12. Teaching Engineering Using Mathematical Packages

John D. Fenton
Department of Mechanical Engineering, Monash University, Clayton, VIC 3168, Australia

Objectives

This chapter demonstrates the use of two mathematical packages which have extensive symbolic manipulation and graphical capabilities. The two packages described are:

- MAPLE
- MATLAB

At the end of the exercises, the reader should be able to solve simple problems with these packages.

Introduction

Traditionally, engineering computations have involved the use of high-level languages such as FORTRAN, BASIC, PASCAL or C. These are mainly numerical languages, and programming them requires a lot of detailed fundamental work. Modern specialist software may be of a considerably higher level, with the provision of many subroutines, specialist software and graphics routines. In this module we examine the use of the two packages described above, as an example of the use of these higher-level languages. Such is the provision of this software that many of the instructions and assistance are best provided on-line. This manual segment will describe some of the fundamental procedures and syntax so that the users can experiment with the demonstration modules that are provided.

MAPLE

About MAPLE

Maple is a language for symbolic mathematical calculation. It does not need numerical information, and is at its most powerful when it is dealing with symbols. It has a large number of useful features, and it is best explored live, possibly following the Tutorial introduction, described as follows. These notes are written in a way that allows the reader to work through them to develop expertise with the program.

Syntax

Maple operates very much like a word processor. One can go back to previous statements, change them, re-evaluate them *etc*. However, typing <Enter> evaluates the block of commands; to insert a new line it is necessary to type <Shift Enter>.

Tutorial – on-line introduction to Maple

Start Maple by double clicking on the Maple icon in the Program Manager. Then type: *tutorial*(); (note that Maple statements are terminated by a semi colon (;), as in C and Pascal).

The function *tutorial*() takes the user through a beginning level, on-line tutorial for Maple. This tutorial is not meant to be a replacement for the on-line help files, but is instead meant to offer a quick method for getting started with Maple. There are 14 chapters in the on-line tutorial. The command *tutorial*(*n*) skips the standard welcoming text for the tutorial and starts the user at chapter *n*. Most chapters have question and answer sections, and there are quizzes included. There is a main menu, from which the reader can begin any of the chapters, which can be accessed from any break in the text.

Quick tour – an overview for engineers and scientists

This takes the user through a tour of the capabilities of MAPLE which includes functions that are of particular interest in more applied disciplines, including special functions, transforms, and statistics. To begin the tour, choose Open from the File menu and make the appropriate selection. Instructions on working through the tours are included in the worksheet.

The way to execute each of the statements is simply to type <Enter> when the cursor is on that line. This is true in all MAPLE sessions. Table 1 shows some of the more useful commands.

Table 1: Fundamental MAPLE commands.

Command	Description
help;	General help menu
help 'topic'; **OR ?'topic';**	Information on a specific topic, *e.g.* help step;
restart;	Clears all the variables stored on the stack
quit;	Leave MAPLE
<Up/Down Arrows>	Moves to previous or later commands
<Enter>	Evaluates commands
<Shift Enter>	Inserts new line

Personal experience

Now it is time to experience the power of such a program. Maple statements are input after the > prompt. Every Maple statement must end with a semicolon ; (or a colon : if the result is not to be printed). If one has forgotten to type ; or : there will be no response until the next command is typed. Until Maple receives a (semi)colon it will neither process the statement nor accept a new statement.

Maple can be used rather like a modern word processor. The cursor can be placed anywhere in the file and material may be deleted, copied, or executed.

Note that the Maple assignment operator is :=, not just = (which is reserved for when an equation is set up). Hence, a typical Maple command line might look like:

```
> y := theta*x*(x+1)*(x-1);
```

The response to this is properly typeset mathematics:

$$y := \theta\, x\left(x+1\right)\left(x-1\right),$$

which can be copied and placed in a document, just as was done here. Note that the names of Greek letters are automatically converted to those symbols.

