Multivariable Mathematics with Maple

Linear Algebra, Vector Calculus and Differential Equations

by James A. Carlson and Jennifer M. Johnson

Introduction

The aim of this book, intended as a companion to a traditional text, is to explore the notions of multivariable calculus using a computer as a tool to help with computations and with visualization of graphs, transformations, etc. The software tool we have chosen is Maple; one could as easily have chosen Mathematica or Matlab. In some cases the computer is merely a convenience which slightly speeds up the work and allows one to accurately treat more examples. In others it is an essential tool since the necessary computations would take many minutes, if not hours or days. We will, for example, use Maple to study the temperature distribution in a thin flat plate by reducing the problem to the solution of a system of, say, one hundred equations in one hundred unknowns, then using the resulting numerical data to construct a contour plot which shows lines of equal temperature. All this could be done by hand, but it would be laborious work indeed. Such problems would be out of reach without tools for computation and visualization.

Difficult computations and fancy pictures are, of course, not ends in themselves. We must understand the underlying mathematics if we are to know which computations to do and which pictures to draw. Likewise we must develop our own intellectual tools sufficiently well in order to understand, interpret, and make use of the data and images that we "compute." Thus our focus will always be on the mathematical ideas and their applications. The role of Maple is to more vividly illustrate them and to widen the range of problems that we can successfully solve.

To get the most from this book, the reader should work through the examples and exercises as they occur. For example, when the text mentions the snippet of Maple code

```
> plot( cos(x) - (1/3)*cos(3x), x = -2*Pi..2*Pi );
```

the reader should try it out at his or her machine. This particular bit of Maple will plot the graph of $y = \cos x - (1/3)\cos 3x$ on the interval $-2\pi \leq x \leq 2\pi$.

Most chapters can be read independently of the others. However, it is best to first work through a good part of Chapter One. It is a brief guide to the essentials of Maple: how to do algebraic computations, solve simple equations, compute derivatives and integrals, and make graphs. It also contains an introduction to programming in Maple, e.g., how to do repetitive computations using loops, and how to define new functions.

The great majority of the problems in the text can be solved with just a few lines of Maple, like the one above for plotting a graph. Occasionally, however, a paragraph or two of "code" is required. As an alternative to typing these in, we have made them available from the web pages at

`http://www.math.utah/books/calc2-maple/`

You are free to copy any of the code you find there.

As you work with Maple you will sometimes find that things don't work as you expect. The usual cause of this misbehavior is that computers, unlaik humans, canit unstond stautements mud with less than purfict spelling, punctuation, grammar and logic. If Maple does not respond or responds with nonsense, carefully check your work. If it is an example in the book, compare what you have typed with what is written. Take special care with the placement of punctuation marks like colons and semicolons and the three kinds of quotation marks — single `'`, double `"`, and backquote `` ` ``. If this fails, take a look at the troubleshooting section at the end of Chapter One, or consult someone with a bit more experience.

It is always important to think about whether the results of a computation make sense. Errors in your logic or quirks in Maple's thought processes may give wrong or incomplete answers. The best way to avoid such pitfalls is, as always, to understand what you are doing.

The authors would like to thank the members of the Calculus II classes they have taught at the University of Utah for the past three years during the preparation of this book, particularly Susan Pollock. The Mathematics Department has been generous in its support, and we are grateful to faculty members Mladen Bestvina, Gerald Davey, Les Glaser, Grant Gustafson, János Kollár, Nick Korevaar, Domingo Toledo, and Andrejs Treibergs for their suggestions and assistance. Special thanks are due Drs. Nelson Beebe, Paul Burchard, and Michael Spivak for their help at crucial points with the TeX macros.

Chapter 1
Introduction to Maple

The purpose of this first chapter is to give a rapid overview of how one can use Maple to do algebra, plot graphs, solve equations, etc. Maple can also compute derivatives and integrals, solve differential equations, and manipulate vectors and matrices. Much can be done with one-line computations. For example,

```
> expand((a + b)^3);
```

expands $(a+b)^3$ to $a^3 + 3a^2b + 3ab^2 + b^3$, while

```
> plot( cos(x) + cos(2*x) + cos(3*x), x = -Pi..Pi );
```

constructs the graph of the function $f(x) = \cos x + \cos 2x + \cos 3x$ on the interval $[-\pi, \pi]$, and

```
> solve( x^2 + 2*x - 5 = 0 );
```

solves the quadratic equation $x^2 + 2x - 5 = 0$.

The best way to learn Maple is by using it. Begin by trying the three examples above. The symbol `>` is the *prompt*, which Maple displays to signal you that it awaits your command. Commands normally end with a semicolon.

Computers are 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 `solve( x^2 + 2x - 5 = 0 )` instead of `solve( x^2 + 2*x - 5 = 0 )`. Check for things like misspelled names, extra or missing parentheses. If further thought doesn't clear things up, ask a human for help. You will soon become an expert troubleshooter.

