ENGINEERING MATHEMATICS-I MATLAB

Department of Science and Humanities

The M-Files:

- In MATLAB, we write programs in M-files.
- They are called M-files because they must have a .m at the end of their name, (for example, myfunction.m).
- M-files are ordinary ASCII text files written in MATLAB’s Language.
- M-files can be created using any editor or word processing applications.
- There are two types of M-files: **Script files and Function files**.

Script Files; Creating and Running Script Files

- A script file is an M-file with a set of valid MATLAB commands in it.
- To create scripts files, we need to use a text editor. We can open the MATLAB editor using the command prompt.
- If we use the command prompt, then we type **edit** and then the filename (with .m extension). This will open the editor.
- The above command will create the file in default MATLAB directory.
- Alternatively, we can choose NEW -> Script. This also opens the editor and creates a file named untitled. We can name and save the file after typing the code.

Creating and Running Script Files, Continued…

- Consider an example:
- Type the following code in the editor:

```
>> a=25; b=26; c=27; d=a+b+c
```

- After creating and saving the file, you can run it by clicking the Run button on the editor window.
- The command window prompt displays the result: d=78

Function Files

- A function is a group of statements that together perform a task.
- In MATLAB, functions are defined in separate files.
- Functions can accept more than one input arguments and may return more than one output arguments.
- The syntax of the function definition line is as follows:

  function [output variables] = function_name(input variables)
- Note that the function_name must be the same as the file name (without the m.extension) in which the function is written.

Function Files, Continued…

- Consider the following example:
- First, create a script file as myname.m for the following:

```
>> a=20; b=21; c=22;
sum=a+b+c     (Press Run icon)
 prod=a*b*c    (Press Run icon)
```

- Then the following result is displayed on the command window:

```
sum = 63; prod =9240.
```

Function Files, Continued…

Create a function file:

```
>> function [sum,prod] = myname(a,b,c)
sum = a+b+c;
 prod=a*b*c;
 end
```

On command window, type the following:

For, >> myname(20,21,22), the output is: ans =63

For, >> [s,p]=myname(20,21,22), the output is: s = 63, p =9240

Loop Control Statements

- With loop control statements, we can repeatedly execute a block of code. There are two types of loops: for loops and while loops.
- for loop: A for loop is used to repeat a statement or a group of statements for a fixed number of times.
- The syntax is:

```
for index = values
<program statements>
………………………………
end
```

for loop, Continued…

- For example, consider the following code:

```
for m=1:3
num=1/(m+1)
end
```

- When we run the file, it displays the following result:

```
num = 0.5000
num = 0.3333
num = 0.2500
```

for loop, Continued…

- For example, consider the following code:

```
for n=100:-2:96
k=1/(exp(n))
end
```

- When we run the file, it displays the following result:

```
k =3.7201e-44
k =2.7488e-43
k =2.0311e-42
```

for loop, Continued…

- For example, consider the following code:

```
>> for a = 1:-0.1: 0
disp(a)
end
```

- When we run the file, it displays the following result:

```
1 0.9000 0.8000 0.7000 0.6000 0.5000 0.4000 0.3000
0.2000 0.1000
```

(These numbers will be displayed as a column matrix)

while loop

- The while loop repeatedly executes statements while condition is true.
- The while loop repeatedly executes program statement(s) as long as the expression remains true.
- The syntax is:

```
while <expression>
<program statements>
………………………………
end
```

while loop, Continued…

- Consider the following example: The code is as follows:

```
>>  a = 10;
 while(a < 15)
fprintf('value of a: %d\n', a);
 a = a + 1;
 end
```

while loop, Continued…

- When we run the file, it displays the following result:

```
value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14
```

Conditional Statements

- There will be some situation where a program or a particular block has to be executed only when a specific condition is true. The conditional statements will be useful in such situations.
- The following are the conditional statements that we can use in MATLAB. if-end; if-else-end; if-elseif-elseif-else-end; switch case.

if-end statement

- if-end statement: It decides whether a particular block of code has to be executed or not, based on the given Boolean condition. When the given condition is true, only then it executes the statements inside the block otherwise not.

if-end statement, Continued…

- For example, consider the following:

```
>> number = 25;
if number > 20
fprintf('The number is greater than 20');
end
```

- When we run the file, it displays the following result:

```
The number is greater than 20.
```

if-end statement, Continued…

- For example, consider the following:

```
>> number = 50;
 if number > 60
 fprintf('The number is greater than 60');
 end
```

- When we run the file, no result will be displayed on the screen.

if-else-end statement

- An if statement can be followed by an optional else statement, which executes when the expression is false. Consider the following:

```
>> number = 25
 if number<20
 fprintf('The number is greater than 20')
else
 fprintf('The number is not less than 20')
 end
```

- When we run the file, it displays the following result: The number is not less than 20.

if-elseif-elseif-else-end

- We can use chain if-else-end statements with more than one condition. For example, consider the following:

```
>> number = 50;
if number<20
          fprintf('The number is less than 20\n');
 else
         if number<30
         fprintf('The number is less than 30\n');
else
         fprintf('The number is less than 60\n');
 end
 end
```

- Output: The number is less than 60.

Switch statement

- The switch statement is used to test for the equality against a set of known values.
- The syntax for switch statement is:

```
switch <switch_expression>
case <case_expression>
          <statements>
case <case_expression>
           <statements>
   ……………………………………….
 otherwise
           <statements>
 end
```

Switch statement, Continued…

- For example, consider the following:

```
>> grade = 'B';
     switch(grade)
     case 'A'
          fprintf('Excellent!\n' );
     case 'B'
           fprintf('Well done\n' );
     case 'C'
           fprintf('Well done\n' );
     case 'D'
           fprintf('You passed\n' );
```

Switch statement, Continued…

```
        case 'F'
               fprintf('Better try again\n' );
        otherwise
                fprintf('Invalid grade\n' );
 end
```

- When we run the file, it displays the following result: Well done

THANK YOU