The ditto symbol " (double quotes) is used to indicate the previous expression in a Maple session. For example, typing

```
> 2*";
```

gives

$$2\,\theta\, x\left(x+1\right)\left(x-1\right),$$

Some important Maple functions are *expand*, *simplify*, and *normal*, which are useful for simplifying expressions; *evalf*, for evaluating to floating-point; *solve*, for solving equations; *int* and *diff*, for integration and differentiation; *series*, for Taylor or Laurent series; and *plot*, for plotting functions. Other useful Maple functions are: *array*, *coeff*, *collect*, *convert*, *degree*, *denom*, *evalc*, *ifactor*, *limit*, *map*, *normal*, *numer*, *op*, *product*, *simplify*, *subs*, *sum*, *table*, and *type*. These commands are the more commonly used ones; this is by no means a complete list of Maple functions. Maple also has many mathematical functions built-in, such as *sin*, *cos*, *tan*, *exp*, *ln*, *GAMMA*, *Zeta*, and *binomial*.

For example,

```
> expand(");
```

gives

$$2\,\theta\, x^3 - 2\,\theta\, x\,,$$

and then

```
> diff(",x);
```

differentiates that with respect to x:

$$6\,\theta\, x^2 - 2\,\theta\,.$$

The command *factor* is useful:

```
> factor(");
```

gives

$$2\,\theta\left(3x^2-1\right).$$

While these commands all seem simple enough, Maple's real power is shown when it operates on complicated expressions, including quantities like series, matrices, systems of equations and so on, which may already have been experienced in the above demonstration programs.

For engineering problem solving, Maple has a number of useful capabilities. The *plot* command needs a minimum of information to be provided. The software provides all necessary scalings *etc*, which can of course, be over-ridden. For example, the command

```
> plot(x^2,x);
```

creates a separate screen, which can be copied if necessary, and which looks like Figure 1.

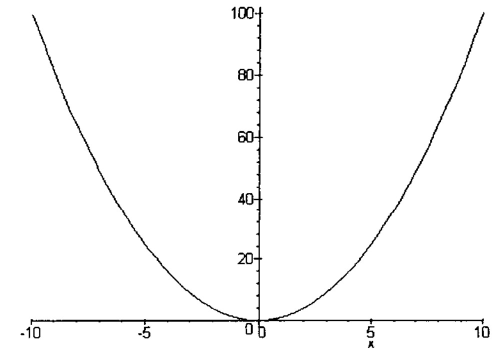


Figure 1: Typical screen output from *plot* command.

The command *solve* can be used to solve a simple equation or a number of simultaneous equations. For example, remembering that *y* has been defined previously, the command

```
> solve(y=0,x);
```

(note the '=' sign, defining an equation), gives all three solutions of the cubic:

$$0, -1, 1\ .$$

The syntax $x = a..b$ is very common, requiring that the variable x take values between a and b. If x is an integer they will be integer values. For example, to generate a sum:

```
> sum(A[k]*z^k,k=0..5);
```

gives

$$A_0 + A_1\ z + A_2\ z^2 + A_3\ z^3 + A_4\ z^4 + A_5\ z^5\ .$$

Note that A[k] is printed as *A* with a subscript the numerical value of *k*.

A useful command is *subs*, which enables substitutions as specified. For example:

```
> subs(A=B,z=Z,");
```

gives

$$B_0 + B_1\ z + B_2\ z^2 + B_3\ z^3 + B_4\ z^4 + B_5\ z^5\ .$$

Series operations are very powerful in Maple. For example, to generate a Taylor series for tan(x) about x=0 with neglected terms of order 6:

```
> f:=series(tan(x),x=0,6);
```

gives

$$f := x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + O\left(x^6\right).$$

An often better form of approximation is the use of Padé approximants, using a rational function with series in numerator and denominator. For example the Padé approximant to tan(x) with cubics top and bottom is obtained by (note the package *numapprox* has to be loaded first):

```
> with(numapprox): g := pade(tan(x), x, [3,3]);
```

which gives

$$\frac{-\frac{1}{15}x^3 + x}{1 - \frac{2}{5}x^2}\ .$$

The various approximations can be compared by first converting f to a polynomial and setting up the plot:

```
> f:=convert(f,polynom);
> p1 := plot(f,x=0..Pi/2,y=0..10,linestyle=0):
```

