ICEA

UNIVERSITÀ DEGLI STUDI DI PADOVA

Lesson 1:
INTRODUCTION TO MATLAB

2020

Giulia Comunale & Paolo Peruzzo

giulia.comunale@dicea.unipd.it

Download the student license of Matlab:

https://www.csia.unipd.it/servizi/servizi-utenti-istituzionali/contratti-software-e-licenze/matlab

Download the student license of Matlab:

https://www.csia.unipd.it/servizi/servizi-utenti-istituzionali/contratti-software-e-licenze/matlab

Download del software e codici - Campus e Student

1- Reperire il Codice di Attivazione sul sito https://software.unipd.it/ (per accedere al sito è necessario utilizzare il proprio account personale @unipd.it oppure @studenti.unipd.it)

2- Creare un account Mathworks cliccando "Create Account" sul sito https://it.mathworks.com/login e utilizzando come e-mail address la propria casella @unipd.it oppure @studenti.unipd.it oppure @phd.unipd.it (si consiglia invece di non utilizzare la stessa password).

3- Una volta autenticati sul portale Mathworks cliccare sul nome del profilo in alto a destra, selezionare Collega licenza ed inserire il Codice di Attivazione di cui al punto 1.

4- Fare clic su "Scarica i tuoi prodotti ora" oppure accedere a mathworks.com/downloads

5- Fare clic sul pulsante "scarica" per ottenere la versione più recente.

6- Scegliere una piattaforma supportata e scaricare il programma di installazione.

Installazione del software - Campus e Student

1- Avviare il programma di installazione.

2- Nel programma di installazione, selezionare Accedi con un account MathWorks e seguire le istruzioni online.

3- Quando richiesto, selezionare la licenza Academic – Total Headcount contrassegnata con Campus oppure Student.

4- Selezionare i prodotti che si desidera scaricare e installare.

5- Dopo aver scaricato e installato i prodotti desiderati, mantenere selezionata la casella Attiva MATLAB e fare clic su Avanti.

6- Quando viene richiesto di fornire un nome utente, verificare che il nome utente visualizzato sia corretto. Continuare con la procedura fino al completamento dell'attivazione.

What is MatLab?

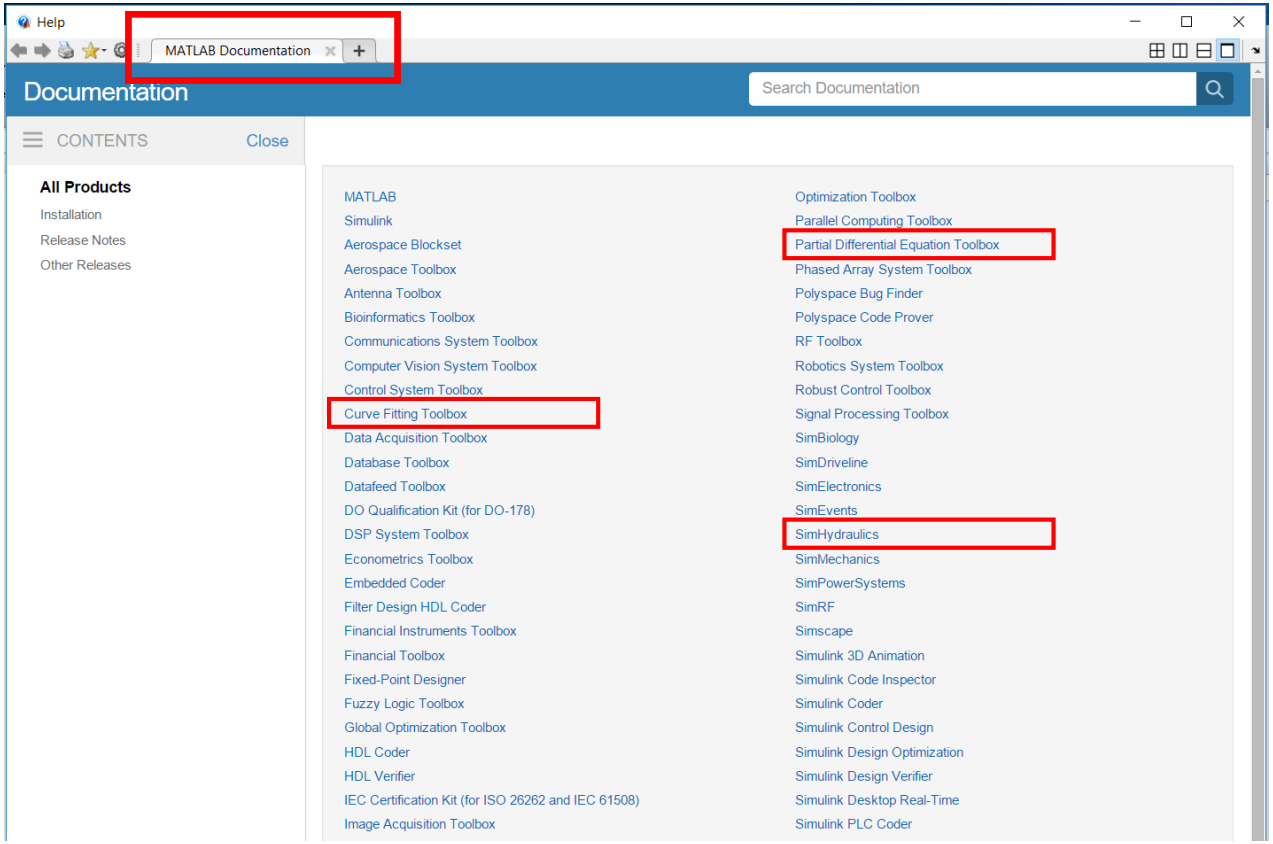
MATLAB (**MAT**rix **LAB**oratory) is a numerical computing environment.

MATLAB allows:

- Matrix manipulations;
- Plotting of functions and data;
- Implementation of algorithms;
- Interfacing with programs written in other languages, including C, C++, Java, Fortran and Python;
- Specialized toolboxes for making things easier (**Matlab has several toolboxes**);

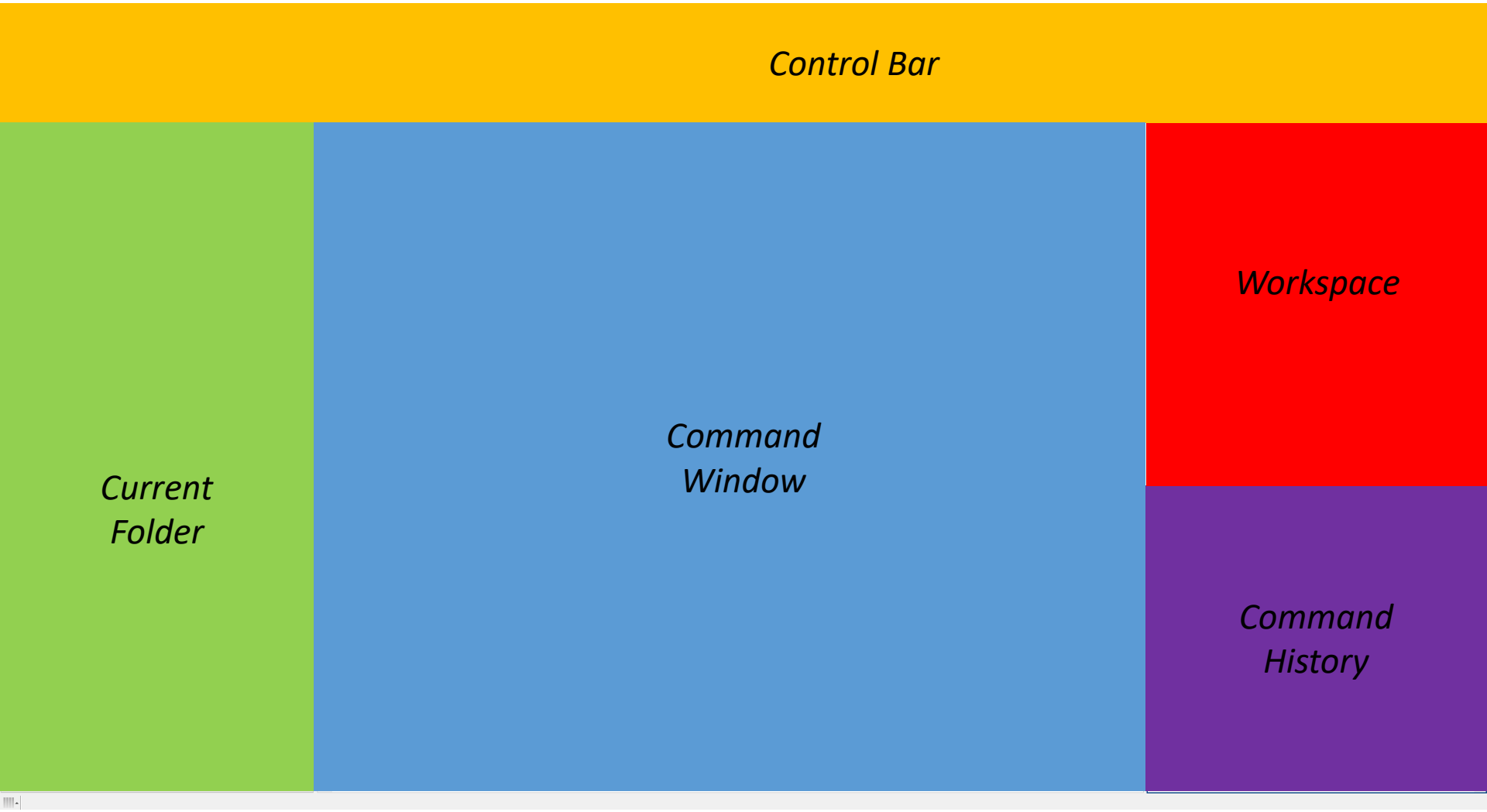
What we can do with it?

- Solve directly systems of equations with implemented functions;
- Solve Ordinary Differential Equations (ODEs);
- Solve Partial Differential Equations (PDEs);
- Elaborate data from file Text, Excel, etc.;
- Elaborate images, generate images, tables, 2D and 3D plot;
- Create functions.

