

MATLAB[®] Introduction

Lorenzo Nosedà

Institute of Mechanical Engineering, EPFL

Course Information

- Office and Email
 - MED 3 2715 and lorenzo.nosedo@epfl.ch
- Getting started with MATLAB[®]:
 - Official user guide: https://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf
 - <https://matlabacademy.mathworks.com/> (Recommended Tutorial)
 - <https://ubcmatlabguide.github.io/>

Lecture Overview

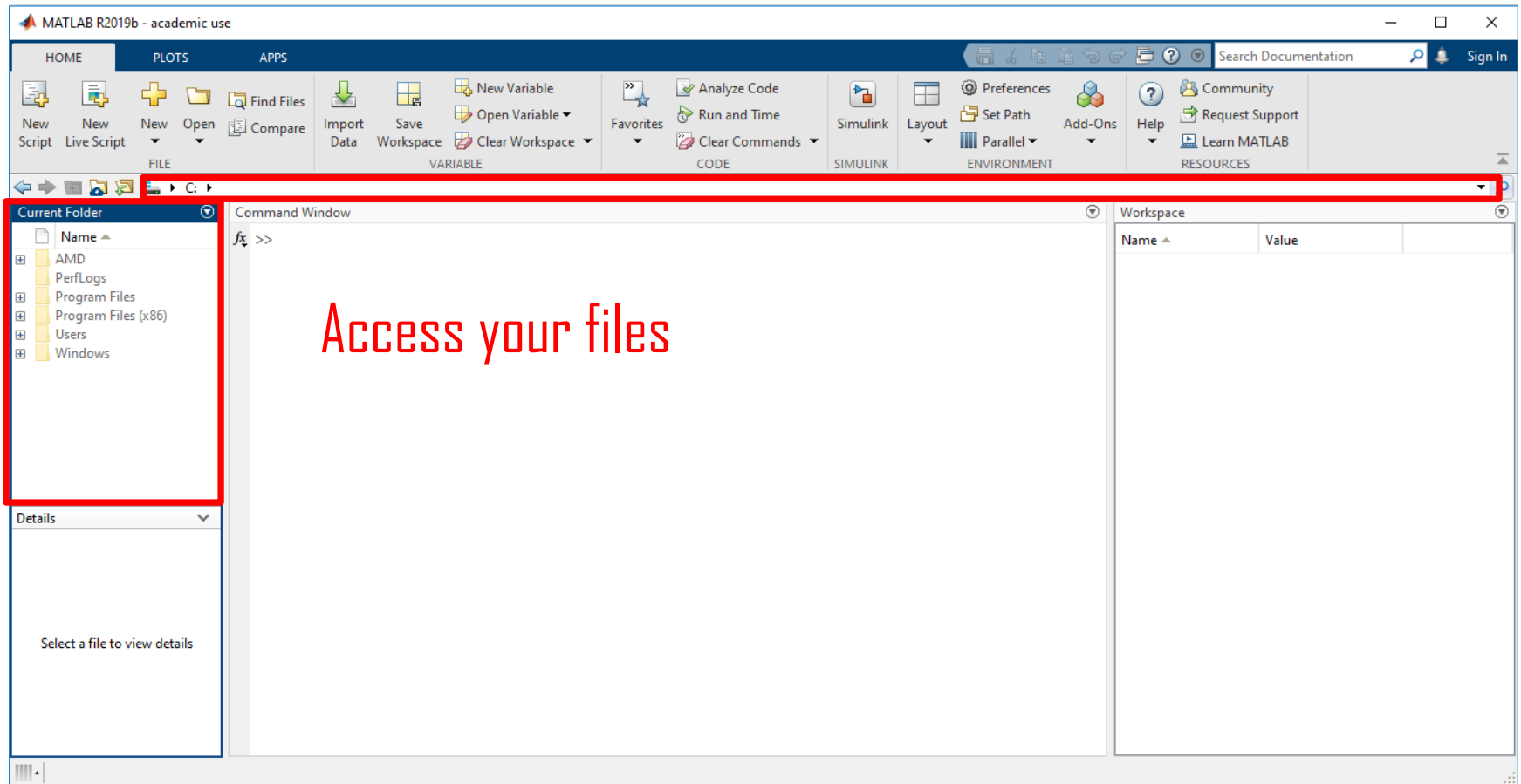
- What is MATLAB[®]
- MATLAB[®] default Layout, help and documentation
- Basic MATLAB[®] Commands and operations
- Basic MATLAB[®] functions + defining functions
- Visualizing Data
- Dynamical system simulation
 - Implementation of ode45 functions
 - Data visualization

What is MATLAB[®]

- Programming environment and a high-level language.
- Suitable for numerical computations, especially computations involving Matrix operations and linear algebra.
- Excellent support for data visualization.
- Comprises multiple toolkits, e.g., Optimization, Signal Processing, Image Processing, System Identification and much more.

