TOPIC 10
USING SCRIPTS M-FILE

There are two kinds of M-files:

- Scripts, which do not accept input arguments or return output arguments. They operate on data in the workspace.
- Functions, which can accept input arguments and return output arguments. Internal variables are local to the function.

MATLAB scripts:

- Are useful for automating a series of steps you need to perform many times.
- Do not accept input arguments or return output arguments.
- Store variables in a workspace that is shared with other scripts and with the MATLAB command line interface.

Script files are nothing more then a sequence of commands saved as a text file with .m extension. Such writing is usually done in Matlab editor, but actually, no Matlab editor is required to create and edit such files, notepad program can be used as well.

Scripts are the simplest kind of M-file because they have no input or output arguments. They are useful for automating series of MATLAB commands, such as computations that you have to perform repeatedly from the command line.

example:

To find the area of triangle , the sequence of commands in **command window** are below:

```
» a=16;
» t=11;
» area=(a*t)/2
```

Using m-file, the commands will be done as below:

1. Click menu **File – New – M-file**
2. In m-file editor type:

```
a=16;
t=11;
 area=(a*t)/2
```

1. Click menu File – save as. Give file name **examp1**

To run the script, in command window type the file name

```
» examp1
```

Example2 :

```
x=rand(1,10)
average=sum(x)/length(x)
```

Save and give file name averg2
Run the file:

```
>> averg2
```

exercise:

- Create script m-file for finding the volume of cylinder, if given the diameter 8 cm, the height 12 cm. save as volume_cyl

Add comment

Comment in program is started by %. The command that is preceded by % is not execute.

Example3:
These statements calculate rho for several trigonometric functions of theta, then create a series of polar plots:
(type these commands in matlab editor, then save as flower.m)

```
% An M-file script to produce “flower petal" plots
theta = -pi:0.01:pi;              % Computations
rho(1,:) = 2 * sin(5 * theta) .^ 2;
rho(2,:) = cos(10 * theta) .^ 3;
rho(3,:) = sin(theta) .^ 2;
rho(4,:) = 5 * cos(3.5 * theta) .^ 3;
for k = 1:4
  polar(theta, rho(k,:))        % Graphics output
  pause
end
```

Interactive input

The response to the input prompt can be any MATLAB expression, which is evaluated using the variables in the current workspace.

user_entry = input('prompt') displays prompt as a prompt on the screen, waits for input from the keyboard, and returns the value entered in user_entry.

example

```
» a=input('Define value: ')
Define value: 20
a =
        20
```

user_entry = input('prompt','s') returns the entered string as a text variable rather than as a variable name or numerical value.

Example:

```
reply = input('Do you want more? Y/N [Y]: ','s');
if isempty(reply)
   reply = 'Y';
end, disp(reply)
```

Another example:

```
% this is example to find the area of triangle
a= input(‘ What is the base=  ‘);
t= input(‘ what is the height= ‘);
area=(a*t)/2
```

Using disp

- disp(X) displays an array in command window, without printing the variable’s name.
- If X contains a text string, the string is displayed.

```
» disp(['Elapsed Time = ',num2str(toc),' sec'])
  Elapsed Time =
        527.2 sec
```

example:

```
% this is example to find the area of triangle
a=16;    % base
t=11;    % height
area=(a*t)/2;  % hide the result
disp(‘the area of the triangle is)  %print the string
disp(area)       % show the result without ‘area’
```

Important functions that are often used in scripts (and functions)

return

The script execution can interrupted with *return* command. Then script execution reaches this command, no further commands are executed.

pause

Causes script (or function) execution to pause until the Enter key is pressed;

pause(t) will delay execution of the script by *t* seconds.

more

If multi page result is printed in command window *more on* command will delay the display after each screen until any key is pressed on a keyboard.

clc

Clear command window (completely).

home
Move cursor to upper left corner of Command Window
warning
A simple use includes switching on and off all Matlab warnings.

Usage: warning on, warning off, w=warning;
» 1/0
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this warning.)
ans =
Inf
» warning off
» 1/0
ans =
Inf
menu - generates menu of choices. Returns number of the chosen option.
» level=menu('Choose Level', 'Simple', 'Moderate', 'Difficult') % *waits after user's chose*
level =
3
First argument is the menu header. The rest are the displayed strings that correspond to possible choices.

Functions

Functions are similar to scripts. Major differences are that function can accept and return values, workspace variables are not visible from the function and that intermediate variables inside the function are automatically cleaned after function return.

MATLAB functions:

- Are useful for extending the MATLAB language for your application.
- Can accept input arguments and return output arguments.
- Store variables in a workspace internal to the function.

Function declaration should start with the word *'function'* and follow the syntax:

function output=function_name(input1,input2,...)

Here 'output' is the name of output variable, 'function_name' is the name of the function (also the name of -.M file), 'input1','input2' is the list of input variables.

Example of the function named 'absorb.m':

function f = absorb(epsilon, l, c)
% function f = absorb(epsilon, l, c) returns sample absorption in OD.

```
% epsilon - extinction coefficient 1/(cm * M), l - path length (cm)
% c - sample concentration (mol / L)
f= epsilon * l * c / log(10);
```

Number of output variables can be more than one. In this case variables should be listed inside the brackets:
function [output1 , output2, output3,....] =function_name(input1,input2,...)

Function can accept several variables with names separated by comma. Commented lines which follow directly the function declaration are this function help entry which can be viewed using *help* function.

Number of supplied input and output arguments can be checked using functions *nargin* and *nargout* respectively. This allows to build flexible control into function. If number of input arguments is less than in the function declaration, default value should be supplied for missing variable (if it is being used). Number of output variables also can be variable.

Example above can be modified such that function can output transmission instead.

```
function f = absorb(epsilon, l, c, do_OD)
.......
% first line in function
if nargin<4, do_OD=1; end
.......
% last line will be
if ~do_OD, f=1 - 10 ^ (-f); end
```