

Introduction to Maple Versions 8 to 12

Information in this document was compiled and updated by Angie Gardiner from:
Multivariate Mathematics with Maple by James A. Carlson and Jennifer M. Johnson, copyright 1996
Introduction to Maple V.4 in the Undergraduate Computer Lab by James A. Carlson and Jennifer M. Johnson, 1993
Math 2250 Maple Tutorial by Nick Korevaar

What is Maple?

Maple is a general purpose software package containing tools useful in the study of a wide variety of problems. We can use Maple like a calculator to carry out numerical computation, or we can use the programming features of Maple to produce a more complicated sequence of computations. In addition to numerical computation, we can do symbolic or algebraic computations in Maple. This software package has built-in functions for solving many kinds of equations, for working with matrices and vectors, and for differentiation and integration. It also contains functions which allow us to create visual representations of curves and surfaces from their mathematical descriptions.

Using a software package like Maple gives new ways to think about problems. For example, we can use the computer to run numerical experiments. This means that we can not only look at more realistic versions of standard problems, but we can easily look at many more examples, in the process developing a better intuitive feel for the essential features of a general problem and its solutions. Often these experiments suggest further questions or interpretations that might otherwise go unnoticed. Patterns that appear in these experiments give insight into and motivation for the standard approaches to these problems - and frequently highlight the limitations of these methods.

In many situations the graphics features of Maple are a useful tool for visualizing the geometric ideas behind important definitions and standard techniques. Again, this goes a long way toward giving a better feeling for the essentials.

Before you start learning Maple, it would be helpful if you familiarize yourself with the basics of the computer lab in the T. Benny Rushing Mathematics Student Center (if you will be working here). Please refer to "Introduction to the Computer Lab" for this information.

The Basics

Starting Maple. If you are using a UNIX station in our lab, either move the cursor into the local window and type **xmacle &**, use the middle mouse button on the background to get a menu from which you can select **X Maple V12** (or another version), or click on the maple leaf icon at the bottom of the screen. Soon the ghost of a window will appear. Move the mouse to position this window, then click with the left mouse button when satisfied. If you are using a Mac in our lab, click on the icon at the upper right corner of your screen labeled **/**. This will open a new window. Click on the **Applications** button in this window, and then select **XDarwin** from the applications and run it **rootless**. After a couple of minutes, you will get a terminal window. In this window, type **ssh sunfire**. Now you can type **xmacle** at the prompt to start Maple. Once Maple opens, you can use the buttons in the upper right corner of each window to resize the windows if you like.

Saving a file. To save a Maple session that you have been working on and give it a name, choose **File** -> **Save As** from the menus at the top of the screen. A **Save As** window will appear. If you want to save your file under the name **hwk1** (for example), enter **hwk1** in the **Filename** box. If there is already a ***.mws** in the box, make sure that you delete the *****. Then hit the **Return** (Macs) or **Enter** (UNIX) key, or click on **OK**. It is best to keep your file names simple, using only letters and numbers, without any special symbols. In particular, it is a good idea to avoid using spaces or periods. Maple will add the **.mws** or **.mw** extension to identify the saved file as a **Maple worksheet**. Be careful about reusing a file name from a previous session. For example, if last week you made a file named **Lines.mws** and this week you open a new Maple session and call it **Lines.mws** also, this will ERASE the file you created

last week. Notice that the **Save As** window lists the Maple files in your current directory, so you can see which names you have already used. Once you have given your file a name, you can do **File -> Save**, or use the save icon at the top of the window. It is a good idea to save your files fairly often, so that you won't lose your work if something goes wrong.

Copy/Paste. Often you will want to re-enter a command that is a small variation on an earlier command. In that case, instead of retyping you can copy an old command and then paste it onto a new command line. Then you can move the cursor into the line and make modifications. If you are using a Mac, use the mouse to highlight the command(s) you want to copy, then choose **Edit -> Copy** from the menus. Move the cursor to where you want to paste the command(s) and choose **Edit -> Paste**. If you are using a UNIX station, you can use the copy/paste menu choices, or you can copy/paste with the mouse as follows. Press and hold down the left mouse button at the beginning of the text you want to copy. Still holding the button down, drag the mouse until you have highlighted the desired text. Release the mouse button. Move the cursor to the position where you want to paste. Click the left mouse button to mark this spot. Click the middle mouse button to paste.

