SMS 3515: Scientific Computing
Lecture 6: MATLAB Programs
2014

Instructor: Nurul Farahain Mohammad

1 More Types of User-Defined Functions

- Functions can be categorized into:
  - functions that calculate and return one value.
  - functions that calculate and return more than one value.
  - functions that just accomplish a task without returning any values i.e. printing.
- In general, any function in MATLAB consists of the following:
  - The function header (the first line); this has:
    - the reserved word *function*.
    - the name(s) of the output argument(s). followed by the assignment operator $=$, if the function returns values.
    - the name of the function (note: this should be the same as the name of the m-file in which this function is saved to avoid confusion)
    - the input arguments in parentheses, if there are any (separated by commas if there is more than one).
  - A comment that describes what the function does (this will be shown when using *help*).
  - The body of the function, which includes all statements, including putting values in all output arguments if there are any.
  - *end* at the end of the function.

1.1 Functions that Return more than One Value

- General form (file name is functionname.m):

```
function [output arguments] = functionname(input arguments)
% Comment describing the function

Statements here; these must include putting values in all of the
 output arguments listed in the header

end
```

- In the vector of output arguments, the output arguments names are separated by commas.
- In recent version of MATLAB, a template of user-defined function is provided at the Editor by clicking *New*, and then choose *Function* at the toolbar.
- As the function is returning two values, it is important to capture and store these values in separate variables when the function is called. If this is not done, only the first value returned is retained.

1.2 Functions that Accomplish a Task Without Returning Values

- As these functions do not return any values, there are no output arguments and no assignment operator = in the function header.
  - General form (file name is functionname.m):

```
function functionname(input arguments)
% Comment describing the function

Statements here;
end
```

  - This type of function cannot be called from an assignment statement.
  - Tasks accomplished by functions that do not return any values are referred as side effects.

1.3 Functions that Return Values Versus Printing

- If a function just prints a value, rather than returning it ('save' it), the value cannot be used later in other calculations.
- A function that calculates and returns values does not normally also print them. Usually it is left to the calling script or function.

1.4 Passing Arguments to Functions

- Call-by-value method: method of passing the values of the arguments to the input arguments in the functions.
- In some cases, it is not necessary to pass any arguments to the function. i.e. a function of print tasks only.
- General form of no input and no output arguments (file name is functionname.m):

```
function functionname
% Comment describing the function

Statements here;
end
```

- General form of no input but with output arguments (file name is functionname.m):

```
function [output arguments] = functionname
% Comment describing the function

Statements here;
end
```

2 MATLAB Program Organization

2.1 Modular Programs

- Modular program: a program in which the solution is broken down into modules, and each is implemented as a function.
- Main program: a script that calls functions.
- In a modular program, there would be one main script that calls appropriate functions to accomplish desired tasks.
- For each modules, there would be separate m-file functions, and later one m-file script as the main program.

2.2 Subfunctions

- It is possible to have more than one function in a given m-file.
- Primary function: A function that calls other function/s.
- Subfunction: A function that is called by other function.
- Both primary function and subfunction/s are stored in the same m-file; first the primary function and then the subfunction/s.
- Name of the m-file would be the same as the name of the primary function.

3 Application: Menu-Driven Modular Program

- Menu-driven program: programs that have interaction with the user. Usually print a menu of choices and then continues to loop to print the menu of choices until the user chooses to end the program.
- A modular menu-driven program would typically have:
  - a function that represents the menu and gets the user's choice.
  - functions to implement the action for each choice (may involve subfunctions).
  - a function that would error-check all user input.

4 Variable Scope

- Scope of any variable: the workspace in which it is valid.
- Base workspace: the workspace created in the Command Window.
- A variable defined in any function is a local variable to that function (only used and known within that function).
- Local variables only exist while the function is executing; they cease to exist when the function stops executing.
- Variables that are defined in the Command Window cannot be used in a function unless passed as arguments to the function. However, scripts interact with variables that are defined in the Command Window.
- Variables in script do become part of the base workspace.
- It is recommended to start your script with command *clear* to eliminate variables that may have already been created.
- Writing a main function instead of a script, when dealing with many functions, is recommended.

4.1 Persistent Variables

- Persistent variable: a variable which its values are not cleared even though a function stops executing. The variable still exists and retains its former value when the next time the same function is called.
- *persistent variablename*: declare a persistent variable.
- The way to restart a persistent variable is to use the *clear* function.

5 Debugging Techniques

- Bug: any error in a computer program.
- Debugging: the process of finding errors in a program, and correcting them.
- *checkcode('filename')*: help find mistakes or potential problems in script and function files. Unfortunately this function is not available yet in Octave.

5.1 Types of Errors

- Syntax errors: mistakes in using the language.
- Runtime, or execution errors: Errors are found when a script or function is executing.
- Logical errors: mistakes in reasoning by the programmer, but it is not a mistake in the programming language.

5.2 Tracing

- *echo*: display every statement as it is executed, as well as results from the code. For script, simply type *echo* to call this function.
- *echo functionname on/off*: calling *echo* for function.

5.3 Editor/Debugger

- Ways to set breakpoints in an m-file:
  - *dbstop filename linenumber*: set a breakpoint in filename at the linenumber.
  - *dbcont*: continue execution.
  - *dbquit*: quit debug mode.
  - *dbstep*: step through the rest of the code one line at a time.
- click on the breakpoint alley (in between the line numbers on the left and the lines of code; which is a thin gray strip) in the Editor. The red dot indicates a breakpoint.

5.4 Function stubs

- Function stubs: a place holder, used so that the script will work even though that particular function hasn't been written yet.
- Put in place so that the script can be executed and tested.
- Consist of proper function headers, followed by a simulation of what the function will eventually do. Then the functions can be written and debugged one at a time.

5.5 Code Cells and Publishing Code

- Code cells
  - Sections that break a code.
  - Run one cell at a time.
  - Can publish the code in an HTML format with plots embedded and with formatted questions.
- How to break code into cells: Create comment lines that begin with two % symbols (these becomes the cell titles).
- When viewing in Editor, the individual cells can be chosen by clicking the mouse anywhere within the cell.
- From Editor tab, you can choose:
  - *Run Section*: to run just that one code cell and remain within that cell.
  - *Run and Advance*: to run that code cell and then advance to the next.
  - *Publish*: to publish the code by default in HTML document.