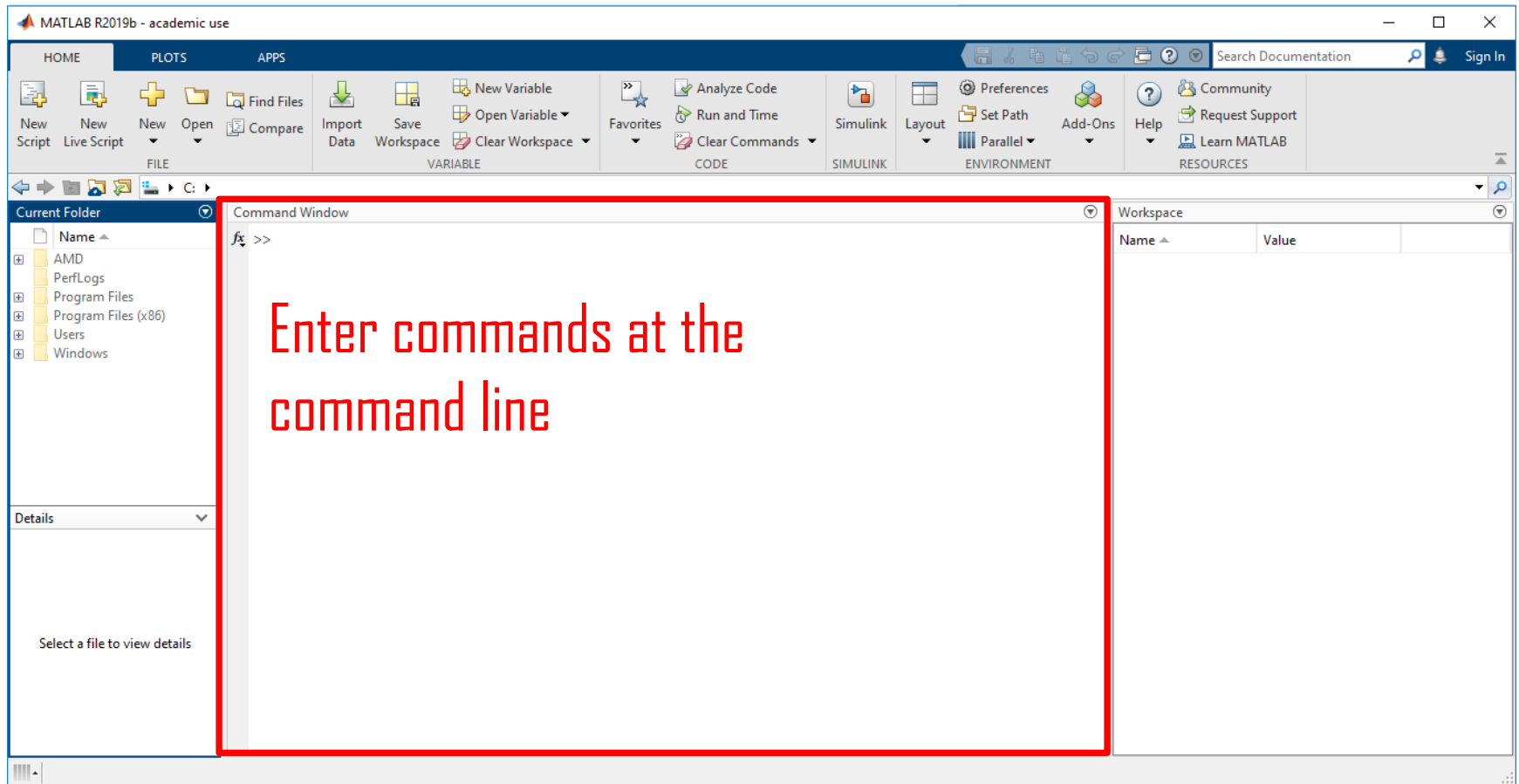
MATLAB[®] Layout

- Default layout



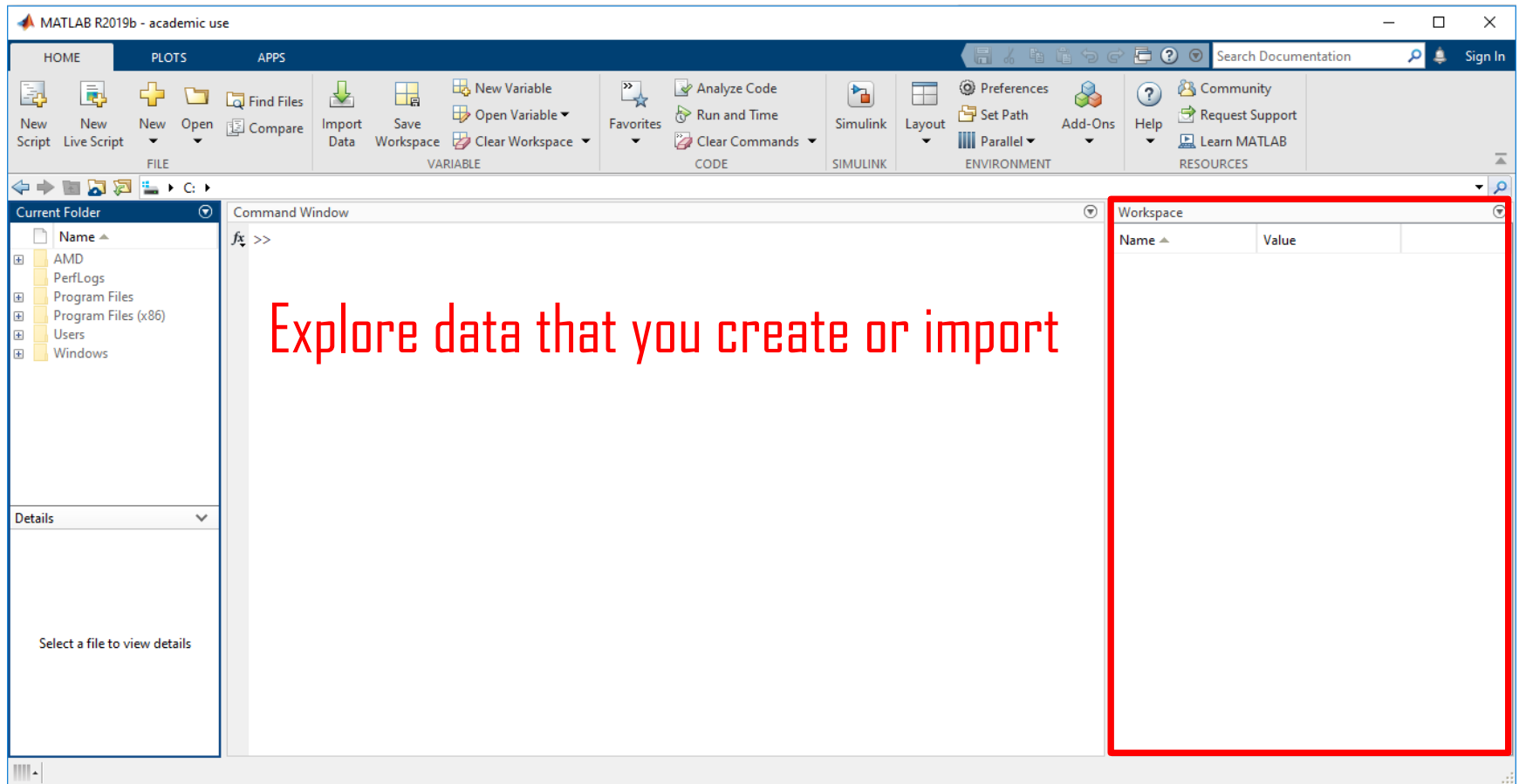
MATLAB[®] Layout

- Default layout



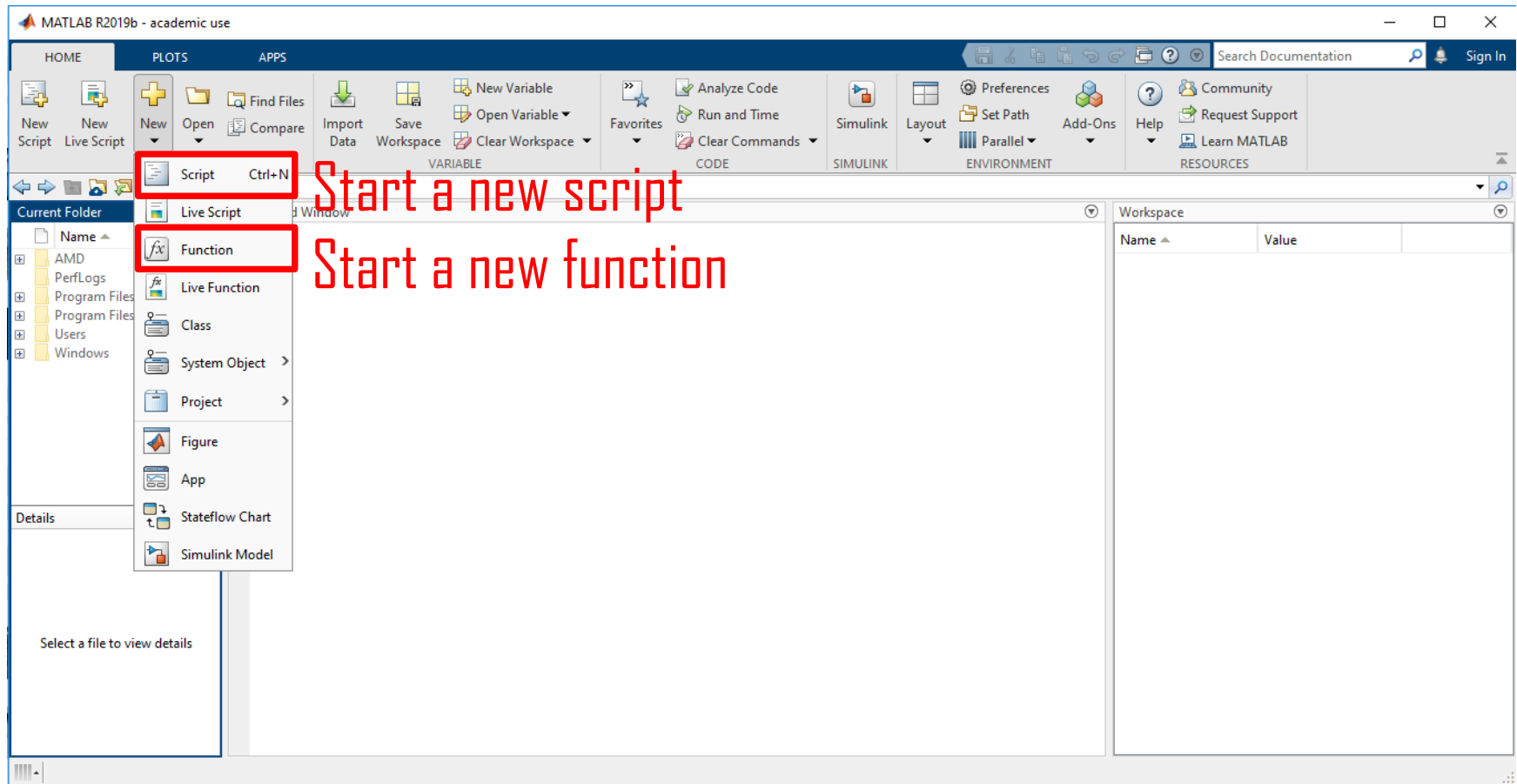
MATLAB[®] Layout

- Default layout



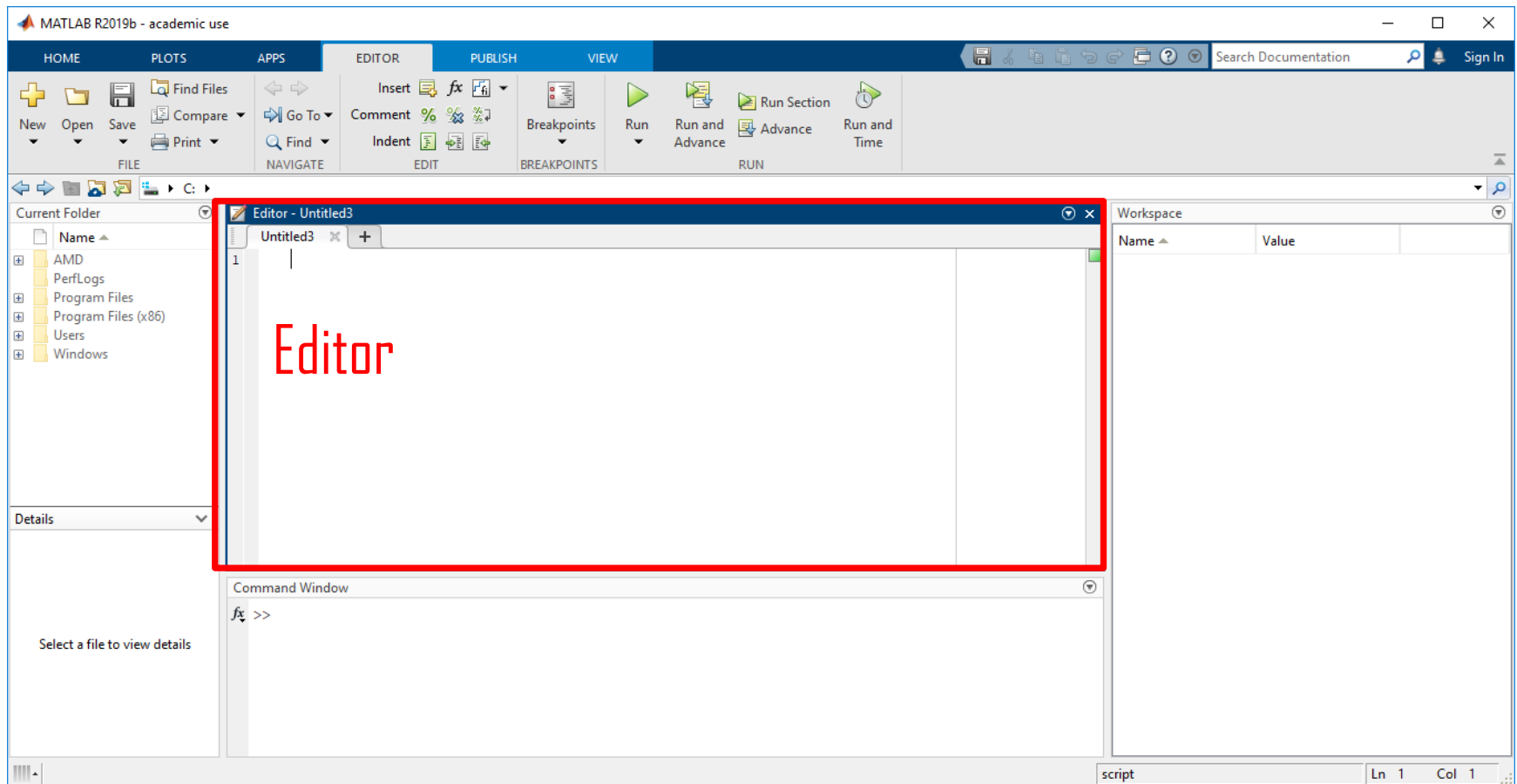
Matlab Layout

- Default layout

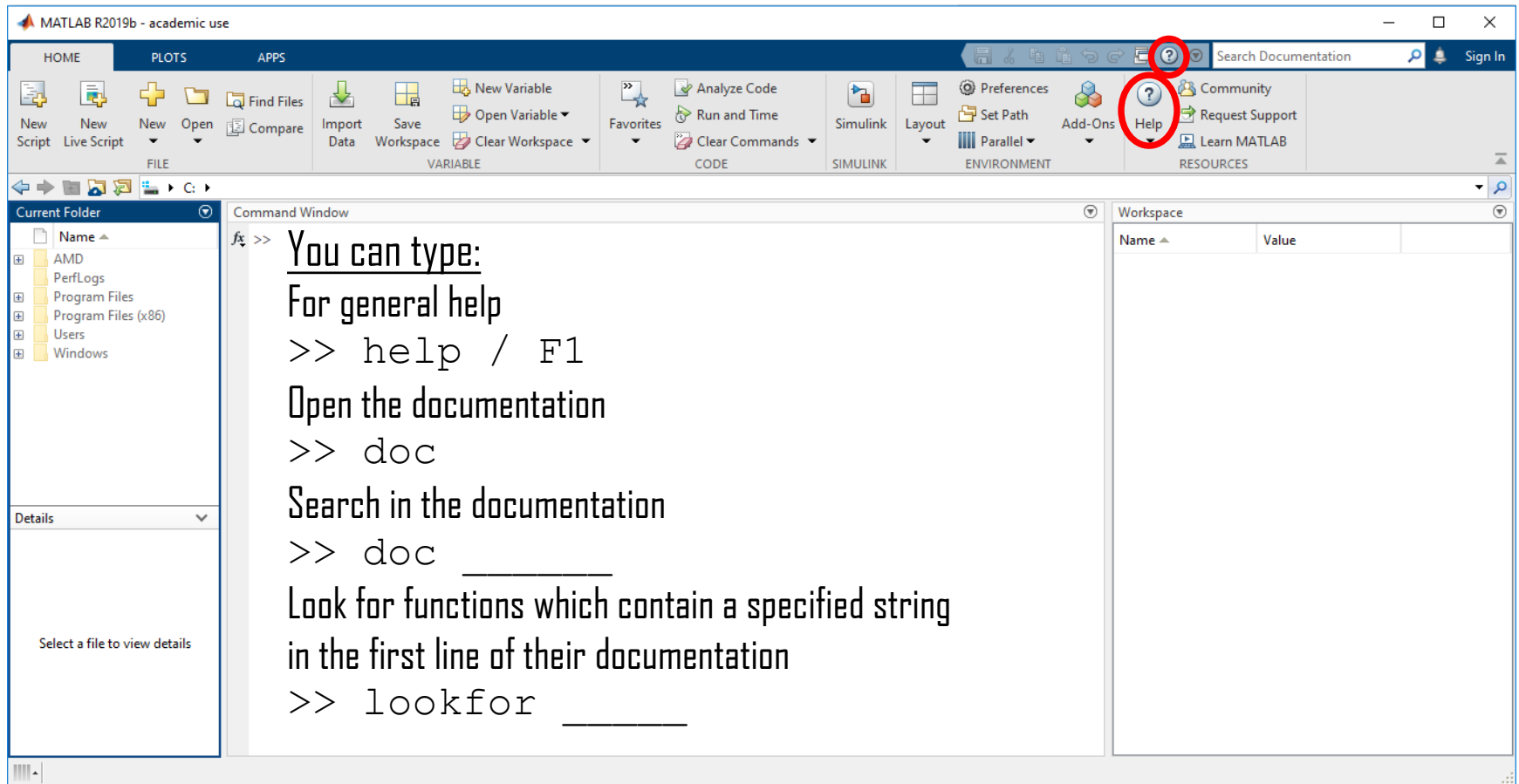


MATLAB[®] Layout

- Default layout



Help and Documentation



Basic MATLAB[®] Commands and operations (Command line / Editor)

- MATLAB[®] supports Arrays comprising real, imaginary and complex numbers.