You can paste repeatedly - the copied text stays in memory and anytime you press the middle mouse button (or choose **Paste** from the **Edit** menu) it will be pasted at the current cursor position. (This can cause some problems for you if you accidentally hit the middle mouse button later on, so you may want to copy some blank space to clear the copied text.)

Editing your Maple file. It may take more than one session at the computer to complete your analysis. You will need to save your work, and come back to the lab and open your Maple file again to continue working. Often it will be useful to take with you a printed copy of your Maple session so you can think about what you have done so far, and how you want to continue when you come back to the lab, or so that you can talk to your instructor about any questions you may have.

Before you learn how to print your Maple session, you should learn how to clean it up a bit. For example, to delete a computation or a plot, just move the cursor into the command that produced the computation or click on the plot. Then while holding down the **Control** key, press the **Delete** key. You could also discover this by looking at the Edit menu. You can learn the key commands for editing from the menus. You can also delete text or commands by using the mouse to highlight the area you want to delete, then hitting the **Delete** key.

It is also a good idea to include some explanations with your work. There are two ways to do this. First, you can use comments. In any command line, Maple ignores anything typed after the **#** symbol. So inserting a line like

```
> solve ( 3*x + 2*y - 5*z, z); # Get z in terms of x and y
```

makes your work much easier to follow.

If you want to add explanatory comments between commands, move the cursor into the first command. Then click on the icon at the top of the screen containing a Maple prompt **>**. A new command line should appear and the cursor will be on this new line. To enter text instead of Maple commands, click on the first toolbar icon labeled **T**. Now you can type your explanation, and it will appear as ordinary text. Notice the pull-down menus and buttons at the top of the Maple window (when you are in text mode) that allow you to control the size and style of the text you enter.

Printing. To print your Maple session directly from Maple, choose **Print** from the **File** menu (or click on the Printer icon). In the Printer Setup window that appears, make sure that **Print Command** is

selected, rather than Output to File, and click **Print**. (If the default is the printer, to print to a file, choose the box marked **Print To File** on the General tab of the print dialog box. Some maple versions don't have the feature.) The default print command (**lpr**) should send a printout of your Maple worksheet to the printer. If only part of your worksheet is printing, try changing the print command to **lpr -l** or **lpr -oraw**.

Alternatively, you can save your Maple session to a postscript file, then print that file from the local window. To do this, in the Printer Setup window choose **Output to File** and click **Print**. This will create a postscript file from your Maple worksheet and will store it with the name **hwk1.ps** (if your Maple session had been named **hwk1.mws** as in the example above). To send this file to the printer, go to the local window and type **print hwk1.ps**, then press Enter. If you are printing from the local window, make sure that you are printing a **.ps** file. The file types a **.mws** file nor a **.pdf** can only be printed from their respective applications, maple and acroread. Do **NOT** use **print** on these file types.

*** If you are having trouble printing, please ask the lab assistant for help.***

Coming back for more. When you come back to work on a Maple file that you've already started, your task now is to reopen that file and get back to where you left off. To open a file, choose **File -> Open**. An **Open File** window will appear. In this window you will see a list of Maple files. Click on the file that you want to open, then press the **Enter** key or click **OK**.

Your Maple window should now look like it did the last time you saved your file. However, you are not quite ready to continue working. Although Maple displays a record of your last session, it does not remember what happened last time. One way to get Maple to remember what happened last time is to go to the beginning of your file and hit the **Return** key until you get to the last line of the file. (You might want to use the arrow keys to skip the lines that displayed plots or requests for help files). This will ensure that Maple's memory of your last session is restored. A shortcut for executing the entire worksheet is to select **Edit -> Execute -> Worksheet**.

Using Maple

The best way to learn Maple is by using it. But first, a few remarks about Maple. The symbol **>** is the command prompt, which Maple uses to signal you that it awaits your command. Commands normally end with a semicolon (if you want to see the output) or a colon (if you don't want to see the output). Maple is much fussier about rules of punctuation, grammar, and spelling than are humans. If something is not working right, check to see if you are following the rules. For example, Maple will get confused if you say **2x** instead of **2*x**. Check for things like misspelled names or extra or missing parentheses. If further thought doesn't clear things up, ask a human for help. You will soon become an expert troubleshooter.

