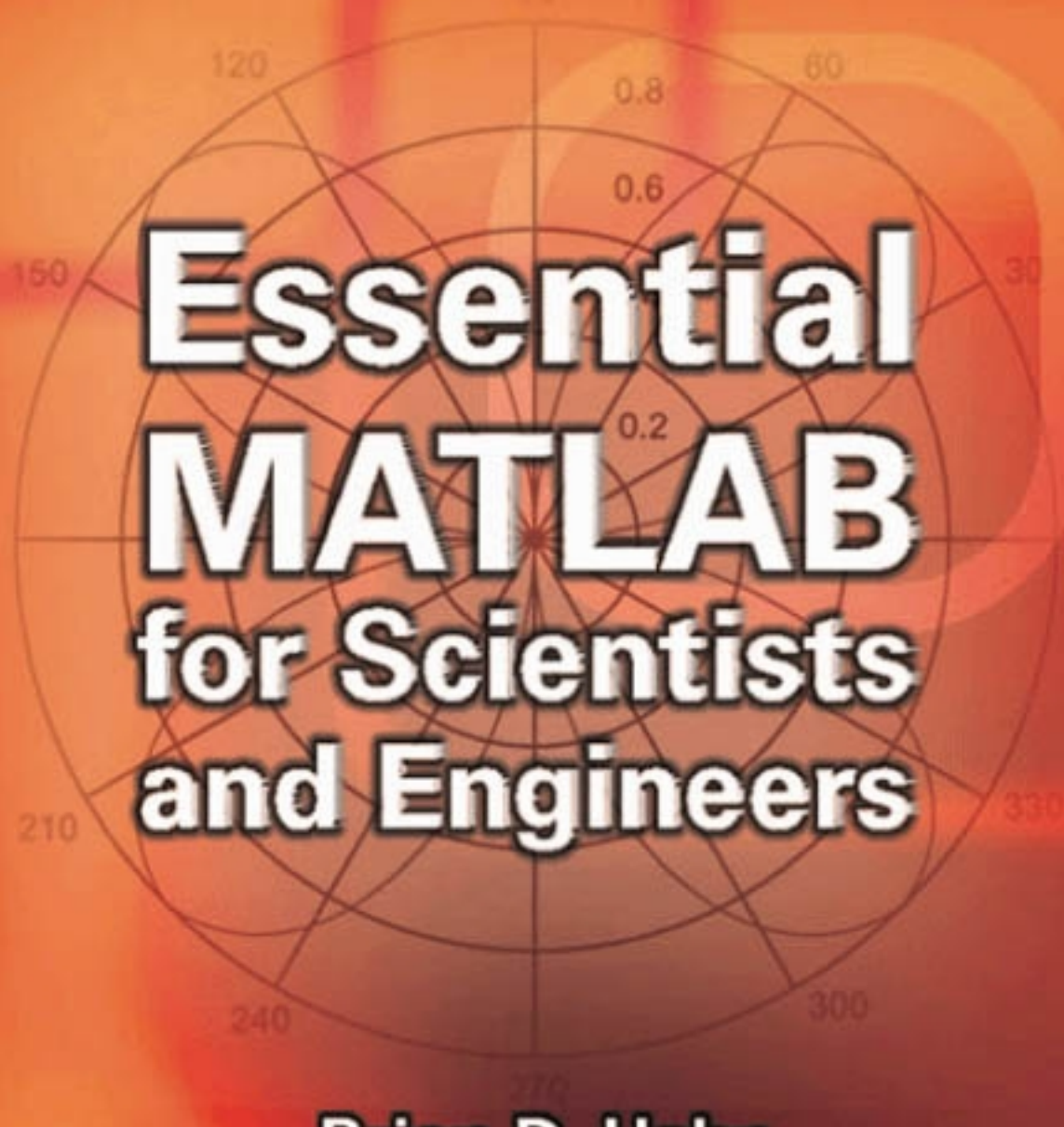


Second Edition



Essential MATLAB for Scientists and Engineers

Brian D. Hahn



Butterworth-Heinemann, an Imprint of Elsevier, Inc.
84 Theobald's Road, London WC1X 8RR, UK
Radarweg 29, PO Box 211, 1000 AE Amsterdam, The Netherlands
Linacre House, Jordan Hill, Oxford OX2 8DP, UK
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
525 B Street, Suite 1900, San Diego, CA 92101-4495, USA