Then with similar commands for p2 for the plot of the function itself and p3 for the plot of g:

```
> with(plots):
> plots[display]({p1,p2,p3}, title =
     `Approximations to tan(x)`, axes=BOXED);
```

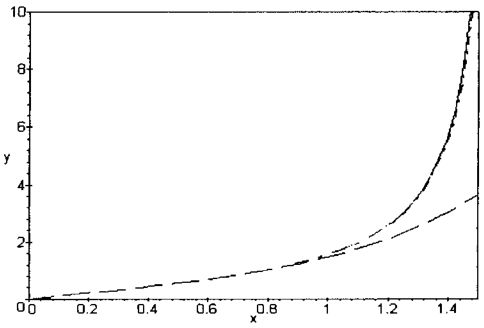


Figure 2: Approximations to tan *x*.

The graphs are shown in Figure 2, which demonstrates the ability of the Padé approximant to handle the singular behaviour of the tan function.

The equation solving properties are very useful. As a small example, consider the set of two simultaneous equations set up and solved by:

```
> Equations:={x+y=2,x-y=0}:
> Unknowns:={x,y}:
> Solution:=solve(Equations,Unknowns);
```

gives

$$\text{Solution} := \left\{y = 1, x = 1\right\}.$$

Example packages

There are a number of sample work sheets which the user can work through to explore the use of MAPLE. Some of these are set out below. Use File|Open to load each of them in turn.

INTRO.MS – An Introduction Worksheet

This is a very useful introduction to some of the commands which can be used automatically to generate series, to expand, to factorise and to simplify.

ODE.MS – Solving Differential Equations with Maple

This worksheet demonstrates Maple's sophisticated differential equation solver. Here are some more examples of equations that would appear in a first course on differential equations. Maple can also solve systems of differential equations. In some cases, we can specify an algorithm to be used in solving the equation. For example, Maple can be asked to use an algorithm to solve the equation, such as Laplace transforms. Maple also has an option for finding numerical solutions. Maple returns a procedure that, when evaluated at a point, yields the solution's value, and the value of all derivatives up to one less than the order of the ODE. The default algorithm used is a Fehlberg fourth-fifth order Runge-Kutta method. We can plot the result using the function *odeplot*. For example, to solve the ordinary differential equation

$$\frac{d^2y}{dt^2} + y = 0,$$

subject to the boundary conditions $y(0) = 0$ and $dy/dt(0) = 1$, the commands are

```
> DE:=diff(y(t),t$2)+y(t);
> Soln:=dsolve({DE,y(0)=0,D(y)(0)=1}, y(t));
```

and the result is the analytical solution:

$$y(t) = \sin(t).$$

If the right hand side is then assigned to the left by

```
> assign(Soln);
```

then the solution can be plotted by the command

```
> plot(y(t),t=0..6);
```

which gives the graph shown in Figure 3.

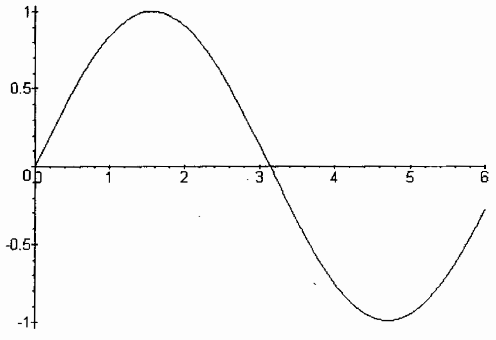


Figure 3: Plot of solution of differential equation.

If the equation does not have an analytical solution, then it can be solved numerically. For example, consider the solution of the nonlinear equation

$$\frac{d^2y}{dt^2} + y^2 = 0,$$

subject to the same initial conditions. The commands are:

```
> DE:=diff(y(t),t$2)+y(t)^2;
> Solution :=
dsolve({DE,y(0)=0,D(y)(0)=1},y(t),numeric):
> with(plots): odeplot(Solution,{t,y},0..6);
```

which gives as output the plot shown in Figure 4.

