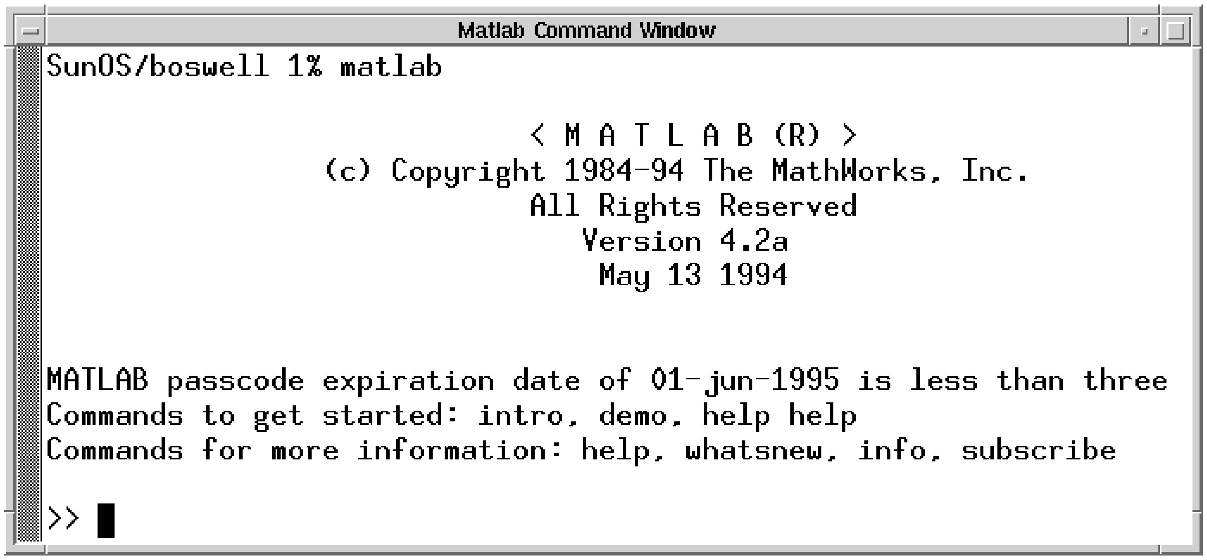
MATLAB TRAINING SESSION I

WRITING M-FILES: SCRIPTS and FUNCTIONS



```
SunOS/boswell 1% matlab

                        < M A T L A B (R) >
              (c) Copyright 1984-94 The MathWorks, Inc.
                        All Rights Reserved
                           Version 4.2a
                            May 13 1994


MATLAB passcode expiration date of 01-jun-1995 is less than three
Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info, subscribe

>> 
```

When starting MATLAB for interactive use you will be prompted for input by the command line interpreter (>>). Single-line commands are processed immediately and the results are displayed. MATLAB can also execute sequences of commands that are stored in files.

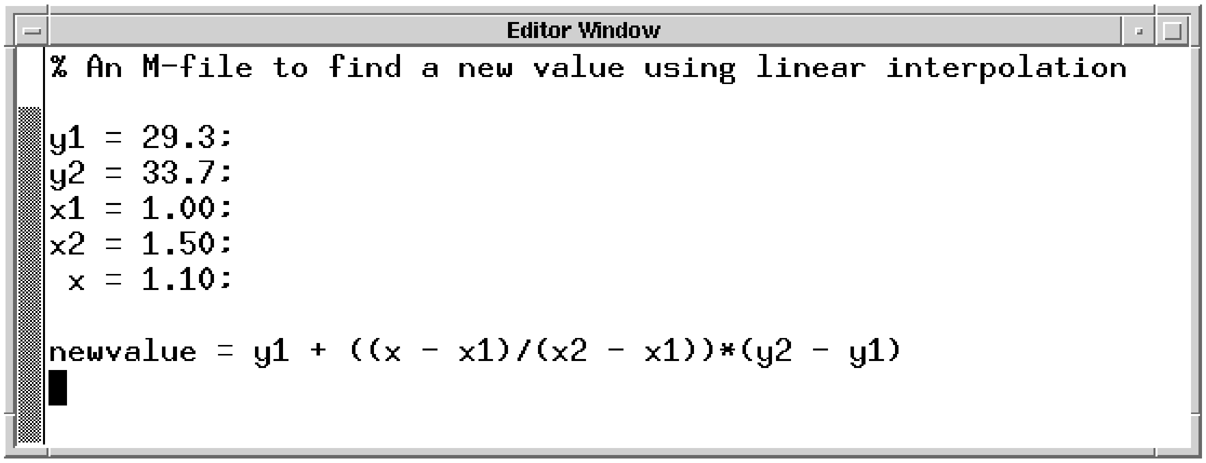
Disk files that contain MATLAB statements are called M-files because the files must have a ".m" extension in their filename. An M-file consists of normal MATLAB statements, which possibly include references to other M-files. Much of your work with MATLAB will be in creating and refining M-files. M-files are created using a local text editor or word processor (must save/export as an ASCII file).

