AYDIN
ADNAN MENDERES UNIVERSITY
FACULTY OF ENGINEERING
Department of Electrical and Electronics Engineering

EE214 – Computer Aided Engineering Applications

Prof. Dr. Yılmaz KALKAN

EE214 – Computer Aided Engineering Applications

Lecture 6

User Defined Scripts & Functions

Objective

- **Describe how functions are created & used.**

User-defined scripts & functions

- MATLAB is built around **commands** & **functions**. Both sets are computer codes that accept input from the user and provide output. The latter does not use or save variables in the workspace.

- **Functions and M-file command scripts** allow you to do technical computing efficiently.

- Well designed code helps you to tasks **multiple times**.

- It is necessary for you, as the programmer, to know exactly **how a function performs its task**; otherwise, how would you know that it is doing the task correctly.

Concept on Function M-files

- **User-defined functions** are like the MATLAB pre-defined functions (sin, max, … etc.).

- They take a certain number of **inputs**, perform some **operation**, and give **output(s)**.

- Just as with MATLAB built-in functions, we need to know what they are supposed to do and know that it does it correctly.

Input/output & Functions

- Several kinds of functions can be created with different combinations of input and/or output:
  - **Functions with input and output arguments**
  - **Functions with input arguments and no output arguments**
  - **Functions with output arguments and no input arguments**
  - **Functions with neither input arguments nor output arguments**

```
function c = addme(a,b)
% ADDME  Add two values together.
%   C = ADDME(A) adds A to itself.
%
%   C = ADDME(A,B) adds A and B together.
%
%   See also SUM, PLUS.

switch nargin
    case 2
        c = a + b;
    case 1
        c = a + a;
    otherwise
        c = 0;
end
```

nargin returns the number of function input arguments given in the call to the currently executing function.

Syntax for Functions

- Calling a user-defined function:
  - my_function(x)

- Defining a user-defined function:
  - function y = my_function(x)
  - x is a value that is the input to my_function.
  - my_function performs a functional operation.
  - y is a value that is the output of my_function.

- Functions must be written in M-files. The M-file must have the same name as the function.

```
function c = addme(a,b)
% ADDME  Add two values together.
%   C = ADDME(A) adds A to itself.
%
%   C = ADDME(A,B) adds A and B together.
%
%   See also SUM, PLUS.

switch nargin
    case 2
        c = a + b;
    case 1
        c = a + a;
    otherwise
        c = 0;
end
```

Notes on Functions

- The **convention for naming functions** is the same as for variables. A function name *must start with a letter, should be meaningful, should not use the name of an existing function, and should not be excessively long*.

- It is important that you give **meaningful variable names** to **variables inside a function** that you write, so that you and others can understand what the function does.

- **Comments** become extremely important in functions. Comments help both you and anyone who might use the function to understand what it does.

Examples of function M-files

Write a MATLAB function to calculate the volume of a sphere

```
volume = (4/3)*pi*(radius^3)
```

Write a MATLAB function to calculate the perimeter of a square

```
perimeter = 4 * side
```

Write a MATLAB function to find the square root of a number

```
root = x^(1/2)
```

Examples of function M-files

```
function volume = volume_sphere(radius)
    volume = (4/3)*pi.*(radius^3);
```

```
function perimeter = perimeter_square(side)
    perimeter = 4 .* side;
```

```
function root = square_root(x)
   root = x.^(1/2);
```

Using a function in the Command Window

- User-defined functions are **accessed** in the same way as any **MATLAB built-in functions**.
- The function M-file, in this case the perimeter_square function from the previous slide, must be saved as "perimeter_square.m" in your current directory (otherwise it must be in the path)*.
- You use it by typing in the command window:

```
perimeter_square(4)
ans=16
```

*Note: The directory that your function is saved in MUST be in the path; otherwise, MATLAB will not be able to find and run the file.