Copyright © 2002 Elsevier Inc. All rights reserved

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording or otherwise without the prior written permission of the publisher. Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone (+44) (0) 1865 843830; fax (+44) (0) 1865 853333; email: permissions@elsevier.com. Alternatively you can submit your request online by visiting the Elsevier web site at <http://elsevier.com/locate/permissions>, and selecting Obtaining permission to use Elsevier material

Essential MATLAB for Scientists and Engineers
by Brian Hahn

ISBN 978-0-7506-5240-3

Part I Essentials	I
1 Getting going	3
Objective	3
1.1 Introduction	3
1.2 Using MATLAB.....	3
1.3 Website	9
Summary	9
2 MATLAB fundamentals	10
Objectives.....	10
2.1 The MATLAB desktop	11
2.2 Programs.....	11
2.3 Variables and the workspace	15
2.4 Arrays: vectors and matrices	17
2.5 Vertical motion under gravity	21
2.6 Operators, expressions and statements.....	22
2.7 Output.....	31
2.8 Repeating with for	33
2.9 Decisions.....	39
2.10 Complex numbers	45
2.11 More on input and output	46
2.12 Odds n ends	48
2.13 Programming style	50
Summary	51
3 Developing algorithms	58
Objective	58
3.1 Structure plans	58
3.2 Structured programming with functions	60
Summary	60
4 MATLAB functions.....	63
Objective	63
4.1 Projectile motion	63
4.2 Some common functions.....	65
Summary	69
5 Logical vectors.....	71
Objectives.....	71
5.1 Examples.....	72
5.2 Logical operators	75
5.3 Subscripting with logical vectors	78
5.4 Logical functions.....	79
5.5 Logical vectors instead of elseif ladders.....	81
Summary	83
6 Matrices	86
Objectives.....	86
6.1 Some basics.....	86
6.2 Matrix operations	98
6.3 Other matrix functions	100
Summary	100
7 Introduction to graphics.....	102
Objective	102
7.1 Basic 2-D graphs	102

7.2 3-D plots	110
Summary	120
8 Loops	124
Objectives.....	124
8.1 Determinate repetition with for	124
8.2 Indeterminate repetition with while	126
Summary	132
9 Errors and pitfalls	136
Objective	136
9.1 Syntax errors	136
9.2 Pitfalls and surprises	138
9.3 Errors in logic	139
9.4 Rounding error	139
9.5 Trapping and generating errors	140
Summary	140
10 Function M-files	142
Objective	142
10.1 Some examples	142
10.2 Basic rules	144
10.3 Function handles	149
10.4 Command/function duality	150
10.5 Function name resolution	151
10.6 Debugging M-files	151
10.7 Recursion	153
Summary	154
Part II More Advanced Topics and Applications	II
11 Vectors as arrays: working with subscripts	159
Objective	159
11.1 Update processes	159
11.2 Frequencies, bar charts and histograms	164
11.3 Sorting	166
Summary	168
12 Arrays of characters: strings	170
Objective	170
12.1 Basic concepts	170
12.2 Two-dimensional strings	173
12.3 eval and text macros	174
Summary	176
13 Advanced data structures	178
Objectives.....	178
13.1 Structures	178
13.2 Cell arrays	180
13.3 Classes and objects	182
Summary	183
14 More graphics	184
Objectives.....	184
14.1 Handle Graphics.....	184
14.2 Editing plots	189
14.3 Animation	190
14.4 Colour etc.	193

14.5 Lighting and camera	195
14.6 Saving, printing and exporting graphs	196
Summary	197
15 Graphical User Interfaces (GUIs)	198
Objectives	198
15.1 Basic structure of a GUI	198
15.2 A first example: getting the time	199
15.3 Newton again	202
15.4 Axes on a GUI	204
15.5 Adding colour to a button	205
Summary	206
16 Importing and exporting data	207
Objectives	207
16.1 The load and save commands	207
16.2 The Import Wizard	208
16.3 Low-level file I/O functions	209
16.4 Other import/export functions	212
Summary	213
17 Simulation	214
Objective	214
17.1 Random number generation	214
17.2 Spinning coins	215
17.3 Rolling dice	216
17.4 Bacteria division	216
17.5 A random walk	216
17.6 Traffic flow	218
17.7 Normal (Gaussian) random numbers	220
Summary	220
18 More matrices	224
Objectives	224
18.1 Leslie matrices: population growth	224
18.2 Markov processes	227
18.3 Linear equations	229
18.4 Sparse matrices	233
Summary	235
19 Introduction to numerical methods	236
Objective	236
19.1 Equations	236
19.2 Integration	240
19.3 Numerical differentiation	242
19.4 First-order differential equations	243
19.5 Linear ordinary differential equations (LODEs)	247
19.6 Runge Kutta methods	247
19.7 A GUI ODE solver: Driver	251
19.8 A partial differential equation	260
19.9 Other numerical methods	262
Summary	263
A Syntax quick reference	267
A.1 Expressions	267
A.2 Function M-files	267
A.3 Graphics	267