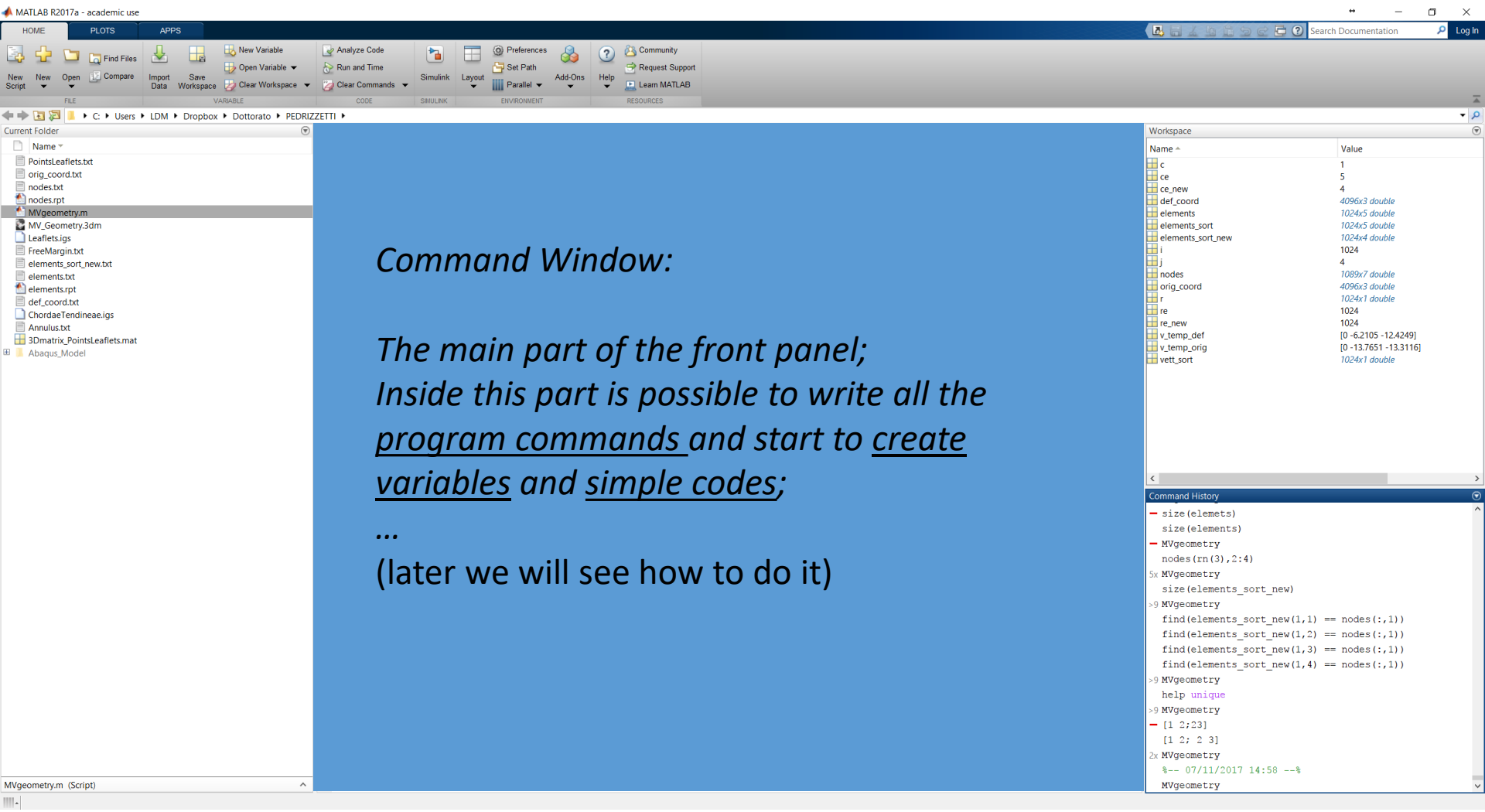


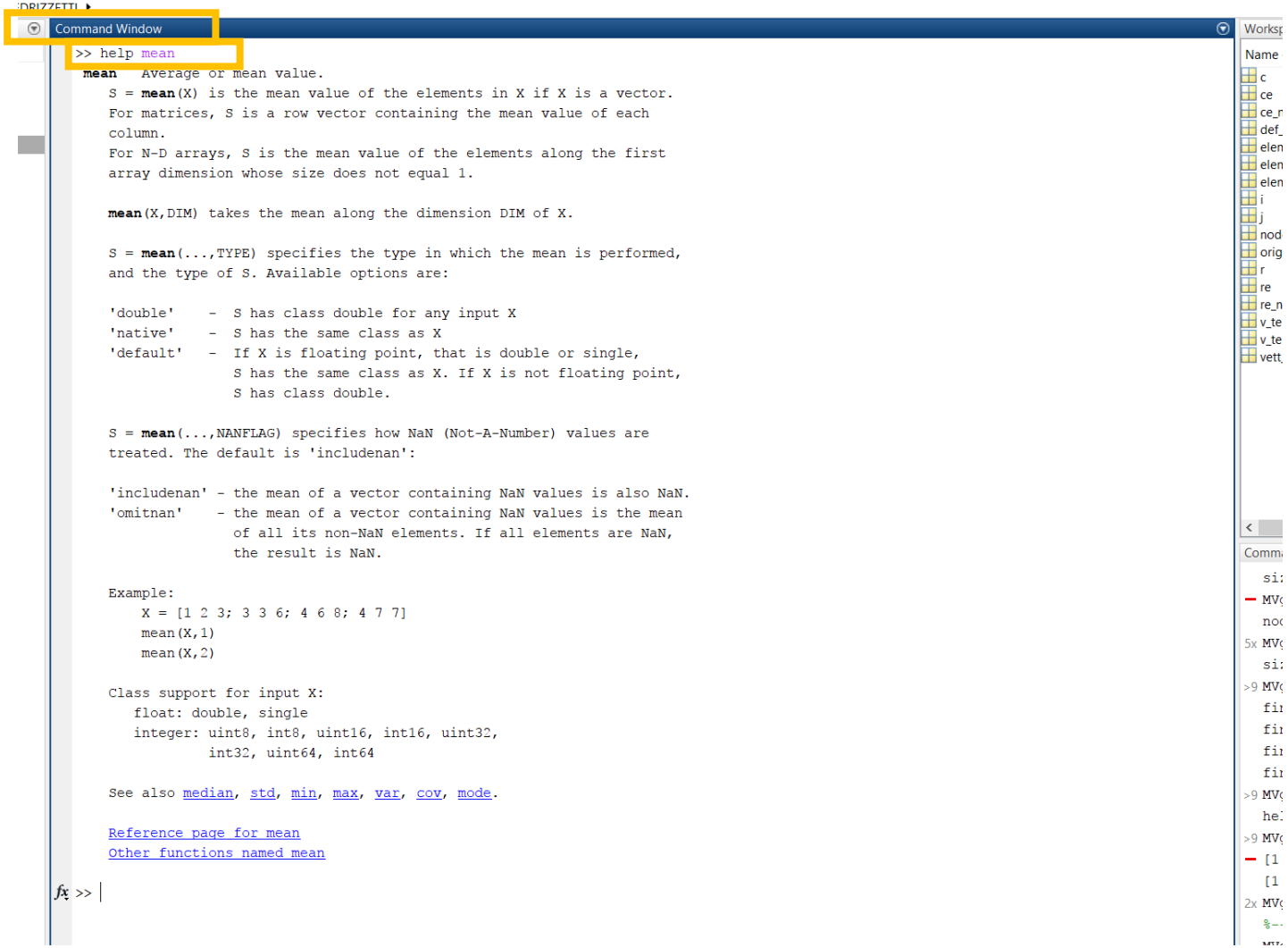
https://it.mathworks.com/help/matlab/

MatLab Screen (Front Panel)

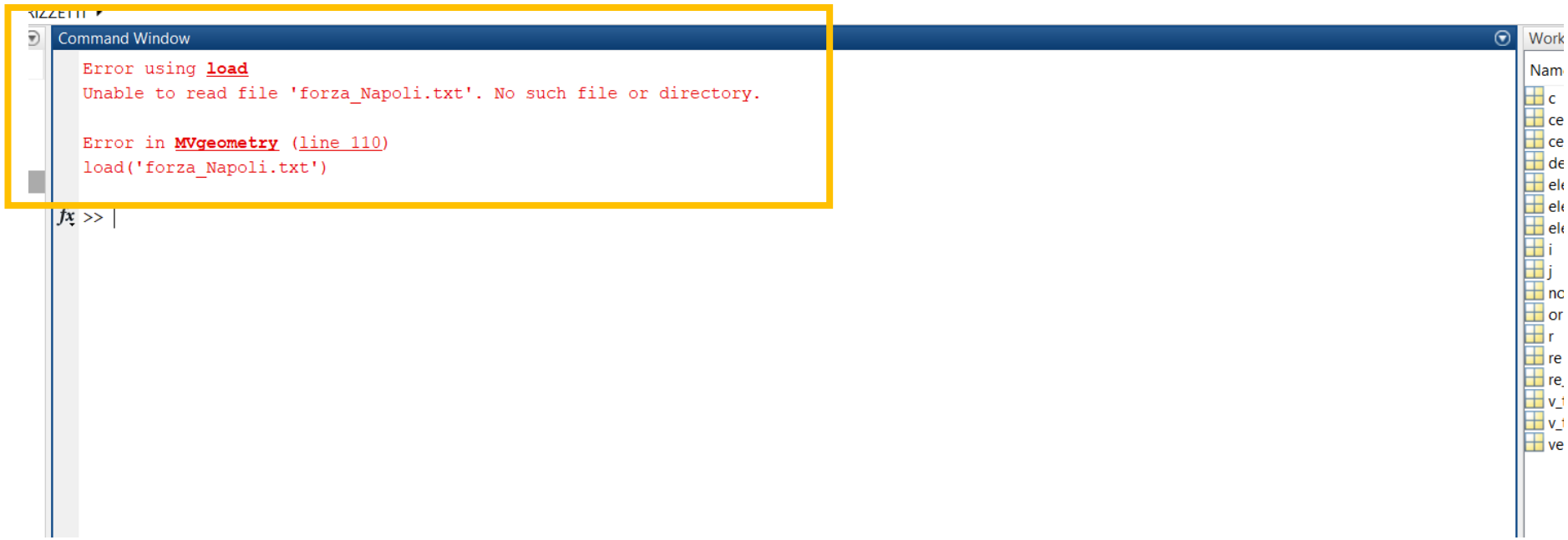


MatLab Screen (Front Panel)

