A Brief Guide to Maple

(c) James Carlson, Jennifer Johnson

Draft of October, 1995

Contents

Chapter 1. Introduction

These notes are intended as an introduction to Maple, a computer language which can be used as a tool in many different kinds of mathematical analysis. To name a few useful features, dealt with in detail in subsequent chapters,

We can use Maple

— like a calculator, to carry out numerical calculations with standard functions

— to define our own functions

— to make plots from functions of one or two variables or from data

— to solve equations in one or more unknowns

— to do calculus: compute derivatives, find definite and indefinite integrals, solve differential equations

— to do linear algebra: work with matrices and vectors.

In any of these situations we can get numerical results or we can ask Maple to work symbolically, carrying out algebraic computations and deriving formulas.

We can also do programming — we can "teach" Maple procedures for solving a problem in general. For example, consider the problem of deciding whether or not a given integer n is prime. We know in principle how to do this; we just have to check if any whole number smaller than n divides n evenly. The process is clear, but the tedious details are unappealing. However, these tedious details consist of many calculations that Maple can carry out. We just have to tell Maple a general plan and how to make decisions along the way. That is what we mean by a Maple "program" that takes an integer n and checks whether it is prime or not. (See the chapter on defining your own functions for more.)

The best way to learn Maple is by using it. For this reason the notes below are intended to be read while at the computer. You should try each example as you read it to see that it really does what it should. (To encourage you to work at the computer, we usually just give the Maple command, not its output). While working in this way Maple will sometimes appear to misbehave. Usually this is the result of a misunderstanding, often because of a typing error — perhaps a function name is misspelled, or a semicolon is missing, or you used "=" instead of ":=". Sometimes it is because you and Maple have slightly different ideas about the meaning of what you typed in. These errors are more difficult to deal with. However, as you develop an understanding of what you are doing, they will be less common and more easily solved.

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

```
> ?int
```

gives information on the command for integrating functions, and `?diff` tells about differentiation. You may find it more useful to look first at the examples at the end of each help section rather than the technical section at the beginning. Here are some other things to try:

```
> ?
> ?intro
> ?library
> ?solve
> ?fsolve
> ?packages
```

Another feature is Maple's tutorial. To use it, type

```
> tutorial();
```

and follow directions. Finally, don't neglect to ask your classmates, the person working at the next workstation, or the lab assistants for help. A few words of explanation from someone who already knows the ropes can often save you hours of frustration.

We have included exercises at the end of each section. You are encouraged to try some of these, as well as to think up your own. Most important, you should begin, starting now, to use Maple in your own work.

Chapter 2. Basics

Technical details on how to open the Maple program, and how to save and print a Maple file are highly dependent on your local system. The notes below are written to be as general as possible, so no information is given here on these important aspects of getting started. Consult your local system support staff if you need help with this.

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

```
> 2 + 2;
                                4
> quit
```

Note Maple's command prompt ">". This is where you type what you want Maple to compute. Once you have typed your "command", press the Return key (on Sun workstations — on Macintosh versions of Maple you use the Enter key instead) to tell Maple to execute it. Commands must be properly punctuated: they end with either a colon or a semicolon. For now we will use semicolons. If you forget to end a command with a semicolon you can just type it in on the next line. *The semicolon tells Maple that you have completed a command; the Return key says go ahead and carry it out now.* (For example

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

has precisely the same effect as `> 2 + 2`.)

Try starting up Maple now and compute $2 + 2$ as in the example (this is your first exercise). Then try something more difficult, maybe an expression with parentheses.

Note: Sometimes Maple will take too long to do a computation, in which case you need to interrupt it. There are different ways to do this on different systems. For plain `maple` on a Unix system press ctrl-C. This is a universal way to abort a running program. For `xmaple` click on the *Interrupt* button in the Maple window.

Addition, subtraction, etc., are standard, and parentheses are used in the usual way. Use a carat, as in `2^3 = 8`, to take powers. An asterix indicates multiplication: `2*3 = 6`. Try:

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

For the most part spacing is unimportant in a Maple command; in this command all the spaces could be omitted without causing any problems. However, generous use of the space bar often gives you a file that is much easier to read, particularly useful when you are trying to locate typing errors.

Note that whenever possible Maple tries to compute exact quantities. In the example above we see that Maple carries out exact rational arithmetic; it gives an answer in fraction form, rather than decimal form as we might expect from our experience with calculators. Thus, if you compute `31416/10000`, which is a way of representing an approximate value of π, Maple gives a result of 3927/1250. To get a decimal result, you then have to say `evalf(")`. This function turns whatever you give it into a floating point or decimal number. Thus `evalf(1/2)` produces `0.5`. The quote symbol, read "ditto", stands for the result of the immediately preceding computation. Here are some further examples (remember — do them now):

```
> 123/111;
> evalf(");
> 129/127;
> evalf(", 20);
> Pi;
> evalf(Pi);
> evalf(Pi,100);
```

The function `evalf` can take a second, optional input which determines the precision of the output. (Inputs, which in mathematics are called independent variables, are known as *arguments* in computer jargon). Another way to control the precision used by `evalf` is to set the `Digits` variable, e.g., `Digits := 2` for a low precision. Here is an example:

```
> Digits;
> evalf(Pi);
> Digits := 100;
> evalf(Pi);
> Digits := 10;
```

Note that we have restored the original value of `Digits` at the end of the computation.

Warning: Some commands, like `fsolve`, which is used to solve equations, will give strange answers if `Digits` is set too low.

We can use all the standard mathematical functions provided that we know (or can guess) their names. Thus, to compute $2 + 14 \cdot 8 + |-14| - \sqrt{\sin(1)}$, say

```
> 2 + 14*8 + abs(-14) - sqrt(sin(1));
> evalf(");
```

Or as another example, try

```
> exp(sin(1.6*Pi)) + sqrt(2) + arctan(3); evalf(");
```

Again notice that Maple prefers to give exact results — we have to force it to compute decimal approximations. Thus, `sqrt(1 + 1)` produces $\sqrt{2}$ as output, while `evalf(sqrt(1 + 1))` produces 1.414213562.

To see whether trig functions are computed in radians or degrees, we do an experiment:

```
> evalf(sin(1));
```

What do you conclude? Here is another thing to try:

```
> tan(arctan(x))
```

The result is x, as expected, since `tan` undoes what `arctan` does.

Remark: It is easy to define new functions if you need them. For example, to define a function `rad` which converts degrees to radians, we say

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

Thus, to compute the number of radians in one degree, we say `rad(1)` and get `.01745329252`. For the reverse conversion we make the definition

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

Then `deg(1)` computes the number of degrees in one radian, giving `57.29577950`. Once we have `deg` and `rad` in hand, we can do conventional trig problems. For instance, to compute $\sin(1°)$, we say `sin(rad(1))`. To compute $\arcsin(0.7)$ in degrees we say `deg(arcsin(0.7))`.

What we just did illustrates an important point: you can often guess the correct form of a Maple expression even if you haven't looked it up in the manual. Thus, `tan(1)` does indeed compute the tangent, `arctan(1)` computes the arctangent, and

```
> 20!;
```

computes a factorial. If your first guess doesn't work use `?` together with your guess to get more information. For example, if you guessed that the inverse tangent function is `arctangent` you would be wrong, but the help query

```
> ?arctangent
```

would get you the right command.

Maple can also do algebra, i.e. work with symbols: Let's illustrate this by manipulating the square of a binomial. Try

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

By factoring it we reverse the process of expansion:

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

Values can be assigned to variables, which makes long computations easier. Thus, the sequence of commands

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

first computes $(a+b)^2$, then stores it in a variable p, sets b to 1, and finally displays the result, in this case $(a+1)^2$. Note the special symbol used to assign a value to a variable. You can read $b := 1$ as "b gets 1". Be aware that the "gets" symbol " $:=$" used for assigning values, is different from the equal sign "=", used to define equations. Note that sometimes we use colons instead of semicolons; this supresses display of the result.

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 $(a+b)^2$ stored in the variable p. Note the kind of quotes used in the command above: the possibilities are the single quote `'`, the backquote `` ` ``, and the double quote `"`, all of which are different.

Let's now look at another example where we assign integer values to variables a and b and then do some computations:

```
> a:=25;
> b:=15;
> (a/b)^17;
> evalf(");
```

Again we use `evalf` to get a decimal approximation for a result just computed. Another way of getting decimal results is to assign decimal values to at least one of the variables a and b. This way Maple knows that all computations are to be done in decimal form:

```
> a:=25.0;  # or 25.00 or just 25. (etc.)
> b:=15;
> (a/b)^17;
```

The idea here is that if one of the numbers in the expression is in decimal format, Maple computes and presents the result in the same format. Note the comment in the first line of our example. Comments always begin with the character `#`, which just tells Maple to ignore the `#` and everything which comes after it, up to the end of the line.

