

Introduction to MAPLE symbolic computation software

By

Gilberto E. Urroz, Ph.D., P.E.

Distributed by

 *infoClearinghouse.com*

©2001 Gilberto E. Urroz
All Rights Reserved

Download at InfoClearinghouse.com

1 Introduction to Maple

This chapter introduces the reader to the basic concepts and commands for using the symbolic mathematical environment Maple. The chapter includes a number of exercises to get the reader acquainted with the environment, its syntax, and a number of basic commands useful for algebraic, calculus, linear algebra operations, and ordinary differential equations.

What is Maple

Maple is a comprehensive computing environment for symbolic, numerical, and graphical applications in mathematics. The software was developed at the University of Waterloo in Canada in the 1980's and 1990's. Currently, *Maple* offers version 6, referred to as *Maple 6*, although plenty of previous versions (e.g., *Maple*, release 5.0 or 5.1) are still available. Programming is also available within *Maple*.

Maple facilitates obtaining solutions to problems that otherwise would have required extensive analysis by hand or by programming your computer. *Maple* has pre-programmed into its functions many algorithms for the solution of single non-linear equations, numerical integration, matrix algebra, linear algebra, etc. It also includes algorithms to solve polynomial equations, eigenvalues and eigenvectors, and other subjects that we will cover in this class.

Symbolic and Numerical Environments

Maple is a symbolic mathematical environment with a large number of functions for arithmetic, algebra, calculus, statistics, and other mathematical operations. It is a symbolic environment as opposite to a numerical environment, such as *Scilab* or *Matlab*. To understand the difference between a symbolic solution, as opposite to a numerical one, consider the following example: We want to calculate the integral

$$\int_0^a x^2 dx$$

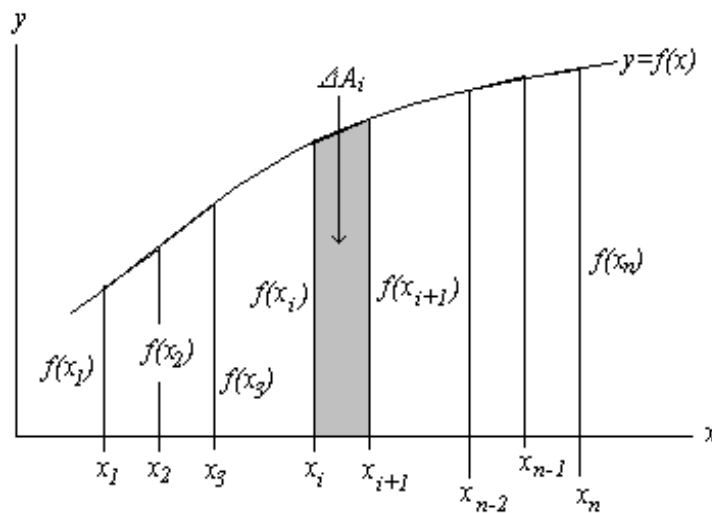
Because the upper limit of integration is a symbol (a variable), a , the result must be symbolic. By using the rules of anti-derivatives that we learn in our Calculus classes, we can write:

$$\int_0^a x^2 dx = \frac{x^3}{3} \Big|_0^a = \frac{a^3}{3}.$$

For a particular value of a , say $a = 1$, we can write the result of this integral as a fraction, e.g.,

$$\int_0^1 x^2 dx = \frac{1}{3}.$$

On the other hand, through the use of high-level computer languages, such as Visual Basic, C, Java, or FORTRAN, you can implement an algorithm to obtain a numerical approximation to the integral. A simple algorithm is the trapezoidal rule in which the interval of integration, $[a, b]$ is divided into a number of equally-spaced sub-intervals, and the area of the trapezoids generated by the sub-intervals is added to approximate the integral. The process is illustrated in the figure shown below.



As illustrated in the figure, we divide the interval $[a, b]$, where $a = x_1$, and $b = x_n$, into $(n-1)$ sub-intervals: (x_i, x_{i+1}) for $i = 1, 2, \dots, n-1$. The area under the curve corresponding to sub-interval i is

$$\Delta A_i = 1/2(x_{i+1} - x_i) (f(x_i)+f(x_{i+1})) = 1/2 \cdot \Delta x_i \cdot (f(x_i)+f(x_{i+1})),$$

where,

$$\Delta x_i = x_{i+1} - x_i.$$

Taking the interval width to be a constant value,

$$\Delta x = (x_n - x_1)/(n-1),$$

the integral is approximated by

$$I = \sum_{i=1}^n \Delta A_i = \frac{1}{2} \cdot [f(x_1) + 2 \cdot \sum_{i=2}^{n-1} f(x_i) + f(x_n)] \cdot \Delta x$$

where

$$x_i = x_1 + (i-1) \Delta x,$$

for $i = 1, 2, \dots, n$.

An implementation of this algorithm in Visual Basic, for example, produces the following values of I for different values of n when the function :