Technical details on how to open the Maple program and how to save or print a Maple file depend heavily on your local system. Thus no information is given here on these important aspects of getting started. Consult your manual or local support staff if you need help.

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. Here are other things to try:

```
> ?
> ?intro
```

§1.1 A Quick Tour of the Basics

Below is a sample Maple session, in which we do some simple arithmetic:

```
> 2 + 2;
> quit
```

In this session you computed 2+2 by typing a one-line command next to Maple's command prompt `>`. This is where you type what you want Maple to compute. You then typed your *Return* (Unix system) or *Enter* (Mac version) to tell Maple to execute your command. Then you typed the command to quit. Alternatively, just select "quit" from the file menu.

Maple commands must be properly punctuated: they usually end with a semicolon. If you forget to type the semicolon, just put it on the next line:

```
> 2 + 2
> ;
```

has precisely the same effect as

```
> 2 + 2;
```

Addition, subtraction, etc. 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 we might expect. The second command gives a decimal or "floating point" answer because we used floating point forms in our question.

To force Maple to give results in floating point form, use `evalf`:

```
> Pi;  evalf( " );
> evalf( Pi );
> evalf( exp(1) );
```

The quote sign `"` or "ditto" 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. (Inputs, called independent variables in mathematics, are known as *arguments* in computer jargon.)

```
> evalf( Pi, 100 );
```

For the most part, spacing is unimportant in Maple. In the commands above, spaces could be omitted 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 quantities

$$\sqrt[3]{2} + 14 \cdot 8 + |-14| - \sin(1) \quad \text{and} \quad e^{\sin(1.6\pi)} + \sqrt{2} + \tan^{-1}(3):$$

use

```
> evalf( 2^(1/3) + 14*8 + abs(-14) - sin(1) );
> exp( sin(1.6*Pi) ) + sqrt(2) + 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(1)` does indeed compute the tangent, and `20!` computes a factorial. If your first guess does not work, use `?` together with your guess to get more information. For instance,

```
> arctangent( 1.0 );
```

produces only an echo, but

```
> ?arctangent
```

leads to the desired command and examples of its use. The query `?library` gives a listing of all the functions available.

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`. Try

```
> ?packages
```

to see what is available. For example, to work with matrices and vectors, you must load the linear algebra package. 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. In general, end Maple commands with a colon to prevent printing the results on screen. Other packages of interest are

```
> ?plots
> ?DEtools
> ?student
```

§1.2 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:

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

In these examples, variables store an expression or a number. Variables can store almost anything, e.g., 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 );            # check
> f := x -> x^2;                           # define a function
> f(2); f(3);                              # check definition
```

Anything we can define or compute in Maple can be assigned to a variable for future reference. The special symbol `:=` is read “gets”. Remember that it is different from the symbol `=`, used to test equality. *Confusing these two symbols can lead to hard-to-spot errors in your Maple code.* Observe that *no space* is allowed between the `:` and the `=` in an assignment statement.

The symbol `#` marks a comment. Maple ignores them, but they are useful for humans trying to read Maple code. On some machines, Maple allows comments in a multi-line command:

```
> f := x -> x^2;                 # define a function
  f(2); f(3);                    # check definition
```

but on other machines Maple ignores everything that appears between the first comment symbol `#` and the next command prompt `>`. On those machines Maple will not see the second line of commands `f(2)` and `f(3)`.

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 );         # check
```

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. Thus, 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.

Warning: As you work through these examples and exercises, you may get strange results because you forgot that you assigned a value to a variable. For example, if you work Exercise 4 below, you will probably assign the value 5 to the variable `r`. If you use `r` later in your Maple session to mean something else, say in a problem about polar coordinates, you must first clear it with `r := 'r'`. Recall that you can also use the `restart` command.

Exercise 1. Compute $(5/3)^{21}$ as an exact expression, as a decimal accurate to standard Maple precision, and as a decimal accurate to 20 digits precision. ◇

Exercise 2. Compute $(x+1)(x+2)(x+3)(x+4)$ in expanded form. ◇

Exercise 3. Find the coefficient of a^2b^7 in $(2a+3b)^9$. ◇

Exercise 4. Use Maple to define the formula for compound interest

$$A = P(1 + r/100)^n,$$

where P is the principal, r is the annual interest rate expressed as a percentage, n is the number of years the principal is invested, and A is the amount of capital (principal plus accumulated interest) at the end of n years. Set P to \$100, r to 5, and compute A. Then compute A with $n = 1, n = 5, n = 10$. (Your results will be more understandable in decimal form.) ◇

Exercise 5. Maple has a built-in function for computing partial fraction decompositions. Start with `?partialfractions` and follow the clues Maple gives to find out how to use this function. Use it to rewrite

$$\frac{x^2 - 2x + 5}{x^4 + 8x^3 - 10x^2 - 104x + 105}$$

as a sum of simpler fractions. ◇

§1.3 Graphing

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

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

produces a *plot window* 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 we 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 );
```

As always: *Think about your results. Are they reasonable?*

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. This graph can be used to solve the equation $\sin x = \sin 3x$. Click with the mouse on a point where these graphs cross. Maple gives the coordinates of the intersection point and the x-value is a solution to the equation.

