An introduction to MATLAB$^{©}$

Fundamental operations in Matlab
Graphical interface

MATLAB = **MAT**rix **LAB**oratory

Matlab is a programming environment for algorithm development, data analysis, visualization, and numerical computation.

`www.mathworks.com`

Main features

- Development environment for managing code, files, and data
- High-level language for technical computing
- 2-D and 3-D graphics functions for visualizing data
- It runs on Unix/Linux, Windows, Mac.
- Matlab files are readable on every o.s.

It has very powerful **mathematical functions**

(for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration)

it provides a lot of **toolboxes**

(collections of special-purpose Matlab functions):

- Control System
- Signal Processing
- Statistics
- Neural Networks
- Fuzzy Logic
- Communications
- ...

Matlab prompt:

```
>>
```

Scalar variable assignment

```
>>a=1.54
```

- a is the variable name (max 31 alphanumeric characters, the first one cannot be a number)
- 1.54 is the number assigned to the variable.

The command

```
>> a=1.54 ↵
```

yields

```
a =
    1.5400
```

```
>> a=1.54; ↵
```

does not produce any answer

`>> 1.67` $\boxed{\hookleftarrow}$ *yields*

```
ans =
    1.6700
```

`ans` is the name of the default variable.

`>> a` $\boxed{\hookleftarrow}$ *yields*

```
a =
    1.5400
```

to print the value of the variable `a`

```
>> b=1+1/2+5/3+1/4+23/6+...
2/9+1/10;
```

to cut an instruction which is too long

Arithmetic operations

^ power

∗ product

/ division

+ sum

- difference

Ex: to compute $x = \dfrac{3 + 5^3 - 2/3}{4(5 + 2^4)}$

```
>> x=(3+5^3-2/3)/(4*(5+2^4))
```

↵

- Classical rules on the priority of operations are observed
- To modify the priority of the operations we can use only **round brackets**

```
>> whos ↵

  Name        Size            Bytes  Class

  a           1x1                 8  double array
  ans         1x1                 8  double array
  b           1x1                 8  double array
  x           1x1                 8  double array
```

Every variable is an array which is not required to be declared or dimensioned.
Matlab uses **double precision variables**, by default.
Each entry of a ”double array class” variable is stored in 8 Bytes.
Each real number is an **array** `1x1` (one row by one column).
Remark. By default capital and small letters are different, either in the variable names and in the commands.

How Matlab stores floating point numbers

Tipically a floating point number can be stored in two different formats:

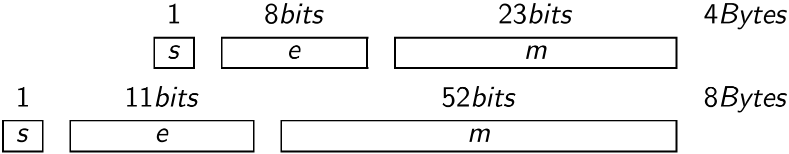
- **Single** (or simple) precision, 4 Bytes
- **Double** precision, 8 Bytes

$$\boxed{\text{How are used these Bytes?}}$$

Let us consider the exponential form of a number:

$$\begin{aligned} x &= 123456.789 = (-1)^0 0.123456789 \cdot 10^6 \\ &= (-1)^0 123456789 \cdot 10^{6-9} = (-1)^s m \cdot \beta^{e-t} \end{aligned}$$

$s = 0, 1$; m mantissa; β basis (e.g.: 2,10); e exponent



1 $8\,bits$ $23\,bits$ $4\,Bytes$

s e m

1 $11\,bits$ $52\,bits$ $8\,Bytes$

s e m

Output format for a number

```
>> c=0.456723
c =
    0.4567
```

the number is printed with 5 digits

```
>> format short e
>> c
c =
    4.5672e-01
```

Exponential format with 5 digits for the mantissa

```
>> format long e
>> c
c =
     4.567230000000000e-01
```

Exponential format with 16 digits for the mantissa

```
>> format long
>> c
c =
    0.456723000000000
```

the number is printed with 16 digits

By default Matlab uses the `format short`. To come back to format short:

```
>> format short
```

Remark The output format can change, but the inner format for storing the number is always the same (8Bytes).