A.4 if and switch	268
A.5 for and while.....	268
A.6 Input/output	269
A.7 load/save.....	270
A.8 Vectors and matrices	270
B Operators	271
C Command and function quick reference.....	272
C.1 General purpose commands	273
C.2 Logical functions	273
C.3 Language constructs and debugging.....	274
C.4 Matrices and matrix manipulation	274
C.5 Mathematical functions	275
C.6 Matrix functions.....	276
C.7 Data analysis	276
C.8 Polynomial functions.....	276
C.9 Function functions.....	276
C.10 Sparse matrix functions	276
C.11 Character string functions.....	277
C.12 File I/O functions	277
C.13 Graphics	277
D ASCII character codes	279
E Solutions to selected exercises	280
Index.....	291

Part I

Essentials

1

Getting going

Objective

The objective of this chapter is to enable you to

- use some simple MATLAB commands from the Command Window.

1.1 Introduction

MATLAB is a powerful computing system for handling the calculations involved in scientific and engineering problems. The name MATLAB stands for MATrix LABoratory, because the system was designed to make matrix computations particularly easy. If you don't know what a matrix is, don't worry—we will look at them in detail later. For the moment we can forget about them.

This book assumes that you have never used a computer before to do the sort of scientific calculations that MATLAB handles. You will however need to be able to find your way around a computer keyboard and the operating system running on your computer (e.g. Windows or UNIX). The only other computer-related skill you will need is some very basic text editing.

One of the many things you will like about MATLAB (and which distinguishes it from many other computer programming systems, such as C++ and Java) is that you can use it *interactively*. This means you type some commands at the special MATLAB prompt, and get the answers immediately. The problems solved in this way can be very simple, like finding a square root, or they can be much more complicated, like finding the solution of a system of differential equations. The point is that you have to enter only one or two commands, and you get the answers at once. MATLAB does most of the work for you.

In the rest of this Chapter we will look at some simple examples for you to try out. Don't bother about understanding exactly what is happening. The understanding will come in later chapters when we look at the details.

1.2 Using MATLAB

In order to use MATLAB it must either be installed on your computer, or you must have access to a network where it is available. Throughout this book the latest version of MATLAB at the time of writing is assumed—Version 6.1 (Release 12.1).