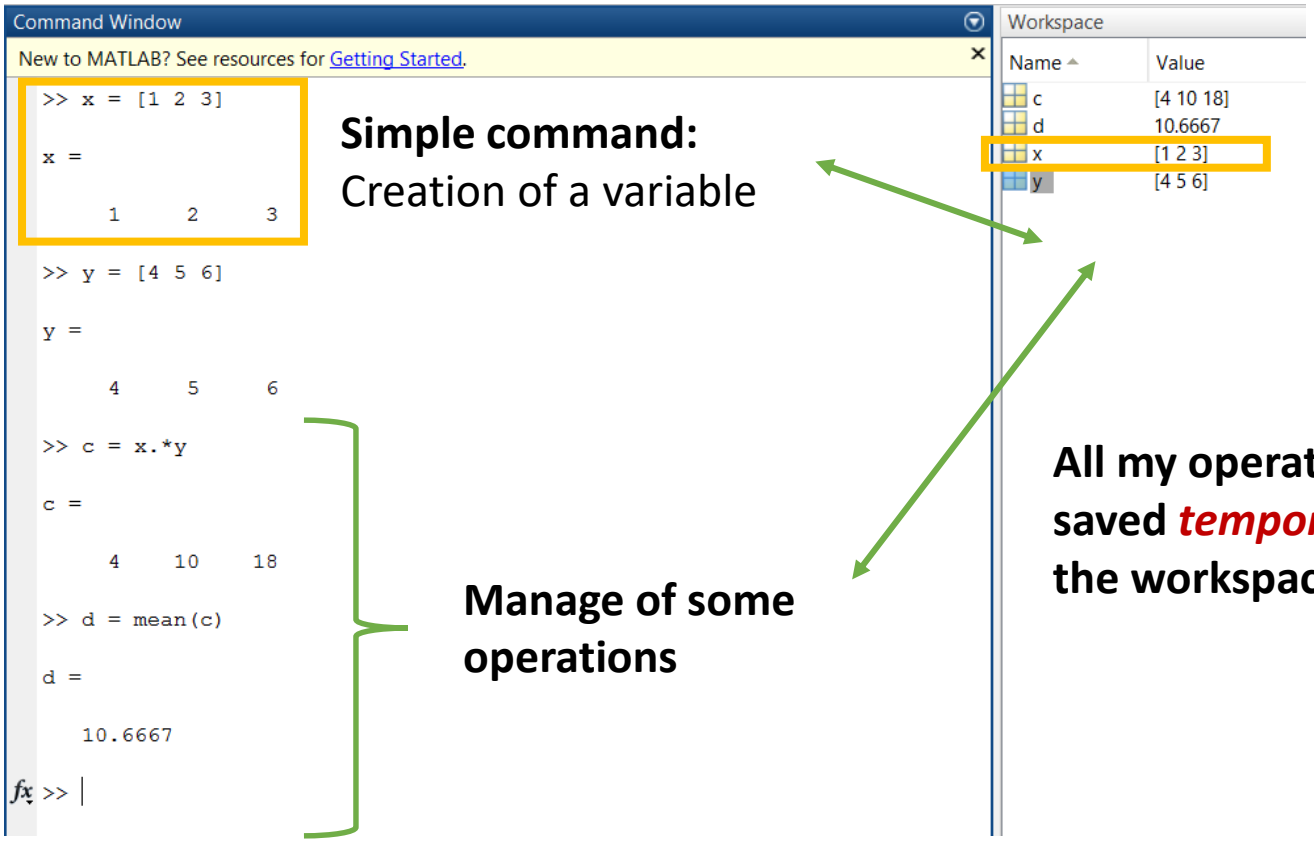



Command Window
>> help mean
mean Average or mean value.
S = mean(X) is the mean value of the elements in X if X is a vector.
For matrices, S is a row vector containing the mean value of each
column.
For N-D arrays, S is the mean value of the elements along the first
array dimension whose size does not equal 1.
mean(X,DIM) takes the mean along the dimension DIM of X.
S = mean(...,TYPE) specifies the type in which the mean is performed,
and the type of S. Available options are:
'double' - S has class double for any input X
'native' - S has the same class as X
'default' - If X is floating point, that is double or single,
S has the same class as X. If X is not floating point,
S has class double.
S = mean(...,NANFLAG) specifies how NaN (Not-A-Number) values are
treated. The default is 'includenan':
'includenan' - the mean of a vector containing NaN values is also NaN.
'omitnan' - the mean of a vector containing NaN values is the mean
of all its non-NaN elements. If all elements are NaN,
the result is NaN.
Example:
X = [1 2 3; 3 3 6; 4 6 8; 4 7 7]
mean(X,1)
mean(X,2)
Class support for input X:
float: double, single
integer: uint8, int8, uint16, int16, uint32,
int32, uint64, int64
See also median, std, min, max, var, cov, mode.
Reference page for mean
Other functions named mean
fx >>
Name
c
ce
ce_r
def_
elen
elen
elen
i
j
nod
orig
r
re
re_n
v_te
v_te
vett

MatLab Screen (Front Panel)



MatLab Screen (Front Panel)



Simple command:
Creation of a variable

Manage of some operations

All my operators are saved *temporarily* in the workspace

Variables: Vectors/Matrix

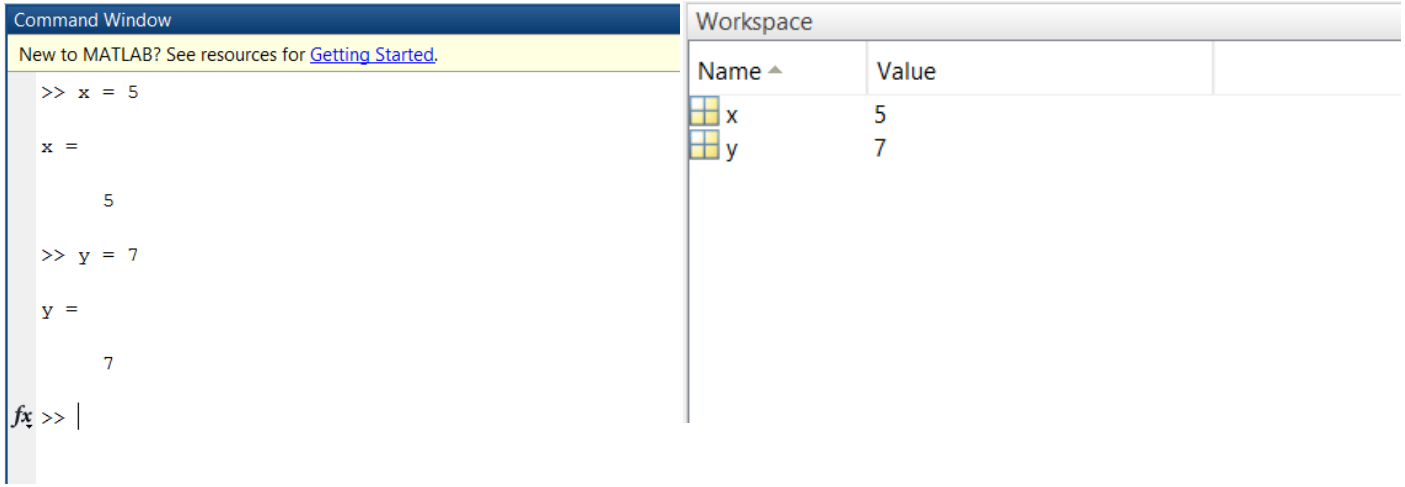
All variables are created with double precision unless specified and they are matrices.

- Example:

>> **x = 5**;

>> **y = 7**;

After these statements, the variables are 1x1 matrices with double precision



Variables: Vectors/Matrix

- **a Row Vector (matrix [1x4])**

x = [1 2 5 1]

- **a Column Vector (matrix [4x1])**

x = [1;2;5;1]

- **a Matrix [3x3]**

A = [1 2 3; 5 1 4; 3 2 -1]

- **Transpose**

a = [1 2 5 1] **[1x4]**

b = a’ **[4x1]**

```
Command Window
>> x = [1 2 5 1]

x =

     1     2     5     1

>> x = [1;2;5;2]

x =

     1
     2
     5
     2

>> A = [1 2 3;5 1 4;3 2 -1]

A =

     1     2     3
     5     1     4
     3     2    -1

>> a = [1 2 5 2]

a =

     1     2     5     2

>> b = a'

b =

     1
     2
     5
     2

fx >>
```

Variables: Vectors/Matrix

- **t =1:10** → all the values spaced 1 (default)
- **k =2:-0.5:-1** → change in spacing
- **B = [1:4; 5:8]** → build a [2x4] matrix
- **zeros**(M,N) [MxN] matrix of zeros

  x = zeros(1,3)
- **ones**(M,N) [MxN] matrix of ones

  y = ones(1,3)

```
Command Window
>> t = 1:10

t =

     1     2     3     4     5     6     7     8     9    10

>> k = 2:-0.5:-1

k =

    2.0000    1.5000    1.0000    0.5000         0   -0.5000   -1.0000

>> B = [1:4;5:8]

B =

     1     2     3     4
     5     6     7     8

>> x = zeros(1,3)

x =

     0     0     0

>> y = ones(1,3)

y =

     1     1     1

fx >>
```

Given the Matrix A[3X3]:

```
A =

    3     5     3
    6     8     2
    2     7     3
```

Two points with extremes = all the values from the first extreme to the last extreme

Two points without extremes = all the columns of my variable

```
>> A(6)

ans =

     7
```

```
>> A(3,2)

ans =

     7
```

```
>> A(2,:)

ans =

     6     8     2
```

```
>> A(1:2,2)

ans =

     5
     8
```

```
>> A = [1 2 3;4 5 6;7 8 9]

A =

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

```
>> B = [3 5 2; 5 2 8; 3 6 9]

B =

     3     5     2
     5     2     8
     3     6     9
```

Addition

```
>> X = A + B

X =

     4     7     5
     9     7    14
    10    14    18
```

Subtraction

```
>> Y = A - B

Y =

    -2    -3     1
    -1     3    -2
     4     2     0
```

Transpose

```
>> T = A'

T =

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

Matrices Operations

```
>> A = [1 2 3;4 5 6;7 8 9]

A =

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

Command Window

New to MATLAB? See resources for Getting Started.

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> size(A)

ans =

     3     3

>>
```

Size of a Matrix

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

x =

     1     2     3

>> size(x)

ans =

     1     3

>> y = x'

y =

     1
     2
     3

>> size(y)

ans =

     3     1
```

Size of a vector

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

    A =
       1    2    3
       5    1    4
       3    2   -1
```

x = A(1,:) **y = A(3,:)**

```
x=                  y=
    1   2   3           3  2  -1
```