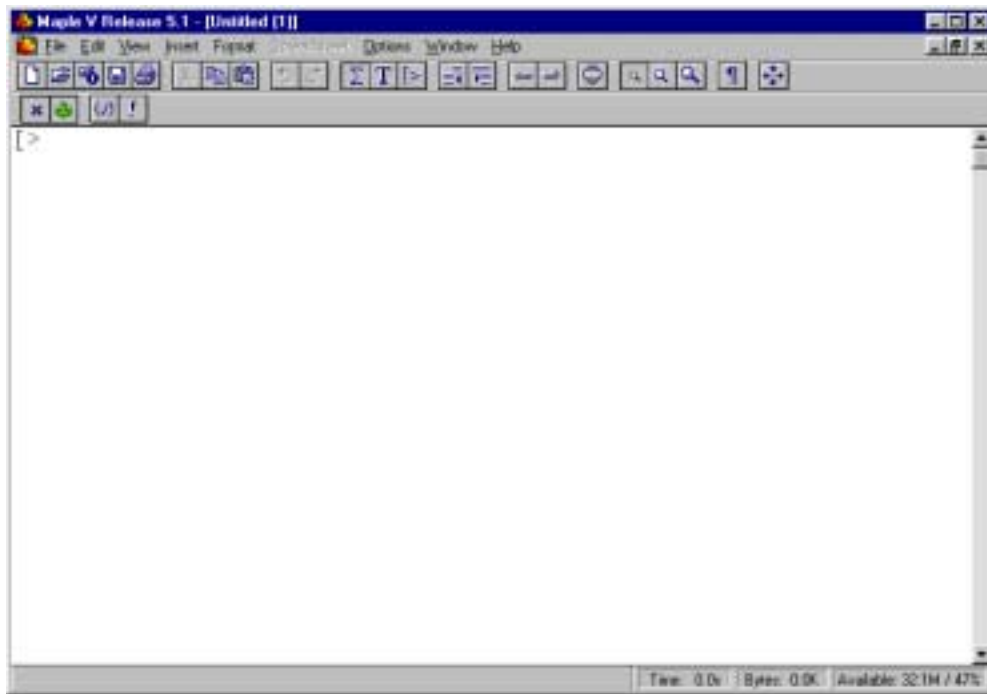
n	I
2	0.5000000
10	0.3353909
50	0.3334027
100	0.3333503
200	0.3333375
500	0.3333340
1000	0.3333335

Modern numerical environments, e.g., *Scilab*, or *Matlab*, provide functions that implement a large number of numerical algorithms, including the trapezoidal rule for integration, through a simple function call. Symbolic environments, such as *Maple* or *Mathematica*, also provide functions for the numerical evaluation of integrals when numerical integration limits are provided.

Numerical calculations are also possible in a symbolic environment by creating a program or procedure using the environment's own programming language. Thus, the classification of a mathematical environment as symbolic does not necessarily preclude the implementation of numerical calculations. The opposite is also true, for example, *Matlab*, being a numerical environment, includes a collection of functions (called a package) that implements a number of elementary symbolic operations such as derivatives, integrals, and solution of simple ordinary differential equations.

Getting started with Maple

Find the *Maple* icon in your computer by pressing the *Start > Programs* button, and double-click on the icon. The starting *Maple* screen looks like this:



Maple's New User's Tour

To get you started, select the option *New User's Tour* in the *Help* menu. This will provide you with an introductory document that explains many of the program's features. At the bottom of the screen there will be an underlined message:

[Click here to begin the New User's Tour](#)

Click on the text to access the index for the tour. The index includes the following items:

- (1) WORKING THROUGH THE NEW USER'S TOUR
- (2) THE WORKSHEET ENVIRONMENT
- (3) NUMERICAL CALCULATIONS
- (4) ALGEBRAIC COMPUTATIONS
- (5) GRAPHICS
- (6) CALCULUS
- (7) DIFFERENTIAL EQUATIONS
- (8) LINEAR ALGEBRA
- (9) STATISTICS AND FINANCE
- (10) PROGRAMMING
- (11) ON-LINE HELP
- (12) SUMMARY

Click on any of the titles shown to access that particular subject. At the bottom of each section in the main menu of the tour there will be an underlined message:

[Click here to Return to the Main Menu.](#)

which you can click to return to the main menu.

Start with item (1) to understand the workings of the New User's Tour. The subject of section (2) is fundamental to understand how a *Maple* worksheet, which will be your basic working environment, operates.

Sections (3) and (4) are important to understand numerical and algebraic calculations that can be useful in future applications. They also teach you many of the standard inputs to the program, such as the constants *Pi*, *I*, *infinity*, and the way in which a range of values of a variable is entered (e.g., *k=1..infinity*). Many important functions, such as *evalf*, *expand*, *factor*, and others use for algebraic manipulation of expressions are also shown.

Graphics, a very useful feature of *Maple*, are introduced in section (5) of the New User's Tour. Work through the examples carefully to understand all the important commands necessary to produce high-quality graphs. *Maple* uses collections of programs that facilitate mathematical calculations, graphs, and programming. Such collections are called *packages*, and they are invoked using the function *with*. The section on graphics includes an introduction to the use of packages.

Read through section (9) on Finance and Statistics to gain knowledge on the statistical packages available, and the statistical operations that can be performed within *Maple*. The first part of the section deals with financial calculations that can be very useful in problems of Engineering Economics.

Whenever you are using *Maple*, you have access to a very efficient on-line help facility through the *Help* menu. The New User's Tour is just one of the options available under that menu. Read section (11) of the New User's Tour to get more insights on the use of the *Help* menu.

In summary, for applications in probability and statistics, I recommend you at least check out items numbers (3), (4), (5), (9), and (10). For other numerical method applications check the remaining items (6), (7) and (8).

To leave the New User's Tour, you can click off the corresponding window, or use the option *Close* in the *File* menu. You will find that *Maple* opened as many windows as sections you requested to review through the tour. Therefore, you may have to close more than one window to clear up all the New User Tour's screens. If you want to keep the New User's Tour screens available, move to your worksheet by selecting the Window entitled *Untitled(1)* in the *Window* menu.

Maple on-line help

If you want to see more examples of any particular command, request the online information using the *?* operator in *Maple*. The help screens provided by *Maple* include application examples

Navigating Through Maple's New User's Tour

Step through the New User's Tour working through the numbered items in order. All you need to do is use your mouse to move the cursor to any *Maple* command and press the Enter key. *Maple* will process the command and show the output. This will help you become familiar with the different *Maple* fields, commands, and results. *Maple* contains a huge amount of commands, therefore, don't try to memorize all of the commands presented in the New User's Tour. You may, if you deem necessary, keep a summary of the commands you find useful in a notebook for easy reference.

Key concepts in the New User's Tour

WORKING THROUGH THE NEW USER'S TOUR

Worksheet

Section box

Expanding and collapsing section boxes

THE WORKSHEET ENVIRONMENT

Computational engine

Execution groups

Symbolic spreadsheets

Commands: typing, context menus, drag and drop

Annotating the worksheet

Paragraphs and text

Sections and sub-sections

Hyperlinks

NUMERICAL CALCULATIONS

Integers, floating-point, and complex numbers

ALGEBRAIC COMPUTATIONS

All subjects are important in this section - study it very carefully

GRAPHICS

Commands: *restart* and *with*

Take notes on all graphics commands that you find useful

Notice how the value of π (Pi) is entered

CALCULUS

All subjects are important in this section

The section is a nice review of key calculus concepts

Take notes on the special functions available (*error*, *gamma*, etc.)