There are two basic types of M-files: *script files* and *function files.*

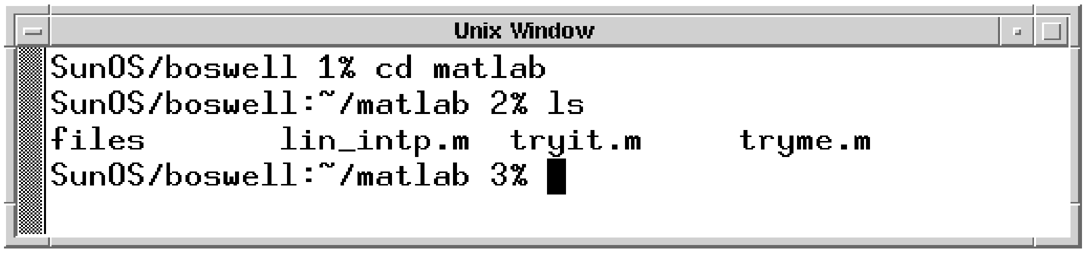
Script files

A script file consists of a sequence of normal MATLAB statements. When the name of the file is typed at the MATLAB prompt, MATLAB will cause the statements in the script to be executed. ***The variables within the script are global and will change the value of the variables of the same name in the environment of the current MATLAB session.*** Scripts are useful for performing analyses, solving problems, or designing long sequences of commands that become monotonous to repeat.

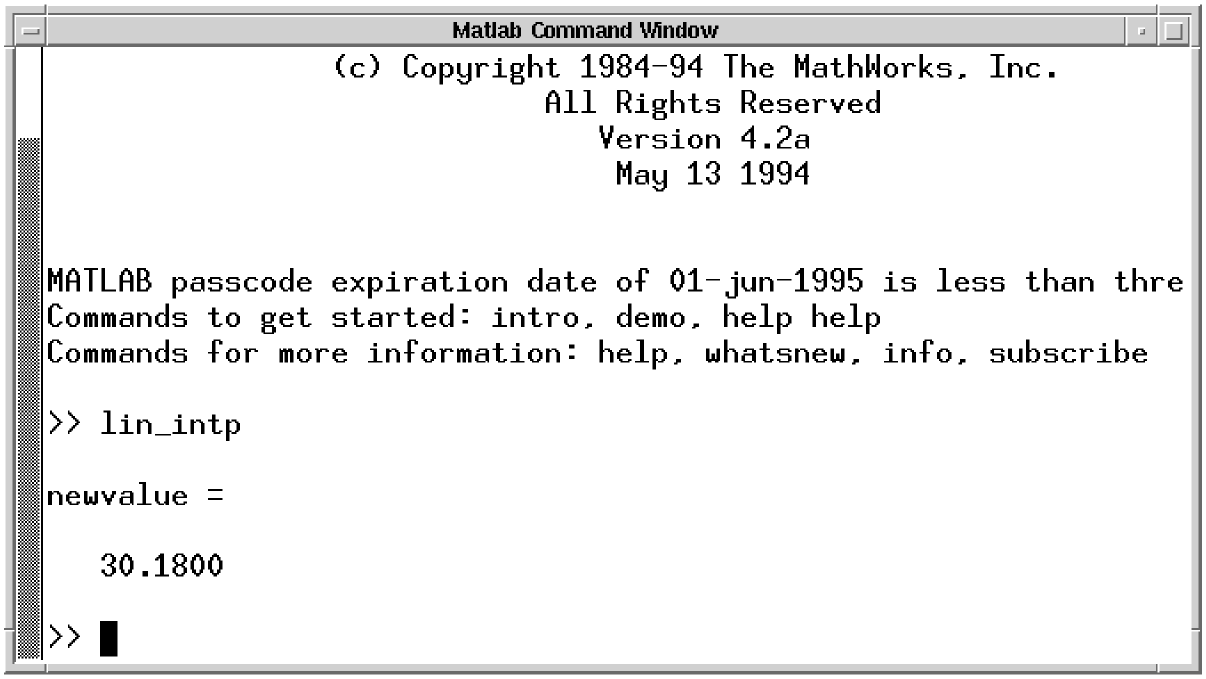
As an example, lets say that as an engineer I find myself needing to do linear interpolation from data in a table. To prevent typing the commands every time, I create a file called `lin_intp.m` that contains the commands (as seen in the following window using the vi editor).


