Set parameters of a nonlinear grey-box model.
    getpvec             - Get a vectorized list of model parameters.
    idnlarx/getDelayInfo - Get maximum delay in each input-output channel.

  Bookkeeping and display facilities.
    present          - Detailed display of identified models.
    get, set         - Getting and setting the model and data object properties.
    getpvec,setpvec  - Get and set vectorized list of model parameters.

  Demonstrations.
    Type "demo toolbox system" for a list of featured examples.

    System Identification Toolbox Documentation
```

The Help Browser

More extensive information about MATLAB and its commands is available through the **Help Browser** using a web based interface. This can be accessed in a number of ways

- You can left-click on the ? symbol in the small circle in the top right-hand corner of the Live Editor window or the main MATLAB window. This will open a new window like the one below



- If you are looking for more specific information on a particular MATLAB command or toolbox, the `doc` command is the most efficient way to browse the MATLAB documentation:

```
doc sqrt
```

MATLAB Scripts and Interactive Live Scripts

A **MATLAB script** is simply a plain text file containing MATLAB commands. It is called also an **M-file**, because the file has extension .m

Using M-files allows you to

- store command for later use;
- build up larger and more complicated programs;
- use the MATLAB editor to debug and then optimize your program.

An M-file is executed (it means all the commands within the M-file are executed) simply by typing the name of the file (without the .m extension) in the Command Window.

The MATLAB Editor

MATLAB provides an editor, which allows you to easily create and edit M-files. Beside the basic editing functionalities, the MATLAB Editor has some features useful specifically for writing programs:

Syntax highlighting:

- **Comments** (anything beginning with a %) is coloured green. It it very important to use comments to document your M-files, just as MATLAB does.
- MATLAB **keywords** are coloured blue. You should not try to change a keyword.
- **Smart indenting** can be used so the structure of your programs becomes clearer.

Debugging:

- **Breakpoints** can be set on any line of an M-file so you can check the values of your variables and more easily find any mistakes.
- **Profiling** can be used to find where your program is spending most of its time, allowing you to produce more efficient code.

MATLAB Script: an Example

The following command opens the existing M-file named `simple_Mscript_example.m` in the MATLAB Editor window

```
edit simple_Mscript_example
```

Do not worry about the meaning of the MATLAB commands for now, but you should try executing the M-file by typing `simple_Mscript_example` in the MATLAB Command Window.

MATLAB Interactive Live Scripts

MATLAB **live scripts** are interactive documents that combine MATLAB code with formatted text, equations and images in a single environment. In addition, live scripts store and display the output together with the code that creates it (from the MATLAB documentation). More details and examples of live scripts are available in the MATLAB documentation.

The actual document is a MATLAB Live Script.

MATLAB Live Script: an Example

This is what the live script that corresponds to the M-file seen above looks like

```
open simple_LiveScript_example
```

Summary

Using this live script you have:

- seen how to start MATLAB and its various windows;
- learnt how to type a command in the MATLAB command window;
- learnt how to use the arrow keys to go back to a previous command and run it again;
- learnt how to get help within MATLAB using the `help` command, and how to access the MATLAB help browser using the `doc` command;
- learnt that M-files are just text files with an extension .m containing MATLAB commands;
- learnt that M-files are executed by typing the name of the file (without the .m extension) in the MATLAB command window;
- learnt how to open the MATLAB Editor and open and save an M-file;
- learnt the difference between a simle M-file and a MATLAB Live Script;
- learnt how to open the MATLAB Live Editor and open and save a Live Script.

Back to the Index

Use this link to go back to the main live script of the collection.

Go to the Next Part: Numbers, Arithmetic Operations, Formats and Variables

Use this link to go to the next live script of the collection.