Parametric Equations

A curve in the plane can be described as the graph of a function, as in the graph of $y = \frac{1}{2}\sqrt{4 - x^2}$ for $x \in [-1, 1]$, or it can be given parametrically as in

$$(x(t), y(t)) = (2\cos t, \sin t) \qquad \text{for } t \in [0, \pi].$$

Often we interpret such a curve as the path traced by a moving particle in the plane, with $(x(t), y(t))$ denoting the position of the 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

$$r^2 = \frac{4}{4 - 3\cos^2\theta}$$

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 );
```

Try the examples below. However, before asking Maple to do the plot, try to predict what the result will be.

```
> 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 our 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 );
> plot( data, style = point, symbol = diamond );
> 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. See `?readdata` or `?stats` to find out how to read a file of data points into a Maple session. (You could discover these commands on your own by typing `?data` or `?reading data`.)

Exercise 1. Graph $y = \dfrac{1}{x^3 - 6x^2 + 11x - 6}$ on the interval $-1 \le x \le 4$. ◇

Exercise 2. Graph $y = \dfrac{1}{\sqrt[3]{x}}$ on the interval $-27 \le x \le 27$. Do your results make sense? ◇

Exercise 3. Graph $y = \tan x$ and $y = x$ together on $-5 \le x \le 5$. Use the mouse and your graph to find solutions to the equation $\tan x = x$ in this interval. (You may need to replot on smaller intervals to locate solutions.) ◇

Exercise 4. Graph

$$y = \cos x - \frac{\cos 3x}{3} \text{ for } -\pi \le x \le \pi.$$

Then graph

$$y = \cos x - \frac{\cos 3x}{3} + \frac{\cos 5x}{5} \text{ for } -\pi \le x \le \pi.$$

What happens if we continue this pattern — that is, what does the graph of

$$y = \cos x - \frac{\cos 3x}{3} + \frac{\cos 5x}{5} + \cdots + (-1)^k \frac{\cos((2k+1)x)}{2k+1}$$

look like on $[-\pi, \pi]$ as k tends to infinity? ◇

Exercise 5. Graph the curve, defined by the polar equation $r = 2(1 + \cos\theta)$. (This curve is called a cardioid.) ◇

Exercise 6. The path of a point P on a circle of radius 2 that rolls along the outside of a larger circle of radius 5 is given by

$$x(t) = 8\cos t - 2\cos\left(\frac{7t}{2}\right) \quad \text{and} \quad y(t) = 8\sin t - 2\sin\left(\frac{7t}{2}\right).$$

Draw the path of this particle for t in $[0, 5]$. Repeat for t in $[0, 20]$. (This curve is called an *epicycloid*.) ◇

Exercise 7. Some interesting curves, called *Bowditch* or *Lissajous* curves, are given by coordinate functions of the form $x(t) = \cos at$ and $y(t) = \sin bt$, for constants a and b. Draw examples of these curves, experimenting with different choices of a and b to get a feeling for what they look like in general. Start with $a = 3$ and $b = 5$. ◇

Exercise 8. Compare the graphs obtained from the two commands:

```
> plot( [2 + 3*sin(t), t, t = 0..2*Pi] );
> plot( [2 + 3*sin(t), t, t = 0..2*Pi], coords = polar );
```

◇

§1.4 Solving Equations

Maple can also solve equations. Consider some examples:

```
> solve( x^2 + 3*x = 2.1 );
> solve( x^3 + x = 27 );
> 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 = 27, 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 some 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 = 0..2 );
```

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 );
```

Note 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
```

How did the sum and product relate to the coefficients of the equation?

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.
```

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 these variables to their unassigned state when you finish your problem. Recall how to do this:

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

Recall also that `restart` 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 );
> solve( a*x^3 + b*x^2 + c*x + d, x );
> solve( { a*x + b*y = h, c*x + d*y = k }, { x, y } );
```

The factor appearing in the denominator of the last computation is the determinant of the system of equations.

