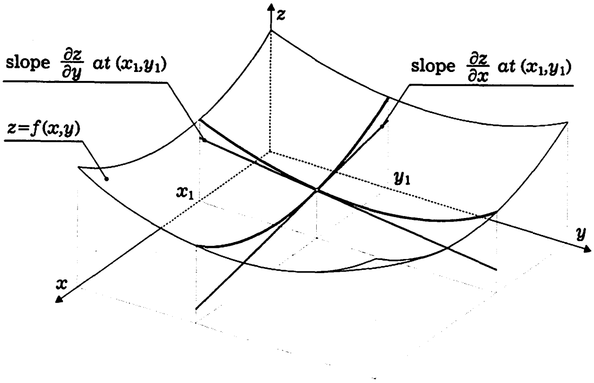

Figure B.1: Diagram showing the interpretation of the partial derivatives as the slopes of the graph of a function of two variables in the directions of the x and y axes.

The notation for these partial derivatives is given below, where it is worth noting the order of x and y in the subscripts.

$$\frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) = \frac{\partial^2 f}{\partial y\,\partial x} = f_{xy},$$

$$\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right) = \frac{\partial^2 f}{\partial x\,\partial y} = f_{yx},$$

$$\frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right) = \frac{\partial^2 f}{\partial y\,\partial y} = f_{yy},$$

$$\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) = \frac{\partial^2 f}{\partial x\,\partial x} = f_{xx}.$$

It is not always true that $f_{xy} = f_{yx}$. However, if all the derivatives concerned are continuous we do have $f_{xy} = f_{yx}$ and the order of differentiation does not matter. This was the case in Example 1 above.

For the partial derivatives then, y fixed at $y = y_1$ represents a vertical plane which intersects the surface $z = f(x,y)$ in a curve, and $\partial z/\partial x$ at the point (x_1, y_1) is the slope of the tangent to this curve at (x_1, y_1). (Similarly for x fixed at $x = x_1$.) This is illustrated in Figure B.1.

Taylor series expansions

We recall Taylor's Theorem for a single variable, and then extend the definition to functions of two variables.

Taylor's Theorem for a single variable. If F and its first n derivatives F', F'', ..., $F^{(n)}$ are continuous on some interval $[a, b]$ and $F^{(n)}$ is differentiable on (a, b), then there exists a number c_{n+1}, between a and b, such that

$$\begin{aligned}F(b) =& F(a) + F'(a)\,(b-a) + \frac{F''(a)}{2!}(b-a)^2 + \dots \\ &+ \frac{F^{(n)}(a)}{n!}(b-a)^n + \frac{F^{(n+1)}}{(n+1)!}(c_{n+1})\,(b-a)^{n+1}.\end{aligned}$$

This equation, in the single variable case, provides an extremely accurate polynomial approximation for a large class of functions which have derivatives of all orders. With reference to Figure B.2, close to $x = \hat{x}$ we can approximate the curve $f(x)$ with the line tangent to the curve at $x = \hat{x}$. For a linear approximation, $F(b) \approx F(a) + F'(a)\,(b-a)$ with an error $e(b)$ having $a < c < b$,

$$|e(b)| \leq \frac{\max|F''(c)|}{2}\,(b-a)^2.$$

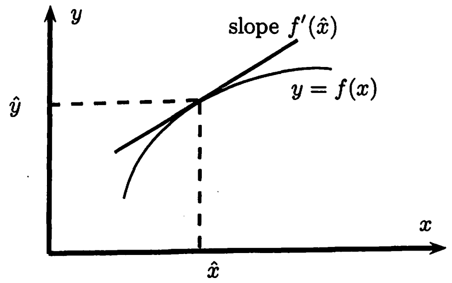

Figure B.2: Diagram showing the first-order Taylor expansion in the vicinity of the point $\hat{x}$.

For a quadratic approximation, $F(b) \approx F(a) + F'(a)(b-a) + \frac{F''(a)}{2}(b-a)^2$ with (again $a < c < b$)

$$|e(b)| \leq \frac{\max|F'''(c)|}{6}|b-a|^3 .$$

Taylor's Theorem can be extended to the two variable case. As illustrated in Figure B.1, close to $(x, y) = (x_1, y_1)$ we can approximate the surface $z = f(x, y)$ with the tangent plane to the surface at $(x, y) = (x_1, y_1)$.

We extend the definition of Taylor's Theorem to functions of two variables in the following way. Let R be some region containing $P(a, b)$. Let $S(a+h, b+k)$ be in R such that the line PS is in R.

Let f have continuous first- and second-order partial derivatives in R. Describe PS parametrically as $x = a + th$, $y = b + tk$, with $0 \leq t \leq 1$. Now we study the values of $f(x, y)$ on PS by considering the function

$$F(t) = f(a + h\,t, b + k\,t).$$

We have

$$F'(t) = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t} = h\,\frac{\partial f}{\partial x} + k\,\frac{\partial f}{\partial y}$$

and F' is continuous and differentiable on $[0, 1]$ because f_x and f_y are (by definition above).

Also

$$\begin{aligned}
F''(t) &= h\,\frac{\partial F'}{\partial x} + k\,\frac{\partial F'}{\partial y} \\
&= h\left(h\,\frac{\partial^2 f}{\partial x^2} + k\,\frac{\partial^2 f}{\partial y\,\partial x}\right) + k\left(h\,\frac{\partial^2 f}{\partial x\,\partial y} + k\,\frac{\partial^2 f}{\partial y^2}\right) \\
&= h^2\,\frac{\partial^2 f}{\partial x^2} + 2\,h\,k\,\frac{\partial^2 f}{\partial x\,\partial y} + k^2\,\frac{\partial^2 f}{\partial y^2} .
\end{aligned}$$

Since F has its first two derivatives continuous on the interval $[0, 1]$ it satisfies the hypothesis for the Taylor series expression in a single variable with $n = 2$, so

$$\begin{aligned}
F(1) &= F(0) + F'(0)(1-0) + F''(c)\,\frac{(1-0)^2}{2!} \qquad (\text{for} \quad c \in [0, 1]) \\
&= F(0) + F'(0) + \frac{1}{2}F''(c) . \qquad \text{(B.5)}
\end{aligned}$$

But $F(1) = f(a+h, b+k)$ and $F(0) = f(a, b)$ by definition. Also $F'(0) = h\,f_x(a, b) + k\,f_y(a, b)$ and $F''(c) = (h^2\,f_{xx} + 2\,h\,k\,f_{xy} + k^2\,f_{yy})$ evaluated at $(a + c\,h, b + c\,k)$.

