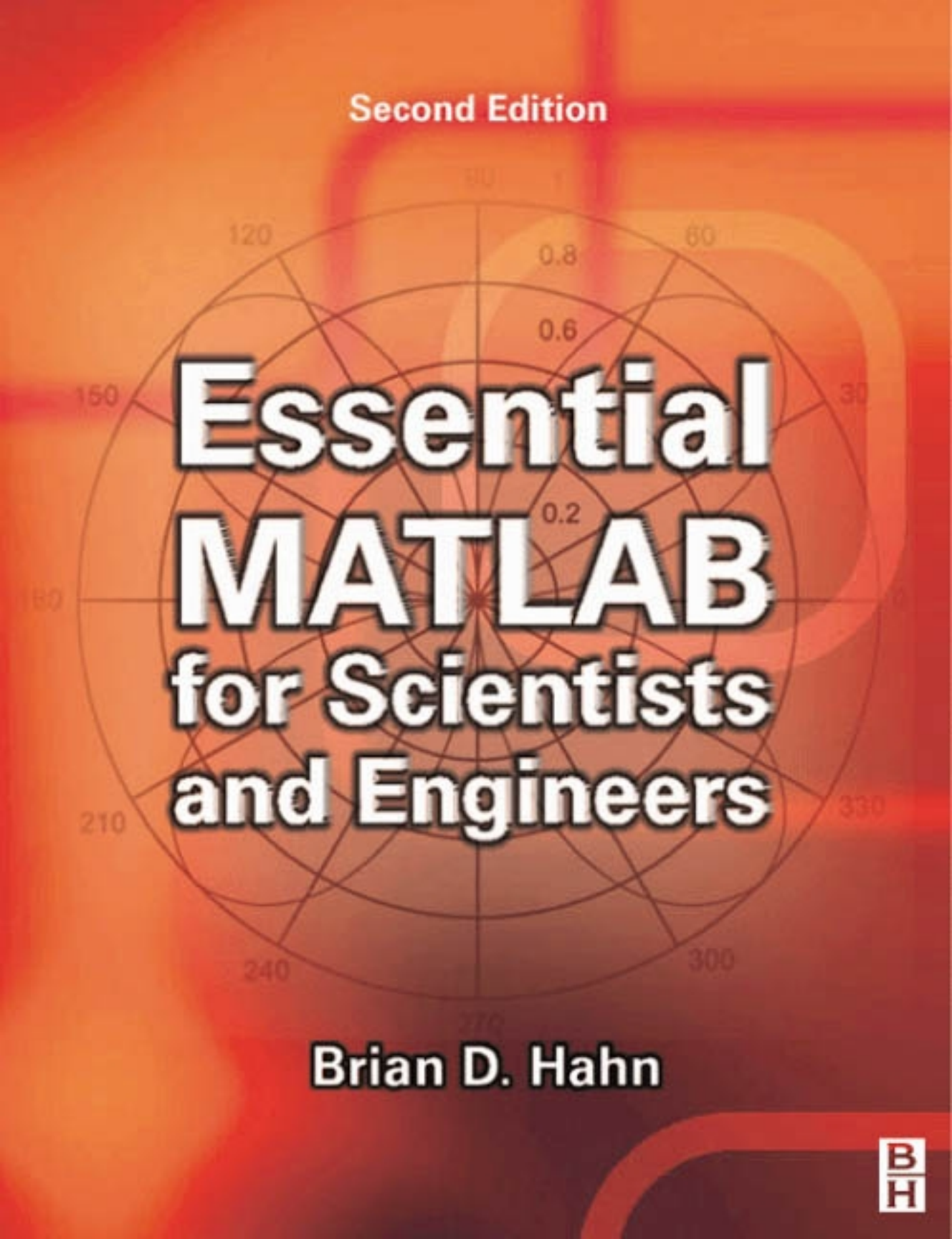
Second Edition
Essential MATLAB for Scientists and Engineers
Brian D. Hahn
BH

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

Essential MATLAB for Scientists and Engineers
by Brian Hahn

ISBN 978-0-7506-5240-3

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).