Option *Order* for series expansion

Commands: *convert*, *piecewise*, *Diff*, *Int*, *series*

DIFFERENTIAL EQUATIONS

Command: *dsolve*

Notation: D for derivatives

LINEAR ALGEBRA

Package: *linalg*

Commands: *matrix*, *inverse*, *det*, *multiply*, *eigenvectors*, *eigenvals*, *map*, *hilbert*, *vandermonde*

STATISTICS AND FINANCE

Packages: *stats*, *finance*

Finance commands: *amortization*, *amortization_table*, *annuity*, *growingannuity*, *cashflow*, etc.

Sub-packages within stats: *fit*, etc.

We will study the stats package in more detail during the course.

PROGRAMMING

Procedures

Local variables

Some useful Maple commands

This section contains a list of some *Maple* commands that have a wide range of applications.

Some useful operations with floating point results:

trunc(x) or **trunc**(n, x): truncate a number to the next nearest integer towards 0

round(x) or **round**(n, x): round a number to the nearest integer

frac(x) or **frac**(n, x): fractional part of a number

floor(x) or **floor**(n, x): greatest integer less than or equal to a number

ceil(x) or **ceil**(n, x): smallest integer greater than or equal to a number

fnormal(e), **fnormal**($e, digits$), or **fnormal**($e, digits, epsilon$): returns an expression equivalent to e under the assumption that all numeric values with magnitude less than $epsilon$ may be considered to be zero.

Commands for algebraic expressions:

coeff(p, x), **coeff**(p, x, n), or **coeff**(p, x^n): extract a coefficient of a polynomial

numer(): numerator of an expression

denom(): denominator of an expression

normal(f) or **normal**($f, expanded$): normalize a rational expression

combine(f), **combine**(f, n), or **combine**($f, n, opt1, opt2, \dots$): combine terms into a single term

List of pre-defined mathematical functions:

Use `>?inifcns` to obtain a list of all pre-defined math functions available in *Maple*.

readlib (): read a library file to define a specified name. Whether a library function needs to be defined is indicated in the online help and manual descriptions for the function.

Environment variables:

Digits, Normalizer, Testzero, mod, printlevel, %, %, %%%

ditto operator: In *Maple*, percentage signs are used to refer to previously computed expressions. Specifically, the following operators are defined:

% last expression

%% second last expression

%%% third last expression

Commands that operates on expressions:

op(): extract operands from an expression

nops(): the number of operands of an expression

subsop(): substitute for specified operands in an expression

map(fcn, expr, arg2, ..., argn): apply a procedure to each operand of an expression

map2(fcn, arg1, expr, arg3, ..., argn): apply a procedure with a specified first argument to each operand of an expression

unapply(expr,x,y,...): returns an Operator from an Expression and Arguments. Useful for defining functions after the required expression has been created. See `>?unapply` for examples.

assign(): similar to using the assign symbol :=

convert(): use `>?convert` for more details

testeql(a = b), **testeql(a, b)**, or **testeql(a)**: equivalence tester → tests for equivalence probabilistically. It returns false if the expressions are not equal (or not equal to 0) and true otherwise for the class of expressions that *testeql* understands. The result false is always correct; the result true may be incorrect with very low probability.

series(expr, eqn) or **series(expr, eqn, n)**: generalized series expansion

Order: The environment variable Order represents the order of series calculations performed by *Maple*. It does not represent the order of the series output. The default value of Order is 6.

copy(a): create a duplicate array or table

linalg[augment](A, B, ...) or **linalg[concat](A,B,...)**: join two or more matrices together horizontally

The Assume Facility

assume(x,prop): tells *Maple* properties about variables and relationships between variables, e.g., `> assume(a>0)`; `> assume(0<x,x<1)`; When you make assumptions about a variable, thereafter it prints with an appended tilde ~ to indicate that the variable carries assumptions. Assumptions made on names may be erased (cleared) by un-assigning names, e.g., `x := 'x'`;

additionally(x,prop): permit additional assumptions to be made.

is(x,prop): determines (computes) whether a given property is true, false or FAIL. FAIL means that it could not determine whether the property was true or false.

about(x): displays information about current status of objects, assumptions and properties.

addproperty(prop,parents,children): installs a new property in the property tables.

Examples: `> assume(x, RealRange(Open(0), Open(1)))`; #x belongs to (0,1).
`> assume(x, RealRange(Open(1), 2))`; #x belongs to (1,2].

Maple V packages

The following are some selected packages available in Maple that may be of interest for engineering mathematics applications:

DEtools	differential equations tools
LREtools	manipulate linear recurrence relations
Matlab	Matlab Link
codegen	Code Generation
combinat	combinatorial functions
finance	financial mathematics
geom3d	Euclidean three-dimensional geometry
geometry	Euclidean geometry
inttrans	integral transforms
linalg	Linear algebra
networks	graph networks
numapprox	numerical approximation
orthopoly	orthogonal polynomials
plots	graphics package
plottools	basic graphical objects
powseries	formal power series
simplex	linear optimization
stats	statistics
student	student calculus
tensor	tensor computations and General Relativity

For information see *?package* where *package* is from the above list. This will give a list of the functions available in the package. To cause all functions in a package to be defined in the session, do: *with(package);*

For information on a particular package function, see *?package,function*

The code generation package

codegen[function](args):

The code generation package is a collection of tools for creating, manipulating, and translating *Maple* procedures into other languages. This includes tools for automatic differentiation of *Maple* procedures, code optimization, translation into C and Fortran, and an operation count of a *Maple* procedure.

See > ?codegen for more information.

Examples:

codegen[optimize](): yields efficient sequence of calculations for an expression.

codegen[makeproc](): makes a *Maple* procedure from formulae.

codegen[cost](): evaluates the number and type of operations in an expression.