Now substituting these expressions into (B.5) above

$$f(a+h, b+k) = f(a, b) + h\,f_x(a, b) + k\,f_y(a, b) + (\text{terms of higher order}),$$

and the *tangent-plane approximation* is

$$f(a+h, b+k) \approx f(a, b) + h\,f_x(a, b) + k\,f_y(a, b).$$

B.3 Review of complex numbers

Complex numbers have the form $z = a + ib$ where $i = \sqrt{-1}$. The number a is called the real part of z, $R(z)$, and the number b is called the imaginary part of z, $I(z)$.

Complex numbers can be added, subtracted, multiplied and divided, as shown by the following examples:

- $(7+3i) + (-3+2i) = (7-3) + (3+2)i = 4+5i$
- $(2+3i)(3-i) = 2 \times 3 - 2i + 3 \times 3i - 3i^2 = 6 + 7i - 3 \times (-1) = 9 + 7i$
- $\dfrac{2+i}{1+i} = \dfrac{2+i}{1+i} \times \dfrac{1-i}{1-i} = \dfrac{2-2i+i-i^2}{1-i^2} = \dfrac{2-i+1}{1+1} = \dfrac{3-i}{2}$.

Also useful when dealing with complex numbers is *Euler's identity*,

$$e^{i\theta} = \cos\theta + i\sin\theta, \tag{B.6}$$

which implies that

$$e^{-i\theta} = \cos\theta - i\sin\theta. \tag{B.7}$$

When solving ODEs, the function $e^{-i\theta}$ is often more convenient to deal with than $\cos\theta$ and $\sin\theta$. Furthermore, the identity is useful for obtaining roots of complex numbers. For example, for $\theta = \pi/2$,

$$e^{i\pi/2} = \cos\left(\frac{\pi}{2}\right) + i\sin\left(\frac{\pi}{2}\right) = i.$$

So

$$i^{1/2} = (e^{i\pi/2})^{1/2} = e^{i\pi/4}$$

and from (B.6),

$$i^{1/2} = e^{i\pi/4} = \cos\frac{\pi}{4} + i\sin\frac{\pi}{4} = \frac{1}{\sqrt{2}}(1+i).$$

Note that $i = \cos\left(\frac{3\pi}{2}\right) + i\sin\left(\frac{3\pi}{2}\right)$, since $\cos\left(\frac{3\pi}{2}\right) = 0$ and $\sin\left(\frac{3\pi}{2}\right) = 1$, and thus we also have

$$i^{1/2} = \cos\left(\frac{3\pi}{4}\right) + i\sin\left(\frac{3\pi}{4}\right) = -\frac{1}{\sqrt{2}}(1+i).$$

B.4 Hyperbolic functions

From Euler's identity (equation (B.6), of B.3) we can define $\sin x$ and $\cos x$ as

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i}, \qquad \cos x = \frac{e^{ix} + e^{-ix}}{2}. \tag{B.8}$$

And recall that $\sin^2 x + \cos^2 x = 1$ indicates that in the (x, y)-plane, the point $(\sin x, \cos x)$ lies on the unit circle and the functions $\sin x$ and $\cos x$ are called circular functions. In the same way we can define the *hyperbolic functions*, where $(\sinh x, \cosh x)$ lies on the rectangular hyperbola with equation $\sinh^2 x - \cosh^2 x = 1$.

Formally they are defined in terms of e^x as

$$\sinh x = \frac{e^x - e^{-x}}{2}, \qquad \cosh x = \frac{e^x + e^{-x}}{2}, \tag{B.9}$$

whence

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

These functions are commonly used to simplify expressions of exponentials and thus often appear in the integral solutions of the text.

To get an idea of how they behave they are graphed below using `Maple` in Figures B.3 to B.5.

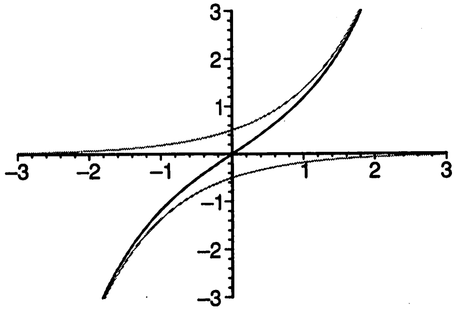

Figure B.3: Graph of sinh x (black) with $e^{-x}/2$ (grey) and $-e^{-x}/2$ (grey).

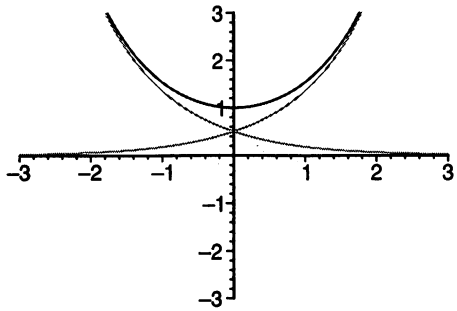

Figure B.4: Graph of cosh x (black) with $e^{-x}/2$ (grey) and $e^{x}/2$ (grey).

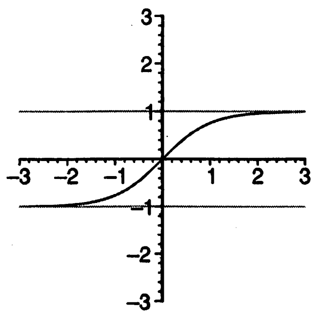

Figure B.5: Graph of tanh x (black) with upper and lower bounds at ± 1 (grey).

B.5 Integration using partial fractions

Integrating a complicated fraction may be simplified by writing it as the sum of simpler fractions. This process is useful in many applications, but in particular for integration where the integral of an original function becomes the sum of integrals, which are much easier to find.

Example 2.2: *Write*

$$\frac{x+4}{x^2-5x+6}$$

as a sum of simpler fractions and hence find its integral.

Solution: *The denominator of the function can be factorised and written as two partial fractions*

$$\frac{x+4}{x^2-5x+6} = \frac{x+4}{(x-2)(x-3)} = \frac{A}{x-2} + \frac{B}{x-3}.$$

Now

$$\frac{A}{x-2} + \frac{B}{x-3} = \frac{A(x-3)+B(x-2)}{(x-3)(x-2)}$$

so that equating the coefficients in the numerator, with those of the original function, gives two simultaneous equations, $A+B=1$ and $-3A-2B=4$. Solving these we have $A=-6$ and $B=7$. Thus the original fraction may be written as the sum of partial fractions

$$\frac{x+4}{x^2-5x+6} = \frac{-6}{x-2} + \frac{7}{x-3}.$$

While integration of the original function is clearly not simple, the latter expression is easy to integrate:

$$\int \frac{-6}{x-2}dx = -6\,\ell\mathrm{n}\,|x-2| + C_1 \quad \text{and} \quad \int \frac{7}{x-3}dx = 7\,\ell\mathrm{n}\,|x-3| + C_2.$$

Thus

$$\int \frac{x+4}{x^2-5x+6}dx = -6\,\ell\mathrm{n}\,|x-2| + 7\,\ell\mathrm{n}\,|x-3| + C_3$$

where $C_3 = C_1 + C_2$.