Exercise 1. Find as many roots of $x^5 + x = 32$ as you can. How many do you expect on theoretical grounds? You may want to see how `fsolve` works on this one. Here are some things to try:

```
> solve( x^5 + x = 32 );
> solve( x^5 + x = 32.0 );
> fsolve( x^5 + x = 32, x );
> fsolve( x^5 + x = 32, x, complex );
```

◇

Exercise 2**.** Find all real solutions to $x^5 + x^4 - x^3 - x^2 + 1 = 0$. How many are there? ◇

Exercise 3**.** Using

```
> e1 := 7*x + 3*y + 8*z = 1;  e2 := ... ;  e3 := ... ;
  solve({e1, e2, e3});
```

as a guide, solve the following systems:

$$(a) \quad \begin{array}{l} 7x + 3y + 8z = 1 \\ 2x - 4y + 5z = 2 \\ 3x - 7y + 9z = 1 \end{array} \qquad (b) \quad \begin{array}{l} 7x + 3y + 8z = 1 \\ 2x - 4y + 5z = 2 \end{array}$$

◇

Exercise 4**.** Use `solve` to find the intersection points of the two ellipses defined by

$$4x^2 + 9y^2 = 1 \quad \text{and} \quad 25x^2 + 4y^2 = 1.$$

Try using `fsolve` instead of `solve`. Finally, change the constant term in the first equation to 1.0 and try `solve` again. Do the results agree? Draw a picture to clarify the situation. You can do this by hand, or in Maple. The first ellipse can be described parametrically by the expression $(\frac{1}{2}\cos t, \frac{1}{3}\sin t)$ where t runs over the interval $[0, 2\pi]$. The second ellipse has a similar parametric description. By filling in the missing information below, you can plot both curves together:

```
> ellipse1 := [ cos(t)/2, sin(t)/3, t = 0..2*Pi ];
  ellipse2 := [ ??? ];
  plot({ ellipse1, ellipse2 }, scaling = constrained );
```

◇

Exercise 5**.** Find all solutions to $\sin x + x = \sec x$ for x in $[-3, 3]$. Use `plot` to get information about how many solutions exist and their approximate location. Then use `fsolve` to locate the solutions to 9 decimal places accuracy. ◇

§1.5 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 `->` as an arrow: it tells what to do with the input `x`, namely, produce the output `18*x^4 + 69*x^3 - 40*x^2 - 124*x - 48`. Once the function `p` is defined, we can do the usual computations with it, e.g.,

```
> p( -2 );
> p( 1/2 );
> p( 0.4 );
> p( a + b );
> simplify( " );
```

Warning: 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

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

Which of these worked? Does the `factor` command work on functions or expressions? What about `plot`?

Functions of several variables

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 := 1.3; g(1,2);
> w := 3.5; g(1,2);
> alpha := 'alpha'; g(1,2);
```

Proc and unapply

There are two other ways to define functions. The first is with `proc`. For example, to define $f(x, y) = e^x \sin y$, we say

```
> f := proc( x, y )
          exp(-x)*sin(y);
       end;
```

Note that there is no semicolon after the `proc(x,y)`. This style is used most often for functions that cannot be defined by a simple one-line expression, a topic we will take up again in Section 8.

It is possible to define functions of zero arguments. For example,

```
> trial := proc()
       rand(0..6)() + rand(0..6)();
  end;
```

defines a function which simulates the operation "throw two dice, then add up the number of spots on the top faces." We test this:

```
> trial(); trial(); trial();
```

The second way to define a function is to use `unapply`. The code

```
> f := unapply( exp(-x)*sin(y), x, y );
```

converts the expression `exp(-x)*sin(y)` into a function of `x` and `y`. To verify this, *apply* `f` to some some arguments:

```
> f(0,0); f(1.2, -0.4); f( u+v, arcsin(w) );
```

This last sentence reveals the secret of `unapply`: it is the reverse of the operation of applying a function to arguments to obtain an expression. *Riddle*: what kind of object is `unapply( f(x), x )`?

A common use of `unapply` is to define a function within a function.

Exercise 1. The function

```
> rad := x -> evalf( x * Pi /180 );
```

converts degrees to radians. Thus `rad(1)` is the number of radians in one degree. Modify this formula to define a function `deg` that converts radians to degrees. Thus `deg(1)` is the number of degrees in one radian. Use these functions together with Maple's standard trig functions (which work in radians) to (a) find how many degrees are in 1 radian, (b) to compute the sine of $1°$, (c) to find an angle in degrees whose cosine is 0.7, (d) to compute the height of a flagpole whose shadow is 14 feet long when the sun's rays make an angle of $11°$ with the vertical. ◇

Exercise 2. Define a function `dist` of (u, v, x, y) that gives, in decimal form, the distance from the point (u, v) to the point (x, y). Use your function to find how far $(11.34, 24.17)$ is from $(-9.61, 12.49)$. ◇

Exercise 3. Define a function `logb( b, x )` that computes the logarithm to the base `b` of the number `x`, expressed in floating-point form. (Note that `log(x)` in Maple means $\ln x$.) ◇

Exercise 4. Define $f(x) = x + 1$ and $g(x) = 2x$. Compute the composite functions $f \circ g(x) = f(g(x))$ and $g \circ f(x) = g(f(x))$. Are they the same? Repeat the computation in Maple. First define `f` and `g` and then

```
> (f@g)(x);
> (g@f)(x);
```

give the composite functions. ◇

Exercise 5. Define $f(x) = \sin(\sqrt{x})$ using each of the three styles of function definition. Use each definition to compute $f(2)$ in decimal form. ◇

§1.6 Calculus

Let us now explore Maple's tools for working with the notions of 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 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 wrt 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
```

Question: what is `D[1,2](k) - D[2,1](k)`?

