MATLAB Workshop 7 - Introduction to Functions

Objectives: Learn about user-defined functions in MATLAB.

MATLAB Features:

general function file heading

```
function  [output value list] = fcn_name(input value list)
```

- **More on scripts in MATLAB**

Consider the following script to accomplish the same goals as Workshop 5, Activity 1 (calculate the absolute pressure at the bottom of a tank).

```
% Solution to Workshop 7
% Your Name(s)
% Class (e.g. Engr 0012, TH 10:00, Instructor Name)
% Today's Date
% Your e-mail address

% Variable dictionary
%   rho         water density, kg/m^2
%   g           gravitational acceleration, m/s/s
%   height      tank height, m
%   pout        outside pressure, Pa
%   pabs_b      pressure at tank bottom in SI units, Pa

% set display format
    format short e

% display header to screen
    wkshp7_header

% get problem values
    [rho, g, height, pout] = wkshp7_getdata;

% calculate Pabs,b in Pa
    pabs_b = wkshp7_pbot(rho, g, height, pout);

% display results
    wkshp7_results(rho, g, height, pout, pabs_b)
```

This script definitely has far fewer lines of code. Furthermore, it is easy to read through the script and see what tasks are being performed to accomplish the goals of the script without being overwhelmed by the multiple individual steps. This script has given up complexity of detail for clarity in understanding **what** is happening and the order in which it is happening (control structure).

In order to accomplish the clarity associated with **what** is happening, the complex details of **how** it is happening have been hidden in the user-defined functions `wkshp7_header`, `wkshp7_getdata`, `wkshp7_pbot`, and `wkshp7_results`. If we are interested in the detail, we need to look at the functions. As the functions are not available, we will have to develop them in this workshop.

• User-defined functions in MATLAB

Functions are used to accomplish a single (complex) calculation or perform a single task. Some examples of a task are (1) displaying the script header/use information when a script is invoked, (2) obtaining input values from the user (keyboard) for further computation, (3) obtaining input values from a file for further computation, (4) doing a series of related computations such as calculating the mean, median, mode, and standard deviation for a data set, (5) displaying a plot of experimental data, (6) determining the "best fit" to a data set, (7) reporting results of computations to the screen, and (8) writing results of computations to a designated file.

Many user-developed functions are general purpose and have become the standard functions that are available in MATLAB for everyone to use. Examples of these are the vector functions described in Workshop 3, e.g., **`mean`**, **`median`**, **`sum`**, and **`max`**. Most functions, however, are specific to a given problem or script or user.

Functions are contained in **`m-files`**. The first line of any function **`m-file`** looks like this

```
function  [output value list] = fcn_name(input value list)
```

where the keyword **`function`** alerts MATLAB to the fact that the **`m-file`** contains a function and not a script. The function file should be saved as **`fcn_name.m`** where **`fcn_name`** is a user-chosen name for the function that is descriptive of what the function does.

Functions generally, but not always, require a set of input values that will be used by the function to do its job. Variable names are used to hold the input values. The input variable names are listed in parentheses following the function name in the first line of the function. If more than one name is listed, the names are separated by commas.

Likewise, functions generally, but not always, return a set of values that were obtained from the function performing its job. The output variable names are listed in square braces following the keyword **`function`** and preceding the assignment operator in the first line of the function. If more than one name is listed, the names are separated by commas. ***Each name listed in the output value list must appear on the left hand side of an assignment statement somewhere in the function.***

• Designing a function

Functions are easy to design if the following steps are taken in the order presented.

(1) Identify precisely what task the function is supposed to accomplish. For example: (a) get some (explicit) values from the user that will be used by another function later on; (b) determine the best fit to a set of (x,y) data points; or (c) display the results of computations.

(2) Identify what values will result if the function completes its task successfully. For example: (a) the user input values; (b) the slope, intercept, and goodness of fit statistic for the line; or (c) nothing - display only.

(3) Identify what values the function requires to perform its tasks. This refers to values that are passed to the function inside the script, not values that come from an outside source such as file or user input. For example: (a) nothing (all values will come from outside the computer); (b) the (x,y) data points; or (c) the values to be displayed.

