Systems and Control Theory

Master Degree Course in ELECTRONICS ENGINEERING

http://www.dii.unimore.it/~lbiagiotti/SystemsControlTheory.html

Introduction to programming in MATLAB

Luigi Biagiotti

e-mail: luigi.biagiotti@unimore.it

http://www.dii.unimore.it/~lbiagiotti

Introduction

- The commands entered in the Command Window cannot be saved and executed again for several times. Therefore, a different way of executing repetitively commands with MATLAB is:
  1. **create a file** with a list of commands
  2. **save the file**
  3. **run the file**

- MATLAB has a text editor specialized for creating M-files that can be opened with the command **`>> edit`** or **`>> edit filename`** to open (or create) the file filename.m

- MATLAB file can be ran by typing the name (without extension)

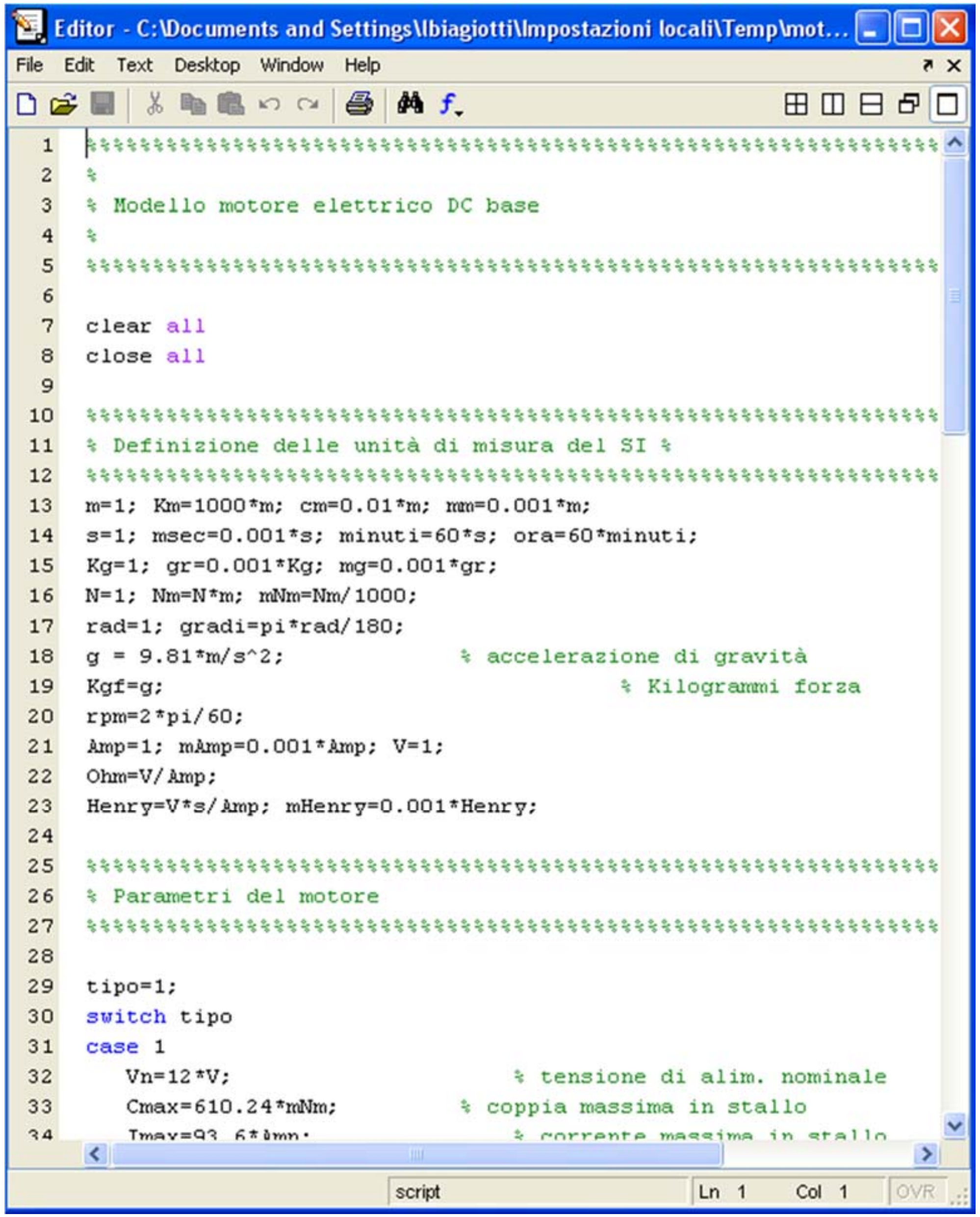
  `>> fileName <ENTER>`

M-File Scripts

- A ***script file*** is an external file that contains a sequence of MATLAB statements (comments are preceded by %).

- Script files have a ***filename extension*** .m and are called M-files.

- M-files can be
  - *scripts* that simply execute a series of MATLAB statements
  - *functions* that can accept arguments and can produce one or more outputs.



M-File Scripts

- By creating a file with the extension .m, we can easily write and run programs.

- We do not need to *compile* the program since MATLAB is an **interpretative** (not compiled) language.

- MATLAB has thousand of *functions*, and you can add your own using m-files.

M-file example

- Write a script for the solution of a linear system

$$\begin{cases} x_1 + x_2 + x_3 - x_4 = 1 \\ x_1 + x_2 - x_3 = 2 \\ x_1 - x_2 + x_3 = 0 \\ x_1 + 2x_2 - 3x_3 = 2 \end{cases}$$

- Solution (in the file `LinearSystemScript.m`)

```
A = [1, 1, 1, -1; 1, 1, -1, 0; 1, -1, 1, 0; 1, 2, -3, 0];
b = [1, 2, 0, 2]';
x = inv(A)*b;
```

Script side-effects

- **All variables created in a script file are added to the workspace**. This may have undesirable effects, because:
  - Variables already existing in the workspace may be overwritten.
  - The execution of the script can be affected by the state variables in the workspace.