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 contained an 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

$$\int_0^1 \sin(\sqrt{1-x^3})\,dx$$

which Maple could not evaluate. We can still find an approximate numerical answer using the definition of the integral: approximate the figure under the graph by little rectangles or trapezoids and add up their areas. You could write a loop to do this (see Section 8) or try:

```
> 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 package
> j := sin( sqrt(1 - x^3) );  # define integrand
> trapezoid( j, x = 0..1 );    # apply trapezoidal 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:

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

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`:

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

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 $0 \le x \le 1$ and $0 \le y \le 1$. To compute the double integral

$$\int_R (x^2 + y^2)\, dxdy$$

in Maple, we compute the repeated integral

$$\int_0^1 \int_0^1 (x^2 + y^2)\, dx\, dy.$$

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.

Exercise 1. Compute

$$\int_{-1}^1 \int_{-1}^1 (x^2 + y^2)\, dy\, dx$$

The integrand is the same, but the order of integration is different. How do the intermediate and final results compare with the previous computation? ◇

Maple can compute double integrals over more complicated regions, such as the triangle T defined by $0 \le x \le 1$ and $x \le y \le 1$. We have

$$\int_T (x^2 + y^2)\, dxdy = \int_0^1 \left(\int_x^1 (x^2 + y^2)\, dy \right) dx$$

In Maple this translates as

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

For more complicated integrands we can still use numerical methods, as illustrated in the problems below.

Exercise 2. Compute

$$\int_{-1}^{1}\int_{-1}^{1}\sqrt{|x^3-y^5|}\,dy\,dx$$

using Simpson's rule:

```
> j := sqrt( abs( x^3 - y^5 ) );          # integrand
  jy := simpson( j, y = -1..1 , 8):       # y integral
  jyx := simpson( jy, x = -1..1, 8 ):     # x integral
  evalf( jyx );                           # numerical result
```

Note the use of colons to suppress output. Repeat the computation with 16 subintervals. How accurate do you think the result is? Now reverse the order of integration and compare answers. ◇

Exercise 3. (Continuation) Compute the double integral

$$\int_T \sqrt{|x^3-y^5|}\,dxdy$$

on the triangular region T defined by $0 \le x \le 1$, $x \le y \le 1$ by computing the repeated integral

$$\int_0^1\int_x^1 \sqrt{|x^3-y^5|}\,dy\,dx.$$

Use Simpson's rule:

```
> jy := simpson( j, y = x..1, 8):       # y integral
  jyx := simpson( jy, x = 0..1, 8 ):    # x integral
  evalf( jyx );                         # numerical result
```

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 );
> sum( i^2, i = 1..100 );     # Be sure i is clear
> sum( x^n, n = 5..10 );      # needs n to be clear
> sum( 1/j^5, j= 1..infinity ); evalf( " );
```

Differential equations:

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

We can specify initial conditions and experiment with parameters.

```
> k := 'k';
> de := diff( y(x), x ) = k*y(x)*(1 - y(x));
> sol := dsolve( {de, y(0) = 0.1}, y(x) );
> k := 1; 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, title = `k = 1` );
> k := 2; plot( rhs(sol), x = 0..10, title = `k = 2` );
```

Exercise 4. Find the derivative of $x^5 + 7x^2 + x + 1$. Find the critical points of $f(x) = x^5 + 7x^2 + x + 1$, and determine if they give maxima, minima, or inflection points. ◇

Exercise 5. Find an antiderivative for $\sqrt{1+7x^2}$. Compute $\int_0^1 \sqrt{1+7x^2}$ in both exact and floating point form. ◇

Exercise 6. Find the Taylor series expansion of $\sqrt{\cos x}$ centered at $x = 0$. Compare the graph of the 4th order Taylor polynomial with the graph of $\sqrt{\cos x}$ itself on the interval $[-\pi/2, \pi/2]$. ◇

Exercise 7. (a) Use the command `sum` to find a value for

$$1 + \frac{1}{2^3} + \ldots + \frac{1}{n^3} \cdots$$

that you believe accurate to three decimal places. How many terms do you need to compute to get this accuracy?
(b) Repeat for

$$1 - \frac{1}{2} + \ldots + (-1)^{n+1}\frac{1}{n} \cdots.$$

◇

§1.7 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] );
> add( v, w );
> dotprod( v, w );
```

What does this tell you about the geometry of `v` and `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

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

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

```
> add(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));
```

Check to see if $AB = BA$:

```
> multiply(B, A);
```

Finally, we can change 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);
> inverse( transpose(B) );
> transpose( inverse(B) );
```