Using quotes to show expression before evaluation

By entering an expression between quotes, the output from *Maple* is the *Maple* input version of the expression. Otherwise, *Maple* will return the evaluated expression. If you want to see the *Maple* input version and the result, enter, in the same line, the expression between quotes, followed by the ditto operator. Example:

```
> 'int(exp(-x), x = 0..tau)';%
```

$$\int_0^{\tau} e^{-x} dx$$
$$-e^{-\tau} + 1$$

Exercises

Use of functions *trunc*, *round*, *frac*, *floor*, and *ceil*.

[1]. Truncate the following numbers to the nearest integers towards zero:

- (a) 112.34 (b) -5.22 (c) 17.85 (d) $-\pi$ (e) $12.76/9.2$ (f) 1.23^3

[2]. Round the following numbers to the nearest integer:

- (a) -11.22 (b) 3.555 (c) -22.2222 (d) 8.22 (e) $5\pi/2$ (f) $23^{1/2}$

[3]. Obtain the fractional part of the following numbers:

- (a) 24.56 (b) $12.34/5.6$ (c) 2.33^3 (d) $18.22/3$ (e) $-11\pi/2$ (f) $18^{1/3}$

[4]. Find the largest integer less than or equal to the following numbers:

- (a) -11.34 (b) 18.32 (c) $23\pi/5$ (d) $(3^2+5^2)^{1/5}$ (e) 28.52 (f) 3.1416

[5]. Find the smallest integer greater than or equal to the following numbers:

- (a) 112.34 (b) $12.34/5.6$ (c) 8.34 (d) $12.76/9.2$ (e) $23\pi/5$ (f) -11.22

Manipulation of some algebraic equations

[6]. Extract the coefficient of x^2 in the expressions shown below:

- (a) $(1+x)(x+x^2)$ (b) $(x+2)(x+3)$ (c) $x(x+1)(x+2)$ (d) $(x^4-1)/(x+1)$ (e) ax^2+bx+c (f) $1+x+x^2+\cos(x)$

[7]. Extract and simplify (if possible) the numerator and denominator of the following algebraic fractions:

- (a) $(x^3+3x+1)/(x+2)$ (b) $(x+1+x^2+3x^4)/(\cos x + 1)$ (c) $(x^2+2x+2)/(1+x^{1/3})$

[8]. Change the parameter *Digits* to show the value of π to:

- (a) 5 decimals (b) 10 decimals (c) 50 decimals (d) 100 decimals

To change the value of the parameter Digits, use: `> Digits: = 20;` etc. To see the floating-point value of π use the command: `> evalf(Pi);`

[9]. Use the function *op* to extract operands from the following expressions:

- (a) x^2+x+2 (b) $[1,x,x^2,x^3]$ (c) $\sin(x)$ (d) $\sin(x) + \cos(x)$

[10]. Use the function *nops* to obtain the number of operators in the expressions of problem [9].

[11]. Use the function *testeq* to check the following statements:

- (a) $\sin^2 x + \cos^2 x = 1$ (b) $3^2+4^2=5^2$ (c) $\tan u = \sin u/\cos u$ (d) $(1+x)^2 = 1 + x + x^2$

[12]. Use the following command to produce a plot of the function $f(x) = |x|$: `> plot(abs(x),x = -2..2);` Next, repeat the plot but assuming that $x > 0$, i.e., use: `> assume(x>0); plot(abs(x), x = -2..2);` Explain the different results. [*Hint*: By definition, $|x| = x$, if $x > 0$, and $|x| = -x$, if $x < 0$.]

2 Calculus, linear algebra, and ODEs - Brief introduction to Maple applications

In this chapter we learn about algebraic expressions and their manipulation, calculus and linear algebra applications. The contents of the chapter represent a brief introduction to those types of operations using *Maple*. More extensive coverage of these subjects will be provided in subsequent chapters.

Comments

Comments in *Maple* can be attached to a command by placing a number (#) symbol after the semi-colon that ends the command. For example,

```
> plot(sin(x), x = -3*Pi.. 3*Pi); #This command will plot the function y = sin(x) between -3Pi and +3Pi
```

We use comments to describe some of the commands below. When trying these exercises yourself, you don't need to type the comments.

Simple algebraic operations

Try the following exercises in your computer. Some input lines contain comments. Comments in *Maple* start with the symbol # and are used only for documenting input lines. Comments are not executed by *Maple* and can be ignored when trying the exercises in your computer.

```
> restart; #use restart to clear worksheet memory.
> 3+2; # a simple arithmetic operation
> sqrt(325); #square root function
> evalf(%); #evalf means "evaluate using floating point". This command is used to produce floating point results
> '1+1/(1+1/(1+1/(1+x)))';%; #Repeated fraction
> simplify(%); #to simplify an expression. The symbol % indicates "last result"
> expand(y*(x-a)*(x-b)*(x-c));
> factor(x^3-3*x^2+9*x-27);
> simplify((x^7+2*x^6-3*x^5+4*x^4-18*x^3+6*x^2-120*x-120)/(x^3-2*x^2-5*x-2));
> factor(%);
```

Polynomials

The following exercises involve polynomials and their operations:

```
> divide(-x^5+9*x^4-18*x^3+x^2+19*x-10,x-2,'q'); #Polynomial division, 'q' stores the
quotient
> q; # to show the result stored in 'q'
> pol1:= -x^5+9*x^4-18*x^3+x^2+19*x-10;
> pol2:=x-2;
> divide(pol1,pol2,'q');
> q;
> rem(pol1,pol2,x,'q'); #remainder of a polynomial division
> rem(pol1,(x^2-7),x,'q');
> q;
> solve(a*x^2-b*x+c,x);
> solve(a*x^2-b*x+c,{x});
```

Solution of equations

Projectile motion equations - example of equation solutions:

```
> eqnProjX:= x = x0 + v0*cos(theta.0)*t;
> eqnProjY:= y = y0 + v0*sin(theta.0)*t - (1/2)*g*t^2;
> v0:=25; theta.0:=45*Pi/180; x0:= 10.0; y0:=15.0; g:= 32.2; t:= 23.0;
> solve({eqnProjX,eqnProjY},{x,y});
```

