Introduction to Matlab

Graeme Chandler

Mathematics Department
The University of Queensland

February 2000

Contents

1. Introduction.

1.1. Learning Matlab

Matlab is one of the fastest and most enjoyable ways to solve problems numerically. The computational problems arising in most undergraduate courses can be solved much more quickly with Matlab, than with the standard programming languages (Fortran, C, Java, etc.). It is particularly easy to generate some results, draw graphs to look at the interesting features, and then explore the problem further. By minimizing human time, Matlab is particularly useful in the initial investigation of real problems; even though they may eventually have to be solved using more computationally efficient ways on super computers.

This introduction gives a quick way to become familiar with the most important parts of Matlab. The first five sections emphasize simple arithmetic, matrix-vector operations (including solving systems of equations), and graphing functions and data. The later sections describe some more advanced features, including 3D graphics. There are also some suggestions about using Matlab to do larger projects, and including Matlab results and graphs in reports.

The best way to used this introduction is to sit down at a computer and type in the commands as they are described. Look at Matlab's response, and check that the answers are what you expect. It is also a good idea to do the small exercises. It makes sure that the commands become part of an active Matlab vocabulary. Each lesson should take less than one hour. More information about any Matlab command can be found by using the on line help features described in lesson 3.

These notes assume basic familiarity with the Windows interface.. For instance, you need to know about

- cutting and pasting within and between windows,
- editing files, and
- moving between windows and resizing them.

If you are unsure about this, seek help from another student or tutor.

1.2. Further References

The complete Matlab Manuals are available on line in the Mathematics department's PC laboratories. They contain an introductory guide for new users, and an especially good introduction to Matlab's graphics.

More information on Matlab can be found in the books

1. D.M. Etter; *Engineering Problem Solving with Matlab*, Prentice–Hall, 1993.
2. Duane C. Hanselman & Bruce Littlefield*; Mastering MATLAB 5: A Comprehensive Tutorial and Reference,* Prentice Hall, 1998
3. Kermit Sigmon, *A Matlab Primer,* 4th Ed. 1995

or try the Mathworks web site

http://www.mathworks.com

If you wish to use Matlab at home, the *student edition* of Matlab is available from bookshops at the cost of about $100.00 (or you can borrow a copy from the library.) The student version is 'crippled' and only works for small problems. Nevertheless, it should suffice for almost all assignments in undergraduate courses. It is usually possible to develop a program on the student version and do the final full size problem on the Mathematics Department's computers if necessary.

1.3. Starting Matlab

Here we describe the steps needed to start Matlab on the PC's in the Mathematics Computer Laboratories (Rooms 519 and 420). On other machines the start-up procedure will be different. However the Matlab commands are the same on all machines.

1. First sit down at a PC. If necessary, close any programs left running by a previous user. The simplest way is to use the Windows 'Start' button then the 'Shut Down' option.
2. When prompted, use the [Ctrl]-[Alt]-[Del] combination to bring up the logon screen. (You will be asked for a name and password, but ignore this and just press the [Enter] key.)
3. You should now see the standard Windows desktop with a screen of icons.
4. Find the icon labelled 'Matlab' and double click on it. After another pause, the Matlab logo appears briefly, then the 'MATLAB Command Window' remains on the screen. It ends with the words:-

```
To get started, type one of these commands:
          helpwin, helpdesk, or demo.
For information on all of the MathWorks products, type tour.
```

```
>>
```

Matlab is now ready!

1.4. Typing Commands

All commands to Matlab are typed in after the Matlab prompt, i.e. the symbol >> . They are then sent to Matlab to be implemented by pressing the [Enter] key.

- Any typing error can be corrected before the [Enter] key is pressed.
  - Use the keypad left and right arrows or the mouse to move to the error.
  - Use the [Del] key to delete the mistakes, and then type in the correction.
  - When the line is correct, just press [Enter] to send the command to Matlab. (It is not necessary to go to the end of the line.)

Usually errors are not noticed until Matlab beeps and displays an error message. However *it is not necessary to retype the whole command.*

- Previous commands can be corrected and reused to save typing.
  - Just press the keypad up arrow and the previous command appears.
  - Make the necessary corrections and press [Enter] to run the corrected command.

Now go on to the first lesson. Type in all the commands as they are shown, and make sure the Matlab response is what you expect.

Exercise .1. *Type in the Matlab command* `logo` . *A nice 3D graph of the Matlab logo should appear. This is produced by advanced use of Matlab's graphics capabilities, discussed later.*

2. Simple Calculations

2.1. Basic Arithmetic.

The simplest way to start with Matlab is to use it for simple calculations. Matlab has a wide range of functions and is able to use complex numbers as well as reals. The following simple commands have obvious meanings. Type them in and see what happens. (In the examples below, all the text after the `%` is just a comment or an explanation. You do not have to type it in. It would just be ignored by Matlab.)

```
                % The words after '%' are comments
                % and explanations.  Do not type them in.

(-1+2+3)*5 - 2/3        % The four arithmetic operations
2^3                     % Means 2 to the power 3
exp( sin( pi/2 ) )        % The usual functions are provided
                          %  log, log10, cos, tan, asin, ...
```

Like a calculator, Matlab does all calculations correct only to about 16 significant decimal digits. This is enough accuracy for most purposes. For convenience, usually only the first 5 significant digits are displayed on the screen. Some more examples.

```
pi
22/7                  % pi is not the same as 22/7!
11*(15/11) - 15       % This shows there is roundoff error
                      %   when Matlab uses fractions.
cos( pi/3 )           % These are familiar trig functions.
sin( pi/6 )           %   Note Matlab always uses radians.
```

Matlab also uses variables to store intermediate answers. A variable can have almost any name, but it must begin with a letter. Matlab distinguishes between upper and lower case letters.

```
x = 2+3
y = 4+5
result1 = x/y
```

Sometimes the values of intermediate calculations are not needed and we do not want them to be displayed. If a semicolon (`;`) is placed after a command, the results are not displayed.

```
p = 2+3 ;                    % The semicolons supress
q = 3+5 ;                    %   unwanted output.
ratio = p/q
```

A number of commands can be placed on the one line, provided they are separated by a comma (`,`) or a semicolon (`;`). (Use a semicolon, unless you want the answer displayed.)

```
p = 2+3 ; q2 = x + 4 , ratio2 = p/q2       % All on one line
```

Parentheses can be used to make expressions clearer. Thus the last calculation could have been written as

```
ratio  =  (2+3)/(x+4)
```

Exercise .2. *In this exercise, save retyping by using the up arrow to edit the previous commands! Remember that multiplication is done with a '$*$', it is not enough to put numbers next to each other.*

1. *Use Matlab to evaluate* $\log(s^2 - 2\,s\,\cos(\pi/5) + 1)$ *where* $s = .5$.
2. *Now use Matlab to evaluate* $\log(s^2 - 2\,s\,\cos(\pi/5) + 1)$ *where* $s = .95$.
3. *Finally, evaluate* $\log(s^2 - 2\,s\,\sin(\pi/5) + 1)$ *when* $s = 1$.

2.2. Complex Numbers.

Matlab handles complex numbers as easily as real numbers. When you start Matlab the variable i stands for $\sqrt{(-1)}$.

```
x  =  2 + 3*i  ,  y  = 1 - 1*i
z1 = x - y ,  z2 = x * y ,  z3 = x / y
abs( x )
angle( y )
                        % Matlab quickly calculates the
sin(x)                  %    sin  of a complex number.
```

Matlab's facility with complex numbers is handy, as using complex numbers often involves complicated arithmetic. Indeed, as the previous example shows, Matlab will effortlessly work out functions of complex numbers that are difficult to do from first principles. But be careful about the name used for $\sqrt{(-1)}$. When Matlab starts, the variable `i` contains $\sqrt{(-1)}$. Often the user overrides this and uses `i` to stand for another number (in coding $\sum_{i=1}^{n}$ for example). In this case the Matlab command `i = (-1)^.5` should be used to reset `i` to $\sqrt{(-1)}$ before it is used to represent complex numbers.

```
i = 2 ; j = 3 ; i+j    % i is used in another calculation.
z = 2 + 3*i            % Complex arithmetic no langer works!.
i = (-1)^.5            % reset i
z = 2 + 3*i            % Complex numbers can now be used.
```

Here are two more complex calculations. Matlab can be used to demonstrate one of the most important relations in Mathematics: that $e^{\pi i} = -1$. (At least to within round off error!).

```
exp( pi*i ) + 1
i^i
```

Exercise .3.

(a.) The last command showed that according to Matlab, i^i is a real number. Can you explain mathematically why this is this so? (Hint: Use $i = e^{\pi i/2}$.)

(b.) If $y = i^i$, what are i^y and y^i? Can you work this out without Matlab?

3. Help in Matlab

There are three ways to get help in Matlab.

3.1. The Help Command

This is the easiest way to find out more about specific Matlab commands. If you have forgotten small details, for example. The command `help <name>` gives information about the Matlab command <name>.

```
help sin     % Information about sin.
help i       % Information about  i .
help log     % This is enough information about log
             %     to show  log  means log to the base  e .
why          %  Provides general answers.
```

By itself, the command `help` gives a list of topics in Matlab. Initially, many of the topics will be mysterious. The most familiar topic from the list is probably 'elementary functions'. To explore this topic use the command `help elfun`. In the most recent versions of Matlab the `lookfor` command can also be used to search for relevant information.

```
help           %  Gives a list topics.
help elfun     %  More help in the area 'elfun', i.e.
               %    elementary functions like sin, exp,..
```

```
help sign      %  Information about a new command
               %     from the list in 'elfun'
lookfor logarithm  %  Gives the name of some
                   %     functions related to logarithms
lookfor legend     %  Finds former students
```

The `lookfor` command is quite useful; however it may be very slow in the Mathematics PC lab. This is due to the heavy network traffic. But `lookfor` should be quite fast on a single user PC.