It can also compute determinants:

```
> det(A);
> det( A + B ); det(A) + det(B);
> det( A &* B );  det( B &* A ); 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.

Exercise 1. Let $\mathbf{v} = (1, 2, 3)$ and $\mathbf{w} = (-1, 4, 1)$. Find the sum and dot product of $\mathbf{v}$ and $\mathbf{w}$. Compute the length of $\mathbf{v}$ using the formula $Length(\mathbf{v}) = \sqrt{\mathbf{v} \cdot \mathbf{v}}$. ◇

Exercise 2. The angle between two vectors $\mathbf{v}$ and $\mathbf{w}$ is given by the formula

$$\theta = \arccos\left(\frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|}\right)$$

where $|\mathbf{v}|$ stands for the length of $\mathbf{v}$. Define a function `Length` that computes the length of a vector $\mathbf{v}$ and a function `Angle` that computes the angle between vectors $\mathbf{v}$ and $\mathbf{w}$. Use it to find the angle between $\mathbf{v} = (1, 2, 3)$ and $\mathbf{w} = (-1, 4, 1)$. You may want to use `evalf` to get results in decimal form. Is your result in radians or degrees? How would you modify your function definitions to get the opposite? ◇

Exercise 3. Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \text{ and } B = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}.$$

Compute AB and BA. Are they the same? Compute $A + B$ and $B + A$. Are they the same? Compute $7A$. Define C to be the sum of A and B. Then compute C^2 and compare to $A^2 + 2AB + B^2$. Compute the determinants of A and of B. Where possible, compute the inverses of A and B. Compute $A\mathbf{v}$ where $\mathbf{v} = (1, 2, 3)$ as above. Is it possible to compute $\mathbf{v}A$? Let $\mathbf{x}$ be an unknown vector and solve $B\mathbf{x} = \mathbf{v}$. ◇

Exercise 4. Let A be the same matrix as in the previous exercise. Use `augment` and `gaussjord` to solve $A\mathbf{x} = \mathbf{v}$, where $\mathbf{v} = (1, 2, 3)$. Repeat for $A\mathbf{x} = \mathbf{w}$, where $\mathbf{w} = (-1, 4, 1)$. ◇

§1.8 Programming in Maple

In this penultimate section we will take an extended look at the problem of summing a series. It will serve as a vehicle for learning about the elements of programming in Maple. We will study loops, conditionals, procedures, and lists. Loops are for automating repetitive work. Conditionals are for choosing between alternatives, i.e., for making decisions. Procedures are really the same as functions, but the `proc`-style definition facilitates more complicated definitions. Lists hold pieces of data and look like this: `L := [ 2, 3, 5, 7 ]`. The third element of this list is 5, and we can form expressions like `L[1] + L[2]` to get the sum of the first and second elements of the list.

Loops

Sometimes the quantity which we wish to compute cannot be gotten by evaluating a simple formula. Consider, for example, the problem of computing

$$s(n) = 1 + \frac{1}{4} + \frac{1}{9} + \ldots + \frac{1}{n^2}. \tag{8.1}$$

These are the partial sums of the infinite series

$$1 + \frac{1}{4} + \frac{1}{9} + \ldots + \frac{1}{n^2} + \cdots$$

One way to compute them is to write out the defining expression for $s(n)$ whenever we need to compute it:

```
> 1;
> 1 + 1/4;
> 1 + 1/4 + 1/9;
> 1 + 1/4 + 1/9 + 1/16;
> etc.
```

This works, but is not a good solution when n is large. We can cut down somewhat on the amount of typing needed by judicious use of the ditto symbol, which remembers the result of the last computation:

```
> 1;
> " + 1/4;
> " + 1/9;
> " + 1/16;
```

The results will be more comprehensible if we work with decimal expansions instead of fractions. We could accomplish this by writing

```
> 1.0;
> " + 1/4;
> etc.
```

The above method for computing $s(n)$ is tedious and repetitive. We can automate it with a *loop*. For example, to compute the first four terms of the series (8.1) we say this:

```
> n := 4;
  total := 0;
  for i from 1 to n do
      total := total + 1.0/i^2;
  od;
```

First, we define `n`, the number of terms to add up. Next, we create a variable `total` to hold the running total. Naturally, we set it to zero before we start. Finally, we run the loop. It consists of two parts. The first, `for i from 1 to n`, determines the number of times we "do" the loop. The variable `i` controls its progress. The second part is the loop body, the phrase `do total := total + 1/i^2; od` . It consists of the key words `do` and `od` (that is, `do` spelled backwards) enclosing a list of instructions. In our case we have only one instruction — add the quantity `1/i^2` to the current value of `total` and then store the result in `total`.

Running our loop is the same as executing the sequence of commands below.