.* : element-by-element multiplication b = x .* y

b = 3 4 -3

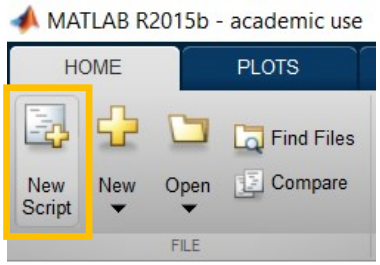
./ : element-by-element division c = x . / y

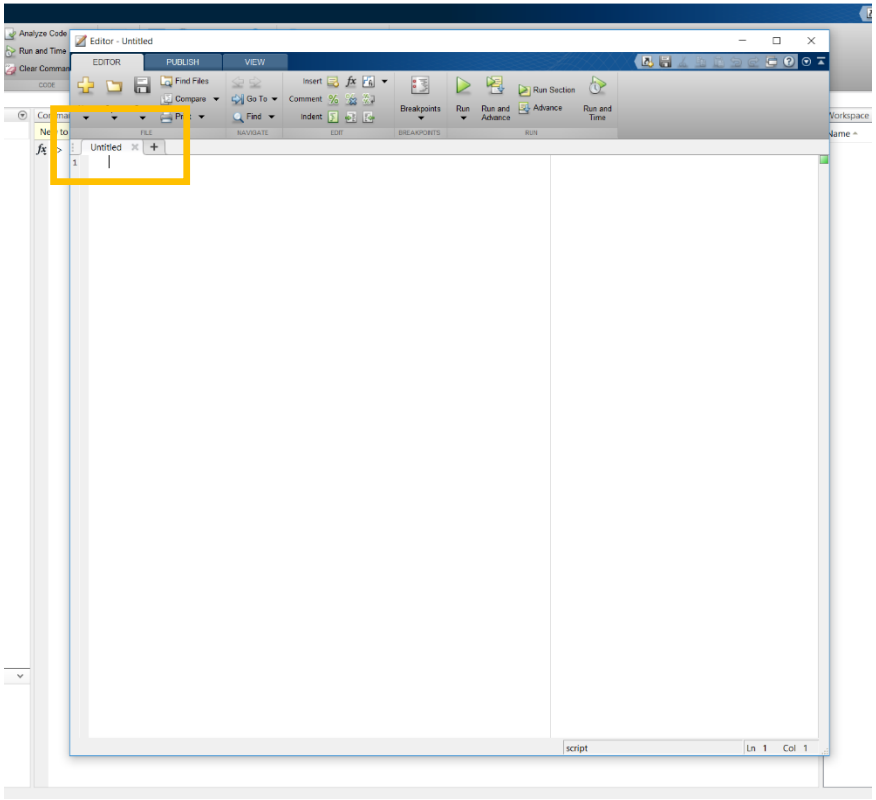
c = 0.33 1 -3

.^ : element-by-element power d = x .^2

d = 1 4 9

Script: m-file



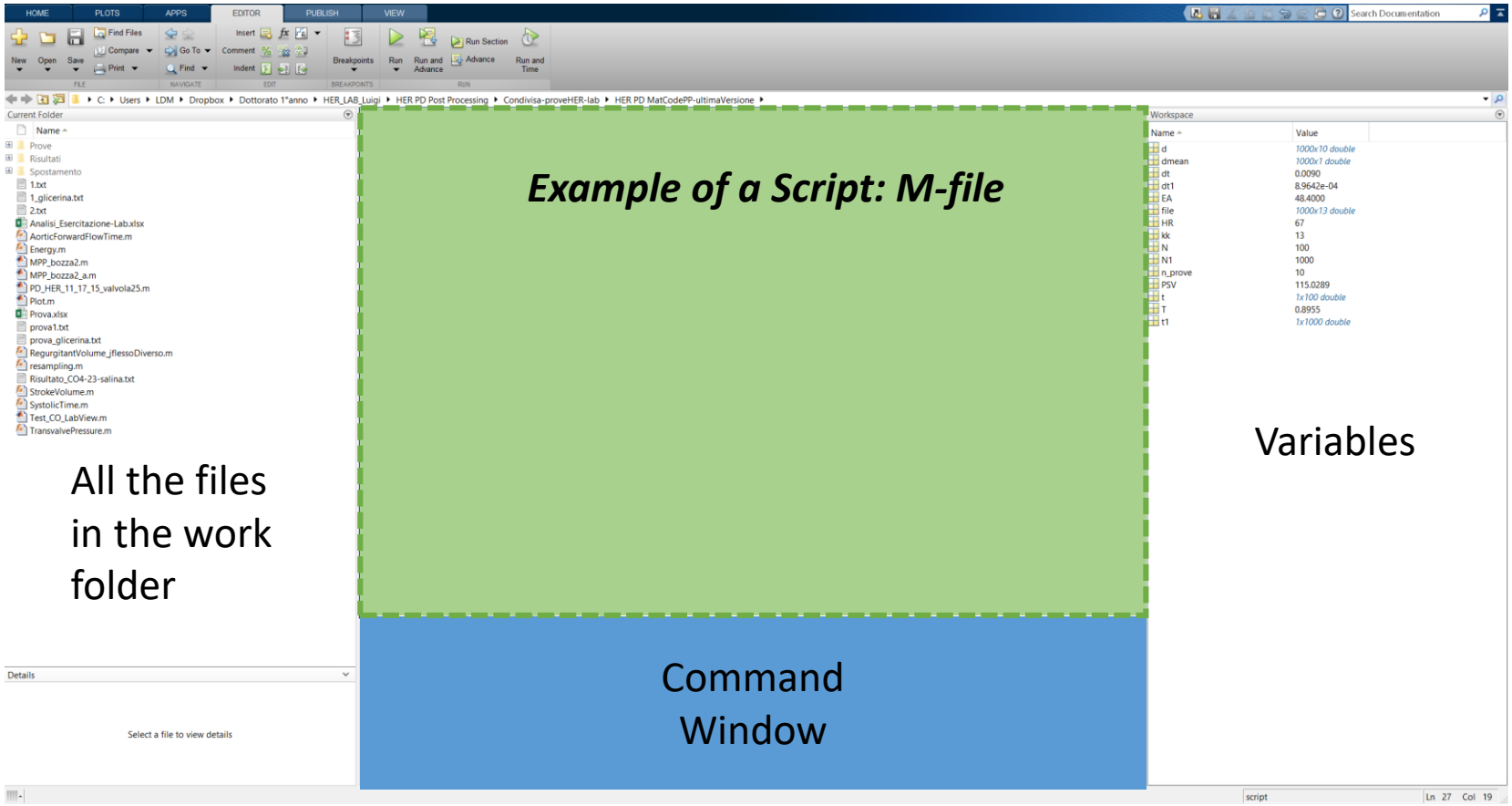


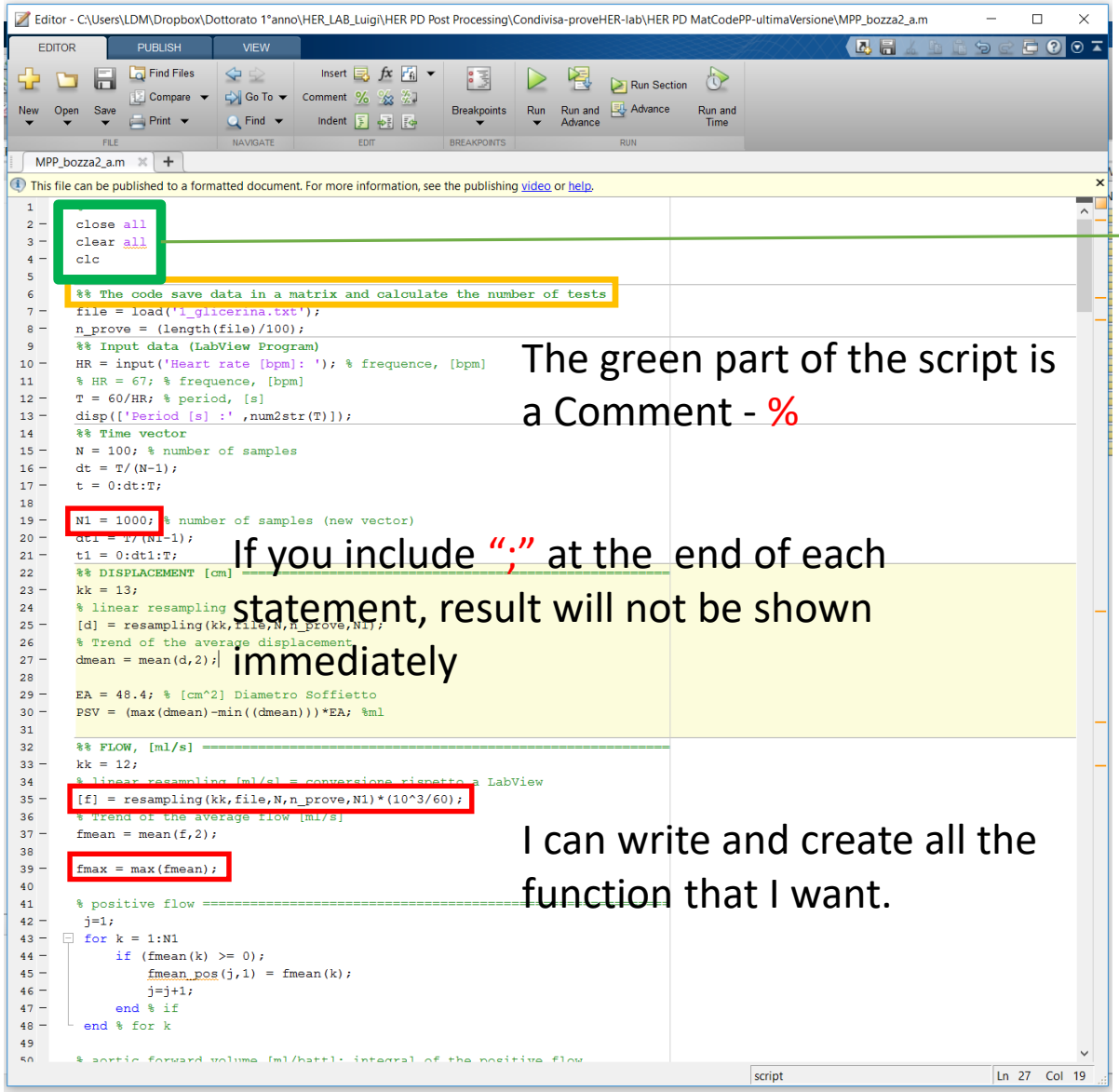
- An m-file is a **text file** used by MATLAB with extension "**.m**"

- It can store a script or an individual function in the MATLAB language

- M files are used for executing algorithms, plotting graphs, and performing other mathematical operations.