3.2. The Help Window

It is also possible to get help by clicking on the 'Help' menu above the command window. From the 'Help' menu, select the 'Help Window' item (by clicking) and the list of help topics is displayed. The advantage of this method is that it is possible to navigate around various topics by double clicking on them. For example, in the help window double click on elfun to find the same information given by the `help elfun` command. This is a good way to explore Matlab's commands.

3.3. The Help Desk

Matlab also comes with help documentation that can be read by Web browsers. This may be used by

- typing the command `helpdesk` in the command window; or
- using the 'HELP' menu and selecting the item 'HELP DESK (HTML)'.

After a slight pause the initial documentation window should come up. Interesting links to pursue are:-

- *Getting Started in Matlab* for introductory information
- *Documentation Road Map*for the complete set of manuals online, and
- the 'search' box to look for more commands.

Exercise .4.

1. *Is the inverse sine function one of Matlab's elementary functions? (The inverse sine is also known as* $\sin^{-1}$*, arcsin, or asin.)*

    1. *How do you find* $\sin^{-1}(.5)$ *in Matlab? Use help to find out.*

2. *If $x = .5$, is $\sin(\sin^{-1}(x)) - x$ exactly zero in Matlab?*
3. *If $\theta = \pi/3$, is $\sin^{-1}(\sin(\theta)) - \theta$ exactly zeros in Matlab? What about $\theta = 5\pi/11$?*

2. *Does Matlab have a specialized Mathematical function to calculate the greatest common divisor? (Use* `help` *or* `lookfor`*)*

    1. *Use Matlab to find the greatest common divisor of 30 and 24. (Check the answer the same as the answer calculated manually.)*
    2. *What is the greatest common divisor of 3072 and 288?*

3. *Does Matlab have a function to convert numbers to base 16, i.e. to hexadecimal form? (Hint: Use* `lookfor` *to find a way to convert a decimal number to hexadecimal.) What is 61453 in base 16? Computers almost always represent numbers internally as hexadecimals.*

4. *Use the search box in the Help Desk and look for information about logarithms. There are 9 entries, including the last command,* `logm`*, for calculating the log of a matrix!*

To wind down, type the Matlab command `demo`. This brings a menu of demonstrations and examples to explore.

1. Visit the *Gallery* to see a number of attractive images.
2. Visit the *Games* section. Some find the game bubble wrap to be extraordinarily relaxing.

4. Simple Plots of Functions

The ezplot commands (`ezplot`, `ezsurf`, `ezmesh`, ...) and `fplot` allow the user to plot functions in 2 and 3 dimensions by simply typing in the formula. Matlab allows the user much more control over colour, style, lighting, and surface texture. These more complicated commands are introduced later.

4.1. 2D Plots

The ez commands are best shown by example. Suppose we wish to plot the function $\sin(x)/(1 + x^2)$. This may be done simply using

```
ezplot( 'sin(x)/(1+x^2)' )
```

If we want to change the range to $[0, 5]$, add a second argument to our command.

```
ezplot( 'sin(x)/(1+x^2)', [0,5] )
```

Occasionally a function is given *implicitly*. For example, the circle of radius 2 is defined by the equation $x^2 + y^2 - 4 = 0$. This is plotted by

```
ezplot( 'x^2+y^2-4' )
```

A better graph is produced when the range is restricted to $[-2.5, 2.5] \times [-2.5, 2.5]$, that is with the command

```
ezplot('x^2+y^2-4', [-2.5,2.5],[-2.5,2.5])
```

Alternatively we may remember the of radius 2 circle is defined *parametrically* as the set of points $\{(2\cos(t),\, 2\sin(t)) \,:\, 0 \le t \le 2\pi\}$. Using this form the circle is plotted as

```
ezplot( '2*cos(t)', '2*sin(t)' )
```

Exercise .5. *1. A nice example of a function that would be difficult to graph without a computer or calculator is*

$$f(x) = x^{(x^x)} - (x^x)^x.$$

Plot $f(x)$. (Hint:- By itself, `ezplot` *uses the range $-2\pi \le x \le 2\pi$. This is inappropriate here: a better choice is to try the range $0 \le x \le 2$.)*

2. A Lissajous curve is a curve of the form

$$x = \sin(nt + c), \quad y = \sin(t) \tag{4.1}$$

Play with small integer values of $n = 1, 2, \ldots$ and then find values of n and c so the Lissajous curve becomes the ABC symbol.

3. ***And*** *finally graph the implicitly defined function*

$$(y^2 - x^2)(x - 1)(2x - 3) = 4(x^2 + y^2 - 2x)^2.$$

The web site

http://mathworld.wolfram.com/topics/GeneralPlaneCurves.html

has a rich collection of interesting curves to draw.

Many interesting curves are expressed simply in polar coordinates. As we follow the curve the radius is given as a simple function of the angle at the origin. That is, the curve is obtained parametrically as the set of points

$$(x, y) = (r(\theta)\cos(\theta),\, r(\theta)\sin(\theta))$$

for some simple function $r(\theta)$. For example, the cochleoid (meaning 'snail like') uses the formula $r(\theta) = \sin(\theta)/\theta$. The resulting curve is plotted in Matlab as

```
ezpolar( 'sin(t)/t' ), [-6*pi,6*pi] )
```

Of course we could get the same curve by writing out the x and y coordinates explicitly. The cochleoid could have been graphed by

```
ezplot('(sin(t)/t)*cos(t)','(sin(t)/t)*sin(t)',[-6*pi,6*pi])
```

The command `fplot` can also plot functions quickly. For example

```
fplot('sin',[0,pi])       % Plot  sin(x)  on [0,pi]
fplot('x-x^3/6',[0,2])    % Plot a polynomial.
```

Although parametric curves or polar coordinates may not be used, `fplot` has the advantage that two or more functions may appear on the same plot

```
fplot('[ cos(x), 1-x^2/2, 1-x^2/2+x^4/24]', [-pi,pi])
```

Exercise .6. *1. From the help for the* `fplot` *command, find out how to change the range of the y axis on the previous plot. For example, a range of* $-1.1 \le y \le 1.1$ *looks better.*

2. Use `fplot` *to do the following.*

   1. *Plot the function* e^x *between -1 and 1.*
   2. *On the same graph plot* e^x *and the polynomial* $1 + x$ *between -1 and 1.*
   3. *Finally on the same graph plot* e^x, $1+x$, *and* $1+x+x^2/2+x^3/6$ *between -1 and 1. Why are these last two polynomials good approximations to* e^x*?*

Now add a title to the graph. Finally find out how to use the command `legend` *to label the three curves.*

4.2. 3D Plots.

Curves may exist three dimensions as well as two. If we fly around in a circle of radius 1 and gain 1 unit of height each circuit we would trace out the curve

$$(x, y, z) = (\sin(2\pi t), \cos(2\pi t), t) \qquad t \geq 0.$$

To show the flight path up to the height $z = 4$, use the command

```
ezplot3('cos(2*pi*t)','sin(2*pi*t)','t',[0,4])
```

Exercise .7. *1. Use* `help ezplot3` *to findout how to animate this curve.*

2. What is the angle the flight path makes with the horizontal? (This is a thinking question; there is no simple Matlab command to help!)

3. How would we plot the path if we gained 2 units of height for each circuit?

Plotting surfaces in 3 dimensions is more interesting. To plot the surface $z = f(x, y)$ where

$$f(x, y) = xy \exp(-(x^2 + y^2))$$

use the Matlab command

```
ezsurf( 'x*y*exp(-(x^2+y^2))' )
```

Some folk prefer to do away with the ugly black lines on the surface. It is also useful to add a color bar, so that it is possible to determine the height of the surface from its colour. To make these modifications, add two extra commands after the `ezsurf` command.

```
ezsurf( 'x*y*exp(-(x^2+y^2))' ) % Plot the surface
shading interp                  % Remove the black lines
colorbar                        % Add a colorbar
```

To see a 3D surface clearly it is necessary to look at it from different angles. Otherwise key features may be hidden. We can change our viewpoint by clicking on the surface with the mouse (a blue outline of a box should appear). Then, without releasing the mouse button, drag the blue outline until the required view has been obtained. Release the mouse button and the surface will be redrawn from the new view. Be warned though, it is easy to get lost.

Exercise .8. *Plot the function* $xy\exp(-(x^2+y^2))$ *using the following commands instead of* `easysurf`.
`ezsurfc ezmesh ezmeshc ezcontour ezcontourf`
(The syntax of these new commands is the same as `ezsurf`*. But a complete description of the new commands is found using help commands;* `help ezcontour, help ezmeshc`*, ...). Which command and which viewpoint would be best to determine the height of the peaks?*

Exercise .9. *(For students of multivariate calculus only.) Produce a nice graph which demonstrates as clearly as possible the behaviour of the function*

$$xy^2/(x^2+y^4)$$

near the point $(x,y)=(0,0)$*. This is surprisingly difficult. Remember, by default, Matlab plots surfaces on a* 60×60 *grid, and many of the finer features of a function are lost. If your computer has enough memory it is possible to increase the number of grid points for a smoother graph.*

Remember a one dimensional curve in 2D may be described parametrically in the form $(x,y)=(x(s),y(s))$; that is its two coordinates are a function of the one variable s. Surfaces may also be described parametrically in the form

$$(x,y,z)=(x(s,t),y(s,t),z(s,t))$$

The three coordinates are functions of 2 variables. For example a spiral ramp is described by

$$\{(s\cos(t),s\sin(t),t)\ :\ .4\le s\le 1, 0\le t\le 6\pi\}$$

This is graphed by

```
ezsurf( 's*cos(t)', 's*sin(t)', 't',[.4,1],[0,6*pi] )
```

Exercise .10. *Use the floor command to plot a spiral staircase*
`ezsurf('s*cos(t)','s*sin(t)','floor(t)',[.4,1],[0,6*pi])`

Exercise .11. *Explain how the Lissajous curve in the previous section, is related to the curve* $(x,y,z)=(\cos(nt),\sin(nt),\sin(t))$*. (Hint: Look at the 3D curve from different angles.)*

Exercise .12. *Use the* `help surf` *command and look at the very last example. (It is spread over two lines). Cut and paste these two lines from the help text to the current command line (i.e. after the last >>) . Now press return and look at*

the graph. (You could of course type in the whole command; but a cut and paste is more reliable.)