Functions with no input

- If you need to **access some constant value several times**, instead of hard-coding the constant value every time it is needed, we can hard-code it once into a function that has no input.
- When the constant is needed, its value is called via the function.
- As an example, if we needed, for whatever reason, the mass of the earth (in kg) multiple times, we could write a function to store it as follows:

```
function mass_of_earth = moe()
mass_of_earth = 5.976E24;
```

Functions with no output

- Functions with no output can be used for many purposes, such as to **plot figures** from a set of input data or **print input information to the command window** (output here implies results from operations on input; not displays).

- The built-in MATLAB function **tic** has no output. It starts a timer going for use with another built-in MATLAB function, **toc**.

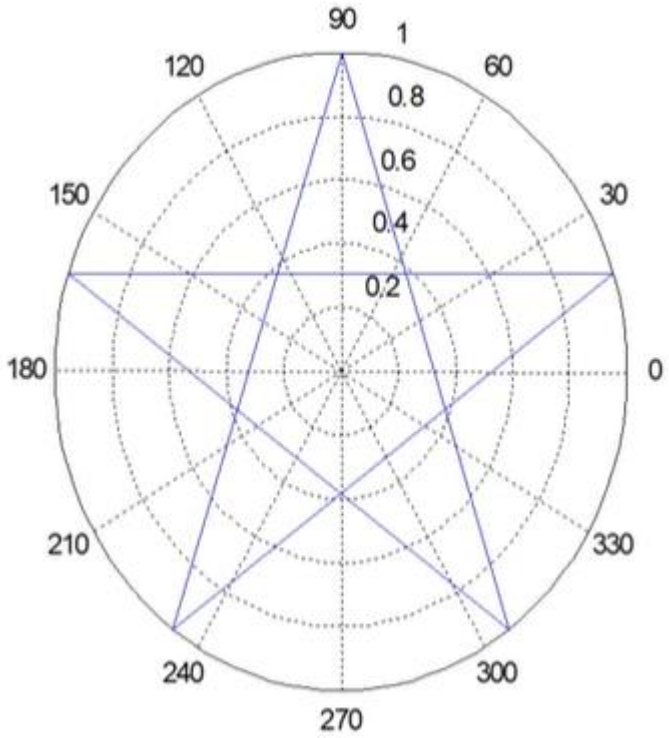
Functions with no input or output

- A function with no input or output consists of some **hard-coded information** that is to be used in a specific procedure.

- An example of this is a function that **plots a specific figure**. Plotting the figure is the only thing the function does; it does not have any input or operations that lead to computed values for output.

A Function with no input or output

```
function [] = star()
theta = pi/2 : 0.8 * pi :
        4.8 * pi;
r = [1 1 1 1 1 1];
polar(theta, r);
```



Although the star function generates a plot, in the strictest sense it does not have any output. If you try to evaluate the expression A=star in MATLAB, you will get an error, because star has no output.

Notes on variables

- Variables that are created **inside of a user-defined function** may only be accessed from inside of that function. They are referred to as **local variables**.
- After the function completes its operations, the local **variables are deleted** from memory.
- The only variable that appears in the workspace is the **output**, if any, of the function.
- Conversely, functions cannot access variables from the workspace (with the exception of any input parameters they might have or "global variables"---check MATLAB help on this matter).

The **type** command

- The **type** command followed by **the name of an M-file** prints the contents of the M-file onto the Command Window. This includes any user-defined or built-in functions and command scripts available in MATLAB. Using the type command instead of the editor to view the contents of M-files can **prevent you from accidentally changing the contents** of your files.

- The type command is an alternative to using the Help Menu or editor for looking up details associated with functions & commands.

Exercises

- Write a function that converts Celsius temperatures into Fahrenheit.
- Write a function that takes force, displacement, and angle (in degrees) as input and outputs work. Use MATLAB help to find a cosine function that uses degrees.
- Write a function that takes the number of stairs, height of each stair (in meters), the mass of an ascending subject (in kg), and the time of ascend (in seconds) as input and outputs mechanical work, power, and change in potential energy.

Summary

- Function concept and syntax
- Different kinds of functions
  - Functions with input and output
  - Functions with input only
  - Functions with output only
  - Functions with no input or output