While learning Maple, you will often have questions about how a particular command or function is used. Fortunately, Maple can help you. To ask about a command whose name you know, just type a question mark, followed by the name of the command. Thus

```
> ?solve
```

gives information on the **solve** command. You will probably find the examples at the end more useful than the technical information at the beginning of the help file. If you don't know the name of the command you want to use, try doing a full text search or topic search of Maple's help files. You will find these options under the **Help** menu in the upper right corner of the Maple window. You will

also find other handy things, such as **Introduction**, **New User's Tour**, and **Using Help**.

Let's get started! Go ahead and type along with these examples.

```
> 2 + 2; 3*5; 6-2:
```

All three computations were done, although only two results are shown (the `:` at the end of a command suppresses the output). If you forget the semicolon, go ahead and put it on the next line:

```
> 3*5
```

```
> ;
```

Addition `+`, subtraction `-`, and division `/` are standard, and parentheses are used in the usual way. An asterisk `*` indicates multiplication and a caret `^` is used for powers:

```
> ( 1 + 2 ) * ( 6 + 7 ) - 12 / 7;
```

```
> 3^(2.1);
```

Whenever possible, Maple tries to compute exact quantities. Our first command gives its answer as a fraction, rather than as a decimal, contrary to what you might expect. The second command gives a decimal, or "floating point" answer because we used this form in our question. To force Maple to give results in floating point (decimal) form, use **evalf**:

```
> Pi;
```

```
> evalf(Pi);
```

```
> exp(1); # the number e
```

```
> evalf(%);
```

The `%` sign stands for the most recently computed quantity.

Be aware that Maple distinguishes upper-case letters from lower-case. Thus **evalf(pi)** is not the same as **evalf(Pi)**. The function **evalf** can take a second (optional) input which determines the precision of the output.

```
> evalf(Pi, 50);
```

For the most part, spacing is unimportant in Maple. In the commands above, spaces could be omitted or added without causing any problems. However, thoughtful use of spacing makes Maple code easier to read, and so easier to understand and, if necessary, to correct.

Standard mathematical functions can be used in Maple so long as we know their names. To compute the quantity

$$|-14| + \sin(1) - \sqrt{2} + e^{\cos(1.6\pi)} + \arctan(3)$$

use

```
> abs(-14) + sin(1) - sqrt(2) + exp( cos(1.6*Pi) ) + arctan(3);
```

```
> evalf(%);
```

This example illustrates another important point. The correct form of a Maple expression can often be found by intelligent guessing. Thus `tan(45)` does indeed compute the tangent, and `20!` computes a factorial. If your first guess doesn't work, use one of Maple's help features to get more information.

In addition to the functions in the Maple library, there are specialized "packages" of functions that can be read into the working memory as needed, using the command `with`. For example, to work with matrices and vectors you must load the linear algebra package. You do this by typing

```
> with (linalg);
```

This command produces a list of all the functions in this package and gives you access to them in your current Maple session. If you close Maple and reopen it later, you must reload any special packages you want to use. Once you become familiar with a package, you will prefer to load it using a colon

```
> with (linalg):
```

rather than a semicolon. Again, this gives you access to the linear algebra commands, but without listing all their names. Some packages that might be of interest to you are `plots`, `DEtools`, and `student`.

Algebra

Maple knows about variables and algebra. Consider, for example, the expression $(a + b)^2$. The commands

```
> (a + b)^2;
```

```
> expand (%);
```

give the expanded form, and

```
> factor (%);
```

brings us back to our starting point. To make long computations easier and more intelligible, we can assign values to variables using `:=`

```
> p := (a + b)^2; b := 1; p;
```

In these examples, variables store an expression or a number. Variables can store almost anything, for example, a list of points, an equation, a set, a piece of text, or a function definition:

```
> pts := [ [1,2], [3,4] ];
```

```
> eqn := 2*x - 3*y = 5;
```

```
> eqns := { 2*x - 3*y = 5, 5*x - 3*y = 1 };
```

```
> tag := `The nth partial sum is`; #backquotes
```

```
> print(pts, eqn, eqns, tag); #checks our assignments
```

```
> f := x -> x^2; #defines a function
```

```
> f(2); f(3);
```

Anything we can define or compute in Maple can be assigned to a variable for future reference using

" := ". Remember that it is different from the symbol =, which is used to test equality. Observe that no space is allowed between the := and the = in an assignment statement.