This formula is very complicated to understand; but the graph is interesting as a work of Matlab art. You may need to rotate a little to see what is happening at the origin. I prefer to use the more aesthetic variation. (Note the `...` *mean the command is continued on the next line. The* `...` *would be left out if the command we typed on one long line.)*

```
ezsurf(  '(1-s)*(3+cos(t))*cos(4*pi*s)', ...
          '(1-s)*(3+cos(t))*sin(4*pi*s)', ...
              '3*s + (1 - s)*sin(t)', ...
          [0,2*pi/3,0,12],120  ),
shading interp
```

Exercise .13. *Plot a sphere. (Hint: Use a parametric 3D plot. At height t the sphere is a circle of radius ... ?)*

Exercise .14. *Plot a swirl. This is a surface whose height above the point $(x, y) = (r\cos\theta, r\sin\theta)$ is $\sin(6\cos(r) - n\theta)$.*

5. Plotting Lines and Data

This section shows how to produce simple plots of lines and data.

Suppose we wish to plot some points. For example we are given the following table of experimental results.

x_k	.5	.7	.9	1.3	1.7	1.8
y_k	.1	.2	.75	1.5	2.1	2.4

To work with the data in Matlab set up two column vectors `x` and `y`.

```
x = [.5 .7 .9 1.3 1.7 1.8 ]'
y = [.1 .2 .75 1.5 2.1 2.4 ]'
```

(Vectors are discussed in detail in the next lesson; but we can use them to draw graphs without knowing all the details.) To graph y against x use the plot command.

```
plot(x,y,'x')
```

The graph should now appear. (If not, it may be hidden behind other windows. Click on the icon 'Figure No. 1' on the Windows task bar to bring the graph to the front.)

This graph marks the points with an 'x'. Other types of points can be used by changing the `'x'` in the plot command. (Use `help plot` to find out the details.) From the graph it is clear that the data is approximately linear, whereas this is not so obvious just from the numbers. Good graphs quickly show what is going on!

When you have finished looking at the graph, just click on any visible part of the command window, or on the Matlab icon on the task bar. More commands can then be typed in.

Exercise .15. *1. Plot the above data with the points shown as circles.*

2. Plot the above data with the points joined by lines (use `help plot`*).*

3. Plot the points as green stars.

4. Find out how to add a title to the plot. (Hint: Use the command `help plot`*. At the end of the help output there is a list of related commands. Find out about one of these with another use of the help command.)*

5.1. Adding a Line

From the graph it is clear that the points almost lie on a straight line. Perhaps the points are off the line because of experimental errors. A course in statistics will show how to calculate a 'line of best fit' for the data. But even without statistics, the line between the points $(.5, 0)$ and $(2, 3)$ is a good candidate for 'a line fitting the data'. Lets add this line to the plot and see how well it approximates the data. We do this by asking Matlab to plot the points $(.5, 0)$ and $(2, 3)$ joined by a line.

```
x_vals = [ .5 2]'        %  The X-coords of the endpoints.
y_vals = [ 0 3 ]'        %  The Y-coordinates
plot( x,y,'x', x_vals,y_vals,'-' )
legend('Original Data', 'Line of Best Fit')
title( 'Data Analysis  (Graph by Effigy Mounds)')
```

Note that `x_vals` contains the x–coordinates, `y_vals` contains the y–coordinates, and the two points are joined by a line because the final argument in the plot command is `'-'`.

file=twoplots.eps,height=8.7491cm,width=11.0578cm

Figure 5.1: Good and Bad Data Plots

5.2. Hints for Good Graphs

There are many opinions on what makes a good graph. From the scientific viewpoint a simple test is to see whether we can recover the original data easily from the graph. A graph is supposed to add something, not remove information. For example if our data runs from 1 to 1.5 it is a bad idea to use an axis running from 0 to 10, as we will not be able to see the differences between the values. We give two more subtle recommendations..

5.2.1. Plot data as points

Often people use the very simplest command to plot data and automatically type just

```
plot( x, y )          % The simplest plot command
```

This simple command does not present the data here very well. It is hard to see how many points were in the original data. It is really better to plot just the points, as the lines between points have no significance; they just help us follow the set of measurements if there are several data sets on the one graph. If we insist on joining the points it is important to mark the individual points as well

Exercise .16. *Change the above plot command to show the data more clearly by plotting the data points as small circles joined by dotted lines. (Hint: There should be a third argument in the plot command containing the symbols for a line of dots and the marker type circles; and perhaps a color symbol as well!)*

5.2.2. Choose a good scale

In the example in this section, it was easy to see the relationship between x and y from the simple plot of x against y. In more complicated situations, it may be necessary to use different scales to show the data more clearly. Consider the following model results

n	3	5	9	17	33	65
s_n	.257	.0646	.0151	3.96×10^{-3}	9.78×10^{-4}	2.45×10^{-4}

.

file=loglog.eps,height=8.9644cm,width=11.0556cm

Figure 5.2: Good Scaling for Plots

A plot of n against s_n directly shows no obvious pattern. (Note the dots, '. . .', in the next example mean the current line continues on the next line. Do not type these dots if you want to put the whole command on the one line.

```
n = [ 3 5 9 17 33 65 ]';
s = [ 2.57e-1  6.46e-2  1.51e-2 ...
             3.96e-3 9.78e-4  2.45e-4 ]' ;

    plot( n, s, 'x' )          % This is a poor plot!!
```

In fact it is hard to read the values of s_n back from the graph. However a plot of $\log(n)$ against $\log(s_n)$ is much clearer. To produce a plot on a log scale we can use either of the commands

```
plot( log10(n), log10(s), 'x')   % Two good plots
loglog( n, s, 'x')               % using log scales!
```

It reveals there is almost a linear relationship between the logs of these two quantities.

Exercise .17. *1. Add a title to the last plot above and label the X and Y axes.*

2. *Using* `help loglog` *find a command which will use a log scale only for the s_n data. Is this better or worse then the* `loglog` *plot?*

3. *(A hard exercise.) After allowing for some experimental error, there is a simple relationship between n and s_n. Can you find a mathematical formula for this?*

4. *What would be the value of s_n when $n = 129$?*

6. Matrices and Vectors

Although Matlab is a useful calculator, its main power is that it gives a simple way of working with matrices and vectors. Indeed we have already seen how vectors are used in graphs.

Remember to keep typing in the commands as they appear here, and observe and understand the Matlab response. If you mistype, it is easy to correct using the arrows and the [Del] key. Try to use the help facility to find out about unfamiliar commands. Otherwise ask another student or a tutor.

6.1. Solving Equations.

Most people will have seen systems of equations from school. For example, we may need to find x_1, x_2, and x_3 so that

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 1 \\ -2x_1 - 6x_2 + 4x_3 &= -2 \\ -x_1 - 3x_2 + 3x_3 &= 1 \end{aligned}$$

Although these problems can be solved manually by eliminating unknowns, this is unpleasant. Besides errors usually occur.

In first year Mathematics the problem is rewritten in matrix-vector notation. We introduce a matrix A and a vector b by

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -2 & -6 & 4 \\ -1 & -3 & 3 \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}.$$

Now we want to find the solution vector $x = [x_1, x_2, x_3]$ so that

$$Ax = b.$$

In spite of the new notation, it is still just as unpleasant to find the solution. In Matlab, we can set up the equations and find the solution x using simple commands.

```
                                        % Set up a system
A = [ 1 2 -1; -2 -6 4 ; -1 -3 3 ]       % of equations.
b = [ 1; -2; 1 ]
x = A\b                                 % Find x with A x = b.
```

Matlab should give the solution

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}.$$

A sound idea from manual computations is to substitute the computed solution back into the system, and make sure all equations are indeed satisfied. In Matlab we can do this by checking that the matrix vector product Ax gives the vector b, or better still, we check that $Ax - b$ is exactly the zero vector.

```
A*x          % Check that  Ax  and  b  are the same.
b
A*x - b ;    % So that we do not miss small differences,
             %  it is better to check that  A x - b = 0 .
```

The vector $Ax - b$ is known as the *residual*.

Exercise .18.
(a.) Change the middle element of A from -6 to 5 (i.e. $a_{22} = 5$.) What is the solution to this new system? What is the new residual? Why is the residual not exactly 0? (Hint: In Matlab the number 1.23×10^{-5} is displayed as `1.23e-005`*. The column vector $[1.23 \times 10^{-5}; 4.44 \times 10^{-6}]$ could be displayed as*

```
1.0e-004 *

                    1.2300e-005
    0.1234    or    4.4400e-006
    0.0444
```

The user can control which form is used by the File→Preferences→ Numeric Format *item on the Matlab Menu bar.)*
(b.) Suppose the middle element is changed from -6 to -5. Can Matlab solve this third system? Explain what has happened.

6.2. Matrices and Vectors.

Much scientific computation involves solving very large systems of equations with many millions of unknowns. This section gives practice with the commands that are needed to work larger matrices.

The following code sets up a 4×4 matrix, A, and then finds some of its important properties. When `[ ]` is used to set up matrices, either blanks or commas (`,`) can be used to separate entries in a row. Semicolons (`;`) are used to begin a new row.

```
A =  [1 2 3 4 ; 1 4 9 16 ; 1 8 27 64 ; 1 16 81 256 ]
A'
det(A)
eig(A)
inv(A)
```

Even in the simple 4×4 case, it is tedious to evaluate determinants and inverses. But in Matlab they can be quickly calculated and used.

Indices can be used to show parts of a larger matrix. For example try

```
A(2,3),  A(1:2,2:4),   A(:,2),  A(3,:)
```

In general `A( i:j, k:l )` means the square sub–block of `A` between rows i to j and columns k to l . The ranges can be replaced by just '`:`' if all rows or columns are to be included.