Other frequently used Maple commands include `int`, `diff`, and variations on `solve`. Here are some examples to try:

```
> diff( x^n, x );               # differentiate x^n with respect to x
> int( x^n, x);                 # integrate x^n with respect to x
> int( x^3, x = 0..1)           # integrate x^3 from 0 to 1
> int( x^n, x = 0..1)           # compute a definite integral
```

The results are not always in the form we expect. For example, `diff( x^n, x )` yields

$$\frac{x^n\ n}{x}$$

If we subsequently say `simplify( " )`, Maple reports with something more reasonable:

$$x^{n-1}\ n$$

Remember that the quote sign (") stands for the result of the last computation done. We could achieve the same effect here by saying `simplify( diff( x^n, x ))`.

To find the roots of a cubic equation $x^3 + x = 27$ we have several options. The most obvious

```
> solve( x^3 + x = 27, x);
```

uses algebraic methods to solve, but we get results that are rather difficult to understand. In this exact but difficult-to-read answer, Maple uses the name `%1` to stand for the number $\frac{27}{2} + \frac{1}{18}\sqrt{19687}\sqrt{3}$ and then expresses the three roots in terms of this number. It may be more useful to use

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

to get solutions expressed in (approximate) decimal form. Sometimes algebraic methods fail altogether — for nonpolynomial equations or for polynomials of higher degree. In that case using a numerical method may be the only way to proceed.

```
> fsolve( x^3 - x = 27, x);             # solve it numerically
```

Sometimes, numerical methods require a good initial estimate of where the roots lie. In that case looking at the graph is helpful:

```
> plot( x^3 - x - 27, x = -5 .. 5);   # solve graphically
```

Exercises

1. Compute $(5/3)^{21}$, both as an exact expression, and as a decimal accurate to (a) standard precision, (b) twenty digits precision.

2. Maple does its trigonometry using radian measure. Thus `sin(Pi/2)` yields 1. Verify this now and then compute $\sin(\pi/4)$.
3. Compute $\sin(\pi/4)$ where the angle is in radians. Then convert the result to decimal form.
4. This exercise uses the `rad` and `deg` functions defined above. (a) Compute $\sin(13.4°)$. (b) Find the angle in degrees whose sine is 0.241.
5. Compute $2^{100}-1$, then find its prime factors using `ifactor`. Example: `ifactor(2^31 - 1)`.
6. Compute $(a + 2b + 3c)^4$. Use `expand` to get the result you expect. What is the coefficient of ab^2c?
7. Find the derivative of $\sin(x^2)$. Check Maple's computation by hand.
8. Find the indefinite integral of $1/(x^2 + 5)$.
9. Compute $\int_0^1 \frac{dx}{x^2+5}$. Use `evalf( " )` to convert the result to decimal form.
10. Solve the cubic equation $x^3 = 27$. Then solve $x^3 + 0.1x = 27$. Notice that the coefficients of the the second equation are close to the coefficients of the first one. So are the solutions.
11. Plot the function defined by $e^{-x} \cos 10x$ on the interval $[0, 2\pi]$. This is a "damped cosine wave." Use `exp` for the exponential function, i.e., use `exp(x)` for e^x.

Problems

1. A rational number — the quotient of two integers — has a decimal expansion which either terminates or eventually settles down to a repeating pattern. (Consider the expansion of $1/3$, for example). Use Maple to find the repeating part of $23/11$, $23/13$, and $713/211$. Try also
$$\frac{524767338531}{3030303030303}$$
and
$$\frac{192414574797}{1111111111111}$$
2. For the sake of argument, let $P = 4$ be the human population of the earth today, measured in billions. Also for the sake of argument, assume that the annual growth rate is $r = .04$ (four percent). Then the population n years later will be
$$Q = (1 + r)^n P.$$
Compute the population in billions for $n = 10, 25, 100, 1000$. What conclusions can you draw? Redo your calculations with different values of r. (Use variables in this problem).
3. Maple has a function called `ifactor` which factors integers. It takes one integer as input and produces the factorization as output. Try factoring some reasonably large numbers, like 123456789. Try also some really large numbers like $2^{89}-1$ and $2^{211}-1$. You might want to evaluate these before factoring them so that you can enjoy the decimal expansions. Comment on the time necessary to perform these calculations. How might it depend on the size of the integers one tries to factor?

There is a function for determining whether a number is prime without finding its factors, as in `isprime(2^(211)-1)`. Try this function on the numbers above. How does its speed compare with `ifactor`?

Warning: Before you try this problem you may want to review the procedure for interrupting a computation that takes too long. (see Note at the beginning of this section.)

4. Maple also has a function called `nextprime`. It takes an integer n as input and returns the smallest prime $p > n$ as output. Use it to find some very large prime numbers. You might also experiment with `ithprime`.

Chapter 3. Algebraic expressions

Let us now try working with algebraic expressions. We begin by defining a binomial:

```
> p := x + y;
```

Note that Maple does not expand automatically, as you will see when you try these:

```
> (x + 2*y)*p;
> p*p*p;
> p^15;
```

Note that Maple does not understand `(x + 2y)p`; the asterix is necessary to indicate multiplication. In any case, to get the expanded form, we do this:

```
> expand(p^15);
```

To continue working, we store the expanded form in a variable *pp*:

```
> pp:= ";
```

Note how much space the expanded form takes. If we don't want it to appear on the screen, remember we use a colon instead of a semicolon:

```
> pp:
```

To get one of the coefficients of *pp*, we do this:

```
> coeff(pp,x,12);
```

The reverse of expansion is factorization:

```
> factor(pp);
```

Notice what happens when we try to get the coefficient of something that is not expanded:

```
> coeff(",x,12);
```

In order to be sure that we get the coefficient as we want it, we can tell Maple to collect everything in powers of x:

```
> collect(pp,x): coeff(",x,12);
```

As we learned in the last section, variables can be assigned values. Thus, we can do the following:

```
> p := (x + y)^3;
> expand(");
> y := 1;
> p;
> expand(");
```

Our substitution works as expected. To get back our original polynomial we "unassign" y:

```
> y := 'y';
> p;
```

Be careful about the kind of quotes you use here — there are three different kinds on the keyboard. Note that assignments don't have to be numerical, and in fact can be anything you like. Experiment with $y := a + b$.

Often it is convenient to make a "temporary" substitution that does not change the original expression. Here is an example:

```
> a := 'a';     # clear the variable a
> F := m*a      # Newton's Law;
> subs(m = 2.1, F);
> subs(m = 2.1, a = 3.7, F);
> F;
```

The purpose of the expression `a := 'a'` is to clear the variable `a` — we had assigned a value to it earlier.

Now let's go through a sample algebraic calculation, one more complicated than the elementary examples above. In algebra we learn how to rewrite a rational expression like

$$\frac{x^2 - x + 1}{x^3 + 2x^2 - 5x - 6}$$

as a sum of simpler rational expressions — this process is called finding a partial fraction decomposition for the original expression.

```
> top := x^2 - x + 1;  bottom := x^3 + 2*x^2 - 5*x - 6;
> factor(bottom);
```

This factorization tells us that the desired partial fraction decomposition is of the form:

$$\frac{x^2 - x + 1}{x^3 + 2x^2 - 5x - 6} = \frac{A}{x+1} + \frac{B}{x-2} + \frac{C}{x+3},$$

where A, B, and C are to be determined from this equality. In Maple we define and name this condition:

```
> maineqn := top/bottom = A/(x+1) + B/(x-2) + C/(x+3);
```

Then

```
> maineqn := simplify( maineqn );
```

puts the condition in a more readable form. The denominators are the same in this expression so to find the partial fraction decomposition we want, we must choose A, B, and C so that the numerators match. We first give the numerator involving the parameters A, B, and C a name, using the functions `numer` and `rhs` to get our hands on the parts of the equation we want to work with.

```
> ABCtop := numer( rhs( maineqn ) );
```

Then we set up three equations, one for each power of x that appears in the numerators:

```
> eqn1 := coeff( ABCtop, x, 2 ) = coeff( top, x, 2 );
  eqn2 := coeff( ABCtop, x, 1 ) = coeff( top, x, 1 );
  eqn3 := coeff( ABCtop, x, 0 ) = coeff( top, x, 0 );
```