Predefinite variables

`pi`	π
`i, j`	$\sqrt{-1}$ imaginary unit
`NaN`	not a number
`eps`	`2.2204e-16` machine precision

The value of these variables can be modified by inizialization:

```
>> pi=18
pi =
    18
```

To reset variable `pi` to its default value π:

```
>> clear pi
>> pi
ans =
    3.1416
```

To delete the value of a variable `a`:

```
 >> clear a
```

To delete the value of all the variable previously defined:

```
 >> clear
```

How to initialize an array

```
>> a=[1 2 3 4];
>> a=[1,2,3,4];
>> a=(1:4);
```

Equivalent ways to define an array 1x4, 1 row and 4 columns, it is a row vector

```
>> a
a =
     1     2     3     4
```

```
>> b=[1;2;3;4]
b =
     1
     2
     3
     4
```

To define an array 4x1, 4 rows and 1 column, it is a column vector

```
>> c=[5 3 4; 2 4 -2]
c =

     5     3     4
     2     4    -2
```

To define an array 2x3, matrix 2 rows and 3 columns

- Either the blank space or the comma separates elements on a row. The semicolon separates rows.

Transposition

```
>> a'
ans =
     1
     2
     3
     4
```

The transpose array of a is stored in the variable ans

```
>> a1=a'
```

The transpose array of a is stored in the variable $a1$

Similarly for matrix transpose:

```
>> c1=c'
c1 =
     5     2
     3     4
     4    -2
>> whos
  Name         Size                    Bytes  Class

  a            1x4                        32  double array
  ans          4x1                        32  double array
  b            4x1                        32  double array
  c            2x3                        48  double array
  c1           3x2                        48  double array
```

```
>> a(2)
ans =
     2
```

To refer an entry of the array

```
>> c(2,1)
ans =
     2
```

To refer an entry of the matrix

```
>> d=c(1,:)
d =
     5     3    4
```

To extract the first row of a matrix and save it in the vector d

```
>> e=c(:,1:2)
e =
     5     3
     2     4
```

To extract the first two rows of a matrix and save them in the matrix e

```
>> b(3)=5
b =
     1
     2
     5
     4
```

To modify one entry of the array. If ”;” is not used, all the array is printed

```
>> c(1,3)=18
c =
     5     3    18
     2     4    -2
```

To modify one entry of a matrix.

Operations with arrays

+ sum of two vectors or matrices (element by element)

− difference of two vectors or matrices (element by element)

∗ product between arrays and/or matrices (rows by columns)

These are the classical operations of linear algebra; therefore:

- sum and difference: the arrays dimensions must agree
- product: inner matrix dimensions must agree

```
>> a1+b                     both are column vectors 4x1

ans =
     2
     4
     8
     8
```

```
>> a-b
??? Error using ==> -
Matrix dimensions must agree.
```

a =*row vector (1x4)*

b =*column vector (4x1)*

```
>> a*b
ans =
    36
```

(1x4)(4x1) -inner product-

```
>> c*d'
ans =
   358
   -14
```

(2x3)(3x1) -matrix vector product-

```
>> d*c
??? Error using ==> *
Inner matrix dimensions must agree.
```

(3x1)(2x3) - this product is not possible-

The "point" operations works on two arrays with the same size:
`.*` product element by element
`./` division element by element
`.^` power element by element

```
>> a1b=a1.*b
a1b =
     1
     4
    15
    16
```

$$(a1b)_i = (a1)_i * b_i$$

with $a1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 4 \end{bmatrix}$

Mathematical functions and graphics

```
>> f=@(x)[(2*x-sqrt(2))^2*sin(2*x)]
f =
@(x)[(2*x-sqrt(2))^2*sin(2*x)]
>> whos
  Name        Size           Bytes  Class
  f            1x1              32  function_handle
```

f is a function handle and occupies 32 Bytes.
To evaluate f at a point:

```
>> x=1.718; y=f(x)
```

otherwise

```
>> y=f(1.718)
```

Problem 1: Evaluate $f(x) = x^2 \cos(x)$ on the interval $I = [-1, 2]$ and draw its plot.