Exercise .19.
(a.) How would you display the bottom left 2×3 *corner of A? How would you find the determinant of the upper left* 3×3 *block of A?*
(b.) Find the Matlab command to calculate the row echelon form of a matrix. The row echelon form is never useful in practice, but it is handy for checking homework in other subjects!!

6.3. Creating Matrices

Matlab also provides quick ways to create special matrices and vectors.

```
c  =  ones(4,3)
d  =  zeros(20,1)
I  =  eye(5)
D  =  diag( [2 1 0 -1 -2] )
L  =  diag( [1 2 3 4], -1 )
R  =  randn(5,5)              % R is a  5 x 5 matrix
                              %     of random numbers.
```

(More information on each of these commands can be found with `help`.)

Matrix-matrix and matrix-vector multiplication work as expected, provided the dimensions agree. Matrices and vectors can both be multiplied by scalars. In the next example, `B` is another 4×4 matrix and `c` is a column vector of length 4 (i.e. a 4×1 matrix).

```
B = [1 1 0 0
     0 2 1 0
     0 0 3 1
     0 0 0 4 ]        % Rows can be separated by
                      % making a new line as well as using ';'.
c = [1; 0; 0; -1]
5*B                   % Multiply scalars and matrices.
B*c                   % Multiply matrices and vectors.
A*B                   % Matrix by matrix multiplication.
B*A                   % Note: AB is not the same as BA !
```

Some of the following commands do not work. There is no harm in trying them.

```
c*c        % Wrong dimensions for matrix multiplication.
c*A        % Wrong dimensions for matrix multiplication.
c'         % The transpose of c, i.e. a row vector.
c'*A       % Now matrix multiplication is permitted.
c*c'       % This multiplies a  4 x 1  by a  1 x 4  matrix.
c'*c       % What is this quantity usually called?
```

Exercise .20.
(a.) Calculate `inv(A)*A` *and* `A*inv(A)`*. Do they give the expected result? (Use the help command if you can't guess what the* `inv` *command does.)*
(b.) Verify for the matrices A and B above that $(AB)^{-1} = B^{-1}A^{-1}$.
(c.) Verify that the Matlab command `A^(-1)` *can also be used to form the inverse of* `A`*.*

6.4. Systems of Equations

Consider again the problem of solving the system of equations $Ax = b$. One obvious way that appeals to Mathematicians is to calculate $A^{-1}b$. As in the previous 3x3 case, we should also check that the solution is correct. We do this by calculating the residual $Ax - b$, for our computed solution x. Because of roundoff errors this residual may not be exactly zero, but all its components should be small; say less than 1×10^{-15}.

```
x = A^(-1)*b        % Solve the equation  A x = b
A*x , b             % Check  x  is correct by making
                    %   sure it satisfies the equations.
resid = A*x - b     % Check the residual is zero,
                    %    at least to within roundoff.
```

In fact it is far quicker, and usually more accurate, to solve equations using the backslash operator (\ ) introduced in the first section; rather than calculating with the inverse A^{-1}. The next lines use the backslash command to the solve the equations $Ax = b$, and check that it gives the same answers as those that were calculated using A^{-1}.

```
x1 = A \ b    % This is the fastest way to solve  A x = b
x - x1        % Here both answers are the same.
```

Exercise .21.
(a.) Solve the system of equations

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} x = \begin{bmatrix} 1/4 \\ 1/5 \\ 1/6 \end{bmatrix}$$

using \. *Check that the computed solution does actually satisfy the equations (i.e. check that* $Ax - b = 0$, *apart from rounding errors).*
(b.) Use `rand` *to generate a 700 x 700 matrix,* `A`, *and a column vector* `b` *of length 700. Solve the system of equations* $Ax = b$ *using the commands* `x = A \ b` *and* `x = inv( A ) * b ,` *timing each command with a watch. Which command is the faster and why? (Hint: before working with these large matrices use the* `clear` *command to remove all variables from Matlab's memory. This frees up enough memory to squeeze in this largish problem. If you have problems, try a smaller system of equations that fits onto the computer you are using.)*
(c.) Use the Matlab command `A = hilb(10)` *to set up the* 10×10 *Hilbert matrix. Create a right hand side vector b by* `b = A(:,1)`. *We are now going to solve the equations* $Ax = b$. *This is a common test problem in linear algebra. (What is the true solution* x *to the equations* $Ax = b$*?)*

(i) Solve the equations $Ax = b$ *using the backslash command, and then check that the computed solution satisfies the equations by calculating the residual,* $Ax - b$.

(ii) Now solve the equations by the command `A^(-1)*b` *and calculate the residual for this second solution.*

(iii) Which gives the better (i.e. smaller) residual?

(iv) Which method gives the smaller error? (Explanation: As b is the first column of A, you can work out the true value of x without computation. (Explain this briefly.) The error is the difference between Matlab's computed solution and the true solution. (The true solution, the one you work out by abstract thought, can be typed directly into Matlab.) The size of the error is the largest absolute value of the components of the error vector.)?

(d.) Extend part (c.) by solving a slightly different problem. Let the right hand side vector b be the first column of A, divided by 3; i.e. `b = A(:,1)/3`. *(What is the solution to the equation $Ax = b$ with this new right hand side?)*

(i) What is the size of the error and the size of the residual when the \ is used to solve the system with the new right hand side?

(ii) Extend the computation further by letting the right hand side be the j^{th} column of A divided by 3; i.e. `b=A(:,j)/3` *for $j = 1, 2, 3, \ldots, 10$. Complete the following table.*

Column j	*Size of the Residual*	*Size of the Error*
1	.	.
2	.	.
⋮	⋮	⋮
10	.	.

What conclusions do you draw? If we are worried about the accuracy of a computed solution is it sufficient to check that the solution satisfies the equations to within round off errors? (Hint: If `x_comp` *and* `x_true` *are the computed and true solutions the size of the error can be found from* `max(abs( x_comp - x_true ))`.

This exercise should show that the backslash operator \ is both faster and more accurate than the inverse. Clearly it is better to use the backslash and avoid the calculating inverses in numerical work!

6.5. Polynomials

Knowing about vectors, we can use the Matlab procedures dealing with polynomials. The polynomial $a_m X^m + a_{m-1} x^{m-1} + \ldots + a_1 X + a_0$ is represented by the Matlab vector $[a_m, a_{m-1}, \ldots a_1, a_0]$, a row vector of length $m + 1$. In the following example, $X^2 - 3X - 4$ is represented by the vector of coefficients $[1 \; -3 \; -4]$.

```
v = [ 1 -3 -4 ]        % Represents X^2 - 3 X -4
z = roots( v )
z(1)^ 2 -3*z(1) -4      %  Two ways to evaluate the poly.
polyval(v,z(1))        %    at the first root.
```

Exercise .22.
(a.) The polynomial $X^2 - 3X - 4$ has real roots, but $X^2 - 3X + 4$ has complex roots. Find these complex roots with `roots`.
(b.) Use help to find out how to use Matlab to calculate the derivative of a polynomial in Matlab. Test this Matlab command by calculating the derivative of $X^4 + X^3 + X^2 + X + 1$.

7. Graphs

Consider again the problem of graphing the functions, for example the function

$$f(x) = x|x|/(1+x^2)$$

over the interval $[-5, 5]$. Although we can use `ezplot` or `fplot`, we wish to illustrate more powerful methods, and the use of component-wise arithmetic.

7.1. Component Arithmetic

The new method is illustrated in the following lines.

```
x = (-5:.1:5)' ;
y = x .* abs(x) ./ ( 1 + x.^2) ;
plot( x , y , '-')
```

The first command produces a vector of x values from -5 to 5 in steps of .1. That is the column vector `x`=$[-5, -4.9, \cdots, 0, \cdots 4.9, 5]'$. The vector y contains the values of f at these x values. As there are so many points the graph of `x` against `y` looks like a smooth curve.

The novelties here are the operators `.*` , `./` , and `.^` in the second command. These are the so called *component–wise operators.* In the above example `x` is a column vector of length 101 and `abs(x)` is the column vector whose i^{th} entry is $|x_i|$. The formula `x*abs(x)` would be wrong, as this tries to multiply the 101×1 matrix `x` by the 101×1 matrix `abs(x)`. However the component-wise operation `x.*abs(x)` forms another vector of length 101 with entries $x_i|x_i|$. Continuing to evaluate the expression using component-wise arithmetic, we get the vector `y` of length 101 whose entries are $x_i|x_i|/(1+x_i^2)$. More explicitly

$$\texttt{x} = \begin{bmatrix} -5 \\ -4.9 \\ -4.8 \\ \vdots \end{bmatrix}, \quad \texttt{abs(x)} = \begin{bmatrix} |-5| \\ |-4.9| \\ |-4.8| \\ \vdots \end{bmatrix}, \quad \texttt{x.*abs(x)} = \begin{bmatrix} -5 \times |-5| \\ -4.9 \times |-4.9| \\ -4.8 \times |-4.8| \\ \vdots \end{bmatrix}$$

$$\texttt{x.*abs(x)./(1+x.^2)} = \begin{bmatrix} -5 \times |-5|/(1+(-5)^2) \\ -4.9 \times |-4.9|/(1+(-4.9)^2) \\ -4.8 \times |-4.8|/(1+(-4.8)^2) \\ \vdots \end{bmatrix}.$$

(Of course we should not become overconfident. This rule for dividing by vectors cannot be used in Mathematics proper.)

Note that p.*q and p*q are *entirely different*. Even if p and q are matrices of the same size and both products can be legitimately formed, the results will be different.

Exercise .23.

1. *(Easy) Find two 2×2 matrices p and q such that* `p.*q` $\neq$ `p*q`.
2. *(Hard) Find all 2×2 matrices p and q such that* `p.*q=p*q`.