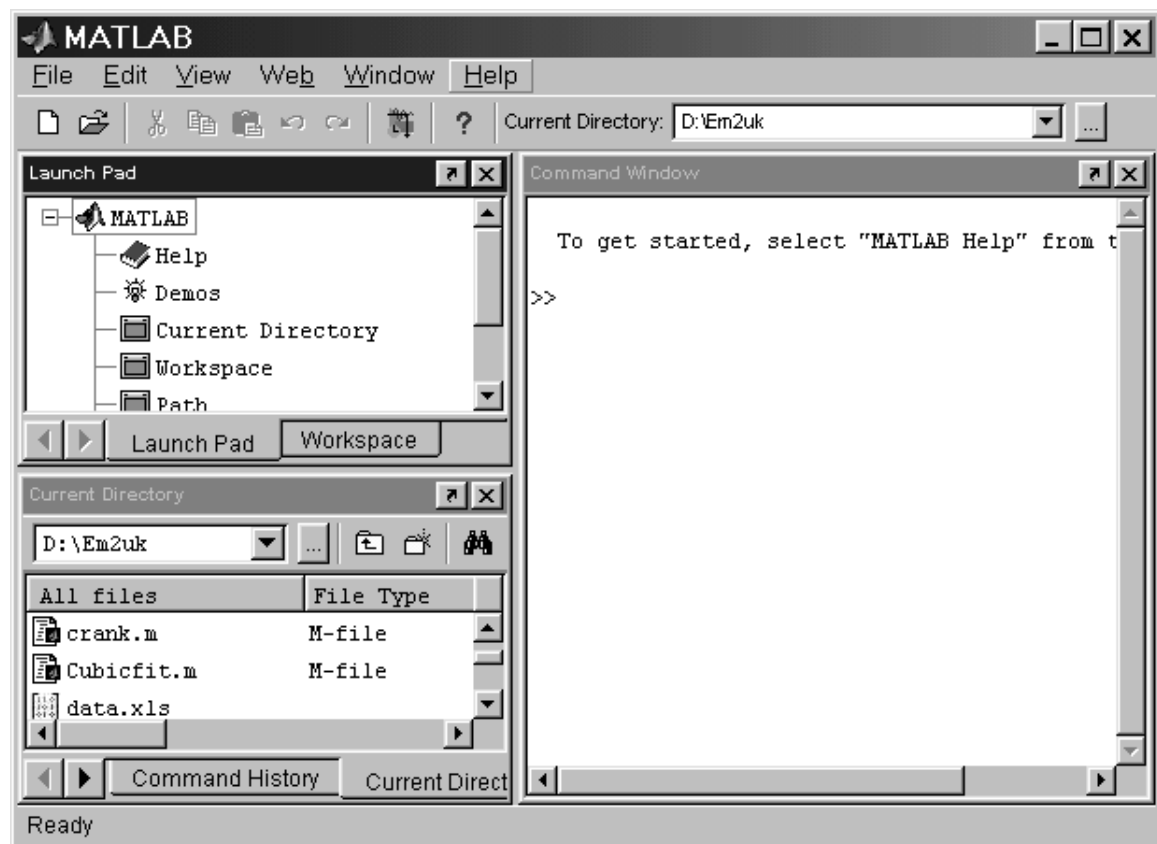


Figure 1.1 The MATLAB desktop

To start MATLAB from Windows, double-click the MATLAB icon on your Windows desktop. To start it from a UNIX platform, type `matlab` at the operating system prompt. When MATLAB starts, the MATLAB desktop opens as shown in Figure 1.1. The window in the desktop that concerns us for this chapter is the Command Window, where the special `>>` prompt appears. This prompt means that MATLAB is waiting for a command. You can quit MATLAB at any time with one of the following:

- Click on the close box in the top right corner of the MATLAB desktop.
- Select **Exit MATLAB** from the desktop **File** menu.
- Enter `quit` or `exit` at the Command Window prompt.

Once you have started MATLAB try the following exercises in the Command Window. If necessary, make the Command Window active by clicking anywhere inside its border.

1. First let's see if MATLAB is any good at arithmetic.

- (a) Type `2+3` after the `>>` prompt, followed by **Enter**, i.e. press the **Enter** key, indicated by `<Enter>` below:

```
>> 2+3    <Enter>
```

Commands are only carried out when you press **Enter**.

- (b) If MATLAB got that right, try $2*3$, i.e.

```
>> 2*3    <Enter>
```

What about $1/2$ and 2^3 ? Can you figure out what the symbols $*$, $/$ and $^$ mean?

- You can edit a MATLAB command before pressing **Enter** by using various combinations of the **Backspace**, **Left-arrow**, **Right-arrow** and **Del** keys.
 - The line with the \gg prompt is called the *command line*.
 - You can select (and edit) previous commands you have entered using **Up-arrow** and **Down-arrow**. But remember to press **Enter** to get the command carried out. This helpful feature is called *command line editing*.
 - MATLAB has a useful editing feature called *smart recall*. Just type the first few characters of the command you want to recall, e.g. type the characters $2*$ and press the **Up-arrow** key—this recalls the most recent command starting with $2*$.
- (c) How do you think MATLAB would handle $0/1$ and $1/0$? Try it.
MATLAB is sensible about errors. It warns you in case you didn't realize you were dividing by zero, but still gives the answer `Inf`. If you insist on using ∞ in a calculation, type the symbol `Inf` (short for *infinity*), e.g. try $13+\text{Inf}$ and $29/\text{Inf}$.
- (d) Another special value that you may meet is `NaN`, which stands for *Not-a-Number*. It is the answer to calculations like $0/0$.