Example 2.3: *Write the fraction*

$$\frac{1}{x^2-a^2}$$

as a sum of simpler fractions.

Solution: *Firstly, the denominator of the original fraction can be factorised*

$$\frac{1}{x^2-a^2} = \frac{1}{(x-a)(x+a)}$$

and we would like to write this as

$$\frac{A}{x-a} + \frac{B}{x+a}.$$

But

$$\frac{A}{x-a} + \frac{B}{x+a} = \frac{A(x+a)+B(x-a)}{(x-a)(x+a)}$$

from which, by equating the numerators,

$$1 = Ax + Aa + Bx - Ba.$$

Thus $A + B = 0$ (equating the coefficients of x) and $a(A - B) = 1$ (equating the constants) which implies that $A = 1/2a$ and $B = -1/2a$. So the original fraction can be written as

$$\frac{1}{x^2 - a^2} = \frac{1}{2a(x-a)} - \frac{1}{2a(x+a)}.$$

In general, fractions can be separated into partial fractions as follows:

$$\begin{aligned}
\frac{f(x)}{x(x+a)(x-a)} &= \frac{A}{x} + \frac{B}{x+a} + \frac{C}{x-a}, \\
\frac{f(x)}{x^2+a} &= \frac{A}{x} + \frac{Bx+C}{x^2+a}, \\
\frac{f(x)}{x(x-a)^2} &= \frac{A}{x} + \frac{B}{x-a} + \frac{C}{(x-a)^2}.
\end{aligned}$$

Note that when splitting a fraction into partial fractions, the numerator of each partial fraction is a polynomial with degree one less than the denominator. (For further details see, for example, Adams (1995).)

Appendix C

Notes on Maple and MATLAB

C.1 Brief introduction to Maple

`Maple` is an extensive symbolic software package, able to carry out many sophisticated calculations in many areas of mathematics. As an extension to the calculator, it is able to cope with extremely long numbers, up to about 500 000 digits and can solve differential equations symbolically and numerically, graph numerical solutions, do complex algebra, and much more. Since this book is concerned with differential equations, and their solution, not all of which are solvable analytically, `Maple` or `MATLAB` (see Section C.3), or some other equivalent software package, is essential for a complete understanding of the models developed and discussed in this book.

This brief introduction is far from adequate for those who are new to such packages, but it serves to illustrate the `Maple` environment, the manner in which statements and functions operate and how to get help on any topic.

Basics

Below (Figure C.1) is a `Maple` screen dump of some basics in value assignments and algebra. Note the use of semicolons at the end of each statement: a semicolon allows the results to be displayed on the worksheet, while a colon suppresses information to the screen, although it exists and can be accessed at any time by typing the variable name. Note too, that `restart` resets all variables to blanks, and it is good practice to start each independent code segment with this command.

Getting help

At any time, a question mark in front of a command will provide full information on that topic, such as, `?plot` for information, options and examples on plotting functions. Information on the commands required for a variety of functions pertaining to equations, plotting procedures and solution finding, typically used in this text, can also be found under '`Maple` code', in the text index.

'Further information' (below) suggests a website for help on particular topics, as well as suggesting a basic text (of which there are many) for those requiring a detailed introduction to `Maple`.

C.2 Solving differential equations with Maple

This book is concerned with both the numerical and symbolic solution of differential equations, and the plotting of these results. Figure C.2 illustrates how to find the analytical and numerical solutions to a differential equation, as well as how to use the plotting and display functions to graph the results. These routines are typical of those which are required repeatedly throughout the book. Note the use of `with(plots)` in the opening line to access routines in the plotting library of `Maple`. (This code segment is taken from Section 2.5.)

Further information

There are any number of excellent books on introducing `Maple`, such as Holmes et al. (1993). More advanced texts are also available, such as Heck (1996).

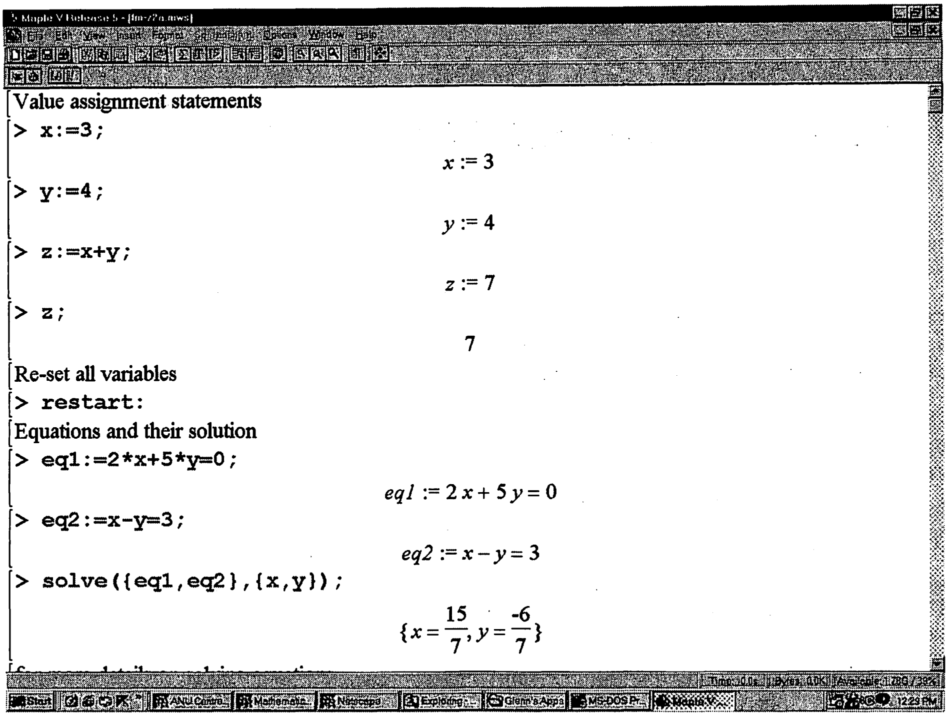

Figure C.1: Maple screen dump illustrating some basic assignment statements, and introducing the Maple environment.

