SEEE1022 INTRODUCTION TO SCIENTIFIC PROGRAMMING

CH5 Function

Dr. Mohd Saiful Azimi Mahmud (azimi@utm.my)
P19a-04-03-30, School of Electrical Engineering, UTM

OBJECTIVES

1. **Create Function**
   - Create and use M-file functions with both single and multiple inputs and outputs.
   - Create and use local functions and nested functions.

2. **Variable scope**
   - Understand the difference between base and function workspace.
   - Understand the variable scope of command window, script, main function, local function and nested function.

3. **Toolbox**
   - Understand what is MATLAB toolbox and how to use it.

OPERATION (RECAP)

STATEMENT

1. ASSIGNMENT

Assign array to a variable using EQUAL OPERATOR

syntax

```
variable = expression
```

TYPE

- Arithmeti
- Boolean

2. REPETITION

Execute statements specified number of times using BOOLEAN OPERATOR or EQUAL OPERATOR

```
while
for
```

3. DECISION

Execute statements if condition is TRUE using BOOLEAN OPERATOR

```
if
if-else
if-elseif
Switch-case
```

FUNCTION

Group of statement that perform a task

syntax

```
[output1,…,outputM]
     = functionName(input1 … inputN)
```

COMMAND

Function that mostly used to change the environment

syntax

```
functionName input1 … inputN
```

Input, if exist, must only be string

MATLAB FUNCTION

WHAT IS FUNCTION?

- **Program file**
  - A .m file that contains a program.
  - There are two types of program file:
    1. Script file- contain sequence of commands, function calls and followed by local function if needed
    2. Function file – contain only functions.

- **What is function?**
  - Programs that accept inputs and return outputs.

- **Why use function?**
  - Cleaner code
  - Maintainability
  - Reuse
  - Hiding implementation

MATLAB SCRIPT

FUNCTION VS SCRIPT

Criteria	Script	Function
Usage	Both scripts and functions allow you to reuse sequences of commands by storing them in program files.	
Program	Fixed variables.	A more flexible and easily extensible program
Input/output	No	Yes
Workspace	Base workspace	Function workspace

MATLAB SCRIPT

TYPES OF FUNCTION

- **Predefined MATLAB function**
  - Built in function – the code is not accessible to user.
  - M-file function – the code is accessible to user.

- **User defined function:**
  - M-file function
  - Local function
  - Nested function
  - Anonymous function
  - Private function

PRE-DEFINED MATLAB FUNCTION

PREDEFINED MATLAB FUNCTION

EXAMPLE 1

```
>> type sin
'sin' is a built-in function.
```

EXAMPLE 2

```
>> type sphere
function [xx,yy,zz] = sphere(varargin)
%SPHERE Generate sphere.
 .
 .
 .
if nargout == 0
    cax = newplot(cax);
    surf(x,y,z,'parent',cax)
else
    xx = x; yy = y; zz = z;
end
```

The sphere function is stored as a function M-file, but is provided by MATLAB. Studying these functions may help you understand how to program better functions yourself

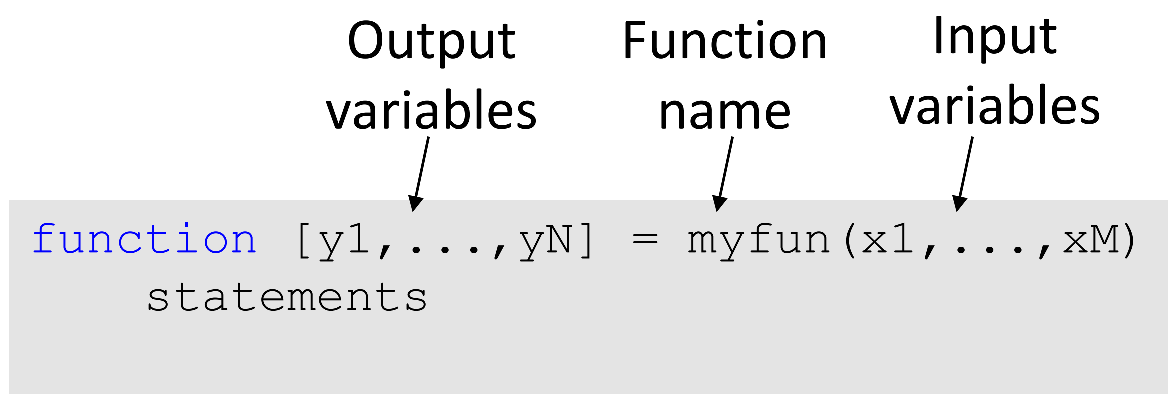
UTM

USER DEFINED FUNCTION

USER DEFINED FUNCTION

M-FILE FUNCTION

- User defined functions are stored as separate M-files.
- Once created, it is available in the command window and MATLAB script.
- To use them, they must be in the current directory.