Before more exercises, we show how to add further curves to the graph above. Suppose we want to compare the function we have already drawn, with the functions $x|x|/(5+x^2)$ and $x|x|/(\frac{1}{5}+x^2)$. This is done by the following three additional commands. (Remember x already contains the x–values used in the plot. These new commands are most easily entered by editing previous lines.)

```
y2 = x .* abs(x) ./ (5 + x.^2) ;
y3 = x .* abs(x) ./ ( 1/5 + x.^2) ;
plot(x,y,'-', x,y2,'-.', x,y3,'--')
```

Exercise .24. *(a.) Use Matlab to graph the functions* $\cos(x)$, $1/(1+\cos^2(x))$, *and* $1/(3+\cos(1/(1+x^2)))$ *on separate graphs.*
(b.) Graph $1/(1+e^{\alpha x})$, *for* $-4 \leq x \leq 4$ *and* $\alpha = .5, 1, 2$, *on the one plot. If you like fine graphics the following command will do a nice label*

```
title( '\it 1/(1+e^{\alpha x})for\alpha=.5,1,2.-4')
```

(Use ^ and _ for superscripts and subscripts in labels. Get Greek letters by using their names preceded by \ .)
(c.) Use `plot` *to draw a graph that shows*

$$\lim_{x\to 0} \frac{\sin(x)}{x} = 1.$$

(Hint: Select a suitable range and a set of x values and plot $\sin(x)/x$. *If you get messages about dividing by zero, it means you tried to evaluate* $\sin(x)/x$ *when* x *is exactly 0.)*

7.2. Printing Graphs

When the graph has been correctly drawn we will usually want to print it. As many graphs come out on the printer, it is a good idea to label the graph with your name. To do this use the `title` command.

```
                                        %Be sure to put quotes
title('Exercise 3.4 Kim Lee.')          %     around the title.
```

Bring the plot window to the front. The title should appear on the graph. If everything is okay, click on the 'file' menu → 'print' item in the graph window. If everything is set up correctly, the graph will appear on the printer attached to your computer.

After printing a graph, it is useful to take a pen and label the axes (if that was not done in Matlab), and perhaps explain the various points or curves in the space underneath. Alternatively before printing the graph, use the Matlab commands `title`, `xlabel`, `ylabel`and `legend` for a more professional appearance.

The most recent versions of Matlab let the user add text, lines and arrows interactively once the graph is on the screen. Use the 'tools' menu, and tick the 'show toolbar' option to make the tools appear. You may then add text, arrows and lines to the plot by pressing the buttons labelled 'A', ↗ and / .

Another option is to use the `print` command in Matlab to print the current graph to a file so that it can be printed later. For example if you have access to a postscript printer use the command

```
print -deps mygraph
```

This will write the current graph to the file `mygraph.eps`. This file can be stored on your floppy and printed out later.

7.3. Graphs in Reports

Instead of printing graphs directly, it is sometimes easier to copy Matlab graphs into a word processor such as Windows *Wordpad* or *MS Word*. (A character editor such as '*Notepad*' is not good enough). Then the graph can be included as a part of a written report.

- The word processor must be open at the same time as we are using Matlab.
- Move to the Matlab plot window, and use the *Edit* menu → COPY sub-command. (This copies the current graph into a buffer, a special storage area.)
- Then move to the word processor, place the cursor where graph is to go, and press Ctrl-V to paste the graph into the word processor.

Several graphs can be saved in a file this way and incorporated into the final report. Ensure though that the report file is saved on your floppy disk when you leave, not left on the internal hard drive.

If you are working in the computer laboratories, make sure the report file is small enough to fit on a floppy; and make sure you store the report on the floppy and take the floppy when you leave.

7.4. Saving Graphs

The easiest way to save a graph is to save the Matlab command you used to create the graph. These are usually only one or two lines. If the commands are more than one or two lines long, they should probably be saved in an M-file (see later chapters.)

7.5. Advanced Graphics

The commands given are sufficient to produce clear, simple graphs that suffice for most projects. However, may people are moved to do more:- to alter the size and fonts of labels, to change the thickness of lines etc. If you think this is necessary, more information is available.

- To demonstrate additional graphics features use the command `graf2d`. This program can also be found via the `demo` command, visiting the *language/graphics* area and the *Line plotting* option).
- A complete tutorial, *Using Matlab Graphics* is available in the Matlab documentation. In the HELP menu → HELP DESK (HTML) item. Look for the link to *Full Documentation Set* then the link to *Using Matlab Graphics.*

8. 3D Graphics

8.1. 3D Basics

We have already seen that Matlab is able spectacular 3 dimensional plots by just typing in the formula for the surface. To undertake more complicated graphing tasks we have to be explicit obout the coordinates of our surface. This means we have to use the component-wise operators and set up matrices for the x and y coordinates This is only a little more complicated than the 2D case.

To draw the surface of $f(x, y) = xy\exp(-(x\hat{}2 + y\hat{}2))$ for $-2 \leq x \leq 2$, $0 \leq y \leq 2$, use the commands.

```
x=(-2:.2:2)' ;  y = (0:.2:2)' ;
[ X, Y ] =  meshgrid(x,y) ;
Z  =  X .* Y .* exp( -( X.^2 + Y.^2 ) ) ;
surf(X,Y,Z)
```

```
xlabel('X')
ylabel('Y')          % Label axes to avoid confusion!
zlabel('Z')
```

To let us to 'walk' around the plot to see things from different angles give the command

```
rotate3d
```

(or press the rotation button on the figure's toolbar). Once this command has been given, click anywhere on the plot without releasing the mouse button. A box appears. Dragging the mouse rotates the box. When the button is released, the graph is redrawn from the new angle. Experiment a little!

The following commands draw slightly different mesh plots of the same function.

```
surfl(X,Y,Z) , surfc(X,Y,Z)
```

Instead of showing a complete surface, a'wire frame' is often faster to compute. This is the purpose of the three `mesh` commands:-

```
mesh(X,Y,Z) meshz(X,Y,Z) meshc(X,Y,Z)
```

Exercise .25.
(a.) Draw the above mesh and surface plots with a finer grid. (Use an 80×40 *grid rather than the* 20×10 *grid above.) Experiment with the three variants of the mesh commands and the three variants of the surf commands.*
(b.) Matlab can vary the colours . Look for a way to draw the surface in cool colors. How can one spin the map?
(c.) Use the help command or the help window to find out how to draw a curve in 3-space. (More help will be found under the topic `graph3d`*). Plot the curve* $(r(t)\sin(t),\ r(t)\cos(t),\ t)$, $r(t) = 1/(1 + (t/\pi)^2$ *for* $0 \leq t \leq 10\pi$. *Try not to use* `ezplot3`.

8.2. Advanced Options

In three dimensions it is more difficult to draw a plot that is pleasing to the eye and shows all the important features of a function.

- The interactive demonstration `graf3d` is the best way to show the effects of *shading* and *coloring* a surface. (This demonstration also found following links after the `demo` command.)

- The tutorial *Using Matlab Graphics* in the online documentation goes into further detail (controlling the position and colour of lights for example.)
- The help command with the topics `graphics`, `graph2d`, and `graph3d` provide more interesting features.

Exercise .26. *Use the* `graf3d` *interactive demonstration to investigate the function of two variables calculated by the Matlab* `peaks` *command. (Hint: By typing the command* `rotate3d` *into the command box you can rotate the surface interactively.) From appropriate plots*
(a.) Find a plot which shows clearly there are 3 local maxima. Choose a color scheme that can be used to determine the value of the maxima as accurately as possible.
(b.) Find another graph that shows as clearly as possible the locations of all local maxima and minima. (If you are submitting answers to this exercise, show the tutor your graphs on the screen, and hand in the commands you use to generate the plots.)

9. Functions

Often we need to work with functions. For example, we may want to find the minimum of a function, its zeros, or the definite integral of the function. This section explains how to create functions and work with them in Matlab.

9.1. Writing a Function

To set up functions in Matlab, we need to create a file containing the Matlab code that evaluates the function. These files are called *function files*. The name of the file must always end with the '.m' extension (for example `fred.m`, `funct.m`, etc.). Suppose we need to work with the function

$$f(x) = \frac{1}{(x-.3)^2+.01} + \frac{1}{(x-.9)^2+.04} - 6.$$

As this function has been called 'f', we will create the file `f.m` using the Matlab editor. To open the editor press the new file icon (the left icon above the command window). Once in the editor, type the following lines into the file.

```
                        % As the function is called 'f'
                        % put this code in the file 'f.m'
function  y  =  f( x )
%
```

```
%                                  % These are
%  This  is the  test function.    % comments.
%  from section 9.
%
   term1   =  1 ./( (x-.3).^2 + .01 )  ;
   term2   =  1 ./( (x-.9).^2 + .04 ) ;
   y = term1 + term2 - 6  ;
```

After the lines have been typed in, save the file (by pressing the disk icon). You will be asked for a name; in this case use `f.m`. Then return to the command window.

If the code has been typed into the file correctly, we should be able to use `f` the same way we use standard functions like sin and cos etc. For instance, can evaluate $f(.3)$, $f(.9)$, and plot f.

```
f(.3)        %  Check the code gives the correct answers
f(.9)        %      at   two test points.

x = (-1 : .05 : 2)';       % Plot  f  between  -1  and  2 .
plot( x , f(x) ) ;
```

To guard against errors in the code we should check the answers. Pencil and paper shows

$$f(.3) = \frac{1}{0^2 + .01} + \frac{1}{(-.6)^2 + .04} - 6 = 100 + \frac{1}{.4} - 6 = 96.5\,;$$

and similarly $f(.9) = 21.7027$. The Matlab answers should be the same. If these commands do not work or give wrong answers, there is probably a typing error. If so, go back into the file and correct the error. Save the file then try to use the function in Matlab again.

Sometimes Matlab does not notice the changes to the code and continues to report errors after we have corrected them. This is usually because we have forgotten to save the file to disk. Annother common cause is that we are editing a file in one directory and Matlab is reading the file in another directory. But occasionally this is due to a Matlab installation bug. To correct this bug, after changing code give the command

```
clear functions
```

9.2. Notes on Functions

9.2.1. Code is private

Let us look at the structure of 'f.m'.