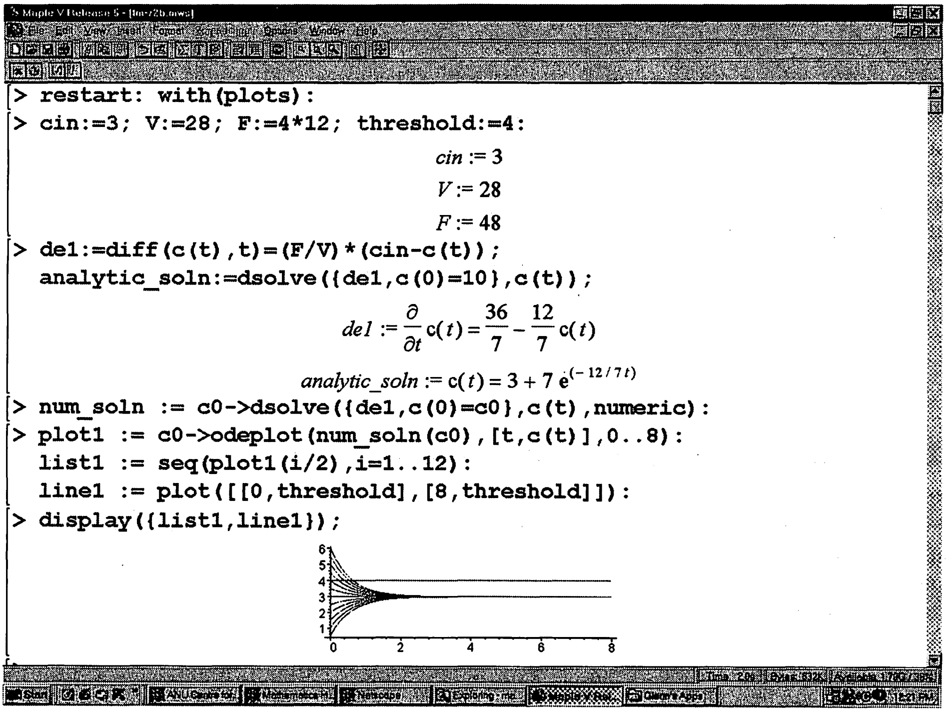

Figure C.2: Maple screen dump illustrating the solution to a differential equation, and a plotting routine.

C.3 Brief introduction to MATLAB

MATLAB is a high-level programming language for mathematics and engineering, in particular. The name MATLAB is short for "MATrix LABoratory". The characterising feature of MATLAB is that it makes dealing with and manipulating matrices both easy and fast. Unlike Maple, it does not do symbolic algebra, but there is a toolbox available (the symbolic toolbox, included with student version of MATLAB) which provides the ability to do symbolic algebra.

Basic commands for vectors and matrices

The basic element (data-type) in MATLAB is the array (or matrix). Square brackets are used to denote arrays, with commas (or blanks) to separate elements and semicolons to separate rows. Thus the following commands are used to store in memory the mathematical constructs

```
a = [1 2 3];
b = [4; 5; 6];
A = [7 -8 9; -4 5 -6];
```

These produce the following mathematical vectors and matrices,

$$a = \begin{bmatrix}1\ 2\ 3\end{bmatrix}, \qquad b = \begin{bmatrix}4\\5\\6\end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix}7 & -8 & 9\\ -4 & 5 & -6\end{bmatrix}.$$

Note that a = [1, 2, 3] is equivalent to a = [1 2 3]. Also note that the semicolon at the end of each statement is used to end the statement, and suppress the output (as opposed to when a semi-colon occurs within the square brackets where it indicates a new line of the array).

Scalar variables are 1×1 matrices but do not require the square brackets, e.g. c = 1 or d = 2.34 are used to store the scalars $c = 1$ and $d = 2.34$ in memory. Note that MATLAB normally regards all variables as real numbers, so c=1 is the same as c=1.0. However, integer valued variables, complex valued variables can also be defined if needed.

The usual addition, subtraction, multiplication and division operations apply to scalar variables. For array variables addition and subtraction work for arrays of the same size and also it is possible to add or subtract a scaler to an array. For multiplication * corresponds to the usual matrix multiplication. Also, .* and ./ and .^ correspond to *element-by-element* operations, e.g. [1, 2, 3] .* [4, 5, 7] results in the array $[1 \times 4, 2 \times 5, 3 \times 7] = [4, 10, 21]$. Similarly [1, 2, 3] .^ 2 produces $[1^2, 2^2, 3^2] = [1, 4, 9]$. Also useful is which is used to extract sub-arrays from arrays, and particularly to extract individual coloums or ropws from a matrix. For example, A(:,2) gives the second column of the matrix $\mathbf{A}$, A(1,:) gives the first row, and A(1:2,1:2) gives the sub-matrix corresponding to rows 1–2 and columns 1–2.

Elements of arrays can be accessed by round brackets. For the above definitions, b(2) has the value 5 and A(2,3) has the value -6. MATLAB also provides the usual constructs for-loops (for sequential operations). In particular, the *for-loop* is used when you know in advance how many sequential operations are needed. The syntax is for k=1:5 which means repeat for a variable k taking values 1 through to 5. Note the use of colon notation similar as was used above for a range of values for the rows or columns in a matrix. As an example of a for-loop, the following code constructs a matrix $\mathbf{A}$, where $A_{ij} = i + 2j$, by sequentially setting each element of the matrix, working along each row (in the inner loop) and then each column in the outer loop. The following is an example of two nested for-loops.

```
N = 10;
for i = 1:N
   for j = 1:N
       A(i,j) = i+2*j;
   end
end
```

MATLAB also provides other constructs, such as a while-loop (for when the number of times through a loop is not known in advance, but is determined by some condition) and the usual if-elseif and logical statements for decisions and branching in programs. The reader is referred to the MATLAB documentation, or some textbooks on MATLAB programming, such as Higham and Higham (2007); Hanselman and Littlefield (2005), for more information on how to use these — they are not required for the examples in this book.

The final important programming construct in MATLAB that we mention here is the MATLAB *function*. Functions are used to help structure complicated code, with each function performing a certain task and with the variables in one function protected from the other functions. A function takes a number of input arguments and returns a number of output arguments (often one output value, which can also be an array, but can be more than one). Some functions can even have a variable number of input and output arguments, e.g. the built-in MATLAB function `max(a, b)` and `\max(a, b, c, d)` which returns the maximum value of the variables in the brackets.

MATLAB provides a substantial number of built-in functions, the simplest being functions to calculate standard mathematical operations, such as `sin(x)` to calculate $\sin(x)$ where `x` can be a scalar variable or an array. It also has functions for certain properties of arrays, such as `length(a)` for returning the length of an array; `size(A)` which returns the size of an array as a 1×2 array; and `eig(A)` for returning the eigenvalues of a square matrix. It is also possible to define one's own functions. The syntax for this is shown in the following code for a function which returns the sum of the cubes of two scalar numbers x and y.

```
function zout = myfcn(x, y)
zout = x^3+y^3;
```

In this the key-word *function* denotes that the following code is a separate function, the variable `zout` is the output variable, which must be assigned a value inside the function. Here `myfcn` is the name of the function. The number `xin` and `yin` here are the inputs for the function, and generally, can be scalars or arrays. To call the function, that is to use the function inside some other function or code, a line of code, for example, is `myfcn(3,2)` which, here, returns the value $3^3 + 2^3 = 34$.