Finally, we use

```
> solve({eqn1, eqn2, eqn3},{A, B, C});
```

This just tells us how we must choose the parameters A, B, and C; the corresponding values will not be assigned unless we explicitly tell Maple to do so:

```
> assign( " );
```

To check redisplay the equation

```
> maineqn;
```

Remember that you can get more information using Maple's help system. Some things to try are

```
> ?factor
> ?collect
> ?expand
> ?coeff
> ?subs
> ?restart
```

Problems

1. Compute $(x+1)(x+2)(x+3)(x+4)$ in expanded form, then find the coefficient of x^3.
2. Factor (and so find the roots of) $p = x^4 + 8x^3 - 10x^2 - 104x + 105$.
3. Use the function `solve` to find the roots of the polynomial in the previous problem. If you have stored it in `p`, then `solve(p, x)` should work. For more information, try `?solve`.
4. Let us study the polynomial $q = x^4 + 8x^3 - 10x^2 - 104x + 104$, which we get by changing the constant term of p by one. First try to factor it. What do you find? Then try to find roots using `solve`. What do you find? Finally, try `fsolve` in place of `solve`. If you need more information on the latter, try `?fsolve`.
5. Find a polynomial with integer coefficients that has $r = -1 + \sqrt{7}$ as a root. Do the same for $r = 1 + \sqrt{11 + \sqrt{7}}$.
6. Find the coefficient of x^{10} in $p := (x + ay)^{14}$. For this problem you may need to use something like $collect(p, x)$.
7. Find the partial fraction decomposition for

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

Check your work using Maple's built-in function for finding partial fraction decompositions. You can find out about this function using

```
> ?convert[parfrac]
```

which I eventually found from the sequence of help queries

```
>  ?partialfraction
>  ?parfrac
```

Chapter 4. Graphing

Maple can make many kinds of graphs — ordinary graphs of $y = f(x)$, parametric plots, plots in polar coordinates, etc. It can also make three-dimensional plots, e.g., plots of surfaces defined by equations of the form $z = f(x, y)$.

§ 1. Ordinary graphs

For example to sketch $y = \sin(3x)$ on the interval $-\pi \leq x \leq \pi$, we say

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

The first argument is the expression to be graphed, the second argument gives the range over which x varies. Sometimes it is useful to restrict the range over which y varies. Consider, for example, the problem of graphing $\tan(x)$ on the interval $[-5, 5]$. We could try

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

You will find (try it!) that the graph is misleading: it does not accurately represent the vertical asymptotes at $x = 0$, $x = \pm\pi/2$, etc. Better results are obtained with

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

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

It is also possible to plot more than one curve at a time. For example, to plot $y = \sin(x)$ and $y = \sin(3x)/3$ we say this:

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

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

Plots can be used to locate solutions of equations graphically. For example, try

```
> plot( x^3 + x - 27, x = -5..5 );
```

The graph crosses the x axis near $x = 3$, so there is a solution to $x^3 + x - 27 = 0$ in that neighborhood. To visualize it better, we try

```
> plot( x^3 + x - 27, x = 2.5..3.5 );
```

For a slightly more difficult example, we look for solutions to the equation $\tan(x) = x$. To do this we plot the graphs of $y = \tan(x)$ and $y = x$ and note where they cross. Try this:

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

For more information about `plot`, use

```
> ?plot
```

§ 2. Parametric equations

For parametric plots, given by $x = f(t)$, $y = g(t)$, imitate the following, which is an attempt to plot an ellipse:

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

You will notice that it looks like a circle because Maple did not choose to use the same horizontal and vertical scales. To fix this, we do the following:

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

or

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

You might enjoy plotting *Bowditch curves*, named for Nathaniel Bowditch (1815), and also called *Lissajous curves*. These are defined parametrically by $x = a\sin(nt+c)$, $y = b\sin(t)$. Here is a nice one:

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

§ 3. Polar equations

Polar plots are a special kind of parametric plot. The polar coordinates (r, θ) of a point on the curve are given by $r = f(t)$, $\theta = g(t)$. If $g(t) = t$, we can think of the parameter t as being the angle. This is the most common form of a polar parametrization.

To make a polar plot in Maple, use

```
> plot( [ f(t), g(t), t = a..b ], coords = polar );
```

For an easy example, consider a circle of radius one centered at the origin. This circle consists of all points at a distance 1 from the origin. Thus we want all points in the plane whose first polar coordinate r is 1. As we let the second polar coordinate θ vary from 0 to 2π we get the desired circle. So the circle is described by the polar equations are $r(t) = 1$, $\theta(t) = t$, and we use

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

to draw it.

For slightly more complicated example, set $r(t) = t$, and keep $\theta(t) = t$. Thus radius is proportional to angle. Think for a moment what the shape is before plotting, then do

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

Use the mouse to click on the point that corresponds to $t = \pi/4$. The coordinates of this point should appear in your plot window. Locate the points that correspond to $t = 0, \pi/6, \pi/3, 5\pi/3, 11\pi/6$.

More examples:

The curve given by the polar equation $r = a(1 + \cos\theta)$ is called a *cardioid*. We parametrize it it the usual way: $r = a(1 + \cos(t))$, $\theta = t$, and use Maple like this, where $a = 2$:

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

After graphing this, you will see why the curve is called a cardioid, particularly if you look up its etymology in a good dictionary.

Here are some other things to try: The *logarithmic spiral*, given by $r = ke^{a\theta}$, and the spirals defined by $r^m = a^m\theta$. For example, let's graph the spiral with $m = -2$, $a = 1$, which has equation $\theta = 1/r^2$. One possible parametric representation is $r = t$, $\theta = 1/t^2$, which translates into Maple as

```
>  plot( [ t, 1/t^2, t = 0.2..4 ], coords = polar );
```

We have started the plot at `t = 0.2` to avoid the singularity at $t = 0$.

You can get some interesting curves by looking at curves of the form

$$r(t) = (a + b\cos(mt))(c + d\sin(nt)) \text{ and } \theta(t) = t.$$

For example, try $r = 2 + 3\sin(3t)$.

§ 4. Plotting data

Maple can also make various kinds of graphs from data. For example,

```
> plot ( [0, 0.23, 1, 1.14, 1, 0.89, 2, 1.84, 3, 4.1], style = POINT );
```

gives a graph with one dot for each of the five data points

```
(0, 0.23), (1, 1.14), (1, 0.89), (2, 1.84), (3, 4.1)
```

We can also construct broken line graphs from data:

```
> plot ( [0, 0.23, 1, 1.14, 2, 1.84, 3, 4.1], style = line );
```

Note that this works best if there is just one y-value for each x-value, so that the data defines a function.

Maple also has a command `readdata` for reading in raw data from an input file. Use `?readdata` for more info.

§ 5. More information

Many students find that the most useful aspect of Maple is its plotting features which allow them to visualize the objects they are working with much more clearly. We have touched on only a few of these features in the discussion above. For more information type

```
> ?plot
```

There you will find reference to 3-D plots, parametric plots, polar plots, etc. To find out about them, use queries like

```
> ?plot3d
> ?plot[parametric]
> ?plot[polar]
> ?plots
```

etc.

Problems

1. Make a graph of $\sin(x) + \sin(3x)/3$ on the interval $[-\pi, \pi]$.
2. Make a graph of $1/(x^3 - 6x^2 + 11x - 6)$ on the interval $-1 \leq x \leq 4$. Be careful about vertical asymptotes. Try limiting the range of y, and try plotting on small intervals around where you think the asymptotes are. Finally, factor $x^3 - 6x^2 + 11x - 6$ to see algebraically where the asymptotes are.

3. Plot the function $x^{-1/3}$ on the interval $-27 \leq x \leq 27$. Do your results make sense?
4. Make a graph of $\sin(x)$ and $x - x^3/6$ on the interval $[-\pi, \pi]$. Both curves should appear on the same plot. The polynomial $x - x^3/6$ is a Taylor approximation to $\sin(x)$. For what could it be used? What happens if we plot these functions on smaller and smaller intervals centered at the origin?
5. Another interesting example of a parametric curve is the *epicycloid* traced by a point P on a circle of radius b that rolls along the outside of a circle of radius $a > b$. This curve is given by the equations

$$x(t) = (a+b)\cos(t) - b\cos\left(\frac{(a+b)t}{b}\right)$$
$$y(t) = (a+b)\sin(t) - b\sin\left(\frac{(a+b)t}{b}\right)$$

   Plot the curve you get when a circle of radius 2 rolls along the outside of a circle of radius 5 using this parametrization. Experiment with different choices of a and b.
6. Plot the curve whose polar coordinates satisfy the equation $r = 1 + 2\cos(\theta)$, $0 \leq \theta \leq 2\pi$.