```
> n := 4; total := 0;
  i := 1; total := total + 1.0/i^2;
  i := 2; total := total + 1.0/i^2;
  i := 3; total := total + 1.0/i^2;
  i := 4; total := total + 1.0/i^2;
  i := 5;
```

Consequently, the counter `i` has the value 5 when the loop is complete. *This may cause trouble later, so you might want to clear the variable* `i` *with* `i := 'i'` *when you have finished the problem.*

Our loop as written prints out all the intermediate sums. For large n this is not a good idea, so we rewrite our computation with colons to suppress display of all but the final results:

```
> n := 10;
  total := 0.0:
  for i from 1 to n do
      total := total + 1.0/i^2:
  od:
  total;
```

Warning: Only one Maple prompt (`>`) appears in the loop. This is the safest way to type in a loop because it ensures that every time you start a loop, *you also end it*. A command like

```
> for i from 1 to 5 do
>    i^2;
> od;
```

works, but can easily lead to disaster. For example, you might start typing a loop and realize that you forgot to do something else first, say define a function that you need inside the loop. Then you move back in your Maple file to take care of this oversight. You have abandoned the loop, but Maple has not. Everything you enter after a line like `for ... do` gets incorporated into the body of your loop until you enter a line containing `od` to end the loop — or until Maple crashes and you lose all your unsaved work because Maple is irredeemably confused by the nonsensical instructions it has been given. To avoid this problem type your loops and procedures on a single Maple command line. To continue a command on a new line without getting a new Maple prompt, you use *Enter* on a Unix machine and *Return* on a Macintosh.

Conditionals

Next, consider the problem of finding partial sums of the series

$$1 - \frac{1}{4} + \frac{1}{9} - \frac{1}{25} + \ldots \pm \frac{1}{n^2} + \cdots \tag{8.2}$$

Depending on whether n is odd or even, we add or subtract the term $1/n^2$. This can be accomplished using an `if-then-else` statement:

```
> n := 10;
  total := 0.0:
  for i from 1 to n do
    if type( i, odd ) then
      total := total + 1.0/i^2;
    else
      total := total - 1.0/i^2;
    fi;
  od:
  total;
```

Note that the loop body now consists of more than one statement.

The general form of a conditional statement is

```
  if <test> then
     <something>
  else
     <something else>
  fi;
```

The short form

```
  if <test> then
     <something>
  fi;
```

is also legitimate.

Procedures

Suppose now that we need to compute partial sums many times in different contexts. We would like to be able to say `S(2)`, `S(20)`, etc. instead of tinkering with the loop above. To do so we define `S` as a function using the `proc` style mentioned earlier.

```
> S := proc( n )
          local i, total;
          total := 0.0;
          for i from 1 to n do
            total := total + 1.0/i^2;
          od;
          total:
       end;
```

Notice that the body of our definition — the part between `proc( n )` and `end` — is almost a word-for-word copy of what we wrote above to compute partial sums. One important difference is that we declared the variables `i` and `total` to be local. This means that they cannot be seen outside the function `S`, and, moreover, that nothing that happens outside `S` can affect them. This kind of protection is useful, since we can't keep in mind the names of all the internal variables used by functions. The value of `S(n)` is the value of the last expression in the definition, namely, `total`. One can also use the `RETURN` statement to define the function value:

```
> S := proc( n )
         local i, total;
         total := 0.0;
         for i from 1 to n do
           total := total + 1.0/i^2;
         od;
         RETURN ( total );
      end;
```

Warning: As with loops, it is much safer to type the entire definition with a single prompt (`>`).

With a small bit of extra work, we can devise a procedure which computes the nth partial sum

$$S_n = f(1) + f(2) + \ldots + f(n)$$

where f is an *arbitrary* function. Here is the definition:

```
> S := proc( f, n )
        local i, total;
        total := 0.0;
        for i from 1 to n do
           total := total + evalf( f(i) );
        od;
        [ n, total ];
      end;
```

The arguments of `S` are a function and a number. The function value `S(f,n)` is a list consisting of two items: the number of terms in the sum, and the value of the partial sum.

Warning: There is no punctuation after the phrase `proc( f, n )`. Also, with procedures as with loops, it is much safer to type the entire definition on a single Maple command line.

Once we have typed in the definition of S, we can use it to recompute our previous example $1 + 1/4 + 1/9 + 1/16$. Define the function f

```
> f := x -> 1.0/x^2;
```

Then `S( f, 4 )` gives the sum. Alternatively, one could say

```
> S( x -> 1.0/x^2, 4 );
```

To sum up a different series, we just change the arguments to S. For example,

```
> S( i -> 1/i, 10 );
```

sums the first ten terms of the harmonic series:

$$1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} + \ldots$$

Lists

Another way to think about the behavior of an infinite sum is to plot the partial sums S_n versus n. This gives a picture of how the sum grows as n increases. We can modify our procedure S to produce a list of points (n, S_n) and then use `plot` to graph them.

To define our list we use Maple's sequence-building command. For instance, to generate a list of squares we can type

