الكلية	العلوم
القسم	الرياضيات
المادة باللغة الانجليزية	Programming (MATLAB)
المادة باللغة العربية	البرمجة بلغة (ماتلاب)
المرحلة الدراسية	الثانية
اسم التدريسي	صفوت عبدالقادر حمد
عنوان المحاضرة باللغة الانجليزية	Algorithms
عنوان المحاضرة باللغة العربية	الخوارزميات
رقم المحاضرة	الثامنة
المصادر والمراجع	MATLAB A Practical Introduction to Programming and Problem Solving
	MATLAB The Language of Technical Computing
	MATLAB numerical computing

University Of Anbar

College Of Science

Math Department

Programing Language (MATLAB)

Lecture 8

Algorithms

By:

Safwat A. Hamad

For the 2nd stage Math Department

Introduction to MATLAB Programming

We have now used the MATLAB® product interactively in the Command Window. That is sufficient when all one needs is a simple calculation. However, in many cases, quite a few steps are required before the final result can be obtained. In those cases, it is more convenient to group statements together in what is called a ***computer program***.

In this course, we will introduce the simplest MATLAB programs, which are called ***scripts***. Examples of scripts that customize simple plots will illustrate the concept. Input will be introduced, both from files and from the user. Output to files and to the screen will also be introduced. Finally, user-defined functions that calculate and return a single value will be described.

1. Algorithms

Before writing any computer program, it is useful to first outline the steps that will be necessary. An algorithm is the sequence of steps needed to solve a problem. In a ***modular*** approach to programming, the problem solution is broken down into separate steps, and then each step is further refined until the resulting steps are small enough to be manageable tasks. This is called the ***top-down design*** approach.

As a simple example, consider the problem of calculating the area of a circle. First, it is necessary to determine what information is needed to solve the problem, which in this case is the radius of the circle. Next, given the radius of the circle, the area of the circle would be calculated. Finally, once the area has been calculated, it has to be displayed in some way. The basic algorithm then is three steps:

- Get the input: the radius
- Calculate the result: the area
- Display the output

Even with an algorithm this simple, it is possible to further refine each of the steps. When a program is written to implement this algorithm, the steps would be as follows.

- Where does the input come from? Two possible choices would be from an ***external file***, or from the user (the person who is running the program) who enters the number by typing it from the keyboard. For every system, one of these will be the ***default input device*** (which means, if not specified otherwise, this is where the input comes from!). If the user is supposed to enter the radius, the user has to be told to type in the radius (and, in what units). Telling the user what to enter is called ***prompting***. So, the input step actually becomes two steps: prompt the user to enter a radius, and then read it into the program.
- To calculate the area, the formula is needed. In this case, the area of the circle is π multiplied by the square of the radius. So, that means the value of the constant for π is needed in the program.
- Where does the output go? Two possibilities are: (1) to an external file, or (2) to the screen. Depending on the system, one of these will be the ***default output device***. When displaying the output from the program, it should always be as informative as possible. In other words, instead of just printing the area (just the number), it should be printed in a nice sentence format. Also, to make the output even more clear, the input should be printed. For example, the output might be the sentence: "For a circle with a radius of 1 inch, the area is 3.1416 inches squared."

For most programs, the basic algorithm consists of the three steps that have been outlined:

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

As can be seen here, even the simplest problem solutions can then be refined further. This is top-down design.

2. Matlab Scripts

Once a problem has been analyzed, and the algorithm for its solution has been written and refined, the solution to the problem is then written in a particular programming language. A computer program is a sequence of instructions, in a given language, which accomplishes a task. To ***execute*** or ***run*** a program is to have the computer actually follow these instructions sequentially.

High-level languages have English-like commands and functions, such as “print this” or “if x < 5 do something.” The computer, however, can only interpret commands written in its ***machine language***. Programs that are written in high-level languages must be translated into machine language before the computer can actually execute the sequence of instructions in the program. A program that does this translation from a high-level language to an ***executable*** file is called a ***compiler***. The original program is called the **source code**, and the resulting executable program is called the ***object code***. Compilers translate from the source code to object code; this is then executed as a separate step.

By contrast, an ***interpreter*** goes through the code line-by-line, translating and executing each command as it goes. MATLAB uses what are called either script files or MATLAB code files, which have an extension on the file name of ***.m***.

These script files are interpreted, rather than compiled. Therefore, the correct terminology is that these are scripts, not programs. However, the terms are used somewhat loosely by many people, and documentation in MATLAB itself refers to scripts as programs. In this book, we will reserve the use of the word “program” to mean a set of scripts and functions, as described briefly in next lectures.

A script is a sequence of MATLAB instructions that is stored in a file with an extension of ***.m*** and saved. The contents of a script can be displayed in the Command Window using the ***type*** command. The script can be executed, or run, by simply entering the name of the file (without the .m extension).

Before creating a script, make sure the Current Folder is set to the folder in which you want to save your files.