Chapter 5. Roots and equations

Maple can find roots of equations using functions like `factor`, `roots`, `solve`, and `fsolve`. Let's try some of these on the polynomial

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

First of all, we can factor it:

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

If you do this you will see 4, 3/2, and $-2/3$ are roots, where the last root has multiplicity two, i.e., " the corresponding linear factor $3x + 2$ appears twice when we factor p." We can also find roots like this:

```
> roots(p);
```

The result `[[3/2, 1], [-4, 1], [-2/3, 2]]` is a list of roots together with their multiplicities. Still another way to find roots is with the `solve` function:

```
> solve( p = 0 );  # solve( p ); also works
```

This form is convenient because we can then solve other equations:

```
> solve( p = 1 );
```

If you did this you might have been surprised: the answer is given in exact form and is rather hard to read. In this case an inexact answer in floating point form is probably more useful. We can get it like this:

```
> solve (p = 1.0);
```

Note that two of the roots are complex.

When searching for roots it is always a good idea to look at the graph:

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

Note: The `roots` function fails when applied to polynomials with floating point coefficients, e.g., $x^3 - 27.0$.

Another way to solve equations is with the function `fsolve`. It uses a numerical (as opposed to exact) method to get decimal approximations to the solutions of an equation.

However, unlike `solve`, it looks only for real solutions unless we explicitly request that it find complex solutions. A word of warning: `fsolve` sometimes misses roots. This is because it uses numerical techniques rather than symbolic ones. Because these usually depend on the way we phrase the problem, we must use `fsolve` with caution and a certain amount of skepticism. Let's look at some examples. We will try to solve $x^2 + x + 1 = 0$, which has complex roots:

```
> Digits := 2;
> q := x^2 + x + 1;
> fsolve( q );
```

The answer is nonsensical because `Digits`, which determines the precision with which `fsolve` looks for roots, is set too low. We try it with a better value:

```
> Digits := 8: fsolve( q );
```

That didn't work either, because it turns out we must instruct `fsolve` to look for complex roots:

```
> fsolve( q, x, complex);
```

In this last example, we have to specify that x is a variable because the syntax for `fsolve` is

$$fsolve(\ equation,\ variables,\ options),$$

We can omit the options, or we can omit both the variables and the options, but if so, we accept the consequences. Our previous examples worked because Maple could guess what the variables (the unknowns) were. For more about `fsolve`, including its options, do

```
> ?fsolve
```

Moral: Don't blindly trust what the computer gives you as an answer. It may be nonsense. You can usually tell by thinking about the problem. In this example, looking at the graph helps, or alternatively, we know that a quadratic polynomial has two roots, counting multiplicities. So if it gives you none, something is wrong.

It is important to be aware that Maple's solving functions frequently have trouble with square roots:

```
> solve( sqrt(x) = -1);
> solve( sqrt(2*x + 9) = x - 13, x);
> fsolve( sqrt(2*x + 9) = x - 13, x);
```

You should check the solutions returned by Maple for these examples! One reason for the trouble with equations like this is that Maple solves them by solving the squared equation.

Thus, in looking for solutions to $\sqrt{2x+9} = x - 13$ it inadvertently (and incorrectly) finds solutions to $\sqrt{2x+9} = -(x-13)$ as well. Thus we must check that the solutions returned really work.

The `fsolve` command is useful because it can find solve nonpolynomial equations like this one:

```
> fsolve( sin(x) + x = 0.2 );
```

Of course, the solutions it does find are approximate ones, where the precision is controlled by `Digits`. Moreover, this command produces one (approximate) value of x that works. There is no guarantee that this is the only solution.

An *excellent* way to understand the roots of equation is to look at graphs. In our example, the root is the place where the graph of $y = \sin x + x$ crosses the line $y = 0.2$. We can visualize this by making a little sketch or by typing something like

```
> plot( { sin(x) + x, 0.2 }, x = -4..4);
```

You can play with the range for x to home in on the place where the action is (near the root).

The `solve` function also works for some inequalities:

```
> solve(x^2 - 3*x > 0);
```

returns

$$\{x < 0\}, \quad \{3 < x\}$$

Thus $x^2 - 3x$ is positive on the interval $(\infty, 0)$ and also on the interval $(3, \infty)$. You can also try

```
> solve(x^2 - 3*x <= 0);
```

Of course, things don't always work as expected:

```
> solve( sqrt( x^2 - 3*x )-x >= 0 );
```

produces an error message

```
Error, (in solve/ineqs) unable to determine sign of expression
```

We should mention that you can use symbolic coefficients as well as numeric ones:

```
> a := 'a';  # clear a
> solve( x^2 + a*x + 1, x);
> solve( x^2 + a*x + 1, a);
```

Note that since two symbols appear, we have to tell Maple, which does not have telepathic powers, which one is the unknown it is to solve for. You will get some kind of answer for polynomials of degree less than five. For degree five and higher there is (by a theorem of Galois) no algebraic formula for the roots in terms of the coefficients. The quadratic formula was known to the ancients, e.g., the Babylonians, while the cubic and quartic cases had to wait until the Renaissance for solution. Galois' work, done around 1830, relied on the theory of groups.

Finally, let us consider systems of equations, beginning with a system of first degree:

$$\begin{aligned} 2x + 3y &= 1 \\ 3x + 2y &= 2 \end{aligned}$$

To solve it using Maple, we say this:

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

Note that we write the system as a *set* of equations. Sets consist of a sequence of elements separated by commas, the whole thing enclosed in curly braces. If we had a symbolic parameter, we would have to specify the variables:

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

Note that we can visualize our solution using `plot`. Our first try fails:

```
> plot( { 2*x + 3*y = 1, 3*x + 2*y = 2 }, x = -2..2 );

Error, (in plot) do not know how to plot, 2*x+3*y = 1
```

However, we can get what we want by doing this:

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

Notice that we had to solve for y in terms of x (to express a relation in terms of a function).

Let us now try a system with one linear equation and one quadratic:

```
> solve( { 2*x + 3*y = 1, x^2 + 3*y^2 = 2 }, { x, y} );
> evalf(");
```

We get one solution but decide, for the sake of prudence to check it. For this we attempt to graph the two curves and see where they cross. Again, we have to solve for y in terms of x. We get two solutions, one for each branch of the square root; however, we can plot both of them simultaneously to get the graph of $x^2 + 3y^2 = 2$. Putting this together:

```
> plot( { (1 - 2*x)/3, sqrt( (2 - x^2)/3 ) , -sqrt( (2 - x^2)/3 ) },
  x = -2..2 );
```

Well, that is not *quite* what we wanted: there was a gap between the upper and lower branches of the ellipse. This is because the resolution that Maple uses is too low near the places where the curve is vertical. To remedy this, we try the following:

```
> plot( { (1 - 2*x)/3, sqrt( (2 - x^2)/3 ) , -sqrt( (2 - x^2)/3 ) },
  x = -2..2, resolution=800 );
```

This takes longer but gives a better picture. The default value of `resolution` is 200. For more information about plotting options, try `?plot[options]`.

Alternatively, we could describe the ellipse parametrically. It consists of all points in the xy-plane of the form $(\sqrt{2}\cos t, \sqrt{2}\sin t/\sqrt{3})$ and can be plotted using

```
> plot({ (1 - 2*x)/3,
        [sqrt(2)*cos(t), sqrt(2)*sin(t)/sqrt(3), t = 0..2*Pi] },
        x = -2..2);
```

After making these plots, do you agree that `solve` did its job properly? Can our geometric result be explained algebraically?

Problems

1. Solve $x^2 + 2x + 1 = a$ for $a = 0,\ 0.1, -0.1$. As we change the coefficients by a small amount, how do the roots change? (By a small amount or a large amount?) Try to explain what happens by using appropriate graphs, either sketched by hand or done with Maple.
2. Solve $x^3 - 27 = 0$ and then solve $x^3 + x - 27 = 0$. Comment on the relation between the solutions in the two cases. Does a small change in the coefficients lead to a small change in the roots? Give a graphic explanation if possible.
3. Find all roots of the equation $x^5 + x^4 - x^3 - x^2 - x + 1 = 0$. How many does it have? Explain the behavior of the roots in terms of of the graph of $p = x^5 + x^4 - x^3 - x^2 - x + 1$. Also, find the intervals on which the p is positive. You can do this with

   ```
   > solve( p > 0 )
   ```

   Does knowing these intervals help to interpret what you have found? Explain.
4. Solve $x^n + ax + 1 = 0$ for $n = 2, 3, 4, 5$.
5. Solve the equation $\tan x = x$ on the interval $[-5, 5]$. Make sure that you get *all* the solutions in this interval. For this a graph might help. You have to be careful with `plot` on this one because of the vertical asymptotes of *tan*. Thus, if

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

   doesn't seem to be working well, try restricting the range of y with something like

   ```
   > plot( { tan(x), x }, x = -5..5, y = -10..10);
   ```

6. Solve $\tan x + x = 0.2$. How is the solution found different from that of $\sin x + x = 0.2$? Does this have a graphic explanation? (Do a plot — perhaps with both $y = \tan x + x$ and $y = \sin x + x$ on the same axes). Are there other solutions?