```
> i := 'i';    # Make sure i is unassigned
> [ i^2 $ i = 1..10 ];
```

Whenever you use $ be sure that the indexing variable is unassigned or you will get an error message. Alternatively, type

```
> [ 'i'^2 $ 'i' = 1..10 ];
```

to tell Maple to ignore any values previously assigned to the variable i when it makes this list of squares. The quotation marks here are single quotes.

Let us call our modified procedure `Spts`. We define it like this:

```
> Spts := proc( f, n )
      local i, j, total, pts;
      total := 0.0;
      for i from 1 to n do
          total := total + evalf( f(i) );
          pts[i] := [i, total]:
      od;
      [ pts[j] $ j = 1..n ];
    end;
```

It produces a list of points of the form $[k, S_k]$ as we wanted. Now try

```
> Spts( f, 10 );
> Spts( f, 10 )[10];
> plot( Spts(f, 10), style = line, labels = [n, Sn] );
```

Since we assigned `n` the value `10` earlier, this last command did not work. Try again:

```
> n := 'n';
  plot( Spts(f, 10), style = line, labels = [n, Sn] );
```

Now we have the desired plot. Replot using the first 100 terms of the series. Does it look like the sum approaches a well-defined number as we take more and more terms?

Exercise 1. Define a function $f(x) = \sin x/x$. Investigate the limit of $f(x)$ as $x \to 0$ by computing the sequence $f(1), f(1/2), \ldots, f(1/10)$ with a loop. $\diamond$

Exercise 2. Define a function to compute the alternating series (8.2). $\diamond$

Exercise 3. (a) Use the procedures `S` and `Spts` to investigate the limit of the partial sums

$$S_n = 1 + \frac{1}{4} + \ldots + \frac{1}{n^2}$$

as $n \to \infty$. Does the limit exist? That is, do the partial sums approach some well-defined number or do they grow without bound?
(b) Repeat for the harmonic series

$$1 + \frac{1}{2} + \ldots + \frac{1}{n} + \ldots$$

(c) Plot the partial sums for both series together:

```
> plot( {Spts(x ->1/x, 100), Spts(x->1/x^2, 100)});
```

$\diamond$

Exercise 4. Study the alternating harmonic series

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots \pm \frac{1}{n} + \ldots$$

Does it converge? If so, does it converge quickly or slowly? Compare with the series

$$1 - \frac{1}{4} + \frac{1}{9} - \frac{1}{16} + \ldots \pm \frac{1}{n^2} + \ldots$$

$\diamond$

Exercise 5. Write a procedure S3 that sums the first n cubes. Notice that the first five values, obtained by

```
> i := 'i';
> [S3(i) $ i = 1..5];
```

are perfect squares. Does this pattern continue? Modify your procedure to sum the first n cubes, then take the square root of this sum, and give

$$\left[\text{n}\ ,\ \text{sum of first n cubes}\ ,\sqrt{\text{sum of first n cubes}}\right]$$

as output. Compute the sequence `[S3(i) $ i = 1..15]`. What pattern do you see? Does this pattern continue forever? ◇

Exercise 6. Consider the sequence of odd counting numbers, grouped as shown below:

$$1, \quad 3, 5, \quad 7, 9, 11, \quad 13, 15, 17, 19, \quad 21, 23, 25, 27, 29, \quad \ldots$$

Notice that the first term is 1^3, the sum of the next two terms is 8 or 2^3, and the sum of the next three terms is 27 or 3^3. The sum of the next four terms is 64 or 4^3. Does this pattern continue? Use Maple to check that this pattern continues up to at least 15^3. How could one be sure that it continues forever? ◇

The next exercise contains a new kind of loop, the `while-do` loop. Its general form is

```
while ( <test> )
  do
    <something>
    <something else>
    <etc>
  od;
```

Exercise 7. Another variation on the procedures `S` and `Spts` is the following:

```
> G := proc( L )
     local s, n;
     s := 0.0;
     n := 0;
     while (s < L)
        do
           n := n+1;
           s := s + 1.0/n;
        od;
     [s, n];
    end;
```

Compute `G(1.0)`, `G(2.0)`, `G(3.0)`, `G(4.0)`, `G(5.0)`, `G(6.0)`, `G(7.0)`. Try to figure out what the procedure `G` is doing. What does it tell you about the harmonic series $1 + 1/2 + \cdots + 1/n + \cdots$? ◇

§1.9 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 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. 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, `ff := 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 `ff(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. In loops, `do` must be balanced by a subsequent `od`. In conditionals, `if` must be balanced by a subsequent `fi`.
9. A procedure definition that begins with `proc( ... )` must be be terminated by a subsequent `end`. There is no semicolon after `proc(...)`.
10. 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 )`.
11. Do not confuse `unassign` with `unapply`.
12. Remember to use `?` to inquire about the details of a Maple function, e.g., `?plot` for information about `plot` or just `?` for general information.