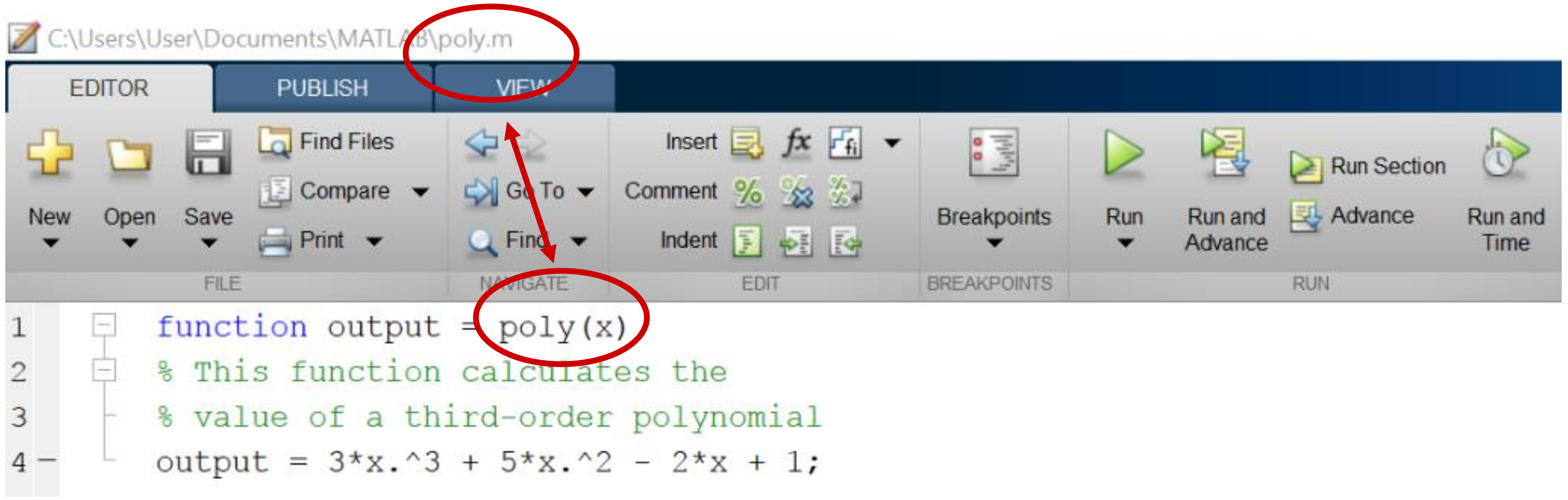


```
function [y1,...,yN] = myfun(x1,...,xM)
    statements
```

USER DEFINED FUNCTION

CREATING M-FILE FUNCTION

EXAMPLE 3



```
function output = poly(x)
% This function calculates the
% value of a third-order polynomial
output = 3*x.^3 + 5*x.^2 - 2*x + 1;
```

```
>> x = poly(3)

x =
   121
```

- The function name must be the same as the file name.
- Input argument is written inside a bracket after the function name.
- Output argument is written before the equal symbol.

USER DEFINED FUNCTION

ADD HELP TO THE PROGRAM

EXAMPLE 4

```
function output = g(x,y)
% This function multiply x and y
% x and y must be the same size
output = x.*y;
```

```
>> help g
  This function multiply x and y
  x and y must be the same size
```

- Comment lines after the function line will appear as the help information.

USER DEFINED FUNCTION

MULTIPLE INPUTS

EXAMPLE 5

```
function output = g(x,y)
% This function multiply x and y
% x and y must be the same size
output = x.*y;
```

```
>> a = [1 3 5];
>> b = [2 3 2];
>> c = g(a,b)

c =
     2     9    10
```

- Multiple inputs are separated by comma.

USER DEFINED FUNCTION

HANDLING INPUTS

EXAMPLE 6

```
>> x = g(3)
Not enough input arguments.

Error in g (line 4)
output = x.*y;
```

- Since line 4 needs two inputs, executing the function return an error of not enough input arguments.
- To create a flexible input or a function that accept any number of input arguments, variables called `varargin` and `nargin` can be used to handle the inputs.
- However, these advance variables is not covered in this course. Instead, refer to MATLAB documentation for detail explanation.

