Basic mathematical prerequisites and MATLAB Programming

Linear Algebra Review

Arrays, matrices, vectors and scalars

- Scalar: A variable with a single number
- Array: A special variable stores multiple values (called elements).
  - Multidimensional array: Arrays containing one or more arrays
  - The dimension of an array indicates the number of indices you need to select an element.
    - For a two-dimensional array you need two indices to select an element
    - For a three-dimensional array you need three indices to select an element
- Matrix:
  - Two dimensional array of numbers
  - Matrix dimension: [number of rows $\times$ number of columns]
  - Matrix A with [$n \times m$] dimension and elements (entries) A_{ij} in the i^{th} row and j^{th} column (indices start from left top to bottom right)
- Vector: one-dimensional array of numbers
- Vectors and scalars are special form of matrices:
  - Column vector: $[n \times 1]$ matrix
  - Row vector: $[1 \times m]$ matrix
  - Scalar: $[1 \times 1]$ matrix

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 11 \\ 13 & 15 & 17 \\ 1 & 3 & 5 \end{bmatrix}$$

Matrix manipulation

- Matrix addition:

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 7 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 1 & 3 \end{bmatrix} \quad \Rightarrow C = A + B = ? \qquad C_{ij} = A_{ij} + B_{ij}$$

$$D = \begin{bmatrix} 2 & 4 \\ 6 & 8 \\ 1 & 3 \end{bmatrix} \qquad \Rightarrow E = C + D = ?$$

- Matrix multiplication:

$$L = A \times B = ?$$

$$M = A \times D = ? \qquad M_{ij} = \sum_{k=1}^{K} A_{ik} D_{kj}$$

$$N = D \times A = ?$$

- Matrix multiplication properties:

  - Not commutative $A \times D \neq D \times A$
  - Associative $A \times D \times B = (A \times D) \times B = A \times (D \times B)$

- Scalar multiplication:

$$\lambda : \text{ number}$$
$$Q = \lambda \times A = A \times \lambda$$
$$Q_{ij} = \lambda A_{ij}$$

- Transpose matrix:

$$Q = A^T = A^{'}$$
$$Q_{ij} = A_{ji}$$

- Identify (square) matrix: $I \times R = R \times I = R$ $\quad I = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$

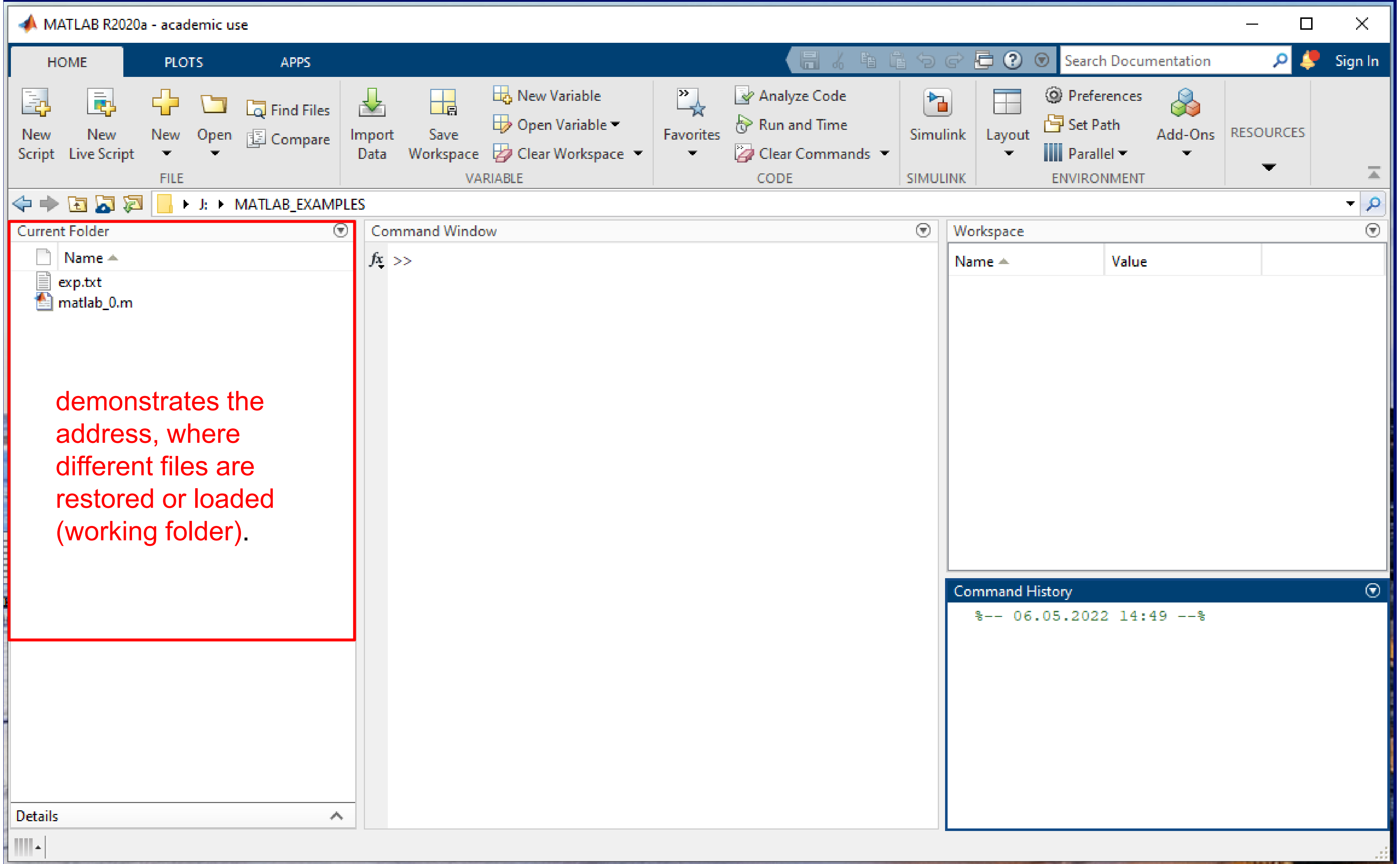
- Inverse of a square matrix: $R^{-1} \times R = R \times R^{-1} = I$

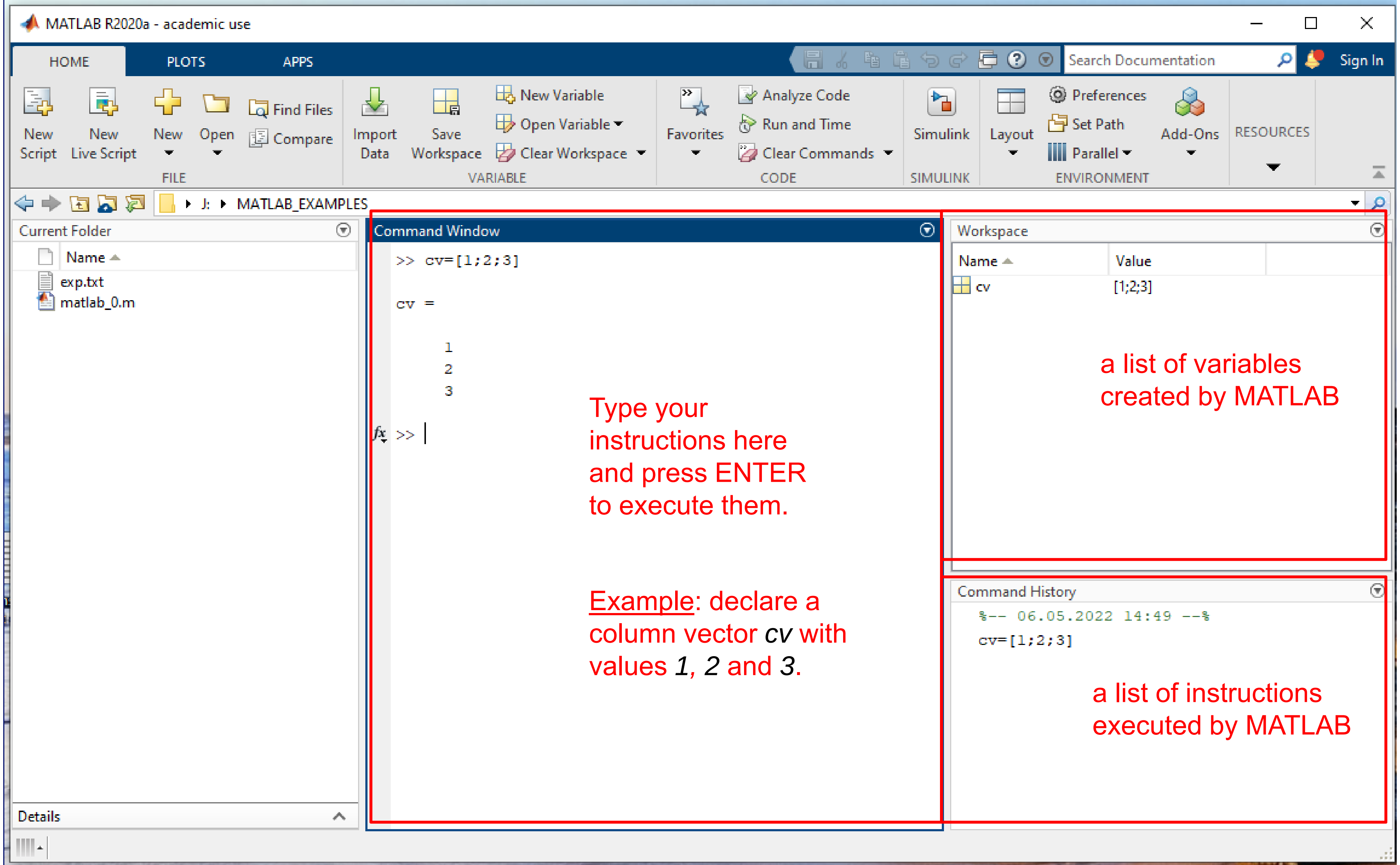
Matrices that don't have an inverse are "singular" or "degenerate"