Variables can be returned to their original symbolic (unassigned) state. The commands

```
> b := 'b';  
> p;
```

first remove the value 1 assigned to the variable b above and then display the updated value of p. Similarly

```
> unassign ( 'pts', 'eqn', 'eqns', 'tag');  
> print (pts, eqn, eqns, tag);
```

clears the variables assigned above. A more drastic way of clearing Maple's memory is to say

```
> restart;
```

This command clears ALL variables and unloads all packages. So, if you need one later, you must reload it using **with**.

Pay special attention to the kind of quotes used in examples. The possibilities are the single quote ', the backquote ` , and the double quote ". They all play different roles.

Here is an extended example of how to use variables and assignment statements:

```
> F := m*a;           # Newton's formula for force  
> m := 2.1; a := 5;   # set the mass and acceleration  
> F;                 # compute the force  
> a := 21.9;         # reset the acceleration  
> F;                 # recompute force  
> a := 'a';          # clear a  
> F;                 # recompute F
```

The **subs** command lets us make temporary substitutions in an expression as opposed to assigning values. For example, try this:

```
> g := (a+1)^2 / (b-1)^3 + a / (b-1);  
> simplify (g);  
> subs( a=3, b=2, g);
```

or this

```
> subs( a = x+y, b = x+1, g);  
> simplify(%); a; b;
```

Notice that the variables **a** and **b** were not permanently assigned a value.

Graphing

Standard Coordinates

Maple can construct many kinds of graphs, a feature that you can use to visualize mathematical objects and processes. The command

```
> plot( sin(3*x), x = -Pi..Pi );
```

produces a plot containing the curve $y = \sin(3x)$ for x in the interval from $-\pi$ to π . No space is allowed between the dots in a plot command. Notice that the scale on the x-axis is not the same as on the y-axis. To fix this you can say

```
> plot( sin(3*x), x = -Pi..Pi, scaling = constrained );
```

Sometimes it is useful to restrict the range over which y varies. We get a misleading graph from

```
> plot( tan(x), x = -5..5 );
```

It does not accurately represent the vertical asymptotes of $y = \tan(x)$. Better results are obtained with

```
> plot( tan(x), x = -5..5, y = -5..5 );
```

We can plot several curves at once. To plot $y = \sin(x)$ and $y = \sin(3x)$ together we say

```
> plot( { sin(x), sin(3*x) }, x = -Pi..Pi );
```

Now the first argument of **plot** is a set of expressions to be graphed. Sets are enclosed in curly braces and individual items are separated by commas.

We could also use the **display** command found in the **plots** package:

```
> with(plots);
```

```
> plot1 := plot( x^2, x = 0..4 );
```

```
> plot2 := plot( 3*x^2 - 2, x = 0..4 );
```

```
> display( { plot1, plot2 } );
```

Parametric Equations

A curve in the plane can be described as the graph of a function, as in the graph of $y = \frac{1\sqrt{4-x^2}}{2}$ for x in the interval from -1 to 1 , or it can be given parametrically as in $(x(t), y(t)) = (2 \cos(t), \sin(t))$ for t in the interval from 0 to π . Often we interpret such a curve as the path traced by a moving particle at time t . Use the **plot** command to draw a curve from this parametric description:

```
> plot( [2*cos(t), sin(t), t = 0..Pi] );
```

The resulting graph is somewhat distorted, because Maple did not use the same vertical and horizontal scales. There are two ways to fix this. Use

```
> plot( [2*cos(t), sin(t), t = 0..Pi], -2..2, -2..2 );
```

or use the option **scaling = constrained**.

Polar Coordinates

Polar plots are a special kind of parametric plot. The polar coordinates (r, θ) of points on a curve can be given as a function of some parameter t . In many cases the parameter is just the angle θ . Consider the ellipse defined in standard coordinates by $x^2 + 4y^2 = 4$. To find an equation relating the polar coordinates r and θ of a typical point on this ellipse, we make the substitution $x = r \cos(t)$ and $y = r \sin(t)$, where $t = \theta$.

```
> subs( x = r*cos(t), y = r*sin(t), x^2 + 4*y^2 = 4 );
simplify( % );
solve( %, r );
```

We find that the ellipse is the collection of points whose polar coordinates (r, θ) satisfy

$r^2 = \frac{4}{4 - 3 \cos(\theta)^2}$, and the following commands draw the right half of the ellipse:

```
> r := 2/sqrt( 4 - 3*cos(t)^2 );
> plot( [ r, t, t = -Pi/2..Pi/2 ], coords = polar );
```

Here are some more of examples of polar plots that you can try:

```
> plot( [ 1, t, t = 0..2*Pi ], coords=polar );
> plot( [t, t, t = 0..2*Pi], coords=polar );
> plot( [ sin(4*t), t, t = 0..2*Pi], coords=polar );
```

As usual, a more realistic picture is obtained with **scaling = constrained**.