(4) Identify a computational pathway/algorithm to get from known (input) values to the desired result (output) values. For example: (a) a series of **`input`** statements; (b) the equations to determine slope, intercept, and goodness of fit statistic; or (c) a series of **`disp`** commands.

We will employ this outline to design the functions for this and succeeding workshops. Several examples follow.

- **Designing a function with no input and no output values**

Displaying a script/program header to the screen is an example of a function that does not require any values to perform its task or create any values from performing its task. Applying the function design steps:

`function wkshp7_header`

(1) Function to display the script/program header to the screen.
(2) Produces no values.
(3) Needs no values.
(4) Algorithm
    `disp` statements to display header text

The MATLAB code that embodies this outline is

```
function  [] = wkshp7_header( )
%
% Header display function for Workshop 7
% Your Name(s)
% Class (e.g. Engr 0012, TH 10:00, Instructor Name)
% Today's Date
% Your e-mail address

% display header to screen
    clc
    disp(' ')
    disp('Solution to Workshop 7')
    disp('Your Name(s)')
    disp('Class (e.g. Engr 0012, TH 10:00, Instructor Name)')
    disp('Today''s Date')
    disp('Your e-mail address')
    disp(' ')
```

(1) Create the function **`wkshp7_ac1_header.m`** using the MATLAB editor by cutting and pasting appropriate lines from **`wkshp5_ac1.m`**. Alternatively, you could create the function file by typing the above code into the editor.

Test your function by entering

```
» wkshp7_header

Solution to Workshop 7
Your Name(s)
Class (e.g. Engr 0012, TH 10:00, Instructor Name)
Today's Date
Your e-mail address
```

Typing

```
    » help wkshp7_ac1_header

      Header display function for Workshop 7
      Your Name(s)
      Class (e.g. Engr 0012, TH 10:00, Instructor Name)
      Today's Date
      Your e-mail address
```

will display the comment lines immediately following the **function** line in the file.

- **Designing a function with one output value but no input values**

Turning conversion factors into functions has at least two very large advantages: (1) The functions are always available when needed if you store them in your MATLAB path; and (2) once they are created, you never need to look up the conversion factor again - you can simply call the function. Applying the function design steps:

```
function  ft_to_m
```

(1) Function to get conversion factor for feet to meters.
(2) Produces conversion factor.
(3) Needs no values.
(4) Algorithm

```
    convfac <-- 0.3048
```

The MATLAB code that embodies this outline is

```
function  [convfac] = ft_to_m( )
%
% Function to return conversion factor for feet to meters
% Your Name(s)
% Class (e.g. Engr 0012, TH 10:00, Instructor Name)
% Today's Date
% Your e-mail address

% Variable dictionary
%   convfac           conversion factor for ft to m

% set conversion factor
    convfac = 0.3048;
```

Note the use of names in this function. Because **ft_to_m** is the name of the function it cannot be used as a variable name within the function. To do so would confuse MATLAB tremendously. Thus, the variable name **convfac** is used to hold the conversion factor value in the function. Placing **convfac** in the output variable (value) list returns the <u>value</u> (not the name) to the calling location. Also note that this function requires a variable dictionary.

(2) Create the function **ft_to_m.m** using the MATLAB editor by cutting and pasting appropriate lines from **wkshp5_ac1.m**. Alternatively, you could create the function file by typing the above code into the editor.

Test your function by entering

```
» ft_to_m
ans =
    0.3048
```

Note that the value of the conversion factor was returned. You could use this directly on the right hand side of an assignment statement or assign the value to a variable for later use.

(3) Create the functions **`yd_to_m.m`** and **`atm_to_Pa.m`**. Apply the function design steps before coding the functions. Test your functions to see whether they return the proper value.

- **Designing a function with more than one output value but no input values**

A common activity or task is to get input values from the user for use later in the script. The important thing in designing for this task is to decide what values (and in what units) are really required for further computations. This then defines what values to ask for and what conversions might have to take place before the values are returned to the calling location.