A second example of equation solutions follows:

```
> restart;
> eqnProjX:= x = x0 + v0*cos(theta.0)*t;
> eqnProjY:= y = y0 + v0*sin(theta.0)*t - (1/2)*g*t^2;
> v0:=25; x0:= 10.0; y0:=15.0; g:= 32.2; x:=300; y:= -2500;
> solve({eqnProjX,eqnProjY},{t,theta.0});
```

Numerical solutions are also possible by using the function *fsolve* (the f stands for floating-point):

```
> fsolve({eqnProjX,eqnProjY},{t,theta.0},0..10,0..20);
> subs(%,{eqnProjX,eqnProjY});
> evalf(%); #evalf stands for "evaluate as a floating point result"
```

Functions and graphs

To define a function, say, $f(x) = x^2 - \exp(-x/2) + 2$, use the arrow operator as follows:

```
> restart;
> f := x -> x^2 - exp(-0.5*x) + 2; # function definition
> f(0); # function evaluation
> f(Pi); f(Pi/2); # more function evaluation
> evalf(%); # floating point results
```

Create a sequence for values of $f(0)$, $f(1)$, $f(2)$, $f(3)$, $f(4)$:

```
> f(0), f(1), f(2), f(3), f(4); # This represents a Maple sequence
> seq(f(j), j=0..4); # Use the function "seq" to generate the sequence automatically
```

A list is a sequence of "tokens" enclosed between square brackets. The following examples involve *Maple* lists:

```
> [f(0), f(1), f(2), f(3), f(4)];
> [seq(f(j), j=0..4)];
> xList := [seq(2.5 + j*0.5, j=0..20)];
> yList := map(f, xList); # the command "map," in this case, applies function f to each
element of xList
```

Check out the package "*plots*" which includes a number of functions for producing two- and three-dimensional plots. *Packages* are collection of functions with a common theme. To see information on the *plots* package use:

```
> ?plots
> with(plots); # Use the command "with" to load a package
> listplot(xList, yList); # Plots points by using values from xList and yList
> nops(xList); # The function "nops" determines the number of elements or operators
in a list
> pointList := [seq([xList[j], yList[j]], j=1..21)]; # Creating a sequence of [x,y] points
> listplot(pointList); # plotting the points with "listplot"
```

The following call to *listplot* includes labels and a title. The plot style is changed to points rather than a continuous line:

```
> listplot(pointList, axes = boxed, labels = ["t(s)", "x(m)"], title = "particle motion", style =
point);
> plot(f(x), x=0..100, style = point, symbol = circle); # The plot command used for plotting
with symbols
```

Composite functions examples:

```
> unassign('x');           #clears the variable name 'x'
> h:= x -> x+sin(x);      #define h(x)
> ?@                       #requests on-line help for the command 'at' (@)
> r:= combine(f@h);       #this is the same as f(h(x))
> r(2);                   #Calculating r(x) = f(h(x))
> evalf(%);              #evaluating as a floating-point
> plot(r(s),s=0..10);     #a plot of the composite function
```

Examples of functions of two variables:

```
> unassign('x','y','f');   #clears variable names x, y, and f
> f;                       #check that f has been cleared
> f:= (x,y) -> x*sin(y) + y*sin(x); #define a function f(x,y)
> plot3d(f(x,y), x = -4*Pi..4*Pi, y = -4*Pi..4*Pi); #Use plot3d to plot the surface z = f(x,y)
> plot3d(f(x,y), x = -2*Pi..2*Pi, y = -2*Pi..2*Pi);
```

Plotting more than one function in a graph:

```
> restart;
> f:= x -> x^2-5*x+2;
> g:= x -> 2*x-3;
> plot({f(x),g(x)},x=0..20);
> plot(f(x),x = 0..10, style = point, symbol = box, axes = boxed, color = MAGENTA);
> plot1:= %;               #Saving a plot into a variable
> plot(g(x),x = -10..10, style = line, linestyle = 4);
> plot2:= %;              #Saving a second plot
> plots[display]({plot1,plot2}); #Use command "display" in package "plots"
```

Examples involving functions of two variables:

```
> restart;
> f := (x,y) -> x^2 + y^2 - 25;
> g := (x,y) -> x^2 - y^2 -5;
> plot3d({f(x,y), g(x,y)}, x = -10..10, y = -10..10, axes = boxed);
> solve({f(x,y) = 0, g(x,y) = 0},{x,y}); #solves system of equations
> allvalues(%); # the command allvalues show symbolic solutions if available
```

Calculus applications

Limits

```
> restart;
> limit(sin(x)/x,x=0);
> f:= x -> x^2+tan(x);
> plot(f(x),x=-2*Pi..2*Pi, discount = true, -10..10); #option 'discount=true' eliminate discontinuities in the graph.
> limit(f(x),x=-2);
```

To find help on the command 'piecewise':

```
> ?piecewise
```

More examples on limits follow:

```
> restart;
> g(x):= piecewise(x<-1,x^2-1,x>=-1 and x<1, x, x>=1, exp(0.1*x));
> plot(g(x),x=-3..3, discount=true);
> limit(g(x),x=-1,left);
> limit(g(x),x=-1,right);
```

Derivatives

```
> restart;
> D(x^2); #You can use 'D' or 'diff' for derivatives.
> diff(x^2,x); #diff(x^2,x) represents d(x^2)/dx
> f:= x -> ln(x);
> diff(f(x),x);
> fp:= x -> diff(f(x),x);
> plot(f(x),x = 0..2);
> plot(fp(x),x=0..2);
> plot({f(x), fp(x)},x = 0..2); #plotting two functions in one plot
> D(u(x)*v(x)); #derivative of a product of functions
> diff(u(x)*v(x),x);
> D(u(x)/v(x)); #derivative of a quotient of functions
> diff(u(x)/v(x),x);
> restart:f(x):= 7.5*x^2 - 2.375*x;
> plot(f(x),x=-2..4);
> plot({f(x),diff(f(x),x)},x=-0.5..0.5,-2..2);
> solve(diff(f(x),x)=0,{x}); #solving an equation based on a derivative
> subs(%,diff(diff(f(x),x),x)); # the function 'subs' is used to substitute variables in
expressions
```

