# Hands On: Introduction to MATLAB - Part 2

# Numbers, Arithmetic Operations, Formats and Variables

This is the second MATLAB live script of the collection ***Hands On: Using MATLAB in the 267MI "System Dynamics" course***, devoted to introduce the MATLAB/Simulink environment and tools for solving practical problems related to the topics of the 267MI course, i.e. performance analysis of dynamic systems, parametric estimation, identification of models from data, and prediction of the evolution of dynamic systems.

Use this link to go back to the main live script of the collection.

**Table of Contents**

## Objectives

The aim of this module is to enable you to enter an arithmetic expression in MATLAB and understand the results produced, particularly when these results involve scientific notation, complex numbers and `Inf` and `NaN`.

What you should know by the end of this module:

- the importance of being very precise when entering arithmetic expressions, particularly when brackets are required;
- how to enter numbers using scientific notation and how to change the format of the answer displayed in the MATLAB Command Window;
- how to enter complex numbers and how to find real and imaginary parts;
- the meaning of `Inf` (infinity) and `NaN` (Not a Number);
- what is a valid variable name;
- how to use = to assign a value to a variable and how to clear a variable;
- MATLAB's predefined variables (constants), pi, i, j, eps, etc. ;
- how to examine the value of a variable, how to list the variables in the current workspace, how to get extra information about the variables and finally how to remove them from the current workspace.

## MATLAB as calculator

This is the simplest way of using it: evaluating numerical expressions.

Suppose we want to evaluate the numerical expression

$$4+\sqrt{2}-\left[\sin\left(\frac{\pi}{5}\right)\right]^2+e^2$$

Simply

```
4 + sqrt(2) - sin(pi/5)^2 + exp(2)
```

The result is assigned to the predefined variable `ans`.

## Arithmetic operations

The most common arithmetic operations in MATLAB are

- Addition $a+b$ $\Leftrightarrow$ `a+b`
- Subtraction $a-b$ $\Leftrightarrow$ `a-b`
- Multiplication $a\cdot b$ $\Leftrightarrow$ `a*b`
- Division $a/b$ $\Leftrightarrow$ `a/b`

- Powers $a^b$ $\Leftrightarrow$ a^b

This is just like on a calculator, in spreadsheets (Excel) and most programming languages.

## Order of Operations

In MATLAB (as many other programming languages) operations are performed in the following order:

1. expressions in brackets: `(      )` ;
2. powers: `^` ;
3. multiplication and division: `*` , `/` ;
4. addition and subtraction: `+` , `-` .

Operations of the **same precedence**, for example multiplication and division, are **evaluated from left to right**.

In ordinary written mathematics we sometimes leave out brackets and rely on the fact that an intelligent human will understand where they ought to go. In contrast, computers obey exactly the rules for evaluating expressions. If you are unsure, adding some extra brackets (parentheses) will not hurt.

### A few Examples

```
1+2*3    % Which operation has higher precedence?

(1+2)*3  % And now?
```

Remember that operations of the same precedence are evaluated left to right:

```
24/2/3   % The same result as the the second case.

24/(2*3) % This version is usually clearer.
```

A final example:

```
5^4^3

5^(4^3)  % Note the different result!

(5^4)^3  % This is the same as the first case, but clearer.
```

## Changing the Displayed Format

By default, MATLAB displays 5 decimal digits, unless the number  is too big or too small, when it switches to scientific notation. The  format of the answers displayed in the command window can be changed  using the **format** command.

```
help format
```

**Pay attention**: format does **NOT** affect how numbers are stored or calculations are done in MATLAB. It **only** affects how numbers are displayed.

### A Simple Example

```
format short
x = 3 + 11/16 + 2^1.2

format compact
x = 3 + 11/16 + 2^1.2

format long eng
x = 3 + 11/16 + 2^1.2

format long
x = 3 + 11/16 + 2^1.2
```

### Scientific Notation

For very large numbers or very small numbers it is more convenient to express the number in scientific notation using a power of 10.

```
3.6e+12

5.2e-9

2^30

2^(-40)
0.5/0.1-5

0.7/0.1 -7  % Sometimes the result may not be what you expect.
```

## Complex Numbers

MATLAB allows and works with complex numbers. All arithmetic with complex numbers works in the usual way.

You may use both $i$ and $j$ to denote the immaginary unit, if you did not already define any variable named $i$ or $j$(see below how to set valid variable names).

### Example

Define the complex number $2+3\,i$

```
2+3*i
```

```
2+3i

2+3j  % Any of these three expression works.
```

Define the complex expression

$$\frac{2+3\,i}{4+6\,i}$$

```
(2+3*i)/(4+6*i)
```

## Special functions for complex numbers

A MATLAB function to **create a complex number**: create the complex number $2+3\,i$

```
complex(2, 3) % the 1st argument is the real part, the 2nd arg. is the imaginary part
```

A MATLAB function to **get the real part** of the complex number $2+3\,i$

```
real(2+3i)
```

A MATLAB function to **get the imaginary part** of the complex number $2+3\,i$

```
imag(2+3j)
```

A MATLAB function to **get the modulus** of the complex number $\frac{2+3\,i}{4+6\,i}$

```
abs((2+3*i)/(4+6*i))
```

A MATLAB function to **get the argument** of a complex number (in radians)

```
angle(+1+j)

rad2deg(angle(+1+j))

rad2deg(angle(+1-j))

rad2deg(angle(-1-j))

rad2deg(angle((2+3*i)/(4+6*i)))

rad2deg(angle(2+3*i))

rad2deg(angle(4+6*i))
```

## Special Values for Infinity and for Undefined Operations

MATLAB uses the IEEE 754 standard for floating point numbers, to provide approximations to the real numbers. This standard, which is used in virtually every current computer microprocessor, specifies how floating point numbers are stored and also includes special values for infinity and for undefined operations.