1. The first line (apart from comments) is the header. It declares that the value of `x` comes from the main Matlab program (it is the input argument).
2. The value of `y` is calculated by the code in 'f.m' and returned to the main program (it is the output argument).
3. The code in the 'f.m' is kept separate from the main Matlab command window, so that `term1` in the code for `f` is different to a variable `term1` used in the main program. More pessimistically, the function does not know the value of a variable in the main program unless it is one of the input arguments.

9.2.2. Use . operators

It is best to use the component-wise operators in the code. (We used `./` rather than `/` for example.) This allows Matlab to correctly evaluate `f(x)`, even when `x` is a vector. Otherwise plot commands will not work for our function.

9.2.3. Other editors

It is possible to use your favourite editor not just the inbuilt Matlab editor. To use the windows *notepad* editor use the command

```
!notepad f.m &      %  Call the ''Notepad'' editor
                    %     to edit the file  f.m .
                    %  Remember the  &  at the end.
```

(If you do not remember the '&' at the end of this command, Matlab will refuse to run until you quit notepad.)

Exercise .27. *Write an M-file to code the function*

$$g(x) = (1 - 2x^2)\exp(-x^2).$$

(a.) Test your function by evaluating it at $x = 1$ and $x = -2$.
(b.) Test that your function works correctly when x is a vector by evaluating g for the vector

$$x = \begin{bmatrix} 1 \\ -2 \end{bmatrix}.$$

(c.) Plot your function for $-3 \leq x \leq 3$ *using both* `fplot` *and the* `plot` *command.*

We will now use our function $f(x)$ to demonstrate minimization, zero finding, and numerical integration in Matlab.

10. Using Functions

We continue to work with the function

$$f(x) = \frac{1}{(x-.3)^2+.01} + \frac{1}{(x-.9)^2+.04} - 6.$$

coded in the previous section.

10.1. 1D Minimization.

Matlab can find maxima and minima, but first we should read off the answers from a graph.

Exercise .28. *From the graph of* f*, write down the values of* x *which give*
(a.) the minimum of f *on the interval* [.25, 1],
(b.) the minimum of f *on the interval* [.5, 1],
(c.) the maximum of f *on the interval* [0, .5], *and*
(d.) the maximum of f *on the interval* [0, 4].

The point at which f takes its minimum is found using the function `fmin`.

```
xmin = fmin( 'f', .25, 1) , ymin = f( xmin )
```

This asks for the minimum of 'f' in the interval $[.25, 1]$. The answer coincides with the value expected from the graph. The procedure `fmin` uses a numerical method to find the minimum. It repeatedly evaluates the function until it is confident the minimum has been found to the required accuracy. In the above call to `fmin` the user did not specify the accuracy required, so the standard (or default) accuracy was used. Reading the information in `help fmin` tells us the default is a relative error of $\leq 1 \times 10^{-4}$.

10.1.1. Accuracy

A better answer can be found by specifying the required accuracy in an extra argument in `fmin`.

```
xmin_8 = fmin( 'f', .25,1, [0,1.e-8]), ymin_8 = f( xmin_8 )
```

(1.e-8 is shorthand for 1×10^{-8}.) With this call to `fmin`, `xmin_8` should be accurate to within a relative error of $\pm 1 \times 10^{-8}$; that is to about 8 significant digits. However, we cannot see this extra accuracy at the moment, as answers are only displayed to 4 or 5 digits. To see the answer fully we must change the display format. (That is the way Matlab displays numbers.) This can be done two ways:-

- choose the 'FILE' menu→'PREFERENCES' item in the Command Window, and then check any of the *long* formats, or
- by use the command `format long e`.

Now Matlab displays enough significant digits in the numbers to see that the answers are different.

```
[xmin ; xmin_8]      % Compare the two answers
```

Displayed in the long format, we see the two answers differ after the 6th decimal place, showing `fmin` has done more computations in the second case to get a slightly better answer.

It is important to realise that format does not change the accuracy of the numbers or the calculations: it just changes the number of digits displayed on the screen. Our first answer xmin is still only accurate to about 6 significant figures, even though it may be displayed to 15 figures.

Exercise .29. *There are two ways to verify our answer*
(1.) Evaluate f at `xmin` and `xmin_8` and see which answer gives the smaller value of f.
(2.) Evaluate f at `xmin_8-1.e-8` and `xmin_8+1.e-8`. The values either side of the computed minimum should be larger.

To save space on the screen, we can change back to the short display using the 'options' menu or the '`format short e`' command. Other useful formats are `long f` and `short f`. Do not use the rational format for serious computations. The fractions displayed are only approximations to the numerical answers.

Exercise .30.
(1.) Find the minimum of f on $[.5, 1]$. Choose an error tolerance so that the minimum is correct to $\pm 10^{-6}$. What is the value of f at this point? If x^* denotes the computed solution, check the accuracy of the answer by evaluating f at $x^* + 10^{-6}$ and $x^* - 10^{-6}$. f should be larger at both these neighbouring points. This shows the true minimum should lie between $x^* - 10^{-6}$ and $x^* + 10^{-6}$
(2.) Use `fmin` *to find the maximum of $f(x)$. (Hint: There is no function* `fmax`*! Instead, to find the maximum of $f(x)$, we can find the minimum of $-f(x)$. This means altering the code in the file 'f.m' so that it calculates $-f(x)$ rather than $f(x)$.) Use $[0, .5]$ as the original interval. Find the maximum point and the value of f at the maximum correct to $\pm 1 \times 10^{-6}$.*
(3.) Repeat the previous part and find the maximum of $f(x)$ in the wider interval $[0, 4]$. How accurate is the answer? Explain by looking at the graph of f ! (It is useful to graph a function before looking for a maximum or minimum!!)

10.2. Finding Zeros

Matlab also finds the zeros of functions; that is values of x for which

$$f(x) = 0.$$

For example, to find a zero of the function `f` near the point $x = -1$ use the command

```
xzero = fzero( 'f' , -1)
```

The answer should be accurate to at least the default tolerance. (Which is in fact around 1×10^{-16}, even though the help file claims it is 1×10^{-4}!!) The command `help fzero` gives more information about changing the default tolerance.

Exercise .31.
(1.) Check the previous result from `fzero` *by*

1. *Checking the answer is approximately true from the graph of f,*
2. *Checking f is approximately 0 at* `xzero`*.*
3. *Checking f is negative at* `xzero`-10^{-4} *and positive at* `xzero`$+10^{-4}$*. This shows that f crosses the axis somewhere in the small interval between* `xzero`-10^{-4} *and* `xzero`$+10^{-4}$`.`

(2.) Find the zero of f near $x = 1$ using the default tolerance. What answer is given when the default tolerance is changed to 1×10^{-6} and then to 1×10^{-12}. Check carefully to find which answer is the most accurate.

10.3. Integration.

Matlab also uses functions when it evaluates definite integrals numerically. This is known as *quadrature* or `numerical integration`. Suppose we want to evaluate the integral of f over $[-1, 2]$, that is to find

$$\int_{-1}^{2} f(x)\,dx. \tag{10.1}$$

(Or equivalently the area under the curve f between $x = -1$ and $x = 2$.) This is done by the function `quad8`; for example.

```
integral1 = quad8( 'f', -1,2 )
```

This command used `quad8` to compute the answer to within the default accuracy. For `quad8` this is a relative accuracy of $\pm 1 \times 10^{-3}$, or $\pm .1\%$. It is always wise to check, so we can now get quad8 to do more work and try to achieve a relative accuracy of 1×10^{-8}. The two answers can then be compared, either by displaying them in a long format or by dinding the difference between them.

```
integral2 = quad8( 'f', -1,2, 1.e-8 )
[ integral1 ; integral2 ]    % Use a long formats
                             %  to show the difference
(integral2-integral1)/integral2
```

The relative difference between these two answers is 8.5×10^{-9}, suggesting the first answer was much more accurate than expected! (Of course they could both be wrong; but this is unusual.)

10.3.1. Accuracy

It is hard to check the accuracy of Matlab's integration routine. One way is to code up a simple method of numerical integration in Matlab. In this case we use the *midpoint rule*. We divide the interval $[-1, 2]$ up into 15 smaller subintervals using the grid points

$$-1, -.8, -.6, \ldots, 1.6, 1.8, 2.$$

The area under the curve on each intervalis approximated by

$$\text{width of the interval } \times\ f(\text{ midpoint of the interval })$$

These individual approximations are then summed to get the overall area. That is we approximate the area under f by

$$\int_{-1}^{2} f(x)\,dx \quad \approx \quad \tfrac{2-(-1)}{15}\left(f(-.9) + f(-.7) + f(-.5) \ldots + f(1.7) + f(1.9)\right)$$

This rule is concisely coded in Matlab by

file=midpoint.eps,height=8.6481cm,width=11.0578cm

Figure 10.1: The Midpoint Rule.

```
x = ( -.9 : .2 : 1.9 )' ; fx = f( x ) ;
integral = sum( fx ) * 3 / 15
```

This crude approximation is not as accurate as that computed by `quad8`. However it is an independent way to show us that our answer is approximately correct.

Exercise .32.

(a.) Find the value of

$$\frac{1}{\sqrt{2\pi}} \int_{-3}^{3} e^{-x^2/2}\, dx$$

to a relative accuracy of .01% using `quad8`.

