MATLAB is an interactive software package which was developed to perform numerical calculations on vectors and matrices. Initially, it was simply a **MAT**rix **LAB**oratory. However, today it is much more powerful.

History

- Developed primarily by mathematician and computer programmer Cleave Moler in 1970s
- In the beginning, MATLAB was not a programming language, it was just a simple interactive matrix system to access matrix routines in the LINPACK and EISPACK packages without the need to write FORTRAN programs
- Released to the public for the first time in February 1979. It had only about 80 functions, primitive graphics and “matrix" was the only data type

- Gained popularity and developed a strong following especially in the math departments
- Cleave Moler, Steve Bangert, and John N. Little created the MATLAB programming language and developed numerous additional features in early 1980s
- Released as a commercial product in 1984, when MathWorks, Inc. was founded by Cleave Moler, Steve Bangert, and John N. Little
- MATLAB has millions of worldwide users (over 4 million active users as of 2020) from various backgrounds, including engineering, mathematics and statistics, physical and life sciences, and economics. More than 5000 colleges and universities around the world use MATLAB for instruction and research

Strengths

- User friendly and easy to learn
- A scripting system that allows users to write and/or modify the software as needed
- Fast, especially when performing matrix operations
- Combines nicely calculations and graphic representations
- A huge number of built-in functions and toolboxes (collections of functions) for specific applications
- Errors are easy to fix
- Provides an interactive development tool for scientific and engineering problems (e.g., in neural networks, machine learning, signal processing, biosciences, etc.)
- Programming style resembling well-known programming environments such as C/C++, FORTRAN, and Java

Weaknesses

- It is not a *general purpose* programming language such as C++, FORTRAN, or Python
- MATLAB commands are specific for MATLAB usage. Many of them do not have a direct equivalent with other programming language commands
- MATLAB is an interpreted language, sometimes slower than a compiled language such as C++

It should be mentioned that there are several software packages that present some similarities to MATLAB, such as: Mathematica, Maple, Scilab, Octave, Freemat, and Julia.

To begin a MATLAB session, click on the MATLAB icon and wait for the prompt (») to appear. (To exit MATLAB, type `exit` or `quit`). You are now in the MATLAB workspace.

MATLAB has a **huge** number of commands and functions. For an overview of the capabilities of MATLAB, type

```
demo
```

Typing

`help <command>` gives complete information about the object `command`

`doc` accesses the main page of the reference manual

`doc <name>` displays documentation for `name`

`type <command>` displays the actual MATLAB code for `command`

`lookfor <keyword>` searches all MATLAB commands for `keyword`

`Ctrl-c` aborts the command which is currently executing (i.e., hold down the control key and type “c”)

Useful commands and suggestions

- `diary <FileName>` keeps track of everything done during a MATLAB session in the file FileName (named by you)
  `diary off` turns off the recording after saving the file
- If you make a typing mistake, just press the **Enter** key until the prompt appears on the screen; then retype the command
- If you want to use again a previous command, you can use the up-arrow key (↑) to scroll back through the commands. When the command is displayed at the prompt, it can be either executed again or modified if needed and executed
- A comma (`,`) separates multiple statements on the same line. The results appear on the screen
- A semicolon (`;`) when this ends a MATLAB command, the result is not printed on the screen. This can also separate multiple statements on the same line
- To continue a long command (or expression) on the next line, type `...`

MATLAB can be used as a scientific calculator. The order in which it performs arithmetic operations is exactly that taught in algebra courses.

Arithmetical Operations:

$a + b$ addition.
$a - b$ subtraction.
$a * b$ multiplication.
a/b division.
$a\hat{}b$ exponentiation (i.e., a^b).

For example, let's suppose you want to calculate the expression $5 - 3 \times 4 + 2^4$. You type it at the prompt (») as follows,

```
» 5-3*4+2^4
ans =
     9
```

Notice that if you do not specify an output variable, MATLAB uses a default variable `ans`, short for *answer*, to store the results of the current calculation. Note that the variable `ans` is created (or overwritten, if it is already existed). To avoid this, you may want to assign a value to a variable or output argument name.
MATLAB variables are created with an assignment.

```
» VariableName = a value (or an expression)
```

For example,

```
» a = 5-3*4+2^4
a =
     9
```

will result in `a` being assigned the value 5-3*4+2^4=9. This variable name can always be used in subsequent calculations. That is, computing $7a$ will result in

```
» 7*a
ans =
     63
```

Scalar Calculations

Variables

In MATLAB, variables

- must start with a letter, and can then be followed by any number of letters, numbers, and/or underscores
- are case sensitive (so `x` is not the same as `X`)
- can contain up to 31 characters (but this is certainly an "overkill")
- do not need to be declared or typed
- should not be the names of built-in functions or the so-called "reserved words"

To display a variable, type it alone at the prompt. For example, typing `a` from above at the prompt

```
» a
a =
     9
```

The following are commands related to variables:

- `who` shows variables that have been defined in the Command Window
- `whos` is a long form of `who`. It lists all the variables in the current workspace, together with information about their size, bytes, class, etc.
- `clear` clears out all variables and functions
  `clear <VariableName>` deletes the variable called VariableName. Other similar commands are
  `clearvars` clears out all variables so they no longer exist, or
  `clearvars <VariableName1 VariableName2 ...>` clears out the variables named VariableName1, VariableName2, and the others that are represented by ...