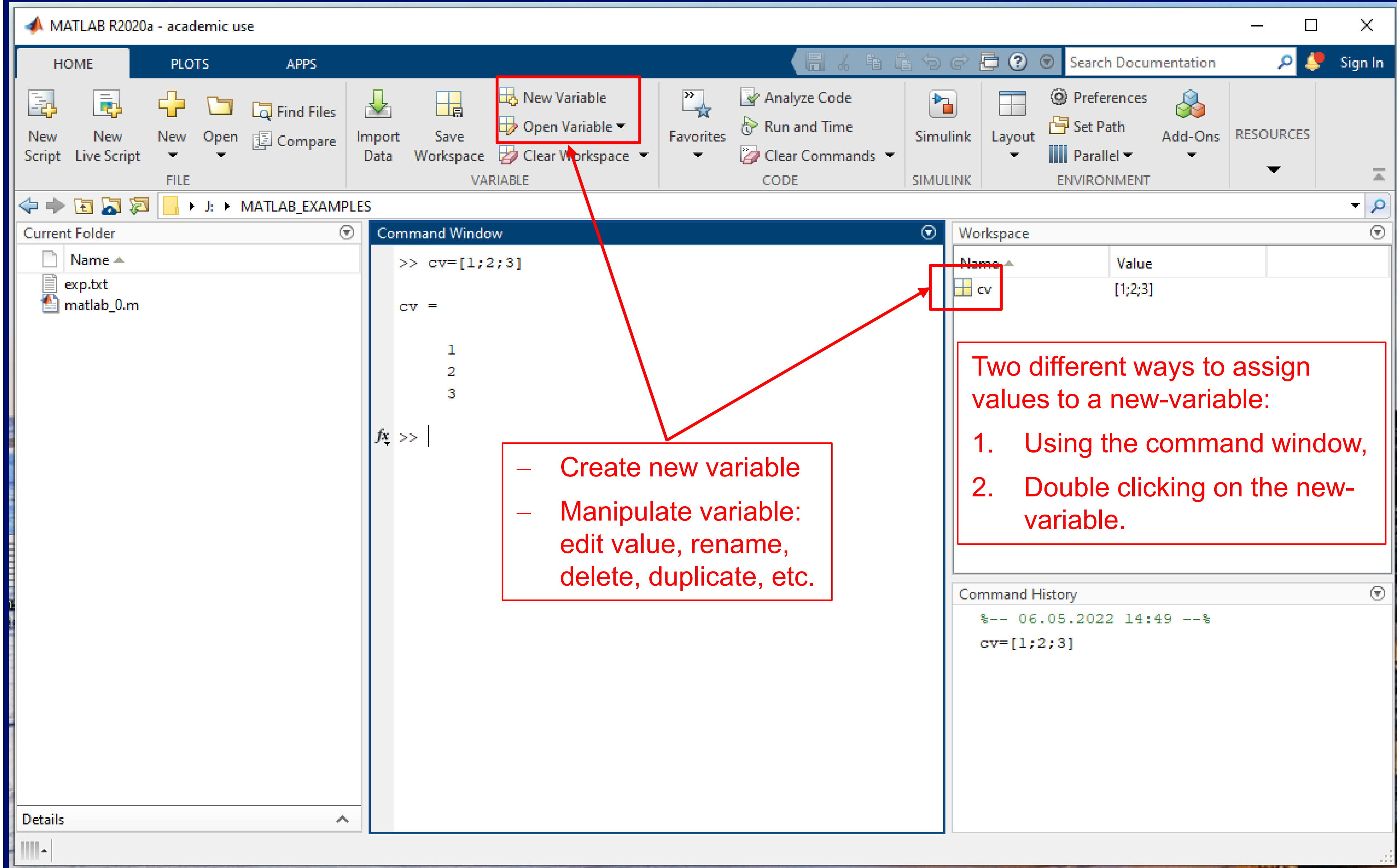
Basic MATLAB Programming

- Basic interface
- Working with a script file (m-file)
- Basic variables
  - Declaring & manipulating matrix variables
- Basic operators
  - Conditional operators
  - Input & output operators
- Functions

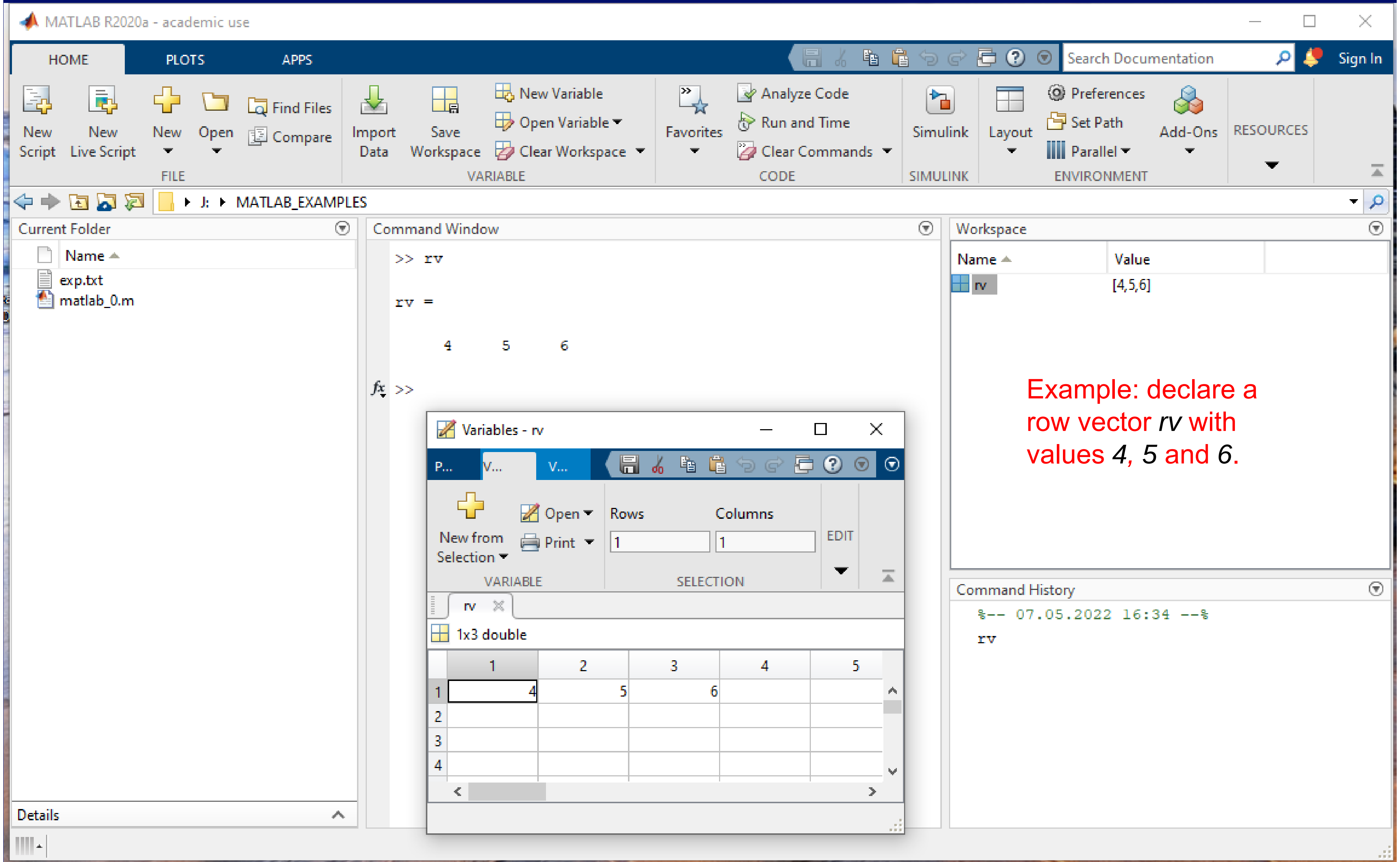
Basic interface


MATLAB R2020a - academic use
HOME
PLOTS
APPS
Search Documentation
Sign In
New Script
New Live Script
New
Open
Find Files
Compare
FILE
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
VARIABLE
Favorites
Analyze Code
Run and Time
Clear Commands
CODE
Simulink
SIMULINK
Layout
Preferences
Set Path
Parallel
Add-Ons
ENVIRONMENT
RESOURCES
J: MATLAB_EXAMPLES
Current Folder
Name
exp.txt
matlab_0.m
demonstrates the address, where different files are restored or loaded (working folder).
Details
Command Window
>>
Workspace
Name
Value
Command History
%-- 06.05.2022 14:49 --%