Editor Window
```
% An M-file to find a new value using linear interpolation

y1 = 29.3;
y2 = 33.7;
x1 = 1.00;
x2 = 1.50;
 x = 1.10;

newvalue = y1 + ((x - x1)/(x2 - x1))*(y2 - y1)
```

Because I have created the file with a text editor (not MATLAB), the file lin_intp.m exists in my directory independent of MATLAB as shown below.


Unix Window
```
SunOS/boswell 1% cd matlab
SunOS/boswell:~/matlab 2% ls
files        lin_intp.m  tryit.m     tryme.m
SunOS/boswell:~/matlab 3%
```

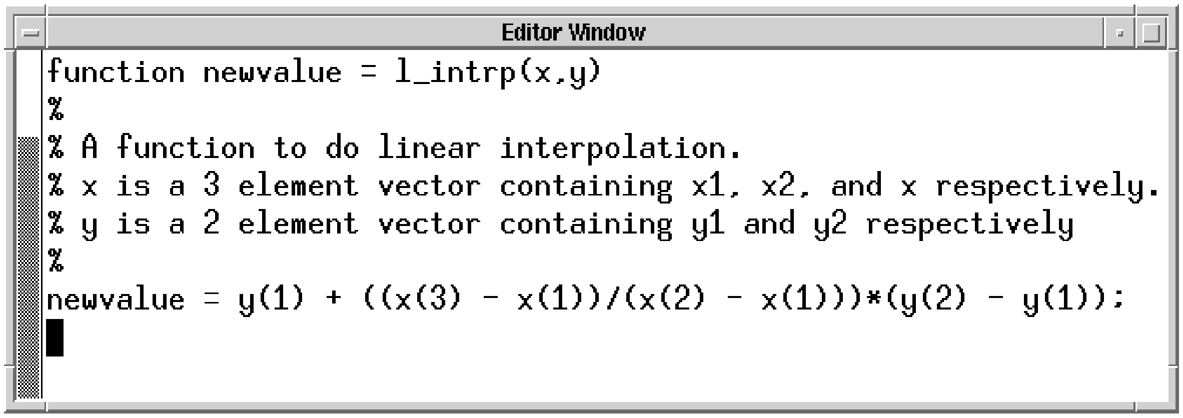
Entering the statement `lin_intp` at the interpreter prompt causes MATLAB to execute these commands and the *newvalue* result will be displayed as shown below. This example does not show much time savings over entering the commands at the prompt but does demonstrate a script file. The disadvantage here is that I would have to edit the file to change the values of y1, y2, x1, x2, and x for each new situation.


