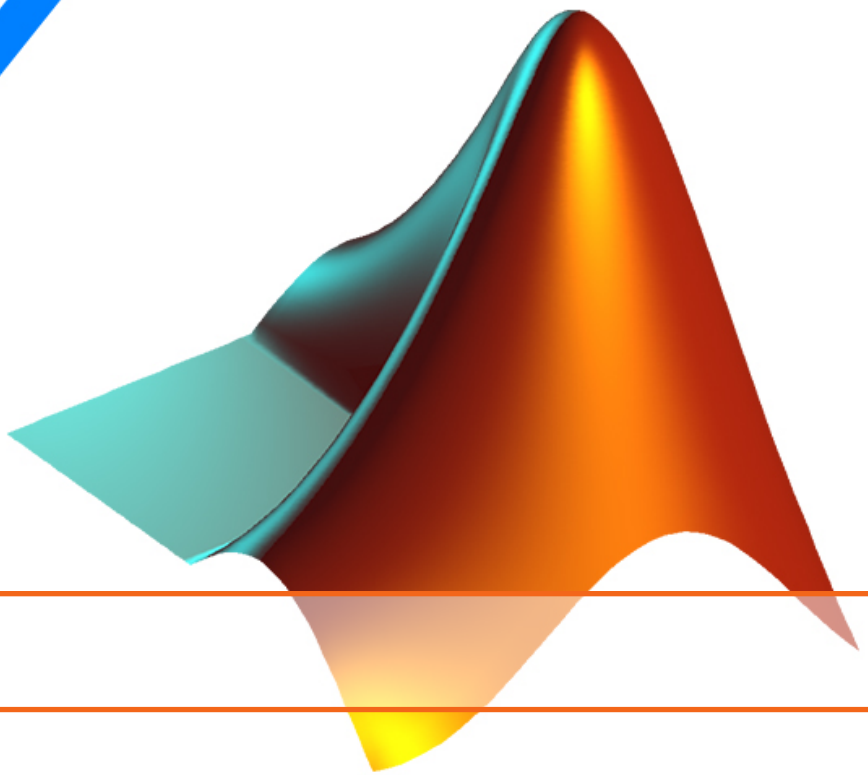
Introduction to the Matlab environment

CYDOMEDIA

WHAT IS MATLAB?

MATLAB PLOT AND MATLAB FUNCTIONS



1. Introduction

Mathematicians, faced with increasingly complex and large-scale problems, encounter significant challenges when trying to solve equations or model complex systems. These problems can be too vast or abstract to solve by hand. That's why programming tools are essential. They allow for automating calculations, handling large datasets, and simulating complex phenomena, greatly facilitating mathematical research and discoveries. These tools enable mathematicians to focus on theoretical analysis while relying on computational power to solve larger and more difficult problems.

Programming tools play a crucial role in the evolution of mathematics. In fact, computational methods and software have significantly transformed the way mathematicians, scientists, and engineers approach mathematical problems. Mathematics often deals with complex problems that may be impractical or impossible to solve by hand. Programming allows for the handling of large data sets, performing symbolic computations, and simulating complex systems that would be unmanageable manually. For example:

- **Numerical Analysis:** Solving equations and optimizing functions in cases where analytical solutions don't exist.
- **Simulation and Modeling:** Tools like *MATLAB*, *Scilab* and *Python* (with NumPy and SciPy), and R are used to simulate real-world phenomena in physics, biology, economics, etc.

Programming tools **(MATLAB, Scilab and Python)** are very important for mathematics students. They help by:

1. Simplifying complex calculations: They solve equations and handle complex math problems easily.
2. Visualization: They allow students to plot graphs and visualize mathematical concepts.
3. Practical learning: Students can experiment with numerical methods and apply them to real-world problems. In short, *MATLAB* and *Scilab* are essential for math students, as they simplify calculations, simulations, and modeling.