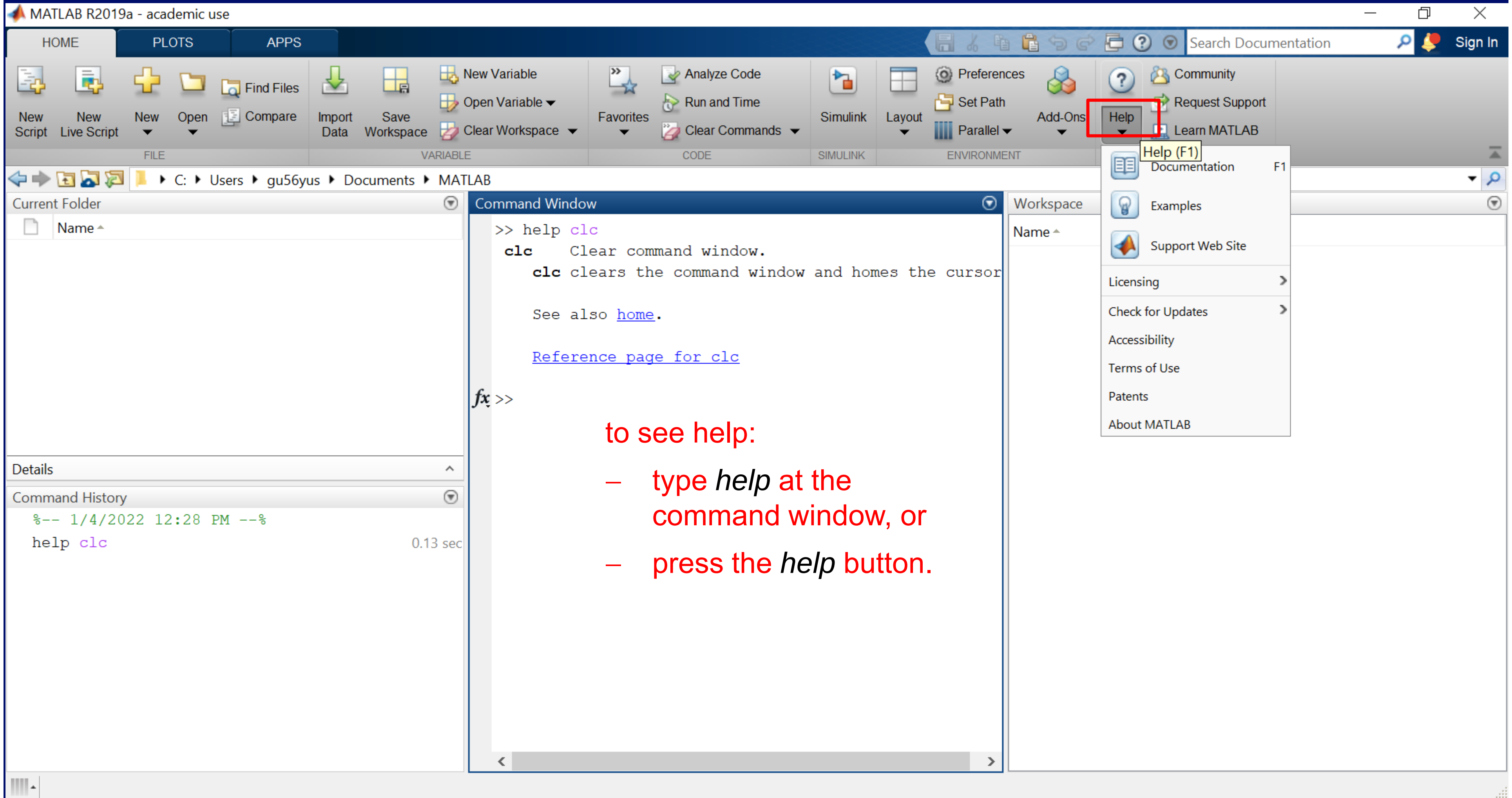

MATLAB R2020a - academic use
HOME
PLOTS
APPS
Search Documentation
Sign In
New Script
New Live Script
New
Open
Find Files
Compare
FILE
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
VARIABLE
Favorites
Analyze Code
Run and Time
Clear Commands
CODE
Simulink
SIMULINK
Layout
Preferences
Set Path
Parallel
ENVIRONMENT
Add-Ons
RESOURCES
J: MATLAB_EXAMPLES
Current Folder
Name
exp.txt
matlab_0.m
Details
Command Window
>> cv=[1;2;3]
cv =
1
2
3
>>
Type your instructions here and press ENTER to execute them.
Example: declare a column vector cv with values 1, 2 and 3.
Workspace
Name
Value
cv
[1;2;3]
a list of variables created by MATLAB
Command History
%-- 06.05.2022 14:49 --%
cv=[1;2;3]
a list of instructions executed by MATLAB



MATLAB R2020a - academic use
HOME
PLOTS
APPS
New Script
New Live Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
VARIABLE
Favorites
Analyze Code
Run and Time
Clear Commands
CODE
Simulink
SIMULINK
Layout
Preferences
Set Path
Parallel
ENVIRONMENT
Add-Ons
RESOURCES
Search Documentation
Sign In
J: MATLAB_EXAMPLES
Current Folder
Name
exp.txt
matlab_0.m
Details
Command Window
>> cv=[1;2;3]
cv =
1
2
3
>>
Workspace
Name
Value
cv
[1;2;3]
Command History
%-- 06.05.2022 14:49 --%
cv=[1;2;3]
– Create new variable
– Manipulate variable: edit value, rename, delete, duplicate, etc.
Two different ways to assign values to a new-variable:
1. Using the command window,
2. Double clicking on the new-variable.



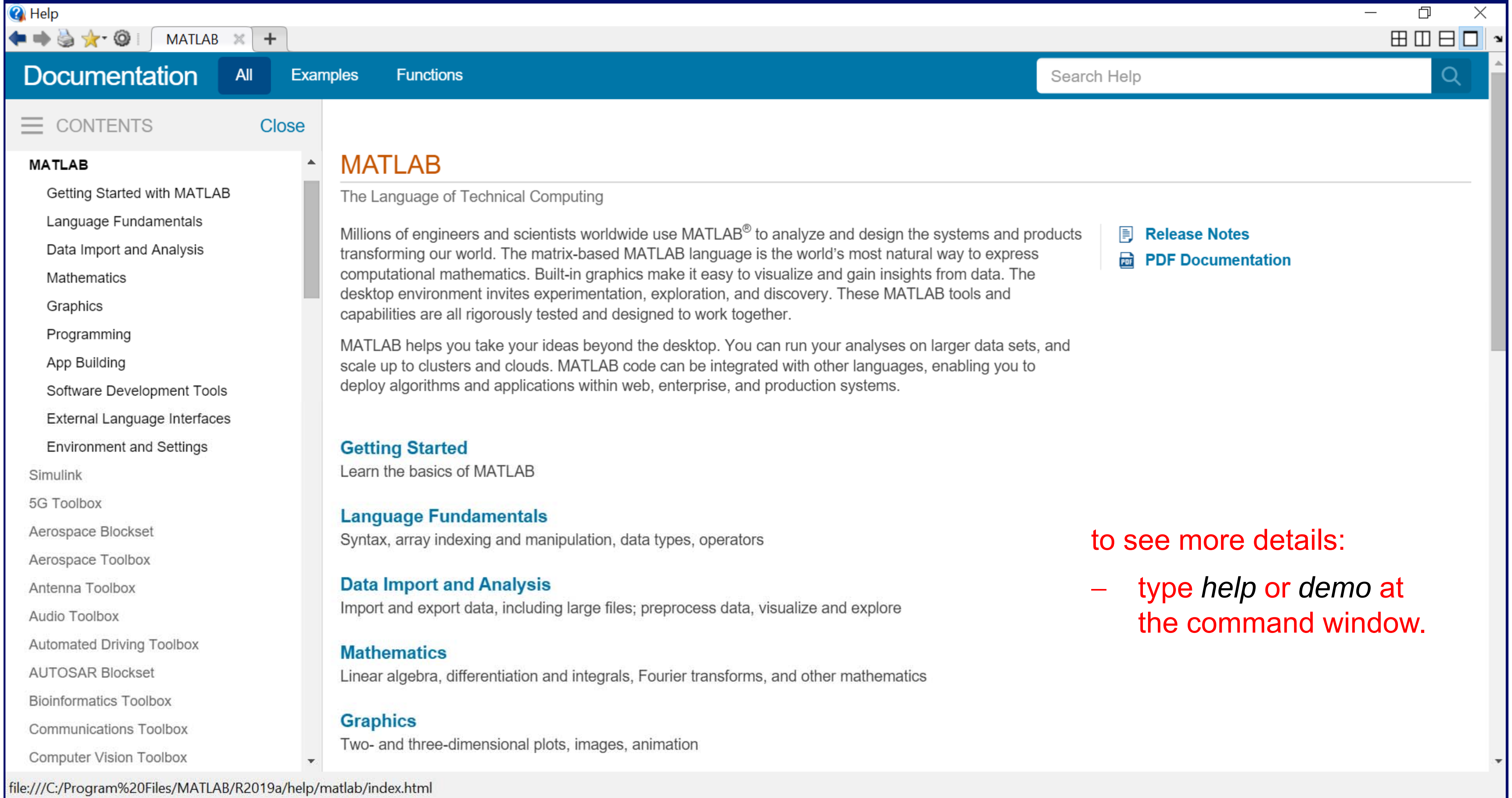
Example: declare a row vector *rv* with values *4*, *5* and *6*.



to see help:

– type *help* at the command window, or

– press the *help* button.



to see more details:

– type *help* or *demo* at the command window.

Useful Commands and Functions in the interactive mode

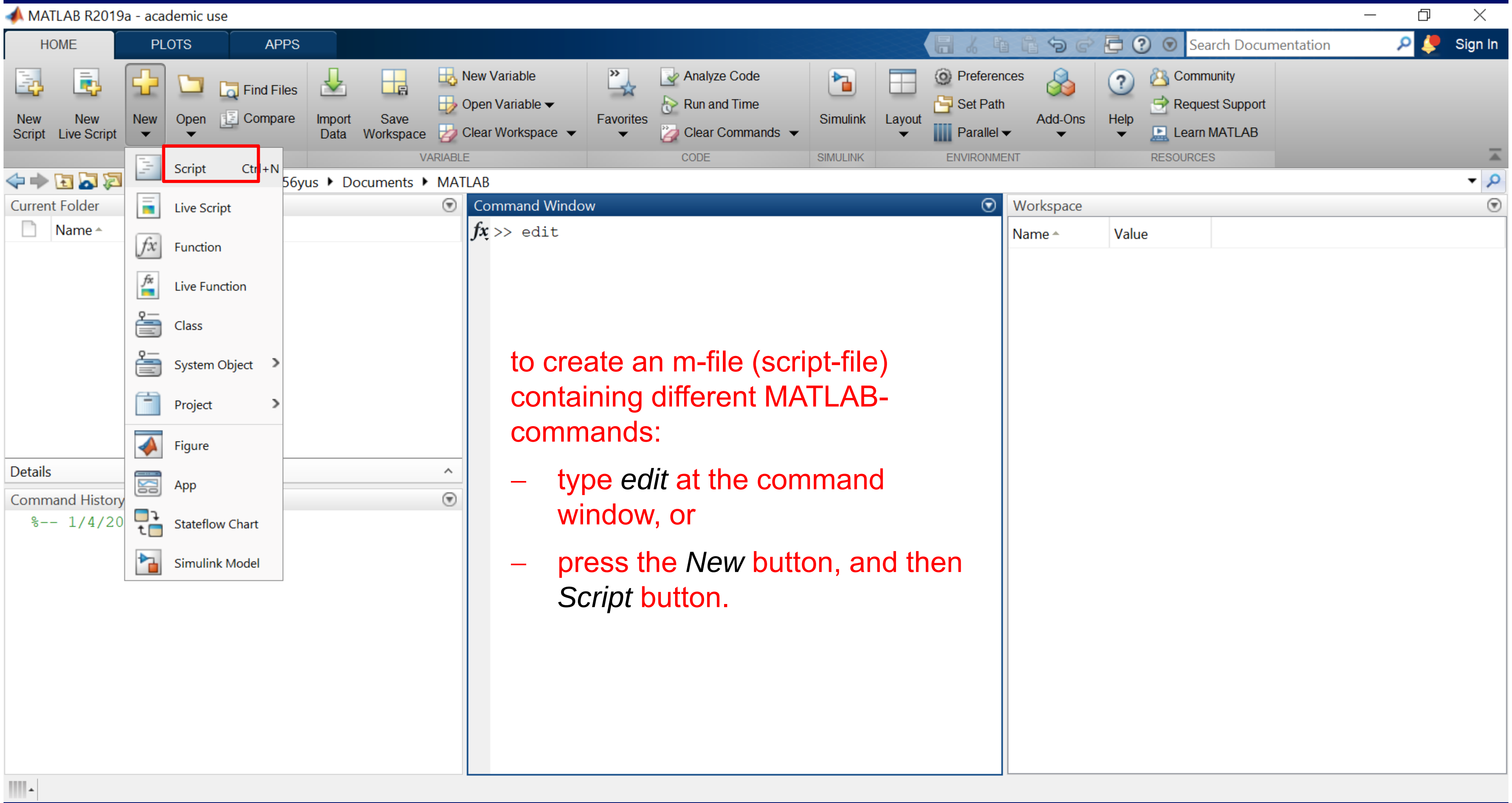
Command/Function	**Meaning**
clc	Clear Command Window
clear	Remove items from workspace
who, whos	List variables in workspace
cd	Change working directory
pwd	Display current directory
computer	Identify information about computer on which MATLAB is running
ver	Display version information for MathWorks products
quit	Terminate MATLAB
exit	Terminate MATLAB (same as quit)