Matlab Command Window
```
                (c) Copyright 1984-94 The MathWorks, Inc.
                            All Rights Reserved
                               Version 4.2a
                                May 13 1994


MATLAB passcode expiration date of 01-jun-1995 is less than thre
Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info, subscribe

>> lin_intp

newvalue =

   30.1800

>>
```

Function files

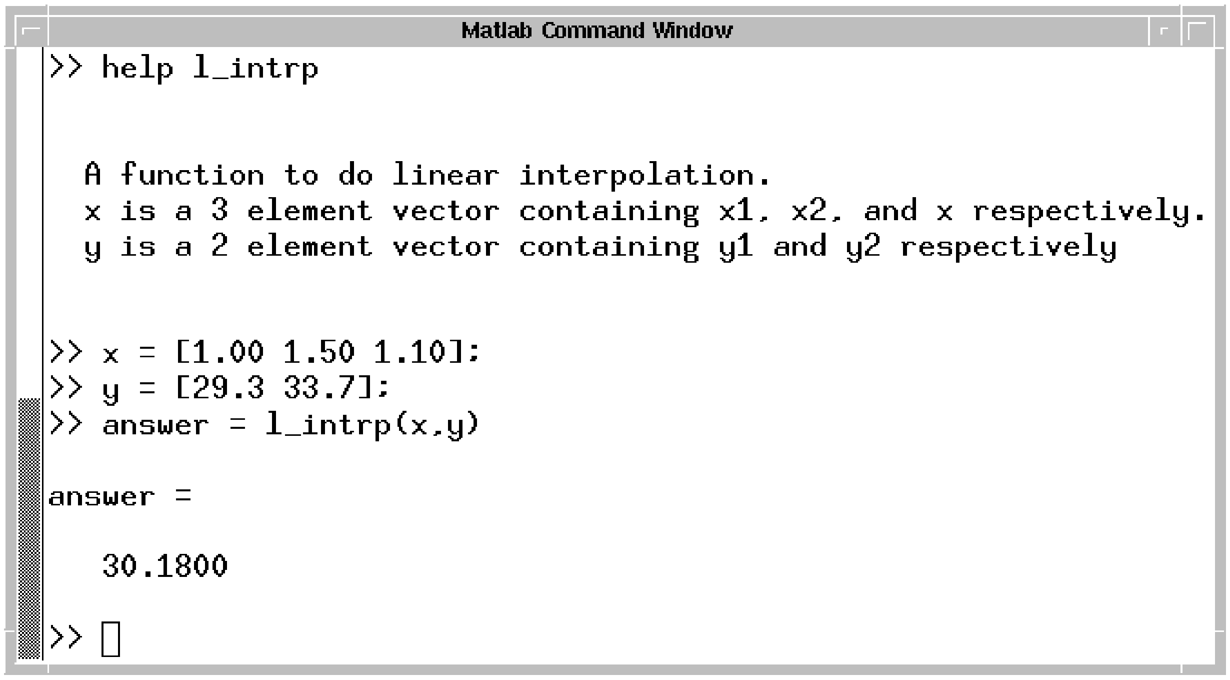
Function files provide extensibility to MATLAB, that is, create MATLAB functions using the MATLAB language itself. You can create new functions specific to your problem which will have the same status as other MATLAB functions. ***A function differs from a script in that arguments may be passed, and variables defined and manipulated inside the file are local to the function and do not operate globally on the workspace. Also, it is required that the name of the m-file have the same name as the function.***

The problem we have been working on can be converted into a function as shown below in the file called `l_intrp.m`.



```
function newvalue = l_intrp(x,y)
%
% A function to do linear interpolation.
% x is a 3 element vector containing x1, x2, and x respectively.
% y is a 2 element vector containing y1 and y2 respectively
%
newvalue = y(1) + ((x(3) - x(1))/(x(2) - x(1)))*(y(2) - y(1));
```