This booklet serves as a support for the **Programming Tools (Matlab)** course in the second year of the Bachelor's degree in Mathematics and Technology Sciences (ST), Bachelor's degree in Material Sciences (SM) (Physics and Chemistry), and engineering training programs (ST, ME, etc.). The

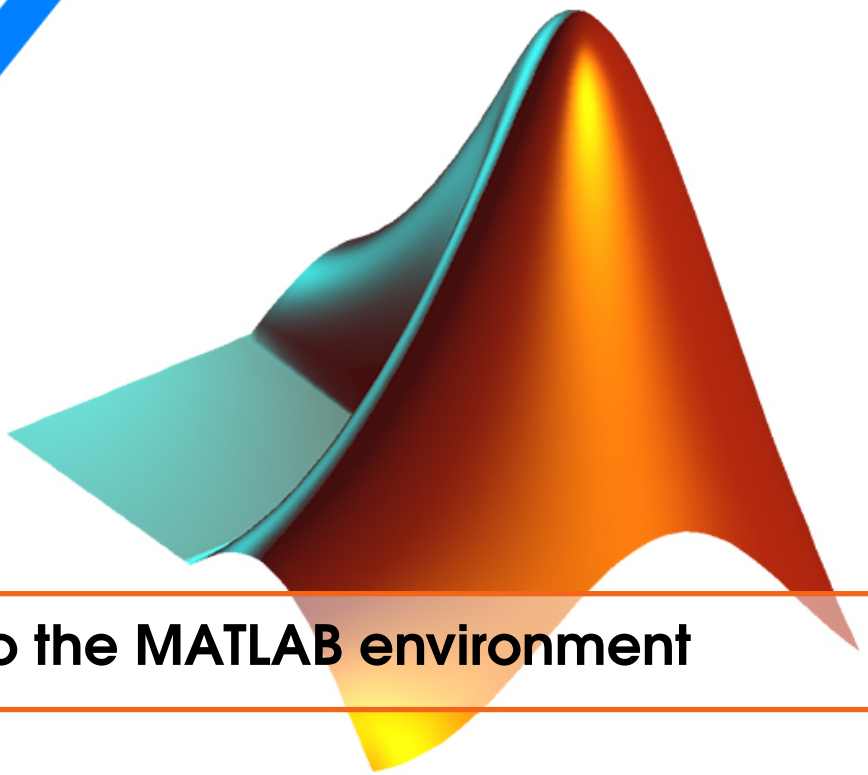
objective of this course is to familiarize students with a set of tools (algorithms) aimed at efficiently solving various problems.
The chapters are illustrated with application examples, and a series of exercises is provided at the end of each chapter. It covers: Introduction to the **MATLAB** environment, Vectors and matrices, Introduction to programming with **MATLAB**, and Graphs and data visualization in **MATLAB**.

Given the time allocated to this unit (1 lecture + 1 practical session) per week, the short duration of the third semester (S3), and the fact that the students are not specialists in computer science, we have chosen not to go into too much detail. Interested readers may refer to the literature for further details.

We have focused our efforts on **MATLAB** in order to enable students to implement it with their instructors on their computers during the practical sessions (TP).
In preparing this handout, we have wisely used a set of references, presented at the end of the document, to deepen the topics discussed. Finally, please let me know if you spot any errors, whether in the content or in the format of this handout.

CYDOMEDIA

WHAT IS
MATLAB?
MATLAB PLOT AND
MATLAB FUNCTIONS



2. Introduction to the MATLAB environment

Introduction

The purpose of this course is to guide you step by step in discovering the basic features of the **MATLAB** software for beginners. It helps you get started with **MATLAB**: understanding what **MATLAB** is, learning some commands, writing your first programs, and plotting graphs.

2.1 What is MATLAB?

MATLAB (short for MATrix LABoratory) is a scientific computing software based on matrix calculations. It is an interactive development environment, developed and marketed by the American company MathWorks.

MATLAB currently offers numerous toolboxes containing specialized functions that allow users to solve specific classes of problems within the **MATLAB** environment. It also provides graphical visualization tools and can be considered a programming language tailored to scientific problems.

MATLAB can be installed on multiple platforms, particularly Windows, Macintosh, etc. Additionally, there are other scientific computing software programs that function similarly to **MATLAB**, such as Octave and Scilab.

2.2 MATLAB environnement

Currently **MATLAB** is at version **MATLAB R2025a** and at startup it displays several windows. Depending on the version you can find the following windows:

- **Current Folder:** indicates the current directory and existing files.