The function can be placed into a separate file, in which case it should have the same name as what you have called the function with the extension 'm', i.e., the name of the file here would be `myfcn.m`. However, it is also sometimes convenient to package up all the functions into one file, provided everything in that file is itself a function. A small modification of the above code for the function `myfcn`, changing the 2nd line to `zout = x.^3 + y.^3`, would allow the function to take arrays as arguments provided the arrays were of the same size.

C.4 Solving differential equations with MATLAB

MATLAB provides functions for solving initial value problems. The standard ODE solver is the `ode45` function. This function needs to be provided with an input argument corresponding to another function for the right-hand-side of the differential equation. It also requires an argument for the range of values of the independent variable on which to solve the differential equation and an argument for the initial condition.

Let us consider the initial value problem

$$\frac{dx}{dt} = \frac{x^2}{10^3} - 3t, \qquad x(0) = 2.5.$$

The entire MATLAB code to solve this problem numerically and plot the results is given in Listing C.1,

Listing C.1: MATLAB code: c_app_solveode.m

```
function c_mainprogram; %define main program as function
trange = [0 2]; % set range of values of t to solve for
x0 = 2.5; % set initial value ofDE
```

```
[tsol, xsol] = ode45(@rhs, trange, x0); %solve the DE
plot(tsol, xsol); %plot the solution

function dxdt = rhs(t, x) % function definition
dxdt = x^2/10^3-3*t; % this defines RHS of dx/dt
```

Let us now examine the various parts of this code. The first line defines the main program as a function. This should have the same name as the filename, i.e. `c_app_solveode.m`. The second line `trange=[0 2]` defines the range of t values to be input as an input argument of `ode45`. The line `x0=2.5` defines a variable to store the initial value $x(0)$. The next line is the one which does the work of solving the differential equation. Here the `ode45` function needs to input another function, called here `rhs` which defines the RHS of the differential equation. Here `@rhs` is a *function handle*, that is, a "pointer" or "handle" to the actual function which is what is needed to make a function an argument of another function. Note that the `ode45` function returns two equal sized arrays, called here `tsol` and `xsol`, which contain the solution $x(t)$ evaluated at a series of time steps contained in `tsol` with the values of the solution contained in the array `xsol`. Finally, the line `plot(tsol, xsol)` creates a figure and plots the array of values `xsol`, corresponding to $x(t)$ on the vertical axis, against the array of values `tsol`, corresponding to t, on the horizontal axis.

Let us consider the second block of code in Listing C.1. This defines the differential equation. This is a separate function, which can be placed in the same file as the main program, as in Listing C.1, or placed in a separate file which has the same name as the function, called here `rhs.m`. Note that the function must always have both the arguments (t, x), in that order, even if the differential equation does not depend explicitly on t. The second line of this block then defines the RHS of the differential equation.

Solving a system of differential equations

A similar approach is taken for solving systems of differential equations. In this case we need to think of the system as a vector-differential equation. Consider

$$\frac{dx}{dt} = 3x - by, \qquad x(0) = 1,$$

$$\frac{dy}{dt} = 3x + ay^2 + 1, \qquad y(0) = 2,$$

where a and b are parameters in the model, i.e.

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{F}(t, \boldsymbol{u}), \qquad \boldsymbol{u} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3x - by \\ 3x + ay^2 + 1 \end{bmatrix}, \qquad \boldsymbol{u}(0) = \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}.$$

The entire code to solve this is given in Listing C.2

Listing C.2: MATLAB code: c_app_systemsolve.m

```
function c_app_systemsolve
global a b;      % define a,b,
trange = [0 1]; % range of values of t
a = 1; b = 3;    % parameter values
u0 = [1; 2];      % define initial values
[tsol, usol] = ode45(@rhs, [0 1], u0); %solve system

xsol = usol(:,1); ysol=usol(:,2); % extract x(t) and y(t)l
plot(tsol, xsol);       % plot x(t)
hold on;                 % cause the plot to not be overwritten
plot(tsol, ysol, 'r:'); % plot y(t)
hold off;                % new plots replace the previous plot

function udot = rhs(t, u)
global a b;              % allows access to variables in main function
x = u(1); y = u(2);    % define x,y as components of vector u
xdot = 3*x - b*y;      % define the RHS of DE for x(t)
ydot = 3*x + a*y + 1; % define the RHS of DE for x(t)
udot = [xdot; ydot]; % assemble the DEs into a column vector
```

The first line, of the first block of code, defines the name of the main program. The second line defines the variables a and b as *global variables*. The global command allows us to define the variables corresponding to the parameter values in the main program, and a similar statement in the function defining the RHS of the differential equation then allows that function access to the values stored in those variables. The third line defines the values of the parameters corresponding to a and b. The fourth line defines the range of t values on which to solve the differential equation. The fifth line of the 1st block defines the initial values. Note the initial values are specified as a column vector.

The first line of the second block of code solves the differential equation which is specified in the third block of code. Finally, we need to plot the solution, but to do this we need to extract the separate solution vectors from the system solution array usol, which is done on the second line. The solution array usol is an $N \times M$ array where N is the number of time steps the solution was calculated at, and N is the number of differential equations in the system. To extract the separate arrays for plotting, a command usol(:,1) takes the 1st column of the array usol, with the colon(:) denoting all values of the column. The remaining lines plot the two solutions $x(t)$, $y(t)$. In this code the hold on; command keeps the plot open so that the second plot command can be included in the same plot without erasing the first command. The third argument of the plot command causes the second plot to be in the colour red with a dotted line.

The third block of code defines the system of differential equations. The first line defines the function, rhs which is referenced by the handle @rhs in the ode45 command in the second block of code. The second line is the global statement which makes the memory containing the parameter values a and b available to the function rhs. The variable u, in the function definition in the 1st line, is an array containing both the variables x and y, as a 2×2 column vector, in this case. So the third line assigns each component of u to the variables x and y. Then in the fourth and fifth lines we define the RHS for the differential equations for $x(t)$ and $y(t)$. Finally, in the sixth line we assign a 2×1 column vector, udot, to the two differential equations.

We could have written the differential equations in terms of $u(1)$ and $u(2)$ instead of defining the variables x and y; however, the use of x and y makes it easier to read and to check that the code for the RHS of the differential equations is correct.

In terms of programming style it is not always a good idea to use global statements, as we have done here. This is because for large, complicated programs there is a risk of forgetting which variables are global and accidently putting the wrong values into them. It is then better to make these variables additional arguments of the rhs function; see the MATLAB documentation for examples of this. For simple programs, as are used in this book, it is convenient and simple to use the global statement.

Code for direction fields

In Chapter 6 plots are produced of the phase-plane using MATLAB. The following MATLAB function can be used to plot direction fields, similar to that produced by Maple. The code is given in Listing C.3.

