**PROGRAMMING IN MATLAB II**

**M-FILES**

* Files that contain code (a sequence of MATLAB statements) in the MATLAB language are called M-files.
* M-files have the extension ".m".
* They are created by going File→New→M-file.
* M-Files can be scripts that simply execute a series of MATLAB statements, or they can be functions that can accept arguments and can produce one or more outputs.
* **Script Files**
* is a plain text file that contains one or more MATLAB commands and, optionally, comments. Thus, A Script File is a user-created file with a sequence of MATLAB commands in it.
* do not accept input arguments or return out put arguments.
* They operate on data in the work space.(i.e. Script variables are part of workspace)
* File must be at the *current working directory.*
* Script files have a filename extension “.m” .That is The file must be

saved with a ‘.m’ extension to its name, thereby, making it an M-file(e.g.try.m)

* script file are executed by typing the name of the script file without the extension at the MATLAB command prompt(i.e. when you type the name of the script, all of the commands contained in it are executed just as if you had typed these commands into the command window).
* if any of the variables in the script file have the same name as the ones in your current workspace, the values of those variables in the workspace are changed by the actions in the script.

* **Creating a script m-file**
* Script files are usually created with a plain text editor.

1. Use "New, M-File" under file menu or use any text editor

2 .Type in a series of commands, such as

* **n=1;**
* **R=0.0821;**
* **T=300;**
* **V=10:.5:30;**
* **P300 = n\*R\*T ./ V;**
* **T=200;**
* **P200 = n\*R\*T ./ V;**
* **plot(V,P200,'r-',V,P300,'g:')**
* **legend('T=200 Kelvin','T=300 Kelvin')**
* **xlabel('Volume (Liters)')**
* **ylabel('Pressure (Atmosphere)')**

3. Save as file name **plotgas.m**

4. In the Matlab window, type in **plotgas** and it will run your series of commands

**Example 2**

Let's put the commands for calculating the roots of a quadratic equation into a file called “quat.m”.

* Use the MATLAB editor to create a file: "**File** --> **New** --> **M-file**".
* Enter the following statements in the file:

a = 2;

b = 1;

c = -4;

x1=(-b+sqrt(b^2-4\*a\*c))/(2\*a)

x2=(-b-sqrt(b^2-4\*a\*c))/(2\*a)

* Save and name the file, for example, “quat.m”.
* Run the file, by typing “quat” at the MATLAB command line prompt:

>>quat

x1 =

1.1861

x2 =

-1.1861

Example 3

It is possible to modify the file “quat.m” so that it prompts you for inputting values of a, b, and c each time it runs. MATLAB treats everything that appears after the % on a line as comments and these lines will be ignored when the M-file runs. Here is the modified version of the file:

% ---------------------------------

% quat.m is to solve quadratic equation ax^2 + bx + c =0

% ---------------------------------

a = input('Enter a: ');

b = input('Enter b: ');

c = input('Enter c: ');

x1=(-b+sqrt(b^2-4\*a\*c))/(2\*a)

x2=(-b-sqrt(b^2-4\*a\*c))/(2\*a)

When “quat.m” is re-run, you may type in the values for a, b, and c.

**Enter a: 3**

**Enter b: 4**

**Enter c: 5**

x1 =

-0.6667 + 1.1055i

x2 =

-0.6667 - 1.1055i

Example 3

n = 0:pi/100:2\*pi; %create an index vector

y = cos(2\*pi\*n); %create a vector y

plot(n,y); %plot y versus n

As shown above the %-sign allows for comments. Saving the script as

foo.m it can be executed as foo from the command prompt or by clicking the run

button in the MATLAB editor.

* **Function Files**

**1.User-Defined Functions**

A user-defined function is a Matlab program that is created by the user, saved as a function file, and then can be used like a built-in function.

A function in general has input arguments (or parameters) and output variables (or parameters) that can be scalars, vectors, or matrices of any size.

There can be any number of input and output parameters, including zero.

Calculations performed inside a function

typically make use of the input parameters, and the results of the

Calculations are transferred out of the function by the output parameters.

