RISIS
Research infrastructure for research and innovation policy studies

Methodological course on Advanced Benchmarking Models and Techniques

22/03/2021

MATLAB

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Giammarco Quaglia

Email: giammarco.quaglia@uniroma1.it

Simone Di Leo

Email: dileo@diag.uniroma1.it

Index

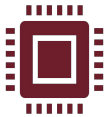
- **Matlab Introduction (some slides)**
  - **about 20 mins**

- **Laboratory session (MATLAB & Slack)**
  - **about 10 mins**

Prerequisites

- Willingness to learn the fundamentals of MATLAB
- Some experience in any coding language
- Having MATLAB installed on your machine
- Coffee ☕

What is MATLAB



MATLAB, standing for MATrix LABoratory, is a programming language and an environment for numerical calculation and statistical analysis

MATLAB allows you to manipulate matrices, view and work with data, implement algorithms and more.
Its workspace is very intuitive, in the top left corner there is the button New > Script, you might want to use this later for the practical sesion

*.m files

An .m file (M-file) is a program recognizable by Matlab.
Writing .m-files allows you to:

Experiment with an algorithm, without having to reintroduce a long list of commands

Get permanent documentation

Get programs that can be reused

Exchange programs files with other users (MATLAB File exchange: https://www.mathworks.com/matlabcentral/fileexchange/)

Saving and Loading variables

To save the workspace to a MATLAB file:

```
>> save file_name
```

and it will be saved with .mat extension

To store a value in a variable:
>> a = 5

To clear the workspace:
>> clear

To clear the command window:
>> clc

If you don't want the output to be shown, just add a semicolon ***;*** at the end of the line

Operate in MATLAB

It is possible to operate in MATLAB in two ways:

In the interactive mode, in which all commands are entered directly in the Command window

By running a MATLAB program stored in *script* file. This type of file contains MATLAB commands, so running it is equivalent to typing all the commands—one at a time—at the command window prompt. It is possible to launch the script in the file by typing its name in the command window prompt.

Script

To create a script:

- click new on the top left corner of the workspace
- select script

To run the entire script:

- click “run”
- Mac: cmd + enter
- Windows: F5

Every variable in MATLAB is an array so let’s look at how to work with them

Arrays and matrices

The variable created before with a=5 is a scalar or a 1 by 1 **vector**

To create an array with multiple elements we use the square brackets:

- x = [1 2] gives the following

  x =

  1 2
- Spaces separate row elements, while semicolons is used for column element

  y = [1 2 ; 3 4]

  y =

  1 2

  3 4
- We can also use functions while assigning values

  v = [0 2^3]

  v =

  0 8

Arrays and matrices

To create **evenly spaced vectors** we use the **:** operator, whose default spacing value is 1

- x = 1:4 gives the following
  x =
  1 2 3 4

- Specifying a different spacing value is easy:
  y = 1 : 2 : 10
  y =
  1 3 5 7 9

- We can also use functions to create a matrix eg x = rand(2, 3)

Arrays indexing

- To extract the value in the 2nd row and 4th column of a matrix M

  M(row, col) → M(2,4)

- To select the last element → *end* operator

  M(end, 3)

- To index all the elements in a dimension → colon ***:*** operator
  M(: , :) → the entire array
  M(1 , :) → entire first row
  M(: , 3) → entire third column

- To index a vector, one number is enough eg v(3)

Operations on vectors

Matlab naturally works with arrays

- You can add a scalar

  ```
  >> a=ones(2,3);
  >> a + 1
  ```

- Or an array

  ```
  >> b=ones(2,3);
  >> a+b
  ```

  In this case getting the same result:

  ```
  ans =
  2    2    2
  2    2    2
  ```

Operations on vectors

Common mathematical and statistical operations can be performed easily

- max → vm = max(vector)
- min
- sqrt
- mean
- round

- For matrix multiplications → * operator

- For element wise matrix multiplication: .* operator

Common mathematical functions

Function	MATLAB syntax[1]
e^x	`exp(x)`
$\sqrt{x}$	`sqrt(x)`
$\ln x$	`log(x)`
$\log_{10} x$	`log10(x)`
$\cos x$	`cos(x)`
$\sin x$	`sin(x)`
$\tan x$	`tan(x)`
$\cos^{-1} x$	`acos(x)`
$\sin^{-1} x$	`asin(x)`
$\tan^{-1} x$	`atan(x)`

[1] The MATLAB trigonometric functions use radian measure.

Order of precedence

Precedence	Operation
First	Parentheses, evaluated starting with the innermost pair.
Second	Exponentiation, evaluated from left to right.
Third	Multiplication and division with equal precedence, evaluated from left to right.
Fourth	Addition and subtraction with equal precedence, evaluated from left to right.