- Mathematical operations can be used elementwise or using Tensor algebra, e.g.:

Command Window

```
>> A = [1 2; 3+1i 7.5] Defining a matrix
```

```
A =
```

```
1.0000 + 0.0000i    2.0000 + 0.0000i  
3.0000 + 1.0000i    7.5000 + 0.0000i
```

```
>> B = [6 25; 7-9j -1i]; Defining a matrix  
>> A.^2 Suppressing the output using  
a semicolon ";"
```

```
ans =
```

```
1.0000 + 0.0000i    4.0000 + 0.0000i  
8.0000 + 6.0000i    56.2500 + 0.0000i
```

```
>> A^2
```

```
ans =
```

```
7.0000 + 2.0000i    17.0000 + 0.0000i  
25.5000 + 8.5000i    62.2500 + 2.0000i
```

```
>> A*B
```

```
ans =
```

```
20.0000 -18.0000i    25.0000 - 2.0000i  
70.5000 -61.5000i    75.0000 +17.5000i
```

```
>> A.*B
```

```
ans =
```

```
6.0000 + 0.0000i    50.0000 + 0.0000i  
30.0000 -20.0000i    0.0000 - 7.5000i
```

Basic MATLAB[®] Commands and operations (Command line / Editor)

- MATLAB[®] supports Arrays comprising real, imaginary and complex numbers.
- Mathematical operations can be used elementwise or using Tensor algebra, e.g.,