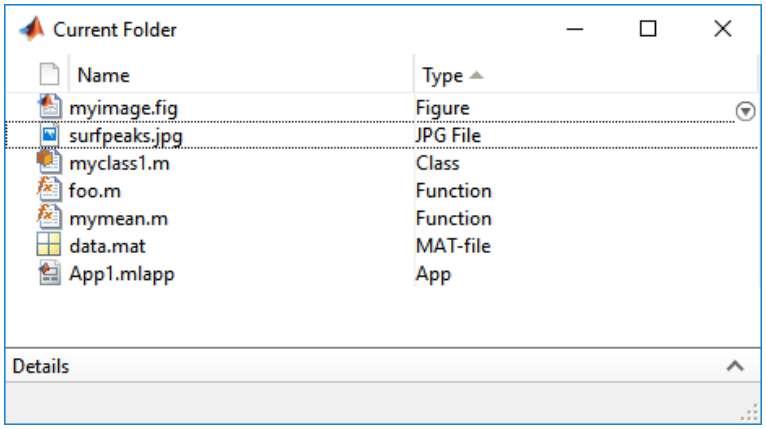


Figure 2.2.1: *Current working folder in **MATLAB**.*

- **Workspace:** indicates all existing variables with their types and values. Since MATLAB is based on matrices, all variables consist of multiple dimensions:
  - A scalar is a 11 matrix.
  - A vector is a matrix of size $1n$ or $n1$.
  - A matrix of size nm.

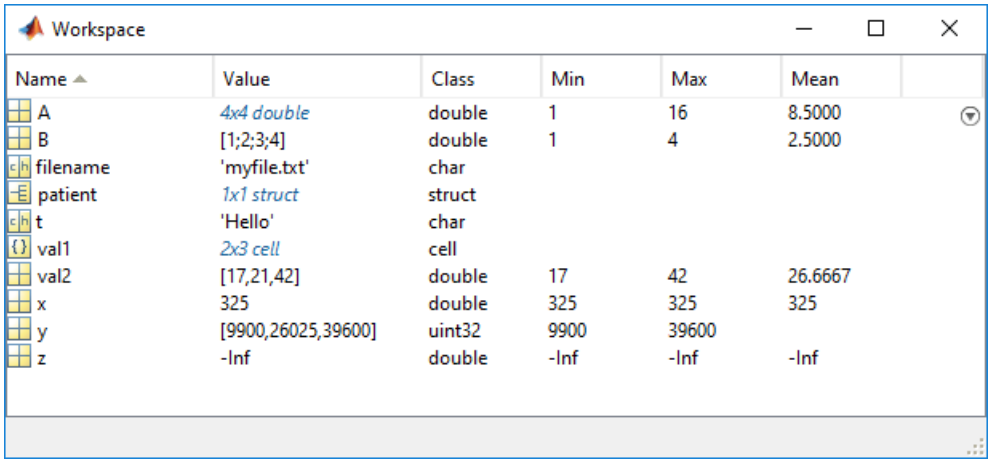


Figure 2.2.2: *Screenshot of the **MATLAB** Workspace.*

- Command History: keeps track of all commands entered by the user.

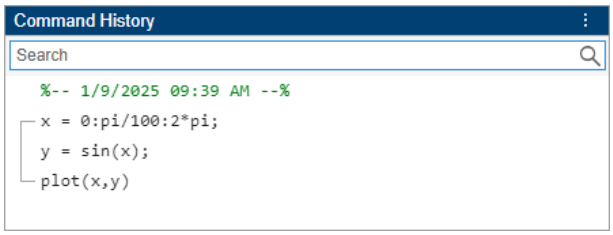


Figure 2.2.3: *Screenshot of the **MATLAB** Command History.*

- **Command Window:** we use to formulate our expressions and interact with **MATLAB**, and this is the window we use throughout this chapter.

