2.2.1 Numbers

MATLAB represents numbers in two form, fixed point and floating point.

Fixed point: Decimal form, with an optional decimal point. For example:

```
2.6349      -381        0.00023
```

Floating point: Scientific notation, representing $m \times 10^e$

For example: 2.6349×10^5 is represented as `2.6349e5`

It is called *floating point* because the decimal point is allowed to move. The number has two parts:

- *mantissa* m: fixed point number (signed or unsigned), with an optional decimal point (`2.6349` in the example above)
- *exponent* e: an integer exponent (signed or unsigned) (`5` in the example).
- Mantissa and exponent must be separated by the letter `e` (or `E`).

Scientific notation is used when the numbers are very small or very large. For example, it is easier to represent 0.000000001 as `1e-9`.

2.2.2 Operators

The evaluation of expressions is achieved with arithmetic *operators*, shown in the table below. Operators operate on *operands* (`a` and `b` in the table).

Operation	Algebraic form	MATLAB	Example
Addition	$a + b$	`a + b`	`5+3`
Subtraction	$a - b$	`a - b`	`23-12`
Multiplication	$a \times b$	`a * b`	`3.14*0.85`
Right division	$a \div b$	`a / b`	`56/8`
Left division	$b \div a$	`a \ b`	`8\56`
Exponentiation	a^b	`a ^ b`	`5^2`

Precedence of operations (order of evaluation)

Since several operations can be combined in one expression, there are rules about the order in which these operations are performed:

1. Parentheses, innermost first
2. Exponentiation (^), left to right
3. Multiplication (*) and division (/ or \) with equal precedence, left to right
4. Addition (+) and subtraction (−) with equal precedence, left to right

When operators in an expression have the same precedence the operations are carried out from left to right. Thus `3 / 4 * 5` is evaluated as `( 3 / 4 ) * 5` and *not* as `3 / ( 4 * 5 )` .

2.2.3 Variables and Assignment Statements

Variable names can be assigned to represent numerical values in MATLAB. The rules for these variable names are:

- **Single string**
- Must start with a letter
- May consist only of the letters `a-z`, digits `0-9`, and the underscore character (_)
- May be as long as you would like, but MATLAB only recognizes the first 31 characters
- Is case sensitive: `items, Items, itEms, and ITEMS` are all different variable names.

Assignment statement: MATLAB command of the form:

- *variable = number*
- *variable = expression*

When a command of this form is executed, the expression is evaluated, producing a number that is assigned to the variable. The variable name and its value are displayed.

If a variable name is not specified, MATLAB will assign the result to the default variable, `ans`, as shown in previous examples.

Special variables:

`ans`: default variable name
`pi`: ratio of circle circumference to its diameter, $\pi = 3.1415926...$
`eps`: smallest amount by which two numbers can differ
`inf` or `Inf` : infinity, e.g. $1/0$
`nan` or `NaN` : not-a-number, e.g. $0/0$
`date`: current date in a character string format, such as `19-Mar-1998`.
`flops`: count of floating-point operations.

I,j: √-1

Commands involving variables:

`who`: lists the names of defined variables
`whos`: lists the names and sizes of defined variables
`clear`: clears all variables, resets default values of special variables
`clear` *var*: clears variable *var*
`clc`: clears the command window, homes the cursor (moves the prompt to the top line), but does not affect variables.
`clf`: clears the current figure and thus clears the graph window.

Punctuation and Comments

- Semicolon (;) at the end of a command suppresses the display of the result

- Commas and semicolons can be used to place multiple commands on one line, with commas producing display of results, semicolons supressing
- Percent sign (%) begins a comment, with all text up to the next line ignored by MATLAB
- Three periods (...) at the end of a command indicates that the command continues on the next line. A continuation cannot be given in the middle of a variable name.

2.3 Basic Mathematical Functions

MATLAB supports many mathematical functions, most of which are used in the same way you write them mathematically.

Elementary math functions (enter `help elfun` for a more complete list):

`abs(x)`	Absolute value $\lvert x \rvert$
`sign(x)`	Sign, returns -1 if $x < 0$, 0 if $x = 0$, 1 if $x > 0$
`exp(x)`	Exponential e^x
`log(x)`	Natural logarithm $\ln x$
`log10(x)`	Common (base 10) logarithm $\log_{10} x$
`sqrt(x)`	Square root $\sqrt{x}$
`rem(x,y)`	Remainder of x/y. For example, `rem(100,21)` is 16. Also called the **modulus** function.

Help elfun

Files and File Management

- File management definitions and commands
- Saving and restoring information
- Script M-files

File: A collection of computer data, either ASCII text or binary, that is stored on an external memory device, such as a disk or tape drive. The format of the file, or the order, size, and location of the data items stored in the file, is determined by the program that writes the file on the external memory device.

File types: For our purposes, there are two types of files: **binary** and **text** (also called ASCII text or plain text). Binary files contain both the machine instructions of executable programs and data stored in machine-readable form. Text files contain keyboard characters represented by one byte (8 bits) per character. Text, but not binary, files can be created and modified with text editors and printed on printers. Binary files can only be read an operated upon by programs designed to handle the internal formats of these files.