Command Window

```
>> A = [1 2; 3+1i 7.5]
```

```
A =
```

```
1.0000 + 0.0000i    2.0000 + 0.0000i  
3.0000 + 1.0000i    7.5000 + 0.0000i
```

```
>> B = [6 25; 7-9j -1i];
```

```
>> A.^2
```

```
ans =
```

```
1.0000 + 0.0000i    4.0000 + 0.0000i  
8.0000 + 6.0000i    56.2500 + 0.0000i
```

```
>> A^2
```

```
ans =
```

```
7.0000 + 2.0000i    17.0000 + 0.0000i  
25.5000 + 8.5000i    62.2500 + 2.0000i
```

$$A.^2 = \begin{pmatrix} a_{11}^2 & a_{12}^2 \\ a_{21}^2 & a_{22}^2 \end{pmatrix}$$

$$A^2 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} a_{11}a_{11} + a_{12}a_{21} & a_{11}a_{12} + a_{12}a_{22} \\ a_{21}a_{11} + a_{22}a_{21} & a_{21}a_{12} + a_{22}a_{22} \end{pmatrix}$$

```
>> A*B
```

```
ans =
```

```
20.0000 -18.0000i    25.0000 - 2.0000i  
70.5000 -61.5000i    75.0000 +17.5000i
```

```
>> A.*B
```

```
ans =
```

```
6.0000 + 0.0000i    50.0000 + 0.0000i  
30.0000 -20.0000i    0.0000 - 7.5000i
```

Basic MATLAB[®] Commands and operations (Command / Editor)

- MATLAB[®] supports Arrays comprising real, imaginary and complex numbers.
- Mathematical operations can be used elementwise or using Tensor algebra, e.g.,

Command Window

```
>> A = [1 2; 3+1i 7.5]
```

```
A =
```

```
1.0000 + 0.0000i    2.0000 + 0.0000i  
3.0000 + 1.0000i    7.5000 + 0.0000i
```

```
>> B = [6 25; 7-9j -1i];
```

```
>> A.^2
```

```
ans =
```

```
1.0000 + 0.0000i    4.0000 + 0.0000i  
8.0000 + 6.0000i    56.2500 + 0.0000i
```

```
>> A^2
```

```
ans =
```

```
7.0000 + 2.0000i    17.0000 + 0.0000i  
25.5000 + 8.5000i    62.2500 + 2.0000i
```

```
>> A*B  $A * B = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$ 
```

```
ans =
```

```
20.0000 -18.0000i    25.0000 - 2.0000i  
70.5000 -61.5000i    75.0000 +17.5000i
```

```
>> A.*B  $A .* B = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix}$ 
```

```
ans =
```

```
6.0000 + 0.0000i    50.0000 + 0.0000i  
30.0000 -20.0000i    0.0000 - 7.5000i
```

Basic MATLAB[®] Commands and operations (Command / Editor)

- MATLAB[®] supports Arrays comprising real, imaginary, and complex numbers.
- Mathematical operations can be used elementwise or using Tensor algebra, e.g.,
- **Array indices in MATLAB[®] start with 1, not 0 as in most programming languages.**

```
C =
```

```
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
>> C(1,3)
```

```
ans =
```

```
    3
```

```
>> C(:,end)
```

```
ans =
```

```
    13
     8
    12
     1
```

```
>> C([1 4],[2 3])
```

```
ans =
```

```
     2     3
    14    15
```

```
>> C([1 4],2:4)
```

```
ans =
```

```
     2     3    13
    14    15     1
```

	1	2	3	4/end
1	16	2	3	13
2	5	11	10	8
3	9	7	6	12
4/end	4	14	15	1

Basic MATLAB[®] Commands and operations (Command / Editor)

- MATLAB[®] supports Tensors and Arrays comprising real, imaginary and complex numbers.
- Mathematical operations can be used elementwise or using Tensor algebra, e.g.,
- Array (matrix) indices in MATLAB[®] starts with 1 , not 0 as in most programming languages.
- Allowed Names of variables/ functions/ files:
 - Names consist of letters, followed by any number of letters, digits, or underscores. MATLAB[®] is case-sensitive; it distinguishes between uppercase and lowercase letters. `A` and `a` are not the same variable.
 - Stored function names can be overridden; however, it is advised to avoid the latter.

Basic MATLAB[®] Commands and operations (Command / Editor)

- Generate a vector:

```
>> a = 1:10
```

1 to 10, spacing: 1

```
a =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
>> a = 1:2:10
```

1 to 10, spacing: 2

```
a =
```

```
1 3 5 7 9
```

```
>> a = linspace(1,10,10)
```

1 to 10, 10 elements

```
a =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
>> a = linspace(1,10,4)
```

1 to 10, 4 elements

```
a =
```

```
1 4 7 10
```

It also possible to
generate logarithmic
spacing using
Logspace

Vectors of zeroes or
ones using
zeros
ones

Basic MATLAB® Commands and operations (Command / Editor)

- Transpose, Hermitian transpose and matrix manipulations

```
C =
```

```
8.0000 +17.0000i    1.0000 +24.0000i    6.0000 + 1.0000i  
3.0000 +23.0000i    5.0000 + 5.0000i    7.0000 + 7.0000i  
4.0000 + 4.0000i    9.0000 + 6.0000i    2.0000 +13.0000i
```

```
>> C'    Hermitian transpose = Transpose + complex conjugate
```

```
ans =
```

```
8.0000 -17.0000i    3.0000 -23.0000i    4.0000 - 4.0000i  
1.0000 -24.0000i    5.0000 - 5.0000i    9.0000 - 6.0000i  
6.0000 - 1.0000i    7.0000 - 7.0000i    2.0000 -13.0000i
```

```
>> C.'
```

```
ans =    Transpose
```

```
8.0000 +17.0000i    3.0000 +23.0000i    4.0000 + 4.0000i  
1.0000 +24.0000i    5.0000 + 5.0000i    9.0000 + 6.0000i  
6.0000 + 1.0000i    7.0000 + 7.0000i    2.0000 +13.0000i
```

Basic MATLAB[®] Commands and operations (Command / Editor)