### Infinity (Inf)

Operations that would result in a mathematical infinity give a result of `Inf`, while those that would result in minus infinity give `-Inf`.

```
1/0

-3/0
```

### Not a Number (NaN)

Operations whose value cannot be mathematically determined result in a *Not a Number*.

```
0/0
```

### Propagation of Inf and NaN in Calculations

Both Inf and Nan propagate in calculations without producing any error message.

```
1e-10*Inf

5*NaN
```

Operations involving `Inf` and/or NaN are also significantly slower than those using standard floating point numbers.

Operations involving `Inf` can produce a NaN.

```
Inf-Inf

Inf+Inf
```

Once you have produced a NaN, it affects all calculations from then on:

```
1e-32*NaN

NaN-NaN
```

## Variables

Variables are essential to store the result of a calculation so it can be subsequently used in a later calculation. This not only makes your programs much more efficient, but also makes them much easier to understand.

## Variable Names

A variable name is a sequence of letters, numbers and underscores "_", **starting with a letter**.

The following are invalid variable names. Try to insert one of those variable name in the code box below, run the section and see what message you will obtain

```
1x %  invalid variable name!

_x % another invalid variable name
```

```
% insert one of the two invalid variable declaration here and see the
% resulting error message

x1
```

Then insert one of the following valid variable names

```
x1  % valid variable name

x_  % valid variable name
```

You should obtain a different error message: indeed the variable name is valid, but the variable does not exist in the Matlab Workspace.

**Length of variable names**: although a variable name may have any length, MATLAB only recognises the first "few" characters. How many characters are recognised is specified by the Matlab command `namelengthmax` (which is typically 63).

## Warnings

- MATLAB is case sensitive; x and X are different variables!
- You should not use a variable name that clashes with a constant, for example `pi`, or with a MATLAB command, as `var` (Hint: `help var`)
- Names cannot be MATLAB keywords, for example `for`, `if`, `end`, etc.

## Assigning Values to Variables

Matlab uses = to assign a value to a variable.

**Note**: the left-hand-side of the = must be a valid variable name, while the right-hand-side must be a valid expression.

```
a = 1
```

```
b = -1
c = -2
delta = b^2 - 4*a*c
x1 = (-b + sqrt(delta))/(2*a)
x2 = (-b - sqrt(delta))/(2*a)
```

### Suppressing output

When Matlab does a calculation and assigns the result to a variable, the result is echoed to the command window.

Sometimes, especially when dealing with large vectors or matrices, it is very useful to keep the screen uncluttered by unwanted output. This is simply done by adding a semicolon ; to the end of the command

```
x = 2;
y = 4;
z = x*y
```

## Useful Commands

There are several useful commands that allow you to see what variables you have created and if necessary remove some or all of them.

Please remember, you can always get more information on a command using the `help` command.

### List of the Variables

- to list all the variables in the current MATLAB workspace

```
who

who x* % This comand lists all the variables starting with the (lower case) letter x.
```

- to lists the variables, but obtaining extra information about their size and data type

```
whos

whos x*
```

### Clearing Variables

To remove a variable, use the `clear` command

```
clear x1

whos
```

## Predefined Constants

When you start MATLAB, some variables already have predefined values:

- the constant $\pi$

```
pi
```

- the imaginary unit of complex numbers $0+i$ or $0+j$

```
i

j
```

- the largest double precision number and the smallest positive double precision number

```
realmax    % the largest double precision number

4*realmax  % What did you expect as the result?

realmin    % the smallest positive double precision number
```

MATLAB uses IEEE 754 standard **double precision** to store floating point numbers which approximate the real numbers. The characteristics of double precision arithmetic are described by the relative machine precision, also known as  the **machine epsilon**

```
eps
```

**Caveat**

You can give these variables new values. If you do so you may get strange effects afterwards!

## Summary

Using this live script you have:

- seen how arithmetic expressions are evaluated in MATLAB and how to use brackets to get the desired result or make an expression clearer;

- learnt how to enter numbers using scientific notation;
- learnt how to change the format of the result displayed in the MATLAB Command Window;
- learnt how to enter complex numbers and how to find real and imaginary parts;
- learnt about Inf (infinity) and NaN (Not a Number);
- learnt what are valid names for variables in MATLAB;
- learnt how assign values to variables using the = operator;
- learnt how to display the value of a variable;
- leant how to see what variables have been defined and how to clear a variable from your workspace.

**Back to the Index**

Use this link to go back to the main live script of the collection.

**Back to the Previous Part: Starting MATLAB**

Use this link to go back to the previous live script of this collection.

**Go to the Next Part: Vectors & Matrices**

Use this link to go to the next live script of the collection.