Keyboard shortcuts

- The up arrow key:
  - It repeatedly recalls the previously entered commands.
  - Likewise, typing the first characters of previously entered line and pressing the up arrow key displays the full command line.
- The Tab Key helps to input the MATLAB-functions names
- The semicolon symbol at the end of a line suppresses the screen output.

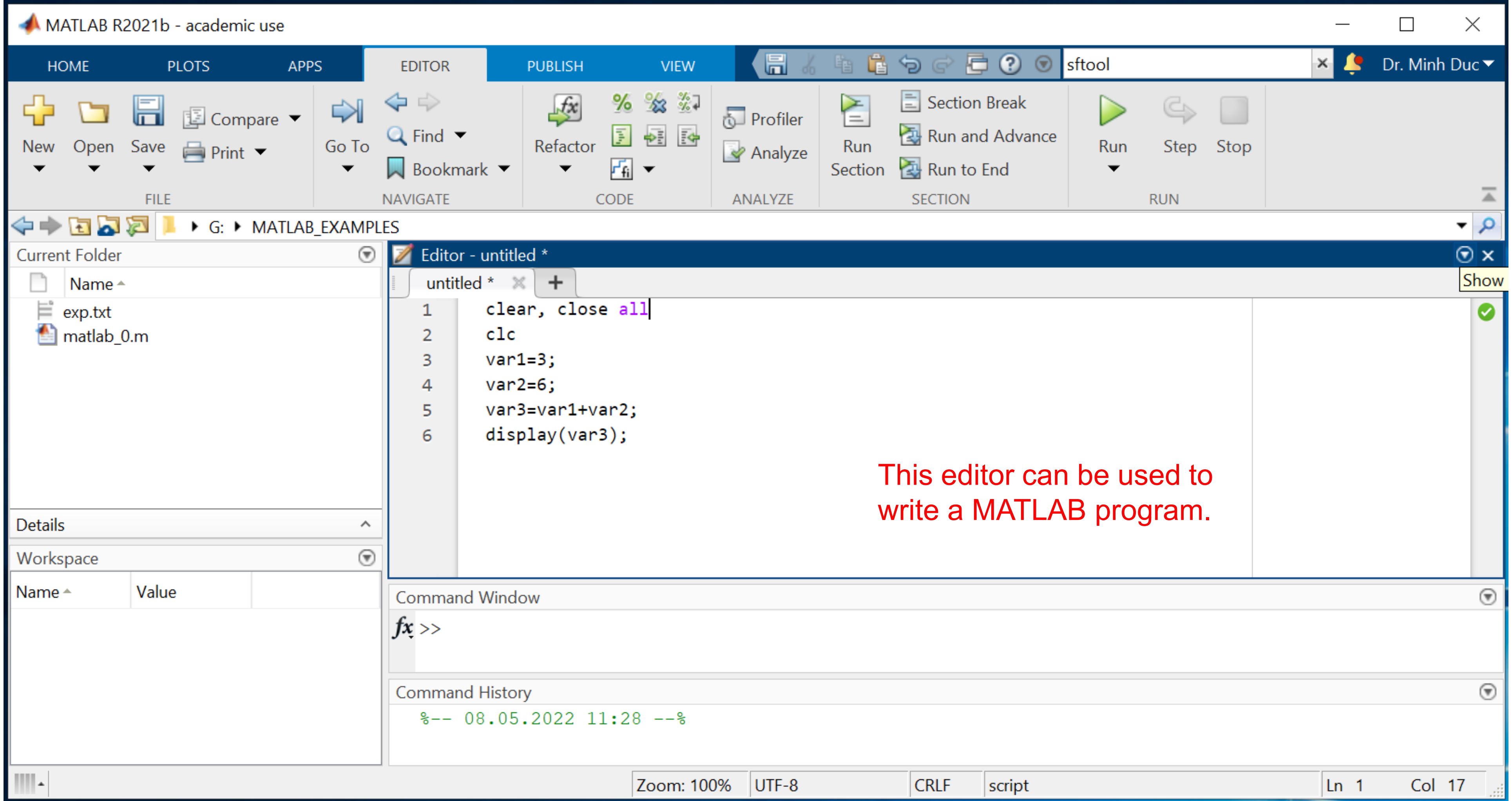
Working with a script-file (m-file)

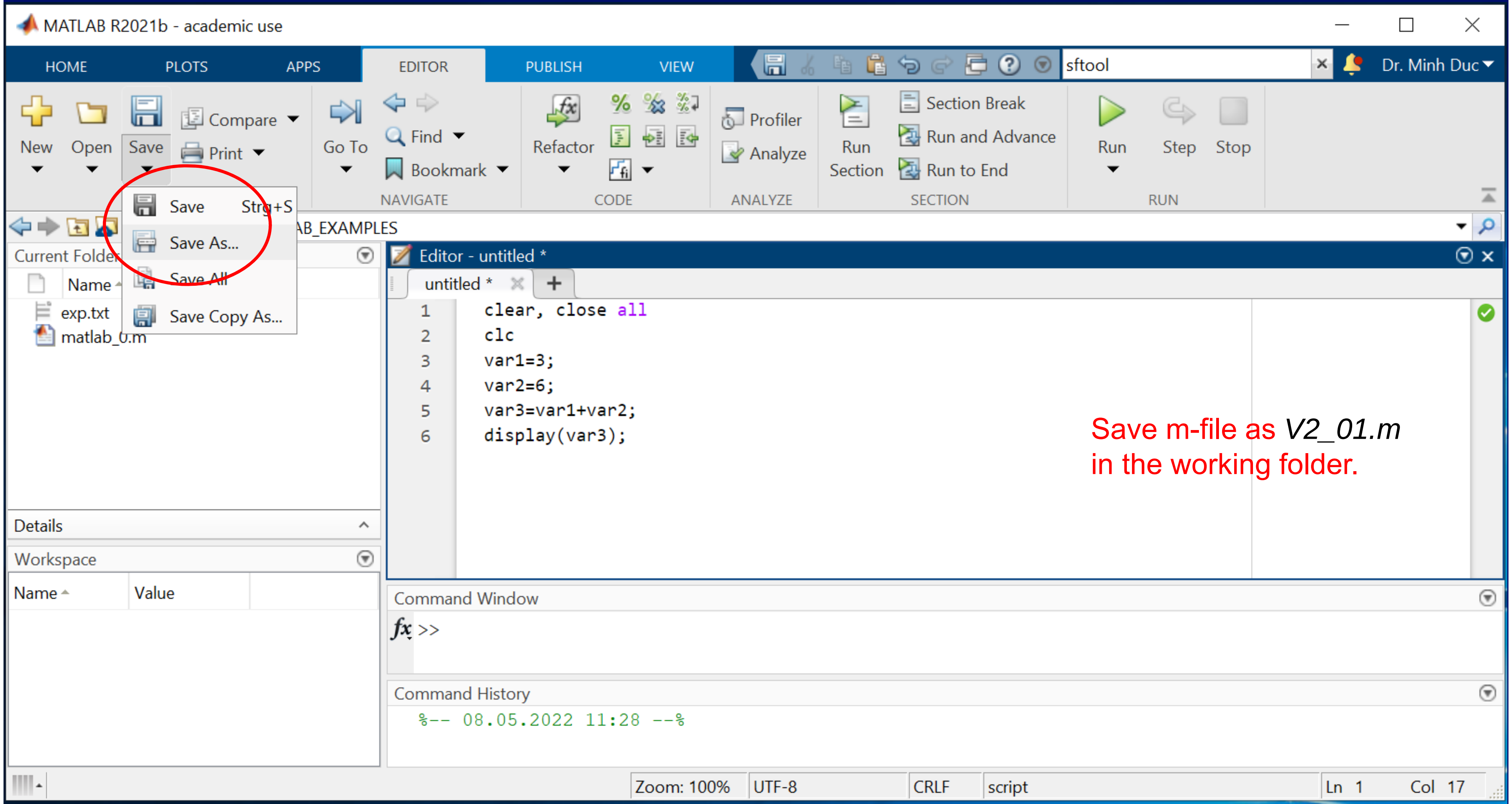
1. Create a file with a list of commands (called as script-file or m-file),
2. Save the file, and
3. Run the file.