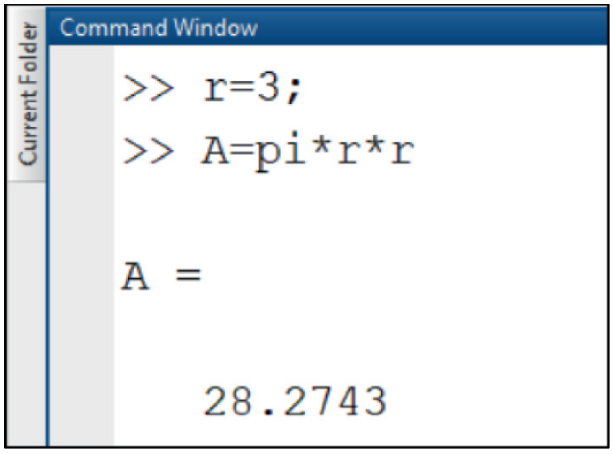


Figure 2.2.4: *Screenshot of the **MATLAB** Command Window.*

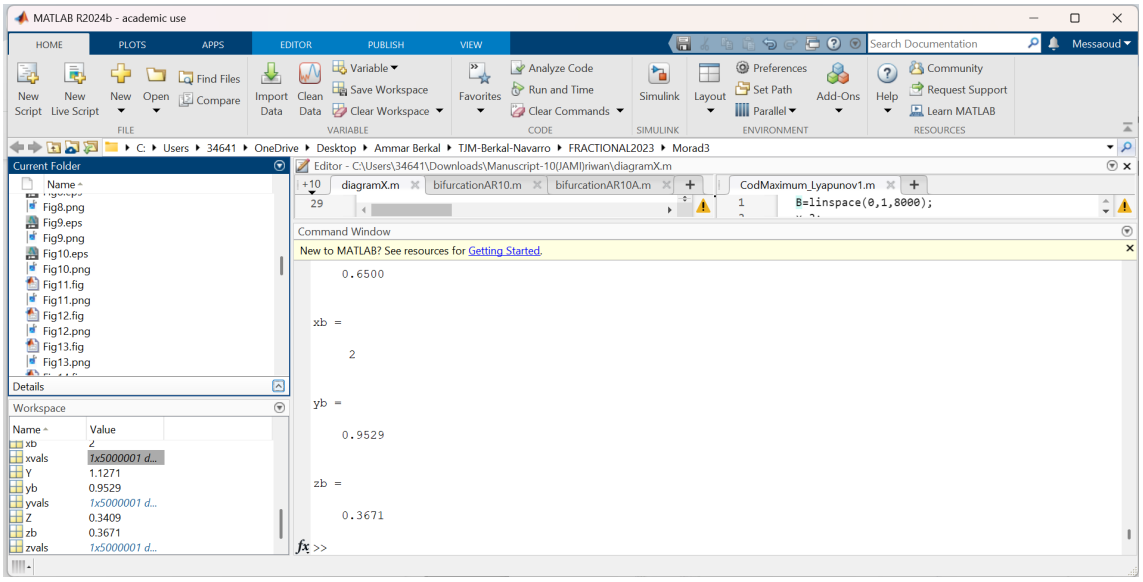


Figure 2.2.5: *The MATLAB interface.*

You can also find the following on the **MATLAB** interface:

- **Editor/Debugger:** It must be used to work with M-files. This window is not part of the basic **MATLAB** interface and opens when you open an M-file or create a new M-file.
  On the toolbar, one button is essential: the RUN button compiles the program, i.e., it executes the commands in the program. It can also be executed with F5.

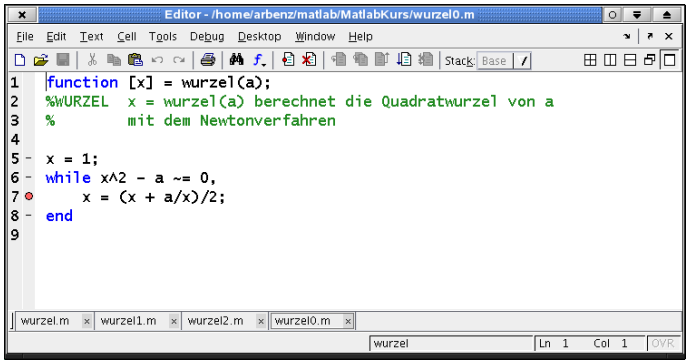


Figure 2.2.6: *The MATLAB editor.*

- **The menu bar and the toolbar:** The menu bar provides access to MATLAB commands. Here are a few examples:
  1. Edit → Clear Command Window: Clears the instructions and/or results visible in the command window.
  2. Edit → Clear Command History : Clears the previously stored commands.
  3. Edit → Clear Workspace : Clears the stored variables from memory.
  4. View : Determines the visual aspects of the different windows.