Script: m-file





The green part of the script is a Comment - %

If you include ";" at the end of each statement, result will not be shown immediately

I can write and create all the function that I want.

Commands:

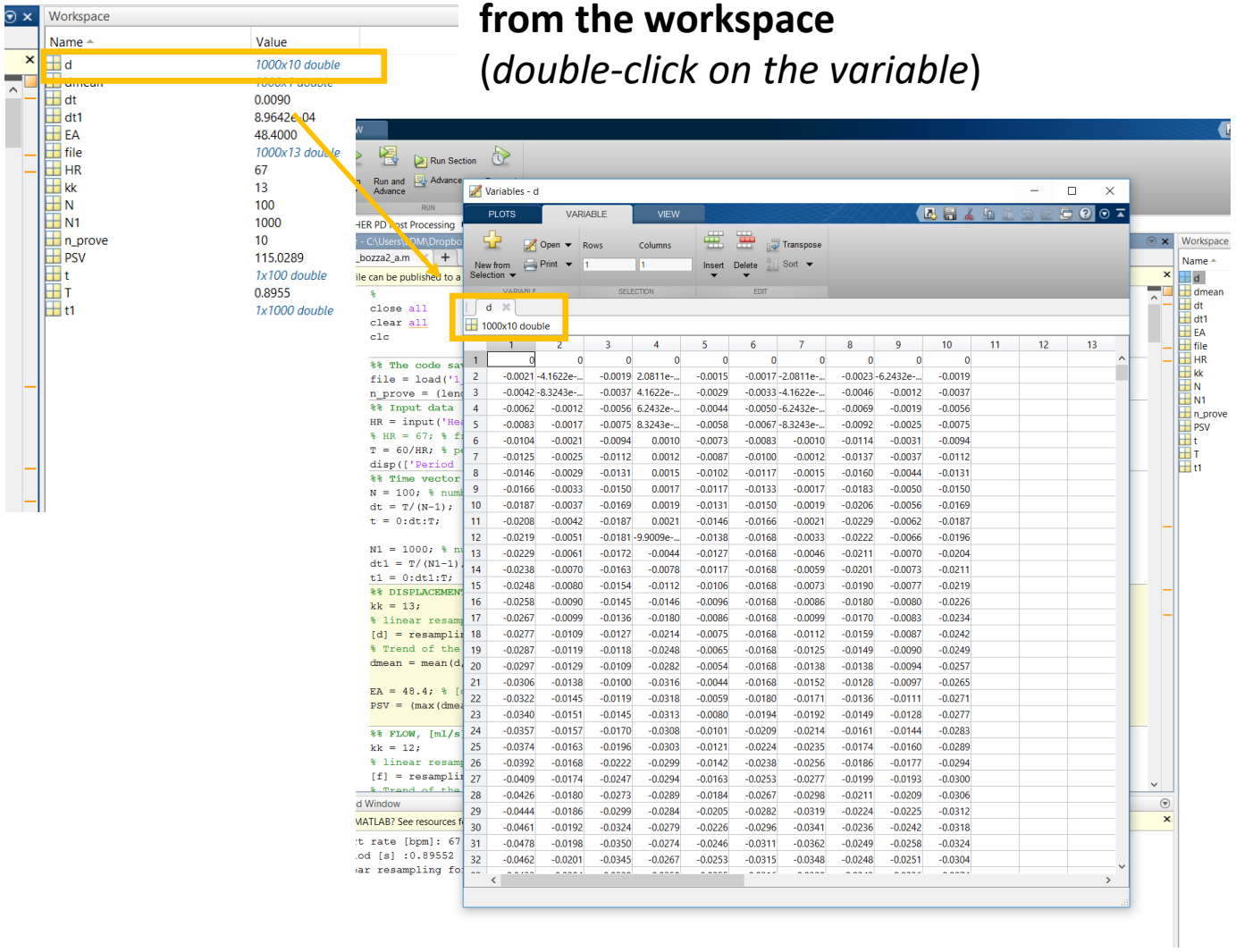
Close all: *closes all the open figure windows*

Clear all: *removes all variables from the workspace*

clc: *Clears the command window*

Script: m-file

Visualization of a Variable from the workspace
(*double-click on the variable*)



Exercise 1: Matrix Operations

- Create a folder (in the desktop or in your Pendrive) : **matlab_introduction**;
- Run MatLab;
- **Open a new script;**
- Create a comment (**%**) with the phrase: **Lesson 1 – Exercise 1: Introduction To Matlab**
- Save the script (file.m) – **exercise_1.m**, inside the folder matlab_introduction;
- Use the command: **clear all, close all, clc**;
- *Use comments for describe the different operations;*
- Create two matrices **A [3x3], B [5x3];**
- Extract two vectors: x [1x3] from A; y [3x1] from B - (***x = A(first row); y = B(first three row, second column)***)
- Calculate the element-by-element multiplication between x and y; (call the result ***a***)
- Calculate the element-by-element multiplication between x and y' (transpose), (call the result ***a_1***);
- *Check the results, which is the difference between these two operations?*
- Calculate the element-by-element division between x and y' (transpose), (call the result ***b***);
- Calculate the element-by-element power of x and y (call the results *q* = x^2 and *w* = *y^2*);
- Matrices Operations: Extract a sub-matrix D [3x3] from B: (***D(second, third and last row of B)***)
- Calculate the addition between the matrix A and D (call the result ***C***);
- Calculate the subtraction between A and D (call the result ***E***);

```
A =

     1     7     5
     4     6    10
    -6     9    15

B =

    -3     8    20
    12     8     2
     0     2     1
     3     4     5
     6    12    -6
```

Editor - C:\Users\LDM\Google Drive\ENVIRONMENTAL FLUID MECHANICS - 2016_2017\Lesson1-MatLab-14_11_2016\matlab-introduction\lesson1.m

lesson1.m

```
% lesson 1: matrices operations
clc
clear all
close all

% Create two matrices A [3x3], B[5x3]
A = [1 7 5; 4 6 10; -6 9 15]
B = [-3 8 20; 12 8 2; 0 2 1; 3 4 5; 6 12 -6]

% Extract two vectors: x [1x3] from A; y [3x1] from B
x = A(1,:)
y = B(1:3,2)

% element-by-element multiplication a = x*y;
a = x.*y

% element-by-element multiplication a = x*y;
a_1 = x.*y'

% element-by-element division b = x/y;
b = x./y'

% element-by-element power q = x^2, w = y^2,
q = x.^2
w = y.^2

% Matrices Oparations: ==========================================
% Extract a sub-matrix D [3x3] from B
D = [B(2,1:3);B(3,1:3);B(5,1:3)]

% addition C = A+B
C = A +D

% subtraction D = A-B;
E = A-D
```

- **plot(Y):** plots the columns of Y versus their index.
- **plot(X,Y):** plots vector Y versus vector X.
- **plot(X,Y,S) :** Various line types, plot symbols, and colors may be obtained with plot(X,Y,S) where S is a character string made from one element from any or all the following 3 columns:

Color

b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

Marker

.	point
o	circle
x	x-mark
+	plus
*	star
s	square
d	diamond
v	triangle (down)
^	triangle (up)
<	triangle (left)
>	triangle (right)
p	pentagram
h	hexagram

Line style

-	solid
:	dotted
-.	dashdot
--	dashed
(none)	no line

Examples:

plot(X,Y,**'r+:'**): *plots a red dotted line with a plus at each data point;*

plot(X,Y,**'bd'**): *plots blue diamond at each data point but does not draw any line.*

- **plot(Y):** plots the columns of Y versus their index.
- **plot(X,Y):** plots vector Y versus vector X.
- **plot(X,Y,S) :** Various line types, plot symbols and colors may be obtained with plot(X,Y,S) where S is a character string made from one element from any or all the following 3 columns:
- **plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...)** combines the plots defined by the (X,Y,S) triples.

Example:

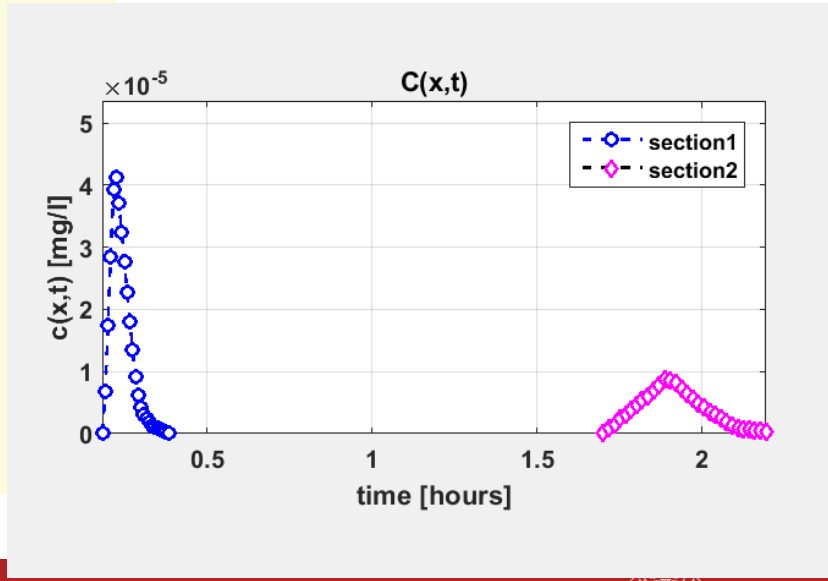
plot(X1,Y1,'y-',X2,Y2,'go') plots the two datasets, with a solid yellow line interpolating green circles at the data points.

- **plot(Y):** plots the columns of Y versus their index.
- **plot(X,Y):** plots vector Y versus vector X.
- **plot(X,Y,S) :** Various line types, plot symbols and colors may be obtained with plot(X,Y,S) where S is a character string made from one element from any or all the following 3 columns:
- **plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...)** combines the plots defined by the (X,Y,S) triples.
- The X,Y pairs, or X,Y,S triples, can be followed by parameter/value pairs to specify **additional properties of the lines**.