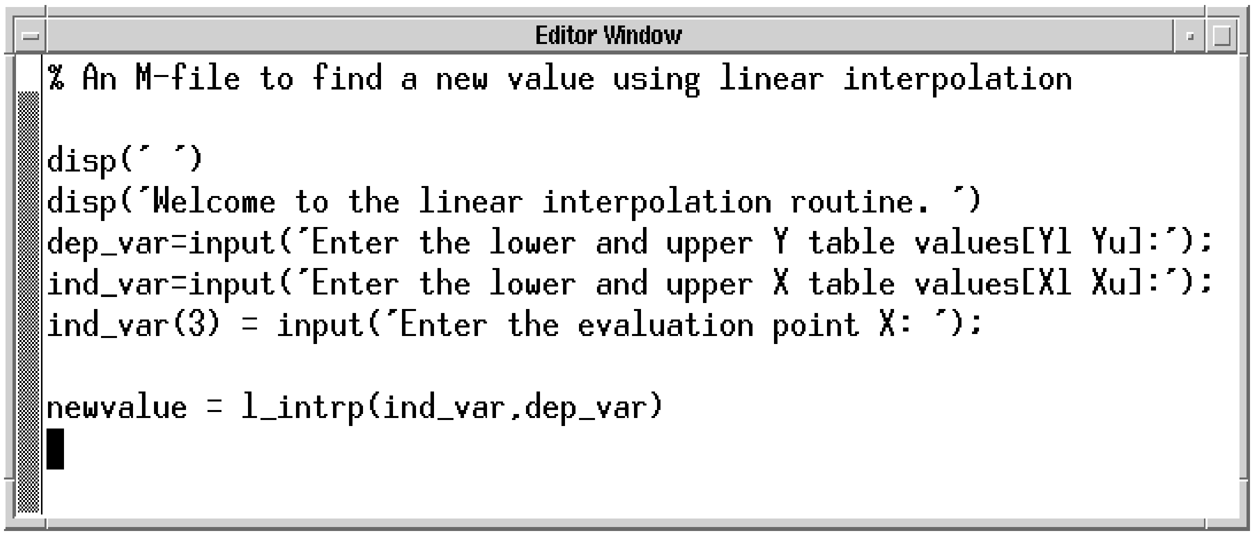
The addition of the comment lines (%) will allow MATLAB return help text when I use the help command as in `>>help l_intrp` (see result below).



```
>> help l_intrp


  A function to do linear interpolation.
  x is a 3 element vector containing x1, x2, and x respectively.
  y is a 2 element vector containing y1 and y2 respectively


>> x = [1.00 1.50 1.10];
>> y = [29.3 33.7];
>> answer = l_intrp(x,y)

answer =

   30.1800

>> 
```

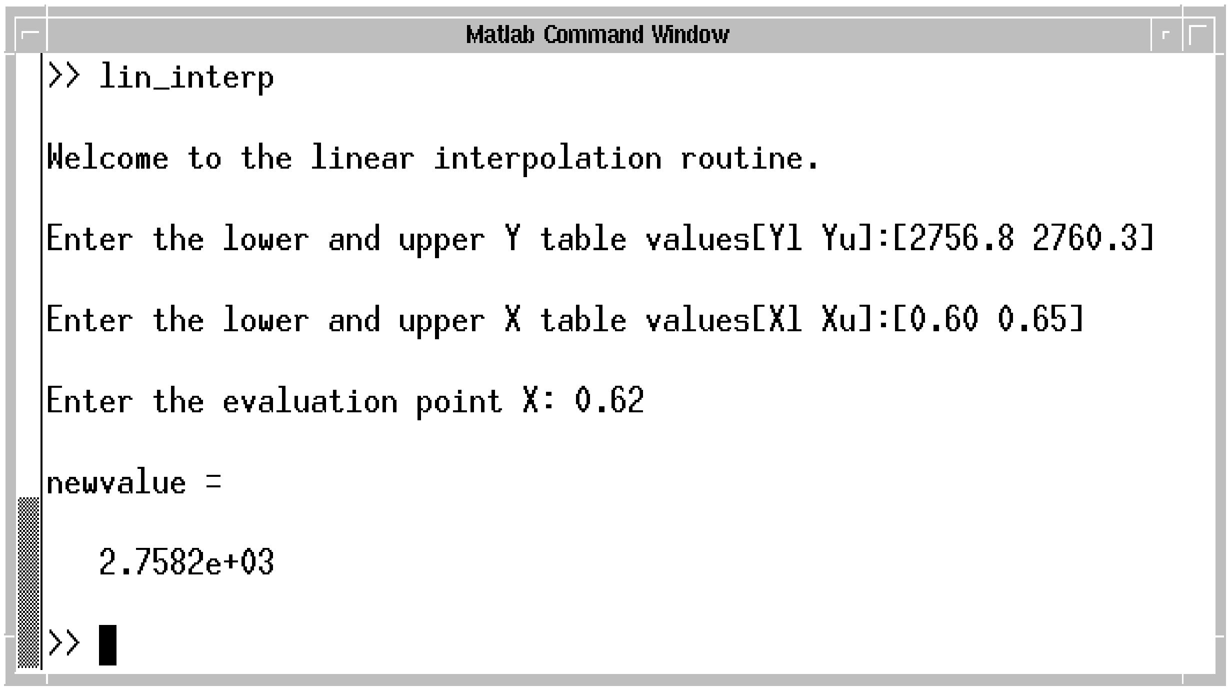
Notice that the result of the function is assigned to the variable Answer in the workspace which is not the same variable created in the function (newvalue). ***Only the number held by return variable in the function statement is returned to the workspace.*** The variable names inside the function and hence the numbers they hold, are not retained by the workspace. Notice that even though the function created the temporary variable “newvalue”, the workspace does not keep a copy.

