What Can We Do with a Function M-File?

Table of Contents

Multiple Parameters

Function M-files can take more than a single variable as an argument. For example suppose we wish to write a function which calculates the area of a rectangle with length `l` and width `w`. We can do this with the following code placed in a function M-file called `rectarea.m`:

```
function a=rectarea(l,w)
  a=l*w;
end
```

Easy! Then we call the function as expected:

```
rectarea(15,72)
```

```
ans =

        1080
```

Passing Functions to Functions

The last thing we want to look at is how to pass a function to a function M-file as one of its arguments. For example suppose you wished to write a function M-file which took two arguments, one being a function and the other a constant, and then it plugged the constant into the function and returned the result. Nothing sophisticated!

We do this in a fairly obvious way. The thing to keep in mind is that Matlab can accept functions as parameters in various ways. Here are a couple. First create the M-file `plugin`:

```
function r=plugin(f,x)
  r=f(x);
end
```

This is it. Now we can pass any function as the first argument `f` provided that `f(x)` makes sense. For example:

```
f = @(x) x^2-3;
plugin(f,-1)
```

```
        ans =

            -2
```

or without a declaration of the function first:

```
plugin(@(x) x^2-3,-1)
```

```
        ans =

            -2
```

A More Complicated Example

Here's a function M-file which takes a function handle, a value and a tolerance. It approximates the derivative of the function at that value by using closer and closer approximations by tangent lines until successive approximations differ by less than the tolerance.

```
function r = derivativeapprox(f,a,tol)
  h = 1;
  oldapprox = (f(a+h)-f(a))/h;
  h = h/2;
  approx = (f(a+h)-f(a))/h;
  while (abs(approx-oldapprox)>=tol)
    oldapprox = approx;
    h=h/2;
    approx = (f(a+h)-f(a))/h;
  end
  r = approx;
end
```

Here is some sample output:

```
derivativeapprox(@(x) x^3,2,0.01)
```

```
        ans =

           12.0059
```

Take some time to see how this function works since you'll need to write similar examples yourself. We're approximating the derivative by looing at `(f(a+h)-f(a))/h` as `h` gets closer to zero (remember that definition of the derivative?) So we start with `h=1`. We find an initial approximation called `approx` using that `h`. Since we're going to be comparing successive approximations we need to preseve the old approximation each time so we create a variable `oldapprox`. At the start there's no old approximation so we give it a value which *guarantees* that we'll get into the `while` loop successfully the very first time. The `while` loop first saves the approximation as the old approximation, changes the value of `h` by cutting it in half, then finds the new approximation. Thus after each iteration of the `while` loop we have the (new) approximation and the old approximation to compare. Note that the `abs` is critical. Do you know why?

Documenting Your M-File

To close this section we'll put in a vote for a good programming practice - documentation! At the Matlab prompt we can always type `help name` to get information on command like `help ezplot`. Now that we've learned to write function M-files we should put some documentation in those files so that we can get help on them. Modify your `rectarea.m` file to read:

```
function a=rectarea(l,w)<br>
% This function rectarea(l,w) finds the area of a rectangle with
% length l and width w.
  a=l*w
end
```

Then try:

```
help rectarea
```

```
  This function rectarea(l,w) finds the area of a rectangle with
  length l and width w.
```

The text which is after the % symbols in the M-file will be printed out as a response to `help rectarea` being called.

Published with MATLAB® 8.0