(b.) Check the answer using the midpoint rule with 30 points. (Hint: Evaluate f at the points `x=(-2.9:.2:2.9)`. *Use a 60 point midpoint rule to give a better answer.*

11. Differential Equations.

11.1. Scalar ODE's

The final application is the solution of ordinary differential equations. Suppose we want to find the solution $u(t)$ of the ordinary differential equation

$$\frac{du}{dt} = k\, u(t)\, (1 - u(t)), \quad t \geq 0; \tag{11.1}$$

This equation is a simple model for the spread of a disease. Consider a herd of P animals, and let $u(t)$ be the proportion of animals infected by the disease after t days. Thus $1 - u(t)$ is the proportion of uninfected animals. New infections occur when infected and uninfected animals meet. The number of such meetings is proportional to $u(t)\,(1 - u(t))$. Thus a simple model for the spread of the disease is that the new infections per day, du/dt, is proportional $u(t)\,(1 - u(t))$. That is we obtain equation (11.1), where the constant k depends on the density of the animals and the infectiousness of the disease. For definiteness, suppose here $P = 10000$ animals and $k = .2$. Initially at time $t = 0$ there is just one infectious animal, and so we add the initial condition

$$u(0) = 1/1000. \tag{11.2}$$

We want to find how many individuals will be infected over the next 100 days, that is to plot $u(t)$ from $t = 0$ to $t = 100$. To solve this problem we need to define a function to calculate the right hand side of (11.1), and then use the Matlab program `ode45`. Use the editor to put the following code into the file 'disease.m'.

```
function  u_dot  =  disease( t , u )      %  Place this code in
%                                         %  file disease.m
%  Evaluates the rhs of the ode

k = .2 ;
u_dot  =  k * u * ( 1  - u ) ;
```

(Although `t` is not used in calculating `u_dot`, `ode45` requires the function `disease` to have two input arguments (in the correct order). Use `help ode45` for more information.)

Having set up the problem, the differential equation is solved and graphed in just one Matlab one line!

```
[ t, u ] = ode45( 'disease', [0,100] , 1/10000  );
plot( t,u,'x')
```

The function `ode45` also has parameters which can be adjusted to control the accuracy of the computation. Mostly, these are not necessary for just plotting the solution to ordinary problems. However `ode45` does not guarantee its accuracy, and it is always a good idea to check our original solution by resolving the problem with higher accuracy requirements. For example, we can use `ode45` so that the estimated absolute and relative errors are kept less than 1×10^{-9}. Also to ensure a very smooth graph, values of the solution are computed every half day. This is done with the commands

```
options = odeset('abstol',1.e-9,'reltol', 1.e-9) ;
[tt,uu] = ode45('disease', [0:.5:100], 1/10000, options);
plot( t,u,'x', tt,uu,'-' )
```

We immediately see that the solution computed in the first call to `ode45` lies on top of the more accurate solution curve. That is the two solutions coincide to *graphical accuracy*.

For this simple model it is possible to find an analytic formula for $u(t)$. In fact

$$u(t) = \frac{1}{1 + c\exp(-kt)} \quad \text{where} \quad c = P - 1\,. \tag{11.3}$$

We can add this formula to the above graph as a final check.

```
t0 =  (0:.5:100)' ;
c = 10000-1 ; k = .2 ; u0 = 1 ./ ( 1 + c*exp( -k*t0 ) ) ;
plot( t,u,'x', tt,uu,'-', t0,u0,'--' )
```

The exact formula gives a curve that is the same as the numerical solution. In fact the difference between the two solutions is at most 2.5×10^{-7}. So we can be confident in using `ode45` to solve the problems below.

Exercise .33.
(a.) Check that (11.3) is a solution to (11.1). That is, when we differentiate this formula for $u(t)$ we get the right side of (11.1). And this formula satisfies (11.2)
(b.)From the graph of $u(t)$, estimate how many days are needed for half the population to be infected? (Hint: The command `grid` will make this easier to answer.)
(c.) Suppose now that after the infection is noted at day $t = 0$, an inoculation program is started from day 5, and 200 of the uninfected animals are inoculated per day. After t days, the proportion of animals inoculated per day is $200\,(t-5)_+/10000$. ($(t-5)_+$ means $\max\{t-5, 0\}$. This could be coded in Matlab as `t5 = max([t-5,0])` for example.) These inoculated animals cannot be infected, an new infections occur when an uninfected, uninoculated animal meets an infected animal. Thus the rate of increase of u is now modelled by the differential equation

$$\frac{du}{dt} \quad = \quad k\,u(t)\;\;\left(1 - u(t) - .02\,(t-5)_+\right)_+ ;$$

Now, how many animals are uninfected after 100 days with the inoculation program?
(d.) How many animals would be uninfected if only 100 animals could be inoculated each day from day 5? How many animals would have been saved if 100 animals per day were inoculated from day 0?

This simple model problem also has a formula for the exact solution. However in Matlab it is very easy to get a numerical solution; and this can be done even in complicated problems when there is no simple formulae for the solution. Using Matlab we can simply explore the model by altering the parameters and plotting the solutions!

11.2. Order 2 ODE's

As a second example, consider the ordinary differential equation

$$\frac{d^2u}{dt^2} + \sin(u(t)) \;\; = \;\; 0 \tag{11.4}$$

$$u(0) = \pi/4\,,\; \frac{du}{dt}(0) \;\; = \;\; 0 \tag{11.5}$$

This is a more complicated example because the second derivative of u appears, not just the first derivative. (This is called a second order ODE.) The Matlab ODE solvers cannot use second derivatives directly as they only work with first derivatives. However a sneaky trick can be used. If we keep track of both $u(t)$ and du/dt, then second derivatives of $u(t)$ are just first derivatives of du/dt. More formally, let $v(t)$ denote the value of du/dt. Then (13.2) is equivalent to the two equations

$$\frac{du}{dt} = v(t)\,, \quad \text{and} \quad \frac{dv}{dt} = -\sin(u(t))$$

Even more elegantly, we let $Y(t)$ be a vector with the two components

$$Y(t) = \begin{bmatrix} Y_1(t) \\ Y_2(t) \end{bmatrix} = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix},$$

and so

$$\frac{dY}{dt} = \begin{bmatrix} du/dt \\ dv/dt \end{bmatrix} = \begin{bmatrix} v(t) \\ -\sin(u(t)) \end{bmatrix} = \begin{bmatrix} Y_2(t) \\ -\sin(Y_1(t)) \end{bmatrix}$$

This is now in the form

$$\frac{dY}{dt} = F(t, Y(t)),$$

where for any vector Y, $F(t, Y)$ is defined by

$$F(t, Y) = \begin{bmatrix} Y_2 \\ -\sin(Y_1) \end{bmatrix}$$

Thus to solve (13.2) and plot $u(t)$ against t we place the following code in the file `F.m`

```
function    Y_dot  =  F( t,Y )
%                                  % Note  Y(2) is the Matlab
%  RHS for the 2nd order ODE       % notation for the second
%                                  % compontent of Y.  It does
%                                  % not mean Y(t) at time t=2
Y_dot  =  [  Y(2) ; -sin( Y(1) ) ] ;
```

Then we use the command

```
[t,Y]  = ode45( 'F', [0:.25:20], [pi/4;0] );  % Solve ODE and plot
u  =  Y(:,1)  ;  plot(  t, u,'-' )            %   u(t).
```

Note that `ode45` stores the first and second components of the solution in the first and second columns of the output.

Exercise .34. *In the equation (13.2) , $u(t)$ is the angle a swinging pendulum makes with the vertical axis at time t. If the pendulum is of length 1, the weight of the pendulum is at $(x, y) = (\sin(u(t)), -\cos(u(t))$. (The pendulum is swinging from the origin and is at position $(0, -1)$ when it is at rest.)*
(a.) From the graph, what is the period of the pendulum?
(b.) Draw a graph of the pendulum's path in the (x, y) plane.
(c.) Often when the pendulum swings through small angles, the equation (13.2) is approximated by

$$\frac{d^2u}{dt^2} + u(t) = 0,$$

$$u(0) = \pi/4\,, \frac{du}{dt}(0) = 0$$

How different is the solution to this linear approximation to the exact solution? Are the periods the same?

12. A Small Assignment

All numbers on a computer are stored as binary numbers. (These are like decimals, except each digit is in base 2). So fractions like 1/3 cannot be stored exactly, but will have to be rounded to the nearest number that can be stored. The accuracy depends on the number of digits used to store numbers on the computer. Typically each number is stored so the accuaracy is at least 16 decimal digits. The relative error will be less than 2.2×10^{-16}. As the details of binary arithmetic vary among computers, the accuracy expected on a particular machine is summarised by the easily computed quantity *machine epsilon*. Machine epsilon is defined as

$$\text{machine epsilon} := \min\{2^{-k} \,:\, (1 + 2^{-k}) - 1 > 0,\ \ k = 1, 2, \ldots\},$$

where here $(1 + 2^{-k}) - 1$ means the value computed by the machine under investigation. Mathematically, $(1 + 2^{-k}) - 1$ is always positive, but when k is large the computer will not have enough digits to store $(1 + 2^{-k})$ exactly, and eventually it will have to be rounded it down to 1. Then the computed value of $(1 + 2^{-k}) - 1$ will then be 0. The model assignment question we will do here is

> **Question:** Use Matlab to calculate machine epsilon on the Mathematics Department's computers. (Hint: Evaluate $(1 + 2^{-k}) - 1$ for k = 50, 51, . . . , 55 .)

In principle, we can just type in the commands to get Matlab to evaluate $(1 + 2^{-50}) - 1$, $(1 + 2^{-51}) - 1$, . . . $(1 + 2^{-55}) - 1$ one after the other. A better way is the remember the component-wise operations and use the following commands.

```
format short e
k = (50:55)' ; e = 2 .^(-k) ; er = (1+e) -1 ; tab = [ k e er ]
```

Here `k` is the vector of integers $50 \ldots 55$, and `er` contains the results of $(1 + 2^{-k}) - 1$. The table `tab` neatly summarises the computations. (Be sure to use the format `short e` to display the results at the terminal. The fixed point format `short f` is not suitable for answering this question, as some important small numbers will be displayed as 0.)

The computed results in `tab` are enough to answer this model assignment question, and we will need to hand up `tab` with the assignment. This could be done by just copying numbers from the terminal screen, but it is quicker to cut and paste these numbers to a word processor. Then we can neaten the table a little. Finally add some explanation to make it into our answer to the question. (Please ask for a demonstration if you do not know how to cut and paste numbers from the Matlab command window to your favourite editor or word processor.)