- Transpose, Hermitian transpose and matrix manipulations

```
>> flipud(C)      Flip up down      C =  
  
ans =  
  
      8.0000 +17.0000i      1.0000 +24.0000i      6.0000 + 1.0000i  
      3.0000 +23.0000i      5.0000 + 5.0000i      7.0000 + 7.0000i  
      4.0000 + 4.0000i      9.0000 + 6.0000i      2.0000 +13.0000i  
      3.0000 +23.0000i      5.0000 + 5.0000i      7.0000 + 7.0000i  
      8.0000 +17.0000i      1.0000 +24.0000i      6.0000 + 1.0000i  
  
>> fliplr(C)      Flip left right  
  
ans =  
  
      6.0000 + 1.0000i      1.0000 +24.0000i      8.0000 +17.0000i  
      7.0000 + 7.0000i      5.0000 + 5.0000i      3.0000 +23.0000i  
      2.0000 +13.0000i      9.0000 + 6.0000i      4.0000 + 4.0000i  
  
>> rot90(C,3)      Rotate by 3*90 deg CCW  
  
ans =  
  
      4.0000 + 4.0000i      3.0000 +23.0000i      8.0000 +17.0000i  
      9.0000 + 6.0000i      5.0000 + 5.0000i      1.0000 +24.0000i  
      2.0000 +13.0000i      7.0000 + 7.0000i      6.0000 + 1.0000i
```

Functions

- Basic useful functions
 - `clc` - clears the command window
 - `clear` - clears the workspace (i.e., delete all variables and stored data)
 - `close all` - close all Figures
 - `save _____` - saves the workspace as : `____.mat`
 - `load _____` - loads a saved workspace or a data file
- For more information on a function, e.g.,
`>> doc linspace`

Functions

- Anonymous Functions

"An anonymous function is a function that is not stored in a program file, but is associated with a variable whose data type is `function_handle`. Anonymous functions can accept inputs and return outputs, just as standard functions do. However, they can contain only a single executable statement."

- Syntax:

```
function_handle = @(input1, input2, ...) expression;
```

- `function_handle`: Variable that holds the reference to the anonymous function.
- `@(input1, input2, ...)`: Defines the input parameters to the anonymous function.
- `expression`: Mathematical expression or operation that the function performs.

Functions

For example, single variable:

```
a = 1.3;
b = .2;
c = 30;
parabola = @(x) a*x.^2 + b*x + c;
>> parabola(1)
ans = 31.5000
```

For example, multiple variables (Pay attention to elementwise vs. matrix algebra):

```
x = 1:3;
y = 10:12;
myFuncEW = @(x,y) a*x.^2 + b*x.*y + c*y;
myFuncMA = @(x,y) a*(x*y).^2 + b*(x*y) + c;

>> myFuncEW(x,y)
ans = 303.3000  339.6000  378.9000

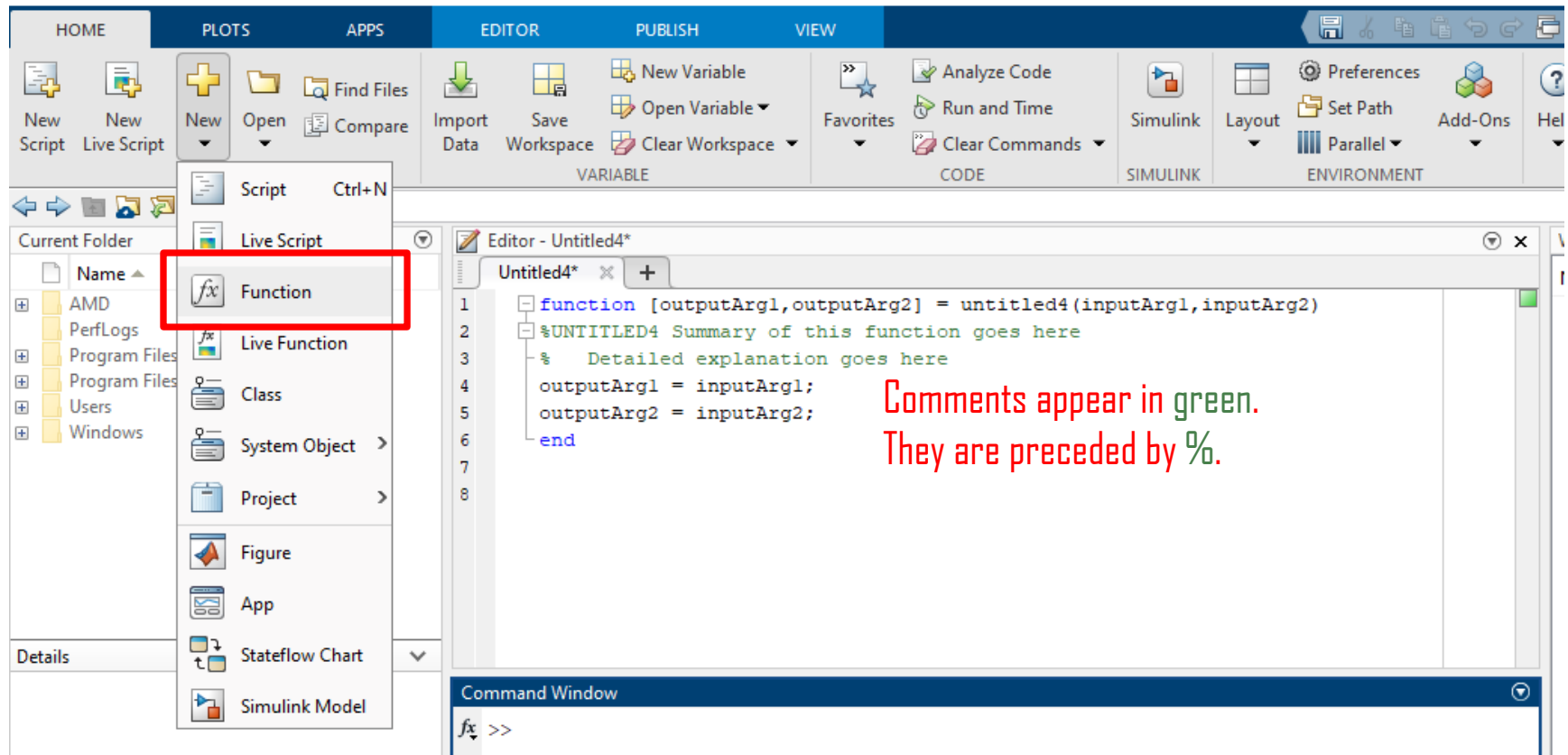
>> myFuncMA(x.',y)
ans =
1.0e+03 *

    0.1620    0.1895    0.2196
    0.5540    0.6636    0.7836
    1.2060    1.4523    1.7220
```

Functions

- Function

MATLAB R2019b - academic use



Functions

- Function

The image displays the MATLAB environment with three main components highlighted by red boxes:

- Current Folder:** Shows the file `myFunc2.m`.
- Editor:** Contains the function definition for `myFunc2`. The function signature `function [eq1,eq2] = myFunc2(x,y)` is highlighted. The code defines a function that takes inputs `x` and `y` and returns `eq1` and `eq2`. The calculations are `eq1 = (x+y).^2;`, `TEMP = (x-y).^2;`, and `eq2 = eq1.^2-TEMP;`.
- Command Window:** Shows the execution of the function: `>> myFunc2(1:2,8:9)`. The output is `ans =` followed by the values `81` and `121`.
- Workspace:** Shows the variable `ans` with the value `[81,121]`.