Chapter 6. Defining Your Own Functions

§ 1. Functions

As we have seen in previous chapters, Maple has a large library of standard functions and we can get information about what they are and how to use them from Maple's help files, accessed with commands like

```
> ?library
> ?exp
```

In addition to these standard functions, we can define our own functions in Maple, just as we do in mathematics. For example, to define the function

$$p1(x) = 18x^4 + 69x^3 - 40x^2 - 124x - 48$$

we say:

```
> p1 := x -> 18*x^4 + 69*x^3 - 40*x^2 - 124*x - 48;
```

Think of the symbol `->` in the definition as an arrow: x "goes to" the expression $18x^4 + 69x^3 - 40x^2 - 124x - 48$. We can now proceed with the usual kind of computation, e.g.,

```
> p1( -2 );
> p1( y );
> p1( a+b );
```

If you don't like the format of some of the results that `p1(*)` gives you, try `expand(")`.

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.

```
> p1;          # function
> p1( x );     # expression
> p1( y );     # expression
> p1( 3 );     # expresssion
```

As further proof, let us try to factor `p1`:

```
> factor( p1 );
> factor( p1( x ) );
> factor( p1( y ) );
> plot( p1( x ), x = -5..5 );
```

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

We can define functions of many variables just as easily as we can functions of a single variable:

```
> f := (x,y) -> exp(-x)*sin(y);
> f(1, 2);
> g := (x,y) -> A*exp(-k*x)*sin(w*y);
> g(1,1);
> A := 2: k := 1.4: g(1,1);
> w := 3.7: g(1,1);
> A := 'A'; g(1,1);
```

You may find it interesting to look at the graph of a function of two variables:

```
> plot3d( f(x,y), x = -1..1, y = -1..1, axes = boxed);
```

Whenever we define any function of one variable x we summarize a process to be carried out on a given number x; for example if we define f by $f(x) = 1/\sqrt{x+1}$, we are saying: given a number x, add 1 to it, take the square root of the result and, finally, invert. In this case the individual steps of the process are very simple — or at least involve standard operations — and can be combined in a single formula. There are other functions we might want to define that cannot so easily be summarized with a single formula. In the introduction we looked at one simple example — we know a general procedure to go through in order to decide if a given integer n is prime or not, although actually carrying out that procedure would be very tedious and time-consuming. If we were to explain the process in words we might say something like:

> Check if n is even. If it is, then n is not prime. If the number is odd we should check if any odd numbers smaller than n divide n evenly. If we find such a number, then n is not prime. If we check all the possibilities and we don't find any divisors, then we know that n is prime.

Remark: Of course, this is a very simple-minded approach. We could refine it by checking only prime numbers smaller than n or by realizing that if we don't find a divisor smaller than or equal to $\sqrt{n}$, then we can conclude already that n is prime. (Or we could just use Maple's built-in function `isprime`.) However, we just want to illustrate the idea that we can use Maple to translate a general process into a new Maple definition, without assuming more than knowledge of high school mathematics, so bear with us as we ignore such questions.

We will come back to this question a bit later in the problems. Before we can define this procedure we need to learn about using *loops* in Maple.

§ 2. Loops

To do a sequence of repetitive calculations on a computer, you use a *loop*. Let us try this to print out a short table of square roots:

```
> for i from 1 to 5 do sqrt(i) od;
```

This is not too interesting because Maple gives results in exact form given a choice. Notice that this loop is just short for the repetitive sequence of commands:

```
> i := 1; sqrt(i);
> i := 2; sqrt(i);
> i := 3; sqrt(i);
> i := 4; sqrt(i);
> i := 5; sqrt(i);
> i := 6;
```

Maple sets the loop counter `i` to 1 and carries out the instructions listed between `do` and `od` — this list of instructions is called the *loop body*. Then the counter is increased by 1 and again we carry out the instructions in the loop body, provided the counter `i` has not exceeded the upper limit 5. When the counter reaches 6, Maple knows that the loop is finished.

We modify the example above slightly to get answers in floating point form:

```
> Digits := 4: for i from 1 to 10 do evalf(sqrt(i)) od;
# Set Digits := 10; to go back to default value
```

In the examples above the loop body contained a single Maple command `sqrt(i)` or `evalf( sqrt(i) )`. We can give a longer list of commands in the loop body as the next example illustrates.

Suppose we want to understand the behavior of the infinite sum

$$1 + 1/4 + 1/9 + \cdots + 1/n^2 + \cdots.$$

Remark: Does it make any sense at all to talk about an infinite sum? We have some experience of infinite sums when we talk about infinite decimal expansions — for example, when we claim that 1/3 is the same as the infinite repeating decimal 0.3333... we are giving a numerical value to an infinite sum since this decimal expression is just short for

$$0 + 3/10 + 3/100 + 3/1000 + 3/10000 + \cdots + 3/10^n + \cdots.$$

We want to understand whether or not the sum

$$1 + 1/4 + 1/9 + \cdots + 1/n^2 + \cdots$$

represents a finite number by taking an experimental approach, using Maple loops to add up the terms. If we take just the first term we get 1. If we add the first two terms we get 1.25 or 5/4. To compute the sum of the first three terms we compute $1+1/4+1/9$, which is 49/36 or approximately 1.361111111. (Here we already run into a subtlety in that 1/9 has an infinite decimal expansion so our expression in decimal form is an approximation.) We can write a loop that will continue this process for us, keeping track of the sum in exact, fractional form or computing decimal approximations.

We will let `S` be a variable that holds the accumulated total. To start with we set S equal to 0. Let's add up the first ten terms of the series. Since the ith term of the series is $1/i^2$:

```
> S := 0:
> for i from 1 to 10 do
      S := S + 1/i^2;
      evalf( S );
  od;
```

Here we computed the result both in exact and decimal form.

Remark: In typing the loop command above, notice that we spread the command out over several lines without getting a Maple prompt > at the beginning of each new line. This is the *safest* way to type a loop (or procedure) command. On Sun workstations we do this by hitting the Enter key at the end of each line in the loop. When the loop command is finished we use the Return key to tell Maple to go carry it out. On Macintosh machines the role of the Return key and the Enter key is reversed.

We get results that are somewhat easier to read if we use

```
> S := 0:
  for i from 1 to 10 do
      S := S + 1/i^2:
     [i, S, evalf( S )];
  od;
```

This gives a list of numbers which seem to grow slowly. It is somewhat difficult to understand what is happening with this series; it is helpful to look at a graph of n versus S_n, the sum of the first n terms in the series. We make the loop keep track of data points $[i, S_i]$ for us, ending all our commands with colons to suppress printing of intermediate results:

```
> S := 0:
  for i from 1 to 10 do
      S := S + 1/i^2:
      pts[i] := [ i, S ]:
  od:
```

This command produces no output, but we can display the list of points computed by the loop with

```
> i := 'i';  # Clear i so that the $ command below works

> [ pts[i] $ i = 1..10 ];
```

Or we can make a plot of n versus the sum of the first n terms of the series:

```
> plot( [  pts[i] $ i = 1..10 ], style = line );
```

From the picture we can see that the growth appears to level off. We can easily modify the loop above to compute more terms. Try to carry this out for the first one hundred terms of the series. Does it look like the sum approaches some well-defined number as we take more and more terms?

Warnings:

1. In the example above the picture gives intuitively convincing evidence that this infinite series represents some finite number between 1 and 2; in fact, this series represents the number $\pi^2/6$. However, it is dangerous to conclude anything from our experiment above. In this case we are led to a correct impression, but there are quite simple examples of series that grow without bound as we add more and more terms, although this growth is so slow that we get a quite flat curve when we apply the method above. The series

   $$1 + 1/2 + 1/3 + 1/5 + \cdots + 1/p + \cdots,$$

   where we add up all terms $1/p$ for p a prime integer, is an example of such a series. This series has no limiting value; the sums go to ∞, but *extremely* slowly.