Plotting Data

Maple can plot data consisting of pairs of x and y values. For example, if we say

```
> data := [ [0, 0.53], [1, 1.14], [2, 1.84], [3, 4.12] ];
```

then **data** refers to a sequence of five points, $(x, y) = (0, 0.53)$, etc. The result is a list: something enclosed in square brackets, with elements separated by commas. Lists are used for collections of objects where the order matters. Individual items are accessed this way:

```
> data[1]; data[2]; data[3]; data[4];
```

In this case, the list items are themselves lists - very short ones made up of the coordinates of a point. To access the second coordinate of the third data point we say

```
> data[3][2];
```

To plot the points in our list we use commands like

```
> plot( data, style = point, symbol = circle, color = black);
> plot( data, style = line, view = [0..4, 0..5] );
```

You could also try

```
> plot( data, style = line, title = `Experiment 1` );
```

Here the backquote ` is used to specify a plot title. See `?plot[options]` for more information, e.g., about symbols and line styles available. You could also try `?readdata` and `?stats` to find out how to read a file of data points into a Maple session.

Solving Equations

Maple can also solve equations. Consider some examples:

```
> solve( x^2 + 3*x = 2.1 );
```

```
> solve( x^3 + x = 27 );           #This gives a complicated answer!
```

```
> solve( x^3 + x = 27.0 );
```

The second command gives an exact, but complicated, answer. Replacing 27 by 27.0 forces Maple to give decimal approximations instead, as do the commands

```
> fsolve( x^3 + x = 27 );
```

```
> fsolve( x^3 + x, x, complex);
```

In general, `solve` looks for exact answers using algebraic methods, whereas `fsolve` uses numerical methods to find approximate solutions in floating-point form. Compare

```
> solve( tan(x) - x = 2 );
```

```
> fsolve( tan(x) - x = 2 );
```

Notice how Maple responds if it cannot find the solution you asked for. Also, notice that `fsolve` may not find all solutions. To understand why not, it is helpful to look at a graph

```
> plot( { tan(x) - x, 2 }, x = 0..10, y = -10..10 );
```

This will give you an idea of how many solutions there are and what their approximate location is.

Then give `fsolve` a range of x -values in which to search:

```
> fsolve( tan(x) - x = 2, x = 4..5 );
```

Often we need to use the solution of an equation in a later problem. To do this, assign a name to it. Here is one example.

```
> r := solve( x^2 + 3*x - 2.1 = 0 );
```

Notice that the answer has the form $r1, r2$, where $r1$ is the first root and $r2$ is the second. Such an object - a bunch of items separated by commas, is called an expression sequence. One picks out items of an expression sequence this way:

```
> r[1];
```

```
> r[2];
```

Here are some computations with items from an expression sequence:

```
> r[1] + r[2];           # sum of the roots
> r[1]*r[2];           # their product
> subs( x = r[1], 2*x + 3 ); # find 2(first soln) + 3
```

We can also solve systems of equations:

```
> solve( { 2*x + 3*y = 1, 5*x + 7*y = 2 } );
> x; y;
```

A system of equations is given as a *set* - a bunch of items enclosed in curly brackets and separated by commas. Sets are often used when the order of the objects is unimportant. In reply to the **solve** command above, Maple tells us how to choose x and y to solve the system, but it does not give x and y these particular values. To force it to assign these values, we use the **assign** function:

```
> s := solve( { 2*x + 3*y = 1, 5*x + 7*y = 2 } );
    assign( s );
    x; y;      #check that it worked
```

A note: To type multiple line commands, as above, use Shift-Return. *And a warning:* You may have trouble later if you leave numerical values assigned to the variables x , y , and r . Maple will not forget these assigned values, even though you have gone on to a new problem where x means something different. It is a good idea to return variables to their unassigned state when you finish your problem. Recall how to do this:

```
> x := 'x'; y := 'y'; r := 'r';
```

Recall that **restart** also clears all variables.