Visualizing data

- There are many built in MATLAB[®] functions for data visualization

The screenshot shows the MATLAB Plot Catalog window. On the left is a list of plot categories including Line Plots, Stem and Stair Plots, Bar Plots, Scatter Plots, Graph Plots, Pie Charts, Histograms, Polar Plots, Geographic Plots, Contour Plots, Image Plots, 3-D Surfaces, Volumetrics, Vector Fields, Analytic Plots, Control Toolbox Plots, Curve Fitting Toolbox Plots, DSP System Toolbox Plots, Image Processing Toolbox Plots, Mapping Toolbox: Projected X-Y Plots, Mapping Toolbox: Geographic Data, Signal Processing Toolbox: Filter Ar, Signal Processing Toolbox: Window, Signal Processing Toolbox: Spectral, Statistics And Machine Learning To, System Id: Parametric Model Evalua, System Id: Non-Parametric Analysis, System Id: IDLTI Model Visualizatio, and System Id: IDLTI Model Simulation. The main panel displays the 'plot' function, which is a 2-D line graph using linear axes. It shows a preview of a sine wave plot. Below the preview, the function signature 'plot(xx,yy)' is shown with a 'x++y' icon. The description states: '2-D line graph using linear axes'. The 'Plot as multiple series' section explains: 'Plots each series on the same plot' and 'Plot as multiple series vs. first input' and 'Plots the second and later series against the fi...'. The 'plotyy(xx,yy)' function is also listed with the description: 'Graphs with y tick labels on the left and right'. The 'semilogx(xx,yy)' function is listed with the description: 'Semi-log scale plot'. The 'semilogy(xx,yy)' function is listed with the description: 'Semi-log scale plot'. The 'loglog(xx,yy)' function is listed with the description: 'Log-log scale plot'. The 'area(xx,yy)' function is listed with the description: 'Filled area plot'. The 'errorbar(xx,yy)' function is listed with the description: 'Error bar plot'. The 'errorbar (horizontal)' function is listed with the description: 'Horizontal error bar plot'. The 'plot3(xx,yy)' function is listed with the description: '3-D line graph using linear axes'. The 'comet(xx,yy)' function is listed with the description: 'Comet-like trajectory'. The 'stackedplot(xx,yy)' function is listed with the description: 'Stacked plot'. On the right side of the window, the 'Plotted Variables' field shows 'xx, yy'. Below this, the 'plot' function is highlighted in orange. The description of 'plot' is: '2-D line plot'. The 'Syntax' section lists the following syntaxes: '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)', and 'plot(ax, __)'. The 'Description' section explains: 'plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.' and lists three bullet points: '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.', and 'If one of X or Y is a vector and the other...'. At the bottom right, there are three buttons: 'Plot', 'Plot in New Figure', and 'Close'.

Plot Catalog

Line Plots
Stem and Stair Plots
Bar Plots
Scatter Plots
Graph Plots
Pie Charts
Histograms
Polar Plots
Geographic Plots
Contour Plots
Image Plots
3-D Surfaces
Volumetrics
Vector Fields
Analytic Plots
Control Toolbox Plots
Curve Fitting Toolbox Plots
DSP System Toolbox Plots
Image Processing Toolbox Plots
Mapping Toolbox: Projected X-Y Plots
Mapping Toolbox: Geographic Data
Signal Processing Toolbox: Filter Ar
Signal Processing Toolbox: Window
Signal Processing Toolbox: Spectral
Statistics And Machine Learning To
System Id: Parametric Model Evalua
System Id: Non-Parametric Analysis
System Id: IDLTI Model Visualizatio
System Id: IDLTI Model Simulation

plot(xx,yy)
2-D line graph using linear axes

Plot as multiple series
Plots each series on the same plot

Plot as multiple series vs. first input
Plots the second and later series against the fi...