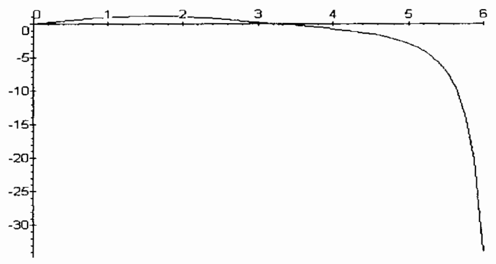


Figure 4: Plot of numerical solution.

The program generates internal functions so that the numerical solution can be used. For example, typing

```
> Solution(1);
```

gives the result for the procedure we have just defined as "Solution" at the point $t = 1$:

$$\left[t=1,\ y(t)=.9204757302791347, \frac{\partial}{\partial t}y(t) = .6928700976193584 \right].$$

Graphics.MS – Graphics with Maple

This worksheet is a demonstration of Maple's graphics capabilities. Place the cursor on the first input line, and step through the examples in order, by pressing <Enter> at each Maple command.

Warning: This worksheet is quite computationally intensive. After each plot, close the plot window before executing the next command by double clicking on the left corner.

Some of the results are quite spectacular. Consider the power of representing the function

$$f := x\, e^{(-x^2 - y^2)},$$

as shown by Figure 5.

Other useful engineering worksheets include:

INT.MS – Shows the power of Maple's integration capabilities. For example, consider the function:

$$\frac{x^3 - x}{x^5 + x^4 + x^3 - x^2 - x - 1}.$$

The command *int(f,x)* gives the integral:

$$-\frac{1}{3}\frac{2x+1}{x^2+x+1} + \frac{2}{9}\sqrt{3}\arctan\left(\frac{1}{3}(2x+1)\sqrt{3}\right).$$

Maple is also capable of computing improper integrals – often symbolically. For example:

```
>Int( exp(-t^2), t=-infinity..infinity );
```

gives

$$\int_{-\infty}^{\infty} e^{-t^2}\, dt,$$

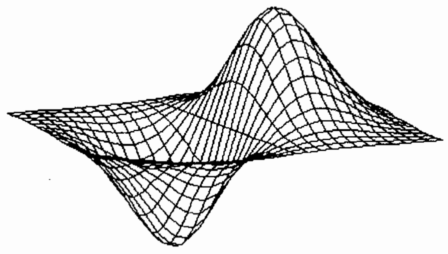
then:

```
> value(");
```

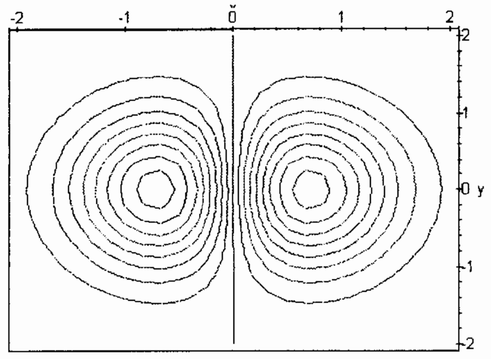
gives: $\sqrt{\pi}$. Note the use of the *ditto* symbol, which always means *the result of the previous calculation.*

LINALG.MS – Linear Algebra with Maple

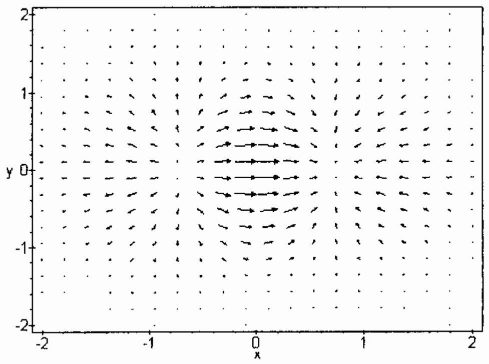
Maple has a library package for linear algebra. With it, we can compute eigenvalues and eigenvectors, determinants of matrices, perform row and column operations. Many of these functions can be used on symbolic matrices. For example, define matrix A:



(a)



(b)



(c)

Figure 5: Graphs of $f = x\, e^{-(x^2+y^2)}$: (a) surface plot, (b) contour plot, (c) gradient vector plot.

$$A := \begin{bmatrix} 1 & x & y \\ 0 & 1 & z \\ 0 & 0 & 1 \end{bmatrix}.$$

```
> A:=matrix([[1,x,y],[0,1,z],[0,0,1]]);
```

Maple uses its &* operator to denote non-commutative matrix multiplication. We must use the function evalm to evaluate a matrix expression:

```
> evalm(A&*A);
```

$$\begin{bmatrix} 1 & 2x & 2y + xz \\ 0 & 1 & 2z \\ 0 & 0 & 1 \end{bmatrix}.$$

Maple can compute determinants, *e.g.* det(A); or, compute the inverse of a matrix:

```
> inverse(A);
```

$$\begin{bmatrix} 1 & -x & xz - y \\ 0 & 1 & -z \\ 0 & 0 & 1 \end{bmatrix}.$$

OUTPUT.MS – Language Output with Maple

Maple has several particularly useful commands for converting Maple output into other languages and forms. This worksheet highlights transformations to Fortran, C, LaTeX, and Troff Eqn printing. Of course, the output from ordinary Maple commands can be simply copied to a Windows word processor.

Consider the Maple expression

```
> e1:=exp(-x^2) * (a*cos(y+t)+b*sin(y-t));
```

which gives

$$e1 := e^{(-x^2)}\left(a\cos\left(y+t\right) + b\sin\left(y-t\right)\right).$$

Now,

```
> fortran(e1);
```

gives the text:

```
t0 = exp(-x**2)*(a*cos(y+t)+b*sin(y-t))
```

or, for the C language, "C(e1)" gives

```
t0 = exp(-x*x)*(a*cos(y+t)+b*sin(y-t));
```

These can be directly copied into the necessary programs. In this case, the advantage did not seem huge. Consider, however, the instructions necessary for, say, the inverse of a 4x4 matrix! For word-processing of equations, the Maple library LaTeX and eqn conversion routines serve two useful purposes. They provide foremost a way of

presenting the results of a Maple session, but they also work well as general purpose math interpreters for making documents. For example:

```
> latex(e1);
```

gives the text

```
{e^{-{x}^{2}}}\left (a\cos(y+t)+b\sin(y-t)\right ),
```

which can be read into a TeX document.

Other useful worksheets include **SERIES.MS**, **SOLVE.MS**, **SOLVE2.MS**, **SPECFUNC.MS**, and **SYMBOLIC.MS**. The student should work his or her way through these.

Overall, the use of symbolic manipulation and the power of a number of inbuilt mathematical operations means that less attention has to be paid to mathematical detail and more can be spent on the physical interpretation of the results obtained.

MATLAB

The first version of MATLAB was written at the University of New Mexico and Stanford University in the late 1970s, intended for use in courses in matrix theory, linear algebra and numerical analysis. The authors had been involved in the development of FORTRAN subroutine packages for matrix manipulation, and sought to enable students to use such packages without writing FORTRAN programs.

Today, MATLAB has been extended far beyond the original *Matrix Laboratory*. It is now an interactive system and programming language for general scientific and technical computation. Its basic data element is a matrix that does not require dimensioning. This allows relatively simple solution of many numeric problems.

The manufacturers of MATLAB offer a series of *Application Toolboxes* that contain sets of MATLAB functions designed for specific applications, including digital signal processing, automatic control system design, nonlinear simulation, parametric modelling, optimisation and spline analysis.

The software has some good introductions and demonstration modules. For example, typing *intro* leads one through a series of demonstrations of elementary commands and their syntax. After doing this it will be necessary to resize the windows of the Matlab Command Window and the Figure Window, so that the comments and the output can both be read. (See the language intro demo obtained by typing *expo* for a brief overview of the MATLAB language and M-files.)

