SMS 3515: Scientific Computing
Lecture 2: Introduction to Matlab Programming
2014

Instructor: Nurul Farahain Mohammad

1 Introduction

- Computer program : A group of statements. A sequence of instructions, in a given language, that accomplishes a task.
- Script or m-file : MATLAB program.

2 Algorithms

Algorithm is a sequence of steps needed to solve a problem. Three basic steps of an algorithm :

1. Get the input(s).
2. Calculate the result(s).
3. Display the result(s).

3 Matlab scripts

- Execute / run program : having the computer follow these instructions sequentially.
- The computer can only interpret commands written in its machine language.
- High-level languages : commands / functions written in human language.
- Compiler : translate high-level language to an executable file.
- Source code : original code written in high-level language.
- Object code : code translated by compiler into machine language.
- Interpreter : translate the code line-by-line, executing each command as it goes.
- Scripts in MATLAB or an m-file is interpreted, not translated.
- You can run / execute an m-file by typing the name of particular m-file at Command Window.
- Before creating a script, make sure the Current Folder is set to the folder in which you want to save your files.

- Create an m-file : Click *New Script*. A window named Editor will be popped out.
- Edit an existing m-file
  : Click on File, then Open, and then click on the name of the file, or
  : Within the Current Folder Window, double-click on the name of the file in the list of files.

3.1 Documentation

- Put comments (% in front of a sentence / statement) for documenting your code.
- The *help* command in MATLAB works with scripts as well as with built-in functions. The first block of comments (defined as contiguous lines at the beginning) will be displayed on Command Window.

4 Input and output

4.1 Input function

- The simplest input function in MATLAB is called *input*. The *input* function is used in an assignment statement.
- In MATLAB, an input can be a value, a character or a string.
- If blank spaces are entered before other characters, they are included in the string.To overcome this, you may use *length* function to check.
- Separate input statements are necessary if more than one input is desired.

4.2 Output statements: disp and fprintf

- *disp* : display the result of an expression or a string without assigning any value to the default variable ans. However, disp does not allow formatting.
- *fprintf* : print formatted output on the screen, in particular, on Command Window.

```
>> fprintf('The value is %d, for sure!\n',4^3)
The value is 64, for sure!
>>
```

- place holder : specifies where the value of the expression that is after the string is to be printed.
  : %d for integer (it actually stands for decimal integer).
  : %f for float (real number).
  : %c for single character.
  : %s for string.
- newline character \n : put at the end of the string so that the output that follows moves down to the next line.
- tab character \t : tab space in output.

- field width can be included in the place holder in *fprintf*, which specifies how many characters total are to be used in printing. (e.g. ٪6.2f means a field width of 6 (including the decimal point and the two decimal places)
- value being printed can be left-justified within the field width using a minus sign $(-)$.
- to print matrices / vectors, you are advised to use *disp* for simplicity.

5 Scripts with input and output

You may put *input*, *disp*, and *fprintf* altogether in one script.

6 Scripts to produce and customize simple plots

You may type *help* and then click on *graph2d* and/or *graph3d* for description and examples on plots.

6.1 The plot function

- *plot(x,y,'color','marker')* : This is for 2D plot where the graph is *y* versus *x*. It is optional to specify a color or a marker. The default marker is solid line.

Table 1: List of markers commonly used for plotting.

o	circle
d	diamond
h	hexagram
p	pentagram
+	plus
.	point
s	square
*	star
v	down triangle
<	left triangle
>	right triangle
∧	up triangle
x	x-mark
–	dashed line
-.	dash-dot line
:	dotted line
-	solid line

Table 2: List of colors commonly used for plotting.

b	blue
c	cyan
g	green
k	black
m	magenta
r	red
y	yellow

- *axis([xmin xmax ymin ymax])* : to control the value of axes by putting appropriate values of *xmin, xmax, ymin*, and *ymax*.
- *xlabel('Write The Name of x-Axis Here')* : to label the x-axis.
- *ylabel('Write The Name of y-Axis Here')* : to label the y-axis.
- *title('Write The Title of The Graph Here')* : to label the graph.

6.2 Simple related plot functions

- *clf* : clears the Figure Window by removing everything from it.
- *figure* : creates a new, empty Figure Window when called without any arguments. Calling it as *figure(n)* where *n* is an integer is a way of creating and maintaining multiple Figure Windows, and of referring to each individually.
- *hold* : freezes the current graph in the Figure Window, so that new plots will be superimposed on the current one. Just hold by itself is a toggle, so calling this function once turns the hold on, and then the next time turns it off. Alternatively, the commands *hold on* and *hold off* can be used.
- *legend* : displays strings passed to it in a legend box in the Figure Window, in order of the plots in the Figure Window. The first string is paired with the first plot, and the second string with the second plot.
- *grid* : displays grid lines on a graph. Called by itself, it is a toggle that turns the grid lines on and off. Alternatively, the commands *grid on* and *grid off* can be used.
- *bar(x,y)* : to plot a bar chart of *y* versus *x*.

7 Introduction to file input/output (load and save)

- Files can be :
  - read from.
  - written to : writing to a file, from the beginning.
  - appended to : writing, but starting at the end of the file rather than the beginning.
- files with extension *.dat* or *.txt*.

7.1 Writing data to a file

- *save filename.extension matrixname -ascii* : write data from a matrix to a data file, or to append to a data file.
- *type filename.extension* : display the contents of the file.

7.2 Appending data to a data file

- *save filename.extension matrixvariable -ascii -append*
  : to append existing a variable / matrix in a data file.
  : save from a matrix to a file.

7.3 Reading from a file

- *load filename.extension*
  : to read from the *filename.extension*.
  : works only if there are same number of values in each line, so that the data can be stored in a matrix.

8 User-defined functions that return a single value

8.1 Function definitions

A function in MATLAB that returns a single result consists of the following:

- The function *header* (the first line), comprised of :
  - The reserved word *function*.
  - Since the function returns a result, the name of the output argument followed by the assignment operator = .
  - The name of the function (IMPORTANT: This should be the same as the name of the M-file in which this function is stored to avoid confusion).
  - The input arguments in parentheses, which correspond to the arguments that are passed to the function in the function call.
- A comment that describes what the function does (this is printed when *help* is used).
- The *body* of the function, which includes all statements and eventually must put a value in the output argument.
- *end* at the end of the function. (Note: This is not necessary in many cases.)

Standard information should be included in the block comment in a function. These can include :

- Name of function
- Description of what the function does
- Format of the function call

- Description of input arguments
- Description of output arguments
- Description of variables used in function
- Programmer name and date written
- Information on revisions

8.2 Calling a function

You can call a function that you created at the Command Window. To recall what the purpose of the function or to check what is the input that you need to provide, you may type *help functionname* or *type functionname* at the Command Window.

8.3 Calling a user-defined function from a script

You can also call or use the function that you defined yourself in your script / M-file. Just make sure that your m-file that consists of the function you defined yourself, and the script that will call the function are in the same path / location.