**Executable Files**

In this tutorial we will assume that you know how to create vectors and matrices, know how to index into them, and know about loops. For more information on those topics see one of our tutorials on [vectors](file:///A:\NHampshire\vector.html), [matrices](file:///A:\NHampshire\matrix.html), [vector operations](file:///A:\NHampshire\operations.html), [loops](file:///A:\NHampshire\control.html), or [plotting](file:///A:\NHampshire\plot.html).

In this tutorial we will introduce the basic operations for creating executable files. Once you have a general routine in a matlab file, it allows you to perform more complex operations, and it is easier to repeat these operations. For example, you might have a set of instructions to use Euler's approximation for a differential equation (see the tutorial on [loops](file:///A:\NHampshire\control.html)), but you want to be able to use those instructions for different equations.

As an example, a simple file to approximate the D.E. y'= 1/y using Euler's method is found. To execute the commands in the file, the step size and the initial value must be specified. Once done, you can easily approximate the given D.E. for a wide variey of initial conditions and step sizes.

First, you will need to create the file. The easiest editor on our system (SGI's IRIX) is called "jot". It will allow you to do some very simple file manipulations. The editor is very simple and easy to start. It is not a very advanced editor, though.

Matlab executable files (called M-files) must have the extension ".m". In this example a file called simpleEuler.m is created. To get Matlab to execute the commands in the file simply type in "simpleEuler". Matlab will then search the current directory for the file "simpleEuler", read the file, and execute the commands in the file.

If you are not familiar with a more advanced editor use jot to create the file. Type in the following command at the unix prompt:   
% jot simpleEuler.m

Once the editor appears on the screen either type or cut and paste the necessary matlab commands:

% file: simpleEuler.m

% This matlab file will find the approximation to

%

% dy/dx = 1/y

% y(0) = starty

%

%

% To run this file you will first need to specify

% the step the following:

% h : the step size

% starty : the initial value

%

% The routine will generate three vectors. The first

% vector is x which is the grid points starting at

% x0=0 and have a step size h.

%

% The second vector is an approximation to the specified

% D.E.

%

% The third vector is the true solution to the D.E.

%

% If you haven't guessed, you cna use the percent sign

% to add comments.

%

x = [0:h:1];

y = 0\*x;

y(1) = starty;

for i=2:max(size(y)),

y(i) = y(i-1) + h/y(i-1);

end

true = sqrt(2\*x+1);

Once the commands are in place, save the file. Go back to your original window and start up matlab. The file is called up by simply typing in the base name (in this case simpleEuler).

>> simpleEuler

??? Undefined function or variable h.

Error in ==> /home/black/math/mat/examples/simpleEuler.m

On line 27 ==> x = [0:h:1];

If you try to call the file without first defining the variables h and starty, you will get an error message. You must first specify all of the variables that are not defined in the file itself.

>> h = 1/16;

>> starty = 1;

>> starty = 1;

>> simpleEuler

>> whos

Name Size Elements Bytes Density Complex

h 1 by 1 1 8 Full No

i 1 by 1 1 8 Full No

starty 1 by 1 1 8 Full No

true 1 by 17 17 136 Full No

x 1 by 17 17 136 Full No

y 1 by 17 17 136 Full No

Grand total is 54 elements using 432 bytes

>> plot(x,y,'rx',x,true)

Once the necessary variables are defined, and you type in the command *simpleEuler*, matlab searched the current directory for a file called simpleEuler.m. Once it found the file, it read the file and executed the commands as if you had typed them from the keyboard.

If you would like to run the program again with a different step size, you have to be careful. The program will write over the vectors x,y, and true. If you want to save these vectors, you must do so explicitly!

>> x0 = x;

>> y0 = y;

>> true0 = true;

>> h = h/2;

>> simpleEuler

>> whos

Name Size Elements Bytes Density Complex

h 1 by 1 1 8 Full No

i 1 by 1 1 8 Full No

starty 1 by 1 1 8 Full No

true 1 by 33 33 264 Full No

true0 1 by 17 17 136 Full No

x 1 by 33 33 264 Full No

x0 1 by 17 17 136 Full No

y 1 by 33 33 264 Full No

y0 1 by 17 17 136 Full No

Grand total is 153 elements using 1224 bytes

>> plot(x0,abs(true0-y0),'gx',x,abs(true-y),'yo');

Now you have two approximations. The first is with a step size of 1/16, and it is stored in the vectors *x0* and *y0*. The second approximation is for a step size of 1/32 and is found in the vectors *x* and *y*.

The [previous tutorial](file:///A:\NHampshire\plot.html) is an introduction to plotting.   
The [next tutorial](file:///A:\NHampshire\subroutine.html) is an introduction to subroutines.

This tutorial written by Kelly Black.