**2. Writing a Function File**

* A function file can be written using any text editor (including the Matlab Editor).
* The file must be in the Matlab Path (current directory) in order for Matlab to be able to locate the file.
* The first executable line in a function file must be the function definition line, which must begin with the key word function and in **lower case letters**.
* The most general syntax for the function definition line is:

**function[out1,out2,...]=functionName(in1, in2,...)**

where functionName is the name of the user-defined function(also name of M-file), in1,in2,... are the input parameters, and out1,out2,...are the

output parameters.

* The parentheses are needed even if the function has no input parameters:

**function[out1,out2,...]=functionName()**

* If there is only one output parameter, then the square brackets can be omitted:

**function out=functionName (in1,in2,...)**

* If there is no output parameter at all, then they are called void functions. The function definition line is written as:

**function functionName (in1,in2,...)**

**NB:** functionName must be the same as the filename (without the ‘.m’ extension) in

which the function is written. For example, if the name of the function is projectile it must be written and saved in a file with

the name projectile.m

**Example**

to define the function

g(x)=x2-1/x2+1

you would type the following in the M-file:

**function** y=g(x)

y=(x.^2-1)./(x.^2+1);

Select Save from the File menu and save the file using the name g.m in the default Directory.

Now return to the Command Window. You can now use the function g.

>> g(-10)

ans=

0.9802

>> g(0)

ans=

-1

>>x=-10:0.1:10;

>>y=g(x);

>>plot(x,y);

Exercise

Create an m-file for each of the following functions and invoke them at the command prompt.

1. function t = thickness(load,area,cbr)

t = sqrt(load ./ (8.1 .\* cbr) + area/pi);

a)>> thickness(60000,240,10)

2. function [x] = f018(r)

% This is an example function.

% I have no idea what r or x are.

% Just take my word for it.

x = r.\*sin(r);

a)>> f018(2)

b)>> f018([1 2])

c)>> m = f018([-3 2 5])

d)>> R = f018([1.2,2.4;3.2,1.5])

3. % ------------------------------

% quatsolv.m is to compute the roots of

% quadratic equation ax^2 + bx + c =0

% --------------------------------

function [x1, x2] = quatsolv(a,b,c)

x1=(-b+sqrt(b^2-4\*a\*c))/(2\*a);

x2=(-b-sqrt(b^2-4\*a\*c))/(2\*a);

To evaluate this function, a main program is needed. This main program provides input arguments (a, b, and c). These inputs are then sent to the function file (“quatsolv.m”) to calculate the roots and then return the results back to the main program. In this example, the main function is to be called “main.m”, which looks like:

% --------------------------------

% main.m is to solve quadratic equation ax^2 + bx + c =0

% it calls the external function quatsolv.m

% ---------------------------------

a = input('Enter a: ');

b = input('Enter b: ');

c = input('Enter c: ');

[x1, x2] = quatsolv(a, b, c);

When “main.m” is invoked, it will function as the previous version of “quat.m”.

Click the run button to run the main.m or type main.m at the command prompt.

function [x,y] = quad(a,b,c)  
x = (-b + sqrt(b^2 - 4 \* a \* c))/(2\*a);  
y = (-b - sqrt(b^2 - 4 \* a \* c))/(2\*a);

If we want to compute the roots of the following expression:

2x2 + x - 1 = 0

We can call our function (first code) like this:

x = rqe(2,1,-1) and we get from Matlab:

x =0.5000   -1.0000

TRY

Use the same function to solve the following quadratic equations

1. 4x2 + 2x - 5 = 0
2. x2 + 6x + 1 = 0
3. -3x2 + 7x-8=0

4.

function p\_area = area\_calc(x,y)

x=input('ENTER X COORDINATES AS [x1 x2 x3 x4 x5 x6 x1 ]: ');

y=input('ENTER Y COORDINATES AS [y1 y2 y3 y4 y5 y6 y1 ]: ');

n=length(x)

for i=1:n-1

p\_area=(x(i) + x(i+1)) \* (y(i) - y(i+1));

end

p\_area = abs(p\_area)/2;

save file as area\_calc.m

enter area\_calc.m at the command prompt and follow what will be happinening.