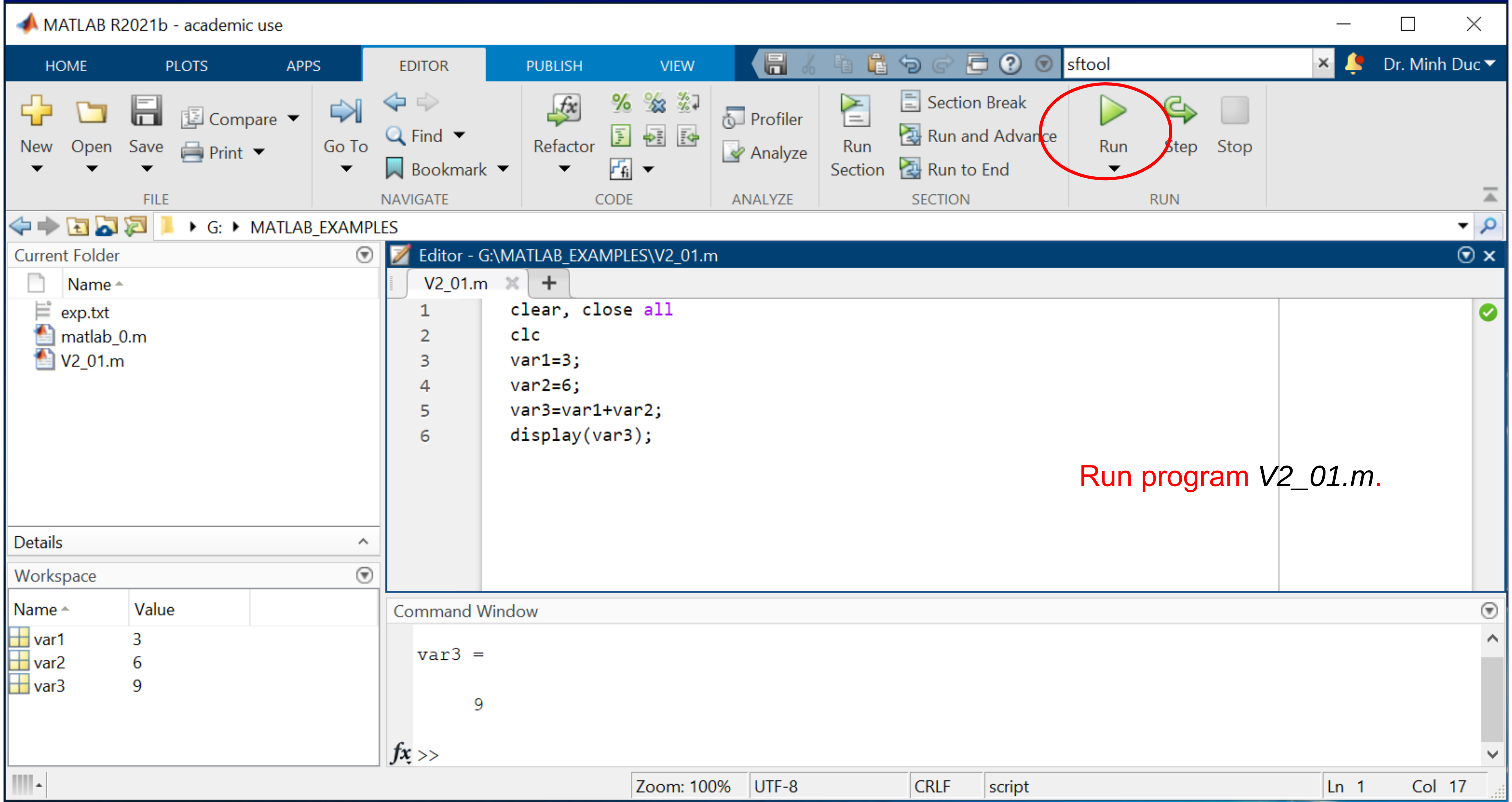


to create an m-file (script-file) containing different MATLAB-commands:

- type *edit* at the command window, or
- press the *New* button, and then *Script* button.


MATLAB R2021b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
sftool
Dr. Minh Duc
New
Open
Save
Compare
Print
FILE
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Profiler
Analyze
ANALYZE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
G: ▸ MATLAB_EXAMPLES
Current Folder
Name
exp.txt
matlab_0.m
Details
Workspace
Name
Value
Editor - untitled *
untitled *
Show
clear, close all
clc
var1=3;
var2=6;
var3=var1+var2;
display(var3);
This editor can be used to write a MATLAB program.
Command Window
fx >>
Command History
%-- 08.05.2022 11:28 --%
Zoom: 100%
UTF-8
CRLF
script
Ln 1
Col 17


MATLAB R2021b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
sftool
Dr. Minh Duc
New
Open
Save
Compare
Print
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Profiler
Analyze
ANALYZE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
Save Strg+S
Save As...
Save All
Save Copy As...
Current Folder
Name
exp.txt
matlab_0.m
Details
Workspace
Name
Value
Editor - untitled *
untitled *
clear, close all
clc
var1=3;
var2=6;
var3=var1+var2;
display(var3);
Save m-file as V2_01.m
in the working folder.
Command Window
>>
Command History
%-- 08.05.2022 11:28 --%
Zoom: 100%
UTF-8
CRLF
script
Ln 1
Col 17


MATLAB R2021b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
sftool
Dr. Minh Duc
New
Open
Save
Compare
Print
Go To
Find
Bookmark
Refactor
Profiler
Analyze
Run Section
Section Break
Run and Advance
Run to End
Run
Step
Stop
FILE
NAVIGATE
CODE
ANALYZE
SECTION
RUN
G: MATLAB_EXAMPLES
Current Folder
Name
exp.txt
matlab_0.m
V2_01.m
Editor - G:\MATLAB_EXAMPLES\V2_01.m
V2_01.m
clear, close all
clc
var1=3;
var2=6;
var3=var1+var2;
display(var3);
Run program V2_01.m.
Details
Workspace
Name
Value
var1
3
var2
6
var3
9
Command Window
var3 =
9
>>
Zoom: 100%
UTF-8
CRLF
script
Ln 1
Col 17

Basic variables

To declare a MATLAB-variable, type in a variable name and specify its value.

Name:

- May contain characters, numbers and symbols
- No numbers or symbols in front of them.

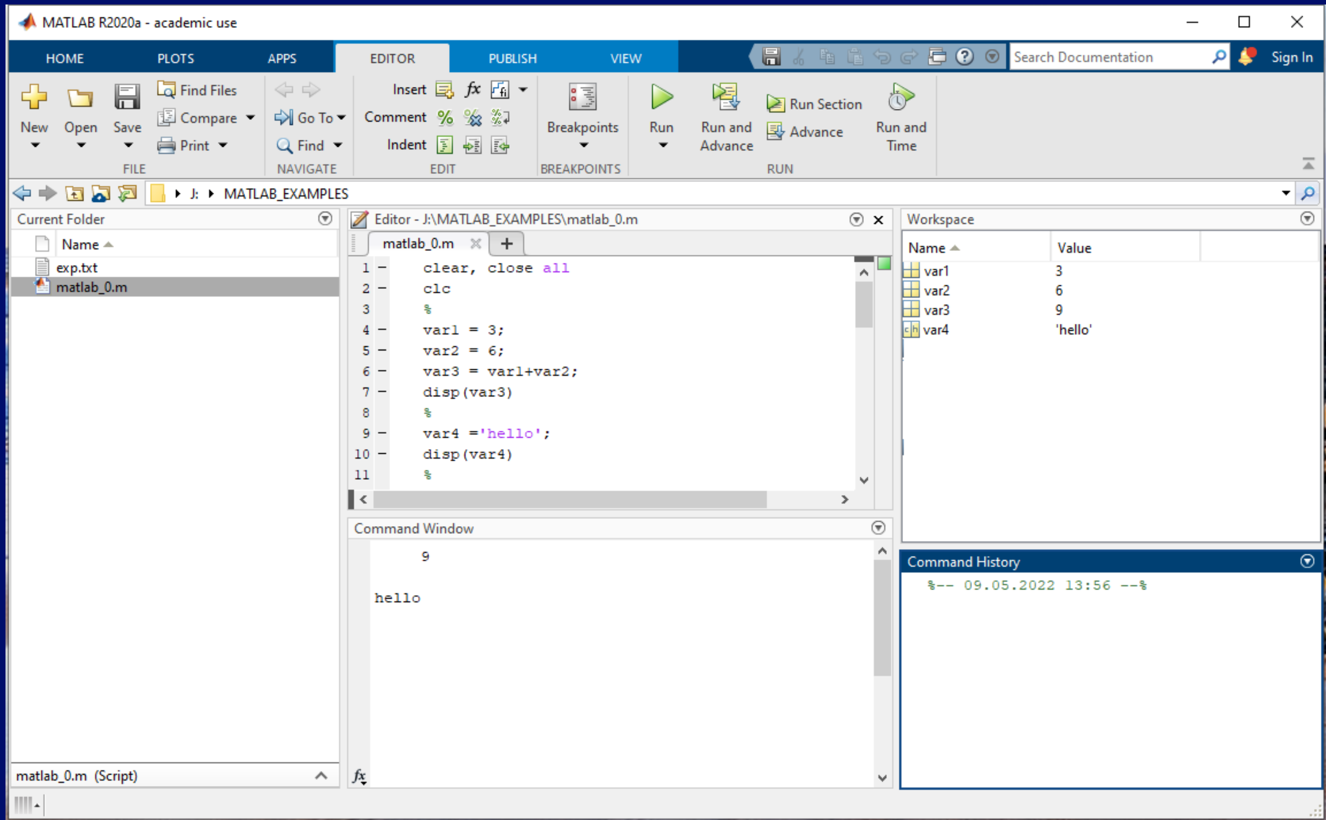
  Example of illegal variable names: *1var* ; *#aaa*
- MATLAB makes a difference between capital or small letters.

Value:

- Three types of variables values: numeric, binary logical and string
- Three forms: scalar, matrix, multidimensional array

Scalar variables

- Single value
- MATLAB will decide on the data type automatically, so you don't have to declare its data type.