  The toolbar provides quick access to certain commands.
- **M-Files:** To avoid retyping a series of commands, you can create a **MATLAB** program, known as an "M-file," with the name derived from the ".m" extension of these files. Using the **MATLAB** editor, you create a text file that contains a series of **MATLAB** commands. To create an M-file, go to the menu **File** → **New** → **M-File** or click on the blank page icon. The file is typically saved in the current directory. Once the file is saved (e.g., under the name filename.m), you can call it in **MATLAB** using the command:

  » filename

  The commands stored in the **M-file** will then be executed, and the results will appear in the **Command Window**. If you need to modify your series of commands, simply edit the corresponding line in the **M-file** and re-run the **M-file** by typing the file name in **MATLAB** again (try the ↑ key).

  Other ways to execute the **M-file** include clicking on the Run button, going to the **Debug** → **Run** menu in the **Editor/Debugger**, or pressing **F5**. **M-files** prevent you from having to repeatedly type the same commands and allow you to save your instructions, commands, and calculations. This is the recommended procedure for your practical assignments.

  Exercise 2.1 **Creating an M-File-Procedure:**
  - Create a file using the button or the menu.
  - Write some instructions. For example, write the following instructions:
    A=8
    B=7
    C=A+B
  - Save the M-file in the current directory. If you don't, MATLAB will prompt you to save it before compiling.
  - Return to MATLAB: the results should appear in the command window. These results should be:

    `>> A =7`

```
B =9
C =16
```

- **The ";" and "...":** The semicolon at the end of a line tells **MATLAB** not to display the result of the operation on the screen. A common practice is to place semicolons at the end of all lines and remove some of them when something isn't working as expected in our program, in order to see what's going on.
  In the editor/debugger as well as in the command window, the use of **"..."** is useful to continue the current instruction onto the next line.

  ■ **Example 2.1** The two instructions are identical except for the semicolon at the end of the second one 2e.

```
>> B = pi *3^2
B =
28.274
>> B = pi * 3^2;
>>
```

  The following two instructions give the same result.

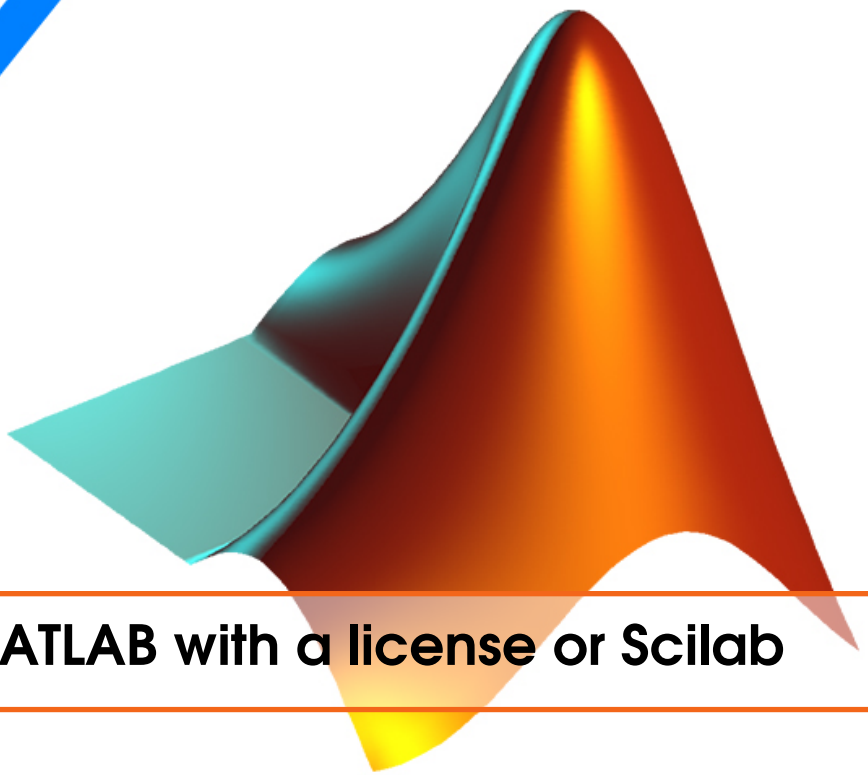
```
>> A = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8
+ 9
A =
45
>> A = 1 + 2 + 3 + 4 + 5 ...
+ 6 + 7 + 8 + 9
A =
45
```