12.1. Model Answer

The matlab code below computes the value of $(1+2^{-k})-1$ for k=50,51,...,55. The results are shown in the table. The smallest value of 2^{-k} for which $(1+2^{-k})-1 > 0$ is $2^{-52} = 2.22 \times 10^{-16}$. Thus machine epsilon is 2.2×10^{-16}

```
   Results for the Machine Epsilon Computations

     k       2^(-k)       (1 + 2^(-k))-1
    --     ---------        ---------
    50     8.8818e-16       8.8818e-16
    51     4.4409e-16       4.4409e-16
    52     2.2204e-16       2.2204e-16
    53     1.1102e-16                0
    54     5.5511e-17                0
    55     2.7756e-17                0


       Matlab Code Used

k = (50:55)' ; e = 2 .^(-k) ;
er = (1+e) -1 ; tab = [ k e er ]
```

This question is very short. But it does illustrate some important points which hold for longer and more complicated problems.

- The question posed is answered in sentences
- Results are presented in tables (or graphs).
- Code is included, but it is usually not necessary to discuss it.

13. Larger Projects

In this section we show how Matlab is best used to do a more complicated assignment. It contains the following new ideas.

- Running longer code from files, rather than typing a long string of commands into the command window.
- Some simple programming in Matlab, using the `for` and `break` statements.

13.1. M-Files.

Script M-file or just *M-files* are plain text files containing Matlab code. For example, we may have 10 lines of Matlab code which computes results and then graphs the output. It is tedious to type in these 10 lines as we develop and change the code (even if we use the up arrow). However if we store the code in an M-file, say `ass.m`, then all the code in the file can be run by just typing in the file's name.

For example, let us place the following code from the 3D graphics section in the file `ass.m`. (Open the Matlab editor and type in the code. If you are reading the introduction on line, cut and paste the commands from the browser window to the editor. Save the file using the name `ass.m`.)

```
%                       This code goes
%                       in the file  ass.m .

x=(-2:.1:2)' ;  y = (-2:.1:2)' ;
X, Y ] =  meshgrid(x,y) ;
Z  =  X .* Y .* exp( -( X.^2 + Y.^2 ) ) ;
surf(X,Y,Z)
xlabel('X'); ylabel('Y'); zlabel('Z')
```

Now this code will be run by the single Matlab command

```
ass        % Run the code in the file  ass.m
```

If there are errors (there always are), we can quickly make corrections with our editor and run the code again. This is faster than repeatedly typing in the individual commands. M-files should be used if our task needs more than about 6 lines of Matlab.

Script M-files are similar to function files, but they do not contain the `function` statement at the beginning. Moreover, all the variables in the script M-file are the same as any variables appearing in the main Matlab command window. Thus running an M-file gives exactly the same result as if the code had been typed in from the keyboard. The names of these files must have the '.m' extension (e.g. 'ass.m').

13.2. Some Programming

If x is an approximate square root of a, then it has long been known that

$$x' = \frac{1}{2}\left(x + \frac{a}{x}\right)$$

is general a better approximation. For example, starting with 1 and working with pencil and paper we have the following rational approximations to $\sqrt{2}$

$$1, \frac{3}{2}, \frac{17}{12}, \frac{577}{408}, \cdots$$

If we decide to stop after 20 approximations, this computation may be written formally as

$$\begin{aligned} &x = 1 \\ &\text{for } k = 1, 2, 3, \cdots, 20 \\ &\qquad x = (x + \frac{a}{x})/2 \\ &\text{end} \end{aligned} \tag{13.1}$$

The third line in (13.1) does not mean x *equals* $(x + a/x)/2$, rather that the old value of x should be replaced with the new value $(x + a/x)/2$.

In practice we may have an accurate answer before we have computed all 20 new values of x. We could perhaps stop if $x^2 - a$ is sufficiently small; but this is extra computation. Instead, we will stop when the new value of x is sufficiently close to the previous value, that is if the calculations are not changing the answer very much. Hopefully this means we are close to the true answer.

Formally our method becomes

$$
\begin{aligned}
&x = 1 \\
&\text{for } k = 1, 2, 3 \cdots, 20 \\
&\qquad x' = (x + \frac{a}{x})/2 \\
&\qquad \text{if } |x - x'| \leq 10^{-14} \text{ stop} \\
&\qquad x = x' \\
&\text{end}
\end{aligned}
\tag{13.2}
$$

This can be coded in Matlab with a `for` statement, an `if` statement, and a `break` statement.

```
% Code to find sqrt(a)
x=1
for k=1:20
    xnew = (x+a/x)/2
  if abs(x-xnew)<=1.e-14, break, end
  x = xnew
end
```

This code can go in the file `mysqrt.m`

Once we have typed the code into the file mysqrt.m, we can test the code in Matlab with the command

```
a = 2 ; mysqrt     % Run ysqrt with a=2
xnew, k            % The approximation to 2^.5 was found
                   % after  k steps.
xnew-sqrt(a)       % Check accuracy.
```

Because `mysqrt` is not a function, the value of `a` can be used by the commands in `mysqrt`. The results of the calculation can be found by looking at the values of `k` and `xnew` in the main Matlab command window. This is particularly useful in finding out what has gone wrong if something does not work.

Exercise .35. *(a.) Adjust the code so that no intermediate results are printed out.*
(b.) Run `mysqrt` *with various values of* `a` *to complete the following table*

a	*Computed* $\sqrt{}$	*Steps needed*	*Error in the approximation*
2			
3			
4			
20			
1000			

Hint: To avoid retyping, use the following code.

```
tab = [ ] ;                          % tab contains the table of results
for a = [2 3 4 10 100 1000]
  mysqrt
  err = xnew - sqrt(a);
  tab = [ tab ; a xnew k err ];      % The new results are added
end                                  % at the end of the table.
```

(c.) Add a new first line to `mysqrt` *to change it to a function* `m-file`*. The input argument should be the number* `a`*, the output argument should be the computed square root,* `xnew`

(d.) Alter the code so that `mysqrt` *forms a table of values of* `k` *and* `xnew`*. Alter the function statement so that the function returns this table as a second output value.*

13.3. More Programming

Matlab contains the usual variety of statements to control the flow of programs. Details of these may be found in the online documentation. From the Help Desk follow the link *Getting Started* then read about *Flow Control*.

In Matlab, traditional programming ideas using loops should be avoided. Suppose I wish to graph a function on $[0, 2]$. Following the ideas of standard programs we could use the code

```
for i = 0:200
  x(i+1) = i/100 ;
  y(i+1) = cos(x(i)*pi) ;
end
plot( x,y,'-')
```

This is very, very slow, as well as being ugly and cumbersome. The elegant Matlab solution using component-wise operations is much nicer.

```
x = (0:.01:2)' ; plot( x, cos(x*pi), 'x')
```

13.4. Saving to Floppies.

To complete a big assignment you will need a number of sessions on the computer, and it is essential to keep a permanent copy of your programs and results from one day to the next. For this you will need at least one '*double sided, high density, 3.5 inch*' floppy disk; that is, one that can contain up to 1.44 Meg. These are obtainable at a number of shops on campus. (It is cheapest to buy a box of ten with 2 or 3 friends.) Please take sensible precautions against loosing your disks.

When Matlab is started the default directory is `D:\TEMP`, that is a directory on the internal hard disk. This means all files and results saved from within Matlab will be stored in this directory. Similarly Matlab looks for all M-files in this direcory. Unfortunately this directory is cleared when a new user logs on, and all work will be lost unless you remember to copy all files from `D:\TEMP` to your floppy. It is easy to forget to do this.

Thus it is better to make your floppy the default directory. Make your very first command in Matlab

```
cd a:\      % This should be the first Matlab command
            %  Your floppy is now the default directory.
```

Then you always have a copy of your work on your disk. Now you only have to remember to take your floppy our of the drive before you go.

14. Command Summary

The following Matlab examples illustrate some commonly used Matlab syntax. Consult this list if you need to know small details (where punctuation occurs for example).

```
x  =  ( 22/7 -16 + 1.9 )*3^2 ;
y  =  exp( -pi*i ) ;

ezplot('x^2/(1+x^2)')
ezplot('x^2+y^2-4',[0,2],[0,2])
ezplot( 'cos(t)','sin(t)',[0,2*pi])
fplot( '[ sin(x), x-x^3 ]',[0,pi] )

ezsurf( 'xy^2 / ( x^2+y^4 )' )
ezsurf('sqrt(1-t^2)*cos(s*pi)', ...
         'sqrt(1-t^2)*sin(s*pi)', 't', [-1,1] )

x =  0:.05:1 ;
```

```
fx =  ( 1 + x ) .* cos( x ) ./ ( 3 - x.^2 )   ;
plot(  x,fx,'--' ) ;

x = -2:.2:2  ;  y = 0:.2:2 ;
[X , Y] =  meshgrid( x,y ) ;
Z  =  X .* Y .* exp( -( X.^2 + Y.^2 ) ) ;
surf(X,Y,Z)

A  =  [  2 3 4; 5 6 7; -1 -1 1] ;
col1 = A(:,1) ; row2 = A(2,:) ;
A_lower = A(2:3,2:3) ;
odd  =  1:2:9 ; odd = odd' ;

b  =  [ 1;2;3 ] ;  x = A \ b ;     %  Solve A x = b
[ U,Lam ] = eig(A) ;               %  Find eigenvalues
                                   %    & eigenvectors.

I   = eye( 3 ) ;              % Set up special matrices
S  =  rand(4,1) ;
T  = diag( ones(4,1),-1 ) ...
       + diag( -2*ones(5,1) ) ...
         + diag( ones(4,1),1 );

x = 3 ;
for  k = 1:30
     x_new  = ( x + 3/x )/2 ;
  if abs( x - x_new ) < 1.e-12 , break , end
     x = x_new ;
end
                                % A new function  ff .
function   yp  =  ff( t,y )    % This code must be in
yp =   y / ( 1+ t^2 ) ;         % the m-file  ff.m
                                %
```