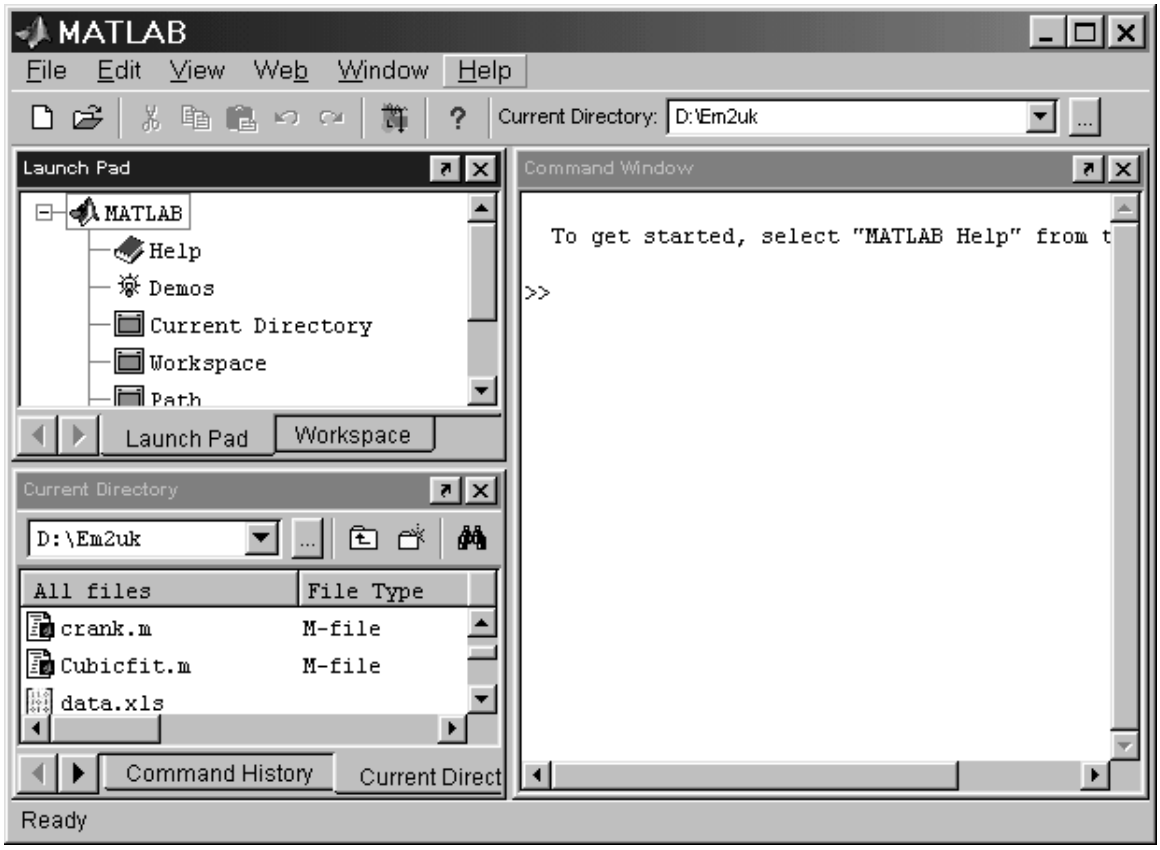

Figure 1.1 The MATLAB desktop

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 ≫ 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+Inf` and `29/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°). 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.