– Predefined scalar variables:

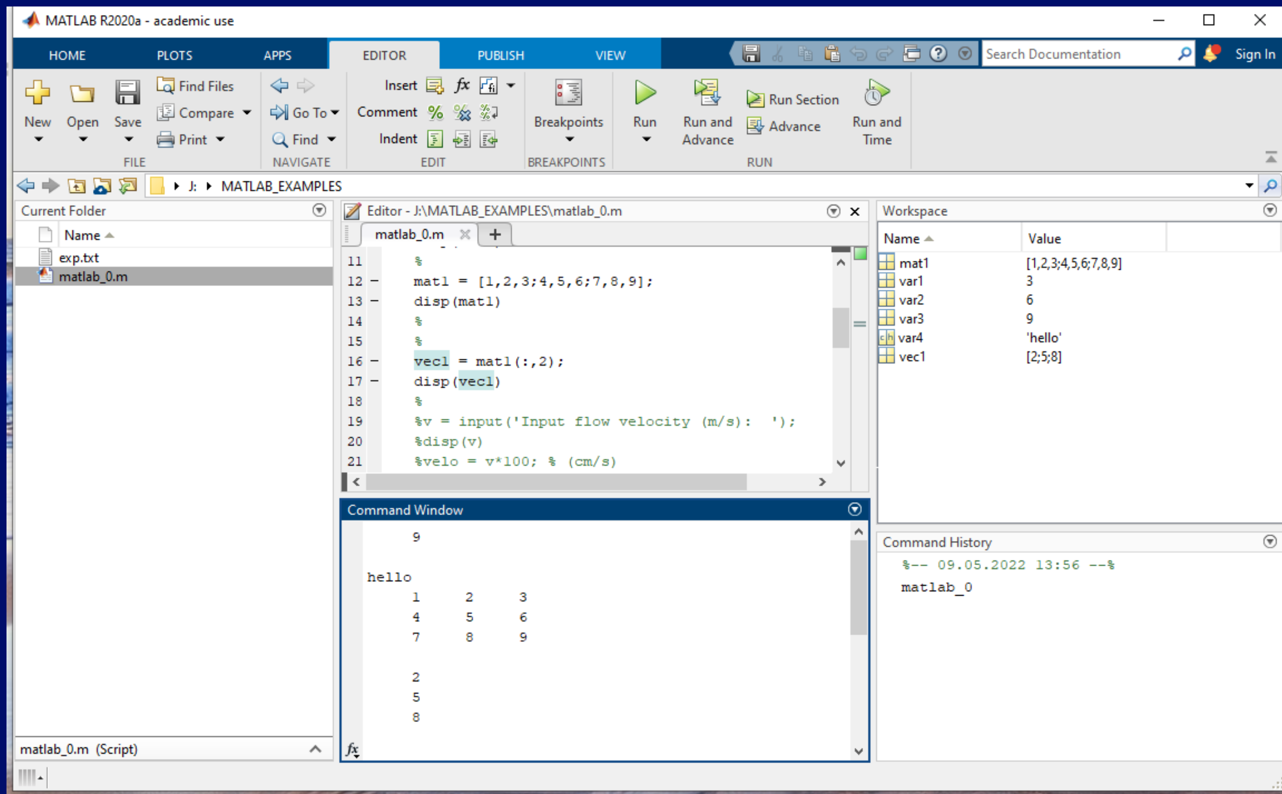
In MATLAB some names have been reserved for specific variables. Creation of a new variable with the name of a predefined variable should be avoided.

Name	Variable
Inf	∞
Eps	2.2204e-16
Pi	3.1416
NaN	Undefined

Matrix variables

- Matrix elements can be considered as a two dimensional array.
  - Its size defined by the number of row and column: n_r-by-n_c
  - Each element considered as a variable with a single value and two indexes.
  - The element values in a matrix variable defined in square brackets.
- To create a matrix, use the comma to separate each value in a row, and a semicolon to enter the value for a new row.

- Scalars and vectors are special form of matrices.



Declaring & manipulating matrix variables

Create a matrix of zeros with size *n*-by-*m*:

- **matrixName = zeros**(*n*, *m*)

Create a matrix of ones with size *n*-by-*m*:

- **matrixName = ones**(*n*, *m*)

Create a matrix of Random Numbers (between 0 and 1) with size *n*-by-*m*:

- **matrixName = rand**(*n*, *m*)

Create a unit diagonal matrix with a size *n*-by-*m*:

- **matrixName = eye**(*n*, *m*)

Access a specific value inside a matrix:

- **matrixName**(*rowNumber, colNumber*)
- Example: access a value inside row *2* and column *3* of **matA** and then assign this value to the scalar variable **scaB** = **matA**(2,3)

Access a whole row / column of a *n*-by-*m* matrix:

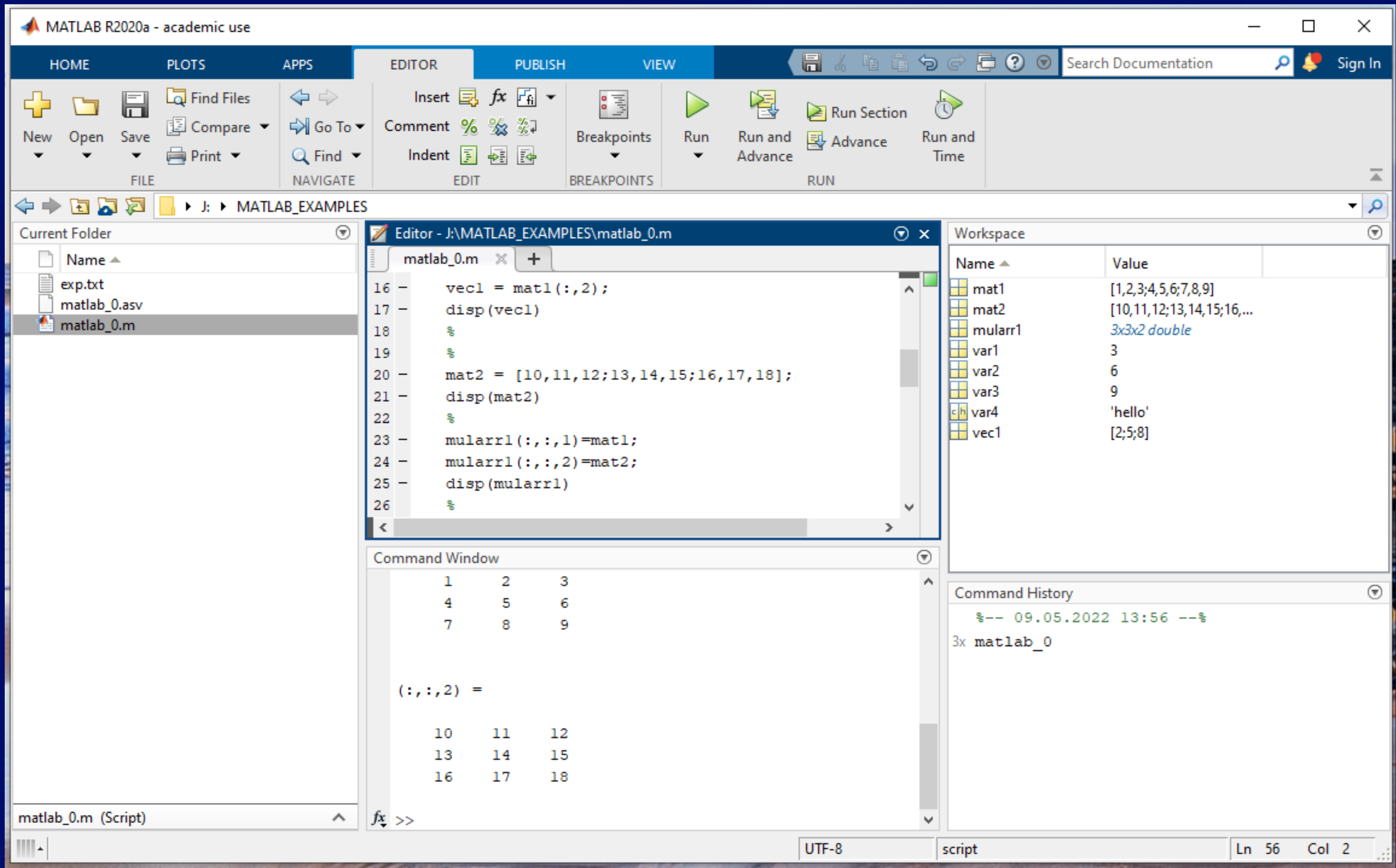
- Access all elements of the row number *rn* of **matA** and then assign these values to the row vector variable **rV** = **matA**(*rn, :*)
- Access all elements of the column number *cn* of **matA** and then assign these values to the column vector variable ***cV*** **= matA**(: , *cn*)

Get information of a matrix:

- **max**(*matA*)
- **min**(*matA*)
- **size**(*matA*)

Multidimensional arrays

- A multidimensional array in MATLAB® is an array with more than two dimensions (m≥3).
  - Each element considered as a variable with a single value and m indexes.
  - Its size defined by the maximal number of each index: $n_1 \times n_2 \times \ldots \times n_m$.