plotyy(xx,yy)
Graphs with y tick labels on the left and right

semilogx(xx,yy)
Semi-log scale plot

semilogy(xx,yy)
Semi-log scale plot

loglog(xx,yy)
Log-log scale plot

area(xx,yy)
Filled area plot

errorbar(xx,yy)
Error bar plot

errorbar (horizontal)
Horizontal error bar plot

plot3(xx,yy)
3-D line graph using linear axes

comet(xx,yy)
Comet-like trajectory

stackedplot(xx,yy)
Stacked plot

Plotted Variables: xx, yy

plot
2-D line plot

Syntax

```
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( __)
```

Description

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

Plot Plot in New Figure Close

Visualizing data

- Stage 1: open a figure

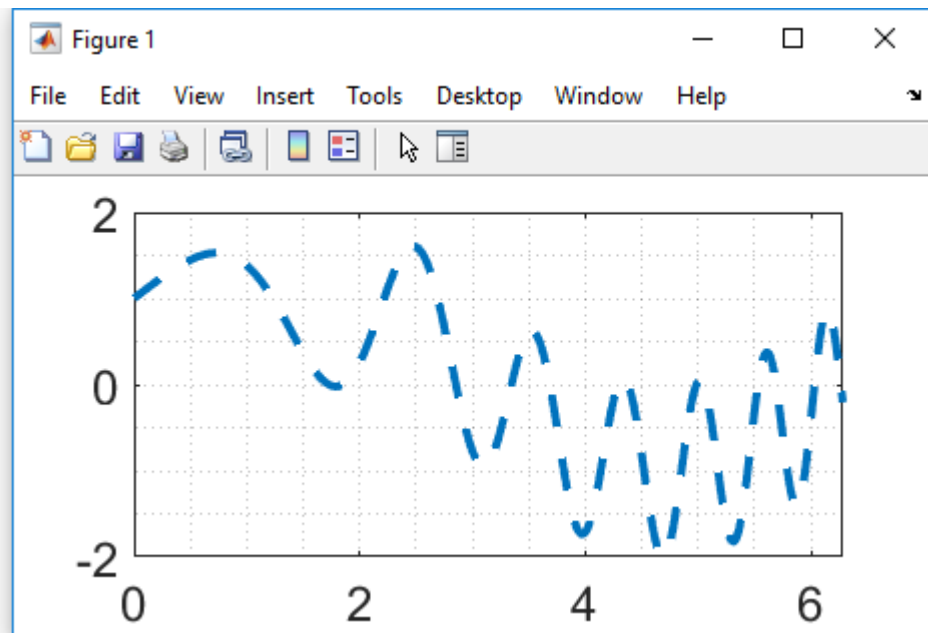
```
>> figure(1)
```

- Stage 2: plot the data (h is a plot handle)

```
>> h = plot(xx,yy)
```

- Edit the plot and axes

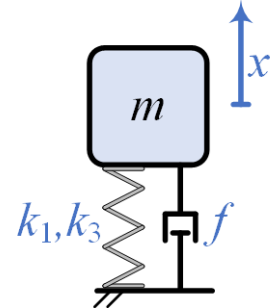
```
xx = linspace(0,2*pi,1e3);  
yy = sin(xx)+cos(xx.^2);  
  
figure(1)  
h = plot(xx,yy);  
set(h,'linewidth',3,'linestyle','--');  
grid minor  
set(gcf,'color','w')  
set(gca,'fontsize',18)
```



Dynamic simulations

- A dynamical system can be represented as an ordinary differential equation (ODE). For example a damped nonlinear spring, mass system:

$$m\ddot{x} + f\dot{x} + k_1x + k_3x^3 = 0$$



- There are various ode solvers in MATLAB[®]. Throughout the course the function `ode45`, which implements Runge-Kutta will be used.
- Compute the system's response to the initial conditions:
 - A) $x(0) = 0.001, \dot{x}(0) = 0$
 - B) $x(0) = 10, \dot{x}(0) = 0$
- For the parameters:

$$m = 1, \quad f = 0.6, \quad k_1 = 100, \quad k_3 = 5.$$

Dynamic simulations

- Stage 1: Prepare the equation for implementation in MATLAB[®]
 - MATLAB ODE solvers only solve first-order equations.
 - Re-write higher-order ODEs as an equivalent system of first-order equations using the following substitutions:

$$z_1 = x$$

$$z_2 = \dot{x}$$

$$z_3 = \ddot{x}$$

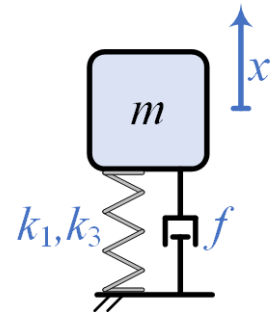
$$\vdots$$

$$z_n = x^{(n-1)}$$

- We get for our example:

$$m\ddot{x} + f\ddot{x} + k_1x + k_3x^3 = 0$$

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}, \quad \dot{\mathbf{z}} = \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ -\left(fz_2 + k_1z_1 + k_3z_1^3\right) / m \end{pmatrix}$$

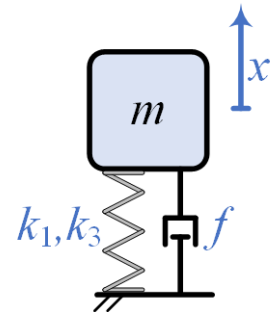


Dynamic simulations

- Stage 1: Prepare the equation for implementation in MATLAB[®]

$$m\ddot{x} + f\dot{x} + k_1x + k_3x^3 = 0$$

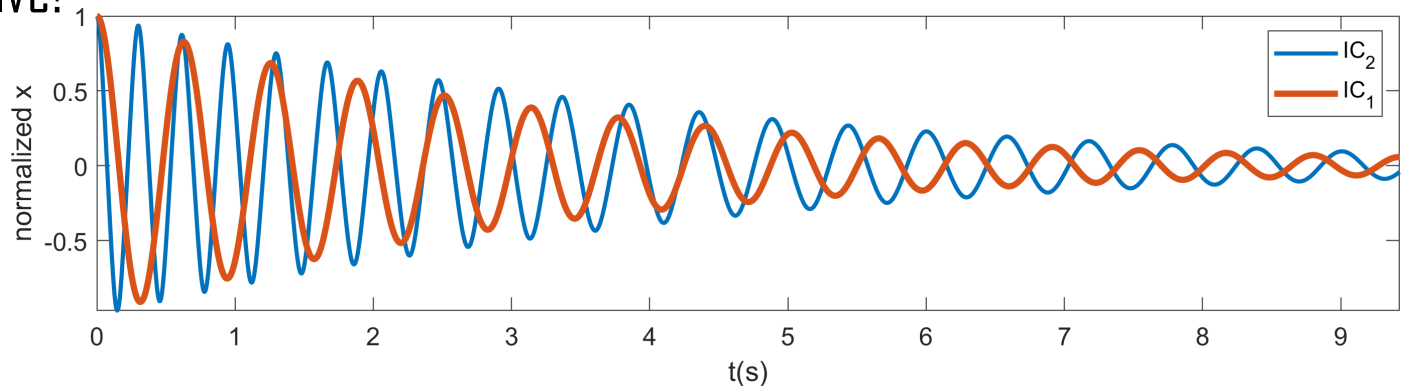
$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix}, \quad \dot{\mathbf{z}} = \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} z_2 \\ -\left(fz_2 + k_1z_1 + k_3z_1^3\right) / m \end{pmatrix}$$



- Stage 2: write a MATLAB[®] function to compute the derivatives

```
function dzdt = Damped_nonlin_sys(t,z,m,f,k1,k3)
% Damped_nonlin_sys computes the derivative of the states z
dzdt(1,1) = z(2);
dzdt(2,1) = -(f*z(2)+k1*z(1)+k3*z(1)^3)/m;
end
```

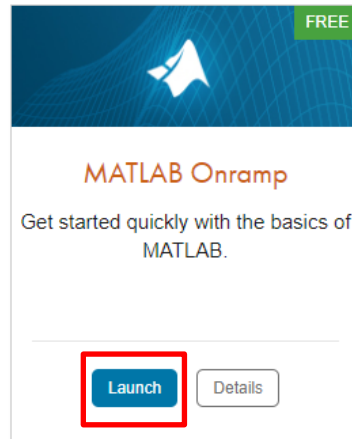
- Stage 3: Solve!



Recommended tutorial

- Complete the first MATLAB[®] training: MATLAB Onramp (approx. 2 hours)

- Go to <https://matlabacademy.mathworks.com/>
- Click: Launch



- You may be asked to create a MATLAB[®] account (it's free and useful!)