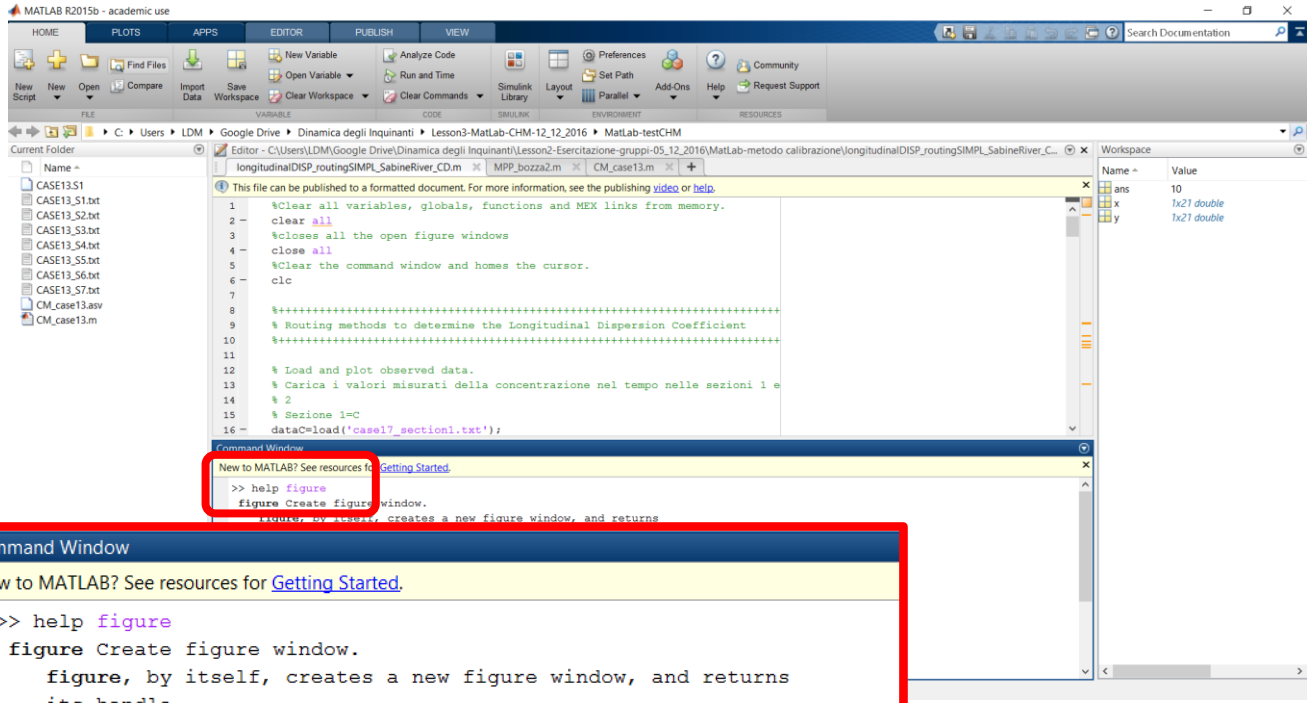
Examples:

```
plot(x,y,'--rs','LineWidth',0.5,'MarkerEdgeColor','k',
'MarkerFaceColor','g','MarkerSize',2)
```

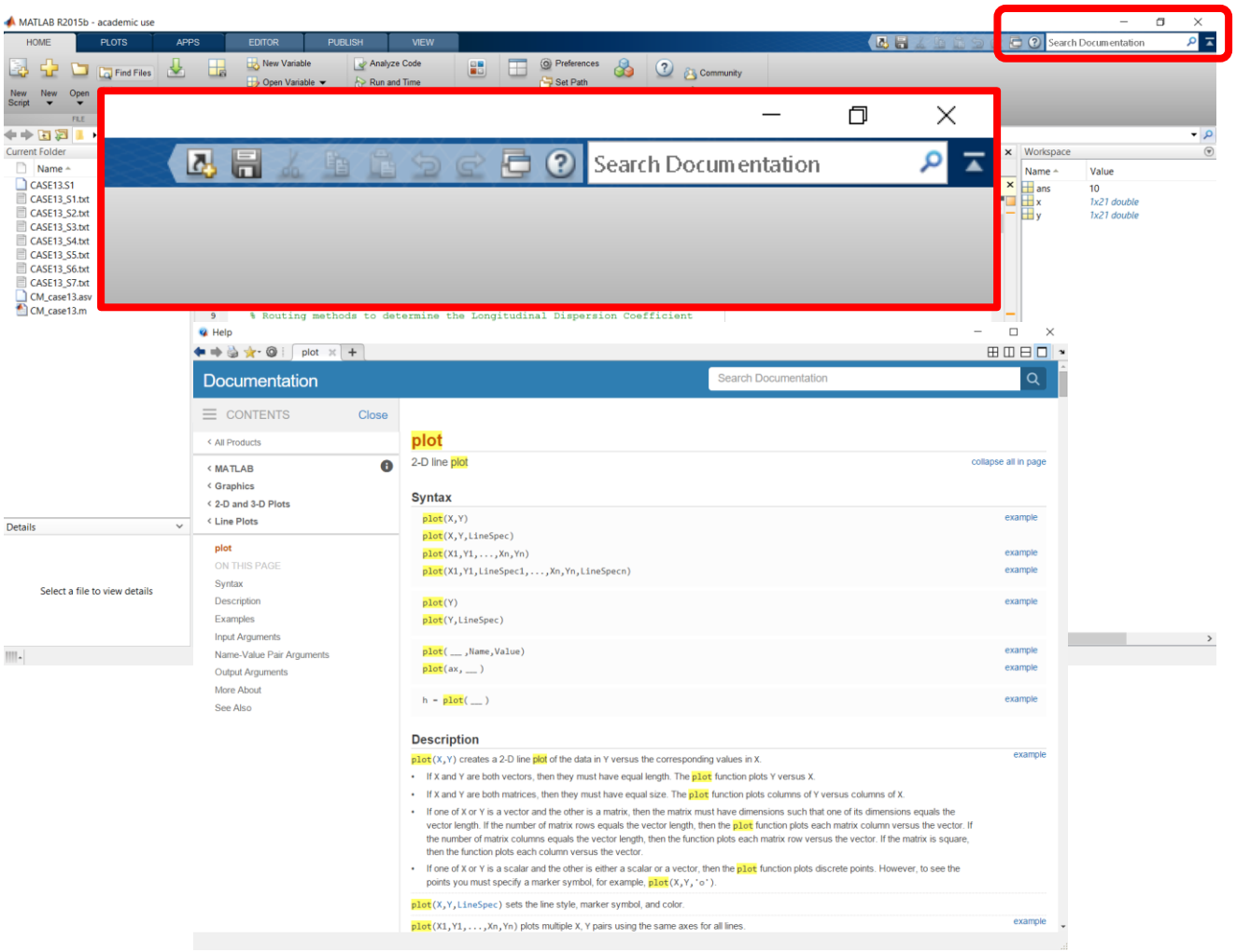
```
46 -    figure_1 = figure();
47 -    hold on
48      %Plot Observed data
49 -    plot(t_dataC,C_dataC,'--b','linewidth',1.5,...
50              'Marker','o','MarkerSize',6,...
51              'MarkerEdgeColor',[0 0 1],...
52              'MarkerFaceColor',[1 1 1]);
53 -    plot(t_dataD,C_dataD,'--k','linewidth',1.5,...
54              'Marker','d','MarkerSize',6,...
55              'MarkerEdgeColor',[1 0 1],...
56              'MarkerFaceColor',[1 1 1]);
57 -    title('C(x,t)','FontSize',12,'FontWeight','bold')
58 -    axis([min(t_dataC) max(t_dataD) 0  max(C_dataC)])
59 -    xlabel('time [hours]','FontSize',12,'FontWeight','bold');
60 -    ylabel('c(x,t) [mg/l]','FontSize',12,'FontWeight','bold');
61 -    set(gca,'FontSize',12,'FontWeight','bold');
62 -    legend('section1','section2')
63 -    box on
64 -    grid on
65      % save image
66 -    set(gcf, 'PaperUnits', 'centimeters');
67 -    set(gcf, 'PaperSize', [10 10]);
68 -    set(gcf, 'PaperPositionMode','manual');
69 -    set(gcf, 'PaperUnits','normalized');
70 -    set(gcf, 'PaperPosition',[0.1 0.1 2 2]);
71 -    fig = 'figure_1';
72 -    print('-dpng',figure_1,fig);
73
```



Help & Documentation



Help & Documentation

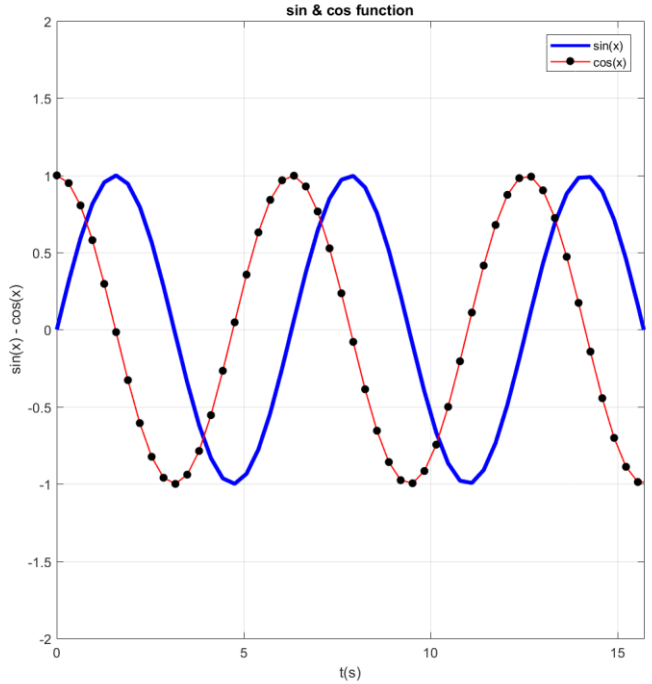


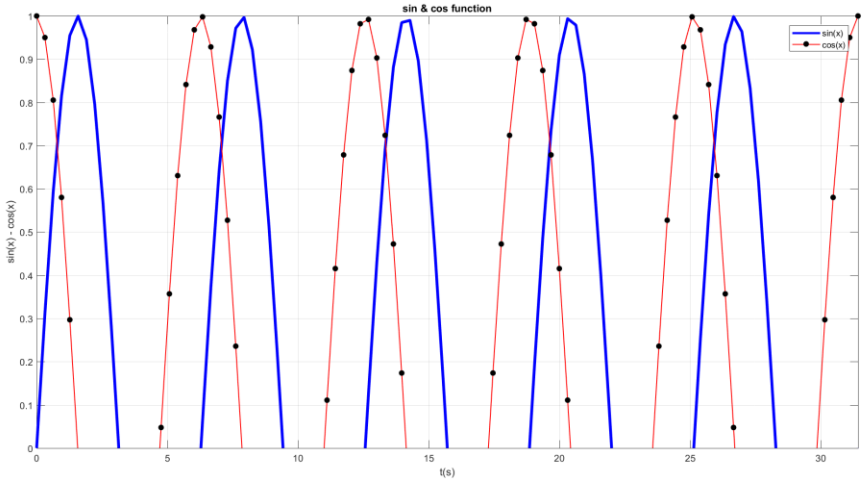
- Create a new script;
- Create a comment with the phrase: **Lesson 1 – Exercise 2: Introduction To Matlab**
- Save the script (file.m) – exercise_2.m, inside the folder matlab_introduction;
- Use the command: clear all, close all, clc;
- Use the help function (in the command window) to understand how to use the function ***linspace***;
- Create a vector (x) of 100 samples between 0 and 10*π. ("linspace");
- Calculate y = **sin(x)** of the vector;
- Crate a figure window (figure_1 = figure());
- Plot x vs y;
- Insert the title (sin function);
- Insert the xlabel(t[s])
- Insert the ylabel(sin(x));
- Add the grid and the external box to the figure; (to understand how to insert box and grid use the help function in the command window);
- Calculate z= **cos(x)** of the vector;
- Use the help function (in the command window) to understand how to use the function ***hold*** for include different plots in the same figure windows;