2. Now for some algebra.

- (a) Enter the command (in programming jargon a *statement*) $a = 2$, i.e. the MATLAB command line should look like this:

```
>> a = 2    <Enter>
```

a is called a *variable*. This statement *assigns* the value of 2 to a . (Note that this value is displayed immediately after the statement is executed.) Now try entering the statement $a = a + 7$ followed on a new line by $a = a * 10$. Do you agree with the final value of a ?

- (b) Now enter the statement

```
b = 3;    <Enter>
```

Can you see the effect of the semi-colon ($;$)? It prevents the value of b from being displayed. However, b still has the value 3 as you can see simply by entering its name without a semi-colon at the prompt.

- (c) Assign any values you like to two variables x and y . Now see if you can in a single statement assign the *sum* of x and y to a third variable z .
3. MATLAB has most of the usual mathematical functions that you will find on your calculator, like `sin`, `cos`, `log` (meaning the *natural* logarithm), as well as a lot more.
- (a) Find $\sqrt{\pi}$ with the command `sqrt(pi)`. The answer should be 1.7725. Note that MATLAB knows the value of `pi`, because it is one of MATLAB's very many built-in functions.
- (b) Trigonometric functions like `sin(x)` expect the argument x to be in *radians*. Multiply degrees by $\pi/180$ to get radians. For example, use MATLAB to calculate $\sin(90^\circ)$. The answer should be 1, i.e. `sin(90*pi/180)`.
- (c) The exponential function e^x is computed in MATLAB as `exp(x)`. Use this information to find e and $1/e$ (2.7183 and 0.3679).
4. MATLAB has a lot of general functions. Try `date` and `calendar` for starters.
5. MATLAB also has a number of *commands*, such as `clc` (for *clear command window*). `help` is another command you will use a lot (see below). The difference between functions and commands is that functions usually return with a value, e.g. the date, while commands tend to change the environment in some way, e.g. by clearing the screen, or saving some statements to disk.

6. Variables such as `a` and `b` above are called *scalars*; they are single-valued. MATLAB also handles *vectors* (generally referred to in MATLAB as *arrays*), which are the key to many powerful features of the language. The easiest way of defining a vector where the elements (components) increase by the same amount is with a statement like

```
>> x = 0 : 10;
```

Enter it. That's a *colon* (`:`) between the 0 and the 10. There's no need to leave a space on either side of it, but it makes it more readable. Enter `x` to check that `x` is a vector now.

- (a) The really cool thing about MATLAB is that other vectors can now be defined in terms of our vector `x`. Try

```
>> y = 2 * x
```

and

```
>> z = sin(x)
```

(no semi-colons).

- (b) All you have to do to draw the graph of $\sin(x)$ is to enter the command

```
>> plot(x, sin(x))
```

(or simply `plot(x, z)` if you defined `z` correctly). The graph appears in a separate figure window (see Figure 1.2). You can select the Command Window or figure windows by clicking anywhere inside them, or you can use the **Windows** pull-down menus in any of these windows.

- (c) The graph looks rather crude, because more points need to be plotted between 0 and 10. To fix this, make the vector `x` go up in steps of 0.1 instead of 1 like this:

```
>> x = 0 : 0.1 : 10;
```

When there are three numbers separated by two colons in this way, the middle number is the *increment*. Now enter `plot(x, sin(x))` to get a much neater graph.

You can put a grid on the graph to make it easier to read with

```
>> plot(x, sin(x)), grid
```

- (d) If you want to see more cycles of the sine graph just use command-line editing to change `sin(x)` to `sin(2*x)`.
- (e) Try drawing the graph of $\tan(x)$ over the same domain. You may find aspects of your graph surprising. A more accurate version is presented in Chapter 5.
- (f) Another useful Command Window editing feature is *tab completion*: type the first few letters of a MATLAB name and then press the **Tab** key. If the name is unique, it is automatically completed. If the name is not unique, press the **Tab** a second time to see all the possibilities. Try this feature out, e.g. by typing `ta` at the command line followed by **Tab** twice.
7. If you're into linear equations, you can solve two simultaneous equations very easily, e.g.