Now I will write a script that is much more helpful. The file is called `lin_interp.m.` The script will use the function `l_intrp` I created earlier



```
% An M-file to find a new value using linear interpolation

disp(' ')
disp('Welcome to the linear interpolation routine. ')
dep_var=input('Enter the lower and upper Y table values[Yl Yu]:');
ind_var=input('Enter the lower and upper X table values[Xl Xu]:');
ind_var(3) = input('Enter the evaluation point X: ');

newvalue = l_intrp(ind_var,dep_var)
```

As an example we are asked to find the specific enthalpy of saturated water vapor at 0.62 MPa. The saturated steam: pressure table has entries for 0.60 and 0.65 MPa which gives h_g as 2756.8 and 2760.3 kJ/kg respectively. To find the answer we must find a better table or use linear interpolation. In MATLAB I call my script and find h_g @0.62MPa to be 2758.2 kJ/kg.



```
>> lin_interp

Welcome to the linear interpolation routine.

Enter the lower and upper Y table values[Yl Yu]:[2756.8 2760.3]

Enter the lower and upper X table values[Xl Xu]:[0.60 0.65]

Enter the evaluation point X: 0.62

newvalue =

   2.7582e+03

>> 
```

An important point to note, if you input the name of something to MATLAB, for example `lin_interp`, the MATLAB interpreter goes through the following steps:

1. Looks for `lin_interp` as a variable.
2. Checks for `lin_interp` as a built-in function.
3. Looks in the current directory for a file named `lin_interp.m`.
4. Looks in the directory specified by MATLAB's search path for a file named `lin_interp.m`.
   *Additional paths can be automatically appended to the built in path in an m-file called `startup.m.` Check the help documentation on startup

Background Execution

If you have a set of MATLAB computations that takes a long time to run you might consider running the code in the background even when you are not logged in.
The system command:

```
matlab < infile > outfile &
```

will run the MATLAB commands in the text file *`infile`* and create an text diary of the MATLAB session in the file *`outfile`*. If you want your computations saved so that you can load them back into MATLAB after the job is finished you can include `save` *`filename`* as the last command in *`infile`* or add `save` *`variablenamelist filename`* to save only selected calculations. `>` *`outfile`* can be left out if a diary of the session is not needed.

If there are graphics commands in *`infile`*, such as plots, and you do not want to remove them, use the system commands;

```
set $OLDDISPLAY=DISPLAY
unsetenv DISPLAY
nohup matlab < infile > outfile &
setenv DISPLAY $OLDDISPLAY
```

to prevent the background job from hanging due to display environment problems, specially if you will be logging out before the computation is done.

Making a Diary

You can also have the MATLAB command window text saved to a file as it appears on the monitor by using the command **diary *filename***. Everything seen from this point until you enter the command **diary off** will go into an ascii text file of the given name in your directory. This file can then be edited or sent to the printer using the **lp** command from a ***unix* prompt**.