Solution

1) Define a grid on the interval $I = [-1, 2]$, i.e. choose a discrete set of points in I:

```
>> x=linspace(-1,2,50);
```

This command defines a row vector (1x50), that holds the values of 50 equispaced points in I

2) Define the function and evaluate it:

```
>> f=@(x)[x.^2.*cos(x)];
>> y=f(x);
```

`x` is a vector, we want to compute $y_i = x_i^2 cos(x_i)$ for any i, thus we use ”point” operations

3) Plot the points (x_i, y_i) in the cartesian plane:

```
>> plot(x,y)
```

The syntax of the command `plot` is:
`plot(x,y,` *'color_linestyle_marker'*`)`

```
>> plot(x,y,'m-*')
```

color: `c,m,y,r,b,g,w,k`

linestyle: `-,--,:,-.,none`

marker: `+,o,*,.,x,s`

To plot 2 or more pairs of vectors on the same graph:

```
>> g=@(x)[sin(x).*exp(x)];
>> yg=g(x);
>> plot(x,y,'b:',x,yg,'r-');
```

To know in details all the options of a command, or if we do not remember the sintax of a command:

`help` *command_name*

```
>> help plot
```

If you do not remember the command name, but you will search by a keyword (english language), or if you look for every command that refers to a keyword:

`lookfor` *keyword*

```
>> lookfor  plot
```

Intrinsic mathematical functions

`sqrt(x)`	$\sqrt{x}$
`round(x)`	rounding: `round(3.6)=4`
`fix(x)`	integer part of a number: `fix(3.6)=`
`sign(x)`	sign of `x` (it is -1, 0 o 1)
`sin(x), cos(x), tan(x)`	$\sin(x)$, $\cos(x)$, $\tan(x)$
`sinh(x), cosh(x), tanh(x)`	$\sinh(x)$, $\cosh(x)$, $\tanh(x)$
`asin(x), acos(x), atan(x)`	$\arcsin(x)$, $\arccos(x)$, $\arctan(x)$
`exp(x), log(x), log10(x)`	e^x, $\log_e(x)$, $\log_{10}(x)$

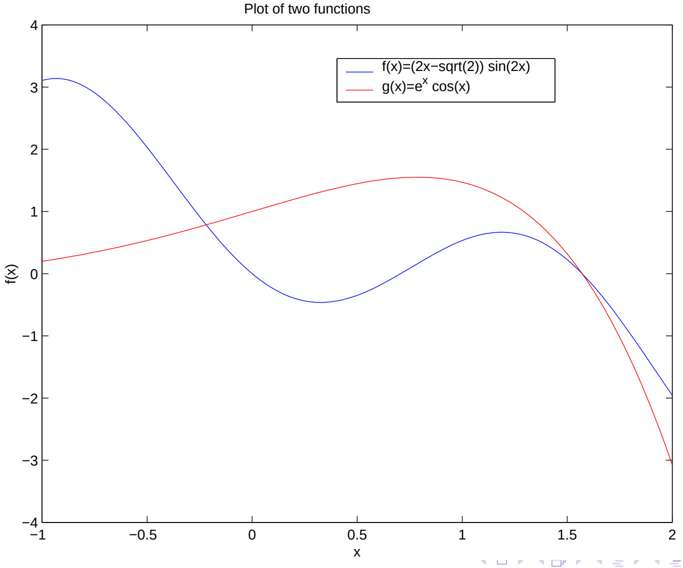
When `z` is a complex number:

```
>> z=3+i*4
```

`real(z)`	real part of `z`
`imag(z)`	imaginary part of `z`
`conj(z)`	conjugate of `z`

How to create a MATLAB script

Problem 2. Plot $f(x) = (2x - \sqrt{2})\sin(2x)$ and $g(x) = e^x\cos(x)$ on the interval $I = [-1, 2]$.