2. In the example above we computed the sums exactly, in fractional form and then at the end we converted to decimal form. This has the major disadvantage that rational arithmetic is *much* slower than decimal arithmetic. We could compute the sum much faster if we said

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

   to force decimal computations instead of rational ones. However, in that case we have to think about the error introduced by approximating numbers like 1/9 by 0.1111111111.

For more about loops, use

```
?for
```

There we find out about the *while* construction. As an example where we use `for ... do ... while` commands consider the infinite series

$$H = 1 + 1/2 + 1/3 + \cdots + 1/n + \cdots.$$

If we experiment with this series as we did with the series above we see that we the growth does not seem to level off as it did when we studied $\Sigma\frac{1}{n^2}$. We can find the number of terms such that the sum

$$H_n = 1 + 1/2 + 1/3 + \cdots + 1/n$$

of the first n terms of the infinite series H exceeds a fixed limit L:

```
> H := 0;
  n := 0;
  L := 6;
  while (H < L)
  do
    n := n+1;
    H := H + 1.0/n;
  od:
  n;
```

This "script" tells us how big n must be so that $H \geq 6$. Note that the loop body (the part between `do` and `od`) can contain more than one statement.

§ 3. Procedures

Finally, let's combine what we have learned about loops with what we know about functions. We will first build a function that computes the sum of the first n cubes. To do this we use the command `proc` as illustrated below. Basically `proc` is used to define a function that is too complicated to express in a single formula.

```
> cubes := proc(n)
  local i, S;
  S := 0:
  for i from 1 to n
    do
      S := S + i^3
    od:
  S;
  end;
```

The `local` statement makes the variables `i` and `S` private to the function `cubes`. This way they can't interfere with variables of the same name used outside the definition of `cubes`. Once the definition is made, we test it:

```
> cubes(1);
> cubes(2);
> cubes(3);
> map( cubes, [1, 2, 3, 4, 5]);
```

The function `map` is used to apply a function or procedure, in this case `cubes` to a list of elements. It then returns a list of values.

There is something we can learn from the experiment just done. It looks like `cubes` is returning numbers which are perfect squares! One way to check this is to apply `sqrt` to the the list, or better, to see what the composite function `sqrt@cubes` does:

```
> map( sqrt@cubes, [ $1..10 ] );
```

Note that `[ $1..10 ]` is a quick way of generating the list `[ 1, 2, ... , 10]`.

Let's do one more example. We will construct a function f such that $f(x)$ is the least integer n such that $H_n > x$. Again, our trick of encapsulation works:

```
> f := proc(L)
          local H, n;
           H := 0;
           n := 0;
           while (H < L)
            do
               n := n+1;
               H := H + 1/n;
             od:
           RETURN( n );
       end;
```

Again, we test:

```
> f(1);
> f(2);
> f(3);
```

Problems

1. In the previous example we looked at the sum

$$H_n = 1 + 1/2 + 1/3 + \cdots + 1/n.$$

We know from calculus that as we take larger and larger values of n this sum gets arbitrarily large. If you choose a number L then no matter how large L is I can find a number n so that the sum H_n is bigger than your number L. In the example above we computed how large n must be if you choose $L = 6$. Find the value of n needed so that H will be bigger than L if $L = 16, 26, 36, 46$. **Warning**: be careful with this one: you may need to stop Maple (or to reset your machine if you can't).

2. Write a loop that computes the value of $m(n) = n^2 + n + 41$ for n from 0 to 25. In each case use the function *isprime* to check that $m(n)$ is prime. Now extend your loop to compute the values of $m(n)$ for n from 0 to 100. What pattern do you see?

Remark: In the previous exercise we might easily be mislead into the conclusion that n^2+n+41 will produce a list of larger and larger prime numbers as we let n increase since this was true for the first 40 values of n. This illustrates the danger of drawing a general conclusion from checking a few cases.

3. Consider the sequence of odd counting numbers:

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

Notice that the first term is just 1^3. The sum of the next two terms is 8 or 2^3. 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 this pattern continues forever?

4. Write a procedure that determines whether an integer n is prime or not using the ideas outlined at the end of the section on functions.

Remarks—More examples with `map` and lists:

1. Sometimes we need to apply functions to lists of things. For this we use `map`, as in the following example:

```
> sqr := x -> x*x;                  # squaring function
  map( sqr, [1, 2, 3, 4, 5] );      # make list of squares
```

A list is a sequence of elements separated by commas and enclosed by brackets. The `map` function takes two arguments: a function, and a list to which the function is applied element by element.

2. Another way to generate the list of square above is to use

```
> i := 'i':  # Clear i
> L := [ i^2 $ i = 1..5 ];
```

3. We can also make lists this way:

```
> L := [ $1..10 ];
```

The `$` operator is used to form an *expression sequence*, and we enclose the result in brackets to turn it into a list. For more about expression sequences, try `?$`.

4. Functions can be composed. This means "put together," from the Latin (*com* = *together*, *ponere* = *to put*). The composition of f and g is $f \circ g$, defined by $(f \circ g)(x) = f(g(x))$. Thus the composite of f and g acts by the rule "do g, then f." The Maple symbol for composition is `@`. Here is an example:

```
> sqr := x -> x*x;
  h:= sqr@sqrt;
  h(2);
  h(5);
```

Here is another:

```
> sqr := x -> x*x;
  k := sqrt@sqr;
  k(2);
  k(5);
```

Notice that h undoes what k does and vice versa: one function is the inverse of the other.

Problems

1. Define a function *inv* which sends a number x to its reciprocal; define another function f which sends x to $\frac{\sin x}{x}$. Study the behavior of f on the sequence $x = 1, 1/2, 1/3, ...1/10$ using *map* and *inv*.
2. Define a function $log(x, b)$ which computes the logarithm of of x to the base b.

Chapter 7. Calculus

Maple also does calculus. With it you can compute derivatives and integrals, both definite and indefinite, find Taylor series, and even solve differential equations (or systems of them).

§ 1. Differentiation

To compute the derivative of the function defined by the expression $x^3 - 2x + 9$ we do this:

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

To compute the second derivative (the derivative of the derivative) we do this:

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

An alternative way is to say

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

This works because the expression $x\$2$ gets translated as the sequence x, x. By analogy, $x\$3$ means x, x, x, so that

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

computes the third derivative — the derivative of the derivative of the derivative.

Suppose now that we are given a function g defined by

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

It seem natural to ask for its derivative. but we must be careful how we phrase our command to Maple:

```
> diff( g, x );    # Maple interprets g as a constant.
> diff( g(x), x ); # Now Maple agrees with us on the meaning of g
```

The subtlety here is that $diff$ operates on expressions, not on functions: g is a function while $g(x)$ is an expression.

We can also compute partial derivatives:

```
> q:=sin( x*y );       # expression, not function
> diff( q, x );        # partial with respect to x
> diff( q, x, y );     # take d/dy of dq/dx
```

§ 2. Integration

To find an indefinite integral (antiderivative) of a function, we use *int*. For example, to compute

$$\int x^n \, dx$$

we do this:

```
> int( x^n, x );
```

Here are some more examples:

```
> int( 1/x, x );
> int( sin(x), x );
> int( x*sin(x), x );                    # Maple knows integration by parts
> int( sin(3*x + 1), x );                # And substitution
> int( (x^2 - 1)/(x^2 + 5*x + 4), x );  # And partial fractions
> int( sqrt(1-x^3), x );                 # but it can't do everything
```

Note that in the last example the answer contains an indefinite integral, a signal that Maple doesn't know how to integrate the given function in terms of elementary functions. Elementary functions are the ones you can build with addition, subtraction, multiplication, division, powers, roots, exponentials, logs, trig functions, and their inverses. Thus $e^{\arcsin x^2}$ is an elementary function. In fact one can prove (by a geometric argument) that the function given by our indefinite integral,

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

cannot be integrated using elementary functions, so even a smarter Maple would not help.

