What is a Function M-File?

Table of Contents

Suppose you wished to write your own function which you could access every time you loaded up Matlab. Suppose it couldn't be as simple as we've seen before - perhaps it needs to test the values passed to it like a piecewise function in calculus. Suppose the function is a process that needs to calculate and return a value. How can we do this?

Matlab allows all of this very conveniently using function M-files. At its most basic level an M-file is a text file containing a series of Matlab commands. Typically these files have the extension .m which is where they get their name. The location of these files is important also since otherwise Matlab won't know where to find them. The best and perhaps most obvious place to put them is in the current working directory. This is the directory indicted in the leftmost directory window "Current Directory" on Matlab. If you wish to store your M-files in another directory you'll have to add that directory to the path. You can do this by hunting under the "File" pull-down menu.

A Simple Function M-File

Let's start by creating a function M-file which you'll then have in your directory for future use. We'll make it something useful from calculus. We know that an object dropped from a height of `h` meters will strike the ground in `sqrt(h/4.9)` seconds. Let's create a function M-file named HowLong. To do this type:

```
edit HowLong.m
```

Matlab will verify that you'd like to open a new file (in the current working directory) so click "Yes".

In the editor window which opens type the following code:

```
function t=HowLong(h)
  t=sqrt(h/4.9);
end
```

Note: The `end` at the end of the m-file is not critical but it is good programming practice and I will always do it. I recommend you do too. This way the function has an `end` and each `for`, `while` and `if` has an end.

Save the M-file. From now on we can access our function simply by calling the HowLong function. You can do this anytime you ever run Matlab (now or in the future) because the file (and hence the function) will be there until you delete it.

```
HowLong(100)
```

```
ans =

   4.517539514526256
```

Stop to anaylze for a minute how this function works. The keyword `function` tells Matlab that we're writing a function. The `t=` indicates the variable which we'll use to return the result. The name `HowLong` is the name of the function and the parenthetical `h` indicates that a single value will be passed to the function and in that function the value will be called `h`.

We can also do things like:

```
syms x;
HowLong(x)
```

```
        ans =

        ((10*x)/49)^(1/2)
```

or:

```
syms q;
HowLong(q)
```

```
        ans =

        ((10*q)/49)^(1/2)
```

The General Idea

The general outline of a function M-file is as follows:

```
function r=functionname(a,b,...)
  do stuff
  any matlab stuff
  by the end assign
  r = whatever you want to return
end
```

You can do any Matlab commands you like in the middle provided that you assign your return value `r` before you end the function.

Note that the use of `r` is arbitrary, you can use any variable you wish, but it must be the variable you assign to return!

The `functionname` must correspond to the filename `functionname.m` you use. Here are some other simple function m-files and what they do:

Example 1: This will accept one value `a` and return that value plus `3`.

```
function bah = justin1(a)
  bah = a+3;
end
```

Example 2: This will accept two values `a` and `b`, solves `a*x+b`, printing the result since the line does not end with a semicolon `;`, and then assigns (and therefore returns) `s=a-b`.

```
function happy = justin2(a,b)
  syms x;
  solve(a*x+b);
  happy = a-b;
end
```

Example 3: This does `ezplot` on `a*x^2` and returns `a-8`. It completely ignores `b`. In practice you'd never ignore a parameter (otherwise what would be the point of having it?) but this just shows that it will work.

```
function y = justin3(a,b)
  syms x;
  ezplot(a*x^2);
  y = a-8;
end
```

A (More) Complicated Function M-File

Function M-files do not need to be so simple. We can define a piecewise function. Try this new M-file:

```
edit pwisef.m
```

In the function place the following:

```
function y=pwisef(x)
  if (x <= 0)
    y = -x;
  else
    y = sin(x);
  end
end
```

Note: The first `end` ends the `if/else` and the second `end` ends the function. The result is as expected:

```
pwisef(-1)
```

```
ans =

     1
```

versus

```
pwisef(pi/4)
```

```
ans =

   0.707106781186547
```

But if you try the following you'll get an error:

```
pwisef(x)
```

```
Conversion to logical from sym is not possible.

Error in pwisef (line 2)
```

```
  if (x <= 0)

Error in Ch16_FunctionMFilesOne (line 158)
pwisef(x)
```

Why is this? The answer is fairly straightforward. The `pwisef` takes the value `x` as a parameter and then hits a brick wall when it tries to execute `if (x<=0)`. How can it compare `x` to `0` if it doesn't know `x`? It can't.

Published with MATLAB® 8.0