Lab III of MA2213

Qian Lilong

Department of Mathematics
National University of Singapore

October 1, 2018

qian.lilong@u.nus.edu

1 m file

1.1 Script

- What it a m-file

  An m-file, or script file, is a simple text file where you can place MATLAB commands. When the file is run, MATLAB reads the commands and executes them exactly as it would if you had typed each command sequentially at the MATLAB prompt. All m-file names must end with the extension '.m' (e.g. test.m). If you create a new m-file with the same name as an existing m-file, MATLAB will choose the one which appears first in the path order (type help path in the command window for more information). To make life easier, choose a name for your m-file which doesn't already exist. To see if a filename.m already exists, type help filename at the MATLAB prompt.

- Why use m-file

  For simple problems, entering your requests at the MATLAB prompt is fast and efficient. However, as the number of commands increases or trial and error is done by changing certain variables or values, typing the commands over and over at the MATLAB command window becomes tedious. M-files will be helpful and almost necessary in these cases.

- If you are using PC or Mac:

  To create an m-file, choose New from the File menu and select Script. This procedure brings up a text editor window in which you can enter MATLAB commands.

  To save the m-file, simply go to the File menu and choose Save (remember to save it with the '.m' extension). To open an existing m-file, go to the File menu and choose Open.

  If you are using Unix:

  To create an m-file, use your favorite text editor (pico, nedit, vi, emacs, etc.) to create a file with .m extension (e.g. filename.m).

- How to run the m-file?

  After the m-file is saved with the name filename.m in the current MATLAB folder or directory, you can execute the commands in the m-file by simply typing filename at the MATLAB command window prompt.

  If you don't want to run the whole m-file, you can just copy the part of the m-file that you want to run and paste it at the MATLAB command window.

- The rule of file name

  The rules are exactly the same as for variable names: start with a letter, followed by letters or numbers or underscore, maximum 64 characters (excluding the .m extension), and must not be the same as any MATLAB reserved word.

During the process of the script, after running, the values of the variables will be stored in the memory of the system. And it can be accessed globally, which is one of the biggest difference between function file

1.2 Examples of script

Example 1. *Plot the solution of differential equation*

$$y' = y + t \tag{1}$$

with different initial value

$$y(0) = -2, -1, 0, 1, 2 \tag{2}$$

on the interval [0,2].

The general solution of this differential equation is

$$y(t) = Ce^t - t - 1, \tag{3}$$

where

$$C = y(0) + 1. \tag{4}$$

```
t = 0:0.05: 2;
C = -2 + 1; plot(t, C * exp(t) -t - 1,' -')
hold on
C = -1 + 1; plot(t, C * exp(t) - t - 1,'-.')
C =  0 + 1; plot(t, C * exp(t) - t - 1,'--')
C =  1 + 1; plot(t, C * exp(t) - t - 1,'.')
C =  2 + 1; plot(t, C * exp(t) - t - 1,':')
grid on
xlabel('t');
ylabel('y');
title('Figure 1 - -Solutions to dy/dt = y + t.');
legend('y(0) = -2',' y(0) = -1',' y(0) = 0',' y(0) = 1',' y
    (0) = 2');
hold off
```

Another solution to this problem is:

```
t = 0 : 0.05 : 2;
Y = []; % create a empty matrix, i.e., 0*0 matrix
for k = -2 : 2
C = k + 1; Y = [Y ; C * exp(t) - t - 1]; % concatenate two
    matrices, and then assign it still to Y. Similar for x =
    x+1;
end
plot(t, Y )
grid on
xlabel('t'); ylabel('y')
title('Figure 1 -- Solutions to dy/dt = y + t.')
```

In this case, we can simplify the for loop for plot command by plotting five rows simultaneously.

Moreover, we can simplify the process further.

```
t = 0 : 0.05 : 2; % a row vector for x-axis
C0 =    exp(t);  % row vector
C1 =   - t - 1; %row vector
e = ones(5,1); % column vector, all 1's
l = -1:1:3; % row vector, (-2,-1,0,1,2)
Y = l'*C0 + e*C1;
plot(t, Y );
```

```
grid on
xlabel('t'); ylabel('y')
legend('y(0) = -2',' y(0) = -1','y(0) = 0','y(0) = 1','y(0) =
    2','Location','NorthWest');
title('Figure 1 - -Solutions to dy/dt = y + t.');
```

In this case, all the operation are matrix,vector operations. We don't have loops here. In fact, the matrix computation is much faster than the loop operation in MATLAB.