  ■

- **Help:** In addition to the basic features of **MATLAB**, a vast library of functions (called 'toolboxes' in **MATLAB** language) is available to you. To get a list of available function families, enter the command `help`. To see the list of functions within a specific family, you can enter `help matfun`, for example, to view the list of matrix functions. To get information on a specific function, simply use the `help` command followed by the function name, such as `help sin` to get information on the sine function. If the function has not been compiled to improve execution speed, you can view the source code by entering `type arrow`, for example.

CYDOMEDIA

WHAT IS
MATLAB?
MATLAB PLOT AND
MATLAB FUNCTIONS



3. Numbers in MATLAB with a license or Scilab

3.1 First interaction with MATLAB:

The easiest way to use MATLAB is to write directly to the command window (Command Window) right after the cursor (prompt) »
MATLAB can be seen as an extremely powerful calculator. Simple operations can be entered directly, and the result is obtained by pressing the "Enter" key.
To calculate a mathematical expression just write it like this:

```
>> 7+6                Then click on the Enter key to see the result
ans =
13
```

If we want an expression to be calculated but without displaying the result, we add a semicolon **';'** at the end of the expression as follows:

```
>> 7+6 ;
>>
```

To create a variable we use the simple structure: 'variable = definition' without worrying about the type of the variable. For example:

```
>> a=10 ;
>>u=cos(a) ;
>>v=sin(a) ;
>>u^2+v^2
ans =
1
>> ans+11
ans =
12
>>
```

It is possible to write several expressions in the same line by making them separated by commas or semicolons. For example:

```
>> 7+6, 2*4-1, 12-2
ans =
13
ans =
7
ans =
10
>> 5+6; 2*4-1, 12-2;
ans =
7
>>
```

The name of a variable must contain only alphanumeric characters or the symbol **'_'** (underscore) and must start with an alphabet. We must also pay attention to capital letters because the MATLAB is case-sensitive (A and a are two different identifiers). The basic operations in an expression are summarized in the following table:

Operation	meaning
+	addition
−	subtraction
*	multiplication
/	division
\	Left division (or reverse division)
^	power
'	The transposed
(and)	Parentheses specify the order of evaluation

To see the list of variables used, either look at the window '**Workspace**' we use the commands either '**whos**' or '**who**':
whos gives a detailed description (variable name, type and size), but **who** just give the names of the variables.
For example, in this course we used 3 variables a, u and v:

```
>> who
Your variables are:
a      ans      u      v
>> whos
Name   Size   Bytes   ClassAttributes
a      1x1      8       double
ans    1x1      8       double
u      1x1      8       double
v      1x1      8       double
```

3.2 The numbers in MATLAB:

MATLAB uses conventional decimal notation, with an optional decimal point '.' and the sign '+' ou '–' for signed numbers. Scientific notation uses the letter 'e' to specify the power scale factor of 10. Complex numbers use characters 'i' and 'j' (indifferently) to design the imaginary part. The

following table gives a summary:

type	Exemples
Integer	5 -83
Real in decimal notation	0.0205 3.1415926
Real in scientific notation	$1.60210e-20$ $6.02252e23$ $(1.60210\times10^{-20}$ $6.02252\times10^{23})$
Complex	$5+3i$ $-3.14159j$

MATLAB always uses real numbers (double precision) to make the calculations, which allows obtaining a calculation accuracy of up to 16 significant digits. But it should be noted the following points:

- The result of a calculation operation is displayed by default with four digits after the decimal point.
- To display more numbers use the command **format long** (14 digits after the decimal point).
- To return to the default view, use the **format short**.
- To display only 02 digits after the decimal point, use the **format bank**.
- To display the numbers as a ration, use the **format rat**.