The steps involved in creating a script depend on the version of MATLAB. The easiest method is to click on “New Script” under the HOME tab. alternatively; one can click on the down arrow under “New” and then choose Script (see Figure 3.1)

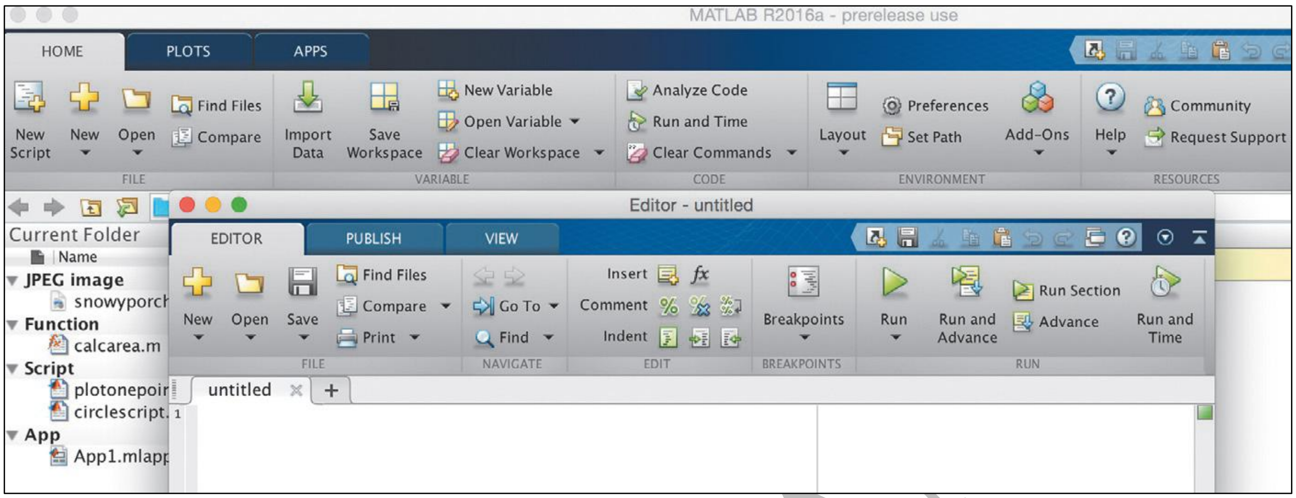

Figure 3.1

A new window will appear called the Editor (which can be docked). In the latest versions of MATLAB, this window has three tabs: “EDITOR”, “PUBLISH”, and “VIEW”. Next, simply type the sequence of statements (note that line numbers will appear on the left).

When finished, save the file by choosing the Save down arrow under the EDITOR tab. Make sure that the extension of ***.m*** is on the file name (this should be the default). The rules for file names are the same as for variables (they must start with a letter; after that there can be letters, digits, or the underscore.)

If you have entered commands in the Command Window and decide that you would like to put them into a script, an alternate method for creating a script is to select the commands in the Command History window, and then right click. This will give options for creating a script or live script and will then prepopulate the editor with those commands.

In our first example, we will now create a script called ***script1.m*** that calculates the area of a circle. It assigns a value for the radius, and then calculates the area based on that radius.

There are two ways to view a script once it has been written: either open the Editor Window to view it or use the type command, as shown here, to display it in the Command Window. The ***type*** command shows the contents of the file named ***script1.m***; notice that the ***.m*** is not included:

```
>> type script1
radius = 5
area = pi * (radius^2)
```

To actually run or execute the script from the Command Window, the name of the file is entered at the prompt (*again, without the .m*). When executed, the results of the two assignment statements are displayed, as the output was not suppressed for either statement.

```
>> script1
radius =
          5
area =
          78.5398
```

Once the script has been executed, you may find that you want to make changes to it (especially if there are errors!). To edit an existing file, there are several methods to open it. The easiest are:

- Within the Current Folder Window, double-click on the name of the file in the list of files.
- Choosing the Open down arrow will show a list of Recent Files

3. Documentation

It is very important that all scripts be **documented** well, so that people can understand what the script does and how it accomplishes its task. One way of documenting a script is to put comments in it. In MATLAB, a comment is anything from a **%** to the end of that particular line. Comments are completely ignored when the script is executed. To put in a comment, simply type the **%** symbol at the beginning of a line, or select the comment lines and then click on the Edit down arrow and click on the **%** symbol, and the Editor will put in the **%** symbols at the beginning of those lines for the comments.

For example, the previous script to calculate the area of a circle could be modified to have comments:

```
% This script calculates the area of a circle
% First the radius is assigned
  radius = 5
% The area is calculated based on the radius
area = pi * (radius^2)
```

The first comment at the beginning of the script describes what the script does; this is sometimes called a block comment. Then, throughout the script, comments describe different parts of the script (not usually a comment for every line, however!). Comments don’t affect what a script does, so the output from this script would be the same as for the previous version.

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. For example, for circle script:

```
>> help circlescript
This script calculates the area of a circle
```

The reason that a blank line was inserted in the script between the first two comments is that otherwise both would have been interpreted as one contiguous comment, and both lines would have been displayed with help. The very first comment line is called the “H1 line”; it is what the function look for searches through.

Application 1
Write a script to calculate the circumference of a circle (C ¼ 2 πr). Comment the script.

Longer comments, called ***comment blocks***, consist of everything in between **%{** and **%}**, which must be alone on separate lines. For example:

```
%{
```

```
this is used for a really
Really
REALLY
long comment
%}
```

Application 2
Write a script to calculate the sum tow numbers and Comment the script.

- All applications in our lectures implement in computer Lap