Example 2. *The output of a forced undamped harmonic oscillator is given by*

$$x(t) = 2\sin(\frac{t}{2})\sin(\frac{23t}{2}). \tag{5}$$

Plot the solution over the interval [-2π, 2π]. In addition plot the envelope 2sin(t/2) with a line width of 2 pixels, and in a distinctive color.

```
t = linspace(-2 * pi, 2 * pi, 1000);
x = 2 * sin(t/2).*sin(23 * t/2);
env = 2 * sin(t/2);
plot(t, x);
hold on
h = plot(t,[env; -env]);
set(h,'linewidth',2,'color','c'); % change the property of
    the graph, h is the handle of the figure
hold off
axis([-2 * pi, 2 * pi, -2.1, 2.1]); % change  the display
    area; can be done by ylim([-2.1,2.1])
xlabel('t')
ylabel('x(t)')
title('Figure 2')
```

Example 3 (For interest). *Ways to change the line width of a graph*

- *Do it in the interface of axis properties*
- *Add the operational control "linewidth" in plot command*

```
plot(t,x,'linewidth',2); % the value 2 is the linewidth
```

- *Use graph handle*

```
h = plot(t,x);
set(h,'linewidth',2);
```

- *Use findall*

```
set(findall(gca,'type','line'),'linewidth',5); % the
    last value changes the line width
```

- *Change the handle values directly*

```
h = plot(t,x);
hold on
h(1).LineWidth  = 2; % change the linewidth of the first
     line in h
```

2 funtion file in MATLAB

To begin with, let consider the following simple example of function file.

The name of the file is “test.m” and the content is

```
function y = test(x)
 y  = x+1; % if we dont have ";" here, the value of y will
    display on the sreen
end
```

If we want to run it, just call “test(1)” (or with other input) in the command window. Different with that of script, we cannot click run bottom here to execute it because it needs an input arguments.

- The structure of function file

```
  function [y1,y2,...,yN] = myfun(x1,x2,...,xM)
        % function declaration, must be the first line
        %except the white line or comments
  %% this function is used to calculate ...
        % explanation for this function, which can be
        % used by the help command, can be removed
        % contents
        ...  % body of the functioin
        ...
    end
```

  Here [y1,y2,...,yN] are the outputs, (enclosed by square brackets with commas separating each other if more than one output variable), x1,x2,...xM are the inputs, and myfun is the function name. The contents must be placed between function declaration and “end”

- Name rule for function

  Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores. Moreover, the file name should be same with function name.

- How to use function

  Create a function, then call it by the name in the command window. If it does not have any input arguments, we can click run bottom.

- Variables are local in function

  When the function is executed, only the values of the inputs are known for this function. It w ill use the values of the inputs to calculate the following statements.

  After the execution, only the values of the outputs are stored in the system. It removed all the variables in the body of the function. It seems a black box, once you have a inputs, you will get a outputs. We will not show the details of inner components for you. This is different from a script file.

2.1 Examples of function

Example 4. *Create a function to calculate the value of*

$$y = x^2 - 1. \tag{6}$$

Create a file named "f.m", the content is:

```
function y = f(x)
%%   this is the function to calculate
%          y = x^2-1
%   input: x, double
  y = x^2; % main statement
end
```

In the command window, call

```
>> f(3)
  ans =
     8
```

Type

```
help f
```

you will get the information you typed after function declaration.

2.2 Two functions in a single file

We can put two functions in a single m file.

One is the main function, i.e., the first , the other is called the subfunction, which can only be accessed by the main function.

Example 5. *We define a function "fun2" to obtain the largest value of two variables. And define a function "fun1" to obtain the largest value of three variables.*

Here we create a m file "fun1.m". Note that the file name should match the name of the first function.

```
  function y = fun1(a,b,c)
     d = fun2(a,b); % the largest one of a,b
     y = fun2(d,c); % the largest value of d,c
  end

  function y = fun2(a,b)
     if a>b
        y = a;
     else
        y = b;
     end
  end
```

The fun1 will call the subfunction fun2 during the process. Similarly, we can have multiple functions in a single m-file. But only the first one is the main function, that has the same name as file name. And we can only call this function.

All the rest sub functions are only accessed by the main function. Any order of them has the same effect, just follows the first one.

3 function in a script file

We can also put a function in a script file.

Example 6. *An object thrown in the air obeys the initial value problem*

$$\frac{d^2y}{dt^2} = -9.8 - \frac{dy}{dt} \tag{7}$$

with

$$y(0) = 0, y'(0) = 120.$$

We find the solution,

$$y = -\frac{49}{5}t + \frac{649}{5}(1 - e^{-t}), \tag{8}$$

where y is the height in meters of the object above ground level after t seconds. Estimate the height of the ball after 5 seconds.

Here we create a m file named "findheight.m" (or other name, the rule of script file name is not strict as that of functions)

```
close all
t = linspace(0,15,200); % t is vector, from 0 to 200
y = height(t); % calculate the height at each point, call
    this function, height, at this line
plot(t,y);
grid on
xlabel('time in seconds');
ylabel('height in meters');
title('Figure 3')
function y = height(t)
   y = -(49/5)* t + (649/5) *(1 - exp(-t)); % subfunction to
    calculate the height for given t
end
```