control	meaning
format short	displays numbers with 04 digits after the decimal point.
format short e	Scientific with 04 digits.
format long	displays numbers with 14 (or 16) digits after the decimal point.
format long e	Scientific with 14 (or 16) digits.
format bank	displays numbers with 02 digits after the decimal point.
format hex	Hexadecimal.
format +	Use the symbols $+$, $-$ and space to display positive, negative, and zero numbers.
format rat	displays the numbers as a rational (a/b).
who	Displays the name of the variables used.
whos	Displays information about the variables used.
clear x y	Deletes the x and y variables.
clear, clear all	Deletes all variables.
clc	Clears the control screen.
exit, quit	Close the MATLAB environment.

■ **Example 3.1**

```
>> 8/3
ans =
2.6667
>>format long
>> 8/3
ans =
2.66666666666667
>>format bank
>> 8/3
ans =
2.67
>>format short
>> 8/3
ans =
2.6667
```

```
>> 7.2*3.1
ans =
22.3200
>>format rat
>> 7.2*3.1
ans =
558/25
>> 2.6667
ans =
 26667/10000
```

■

vpa function can be used to force the compute to have more significant decimals by specifying the desired number of decimals.

■ **Example 3.2**

```
>> sqrt(2)
ans =
1.4142
>> vpa(sqrt(2),50)
ans =
1.4142135623730950488016887242096980785696718753769
```

■

3.3 The main constants, functions and commands:

MATLAB defines the following constants:

Constant	Value
Pi	$\pi = 3.1415...$
I and J	$= \sqrt{-1}$
NaN	Not a Number
exp(1)	$e = 2.7183...$
Inf	∞
Eps	$\varepsilon \approx 210^{-16}$

Frequently used functions include:

function	meaning
sin(x)	The sine of x (in radian)
cos(x)	The Cosine of x (in radian)
tan(x)	The tangent of x (in radian)
asin(x)	The arc sine of x (in radian), (the inverse sine function sin^{-1} of x)
acos(x)	The arc cosine of x (in radian). (the inverse sine function cos^{-1} of x).
atan(x)	The arc tangent of x (in radian), (the inverse tangent function tan^{-1} of x).
sqrt(x)	The real nth square root of number x ($\sqrt[n]{x}$).
nthroot(x,n)	The square root of x ($\sqrt{x}$).
abs(x)	The absolute value of x ($\|x\|$).
exp(x)	The exponential function, which is defined as e^x.

function	meaning
log(x)	Natural logarithm of x ($\ln(x) = \log(x)$).
$\log_{10}$(x)	Logarithm based on 10 of x $\log_{10}(x)$.
imag(x)	The imaginary part of the complex number x.
real(x)	The actual part of the complex number x.
round(x)	Round a number to the nearest integer.
floor(x)	round a number to the smallest integer ($\max\{n \mid n \leq x, n \text{ entire}\}$).
ceil(x)	round a number to the largest integer ($\max\{n \mid n \geq x, n \text{ entire}\}$).

3.4 The priority of operations in an expression:

The evaluation of an expression runs from left to right considering the priority of the operations indicated in the following table:

Operations	Priority (1=max, 4=min)
The parentheses $\wedge$ (and)	1
Power and the transposed $\wedge$ and $'$	2
Multiplication and division $*$ and $/$	3
Addition and subtraction $+$ and $-$	4

For example:

```
5+2*3 = 11 and 2*3^2 = 18
```

Exercise 3.1 Create an x variable and set it to 2, then write the following expressions:

- $3x^3 - 2x^2 + 4x$.
- $e^{\frac{1+x}{1-\sqrt{2x}}}$.
- $\left|\sin^{-1}(2x)\right|$.
- $\dfrac{\ln(x)}{2x^3} - 1$.
- $\cos(x^5) - 52$.

Solution:

```
>> x=2 ;
>> 3*x^3-2*x^2+4*x ;
>> exp(1+x)/(1-sqrt(2*x));
>> abs(asin(2*x));
>> log(x)/(2*x^4)-1 ;
>> cos(x^5)-52 ;
```