For more help on directory/topic, type *help topic*. Commands to get started include : *intro*, *demo*, *help help*. Commands for more information include:

help, *whatsnew*, *info*, *subscribe* and others briefly described in Table 2.

Table 2: Basic MATLAB commands.

General purpose commands.	
help	On-line documentation.
doc	Load hypertext documentation.
what	Directory listing of M-, MAT- and MEX-files.
type	List M-file.
lookfor	Keyword search through the HELP entries.
which	Locate functions and files.
path	Control MATLAB's search path.
Managing variables and the workspace.	
who	List current variables.
whos	List current variables, long form.
load	Retrieve variables from disk.
save	Save workspace variables to disk.
clear	Clear variables and functions from memory.
Quitting MATLAB.	
quit	Terminate MATLAB.

The results from typing *intro* include a demonstration of how, unless advised otherwise, everything is interpreted as an array. MATLAB requires no special handling of vector or matrix math.

For example

```
>> a = [1 2 3 4 6 4 3 4 5]
>> b = a+2
```

adds 2 to each element of our vector, with the result lines:

```
a = 1   2   3   4   6   4   3   4   5
b = 3   4   5   6   8   6   5   6   7
```

Creating graphs in MATLAB is made simple. Here we plot the result of the vector addition with grid lines, shown in Figure 6.

```
>> plot(b)
>> grid
```

Creating a matrix is as easy as making a vector, using semicolons (;) to separate the rows of a matrix.

```
» A = [ 1 2 0 ; 2 5 -1 ; 4 10 -1]
```

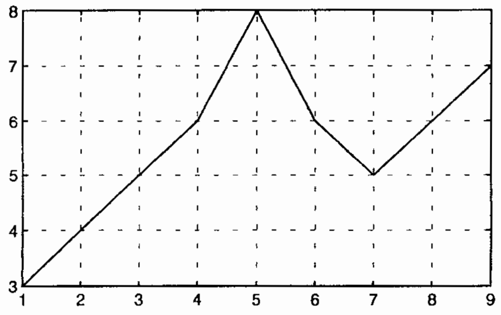


Figure 6: Typical screen output from *plot* command.

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 5 & -1 \\ 4 & 10 & -1 \end{bmatrix}$$

Now we calculate the inverse of the matrix:

```
» X=inv(A)
```

$$A = \begin{bmatrix} 5 & 2 & -2 \\ -2 & -1 & 1 \\ 0 & -2 & 1 \end{bmatrix}$$

and then illustrate the fact that a matrix times its inverse is the identity matrix.

```
» A*X
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

MATLAB has functions for nearly every type of common matrix calculation. There are functions to obtain eigenvalues:

```
» eig(A)
```

$$\text{ans} = \begin{bmatrix} 3.7321 \\ 0.2679 \\ 1.0000 \end{bmatrix}$$

Or, one can obtain the characteristic polynomial of a matrix A, det(lambda*I – A), for which one types

```
» p=round(poly(A))
```

giving: p = 1 -5 5 -1

The *poly* function generates a vector containing the coefficients of the characteristic polynomial. We can easily find the roots of a polynomial using the *roots* function:

```
» roots(p)
    ans =   3.7321
            0.2679
            1.0000
```

which are the eigenvalues of the original matrix.

MATLAB has many applications beyond just matrix computation. At any time, we can get a listing of the variables we have stored in memory using the *who* or *whos* command. The value of a particular variable can be had by typing its name. More than one statement may be placed on a single line by separating each statement with commas or semicolons:

```
» sqrt(-1), log(0)
ans =         0 + 1.0000i
Warning: Log of zero
ans =         -Inf
```

If a variable is not assigned to store the result of an operation, the result is stored in a temporary variable called *ans*. In this case, since we separated the statements with commas, the result of each operation was echoed to the screen. As can be seen, MATLAB easily deals with complex and infinite numbers in calculations.

MATLAB has functions which make it ideal as a signal processing tool. For more details, see the Expo demos of the Signal Processing Toolbox.

The directories shown in Table 3 contain specialised functions which can be used. A guide or summary to them can be obtained by typing *eg* help datafun, without bothering to specify the directory.

An overview of the language

Complex numbers and matrices

Complex numbers are allowed in all operations and functions in Matlab. They are entered using the special functions i or j, for example