- When in doubt, just add an extra parentheses

Relational and logical operators

The most common relational operators are:

== equal

~= different from

< less than

<= less or equal

The most common logical operators are:

& logical AND

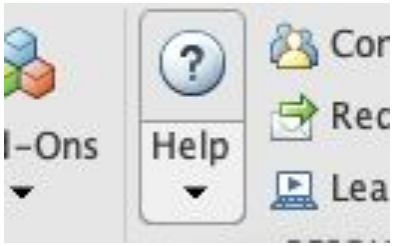
| logical OR

~ logical NOT

xor()

Help

- To check the documentation of a certain use the help button in the top right corner



- Directly in the script typing
  >>help function eg

```
help randn
```

Scalar arithmetic operations

Symbol	Operation	MATLAB form
^	exponentiation: a^b	a^b
*	multiplication: ab	a*b
/	right division: $a/b = \frac{a}{b}$	a/b
\	left division: $a\backslash b = \frac{b}{a}$	a\ b
+	addition: $a + b$	a+b
-	subtraction: $a - b$	a-b

If, For and While

The ***if*** construct has the structure:

if *expression*

instructions

end

```
if mod(4,2)==0
    disp("even")
end
```

The ***for*** cycle has the structure:

for *variable* = *expression*

instructions

end

```
for i = 1:5
    disp(i)
end
```

The ***while*** cycle has the structure

while *expression*

instructions

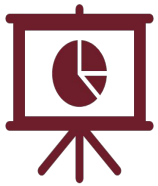
end

```
i=0;
while i < 5
    disp(i)
    i = i + 1;
end
```

Plot

To get the plot of a function, it is necessary:

- Prepare an abscissae vector
- Prepare an order vector
- Make the graph



Example: cos(4x)*exp(x) graph, on [0, 5]

```
>> x=0:0.01:5;
>> y=cos(4*x).*exp(x);
>> plot(x,y)
>> plot(x,y,”rx”) %red crosses
```

MATLAB plotting commands

Command	Description
`[x,y] = ginput(n)`	Enables the mouse to get *n* points from a plot, and returns the *x* and *y* coordinates in the vectors `x` and `y`, which have a length *n*.
`grid`	Puts grid lines on the plot.
`gtext('text')`	Enables placement of text with the mouse.
`plot(x,y)`	Generates a plot of the array `y` versus the array `x` on rectilinear axes.
`title('text')`	Puts text in a title at the top of the plot.
`xlabel('text')`	Adds a text label to the horizontal axis (the abscissa).
`ylabel('text')`	Adds a text label to the vertical axis (the ordinate).

Functions

To create a function the structure is the following.

```
function [product] = multiply_numbers(a,b)
    product = a*b;
end
```

In this example the function:

- is named *multiply_numbers*
- takes two values *a* and *b* as inputs
- multiply them
- returns their product *product*
- must be closed with and *end* as with *for*, *while* or *if* structures

Remember to save it (cmd+s) then you can then call it and use it

```
1    function [product] = multiply_numbers(a,b)
2    %example of a function.
3 -  product = a*b;
4 -  end
```

```
Command Window

>> multiply_numbers(2,3)

ans =

     6
```

Comments

The comment symbol may be put anywhere in the line. MATLAB ignores everything to the right of the % symbol. For example,

```
>>% This is a comment.
>>x = 2+3 % Also this one.
x =
5
```

To comment a block of code:

```
%{
…
code
...
%}
```

Programming Style

Programming Style: Comment Section

The first part of the script should include the following as comments

a. The name of the program and a few key words in the first line.

b. The date of creation and the authors' names in the second line.

c. The definitions of the variables' names for every input and output variable. Include definitions of variables used in the calculations and units of measurement for all input and all output variables.

d. The name of every user-defined function called by the program.

Programming Style

Input section Include input data and/or the input functions, eg *load data*.

Calculation section (the algorithmic part)

Output section This section contain functions for displaying the output on the screen (eg plots).

Some advice on MATLAB scripting

Keep in mind when using script files:

The name of a script file must begin with a letter, and may include digits and the underscore character, up to 63 characters.

Do not give a script file the same name as a variable.

Do not give a script file the same name as a MATLAB command or function. You can check to see if a command, function or file name already exists by using the `exist` command.

Program errors

Program errors usually fall into one of the following categories.

Syntax errors such as omitting a parenthesis or comma, or spelling a command name incorrectly. MATLAB usually detects the more obvious errors and displays a message describing the error and its location.

Errors due to an incorrect mathematical procedure, called *runtime errors*. Their occurrence often depends on the particular input data. A common example is division by zero.

Debugging Script Files

To locate program errors, try the following:

Test your program with a simple version of the problem which can be checked by hand.

Display any intermediate calculations by removing semicolons at the end of statements, or printing them out using *disp()* or *fprintf()*

Use the debugging features of the Editor/Debugger.

Exercises 1/2

1. Create the following two vectors and then sum them up:
   a. vector_1 = [1 2 3 4 5 6 7 8 9 10]
   b. vector_2 = [11 12 13 14 15 16 17 18 19 20]
2. Create a column vector of strings with the first 5 letters of the alphabet
3. Create a vector that goes from -1 to +1 containing 30 values using *linspace()*
4. Create a matrix of size 3x3 with values from 1 to 9
5. Create two identity matrices of length 3 and sum them up.
6. Verify that exp() and log() functions cancel each other out
7. Use a for loop to assign the value 3 to the diagonal elements of the resulting matrix from Ex.2
8. Create an identity matrix (command *eye()*) of length 3, then sum the scalar 2 to it, then transpose it and perform both Hadamart and matrix products with the matrix of ex. 4
9. In the command line display a string of text eg "The biggest value of the matrix is:" together with the corresponding value for the matrix computed in Ex.8 with the Hadamart product.

Exercises 2/2

10. Convert a number into a string with *num2str()* and then use it as the second element of a vector inside disp() command to print the two things consecutively.
    a. E.g. disp(['3 times 3 equal: ' num2str(9)]);
11. Create a for loop printing out the first 5 integer numbers
12. Create a for loop which prints out the square if it's even and the cube if it's odd for the first 10 integer numbers
13. Create a function that multiplies three numbers given as inputs
14. Create a function that prints out the first 10 elements of the Fibonacci Sequence

See you on slack!