$$\begin{aligned} x + 2y &= 4, \\ 2x - y &= 3. \end{aligned}$$

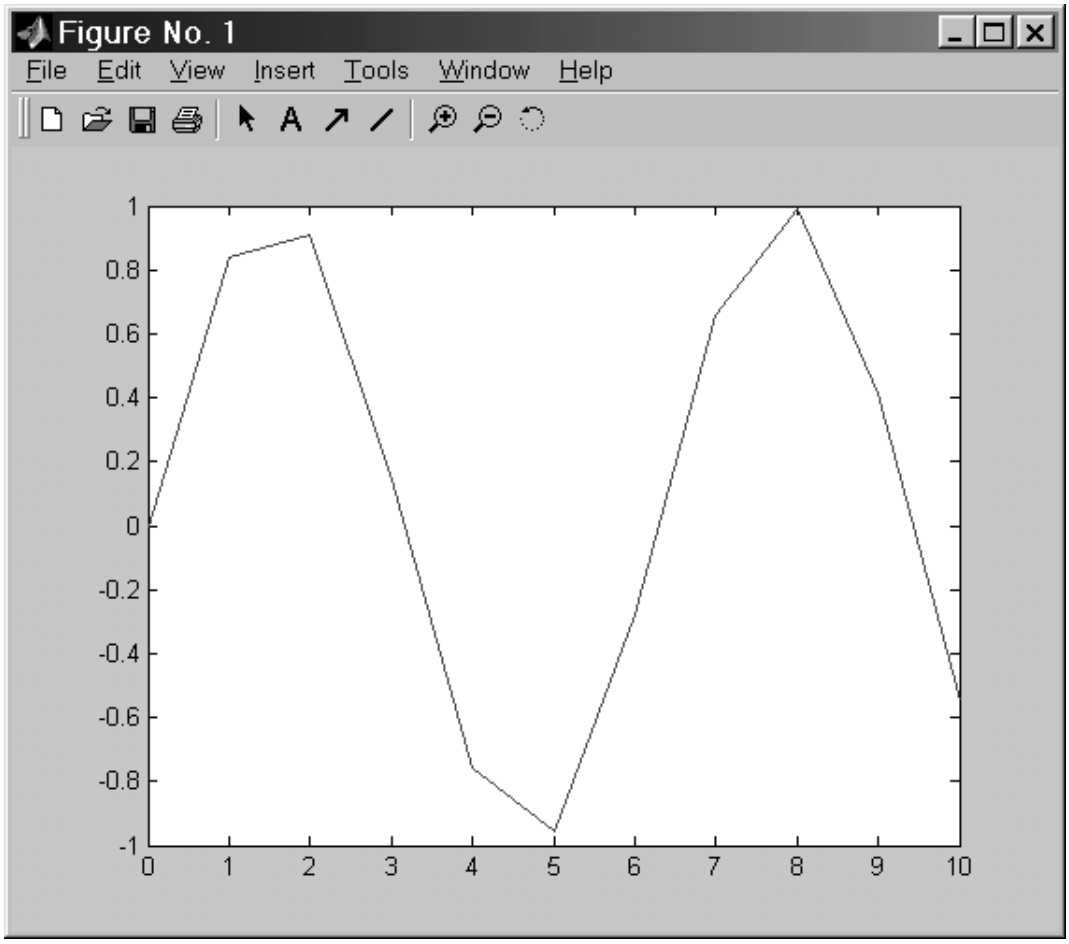

Figure 1.2 A figure window

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.
    `surfc(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.

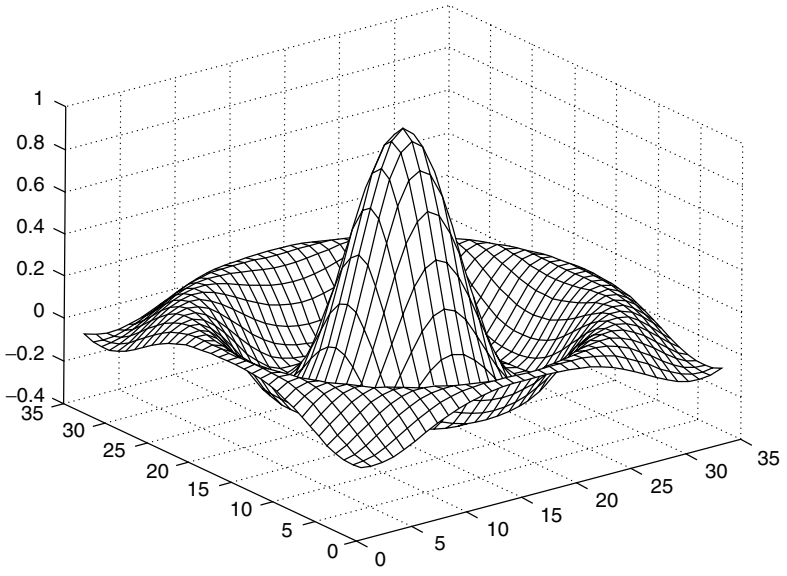

Figure 1.3 The Mexican hat

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.