USER DEFINED FUNCTION

MULTIPLE OUTPUTS

EXAMPLE 7

```
function [dist, vel, accel] = motion(t)
% this function calculates the distance, velocity
% and acceleration of a car for a given value of time t,
% assuming all of the three parameters are initially 0.
 dist = t.^3/12;
  vel = t.^2/4;
accel = 0.5.*t;
```

```
>> [dist,vel,accel] = motion(5)

dist =
    10.4167

vel =
    6.2500

accel =
    2.5000
```

- Use square bracket and comma to create multiple outputs.
- When using a function, different variables name is allowed (detail discussion, later in variable scope topic).

USER DEFINED FUNCTION

OUTPUT FLEXIBILITY

EXAMPLE 8

```
>> t = 0:0.1:0.5
t =
       0    0.1000    0.2000    0.3000    0.4000    0.5000

>> dist = motion(t)
dist =
       0    0.0001    0.0007    0.0022    0.0053    0.0104

>> [dist,vel] = motion(t)

dist =
       0    0.0001    0.0007    0.0022    0.0053    0.0104
vel =
       0    0.0025    0.0100    0.0225    0.0400    0.0625
```

If you don't ask for all output variables, the program returns the number of variables you asked starting from the left.

USER DEFINED FUNCTION

OUTPUT FLEXIBILITY

EXAMPLE 9

```
>> t = 0:0.1:0.5
t =
       0    0.1000    0.2000    0.3000    0.4000    0.5000

>> [dist,accel] = motion(t)

dist   =
       0    0.0001    0.0007    0.0022    0.0053    0.0104

accel =
       0    0.0025    0.0100    0.0225    0.0400    0.0625
```

- If you need to return the last output variable, all of the preceding output variables need to be returned as output.
- In this example, pecutan is not acceleration but velocity, since the function written it as the second variable.

USER DEFINED FUNCTION

FUNCTION WITH NO INPUT/OUTPUT

EXAMPLE 10

```
function [] = stars()
asteriks = char( abs('*')*ones(1,10));
disp( asteriks )
```

```
function stars
asteriks = char( abs('*')*ones(1,10));
disp( asteriks )
```

```
>> stars()

**********

>> stars

**********
```

- No input/output argument can be written with either empty bracket or no bracket at all.
- Using the function can also be with or without the empty bracket.
- Alternatively, you can replace function with no input and output with simply a script file.

LOCAL FUNCTIONS

- MATLAB® program files can contain more than one function.
- Recap: There are two types of program files:
  1. Script file
  2. Function file
- Local function can exist on both program files.
- Local function is also known as sub-function.

UTM UNIVERSITI TEKNOLOGI MALAYSIA

USER DEFINED FUNCTION

MAIN VS LOCAL FUNCTION

Criteria	Main Function	Local Function
Script file	Not exist.	Created after the last line of script code.
Function file	The first function.	Additional function
Accessibility	Can be accessed by other function file and command window.	Can be accessed only within the function file.
Function name	Must be unique to all MATLAB functions.	Must be unique only within the function file.

USER DEFINED FUNCTION

LOCAL FUNCTION IN FUNCTION FILE

Syntax

```
function [y1,...,yN] = main(x1,...,xM)
    statements



function [y1,...,yN] = local1(x1,...,xM)
    statements

function [y1,...,yN] = local2(x1,...,xM)
    statements
 .
 .
 .
function [y1,...,yN] = localN(x1,...,xM)
    statements
```

USER DEFINED FUNCTION

LOCAL FUNCTION

EXAMPLE 11

```
function [add_result, minus_result] = compute(x,y)
% This function add and subtract the elements
% stored in x and y
add_result = add(x,y);
minus_result = minuss(x,y);

function result = add(x,y)
result = x+y;

function result = minuss(x,y)
result = x-y;
```

```
>> [plus,minus] = compute(2,3)

plus =
     5

minus =
    -1
```

USER DEFINED FUNCTION

NESTED FUNCTION

- A function that is contained within a parent function.
- Both main and local functions in a program file can include a nested function.
- Since nested function resides within a function, end command is use to mark the end of the nested function and the parent function.
- **Recap:** previously end command is not compulsory for both main and local functions.

USER DEFINED FUNCTION

NESTED FUNCTION

EXAMPLE 12

```
function [circumference, area] = mycircle(d)
r = d/2;
circumference = ci;     area = ar;

disp(['Circumference: ',num2str(circumference)])
disp(['          Area: ',num2str(area)])

    function c = ci()
    c = pi*d;
    end

    function c = ar()
    c = pi*r.^2;
    end
end
```

```
>> [a,b] = mycircle([2 3]);
Circumference: 6.2832      9.4248
         Area: 3.1416      7.0686
```

- `end` commands is use to show that the nested function reside within the parent function.