Exercise 2: plot – part B

- Create e new figure window (figure_2 = figure())
- Plot together x vs y & x vs z;
- Insert a new title (sin and cos functions)
- Insert the new ylabel(sin(x) & cos(x));
- Insert the xlabel (t(s));
- Add the grid and the external box to the figure;
- Add the legend to the differ curves;
- **For plot sin(x):** Set the color (blue), line style (solid line) and line width = 3;
- **For plot cos(x):** Set the color (red), line style (solid line) , line width (1), marker (o), marker size (10), ; Marker Face and edge Color (black).
- Set the axis between **0 to 5*pi** along x direction and y **-2 to 2** along y direction
- Save the image 2: figure_2 with ***PaperSize 10x10***
- Create a figure 3 equal to the previous one but setting the axis along y direction between 0 to 2 and 0 to 10*pi along x direction.
- Save the image 3: figure_3 with ***PaperSize 20x10***

Solution Exercise 2





Syntax:

for index = values

Statements

End

This syntax that describes a for cycle executes a group of statements in a loop for a specified number of times.

values has one of the following forms:

- ***initVal : endVal*** — Increments the index variable from initVal to endVal by 1, and repeats execution of statements until index is greater than endVal.
- ***initVal :step: endVal*** — Increments index by the value step on each iteration, or decrements index when step is negative.

Syntax:

for index = values

Statements

End

This syntax that describes a for cycle executes a group of statements in a loop for a specified number of times.

```
s = 10;
```

```
for c = 1:s
    for r = 1:s
        H(r,1) = 1/(r+r-1);
    end
end
```

```
s = 10;
% H = zeros(s);
%
% for c = 1:s
      for r = 1:s
          H(r,1) = 1/(r+r-1);
      end
% end
```

10x1 double

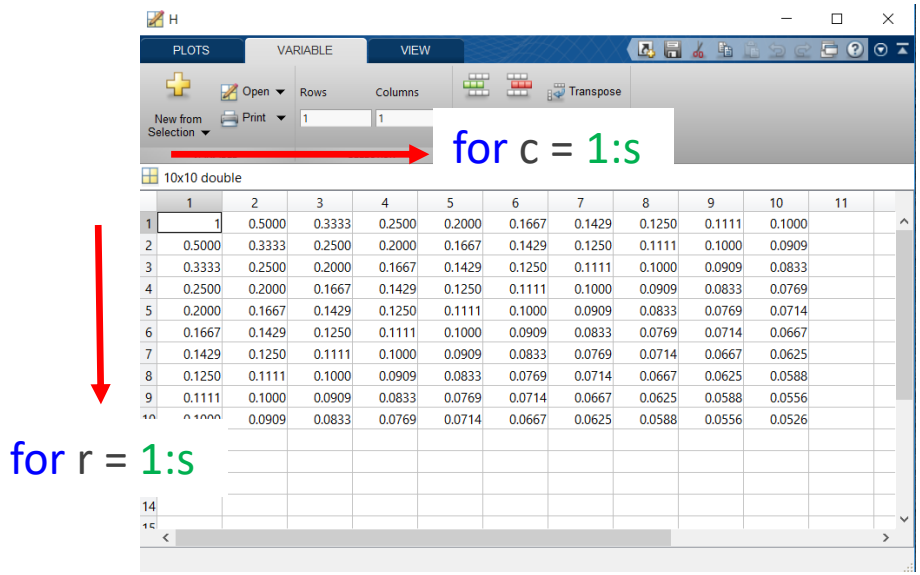
	1	2	3	4
1	1			
2	0.3333			
3	0.2000			
4	0.1429			
5	0.1111			
6	0.0909			
7	0.0769			
8	0.0667			
9	0.0588			
10	0.0526			
11				
12				

Syntax:

```
for index = values
    Statements
End
```

This syntax that describes a for cycle executes a group of statements in a loop for a specified number of times.

```
s = 10;

for c = 1:s
    for r = 1:s
        H(r,c) = 1/(r+c-1);
    end
end
```



	1	2	3	4	5	6	7	8	9	10	11
1	1	0.5000	0.3333	0.2500	0.2000	0.1667	0.1429	0.1250	0.1111	0.1000	
2	0.5000	0.3333	0.2500	0.2000	0.1667	0.1429	0.1250	0.1111	0.1000	0.0909	
3	0.3333	0.2500	0.2000	0.1667	0.1429	0.1250	0.1111	0.1000	0.0909	0.0833	
4	0.2500	0.2000	0.1667	0.1429	0.1250	0.1111	0.1000	0.0909	0.0833	0.0769	
5	0.2000	0.1667	0.1429	0.1250	0.1111	0.1000	0.0909	0.0833	0.0769	0.0714	
6	0.1667	0.1429	0.1250	0.1111	0.1000	0.0909	0.0833	0.0769	0.0714	0.0667	
7	0.1429	0.1250	0.1111	0.1000	0.0909	0.0833	0.0769	0.0714	0.0667	0.0625	
8	0.1250	0.1111	0.1000	0.0909	0.0833	0.0769	0.0714	0.0667	0.0625	0.0588	
9	0.1111	0.1000	0.0909	0.0833	0.0769	0.0714	0.0667	0.0625	0.0588	0.0556	
10	0.1000	0.0909	0.0833	0.0769	0.0714	0.0667	0.0625	0.0588	0.0556	0.0526	

If cycle

Syntax

```
if expression
   statements
elseif expression
   statements
else
   statements
end
```

Executes a group of statements when the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.

The elseif and else blocks are optional. The statements execute only if previous expressions in the *if*...*end* block are false.

An *if* block can include multiple *elseif* blocks.

Syntax

```
if expression
   statements
elseif expression
   statements
else
   statements
end
```

Assign to a matrix values that depend on their indices

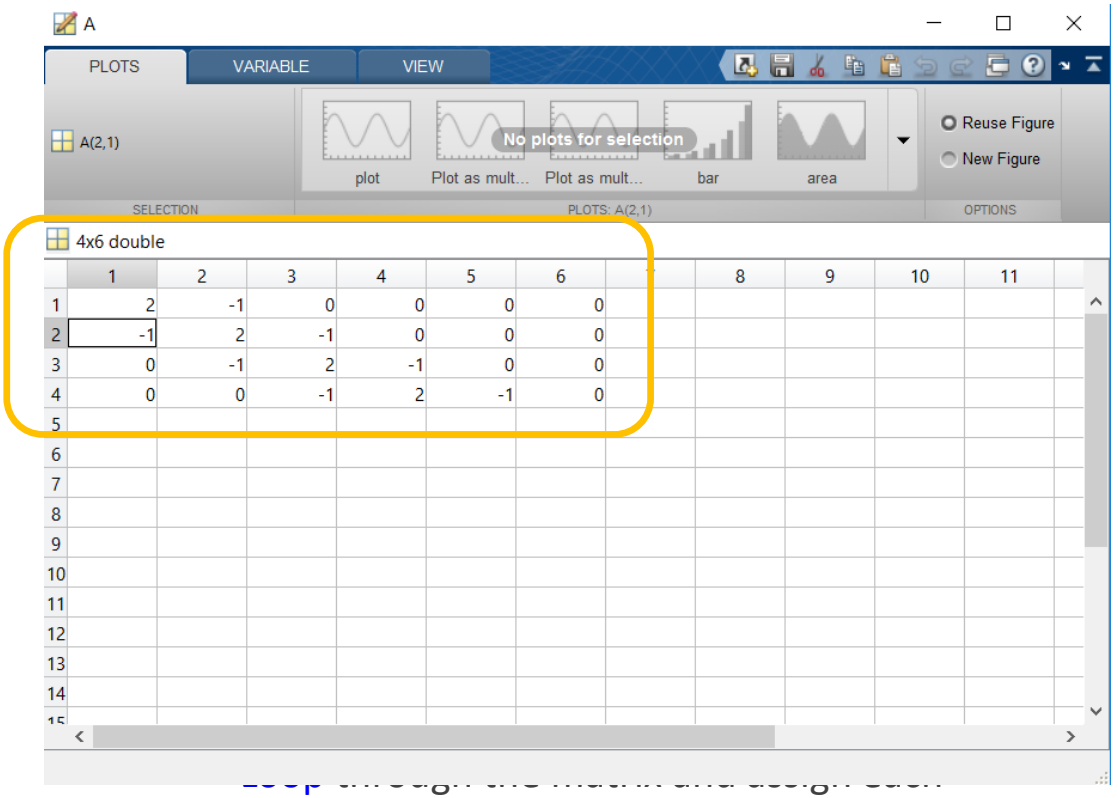
```
for c = 1:ncols
    for r = 1:nrows
        if r == c
          A(r,c) = 2;
        elseif abs(r-c) == 1
          A(r,c) = -1;
        else
          A(r,c) = 0;
        end
    end
end
```

Loop to define each element of matrix A. Assign 2 on the main diagonal, -1 on the adjacent diagonals, and 0 everywhere else.

Assign to a matrix values that depend on their indices

Syntax

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```



	1	2	3	4	5	6	8	9	10	11
1	2	-1	0	0	0	0				
2	-1	2	-1	0	0	0				
3	0	-1	2	-1	0	0				
4	0	0	-1	2	-1	0				
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										

Loop through the matrix and assign each element a new value. Assign 2 on the main diagonal, -1 on the adjacent diagonals, and 0 everywhere else.

function LOAD, file .txt
Load data from a folder into workspace.