3.2.2 Saving and Retrieving Matlab Variables

There will be occasions when you want to save your MATLAB variables so that you can later retrieve them to continue your work. In this case, you must save the information in the MATLAB binary format, so that the full precision of the variables is retained. Files in this MATLAB binary format are known as MAT-files and they have an extension of `mat`.

Storing and Loading Workspace Values

`save`	Stores workspace values (variable names, sizes, and values), in the binary file `matlab.mat` in the present working directory
`save data`	Stores all workspace values in the file `data.mat`
`save data_1 x y`	Stores only the variables `x` and `y` in the file `data_1.mat`
`load data_1`	Loads the values of the workspace values previously stored in the file `data_1.mat`

Exporting and Importing Data

There are also situations in which you wish to export MATLAB data to be operated upon with other programs, or to import data created by other programs. This must be done with text files written with `save` or read with `load`.

To write a text file `data1.dat` in the current working directory containing values of MATLAB variables in `long e` format:

```
save data1.dat -ascii
```

Note that the individual variables will not be identified with labels or separated in any way. Thus, if you have defined variables `a` and `b` in the MATLAB workspace, the command above will output first the values in `a`, then the values in `b`, with nothing separating them. Thus, it is often desirable to write text files containing the values of only a single variable, such as:

```
save data2.dat a -ascii
```

This command causes each row of the array `a` (more on arrays later) to be written to a separate line in the data file. Array elements on a line are separated by spaces. The separating character can be made a tab with the command:

```
save data2.dat a -ascii -tab
```

Per i files ascii si può usare anche l'estensione ".txt"

The `load` command followed by the filename will read the information into an array with the same name as the base name of the data file (extension removed). For example, the command

```
load data3.dat
```

Metodo alternativo:
Utilizzo del comando “Import Data” dalla riga dei comandi

3.3 Script M-Files

For simple problems, entering commands at the MATLAB prompt in the Command window is simple and efficient. However, when the number of commands increases, or you want to change the value of one or more variables, reevaluate a number of commands, typing at the MATLAB becomes tedious. You will find that for most uses of MATLAB, you will want to prepare a *script*, which is a sequence of commands written to a file. Then, by simply typing the script file name at a MATLAB prompt, each command in the script file is executed as if it were entered at the prompt.

Help script file

```
%prova di m-file

% calcolo il valore di a. Visualizzo
% il risultato
a = ((3^2)+4)/(2*7)

% calcolo il valore di b. Non visualizzo
% il risultato
b = 9^(3+6)+13.9*7.5;

% calcolo il valore di c e di d.
% Unica riga di comando. visualizzo solo il
% risultato di c.
c = a^3+ (7.2/b),...
d = (a^2+b^4)/(a^4*b^2);
```

The following are some suggestions on the effective use of MATLAB scripts:

1. The name of a script file must follow the MATLAB convention for naming variables; that is, the name must begin with a letter and may include digits and the underscore character.

2. Do not give a script file the same name as a variable it computes, because MATLAB will not be able to execute that script file more than once unless the variable is cleared. Recall that typing a variable name at the command prompt causes MATLAB to display the value of that variable. If there is no variable by that name, then MATLAB searches for a script file having that name. For example, if the variable `rqroot` was created in a script file having the name `rqroot.m`, then after the script is executed the first time, the variable `rqroot` exists in the MATLAB workspace. If the script file is modified and an attempt is made to run it a second time, MATLAB will display the value of `rqroot` and will not execute the script file.

3. Do not give a script file the same name as a MATLAB command or function. You can check to see whether a function already exists by using the the `which` command. For example, to see

3.4 Errors and Debugging

Some examples of syntax errors and associated messages include:

- Missing parenthesis

```
>> 4*(2+5
??? 4*(2+5
           |
A closing right parenthesis is missing.
Check for a missing ")" or a missing operator.
```

This message is helpful, as it even points out the location of the error.

- Missing operator

```
>> 4(2+5)
??? 4(
     |
Missing operator, comma, or semi-colon.
```

- Misspelled variable name

```
>> 2x = 4*(2+5)
??? 2
```

Run-time and Logic Errors

After correcting syntax errors, your script will execute, but it still may not produce the results you desire.

Run-time Errors: These errors occur when your script is run on a particular set of data. They typically occur when the result of some operation leads to `NaN` (not a number), `Inf` (infinity), or an empty array (to be covered later in the course).

Logic Errors: These are errors in your programming logic, when your script executes properly, but does not produce the intended result. When this occurs, you need to re-think the development and implementation of your problem-solving algorithm.

Suggestions for debugging

- Execute the script on a simple version of the problem, using test data for which the results are known, to determine which displayed results are incorrect.
- Remove semicolons from selected commands in the script so that intermediate results are displayed.
- Add statements that display variables of interest within the script.
- Place the `keyboard` command at selected places in the script to give temporary control to the keyboard. By doing so, the workspace can be interrogated and values changed as necessary. Resume script execution by issuing a `return` command at the keyboard prompt.