```
>> z = 3 + 4*i
```

Functions

Matlab has well over 200 functions and many more in specialist Toolboxes.

Table 3: Specialised MATLAB function directories.

Directory	**Topic**
datafun	Data analysis and Fourier transform functions.
elfun	Elementary math functions.
elmat	Elementary matrices and matrix manipulation.
funfun	Function functions-nonlinear numerical methods.
general	General purpose commands.
color	Color control and lighting model functions.
graphics	General purpose graphics functions.
lang	Language constructs and debugging.
matfun	Matrix functions – numerical linear algebra.
ops	Operators and special characters
plotxy	Two dimensional graphics.
plotxyz	Three dimensional graphics.
polyfun	Polynomial and interpolation functions.
sounds	Sound processing functions.
specfun	Specialized math functions.
specmat	Specialized matrices.
demos	The MATLAB Expo and other demonstrations.
simulink\ simulink	SIMULINK model analysis and construction functions.
simulink\ simdemos	SIMULINK demonstrations and samples.

Quitting and saving the workspace

To quit, type *quit* or *exit.* Termination of a Matlab session causes the variables to be lost. Before quitting, the workspace may be saved for later use by typing *save filename*, which saves to a file filename.mat. Similarly, *load filename* loads that file.

Matrix operations

We have seen something of these above. Addition and subtraction are performed simply by typing "+" or "-", Multiplication by "*". Of course, the dimensions of the matrix must be such that the operation has significance. There are two matrix division symbols in Matlab, / and \, corresponding to left and right multiplication. Thus, X=A\B is a solution to A*X=B, while X=B/A is a solution to X*A=B. Other elementary matrix functions include: *poly*, *det* for determinant.

Mathematical functions

The usual trigonometric and hyperbolic functions are included, but in addition there are *bessel*, *gamma*, *rat* (for rational approximation), *erf*, *inverf*, *ellipk* for the elliptic integral of the first kind and *ellipj* for Jacobian elliptic functions.

Vectors and matrix manipulation

Matlab allows manipulation of rows, columns and individual elements of matrices. There are a number of special matrices.

Data analysis

There are a number of elementary statistical tools. More powerful techniques are available for linear algebra and signal processing functions.

Polynomials and signal processing

As we have seen already, polynomials are represented in Matlab as row vectors containing the coefficients ordered by descending powers. Vectors are used to hold sequences for data processing. Some signal processing functions include: *conv* for convolution, *cov* for covariance, *deconv* for deconvolution, *fft* for a fast Fourier transform, *ifft* for an inverse. Some of these have two-dimensional counterparts which can be applied to matrices, such as *fft2*, *ifft2*, *conv2*.

The function y=filter(b,a,x) filters the data in vector x with the filter described by vectors a and b.

Function functions

A class of functions in Matlab works not with numerical matrices but with mathematical functions. These include numerical integration, nonlinear equations and optimisation and differential equation solution. Mathematical functions are represented in Matlab by function M-files. For example, the function

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

can be made available to Matlab by creating an M-file called humps.m:

```
>> function y = humps(x)
>> y=1./ ((x-0.3).^2+0.01)+1./ ((x-
0.9).^2+0.04)-6;
```

A graph of the function is then obtained by