$$x + 2y = 4,$$

$$2x - y = 3.$$

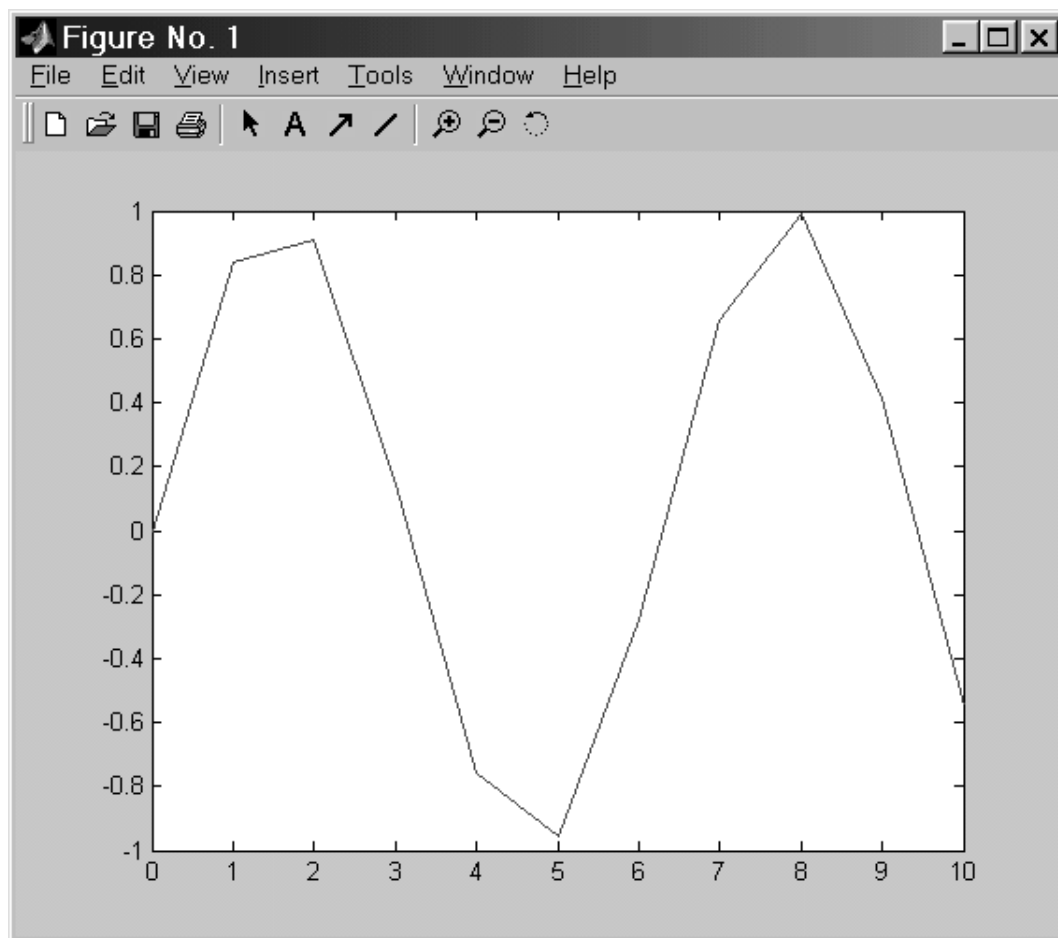


Figure 1.2 A figure window

All you have to do is type the following commands (exactly as they are)

```
>> a = [1 2; 2 -1];
>> b = [4; 3];
>> x = a\b
```

which result in

```
x =
    2
    1
```

i.e. $x = 2$, $y = 1$.

8. If you want a spectacular sample of what MATLAB has to offer, try `demo` at the command line. Alternatively, double-click **Demos** in the Launch Pad. (If you can't see **Demos**, click on the + next to the MATLAB item in the Launch Pad to expand it.) For a listing of demonstration programs by category try `help demos`.

9. MATLAB has a very useful ‘help’ system, which we look at in more detail in Chapter 2. For the moment type `help` at the command line to see all the categories on which you can get help. E.g. type `help elfun` to see all MATLAB’s elementary mathematical functions. `lookfor` enables you to search for a particular string in the help text of functions, e.g. `lookfor eigenvalue` displays all the functions relating to eigenvalues.
10. MATLAB has all sorts of other goodies. For example, you can generate *magic squares*, e.g. `magic(10)`, where the rows, columns and the main diagonal all add up to the same value. Try it. In general, an $n \times n$ magic square has a row and column sum of $n(n^2 + 1)/2$. You can even get a *contour* plot of the elements of a magic square. MATLAB pretends that the entries in the square are heights above sea level of points on a map, and draws the contour lines. `contour(magic(22))` looks rather nice.
11. If you want to see the famous Mexican hat shown in Figure 1.3, enter the following four lines (be careful not to make any typing errors):

```
>> [x y] = meshgrid(-8 : 0.5 : 8);
>> r = sqrt(x.^2 + y.^2) + eps;
>> z = sin(r) ./ r;
>> mesh(z);
```

Try `surf(z)` to generate a faceted (tiled) view of the surface. `surf(z)` or `meshc(z)` draws a 2-D contour plot under the surface. The command

```
>> surf(z), shading flat
```

produces a rather nice picture by removing the grid lines.