VARIABLE SCOPE

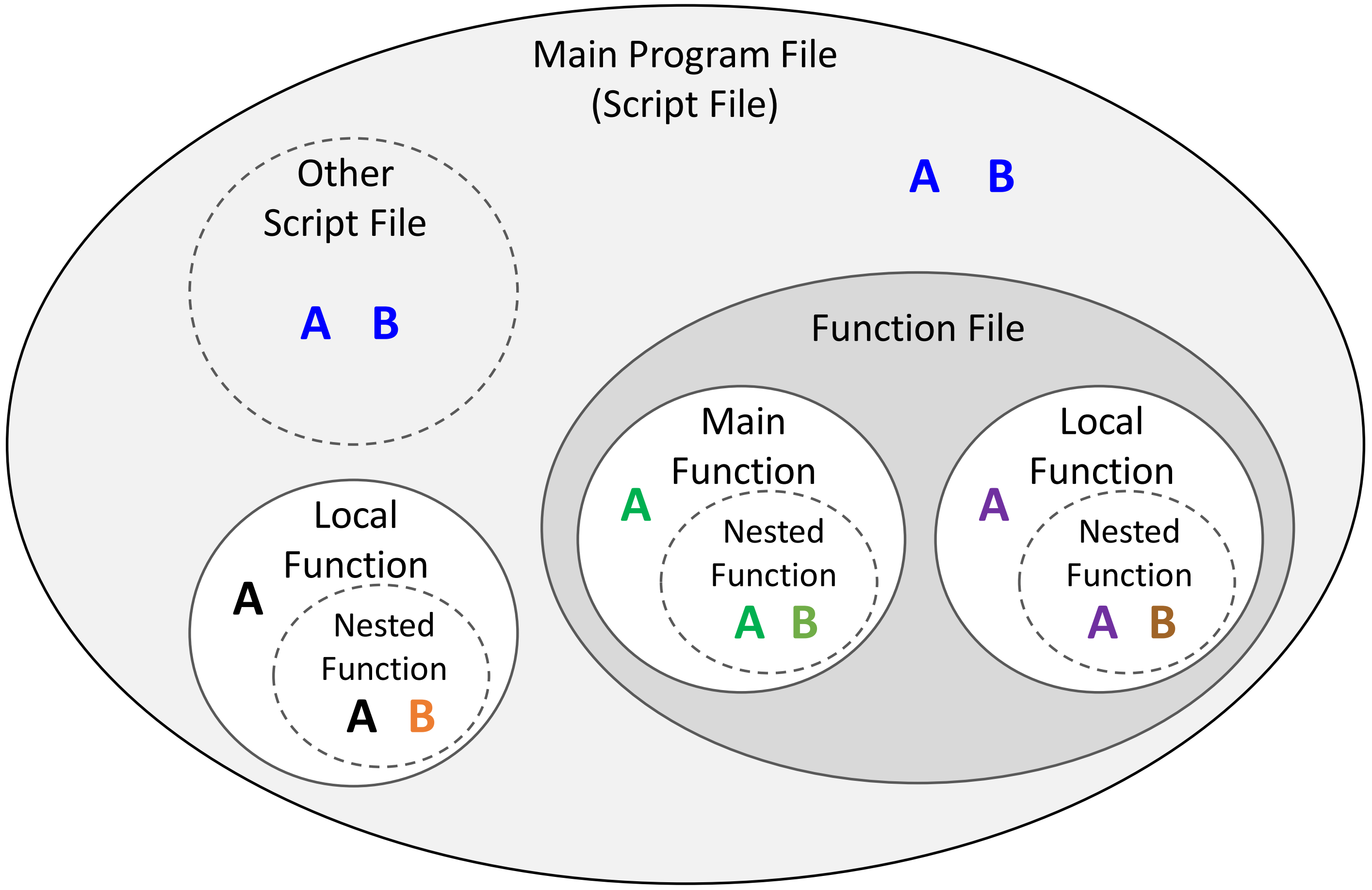
VARIABLE SCOPE

BASE AND FUNCTION WORKSPACE

- **Base workspace**
  - Stores variables that you create at the command line. This includes any variables that scripts create.
  - Stored variables are shown in the MATLAB workspace window.
- **Function workspace**
  - Functions do not use the base workspace.
  - Every function has its own ***function workspace***.
  - Each function workspace is separated from the base workspace and all other workspaces to protect the integrity of the data.
  - Variables specific to a function workspace are called ***local variables***.
  - Local variables are not known to other functions, other script file and command window. Thus, they are not shown in the MATLAB workspace window.