Integrals

```
> restart;
> int(f(x),x);           #indefinite integrals
> int(x*ln(x),x);
> int(x*ln(x),x = 1..tau);   #definite integrals
> int(exp(-z^2/2),z=0..y);   #result given in terms of the error function
> plot(erf(lambda), lambda=-4..4); #using the error function
> restart;
> Q:=int(2*Pi*r*v(r),r = 0..R); # formula for discharge in a pipe
> v:= r -> vmax*(1-(r/R)^2);   #laminar flow velocity distribution in a pipe
> Q:=int(2*Pi*r*v(r),r = 0..R); #formula for discharge in a pipe using the velocity v(r)
> A:=Pi*R^2;                 #cross-sectional area of the pipe
> V:=Q/A;                   #mean velocity in the cross-section
> plot(v(r),r=0..R);        #Error produced because R, vmax are not defined
> R:=1;vmax:=1;             #Values of R and vmax evaluated to a number
> plot(v(r),r = -R..R);
```

The *student* package

The student package contains a number of functions that can be used to illustrate typical calculus operations.

To find help on packages in general use:

```
> ?packages
```

To find help on the *student* package use:

```
> ?student
> with(student);           #load student package
> leftbox(x^2, x = 0..2);   #integral approximation - graph
> leftbox(x^3+3*x^2-2*x+5, x = 0..2, 10);
> leftsum(x^3+3*x^2-2*x+5, x = 0..2, 10); #integral approximation - summation
> evalf(%);               #find floating-point value of previous result
> int(x^3+3*x^2-2*x+5, x = 0..2); #exact integration
> rightbox(exp(0.1*x), x = 0..10, 10); #a second graph for integral approximation
> simpson(exp(0.1*x), x= 0..10, 10); #Simpson's rule for numerical integration
> evalf(%);
> int(exp(0.1*x), x = 0..10);
> middlebox(1/x, x = 1..2, 8);
> extrema( a*x^2+b*x+c,{},x ); #Calculating extreme values of a function
> ?Doubleint              #Help on double integrals
> Doubleint(x*y, y = 0..x, x = 0..10); #Calculating a double integral
> evalf(%);
> Doubleint(f(x,y),x,y,A); #Expression for an integral over an area
```

Vectors and matrices (arrays)

Two-dimensional vectors

```
> restart;
> u:=array([2,1]); v:=array([3,5]);      #use the command 'array' to define a row vector
> with(linalg);                          #load package 'linalg' (linear algebra) for vector operations
> dotprod(u,v);                          #dot product
> crossprod(u,v);                        #cross product
> norm(u,2);                             #absolute magnitude (length) of vector u
> norm(v,2);                             #absolute magnitude (length) of vector v
> arccos(dotprod(u,v)/(norm(u,2)*norm(v,2))); #find the angle between the two vectors
> evalf(%);                              # result given in radians
> %*180/Pi;                              # convert to degrees
> evalf(%);
```

Vectors in 3 dimensions

```
> restart;
> with(linalg);
> u:=vector([1,-1,7]);
> v:=vector([-5,2,4]);
> norm(u,2);
> norm(v,2);
> dotprod(u,v);
> crossprod(u,v);
> d:=u-v;
> d:=evalm(u-v);
> norm(d,2);
> w:=array([a,2,-5]);
> dotprod(u,w);
> solve(%=0,{a});
```

Matrices

```
> restart: with(linalg):                #two commands in the same line separated by a colon
> A:=array([[2,3,-1],[3,-6,2],[-1,2,5]]); #use 'array' to define matrices too
> I A:= evalm(array(identity,1..3,1..3)); #identity matrix - 'evalm' means 'evaluate as a
matrix'
```

Use the 'evalm' command for some matrix operations as shown below:

```
> evalm(A &* I A);           #Matrix product requires the use of '&*' rather than '**'
> evalm(I A &* A);
> det(A);                   #determinant
> transpose(A);            #transpose of a matrix
> AM:=inverse(A);          #inverse of a matrix
> evalm(A &* AM);          #product of A and its inverse
> evalm(AM &* A);
> B:=evalm(A - lambda*I A); #equation defining eigenvalues
> CharEq:= det(B) = 0;     #characteristic equation
> solve(CharEq,{lambda});  #solve characteristic equation to find eigenvalues
> evalf(%);
> eigvals(A);              # Maple's own function for finding eigenvalues
> evalf(%);
> eigenvects(A);          # Maple's own function for finding eigenvectors
> evalf(%);
> v1:=vector([-1.179489869+.1e-8*I, .127810770-.2e-9*I, 1.]); #I just copied the vectors from the
> v2:=vector([1.79828530, -5.298661501+.1e-9*I, 1.]);         # previous result in the right-hand-side
> v3:=vector([7.524061680, 2.742279289+.1e-9*I, 1.]);         # of these equations
> dotprod(v1,v2);         #these three dot-products are used
> dotprod(v1,v3);         # to verify that vectors v1, v2, and v3
> dotprod(v2,v3);         # are orthogonal to each other
> subs(lambda = A, CharEq); # the matrix A satisfies its own characteristic equation
> evalm(%);
```

Solution of systems of linear equations

```
> restart;
> with(linalg);
> eq1:= 2*x - 5*y + 4*z = 0;
> eq2:= -5*x + 4*y +10*z = 23;
> eq3:= 2*x + y - 40*z = -28;
> solve({eq1,eq2,eq3},{x,y,z});
> A:=matrix(3,3,[2, -5, 4, -5, 4, 10, 2, 1, -40]);
> b:=matrix(3,1,[0,23,-28]);
> soln:= evalm(inverse(A) &* b);
```

Solution to simple ordinary differential equations

This exercise emphasize solution of ordinary differential equations:

```
> restart;
```