M-functions

- Each M-function has **its *own* area of workspace**, separated from the MATLAB base workspace
- Structure of a M-function

```
function [Output]= FuncName(Input)
        % FuncName returns...
        % ...
        instructions;
              .
              .
              .
```

Function definition line (keyword **function**): it defines the function name, and number and order of input and output arguments

Description of the program, displayed when you request help

Function body: Program code that performs the actual computations

- **FuncName** must begin with a letter, and must be no longer than the maximum of 63 characters.
- **The name of the text file containing the function must be equal to the function name** with the extension .m

Control flow and operators

- Like other computer programming languages, MATLAB has some **decision making structures** for control of command execution. These *control flow* structures include *for loops*, *while loops*, and *if-else-end* constructions.

- Control flow structures are often in script M-files and M-function.

‘if...end’ structure

- MATLAB supports the variants of *if* construct:
  1. `if ... end`
  2. `if ... else ... end`
  3. `if ... elseif ... else ... end`
- Example (computation of the discriminant):

1.
```
discr = b*b - 4*a*c;
if discr < 0
disp('Warning: discriminant is negative, roots are
imaginary');
end
```

2.
```
discr = b*b - 4*a*c;
if discr < 0
disp('Warning: discriminant is negative, roots are
imaginary');
else
disp('Roots are real, but may be repeated')
end
```

'if...end' structure

- Example (computation of the discriminant):

3.
```
discr = b*b - 4*a*c;
if discr < 0
disp('Warning: discriminant is negative, roots are imaginary');
elseif discr == 0
disp('Discriminant is zero, roots are repeated')
else
disp('Roots are real')
end
```

- Note that
  - elseif has no space between else and if (one word)
  - no semicolon (;) is needed at the end of lines containing if, else, end
  - indentation of if block is not required, but facilitate the reading.
  - the **end statement** is **required**

Relational and logical operators

- A relational operator compares two expressions by determining whether a comparison is *true* or *false* (**comparison is made element-by-element**). Relational operators are shown in the following table

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
~=	Not equal to
&	AND operator
|	OR operator
~	NOT operator

The ‘for...end’ loop

- In the **for ... end** loop, the execution of a command is repeated at a fixed and predetermined number of times.
- The syntax is

```
for variable = expression
statements
end
```

where **expression** is usually a vector of the form **i:s:j**

- Example: definition of a row vector

```
y=[];
for t=0:0.1:5
    y= [y t];
end
```

- Multiple for loops can be nested

The ‘while...end’ loop

- This loop is used when the number of *passes* is not specified. The looping continues until a stated condition is satisfied.
- The while loop has the form

```
while expression
statements
end
```

where **statements** are executed as long as **expression** is true.

- Example

```
x = 1
while x <= 10
x = 3*x
end
```

- If the condition inside the looping is not well defined, the looping will continue *indefinitely*. If this happens, we can stop the execution by pressing **Ctrl-C**.

Exercise #1

- Define the MATLAB function `[x] = LinearSystem(A,b)`, that finds the solution x of a generic system of linear equations $A\,x = b$

- Use the function to solve the system

$$\begin{cases} x_1 + x_2 + x_3 - x_4 = 1 \\ x_1 + x_2 - x_3 = 2 \\ x_1 - x_2 + x_3 = 0 \\ x_1 + 2x_2 - 3x_3 = 2 \end{cases}$$

Exercise #2

- Define the MATLAB function

  `[A,B,C,D] = ControllableCanonicalForm(Num,Den)`

  that, starting from the transfer function of a SISO system (**`Num`** and **`Den`** are the vectors of the coefficients of the numerator and denominator polynomials, respectively), provides the matrices of the state-space representation of the system in the controllable canonical form.

- Given the system

$$G(s) = \frac{10s + 10}{s^3 - 1.6s^2 - 15.4s + 6.1}$$

  find its model in the state-space representation by using the newly defined function.

Exercise #2

- The relation between the *n*-th order transfer function

$$G(s) = \frac{c_{n-1}s^{n-1} + c_{n-2}s^{n-2} + \ldots + c_1 s + c_0}{s^n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \ldots + a_1 s + a_0}$$

and the matrices of the system is

$$A = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -a_0 & -a_1 & -a_2 & \ldots & -a_{n-1} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

$$C = \begin{bmatrix} c_0 & c_1 & c_2 & \ldots & c_{n-2} & c_{n-1} \end{bmatrix}, \qquad D = 0.$$

Exercise #3

- Define the MATLAB function **[q_t] = TrjPoly3(q0,q1,T,dt)**, that returns a vector containing the samples (computed with time-step **dt**) of a third-order polynomial trajectory from the initial point q_0 to the final point q_1 in a duration T.
- the analytical equation of the trajectory is

$$q(t) = q_0 + h\left(3\left(\frac{t}{T}\right)^2 - 2\left(\frac{t}{T}\right)^3\right), \quad 0 \leq t \leq T,$$

- With the new function compute a trajectory from $q_0 = 0$ to $q_1 = 3$ (T=2) and plot its behavior as a function of time.

Systems and Control Theory

Master Degree Course in ELECTRONICS ENGINEERING

http://www.dii.unimore.it/~lbiagiotti/SystemsControlTheory.html

Introduction to programming in MATLAB

Luigi Biagiotti

e-mail: luigi.biagiotti@unimore.it

http://www.dii.unimore.it/~lbiagiotti