Applying the function design steps to the present workshop:

`function  wkshp7_getdata`

(1) Function to get fluid density (rho) in SI, gravitational acceleration (g) in f/s/s, tank height (height) in yards, and atmospheric pressure (pout) in atmospheres from user and return all in SI units to calling location.
(2) Produces **`rho`**, **`g`**, **`height`**, **`pout`** - all in SI units.
(3) Needs no values - will get them from the user.
(4) Algorithm
    `input` statement for each value followed by conversion to SI if necessary.

The MATLAB code that embodies this outline is

```
function  [rho, g, height, pout] = wkshp7_getdata( )
%
% Function to get fluid density (rho) in SI, gravitational
% acceleration (g) in ft/s/s, tank height (height) in yards,
% and atmospheric pressure (pout) in atmospheres from user
% and return all in SI units to calling location.
%
% Your Name(s)
% Class (e.g. Engr 0012, TH 10:00, Instructor Name)
% Today's Date
% Your e-mail address

% Variable dictionary
%   rho          fluid density, kg/m^3
%   g            gravitational acceleration, m/s/s
%   height       tank height, m
%   pout         atmospheric pressure, Pa

% get user input
```

```
        rho = input('Please input fluid density in kg/m^3 ==> ');
        g = input('Please input grav accel in ft/s/s ==> ');
        height = input('Please input tank height in yd ==> ');
        pout = input('Please input atm pressure in atm ==> ');

    % convert to SI units
        g = ft_to_m*g;
        height = yd_to_m*height;
        pout = atm_to_Pa*pout;
```

Note that only a single variable is needed for each input and converted value. Also note that use is made of the user-defined conversion factor functions **`ft_to_m.m`**, **`yd_to_m.m`**, and **`atm_to_Pa.m`** in the last three lines of the function.

(4) Create the function **`wkshp7_getdata.m`** using the MATLAB editor by cutting and pasting appropriate lines from **`wkshp5_ac1.m`**. Alternatively, you could create the function file by typing the above code into the editor.

Test your function by entering

```
    » wkshp7_getdata
    Please input fluid density in kg/m^3 ==> 1
    Please input grav accel in ft/s/s ==> 1
    Please input tank height in yd ==> 1
    Please input atm pressure in atm ==> 1
    ans =
         1
```

Note that only one value was reported (as ans) - yet four should have been returned. If you want to capture all four values, you will need to "call" the function, giving variable names to assign the values to. Try

```
    » [rho, g, height, pout] = wkshp7_getdata
    Please input fluid density in kg/m^3 ==> 1
    Please input grav accel in ft/s/s ==> 1
    Please input tank height in yd ==> 1
    Please input atm pressure in atm ==> 1
    rho =
         1
    g =
        0.3048
    height =
        0.9144
    pout =
          101300
```

Now all four values have been captured by the call and assigned to the specified variables. Note that even though the value 1 was input for all values, the "converted" values (SI units) were returned for use.

Finally, try this (add a semi-colon to the end of the function call)

```
    » [rho, g, height, pout] = wkshp7_getdata;
    Please input fluid density in kg/m^3 ==> 1
    Please input grav accel in ft/s/s ==> 1
    Please input tank height in yd ==> 1
```

```
Please input atm pressure in atm ==> 1
»
```

As before, use of the semi-colon suppresses display of the values assigned to the output variables.

- **Designing a function with one output value and more than one input value**

Although we would not usually design a calculation function for one line of calculation, we do so here in order to show a function that requires input values to do a calculation. The design of this type of function assumes that the values have already been obtained from the user or produced by another function and are available in the workspace.

function wkshp7_pbot

(1) Function to calculate absolute pressure at the bottom of the tank in SI units.

(2) Produces absolute pressure at bottom of the tank (**pbot**) in SI units.

(3) Requires fluid density (**rho**), gravitational acceleration (**g**), tank height (**height**), and atmospheric pressure (**patm**) - all in SI units.