Listing C.3: MATLAB code: c_dirplot.m

```
function c_dirplot(rhsfcn, xmin, xmax, ymin, ymax, Ngrid)
% function to plot direction fields given the RHS of a system of 2 DEs
% xmin, xmax, ymin, ymax scalar values defining plot region
% Ngrid= number of points on grid for arrows


lflag = 1; %a flag for whether arrows of same length (flag=1) or not

%make a grid
disp(xmin)
x = linspace(xmin, xmax, Ngrid);
y = linspace(ymin, ymax, Ngrid);
[Xm, Ym] = meshgrid(x,y);

%insert the function values at each point of the grid
%into the arrays xd and yd
```

```
for i = 1:Ngrid
    for j=1:Ngrid
        uvec = rhsfcn(0, [x(i), y(j)]);
        if lflag==1
            xd(j,i) = uvec(1)/norm(uvec);
            yd(j,i) = uvec(2)/norm(uvec);
            sfactor=0.6
        else
            xd(j,i) = uvec(1);
            yd(j,i) = uvec(2);
            sfactor=1.0;
        end
    end
end
%use Matlab quiver function to do the dirfield plot
quiver(Xm, Ym, xd, yd, sfactor,'r');
```

Appendix D

Units and scaling

D.1 Scaling differential equations

When dealing with models involving differential equations with several parameters it can be useful for reducing the number of independent parameters in the system by making all the variables dimensionless. This process is called *scaling* the model. The basic idea is to define new variables which are the old variables divided by some constant in the problem which has the same units as the variables.

Example 4.1: *For the population model*

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right), \quad N(0) = N_0,$$

where $N(t)$ is the population at time t, and K is the carrying capacity of the population (measured in number of people), n_0 the initial population (measured in number of people) and r is the intrinsic growth rate measured in years^{-1} (persons/person per year). Determine new variables n and τ which are dimensionless, to replace N and t.

Solution: *The variable N measures population size. There are two possible scales with the same dimensions, the carrying capacity K and the initial population N_0. We will choose the constant K. The variable t measures time (in hours). The constant r is measured in* hours^{-1} *therefore r^{-1} has the same dimensions as t. We can therefore define dimensionless variables*

$$n = \frac{N}{K}, \qquad \tau = \frac{t}{r^{-1}} = rt.$$

We can use the chain rule to express any derivatives in a differential equation in terms of scaled variables. Suppose X and T are dimensioned variables and x and τ are scaled variables, where x_s and y_s are the constants used to scale the variables, so that

$$x = \frac{X}{x_s}, \quad y = \frac{Y}{y_s}, \qquad \text{or} \quad X = x_s x, \quad Y = y_s y.$$

Then we have the following scaling law,

$$\frac{dX}{dY} = \left(\frac{x_0}{y_0}\right)\frac{dx}{dy}. \tag{D.1}$$

Equation D.1 follows directly from the chain rule,

$$\frac{dx}{dy} = \frac{d}{dy}(x_0 X) = x_0\frac{dX}{dy} = \frac{dX}{dY}\frac{dY}{dy} = x_0\frac{dX}{dY}\frac{1}{y_0}.$$

For second-order, and more generally, for nth-order derivatives, the scaling laws are

$$\frac{d^2X}{dY^2} = \left(\frac{x_0}{y_0^2}\right)\frac{d^2x}{dy^2}, \qquad \frac{d^nX}{dY^n} = \left(\frac{x_0}{y_0^n}\right)\frac{d^nx}{dy^n}. \tag{D.2}$$

Example 4.2: *Using the above scaling laws we now scale the model in Example 1.*

Solution: *Write the model*

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right), \quad N(0) = N_0,$$

in terms of the scaled variables $n = N/K$, $\tau = rt$. *The scales are* K *for the variable* N *and* r^{-1} *for the variable* t. *Using* (D.1) *we can write the differential equation as*

$$\frac{K}{r^{-1}}\frac{dn}{dt} = rn\left(1 - \frac{Kn}{K}\right)$$

which simplifies to

$$\frac{dn}{d\tau} = n(1-n).$$

To scale the initial condition $N(0) = N_0$ we write this as $N = n_0$ when $t = 0$. In dimensionless variables the initial condition becomes $Kn = N_0$ when $r^{-1}\tau = 0$ which simplifies to $n = N_0/K$ when $\tau = 0$. Hence we can write the dimensionless differential equation and initial condition as

$$\frac{dn}{dt} = n(1-n), \qquad n(0) = n_0, \quad \text{where} \quad n_0 = \frac{N_0}{K}.$$

This is a simpler form. Where we previously had three parameters, K, r and N_0, now we have just the one independent parameter $n_0 = N_0/K$ (which is a dimensionless combination of N_0 and K).

Example 4.3: *Scale the SIR epidemic model*

$$\frac{dS}{dt} = -\beta SI,$$
$$\frac{dI}{dt} = \beta SI - \gamma I.$$

Solution: *We scale the variables as follows,*

$$s = \frac{S}{s_0}, \qquad i = \frac{I}{s_0}, \qquad \tau = \frac{t}{\gamma^{-1}}$$

and so

$$S = s_0 s, \qquad I = s_0 i, \qquad t = \gamma^{-1}\tau.$$

Using the scaling law (D.1) *the system becomes*

$$\frac{s_0}{\gamma^{-1}}\frac{ds}{d\tau} = -\beta s_0^2 si$$
$$\frac{s_0}{\gamma^{-1}}\frac{di}{d\tau} = \beta s_0^2 si - \gamma s_0 i,$$

which can be simplified to

$$\frac{ds}{d\tau} = -R_0 si$$
$$\frac{di}{d\tau} = R_0 si - i,$$

where $R_0 = \beta s_0/\gamma$ *is a single dimensionless parameter.*

Advantages and disadvantages of scaling

There are some obvious advantages in scaling the equations of a model. Firstly, there is the advantage in having fewer parameters to deal with. On the other hand, scaling a set of equations abstracts the model, to some extent. It can put an additional obstacle (even a minor one) in interpreting the results or graph of a model when it is desired to obtain directly the original quantities. For example, if the important question in a population model is the number of years until the population reaches a certain value, then it is more appropriate to produce a graph of dimensioned quantities. Furthermore, it can sometimes be useful to retain the physical interpretation of the coefficients of various terms, which can be lost when using a scaled model.

Generally speaking, if the parameter values for a model are well known, then a dimensional model may be more appropriate, but if several parameters are not known, and we need to explore the model, then there is a big advantage to scaling the equations.