Basic operators

Relational Operators

Symbol	Role
==	Equal to
~=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Arithmetic Matrix Operators

Symbol	Role
+	Addition
-	Subtraction
.*	Element-wise multiplication
*	Matrix multiplication
./	Element-wise right division
/	Matrix right division
.\	Element-wise left division
\	Matrix left division
.^	Element-wise power
^	Matrix power
'	Matrix Transpose

Logical Operators

<table>
<tr><th>Symbol</th><th>Role</th></tr>
<tr><td>&</td><td>Logical AND</td></tr>
<tr><td>|</td><td>Logical OR</td></tr>
<tr><td>&&</td><td>Logical AND (with short-circuiting)</td></tr>
<tr><td>||</td><td>Logical OR (with short-circuiting)</td></tr>
<tr><td>~</td><td>Logical NOT</td></tr>
</table>

Conditional operators

if … else … end

- Evaluates an *expression*, and executes a group of *statements* depending on the logical value of the expression.
- Syntax

  if *expression*

  1-statements (when the expression's value is true)

  else

  2-statements (when the expression's value is false)

  end
- Extended: if … elseif … … else … end
- An *expression* can include relational operators and logical operators.

```
% EXAMPLE: Determine if a value falls within a specified range
x = 10;
minVal = 2;
maxVal = 6;

if (x >= minVal) && (x <= maxVal)
    disp('Value within specified range.')
elseif (x > maxVal)
    disp('Value exceeds maximum value.')
else
    disp('Value is below minimum value.')
end
```

switch … case … … otherwise … end

- Executes one of several groups of statements by comparing *switch_expression* to *i-case_expression* and choosing a true case.
  - The switch block tests each case until one of the case expressions is true.
- Syntax

  switch *switch_expression*
  case *1-case_expression*
  1-statements (when 1-case is true)
  case *2-case_expression*
  2-statements (when 2-case is true)
  …
  otherwise
  other-statements (for other cases)
  end

EXAMPLE:

Display different text conditionally, depending on a value entered at the command prompt.

```
n = input('Enter a number: ');
switch n
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```

for … end

- Repeats a set of statements in a loop for a specified number of times.
- Syntax

  for *index = iniVal : step : endVal*
  statements
  end

- Increments *index* variable from *iniVal* to *endVal* by the positive *step* on each iteration, and repeat execution of *statements* until *index* is greater than *endVal*, or
- Decrements *index* when *step* is negative, and repeat execution of *statements* until *index* is less than *endVal*.

```
% EXAMPLE: Creat a Hilbert matrix of order 10
s = 10;
H = zeros(s);

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

while … end

- Repeats the execution of a group of *statements* in a loop while the *expression* is true.
- Syntax

  while *expression*
  statements
  end

```
% EXAMPLE: Use a while loop to calculate factorial(10)
n = 10;
f = n;
while n > 1
    n = n-1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

break and continue statements

Can be used to control the operation of ***while*** and ***for*** loops:

– ***break*** terminates the execution of a loop and passes control to the next statement after the ***end*** of the loop

```
for i = 1:5
    if i == 3;
        break;
    end
    fprintf('i = %d\n',i);
end
disp('End of loop!');
```

– If a ***continue*** statement is executed in the body of a loop, the execution of the current pass through the loop will stop, and control will return to the top of the loop

```
for i = 1:5
    if i == 3;
        continue;
    end
    fprintf('i = %d\n',i);
end
disp('End of loop!');
```

Input / output operators

Input

1. Request user input:

- Syntax

  x = **input**(*prompt*) - numeric input

  str = **input**(*prompt* , 's') - string input

- It displays the text in *prompt* and waits for the user to input a value and press the **Return** key.
- The user can enter expressions, e.g. *pi/4* or *rand(3)*, and can use variables in the workspace.

Example

v = **input** ('Input flow velocity')

2. **Create dialog box to gather user**:

- Syntax

  answer = **inputdlg** (*prompt*,*dlgtitle*,*dims*)

  prompt - displaying text; *dlgtitle* - box title;

  dims - height (amount of line) and width (amount of text) of the dialog field that users can enter.

  answer - string value

- It creates a dialog box containing one or more text edit fields and returns the values entered by the user.

Example

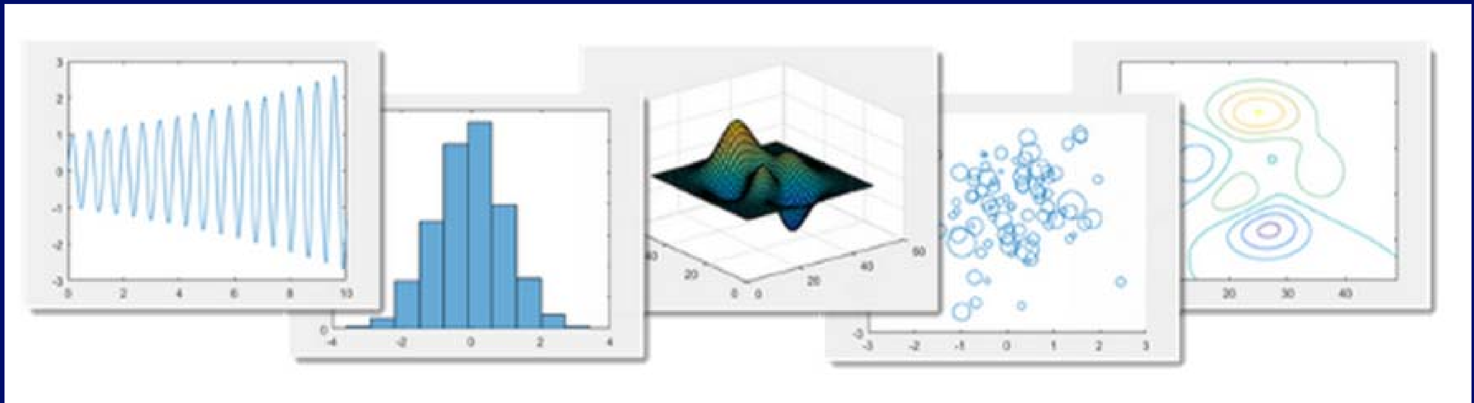
stationA = **inputdlg** ({'Name',‘Depth',‘Velocity'},...

‘Flow condition', [2 50; 1 5; 1 5]))

Output

1. **Text information of a variable in the command window**:
   - MATLAB calls the **display** function, when a statement or expression is not terminated by a semicolon, to show information about an intermediate result, such as the values, size, type, and variable name: **display**(*x*)
   - **disp (x)** shows the value of variable x.
   - When you execute an expression without a semicolon, MATLAB assigns the result to a variable called **ans**, which the **display** function shows in the command window.

2. **Graphic Visualization of data and results**:
   - Graphics functions including 2D / 3D plots, images and animation can be applied to visualize data and communicate results.
   - Customize can be either interactively or programmatically.

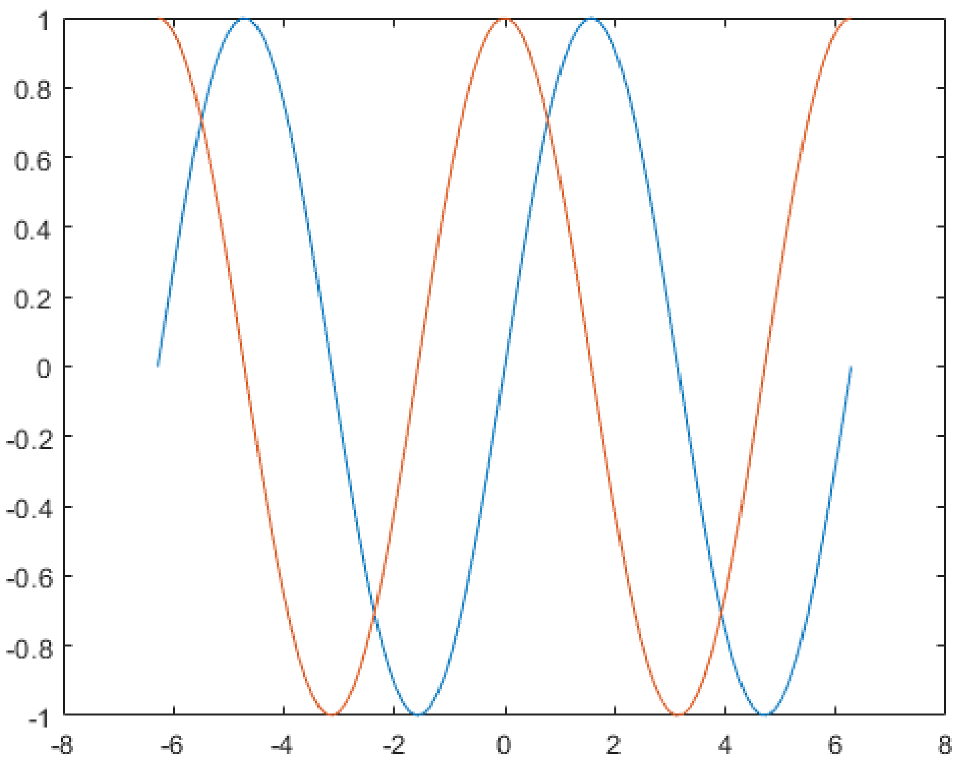


2D plot: Lines or 2D curves in one xy-coordinate plane

```
% EXAMPLE: Plot sine and cosine functions between -2pi and 2pi
x = linspace(-2*pi,2*pi);
y1 = sin(x);
y2 = cos(x);

figure
plot(x,y1,x,y2)
```

Graphic can be customized directly from the figure window or in the m-code.



Symbol	Color (R G B)
r	red (1 0 0)
g	green (0 1 0)
b	blue (0 0 1)
y	yellow (1 1 0)
m	magenta (1 0 1) (a deep purplish red)
c	cyan (0 1 1) (greenish blue)
w	white (1 1 1)
k	black (0 0 0)

Symbol	Line Style
-	solid line (default)
--	dashed line
:	dotted line
-.	dash-dot line

Marker	Description
+	plus sign
o	circle
*	asterisk
.	point
x	cross
s	square
d	diamond
^	upward pointing triangle
v	downward pointing triangle
>	right pointing triangle
<	left pointing triangle
p	pentagram
h	hexagram

Customize line property: color, style and marker.

EXAMPLE:

Plot the sine function over three different ranges using different line styles, colors, and markers.

```
figure
t = 0:pi/20:2*pi;
plot(t,sin(t),'-.r*')
hold on
plot(t,sin(t-pi/2),'--mo')
plot(t,sin(t-pi),':bs')
hold off
```

Commenting and labeling a plot

Example:

Plot *sin* and *cosine* functions in the range of $[-\pi,\pi]$

```
x=-pi:pi/10:pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'k-',x,y2,'m-');
xlabel 'X';
ylabel 'Y';
legend ({'sin(x)','cos(x)'});
text (2,0.2,'y=sin(x)');
text(-2,0.2,'y=cos(x)');
```

```
xlabel (' text')
```

writes a text on x-axis.