```
>> x = -1: 0.01 : 2;  plot(x,humps(x))
```

as shown in Figure 7.

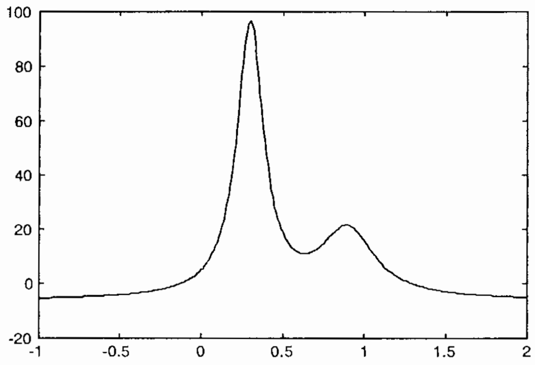


Figure 7: Graph of function.

Numerical integration

Two Matlab functions for numerical integration are:

quad – adaptive recursive Simpson's rule
quad8 – Newton Cotes 8 panel rule

To integrate *humps* from 0 to 1:

```
>> q = quad('humps',0,1)
```

which gives q = 29.8583, as does *quad8.*

Nonlinear equations and optimisation

The functions for this are shown in Table 4. The use of *help fmin*, for example, shows how these are implemented.

Table 4: functions for nonlinear equations and optimisation.

fmin	minimum of a function of one variable
fmins	minimum of a multivariable function
fsolve	solution to a system of nonlinear equations
fzero	zero of a function of one variable

Differential equation solution

The standard functions for numerical solution are: *ode23*, a 2nd/3rd order Runge Kutta method, and *ode45*, a 4th/5th order Runge Kutta method. Consider the second-order differential equation used previously:

$$\frac{d^2y}{dt^2} + y^2 = 0$$

subject to the same initial conditions. In this case it is necessary to write higher-order equations as systems of first order equations. Hence introducing the variable $z = dy/dt$, the system is

$$\frac{dy}{dt} - z = 0$$
$$\frac{dz}{dt} + y^2 = 0$$

It is necessary to create a function M-file containing these differential equations dydt.m:

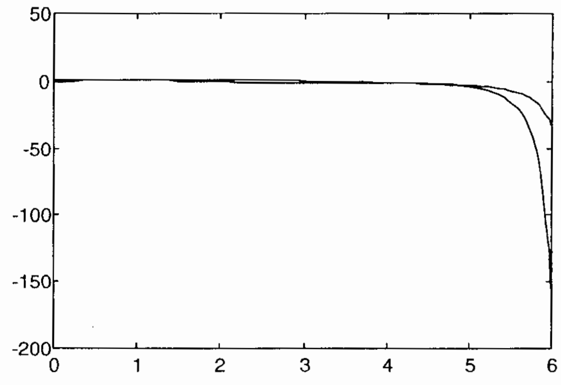


Figure 8: Plot of solution of differential equation.

```
function dydt = eqns(t,y)
dydt = zeros(2,1);
dydt(1) = +y(2);
dydt(2) = -y(1).^2;
```

and then to run the commands in Matlab:

```
» t0=0; tz=6; x0 = [0 1]'; %range & initial conds.
» [t,y] = ode23('dydt',t0,tz,x0); %solve
» plot(t,y)
```

which gives as output Figure 8 containing both the function and its derivative.

Comparison

The extensive built-in facilities of Matlab make it also a very powerful tool for the routine analysis of numerical data and the numerical solution of problems. Increasingly, around the world there are software packages being written using Matlab, and it may become, like Fortran, the *de facto* standard for scientific computing.

The two packages described in this article are complementary to a certain extent. Maple has considerable power, but it is necessary to have a certain amount of experience to operate it most effectively. Also, because it has the power of symbolic manipulation, it usually will be slower than Matlab when engaged upon routine computational tasks. Maple is, however, the personal preference of the author for the solution of research problems.

There are other symbolic manipulation packages, which are similar to Maple in appearance and operation. These include Mathematica, Macsyma, Reduce and Derive. The personal experience of the user will usually be the most powerful determinant of choice.

References

1. Char, B.W., et al, *First Leaves: A Tutorial Introduction to Maple V.* Berlin: Springer (1992).
2. Etter, D.M., *Engineering Problem Solving with MATLAB.* Englewood Cliffs: Prentice-Hall (1993).
3. The Mathworks Inc., *The Student Edition of MATLAB.* Englewood Cliffs: Prentice-Hall (1992).