D.2 SI Units

Table D.1: Fundamental Units of Primary Quantities

Primary Quantity	SI Unit	cgs Unit
mass	kilogram, kg	gram, g
length	metre, m	centimetre, cm
time	second, s	second, s
temperature	Kelvin, K	Kelvin, K

Table D.2: Secondary Units Comprised of Fundamental Units

Quantity	SI Units
density ρ	$\mathrm{kg\,m^{-3}}$
velocity v	$\mathrm{m\,s^{-1}}$
acceleration a	$\mathrm{m\,s^{-2}}$
force F	Newtons $\mathrm{N = kg\,m\,s^{-2}}$
pressure p	Pascal $\mathrm{Pa = N\,m^{-2} = kg\,m^{-1}\,s^{-2}}$
energy E	Joule $\mathrm{J = kg\,m^2\,s^{-2}}$
power $\dot{E}$	Watt $\mathrm{W = J\,s^{-1}}$
heat flux J	$\mathrm{W\,m^{-2}}$
thermal conductivity k	$\mathrm{W\,m^{-1}\,K^{-1}}$
specific heat c	$\mathrm{J\,kg^{-1}\,K^{-1}}$
thermal diffusivity α	$\mathrm{m^2\,s^{-1}}$
Newton cooling coefficient h	$\mathrm{W\,m^{-2}\,K^{-1}}$

References

Adams, R. A. (1995). *Calculus: A Complete Course.* (3rd ed.). Ontario: Addison-Wesley.

Allen, L. (2003). *An Introduction to Stochasitic Processes with Applications to Biology.* N.J.: Prentice-Hall.

Anderson, A. (1996). Was *rattus exulans* in New Zealand 2000 years ago? AMS radiocarbon ages from the Shag River Mouth. *Archaeology in Oceania. 31*, 178–184.

Anderson, R. M. and R. M. May (1991). *Infectious Diseases of Humans.* Oxford.: Oxford University Press.

Banks, R. (1994). *Growth and Diffusion Phenomena: Mathematical Frameworks and Applications.* Berlin: Springer-Verlag.

Barnes, B., H. Sidhu and D. Gordon (2007). Host gatro-intestinal dynamics and the frequency of antimicrobial production by escherichia coli. *Microbiology 153*, 2823–2827.

Barro, R. J. (1997). *Determinants of Economic Growth: A Cross-Country Empirical Study.* Cambridge, MA.: MIT Press.

Beck, M., A. Jakeman and M. McAleer (1993). *Construction and Evaluation of Models of Environmental Systems*, Chapter 1, pp. 3–35. New York: Wiley.

Beddington, J., C. Free and J. Lawton (1978). Characteristics of successful natural enemies in models of biological control of insect pests. *Nature 273*, 513–519.

Begon, M., J. Harper and C. Townsend (1990). *Ecology, Individuals, Populations and Communities* (2nd ed.). Oxford: Blackwell.

Borelli, R. and C. Coleman (1996). *Differential Equations; A Modelling Perspective.* New York: Wiley.

Brauer, F. and C. Castillo-Chàvez (2001). *Mathematical Models in Population Biology and Epidemiology.* NY: Springer-Verlag.

Braun, M. (1979). *Differential Equations and Their Applications* (2nd ed.). Berlin: Springer-Verlag.

Burgess, J. and L. Olive (1975). Bacterial pollution in lake burley griffin. *Water (Australian Water and Wastewater Association) 2*, 17–19.

Caulkins, J., R. Marrett and A. Yates (1985). Peruvian anchovy population dynamics. *U.M.A.P. Journal 6*(3), 1–23.

Costantino, R., J. Cushing, B. Dennis and R. Desharnis (1995). Experimentally induced transitions in the dynamic behaviour of insect populations. *Nature 375*, 227–375.

Costantino, R., R. Desharnais, J. Cushing and B. Dennis (1997). Chaotic dynamics in an insect population. *Science 275*, 389–391.

Cramer, N. F. and R. May (1972). Interspecific competition predation and species diversity: A comment. *Journal of Theoretical Biology 34*, 289–293.

Cyrix (1998). Technical documentation for Cyrix 6x86-P-166 processor.

Daley, D. and J. Gani (1999). *Epidemic Modelling. An Introduction.* Cambridge: Cambridge University Press.

Davies, P. (1994). *War of the Mines: Cambodia, Landmines and the Impoverishment of a Nation.* London: Pluto Press.

Dekker, H. (1975). A simple mathematical model of rodent population cycles. *Journal of Mathematical Biology 2*, 57–67.

Diekmann, O. and J. A. P. Heesterbeek (2000). *Mathematical Epidemiology of Infectious Disease: Model Building, Analysis and Interpretation.* Hokoben, N.J.: Wiley.

Dodd, A. (1940). The biological campaign against prickly-pear. Technical report, Brisbane Government Printer, Australia.

Edelstein-Keshet, L. (1988). *Mathematical Models in Biology.* New York: Random House.

Feichtinger, G., C. V. Forstand and C. Piccardi (1996). A nonlinear dynamical model for the dynastic cycle. *Chaos, Solitons and Fractals 7*, 257–271.

Frank, S. A. (1994). Spatial polymorphism of bacteriocins and other allelopathic traits. *Evolution Ecology 8*, 369–386.

Fulford, G. R. and P. Broadbridge (2000). *Industrial Mathematics: Case Studies in Heat and Mass Transport.* Cambridge: Cambridge University Press.

Fulford, G. R., P. Forrester and A. Jones (1997). *Modelling with Differential and Difference Equations.* Cambridge: Cambridge University Press.

Fulford, G. R., M. G. Roberts and J. A. P. Heesterbeek (2002). The metapopulation dynamics of an infectious disease: Tuberculosis in possums. *Theoretical Population Biology 61*, 15–29.

Gani, J. (1972). Model-building in probability and statistics. In O. Sheynin (Ed.), *The Rules of the Game*, pp. 72–84. London: Tavistock Publications.

Gani, J. (1980). The role of mathematics in society. *Mathematical Scientist 5*, 67–77.

Gordon, D. M. and M. A. Riley (1999). A theoretical and empirical investigation of the invasion dynamics of colicinogeny. *Microbiology 145*, 655–661.

Gotelli, N. J. (1995). *A Primer of Ecology.* Sunderland, MA: Sinauer Associates.

Grenfell, B. and A. Dobson (1995). *Ecology of Infectious Diseases in Natural Populations.* Cambridge: Cambridge University Press.

Hanselman, D. and B. Littlefield (2005). *Mastering MATLAB 7.* NJ: Pearson.

Hassell, M. (1978). *The Dynamics of Arthropod Predator-Prey Systems.* Princeton, N.J.: Princeton University Press.

Hassell, M. P. (1976). *The Dynamics of Competition and Predation.* London: Edward Arnold.

Hearn, C. (1998). Private communication.

Heck, A. (1996). *An Introduction to Maple* (2nd ed.). New York: Springer-Verlag.

Higham, D. J. and N. J. Higham (2007). *MATLAB Guide* (2nd ed.). Philadelphia: SIAM.