Predefined Variables:

- `ans` the default variable name when one has not been specified.
- `pi` π.
- `eps` approximately the smallest positive real number on the computer such that $1 + \text{eps} > 1$.
- `Inf` the infinity, ∞
- `realmin` the smallest "usable" positive real number on the computer. This is "approximately" the smallest positive real number that can be represented on the computer (on some computers realmin/2 returns 0).
- `realmax` the largest "usable" positive real number on the computer. This is "approximately" the largest positive real number that can be represented on the computer (on most computers 2*realmax returns Inf).
- `i` the imaginary unit $i = \sqrt{-1}$. Another notation for it: `j`
- `NaN` not a number

Characters and strings in MATLAB are typed in using single quotes, otherwise they would be interpreted as a variable name.

```
» me = 'Good afternoon!'
me =
     'Good afternoon!'
» you = 'Hi!'
you =
     'Hi!'
```

Characters are listed in an order using what is called a *character encoding*. The most common character encoding is the American Standard Code for Information Interchange (ASCII). Standard ASCII has 128 characters, which have corresponding integer values from 0 to 127. In the character encoding, all characters in the computer's *character set* are placed in an ordered list and given corresponding integer values. The character set includes all alphabet letters, digits, punctuation marks, and special characters (such as the **Enter** key).

Computers are Dumb: computers cannot add, subtract, multiply, or divide correctly!

Proof? For example, when one enters

```
» n = 5; ( n^(1/2) )^2 - n
```

MATLAB returns 8.8818e-16 rather than the correct result of 0.

The reason that $(n^{1/2})^2 - n$ can be nonzero numerically is that MATLAB only stores real numbers to a certain number of digits of accuracy. Type

```
» log 10(1/eps)
```

and remember the integer part of the result. This is approximately the maximum number of digits of accuracy of any calculation performed in MATLAB.

Caution: eps and realmin are very different numbers. realmin is approximately the smallest positive number that can be represented on the computer, whereas eps is approximately the smallest positive number on the computer such that 1+eps >1.

To change how the results are printed out, use the `format` command in MATLAB.

- `format short` the default setting.
- `format long` results are printed to approximately the maximum number of digits of accuracy in MATLAB.
- `format short e` results are printed in scientific notation using five significant digits.
- `format long e` results are printed in scientific notation to approximately the maximum number of digits of accuracy in MATLAB.
- `format short g` results are printed in the best of either format short or format short e.
- `format long g` results are printed in the best of either format long or format long e.

Exercise: Use each of these format functions and then type in 1/3 to see how the result is printed out.

MATLAB contains a large number of mathematical functions. Most are entered exactly as you would write them mathematically. For example:

```
sin(1)
cos(2)
asin(1/2)
acos(-0.3)
exp(2)
log(3)
log10(100)
factorial(4)
floor(10.7)
```

return exactly what you would expect. Observe that to apply a function, the name of the function comes first followed by the argument(s) that are passed to the function in parentheses.

As is common in programming languages, the trig functions are evaluated in radians. If you need to use degrees, just add the letter “d” at the end, e.g.,

```
sind(30)
cosd(60)
asind(1/2)
acosd(-0.5)
```

As an example, the value of the expression $y = e^a \cos(\pi) + 7\sqrt{b} + |c| + \log(y)$, for $a = -2$, $b = 4$, $c = -3$, and $y = 1000$ is computed by

```
» a = -2; b = 4; c = -3; y = 1000;
» y = exp(a)*cos(pi)+abs(c)+log10(y)
y =
    5.8647
```

Remark: notice the reassignment of `y`.

MATLAB can handle complex numbers. For example

```
» a=1+i; b=2-3i
» c=a*b
```

produces the answer c=5-i.

The modulus, argument ($\geq -\pi$ and $\leq \pi$ radians), and real part of a complex number a are obtained by

```
abs(a)
angle(a)
real(a)
```

The command

```
imag(a)
```

produces the imaginary part without the i attached, and

```
conj(a)
```

provides the complex conjugate of `a`.