

## In-Class Exercise

### 1. Function m-files:

- (a) Create a new m-file. (File >> New >> M-file). This opens the editor.
- (b) Type the following exactly as is. Including the comments.

```
function [x,y] = test(a,b)
% function [x,y] = test(a,b)
% a and b are inputs
% x returns the sum a + b
% y returns the product a*b
```

```
x = a + b;
y = a*b;
```

- (c) Save the file somewhere (don't use the default, but create a new directory). Notice that the file name is (should be) the same as the function name.
- (d) Now, return to the command window (the main MATLAB window) and type

```
[sum, product] = test(5,8)
```

One of two things will happen. It might give you the sum and product. More likely, you'll get a message '??? Undefined function or variable 'test'.'

- (e) If you get the error message, it means that whatever directory you saved 'test.m' in is not currently part of the MATLAB search path, and is not your present working directory either. To fix this, from the command window go File >> Set Path. This will open up the Path Browser window. From there, select Path >> Add to Path. Enter the path or browse to it. OK it, close it, and it'll ask you if you want to save for future sessions. Tell it yes.
- (f) Assuming you got the error message, you'll want to run the command again. Hit the 'up arrow' key. It'll cycle through the previous commands and save you some retyping on occasion. Enter. You should get an answer.
- (g) Now type

```
help test
```

It'll return the first few lines of comments. If you forget the syntax (say, what order the input and output variables go in), you can look it up without having to open the file and check. Don't omit those lines of comments!

- (h) Note: The function variables x,y,a,and b are dummy variables. When you're calling the function, you can pass the function variables named 'sum' 'product' 'dog' and 'cat' for all it cares.

### 2. Functions and functions

MATLAB doesn't distinguish between a 'function' in the math sense (you pass it some numbers; it returns the result of a formula like  $(x + y)^2$ ), and a 'function' in the abstract sense of some routine (which may return several values or no values). MATLAB does not require type declarations for functions, and as you've already seen, it can return a several variables at once. You can call a function from a function, and you can also pass the

*name* of a function to another function, and have it evaluate the function you've specified. This feature is very convenient; for example, if 'functionA' made a function call to another function named 'f', we'd be forced to name our second function 'f' every time we wanted to use 'functionA'. However, if 'functionA' allows us to pass it the name of our function, and uses the command 'feval' to evaluate that function, we can name our second function whatever we like. To see this in action, create the following 3 functions:

```
function b = final_f(x,y,name)
% function b = final_f(x,y,name)
% x will be used by this function
% y will be passed to some other function
% name will be the name of that other function
b = x + feval(name,y);
```

```
function m = first_f(x)
% your descriptive comments here
m = x.^2;
```

```
function m = second_f(x)
% your descriptive comments here
m = x - 1;
```

The 'mixing up' of the x's and y's is deliberate; just to remind you that the name of the variable as you pass it along doesn't have to match its name in the function. On the command line, enter:

```
q = final_f(2,3,'first_f')
```

It should pass the 2, the 3, and the name (note the quotes) along to 'final\_f'. From 'final\_f', it'll send the 3 down to 'first\_f' for evaluation. The result should be  $2 + 3^2$ . Now try:

```
q = final_f(2,3,'second_f')
```

This time, the 3 will get sent down to 'second\_f' for evaluation. The result should be  $2 + (3 - 1)$ .

### 3. Scripts and workspaces

The only major disadvantage of functions is you can't access the contents of any of their internal variables by their names inside the function. After running the above examples, you could enter 'm' or 'b' on the command line, and you'd get an error message. Those variables are invisible to MATLAB. So, suppose you're working on some project, and you enter the following lines of code (hitting 'enter' after each line, of course):

```
a = 1:.01:2;
b = 2:.01:3;
x1 = a.^2 -2*a;
x2 = 3*b + 7;
plot(a,x1,'k')
```

```
hold on
plot(b,x2,'k')
y = [x1, x2];
```

...and you run out of time for the day. You want to get back to it later. You could turn it into a function, but then you'd either have to pass in a,b,x1,x2, etc. as parameters, or type it as is, but have no access to a,b etc. outside the function. You want to save it as a **script** instead. Scripts are created just like functions (File >> New >> M-file), but there's no 'function[x,y] = f(a,b)' part. Just type (or copy and paste) the commands directly into the file and save it. To run the script, type the name of the script. It'll execute the commands exactly as typed, AND you have access to the variables.