From the menu, select **File**, then **New** and then **Script**. A new window appears, it is an **Editor/Debug** window. We can write matlab commands.

```
clf;
f=@(x)[(2*x-sqrt(2))*sin(2*x)];
fplot(f,[-1,2])
xlabel('x'); ylabel('f(x)')
title('Plot of two functions')
hold on
g=@(x)[exp(x)*cos(x)];
fplot(g,[-1,2],'r')
legend('f(x)=(2x-sqrt(2)) sin(2x)','g(x)=e^x cos(x)')
hold off
```

To save the script: from the Editor menu select **File**, **Save as**.
Choose the directory where you will save the file (e.g.: `c:\tmp` or `e:\`) and the script name (for ex: `dis2d.m`).
Remark: the extension of a matlab file is **m**.
From the command window:

```
>> addpath c:\tmp
```

or

```
>> addpath a:\
```

in order to say matlab to look for in that directory, then type the name of the file (without the extension):

```
>> dis2d
```

Does Matlab report an error?

1) Read the message and try to understand the error

2) Come back to the editor window, look for the error and correct the script

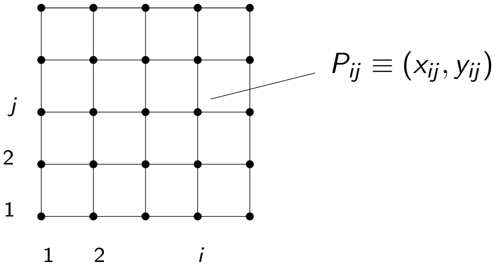
3) Save the file

4) Come back to the command window and retype the name of the commmand

```
>> dis2d
```

3D plots

Problem: Plot $f(x,y) = xe^{-(x^2+y^2)}$ on the domain $\Omega = [-2,2]^2$.



First of all we define a cartesian grid (or mesh) in Ω.

```
>> [x,y]=meshgrid(-2:.1:2,-2:.1:2);
```

x and y are two matrices .

```
>> clf
```

To clear the previous figure

```
>> f=@(x,y)[x.*exp(-x.^2-y.^2)];
>> z=f(x,y); surf(x,y,z); colorbar
```

Other commands for 3D plots:

`>> mesh(x,y,z)`	*Surface*
`>> meshc(x,y,z)`	*Surface and countour-lines*
`>> surfc(x,y,z)`	*Surface and countour-lines*
`>> pcolor(x,y,z)`	*Coloured flat surface*
`>> surf(x,y,z,gradient(z))`	*Coloured surface with the map of* $\partial z/\partial x$
`>> contour(x,y,z)`	*Contour-lines*
`>> plot3(x,y,z)`	*Lines in the direction* y *it is useful to draw lines in 3d plots too*

To create more than one figure, type the command `figure(k)` where `k` is a positive integer.
For example:.

```
>> mesh(x,y,z);
>> figure(2); surf(x,y,z,gradient(z));
>> figure(3); plot3(x,y,z);
```

To move from one figure to another, in order to modify a plot:

```
>> figure(2)
>> colorbar
```

If you want only one figure with more than one subplot:

```
>> figure(1)
>> subplot(2,2,1); mesh(x,y,z);
>> title(’mesh’)
>> subplot(2,2,2); surfc(x,y,z);
>> title(’surfc’)
>> subplot(2,2,3); plot3(x,y,z);
>> title(’plot3’)
>> subplot(2,2,4); surf(x,y,z,gradient(z));
>> title(’surf,gradient’)
```

Save and print a figure

In order to save a matlab figure:
From the Menu of the graphic window choose **File**, **Save as**; then choose the directory and the name with extension **.fig**.
To re-open the figure: from the Menu of Matlab main window choose **File**, **Open** and select the name of the file you want to open.

To save a figure in jpeg format: from the menu of the graphic window choose **File**, **Save as**, select **JPEG image (*.jpg)** and specify the file name.

Other formats: .eps, .tiff, .png, .pdf,....

Problem. Plot a surface in parametric form

$$\gamma(r,\theta) = (\quad \underset{x}{r\cos(\theta)}, \quad \underset{y}{r\sin(\theta)}, \quad \underset{z}{\theta} \quad)$$

with $r \in [0,2]$ and $\theta \in [0,6\pi]$.
Solution. Create a script with, at least, the following instructions:

```
[r,theta]=meshgrid(0:.1:2,0:.1:6*pi);
x=r.*cos(theta);
y=r.*sin(theta);
z=theta;
surf(x,y,z)
```