(4) Algorithm

```
    pbot <-- rho*g*height+patm
```

The MATLAB code that embodies this outline is

```
function  [pbot] = wkshp7_pbot(rho, g, height, patm)
%
% Function to calculate absolute pressure
% at the bottom of the tank in SI units
%
% Your Name(s)
% Class (e.g. Engr 0012, TH 10:00, Instructor Name)
% Today's Date
% Your e-mail address

% Variable dictionary
%   rho          fluid density, kg/m^3
%   g            gravitational acceleration, m/s/s
%   height       tank height, m
%   patm         atmospheric pressure, Pa
%   pbot         pressure at tank bottom, Pa

% calculate pressure
    pbot = rho*g*height+patm;
```

(5) Create the function **wkshp7_pbot.m** using the MATLAB editor by cutting and pasting appropriate lines from **wkshp5_ac1** and the other Workshop 7 functions that have been created. Alternatively, you could create the function file by typing the above code into the editor.

Test your function by entering

```
» wkshp7_pbot(1,1,1,1)
ans =
     2
```

Why is the answer 2? Is this a good test of the function?

- **Designing a function with no output value and more than one input value**

Functions with input values but no output values are typically found at the end of a script and are used to display the results from running the script.

function wkshp7_results

(1) Function to display the results of calculating the pressure at the bottom of the tank.
(2) Produces nothing.
(3) Requires fluid density (**rho**), gravitational acceleration (**g**), tank height (**height**), and atmospheric pressure (**patm**) , and bottom pressure (**pbot**) - all in SI units.
(4) Algorithm
    disp call to display information and results

The MATLAB code that embodies this outline is

```
function  [ ] = wkshp7_results(rho, g, height, patm, pbot)
%
% Function to calculate absolute pressure
% at the bottom of the tank in SI units
%
% Your Name(s)
% Class (e.g. Engr 0012, TH 10:00, Instructor Name)
% Today's Date
% Your e-mail address

% Variable dictionary
%   rho          fluid density, kg/m^3
%   g            gravitational acceleration, m/s/s
%   height       tank height, m
%   patm         atmospheric pressure, Pa
%   pbot         pressure at tank bottom, Pa

% display results
    disp( ' ' )
    disp( 'Tank of height, m' )
    disp( height )
    disp( 'containing fluid of density, kg/m^3' )
    disp( rho )
    disp( 'in gravitational field, m/s/s' )
    disp( g )
    disp( 'and atmospheric pressure, Pa' )
    disp( patm )
    disp( ' ' )
    disp( 'has absolute pressure at bottom, Pa' )
    disp( pbot )
```

(6) Create the function **wkshp7_results.m** using the MATLAB editor by cutting and pasting appropriate lines from **wkshp5_ac1** and the other Workshop 7 functions that have been

created. Alternatively, you could create the function file by typing the above code into the editor.

How would you test this function for proper operation?

- **Script for `wkshp7.m`**

We have now created all of the functions necessary to run the script **`wkshp7.m`**, which is available on the program disk and should be placed in your current directory. Try it using the same values used to test the original Workshop 2, Activity 2 problem (rho = 1000 kg/m^3, g = 32.2 ft/s/s, height = 7 yards, atmospheric pressure = 1 atm). It should produce the same answer as obtained in Workshop 2.

- **Closing notes**

User-defined functions are very useful because they allow us to focus on **what** is happening in a program. They also allow us to focus on a small portion of the overall script requirements (such as an input or output function) and develop a solution for the small portion following the four steps of the function design procedure. The solution can then be tested independently of other functions or the overall problem. When all of the functions required for a script have been designed and tested, putting them together in the proper sequence should create a script that produces the proper answers.

- **Exercises**:

1. Convert the script **`wkshp5_ex1.m`**. to one which uses functions.
2. Convert the script **`wkshp5_ex2.m`**. to one which uses functions.
3. Convert the script **`wkshp5_ex3.m`**. to one which uses functions.
4. Convert the script **`wkshp5_ex4.m`**. to one which uses functions.
5. Convert the script **`wkshp5_ex5.m`**. to one which uses functions.
6. Convert the script **`wkshp5_ex6.m`**. to one which uses functions.

Recap: You should have learned
- The basic ideas behind functions and function organization.
- The basic design algorithm (procedure) for functions.
- That MATLAB functions are saved as **`m-files`**.
- That all functions must have a header section.
- That all functions must have a variable dictionary as appropriate.
- That functions can be tested independently of scripts.