S = load(FILENAME) loads the variables from a MAT-file into a structure array (matrix)

CASE13_S1.txt - Blocco note

File Modifica Formato Visualizza ?

time	C
4.000	0.00
4.200	0.97
4.400	3.98
4.600	9.21
4.800	11.46
5.000	10.23
5.200	7.23
5.400	4.64
5.600	2.86
5.800	1.75
6.000	0.88
6.200	0.54
6.400	0.37
6.600	0.28
6.800	0.22
7.000	0.18
7.200	0.16
7.400	0.14
7.600	0.13
7.800	0.11
8.000	0.08
8.200	0.07
8.400	0.05
8.600	0.03
8.800	0.01
9.000	0.00

Editor - C:\Users\LDM\Google Drive\Dinamica degli Inquinanti\Lesson3-MatLab-CHM-12_12_2016\MatLab-testCHM\CM_case13.m

longitudinalDISP_routingSIMPL_SabineRiver_CD.m | MPP_bozza2.m | CM_case13.m

This file can be published to a formatted document. For more information, see the publishing video or help.

```
% matlab program  - chatwin method
% case 13 - Chattahoochee river
% ------------------------------------------------------------------------
%%
close all
clear all
clc
%%
% loading data - function LOAD, file .txt
% Load data from MAT-file into workspace.
% S = load(FILENAME) loads the variables from a MAT-file into a structure
% array (matrix)

file_s1 = load('CASE13_S1.txt'); % load file section S1

t_s1 = file_s1(:,1); % extract the first column = time

c_s1 = file_s1(:,2)/1000000; % extract the second column = concentration
```

Workspace

Name	Value
c_s1	26x1 double
file_s1	26x2 double
t_s1	26x1 double

Loading of the entire matrix (*file .txt*) and extraction of the vectors of the time (*first column*) and concentration (*second column*)

function ***importdata (filename)***
A = *importdata* (FILENAME)

Editor - test.m

test.m

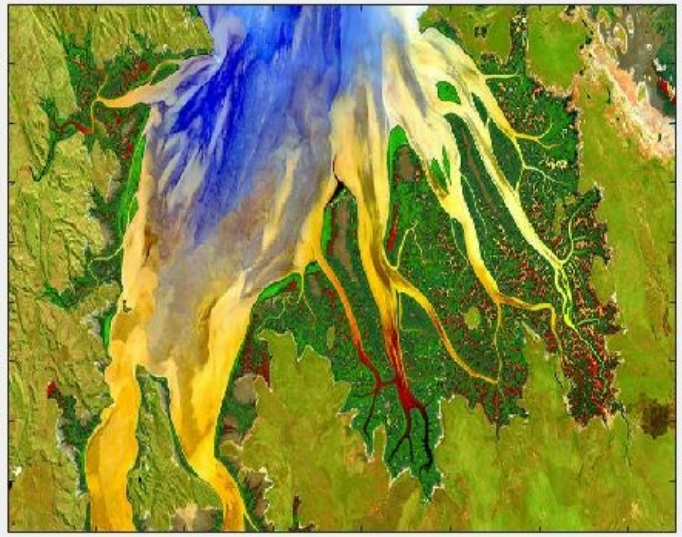
This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
% A = importdata(filename) loads data into array A.

clear all
close all
clc

A = importdata('Cambridge Gulf_Australia.jpg');
image(A)
```

Variables - A.colheaders

Figure 1

File Edit View Insert Tools Desktop Window Help



Editor - test.m

test.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%% Example 2
% Using a text editor, create a space-delimited ASCII file with column
% headers called myfile01.txt.
%
% Day1  Day2  Day3  Day4  Day5  Day6  Day7
% 95.01 76.21 61.54 40.57  5.79 20.28  1.53
% 23.11 45.65 79.19 93.55 35.29 19.87 74.68
% 60.68  1.85 92.18 91.69 81.32 60.38 44.51
% 48.60 82.14 73.82 41.03  0.99 27.22 93.18
% 89.13 44.47 17.63 89.36 13.89 19.88 46.60
```

myfile01.txt - Blocco note di Windows

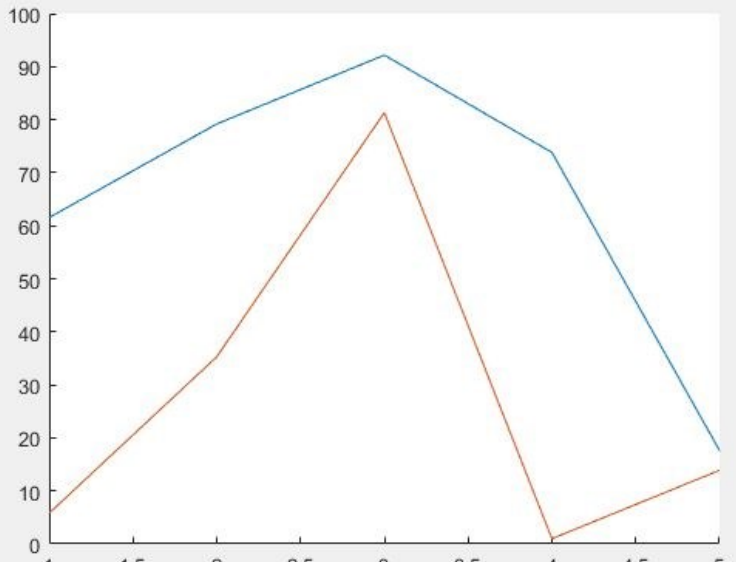
File Modifica Formato Visualizza ?

```
Day1  Day2  Day3  Day4  Day5  Day6  Day7
95.01 76.21 61.54 40.57  5.79 20.28  1.53
23.11 45.65 79.19 93.55 35.29 19.87 74.68
60.68  1.85 92.18 91.69 81.32 60.38 44.51
48.60 82.14 73.82 41.03  0.99 27.22 93.18
89.13 44.47 17.63 89.36 13.89 19.88 46.60
```

Variables - A.colheaders

Figure 2

File Edit View Insert Tools Desktop Window Help



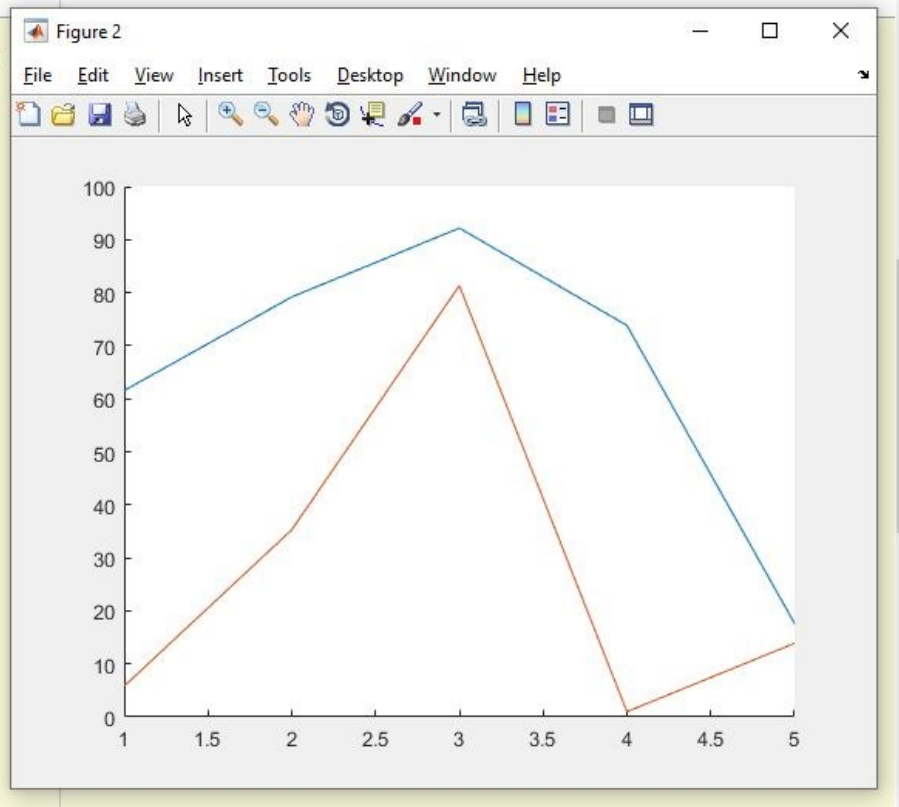
Editor - test.m

test.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%% Example 2
% Using a text editor, create a space-delimited ASCII file with column
% headers called myfile01.txt.
%
% Day1  Day2  Day3  Day4  Day5  Day6  Day7
% 95.01 76.21 61.54 40.57  5.79 20.28  1.53
% 23.11 45.65 79.19 93.55 35.29 19.87 74.68
% 60.68  1.85 92.18 91.69 81.32 60.38 44.51
% 48.60 82.14 73.82 41.03  0.99 27.22 93.18
% 89.13 44.47 17.63 89.36 13.89 19.88 46.60

% Import the file
filename = 'myfile01.txt';
% delimiterIn = ' ';
% headerlinesIn = 1;
A = importdata(filename);%,delimiterIn,headerlinesIn);

% View columns 3 and 5.

for k = [3, 5]
    disp(A.colheaders{1, k})
    disp(A.data(:, k))
%    disp(' ')
figure(2)
hold on
plot(A.data(:, k))
end
```

Variables - A.colheaders



Editor - test.m

Variables - A.colheaders

test.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%% Example 3 From Excel file

% A Microsoft Excel spreadsheet labeled Fogliol in file test_Excel.xls
% contains two columns of numeric data with text headers for each column:
% time (s)  d (mm)
% 0.000 0.000
% 0.001 0.055
% 0.002 0.079
% 0.003 0.109
% 0.004 0.140
% 0.006 0.214
% 0.007 0.252
% 0.008 0.341
% 0.010 0.443
% 0.011 0.497
% 0.012 0.553
% 0.014 0.671 .... (Continua...)

% If you want to import only the numeric data, use xlsread with a
% single return argument. Specify the filename and sheet name as inputs.
%
% xlsread ignores any leading row or column of text in the numeric result.
%
ndata = xlsread('test_Excel.xlsx', 'Fogliol')
%
% To import both the numeric data and the text data, specify two return values for xlsread:
%
[ndata, headertext] = xlsread('test_Excel.xlsx', 'Fogliol')

num = xlsread('test_Excel.xlsx', 'Fogliol', 'A2:B16')
```