Holman, J. (1981). *Heat Transfer.* New York: McGraw-Hill.

Holmes, M., J. Ecker, W. Boyce and W. Siegmann (1993). *Exploring Calculus with Maple.* New York: Addison-Wesley.

Hurewicz, W. (1990). *Lectures on Ordinary Differential Equations.* New York: Dover.

Jones, J. C. (1993). *Combustion Science. Principals and Practice.* Brisbane: Milennium Books.

Keeling, M. J. and P. Rohani (2008). *Modelling Infectious Diseases in Humans and Animals.* N.J.: Princeton University Press.

Keeton, W. T. (1972). *Biological Science* (2nd ed.). New York: W W Norton.

Kermack, W. and A. McKendrick (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society. A 115*, 700–721.

Kerr, B., M. A. Riley, M. W. Feldman and B. J. Bohannan (2002). Local dispersal promotes biodiversity in a real-life game of rock-paper-scissors. *Nature 418*, 171–174.

Kincaid, D. and W. Cheney (1991). *Numerical Analysis.* Belmont, CA.: Brooks Cole.

Kirkup, B. C. and M. A. Riley (2004). Antibiotic-medicated antagonism leads to a bacterial game of rock-paper-scissors in vivo. *Nature 428*, 412–414.

Kormondy, E. K. (1976). *Concepts of Ecology* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Krebs, C., M. Gaines, B. Keller, J. Myers and R. Tamarin (1973). Population cycles in small rodents. *Science 179*, 35–41.

Lay, D. C. (1994). *Linear Algebra and its Applications.* Reading, MA: Addison-Wesley.

Levin, B. R. (1988). Frequency dependent selection in bacterial populations. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 319*, 2823–2827.

Lewin, R. (1983a). Predators and hurricanes change ecology. *Science 221*, 737–740.

Lewin, R. (1983b). Santa rosalia was a goat. *Science 221*, 636–639.

Lignola, P. and F. D. Maio (1990). Some remarks on modelling CSTR combustion processes. *Combustion and Flame 80*, 256–263.

Louie, K., M. Roberts and G. Wake (1993). Thresholds and stability analysis of models for the spacial spread of a fatal disease. *IMA Journal of Mathematics Applied to Medicine and Biology 10*, 207–226.

Loyn, R. H., R. G. Runnalls, G. Y. Forward and J. Tyers (1983, Sep). Territorial bell miners and other birds affecting populations of insect prey. *Science 4618*, 1411–1413.

Lucas, R. (1988). On the mechanics of economic development. *Journal of Monetary Economics 22*, 3–42.

MacArthur, R. and J. Connell (1966). *The Biology of Populations.* New York: Wiley.

May, R. (1981). *Theoretical Ecology: Principles and Applications* (2nd ed.). Oxford: Blackwell Scientific Publications.

Mesterton-Gibbons, M. (1989). *A Concrete Approach to Mathematical Modelling.* Reading, MA: Addison-Wesley.

Michaelis, L. and M. Menten (1913). Die kinetik der invertinwirkung. *Biochemische Zeitschrift 49*, 333–369.

Microsoft (1995). Microsoft encarta.

Miller, T. (1995). Mathematical model of heat conduction in soil for landmine detection. Technical Report DMS-C95/95, CSIRO Division of Mathematics and Statistics.

Monro, J. (1967). The exploitation and conservation of resources by populations of insects. *Journal of Animal Ecology 36*, 531–547.

Murray, J. (1990). *Mathematical Biology.* New York: Springer-Verlag.

Myers, J. and C. Krebs (1974). Population cycles in rodents. *Scientific American 230*, 38–46.

Oakley, R. and C. Ksir (1993). *Drugs, Society and Human Behaviour* (6th ed.). N.Y.: McGraw-Hill.

Parrish, J. and S. Saila (1970). Interspecific competition, predation and species diversity. *Journal of Theoretical Biology 27*, 207–220.

Przemieniecki, J. (1994). *Mathematical Methods in Defence Analysis.* Washington DC: American Institute of Aeronautics and Astronautics.

Quammen, D. (1997). *The Song of the Dodo: Island Biogeography in an Age of Extinctions.* London: Pimlico.

Renshaw, E. (1991). *Modelling Biological Populations in Space and Time.* Cambridge: Cambridge University Press.

Rivers, C., G. Wake and X. Chen (1996). The role of milk powder in the spontaneous ignition of moist milk powder. *Mathematics and Engineering in Industry 6*, 1–14.

Roberts, M. (1992). The dynamics and control of bovine tuberculosis in possums. *IMA Mathematics Applied to Medicine and Biology 9*, 19–28.

Roberts, M. (1996). The dynamics of bovine tuberculosis in possum populations, and its eradication by culling or vaccination. *Journal of Animal Ecology 65*, 451–464.

Roberts, M. G. and J. A. P. Heesterbeek (1993). Bluff your way in epidemic models. *Trends in Microbiology 1*, 343–348.

Roberts, M. G. and M. I. Tobias (2000). Predicting and preventing measles epidemics in New Zealand. *Epidemiology and Infection 124*, 279–287.

Rolls, E. (1969). *They All Ran Wild: A Story of Pests on the Land in Australia.* Melbourne: Angus and Robertson.

Romer, P. (1986). Increasing returns and long run growth. *Journal of Political Economy 94*, 1002–1037.

Sansgiry, P. and C. Edwards (1996). A home heating model for calculus students. *College Mathematics Journal 27*(5), 394–397.

SBS (1998). *SBS World Guide.* Sydney: Australian Broadcasting Commission.

Smedley, S. I. and G. C. Wake (1987). Spontaneous ignition: Assessment of cause. *Annual Meeting of the Institute of Loss Adjusters of NZ. Palmerston North 10-12 June*, 1–14.

Solow, R. M. (1956). A contribution to the theory of economic growth. *Quarterly Journal of Economic Growth 70*, 65–94.

Stewart, I. (1989). Portraits of chaos. *New Scientist 1689*(4 Nov), 22–27.

Swan, T. W. (1956). Economic growth and capital accumulation. *Economic Record 32*, 334–361.

Taylor, J. G. (1980). *Force-on-Force Attrition Modelling.* Virginia: Operations Research Society of America.

Tung, K. K. (2007). *Topics in Mathematical Modeling.* Princeton: Princeton University Press.

van de Koppel, J. Huisman, J. van der Wal and H. Olff (1996). Patterns of herbivory along a productivity gradient: An empirical and theoretical investigation. *Ecology 77*(3), 736–745.

Vivaldi, F. (1989). An experiment with mathematics. *New Scientist 1689*(28 Oct), 30–33.

Wake, G. and M. Roberts (1995). Percy possum plunders. *CODEE Winter*, 13–14.

Wilmott, P. (1998). *Derivatives. The Theory and Practice of Financial Engineering.* Chichester: Wiley.

Index