Finally, note that symbolic parameters are allowed in **solve** commands. However, in that case we have to tell Maple which ones to solve for and which ones to treat as unspecified constants:

```
> solve( a*x^2 + b*x + c, x );           # solving for x
> solve( a*x^3 + b*x^2 + c*x + d, x ):
> solve( { a*x + b*y = h, c*x + d*y = k }, { x,y } ): #
    solving a system for x and y
```

Functions

Although Maple has a large library of standard functions, we often need to define new ones. For example, to define

$$p(x) = 18x^4 + 69x^3 - 40x^2 - 124x - 48$$

we say

```
> p := x -> 18*x^4 + 69*x^3 - 40*x^2 - 124*x - 48;
```

Think of the symbol \rightarrow as an arrow: it tells what to do with the input x , namely, produce the output $18x^4 + 69x^3 - 40x^2 - 124x - 48$. Once the function p is defined, we can do the usual computations with it, e.g.,

```
> p(-2);  
> p( 1/2 );  
> p( a+b );  
> simplify(%);
```

It is important to keep in mind that functions and expressions are different kinds of mathematical objects. Mathematicians know this, and so does Maple. Compare the results of the following:

```
> p;      # function  
> p(x);  # expression  
> p(y);  # expression  
> p(3);  # expression
```

As further proof, try the following:

```
> factor(p);  
> factor( p(x) );  
> plot( p, -2..2 );  
> plot( p, -2..2 );  
> plot( p(x), x = -2..2 );
```

Functions of several variables can be defined as easily as can functions of a single variable:

```
> f := (x,y) -> exp(-x) * sin(y);  
> f(1,2);  
> g := (x,y) -> alpha*exp(-k*x)*sin(w*y);  
> g(1,2);  
> alpha := 2; k := 3; g(1,2);  
  
> w := 3.5; g(1,2);  
> alpha := 'alpha'; g(1,2);
```

Calculus

Derivatives

To compute the derivative of the expression $x^3 - 2x + 9$, say

```
> diff( x^3 - 2*x + 9, x );
```

To compute the second derivative, say

```
> diff( x^3 - 2*x + 9, x, x );
```

or alternatively,

```
> diff( x^3 - 2*x + 9, x$2 );
```

This works because Maple translates the expression `x$2` into the sequence `x, x`. By analogy, `x$3` would give the third derivative. Thus one can easily compute derivatives of any order. Now suppose that we are given a function g defined by

```
> g := x -> x^3 - 2*x + 9;
```

It seems natural to use

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

to get the derivative of g . However, Maple expects an expression in x and so interprets g as a constant, giving the wrong result. The command

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

which uses the expression $g(x)$, works correctly. The subtlety here is an important one: `diff` operates on expressions, not on functions - g is a function while $g(x)$ is an expression. To define the derivative of a function, use Maple's `D` operator:

```
> dg := D(g);
```

The result is a function. You can work with it just as you worked with g . Thus you can compute function values and make plots:

```
> dg(1);
```

```
> plot( { g(x), dg(x) }, x = -5..5, title = `g(x) = x^3 - 2x + 9  
and its derivative` );
```

Partial Derivatives

Maple can compute partial derivatives:

```
> q := sin(x*y);      # expression with two variables
```

```
> diff( q, x );      # partial with respect to x
```

```
> diff( q, x, y );   # compute d/dy of dq/dx
```

As in the one variable case, there is an operator for computing derivatives of functions (as opposed to expressions):

```
> k := (x,y) -> cos(x) + sin(y);
```

```
> D[1](k);          # partial with respect to x
```

```
> D[2](k);          # partial with respect to y
```

```
> D[1,1](k);        # second partial with respect to x
```

```
> D[1,2](k);        # partial with respect to y, then x
```

```
> D[1](D[2](k));    # same as above;
```

Integrals

To compute integrals, use `int`. The indefinite integral (antiderivative) $\int x^3 dx$ is given by `int(x^3, x)`. The following examples illustrate that Maple knows integration by parts, substitution, and

partial fractions:

```
> int( 1/x, x );  
> int( x*sin(x), x );  
> int( sin(3*x + 1), x );  
> int( x/ (x^2 - 5*x + 4), x );
```

Nonetheless, Maple can't do everything:

```
> int( sin( sqrt(1-x^3)), x );
```

The last response echoed back the indefinite integral, a signal that Maple does not know how to find an antiderivative for $\sin(\sqrt{1-x^3})$ in terms of elementary functions (the ones built from addition, subtraction, multiplication, division, powers, roots, logarithms and exponentials, trig functions and their inverses). In fact, it can be proved that no such antiderivative exists. Thus, even a smarter Maple would not help.