```
ylabel (' text ')
```

writes a text on y-axis.

```
title (' title')
```

writes title of a curve.

```
text (x0,y0,'text on the curve')
```

writes text on x0,y0 coordinate.

```
legend ({'comment'})
```

provides a comment on the page.

2D plot: Multiple curves in a page

```
subplot(a,b,c)
```

where a is number of rows,

b is number of columns, and

c is the index of each cell from top left.

Example:

Plot four curves of $y = \cos(x)$, $y = \cos(2x)$, $y = \cos(3x)$, and $y = \cos(4X)$ on a page

```
x=(-4:0.1:4);

y1=cos(x);
y2=cos(2*x);
y3=cos(3*x);
y4=cos(4*x);

subplot (2,2,1); plot(x,y1,'k.');title 'y=cos(x)';
xlabel 'x';ylabel 'y';
subplot (2,2,2); plot(x,y2,'k.');title 'y=cos(2x)';
xlabel 'x';ylabel 'y';
subplot (2,2,3); plot(x,y2,'k.');title 'y=cos(3x)';
xlabel 'x';ylabel 'y';
subplot (2,2,4); plot(x,y2,'k.');title 'y=cos(4x)';
xlabel 'x';ylabel 'y';
```

2D plot: Logarithmic and semi-logarithmic curves

```
semilogy(x,y)
loglog(x,y)
```

Example:

For $x = 1{:}100$ and $y = \exp(x)$, plot a semilog (y-axis) and loglog plot

```
x=1:100;
y=exp(x);
subplot(1,2,1); semilogy(x,y); xlabel 'x'; ylabel 'log
y'; title 'semilog';
subplot(1,2,2); loglog(x,y); xlabel 'log x'; ylabel
'log y'; title 'loglog';
```

3D plot: 3D curves in a xyz-coordinate

```
plot3(x,y,z)
```

Example:

Plot the following curve in range of $t = [-40,40]$

$$f(x,y,z) = \begin{cases} x = \sin(t) \\ y = \cos(t) \\ z = \sin(t) + \cos(t) \end{cases}$$

```
t=(-40:40);
x= cos(t);
y=sin(t);
z=sin(t)+cos(t);
plot3(x,y,z);
xlabel 'x'; ylabel 'y';zlabel 'z';title 'f(x,y,z)';
```

Print graphic or save it to specific file format.

`print`	Print figure or save to specific file format
`saveas`	Save figure to specific file format
`getframe`	Capture axes or figure as movie frame

`savefig`	Save figure and contents to FIG-file
`openfig`	Open figure saved in FIG-file

`orient`	Paper orientation for printing or saving
`hgexport`	Export figure
`printopt`	Configure printer defaults

Input/output data from/into a file

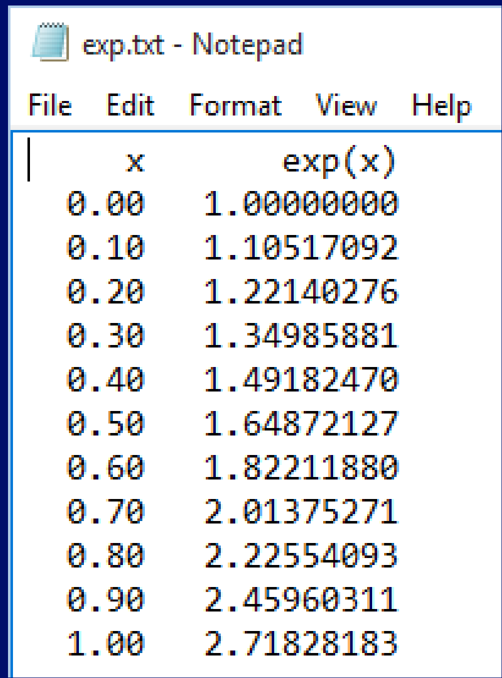
When data is large, the command-line arguments and input/output from/in terminal window are not efficient anymore. In such cases, the most common approach is to let the code read/write data from/into a pre-existing file.

Function	Description
load()	Load MATLAB variables from file into MATLAB workspace
save()	save MATLAB variables from MATLAB workspace into a MATLAB `.mat` file.
fscanf()	Read data from text file
fprintf()	Write data to a text file
dlmread()	Read ASCII-delimited file of numeric data into matrix
dlmwrite()	Write a numeric matrix into ASCII-delimited file
csvread()	Read comma-separated value (CSV) file
csvwrite()	Write values of a matrix into a comma-separated (CSV) file
xlsread()	Read Microsoft Excel spreadsheet file
xlswrite()	write data into a Microsoft Excel spreadsheet file

Function	Description
readtable()	Create table from file
writetable()	Write table to file
imread()	Read image from graphics file
imwrite()	Write image to graphics file
importdata()	Load data from file
textscan()	Read formatted data from text file or string
fgetl()	Read line from file, removing newline characters
fread()	Read data from binary file
fwrite()	Write data to binary file
type()	Display contents of file

Example of writing a simple file

```
x = 0:.1:1;
A = [x; exp(x)];
%
        fileID = fopen('exp.txt','w');
        fprintf(fileID,'%6s %12s\r\n','x','exp(x)');
        fprintf(fileID,'%6.2f %12.8f\r\n',A);
        fclose(fileID);
%
type exp.txt
```



exp.txt - Notepad

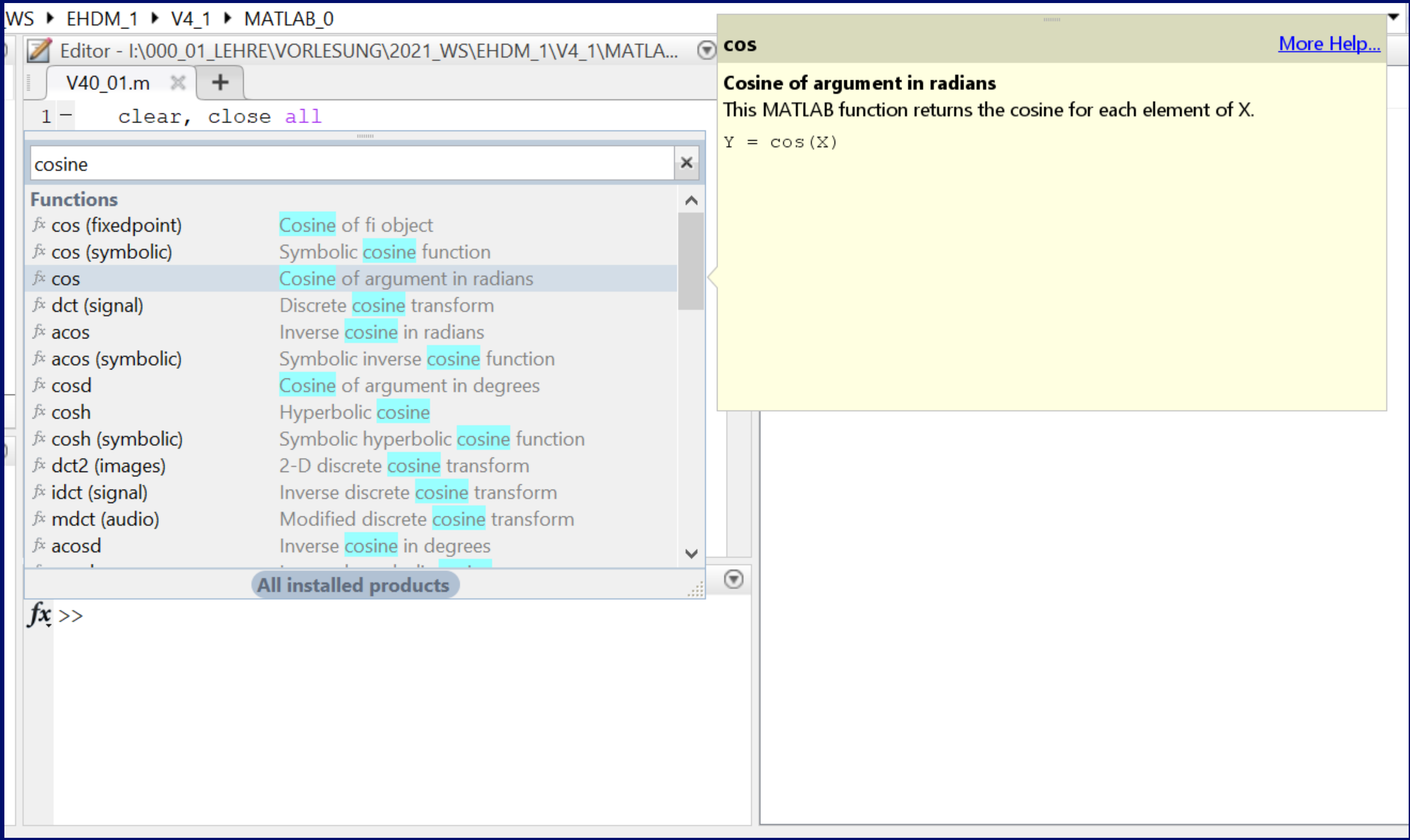
File Edit Format View Help

```
     x        exp(x)
  0.00    1.00000000
  0.10    1.10517092
  0.20    1.22140276
  0.30    1.34985881
  0.40    1.49182470
  0.50    1.64872127
  0.60    1.82211880
  0.70    2.01375271
  0.80    2.22554093
  0.90    2.45960311
  1.00    2.71828183
```

Functions

MathWorks® functions

- A wide variety of predefined mathematical functions in MATLAB, from basic functions to special functions.
- Find the name and description of a MathWorks® function from the Command Window or Editor using the Function browser.



User-defined functions

- A complex program may be divided into several functions.
- These functions can improve readability of the code, as well as promote re-usability of the code.
- The format of a function is:

function *returnValue* = **fcnName**(*inputValue*)

Executable code

end

HOMEWORK 2

Write a MATBAB script to

- Define the function **solv2** for solving a second-order equation,
- Apply this function for the following 2 cases:
  1. a = 7, b = 2, c = 12
  2. a = 4, b = -15, c = 2
- Display the calculated results on the screen by using the statement *display* ,
- Write the results into a text-file.

$$ax^2 + bx + c = 0$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$