The equation 'eq1' stores the ODE: $d^2y/dx^2 + y(x) = 0$:

```
> eq1:=diff(y(x),x$2)+y(x)=0; # the symbol 'x$2' represents a second derivative
> sol1:=dsolve(eq1,y(x)); # a general solution for the equation
```

The equation 'eq2' stores the ODE: $d^2y/dx^2 + y(x) = 0$, etc.:

```
> eq2:=diff(y(x),x$2)+y(x)=x;
> sol2:=dsolve(eq2, y(x));
> sol3:=dsolve(diff(y(t),t$2)+w0^2*y(t)=sin(w*t),y(t));
> simplify(%);
> sol4:=dsolve(diff(y(t),t$2)+w0^2*y(t)=sin(w0*t),y(t));
> simplify(%);
> sol4:=dsolve(diff(y(t),t$2)+w0^2*y(t)=sin(w0*t),y(t),method=laplace); #Uses Laplace
transforms
> sol4a:=subs({y(0)=0, D(y)(0)=0},sol4); #substituting for initial conditions
```

The following exercise uses the command *dsolve* including initial conditions and Laplace transforms:

```
> eq5:=dsolve({diff(y(t),t$2)+y(t) = sin(t), y(0) = 0, D(y)(0)=1},y(t),method = laplace);
> plot(rhs(eq5),t=0..4*Pi);
```

The following exercise shows a number of solutions by changing a parameter:

```
> spring1:=diff(y(t),t$2)+b*diff(y(t),t)+w0^2*y(t)=0;
> sys1:=dsolve({spring1,y(0)=1,D(y)(0)=0},y(t),method=laplace);
> simplify(%);
> bb:=[0,0.75,1.5,2.25,3];
> plot({seq(subs({w0=1,b=bb[i]}, rhs(sys1)),i = 1..5)}, t = 0..6*Pi);
```

Note: In the previous command, the function 'rhs' selects the right-hand side of sys1

The following exercise shows how to separate the amplitude of the transient part of a solution to an ordinary differential equation. The ODE represents an oscillatory motion:

```
> restart;
> spring2:=diff(y(t),t$2)+2*diff(y(t),t)+y(t)=cos(w*t);
> sys2:=dsolve({spring2,y(0)=0, D(y)(0)=0},y(t),method=laplace);
> ww:=[0, 0.5, 1, 1.5, 2];
> plot({seq(subs(w=ww[i],rhs(sys2)),i=1..5)},t=0..6*Pi);
> s2:=rhs(sys2);
```

```

> collect(s2,exp(-t));          #collects terms in exp(-t) from expression s2
> ss := op((4,s2)) + op((5,s2)) + op((6,s2));
Note: the function op(n,expression) in the previous result gets n-th operator (factor) from
expression
> collect(ss, cos(w*t));       #collects terms in cos(w*t)
> coeff(ss, cos(w*t));         #gets the coefficients from the terms in cos(w*t)
> amp:=sqrt(%^2+coeff(ss, sin(w*t))^2);    #calculates amplitudes
> assume(w>0);                 #use 'assume' facility to specify positive freqs. only
> amp:=simplify(amp);
> plot(amp, w=0..10);
> phi:=simplify(arctan(coeff(ss,cos(w*t))/coeff(ss,sin(w*t)))); #calculates the phase
> plot((180/Pi)*phi, w = 0.1 ..3);

```

The package *DEtools* contains the function *odeadvisor* that lets you classify a particular ordinary differential equation:

```

> with(DEtools, odeadvisor);
> odeadvisor(diff(y(t),t$2)+w0^2*y(t)=sin(w0*t));

```

Exercises

[1]. Expand the following algebraic expressions

(a) $(a+2)^3(b-2)$ (b) $(\sin x + \cos x)^2$ (c) $(1+x^3)(1+x)^2$ (d) $(3+5x)(x^2 + 2x + 2)$

[2]. Simplify the following algebraic expressions

(a) $(2x^2+x-10)/(x-2)$ (b) $\sin^2 x + \cos^2 x + 2\sin x \cos x$ (c) $(x^3-3x^2+3x-1)/(1-x)$

[3]. Using the functions *quo* and *rem*, determine the quotient and remainder of the following polynomial divisions:

(a) $(3x^3+11x^2+8x+3)/(x^2+3x+1)$ (b) $(7y+2+3y^2)/(y+2)$ (c) $(z+z^2+z^3+1)/(1+z+z^2)$

[4]. Use the function *solve* to find the solution(s) to the following equations:

(a) $x^2+2x = 0$ (b) $\lambda^3+\lambda+\sin(\pi/6) = 0$ (c) $(r+1)(r+2) - 2(r+3) + 5 = 0$ (d) $z + z(z+2) - z^3 = (z-2)^2$

[5]. Use the function *solve* to obtain solutions to the following systems of equations. Use the function *allvalues* if necessary:

(a) $\{x + y + z = 2, x + y = 3, x - z = 5\}$ (b) $\{x^2 + y^2 = 50, x + y = 10\}$
(c) $\{y = 5 + 3t + t^2, x = t + 1, y = 2t\}$ (d) $\{\sin(x) + \cos(y) = 1, \sin(y) + \cos(x) = 1\}$

[6]. Define the following functions using the arrow operator (e.g., $f := x \rightarrow x^2$;). Plot the functions in the range indicated:

(a) $r(s) = \sin s + 2, s = -2..2$ (b) $h(t) = t^2+t-\sin(t), t = 0..3\pi$
(c) $g(z) = \exp(-z^2/2), z = -2..0$ (d) $\phi(\xi) = (\xi + 1)^{1/2}, \xi = 0..2$