To compute definite integrals like $\int_0^1 x^3 dx$ we say `int(x^3, x = 0..1)`. Note that the only difference is that we give an interval of integration. Let us return to the integral $\int \sin(\sqrt{1-x^3}) dx$, which Maple could not evaluate. We can, however, ask Maple to find a numerical value for the definite integral:

```
> int( sin( sqrt(1 - x^3) ), x = 0..1 );  
> evalf(%);
```

The second command forces Maple to apply a numerical method to evaluate the integral. Of course, you could also put everything on one line:

```
> evalf( int( sin( sqrt(1 - x^3) ), x = 0..1 ) );
```

Numerical Integration

Another approach to numerical integration is to use the `student` package.

```
> with( student ):           # load the package  
  
> j := sin( sqrt(1 - x^3) ); # define the integrand  
  
> trapezoid( j, x = 0..1 );  # apply trapezoid rule  
  
> evalf(%);                 # put in decimal form
```

By default the `trapezoid` command approximates the area under the graph of $\sin(\sqrt{1-x^3})$ with four equal trapezoids. For greater accuracy use more trapezoids, i.e., a finer subdivision of the interval of integration:

```
> evalf( trapezoid( j, x = 0..1, 10 ) );
```

Better yet, use a more sophisticated numerical method like Simpson's rule:

```
> evalf( simpson( j, x = 0..1 ) );
```

Only an even number of subdivisions is allowed, as in `simpson(j, x = 0..1, 10)`.

The `student` package is well worth exploring. Among other things it has tools for displaying figures which explain the meaning of integration:

```
> leftbox( j, x = 0..1, 10 );
```

The area of the figure displayed by `leftbox` is computed by `leftsum`:

```
> evalf( leftsum( j, x = 0..1, 10 ) );
```

One can also experiment with `rightbox` and `middlebox` and their companion functions `rightsum` and `middlesum`.

Multiple Integrals

Let R be the rectangular region defined by x between 0 and 1, and y between 0 and 1. To integrate $\iint (x^2 + y^2) dx dy$ over R in Maple, we compute the repeated integral. Here is one way to do this:

```
> int( x^2 + y^2, x = 0..1 );
```

```
> int( %, y = 0..1 );
```

The first command integrates with respect to the x variable, producing an expression in y . The second command integrates the result with respect to y to give a number.

Other Calculus Tools

Limits:

```
> g := x -> (x^3 - 2*x + 9)/(2*x^3 + x - 3);
```

```
> limit( g(x), x = infinity );
```

```
> limit( sin(x)/x, x = infinity );
```

```
> limit( sin(x)/x, x = 0 );
```

Taylor expansions and sums:

```
> taylor( exp(x), x = 0, 4 ); # expansion around x = 0
```

```
> sum( i^2, i = 1..100 ); # be sure i is unassigned
```

```
> sum( x^n, n = 5..10 ); # needs n to be unassigned
```

```
> sum( 1/d^5, d=1..infinity ); evalf(%);
```

Differential Equations

Maple can solve differential equations for you. One way is to use `dsolve`:

```
> deq := diff( y(x), x$2 ) + y(x) = 0;
```

```
> dsolve( deq, y(x) );
```

Try `?dsolve` for more information. We can specify initial conditions and experiment with parameters.

```
> de := diff( y(x), x ) = y(x) - sin(x);
```

```
> sol := dsolve( {de, y(0) = 0.1}, y(x));
```

```
> sol;
```

To graph the solution, we need only the right-hand side of `sol`, which we can get by

```
> plot( rhs(sol), x = 0..10);
```

Now let's try to make the slope field and draw some trajectories. We can get the slope field with the `dfieldplot` command from the `DEtools` package:

```
> with(DEtools);
```

```
> dfieldplot( de, y(x), x = -3..3, y = -3..3, arrows = line,  
  dirgrid = [30,30]);
```