Maple can also compute definite integrals, e.g.,

$$\int_0^1 x^3 \, dx,$$

We do them like this:

```
> int( x^3, x = 0..1 );
```

Note that the only difference is that we give the interval of integration. You can compute definite integrals numerically even when there is no antiderivative in terms of elementary functions. Returning to our example of the previous paragraph, consider

$$F(1/2) = \int_0^{1/2} \sqrt{1 - x^3} \, dx$$

The corresponding indefinite integral cannot be expressed in terms of elementary functions so we cannot use the Fundamental Theorem of Calculus to compute this definite integral. On the other hand, we could estimate the value of $F(1/2)$ if we use Maple to plot the function $\sqrt{1-x^3}$ on the interval $[0,1/2]$ and then draw in a grid and count squares to estimate the area under the curve. (It would probably be helpful to use the option `scaling = constrained` in your `plot` command to ensure that Maple chooses the same scale on the vertical and horizontal axes.) We could write a Maple procedure to carry out this process, using the *definition* of the definite integral as a guide to set up a sum that approximates the area under the curve. By using a fine enough grid we could ensure that we find an approximate value for $F(1/2)$ to any precision we want.

There is a lot to be learned from learning how to set up numerical integration as outlined above, but Maple has already built-in procedures for this:

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

Notice that (unlike the previous definite integral) we had to use `evalf` to force numerical evaluation.

§ 3. The fundamental theorem of calculus

Let us now look at the relationship between differentiation and integration.

```
> f :=  x^2/(x+1);
> int( f, x );
> diff(", x);
> simplify( " );
```

The fundamental theorem of calculus tells us that the derivative of an indefinite integral of f is f once again. We would expect these to match but they do not. Maple agrees with this statement, but only after we apply `simplify` to the result.

Let us now try integrating the derivative:

```
> f :=  x^2/(x+1);
> diff( f, x );
> f2 := int (", x);
> simplify( f2 );
```

This time the result `f2` is definitely different from f. However, if we try

```
> simplify( f - f2 );
```

we see that the two functions differ by a constant. This is as it should be, since any two antiderivatives differ by a constant.

§ 4. Other features

Maple can compute limits:

```
> g := x -> x^3 - 2*x + 9;
> limit( g(x), x = infinity );
> limit( g(x), x = -infinity );
> limit( sin(x)/x, x = 0 );
```

It can also find taylor series and compute sums:

```
> taylor( exp(x), x=0, 4 );   # Taylor series of order 4
> sum(i^2, i=1..100);         # sum of series, returns numerical value
> sum(x^n, n=1..5);           # returns polynomial
> sum(x^n, n=1..infinity)
```

§ 5. Differential equations

To solve the diffferential equation

$$y'' + y = 0$$

we do this:

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

For more information, consult `?dsolve`. There you will find out how to solve initial value problems, systems, find series, numerical, and Laplace transform solutions.

Maple also contains a special package of functions, called the **DEtools** package, for working with differential equations. These functions are not automatically loaded into Maple's working memory when you start your Maple session; to use these functions you must type one of the following:

```
> with(DEtools);   # loads the package; lists the functions in it

or

> with(DEtools):   # loads the package without listing its contents.
```

For more information about the functions in this package type

```
> ?DEtools
```

and then bring up the help files for any commands listed there that look useful to you. Again, pay special attention to the examples at the end of the file.

Chapter 8. Linear Algebra

Maple can solve systems of linear equations and perform most of the computations you need to do with vectors and matrices. You can add, multiply and invert matrices, compute determinants, do row reduction, find eigenvalues and eigenvectors, etc.

§ 1. Linear Equations

Let us consider the problem of solving a system of three equations in three unknowns, say

$$\begin{aligned} 3x + 2y + 4z &= 1 \\ x + 7y - 12z &= 2 \\ 5x + y + 11z &= 4 \end{aligned} \tag{a}$$

In Maple we do this by saying

```
> solve( { 3*x + 2*y +  4*z = 1,
             x + 7*y - 12*z = 2,
           5*x +   y + 11*z = 4 }, { x, y, z } );
```

Notice that we use the asterisk * to represent multiplication. Notice also that the equations and the variables to solve for are enclosed by curly braces. If you have typed in everything correctly you should get as output something like this:

```
( z = -55/11, x = 155/11, y = -115/11 )
```

The coefficients of an equation can be symbols as well as numbers:

$$\begin{aligned} ax + 2y + 4z &= 1 \\ x + by - 12z &= 2 \\ 5x + y + 11z &= 4 \end{aligned} \tag{b}$$

For this example we use

```
solve( {  a*x + 2*y +  4*z = 1,
            x + 7*y - 12*z = 2,
          5*x +   y + 11*z = 4 }, { x, y, z } );
```

Try this now. You will note that the output is more complicated. Let us now study the solution for various values of a. We begin by storing the solution set in the variable `sol`, whiche we do like this:

```
> sol := ":
```

Try this now. We use a colon instead of a semicolon in order to avoid displaying the solution set, which we've already seen. To see what the solution is when $a = 1$, we substitute this value for a in `sol`, which we do like this:

```
subs( a = 1, sol );
```

Try this now. To convert to decimal form:

```
evalf( " );
```

Try this now.

Problem. Find solutions to

$$\begin{aligned} ax + 13y + 11z + 17w &= 1 \\ 2x + 3y + z - w &= 1 \\ 7x + y + 2z + 4w &= 1 \\ x + 11y + 7z - 3w &= 1 \end{aligned}$$

Store the result in a variable `sol`. Then determine the value of `sol` when `a = 9`. Finally, convert the result to decimal form.

Notes:

1. Parts of the solution (stored, say, in `sol`) can be accessed like this:

```
> sol[1];          # gives the first component of the solution.
  z = -56/11       # this is the output
```

Thus `s[2]` is the second part and `s[3]` is the third part. You may want to pick out `-56/11` instead of `z = -45/11`. In that case you say

```
> rhs( sol[1] );
  -56/11;
```

The `rhs` function selects the right-hand side of an equation.

2. You can directly store the result of `solve`, as in

```
S := solve( {9*x + 2*y = 1, 3*x + 5*y = 2}, {x, y} );
```

Then `S[1]`, `S[2]` give the parts of the solution.

§ 2. Basic vector and matrix operations

To work with vectors and matrices we first load the linear algebra package:

```
> with(linalg);  # load the package
```

This is mandatory. As the package is read in, the functions defined in it are displayed. (If you don't want to see them, punctuate with a colon instead of a semicolon).

Let us begin by defining and displaying a vector:

```
> v := vector( [1, -1] );
> v;         # try to display it
> print(v);  # try to display it another way
```

Let us now define another vector w and then do some computations — we will form the sum and the dot product:

```
> w := vector( [1, 1] ):
> add(v,w);
> dotprod(v,w);
```

What does this tell you about the geometry of v and w?

Let us now define two matrices

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

To display them we use `print`:

```
> print(A); print(B);
```

It is easy to use old matrices (or vectors) to make new ones. Try the following to see what you get:

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

Next, we do some computations:

```
> add(A,B);        # can also use evalm( A + B);
> multiply(A,B);   # can also use evalm( A &* B);
```

We check to see if $AB = BA$:

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

It is tiresome to type “multiply” all the time, so we introduce an alias:

```
> alias(m=multiply);
```

We could also have typed `m := multiply`, but aliasing is more efficient. (To unalias, if necessary, we type `alias(m=m)`). Now we can do the following to multiply a matrix times a vector:

```
> m(A,v); # matrix times vector
```

Note that Maple treats vectors as columns, even though it displays them as rows (confusing, isn’t it?):

```
> m(v,A); # doesn't work: Maple treats our vector as a column
> m(transpose(v), A); # transpose row to column
```

The latter works, but note that result is a column vector, and so is displayed in a funny way.

Remark: You can write ordinary algebraic expressions with matrices, but then you have to use `evalm` (“evaluate-matrix”) to get the results you want. Thus `evalm( 2*A )` multiplies a matrix by the scalar 2, while `evalm( A &* B )` multiplies two matrices. We note that things like `print( A + B )` does not work. Consult `?evalm` for more information.

We can also work with determinants (square matrices only!):

```
> det(A);
```

Of course, we can also use symbolic matrices:

```
> A := matrix( [[a, b], [c, d]] );
> det(A);
> a := 2; print(A);
```

Maple didn’t reflect the definition `a := 2`, but it really is there, as we see when we do computations:

```
> det(A);
> a := 'a';  # unassign value (clear a)
> det(A);    # check determinant
```

If the determinant of a square matrix is nonzero, the matrix has an inverse A^{-1}. This is a matrix with the property

$$AA^{-1} = A^{-1}A = I$$

where I is the $n \times n$ identity matrix (one’s on the diagonal, zeroes elsewhere). The identity matrix satisfies

$$Ix = x$$

for any vector x. The inverse is useful in solving equations:

$$\begin{aligned} Ax &= b \\ A^{-1}Ax &= A^{-1}b \\ Ix &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

Here is an example:

```
> A := randmatrix(4, 4, entries = rand(-2..2));
```

creates a 4×4 matrix A with entries chosen randomly from the set $\{-2, -1, 0, 1, 2\}$. Then we can invert A, storing under the name B:

```
> B := inverse(A);
> multiply(A,B); multiply(B,A);  # check result
```

It is an interesting fact that a matrix with random entries is invertible with probability one: the odds favor generation of invertible matrices. Note that you can also say

```
> A := randmatrix(4, 4);
```

In that case the entries are random integers in the range $[-99,\ 99]$.