- [7]. Generate sequences of 5 values of the functions in problem [6] in the domain indicated.
- [8] For the sequence of values of $x = [-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5]$, use the function *map* to calculate sequences of values of the functions in problem [6].
- [9]. For the sequence in problem [8] and the functions in problem [6] produce a list of data points $[x,y]$, and produce a plot using the function *listplot* in package *plots*.
- [10]. For the functions of problem [6] determine the expressions for the following composite functions:
 (a) $r(g(t))$ (b) $f(h(x))$ (c) $g(h(t))$ (d) $h(\phi(r))$
- [11]. Define the following functions of two variables using the arrow operator (e.g., $f := (x,y) \rightarrow x*y^2$;). Plot the functions in the range indicated using *plot3d*:
 (a) $f(x,y) = x \sin y$, $x = 0..10$, $y = 0..10$ (b) $g(x,y) = 1/(x^2+y^2)^{1/2}$, $x = -10..10$, $y = -10..10$
 (c) $h(x,y) = xy + x + y$, $x = 0..10$, $y = 0..10$ (d) $\psi(x,y) = x^2 - y^2$, $x = -10..10$, $y = -10..10$
- [12]. For the functions of problem [6] determine the limits when the independent variable takes the values indicated below:
 (a) $s = -5$ (b) $t = \pi/2$ (c) $z = \text{infinity}$ (d) $\xi = -1/2$
- [13]. Determine the first and second derivatives of the functions in problem [6] with respect to the independent variables.
- [14]. Determine the indefinite integrals of the functions in problem [6] with respect to the corresponding independent variables.
- [15]. Determine the definite integral of the functions in problem [6] in the interval given.
- [16]. Use the functions *leftbox*, *midbox*, and *rightbox*, from the *Student* package, to plot the approximation of the definite integrals for the functions of problem [6] in the intervals indicated.
- [17]. Use the functions *leftsum*, *midsum*, and *rightsum*, from the *Student* package, to calculate the approximation of the of the definite integrals for the functions of problem [6] in the intervals indicated.
- [18]. Use the function *extrema*, from the *Student* package, to determine local minima and maxima for the functions in problem [6] in the intervals indicated.
- [19]. Consider the vectors $\mathbf{u} = 2\mathbf{i} + 3\mathbf{j} - \mathbf{k}$, $\mathbf{v} = -2\mathbf{i} + \mathbf{k}$, and, $\mathbf{w} = 3\mathbf{i} + 2\mathbf{j} - \mathbf{k}$. Calculate the following values:
 (a) $\mathbf{u} + \mathbf{v}$ (b) $\mathbf{u} - \mathbf{w}$ (c) $\mathbf{u} \cdot \mathbf{v}$ (d) $\mathbf{u} \times \mathbf{w}$ (e) $|\mathbf{u}| |\mathbf{v}| / |\mathbf{w}|$
 (f) angle between vectors \mathbf{u} and \mathbf{w}
- [20]. Consider the matrices $\mathbf{A} = \begin{bmatrix} 2 & 3 & -1 \\ 5 & 5 & 2 \\ 0 & 2 & -1 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}$. Obtain:
 (a) \mathbf{A}^T (b) \mathbf{A}^{-1} (c) $\mathbf{A} \cdot \mathbf{b}$ (d) solve for \mathbf{x} from $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$
 (e) characteristic equation for \mathbf{A} (f) eigenvalues of \mathbf{A} (g) eigenvectors of \mathbf{A}
- [21]. Determine the general solution to the following ordinary differential equations:
 (a) $dy/dx + y - x = 0$; (b) $d^2u/dt^2 + 2(du/dt) - u = 0$ (c) $d^3h/dt^3 + h = \exp(x)$

[22]. Determine the solution to the following ordinary differential equations with the initial conditions indicated:

(a) $dy/dx + y - x = 0, y(0) = 1$ (b) $d^2u/dt^2 + 2(du/dt) - u = 0, u(0) = 1, u'(0) = 1$

REFERENCES

- Farlow, Stanley J., 1982, "*Partial Differential Equations for Scientists and Engineers*," Dover Publications Inc., New York.
- Friedman, B., 1956 (reissued 1990), "*Principles and Techniques of Applied Mathematics*," Dover Publications Inc., New York.
- Gullberg, J., 1997, "*Mathematics - From the Birth of Numbers*," W. W. Norton & Company, New York.
- Harris, J.W., and H. Stocker, 1998, "*Handbook of Mathematics and Computational Science*," Springer, New York.
- Heal, K.M., M.L. Hansen, and K. M. Rickard, 1998, "*Maple V - Learning Guide*," Springer, New York.
- Kamerich, E., 1999, "*A Guide to Maple*," Springer, New York.
- Keener, J. P., 1998, "*Principles of Applied Mathematics*," Perseus Books, Reading, Massachusetts.
- Kottegoda, N. T., and R. Rosso, 1997, "*Probability, Statistics, and Reliability for Civil and Environmental Engineers*," The Mc-Graw Hill Companies, Inc., New York.
- Kreysig, E., 1983, "*Advanced Engineering Mathematics - Fifth Edition*," John Wiley & Sons, New York.
- Monangan, M.B., K.O. Geddes, K.M. Heal, G. Labahn, and S.M. Vorkoetter, 1998, "*Maple V - Programming Guide*," Springer, New York.
- Newland, D.E., 1993, "*An Introduction to Random Vibrations, Spectral & Wavelet Analysis - Third Edition*," Longman Scientific and Technical, New York.
- Robertson J.S., 1996, "*Engineering Mathematics with Maple*," Mc Graw Hill, Inc., New York.
- Schwartz, D.I., 1999, "*Introduction to Maple*", E-Source, Prentice Hall, Upper Saddle River, New Jersey.
- Spiegel, M. R., 1971 (second printing, 1999), "*Schaum's Outline of Theory and Problems of Advanced Mathematics for Engineers and Scientists*," Schaum's Outline Series, McGraw-Hill, New York.
- Tinker, M. and R. Lambourne, 2000, "*Further Mathematics for the Physical Sciences*," John Wiley & Sons, LTD., Chichester, U.K.
- Tveito, A. and R. Winther, 1998, "*Introduction to Partial Differential Equations - A Computational Approach*," Texts in Applied Mathematics 29, Springer, New York.
- Urroz, G., 2000, "*Science and Engineering Mathematics with the HP 49 G - Volumes I & II*", www.greatunpublished.com, Charleston, S.C.