12. If your PC has a speaker you could try

```
load handel
sound(y, Fs)
```

for a snatch of Handel’s Hallelujah Chorus.

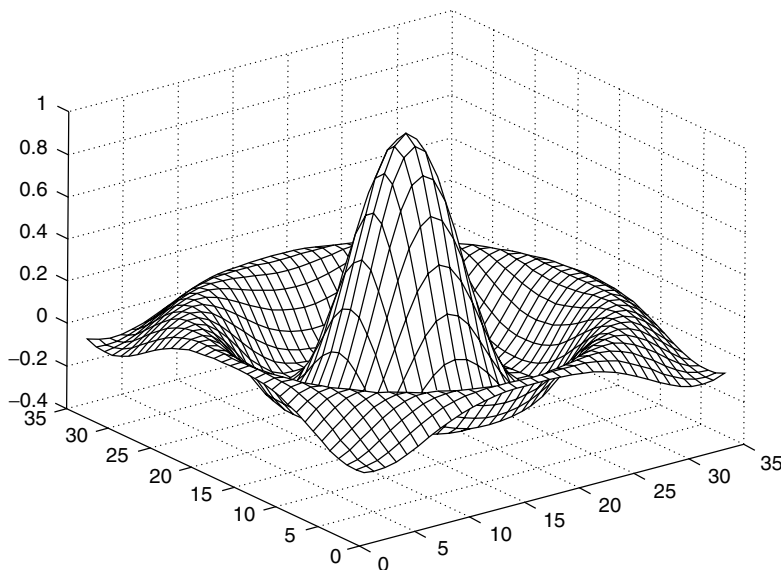


Figure 1.3 The Mexican hat

For some different sounds you can try loading `chirp`, `gong`, `laughter`, `splat` and `train`. You have to run `sound(y, Fs)` for each one.

13. If you want to see a view of the Earth from space, try

```
load earth
image(X); colormap(map)
axis image
```

14. Finally, if you're really bored, try `why`. Why not?

1.3 Website

Source code for most of the examples and solutions to exercises in this book can be downloaded from its website at www.bh.com/companions/essentialmatlab

Summary

- MATLAB is a matrix-based computer system designed to assist in scientific and engineering problem solving.
- To use MATLAB, you enter commands and statements on the command line in the Command Window. They are carried out immediately.
- `quit` or `exit` terminates MATLAB.
- `clc` clears the Command Window.
- `help` and `lookfor` provide help.
- `plot` draws an x - y graph in a figure window.
- `grid` draws grid lines on a graph.

Exercise

- 1.1 Give values to variables `a` and `b` on the command line, e.g. `a = 3` and `b = 5`.
Write some statements to find the sum, difference, product and quotient of `a` and `b`.

Solutions to many of the exercises are in Appendix E.

2

MATLAB fundamentals

Objectives

The objectives of this chapter are to introduce you to some of the fundamentals of MATLAB programming, including:

- various MATLAB desktop and editing features;
- variables, operators and expressions;
- vectors (arrays);
- basic input and output;
- repetition (`for`);
- decisions (`if`).

By now you will probably be wanting to use MATLAB to solve problems of your own. In this chapter we will look in detail at how to write MATLAB statements to solve simple problems. There are two essential requirements for successful MATLAB programming:

- You need to learn the *exact* rules for writing MATLAB statements.
- You need to develop a logical plan of attack for solving particular problems.

This chapter is devoted mainly to the first requirement: learning some basic MATLAB rules. Computer programming is a precise science (some would say it is also an art); you have to enter statements in *precisely* the right way. If you don't, you will get rubbish. There is a saying among computer programmers:

Garbage in, garbage out.

If you give MATLAB a garbage instruction, you will get a garbage result.

Once you have mastered the basic rules in this chapter, you can go on to more interesting and substantial problems. But first we need a quick tour of the MATLAB desktop.