Problems.

Vectors. In the problems below, let $v = (1, 2, 3)$, $w = (-1, 4, 1)$.

1. Find the sum and dot product of v and w.
2. Compute the length of v using the formula

$$\text{Length}(v) = \sqrt{v \cdot v}.$$

3. Define a length function for vectors by

```
Length := x -> sqrt( dotprod(v,v) );
```

Then compute `Length(v)` and make sure that it agrees with what you obtained in the previous problem. Finally, convert to floating point form with `evalf`, e.g., `evalf(Length(v))`.

4. The angle between two vectors is given by the formula

$$\theta = \frac{\arccos(v \cdot w)}{|v|\,|w|}$$

where $|v|$ stands for the length of v. Define a function for computing angles by

```
Angle = (v,w) -> arccos( dotprod( v, w ) )/(Length(v)*Length(w));
```

Then use this function to compute the angle between v and w. You may want to use `evalf` to put the result in decimal form.

5. Is the result of the previous problem in radians or degrees? You may want to modify the definition of `A` to suit your preferences as regards unit of measure.
6. Maple also contains a `crossprod` function:

$$v \times w = crossprod(v, w)$$

Use it to compute $v \times w$ and comment on the geometric significance of the result.

Matrices. In the problems below, let

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

7. Compute AB and BA. Are they the same?
8. Compute $A + B$ and $B + A$. Are they the same?
9. Compute $7A$.
10. Define `C := add(A,B)`. Then compute C^2.
11. Compute the determinants of A and B. Is there a geometric interpretation of your result?
12. Where possible, compute the inverses of A and B. Check your result by multiplying: If $U := A^{-1}$, then $UA = I$ and $AU = I$, where I is the appopriate identity matrix.
13. Compute Av. Can you also compute vA? (Think about row versus column vectors).
14. Let x be an unknown vector. Solve the equation $Bx = v$. You can do this by noting that $x = B^{-1}v$.

    The next two problems require a bit of thought (or theory).
15. Can you solve the equation $Ax = v$?
16. Can you solve the equation $Ax = u$, where u is the column vector with components $(6, 15, 24)$?

§ 3. Row operations and row reduction

The functions *addrow*, *mulrow*, *swaprow* allow you to do row reduction in much the same way you do it by hand, just skipping the arithmetic. There are companion functions for column operations. Alternatively, you can use `gausselim` or `gaussjord` to put a matrix in reduced form without going through the reduction process yourself. For example, try:

```
> A := matrix( [[3, 2, 2], [-3, -1, -2], [1, 1, 1]] );
> gaussjord( A );
> gausselim( A );
```

If you want to go through the reduction process yourself you can use the example below as a guide. Here we work through the standard procedure for computing the inverse of a matrix by row reduction.

First, we define the 3×3 matrix A, a 3×3 identity matrix, and the matrix we get by augmenting A with the identity.

```
> A := matrix( [[3, 2, 2], [-3, -1, -2], [1, 1, 1]] );
> B := diag(1,1,1);
> C := augment(A,B);
```

Next we clean the part of A on and below the main diagonal: At each step we rename the transformed matrix C. At the end of the process, C will be the matrix whose left 3×3 block is the identity and whose right 3×3 block is A^{-1}.

```
> C := swaprow(C, 1, 3); # get a one in the (1,1) position
> C := addrow(C, 1, 2, 3); # start cleaning column 1
> C := addrow(C, 1, 3, -3); # finish cleaning column 1
> C := mulrow(C, 2, 1/2); # get a one in the (2,2) position
> C := addrow(C, 2, 3, 1); # clean up lower part of column 2
> C := mulrow(C, 3, -2); # get a 1 in the (3,3) position
```

At this point, the left block of C should be upper triangular, with $1's$ along the main diagonal. We continue:

```
> C := addrow(C, 3, 1, -1); # start cleaning upper part of column 3
> C := addrow(C, 3, 2, -1/2); # finish cleaning upper part of column 3
> C := addrow(C, 2, 1, -1); # clean upper part of column 2
```

Now C has the form we want. Now we extract the right 3×3 block; this must be A^{-1}:

```
> D := submatrix(C, 1..3, 4..6);
> multiply(A,D); # should get identity matrix
```

Here is one more example:

```
> A := randmatrix( 3, 7, entries = rand(0..4) );
> gausselim(A);
> gaussjord(A);
```

Notice that *gausselim* gives the row echelon form, while *gaussjord* gives the completely reduced row echelon form.

§ 4. Eigenvalues and eigenvectors

There will be more in this space later. For now consult ?*eigenvals*, ?*eigenvectors*, and ?*charpoly*. However, we would like to mention now one numerical method which is quite easy to implement in Maple. Let A be a matrix, x a vector. Form the product $y = Ax$, then replace x by $y/||y||$, where $||y||$ is the norm, or length. Repeat this process many times and call the resulting vector z. Then with probability one the z is very nearly an eigenvector of A, provided that A has eigenvectors. In fact, it approximates the eigenvector associated with the eigenvalue of largest absolute value. That eigenvalue (the dominant eigenvalue) is approximately

$$\lambda = \frac{Az \cdot z}{z \cdot z},$$

where the dot is the dot product. Here is a definition of a Maple function which does the necessary work:

```
deigenvector := proc(A,x,n)
  x[1] := x[1]/1.0; # force floating point calculations
  for i from 1 to n do
    x := multiply(A,x);
    x := scalarmul(x, 1/norm(x) );
    print(i, x );
  od;
  x;
end;

deigenval := proc(A,x)
   dotprod( multiply( A, x ), x )/dotprod( x, x );
end;
```

Here is an example:

```
> with(linalg):
> A := matrix( [[3,1], [1,2]] );
> v := vector( [1, 1]);
> w := deigenvector( A, v, 10);
> deigenval( A, w );
```

There are also several built-in functions for getting eigenvalues and eigenvalues. Here are some examples:

```
>  A := matrix( [[2, 1], [1, 2]]);
>  eigenval(A); # computes eigenvalues of A
```

The method used is an algebraic one, so it works less well when the eigenvalues are not rational numbers or simple algebraic ones:

```
> A := matrix( [[ -3, -2,  3, -1 ],
                [ -1, -3,  2,  3 ],
                [ -1, -2,  1, -3 ],
                [  2,  3, -2, -2 ]]);
  eigenvals(A);
```

We can (if we want) convert the result to readable decimal form with `map( evalf, ["]);` the brackets are needed to convert the sequence of eigenvalues (an expression sequence) into a list, which is what `map` needs.

For eigenvectors we can use

```
> eigenvects(A);
[1, 1, {[ -1, 1 ]}], [3, 1, {[ 1, 1 ]}]
```

The output is a bit hard to read, but here is how we parse it: the result is an expression sequence, each element of which is a list giving (a) eigenvalue, (b) multiplicity, (c) a set whose elements form a basis for the associated eigenspace. The first element is `[1, 1, {[ -1, 1 ]}]`, so the eigenvalue is 1 (with multiplicity one) and the eigenspace is spanned by the vector `[ -1, 1 ]`.

This last method was pretty complicated. It works best for matrices with rational coefficients and rational eigenvalues and other special situations. Another method is to use the `Eigenvects` function (notice the capital `E`). This implements a numerical method (the QR algorithm).

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

  e := evalf( Eigenvals(A, v);  # store eigenvalues in  e
  print( v );                   # print matrix of eigenvectors
```

The eigenvalues are stored in `e`, and the eigenvector corresponding to the i-th eigenvalue is stored as the i-th column of v. Thus the vector that belongs to `e[1]` is `col( V, 1 )`. Notice that we used `evalf` to force floating point evaluation.

For more information consult the following:

```
> ?eigenvals
> ?eigenvects
> ?Eigenvects
> ?col
```

Bibliography

[1] Bruce W. Char et al., *First Leaves: A Tutorial Introduction to Maple*, WATCOM Publications 1988, pp. 115.

[2] Bruce W. Char et al., *Maple Reference Manual*, 5th Edition, WATCOM Publications 1988, pp. 403.