Now let's use another command from the `DEtools` package to simultaneously plot the slope field and a few trajectories:

```
> DEplot( de, y(x), x = -3..3, { [y(0)=0], [y(0)=1], [y(0)=3] }, y  
  = -3..3, color='black', linecolor='red', arrows='line', dirgrid=  
  [30,30]);
```

Vector and Matrix Operations

To work with vectors and matrices in Maple, first load the linear algebra package:

```
> with(linalg):
```

Define and display a vector like this:

```
> v := vector( [1, -1] );
```

```
> v[1]; v[2]; v;
```

```
> print(v);
```

Note that Maple treats vectors as columns, even though it displays them as rows. Now define another vector `w` and do some simple computations:

```
> w := vector( [1, 1] );
> matadd( v,w );      # vector or matrix addition
> dotprod( v,w );
```

Next we define two matrices:

```
> A := matrix([ [2,3], [1,2] ]);
> B := matrix([ [1,1], [0,1] ]);
```

It is easy to make new matrices (or vectors) from old ones as in

```
> augment( A, v, w );
```

Next, we do some computations with our matrices **A** and **B** and our vectors **v** and **w**:

```
> matadd( A, B );      # or use evalm( A+B );
> scalarmul( A, 2 );  # or use evalm( 2*A );
> multiply( A,B );    # or use evalm( A &* B );
> multiply( A, v+w ); # or use evalm( A &* (v+w) );
```

Finally, we can change the individual entries of a matrix with commands like:

```
> A[1,1] := 5; B[2,1] := 1;
> evalm(A); evalm(B); # check
```

Maple can compute inverses and transposes:

```
> inverse(A);      #or use evalm( A^(-1) );
> transpose(A);
```

It can also compute determinants:

```
> det(A); det(B);
> det( A+B ); det(A) + det(B);
```

Symbolic matrices are as legitimate as numerical ones:

```
> A := matrix([ [a,b], [c,d] ]);
> det(A);
```

Let

```
> C := matrix([ [3,2,2], [3,1,2], [1,1,1] ]);
```

We can put **C** into row-reduced form using

```
> gausselim(C);
```

or

```
> gaussjord(C);
```

There are also commands for doing elementary row and column operations on a matrix: `addrow`, `swaprow`, `mulrow`, etc. You can find a list of functions in the help files for the `linalg` package.

Troubleshooting

Although errors in Maple can be quite confusing at first, you will quickly gain the experience needed to understand and fix them. Below is a list of common errors to consider when troubleshooting.

1. Do statements end with a semicolon or a colon?
2. Are parentheses, braces, brackets, etc., balanced? Code like `{ x,y }` is good, but `x,y` will cause trouble.
3. Are you trying to use something as a variable to which you have already assigned a value? To "clear" a variable x , say `unassign('x')`. You can clear many variables at once, e.g., `unassign('x', 'y')`. You will not have to reload any packages after using the unassign command. Remember, a variable that has a value assigned to it no longer can function as a variable. To display the value of an ordinary variable, type its name, followed by a semicolon, e.g.,

```
> x;
```
4. If things seem hopelessly messed up, issue the command `restart`; You will have to reload any needed packages after this.
5. Are you using a function when an expression is called for, or vice versa? Remember `g := x^2 + y^2` defines an expression while `f := (x,y) -> x^2 + y^2` defines a function. It makes good sense to say `f(1,3)` but not so much sense to say `g(1,2)`.
6. Are you using `=` when `:=` is called for? Remember, the first tests for equality while the second assigns a value to a variable.
7. Are you distinguishing between the three kinds of quotes? They are: `"` the double quote, `'` the single quote, and ``` the backquote.
8. If you use a function in a package, load the package first. Thus, to use `matrix`, load `linalg`; to use `display`, load `plots`. You load `linalg` by the command `with(linalg)`.
9. Remember to use `?` to inquire about the details of a Maple function, e.g., `?plot` for information about plot or just `?` for general information.
10. If all of the licenses for the current version of Maple are being used, you can use a different version by calling it up by name, i.e. by typing `xmapleV8` or `xmapleV11` in a terminal window.

If you want to see more examples of Maple code, there is an abundance of information on the web. A good place to start is www.mapleapps.com, but you will also find many good sites by using a search engine. We also have a Maple V Learning Guide that you can check out for use in the Undergraduate Computer Lab. Even though it is for an older version of Maple, it still has a lot of good information.

