Functions & Programs in MATLAB

There are four ways of doing functions/programs in MATLAB . One can directly enter code in a terminal window. This amounts to using MATLAB as a kind of calculator, and it is good for simple, low-level work. The second method is to create a *script* m_file. Here, one makes a file with the same code one would enter in a terminal window. When the file is "run", the script is carried out. The third method is the *function* m_file. This method actually creates a function, with inputs and outputs.

1 Terminal Input & Anonymous Functions

One can type the code in the MATLAB terminal window. For example, if we wish to plot $x\sin(3x^2)e^{-x^2/4}$ in the range $[-\pi,\pi]$, we could type the following in the terminal window.

```
x=(-pi):pi/40:pi;                      ENTER
y=x.*sin(3*x.^2).*exp(-x.^2/4);        ENTER
plot(x,y)                              ENTER
```

The code listed above creates the row vector `x`, and then uses it to form a row vector `y` whose entries are the values of the function $x\sin(3x^2)e^{-x^2/4}$ for each entry in `x`. The operations preceded by a "dot," such as .* or .^ are array operations. They allow entry-by entry multiplications or powers. Without the "dot," these are matrix operations. After creating the arrays x and y, an x-y plot is made.

This method of doing calculations is good for short, one-time-only calculations or for calculations where one does not wish to change any parameters. A shortcoming of this method is that the code for the function only produces a row vector y containing numerical values. To compute a y for a different x requires running the same script, but with x changed. A way that avoids this shortcoming is discussed next.

Anonymous Functions There is another important way to represent a function in MATLAB . Frequently, we want to use simple functions, such as the one we've been working with, in a single session. While an m_file can be used to do this, it adds clutter to the collection of m_files. To do this, MATLAB provides what it calls an *anonymous function.* Here is an example of how it works.

```
fnct=@(x) x.*sin(3*x.^2).*exp(-x.^2/4);
ENTER
```

The creates fnct, which can be used in a session that same way as sin, say. To find the value of the function at $x = 2.3$, use type in fnct(2.3). To plot it against x=(-pi):pi/40:pi; use the command plot(x,fnct(x)).

The name used doesn't have to be fnct. It can be pretty much anything that doesn't conflict with other MATLAB function names, g for example. Moreover, using a different independent variable produces exactly the same function. Thus the code below will will give exactly the same behavior as that above.

```
fnct=@(t) t.*sin(3*t.^2).*exp(-t.^2/4);
ENTER
```

In using fnct, even if it's defined with t instead of x, the command plot(x,fnct(x)) produces exactly the same plot.

Anonymous functions can have more than one variable. Here is an example using three variables.

```
h=@(x,y,z) x.*sin(3*x.^2.*y.*z).*exp(-y.^2/4);
ENTER
```

Again, to find the value of the function h at (1,2,3), you would use h(1,2,3). The result is h(1,2,3) $= -0.2763$.

2 Script M-Files

If we wish to execute repeatedly some set of commands, and possibly change input parameters as well, then one should create a script m_file. Such a file always has a ".m" extension, and consists of the same commands one would use as input to a terminal. For example, to do the plot in section 1, one would create the file my_plot.m:

```
x=(-pi):pi/40:pi;
y=x.*sin(3*x.^2).*exp(-x.^2/4);
plot(x,y)
```

To execute the "script", one would use this command:

```
my_plot                                  ENTER
```

Script M-files are ideal for repeating a calculation, but with some parameters changed. They are also useful for doing demonstrations. As an exercise, create and execute the script file sc_plot.m:

```
m=menu('Pick a plot','Sine plot','Cosine plot');
if m==1,
x=(-pi):pi/40:pi;
y=sin(x);
title('Sine')
else
x=(-pi):pi/40:pi;
y=cos(x);
title('Cosine')
end
plot(x,y)
```

As before, to execute the script file, type

```
sc_plot                                  ENTER
```

3 Function M-Files

Most of the M-files that one ultimately uses will be *function* M-files. These files again have the ".m" extension, but they are used in a different way then scripts. Function files have input and output arguments, and behave like FORTRAN subroutines or C-functions. The structure of a typical function file, say my_fun.m, is as follows:

```
function outputs=my_fun(inputs)
code
⋮
code
outputs=···
```

Note that the word *function* appears at the start of the file, and in lower case letters. In addition, the outputs and inputs and name of the function are listed. Let us return to the plot done in section 1. Suppose that instead of giving the vector `x`, we want to make it a variable. At the same time, we want to have available the data that we plotted. Here is a function file that would do this.

```
function y=my_fun(x)
y=x.*sin(3*x.^2).*exp(-x.^2/4);
plot(x,y)
```

Function files are normally used to combine functions in MATLAB to get new functions. For example, suppose that we want to have at our disposal a function that computes the inverse of the square of a matrix, and returns an error message if the matrix is close to singular or singular. Call this file `inv_sq.m`.

One more thing. In the code below, note that A^2 is used, not A.^2. This is because we are computing the *square* of a matrix, not a matrix with the *entries* of A squared.

```
function B=inv_sq(A)
if abs(det(A))< 0.0001,
error('The matrix is singular')
else
B=inv(A^2);
end
```