VARIABLE SCOPE

SCOPE



🟦 Base variables

⬛🟨🟩🟥🟪🟫 Local variables

VARIABLE SCOPE

DIFFERENT WORKSPACE

EXAMPLE 13

```
function result = distance(t)
result = 1/2*g*t.^2;
```

```
>> g = 9.8;
>> T = 0:0.1:1;
>> a = distance(T)
Undefined function or variable ‘g'.

Error in distance (line 2)
result = 1/2*g*t.^2;
```

- `g` need to be defined inside function `distance`, not in the command window because function `distance` has a different workspace.

VARIABLE SCOPE

PASSING ARGUMENTS

EXAMPLE 14

```
function [avg, med] = mystats(x)
n = length(x);
avg = mymean(x,n);
med = mymedian(x,n);

function a = mymean(v,n)
% MYMEAN Example of a local function.
a = sum(v)/n;

function m = mymedian(v,n)
% MYMEDIAN Another example of a local function.
w = sort(v);
if rem(n,2) == 1
    m = w((n + 1)/2);
else
    m = (w(n/2) + w(n/2 + 1))/2;
end
```

VARIABLE SCOPE

PASSING ARGUMENTS (CONT.)

EXAMPLE 14

```
>> data = randi(100,1,200);
>> [a,b] = mystats(data)

a =
    49.4850

b =
    49
```

- `data` is a base workspace variable. It is pass to the `mystats` function through its input argument `x`.
- In function `mystats`, `x` is then pass to function `mymean` and `mymedian` through input arguments `v`.
- Result of the mean and median are then pass back to the `mystats` function through output arguments `a` and `m` respectively.
- Lastly, the `avg` and `med` variables are pass back to the base workspace into `a` and `b` respectively.

VARIABLE SCOPE

NESTED FUNCTION SCOPE

EXAMPLE 15

```
function [circumference, area] = mycircle(d)
r = d/2;
circumference = ci;     area = ar;

disp(['Circumference: ',num2str(circumference)])
disp(['         Area: ',num2str(area)])

    function c = ci()
    c = pi*d;
    end

    function c = ar()
    c = pi*r.^2;
    end
end
```

```
>> [a,b] = mycircle([2 3]);
Circumference: 6.2832      9.4248
         Area: 3.1416      7.0686
```

- Parent variables `r` and `d` are accessible to the nested function.
- Variable `c` remain local at both of the nested functions because it is not use by the parent function.

VARIABLE SCOPE

COMMAND

- Command is function that altered the environment, but does not return result.
- Syntax:

```
functionName input1 … inputN
```

- Input, if exist can only be string. Thus, **command is generally a function that takes string arguments**.
- Example of command:
  - `clear, who, save, axis`
- Since command is a function, then a new command can be created using M-file similar to how a function is created.

TOOLBOX

UTM UNIVERSITI TEKNOLOGI MALAYSIA

TOOLBOX

WHAT IS TOOLBOX?

- Toolbox is a collection of the followings into one package:
    1) MATLAB functions
    2) System object
    3) Simulink blocks
    4) Examples
    5) Documentation
    6) Data
    7) Apps
- Toolboxes can be add-on to the MATLAB from three sources:
    1) MathWork toolbox – official toolboxes.
    2) Community toolbox – created and shared by MATLAB users.
    3) Shared toolbox installation file – the .mltbx file.

TOOLBOX

CREATING YOUR OWN TOOLBOX

- Once you've created a set of functions, you can manage them by either:
  1) Group them into a directory (folder) and add them to the MATLAB search path.
  2) Package the functions into MATLAB toolbox, which gives you a .mltbx file. It is an installation file of the toolbox.

  The .mltbx can be easily shared by typical file sharing method. You can also upload the file to MATLAB Central File Exchange where other users can download your toolbox from within MATLAB.

- To create a toolbox installation file:
  - In the **Environment** section of the **Home** tab, select **Package Toolbox** from the **Add-Ons** menu.
  - In the Package a Toolbox dialog box, click the ✚ button and select your toolbox folder.
  - Fill in all required information and click ✓ **Package** to finish the process.

KERANA TUHAN UNTUK MANUSIA • UNIVERSITI TEKNOLOGI MALAYSIA •

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

univteknologimalaysia

utm_my

utmofficial

Thank You

www.utm.my

innovative ● entrepreneurial ● global