A Brief Guide to Maple

(c) James Carlson, Jennifer Johnson

Draft of September 13, 1993

Note: *This manuscript is available in the form of a postscript file. Look for* `briefGuide.ps` *in the directory* `/u/cl/lib/maple`.

Contents

1. Introduction 2
2. Basics 6
3. Algebraic expressions 11
4. Graphing 14
   1. Ordinary graphs 14
   2. Parametric equations 15
   3. Polar equations 15
   4. Plotting data 16
   5. More information 16
   6. Printing plots 17
5. Roots and equations 18
6. Functions 23
7. Loops 26
8. Calculus 29
   1. Differentiation 29
   2. Integration 29
   3. The fundamental theorem of calculus 30
   4. Other features 31
   5. Differential equations 31
9. Linear Algebra: Equations, Vectors, and Matrices 33
   1. Linear Equations 33
   2. Basic vector and matrix operations 34
   3. Row operations and row reduction 38
   4. Eigenvalues and eigenvectors 39
10. Saving your work 41

Chapter 1. Introduction

These notes are intended as an introduction to Maple, a computer language which can be used to do many kinds mathematical computations, plot graphs, etc. Let us take a quick tour to see what is possible. First of all, we can do ordinary arithmetic:

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

Maple will will respond with:

$$\frac{261}{7}$$

If we want our answer in decimal (approximate) form, we say

```
>  evalf(");
```

and Maple responds with

$$37.28571429$$

The function *evalf* converts expressions to decimal (floating point) form. The symbol " stands for the result of the last calculation.

Notice that Maple commands end with semicolon, although we won't always put one when a command is described in the middle of a paragraph. Thus, if we say `(1 + 2)*(6 + 7) - 12/7`, the semicolon is implied.

We can use all the standard mathematical functions provided that we know (or can guess) their names. Thus the command

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

produces the output

$$\sin(\sin(1.6*\pi)) + \sqrt{2} + \arctan(3);$$
$$1.849229726$$

Note that whenever possible Maple tries to compute exact quantities and 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.

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

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

produces

$$(a+b)^2$$

as output. If we then say

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

we obtain

$$a^2 + 2ab + b^2$$

By factoring it we reverse the process of expansion: the command

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

yields

$$(a + b)^2$$

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:

$$p := (a + 1)^2$$

Note that sometimes we use colons instead of semicolons; this supresses display of the result.

To expand the result of our last computation, we say `expand(")` and get, as expected, $a^2 + 2a + 1$. Variables can be returned to their original symbolic (unasssigned) state. To do this we say `b := 'b'`. Note the kind of quotes used: the possibilities are `‘`, `’`, and `"`, all of which are different. Now if we say `p` to display the value of p, we get $(a + b)^2$ as output.

Note that in the above examples, the symbol used for assigning values, sometimes called "gets", is different from the equal sign "=." Note also how we "unassign" a variable, so that it becomes a pure symbol once again.

Maple knows much more than arithmetic and algebra — with it you can define new functions and plot them, solve equations in one or more unknowns, do calculus (compute derivatives, integrals, and solve some differential equations), work with vectors and matrices, etc. Here are some examples:

```
> diff( x^n, x );             # differentiate x^n with respect to x
> int( x^n, x);               # integrate x^n with respect to x
> int( x^n, x = 0..1)         # compute a definite integral
> solve( x^3 + x = 27.0, x);  # solve an equation
> fsolve( x^3 - x = 27, x);   # solve it numerically
```

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 ))`. Note also the difference between `solve( x^3 + x = 27, x )` and `solve( x^3 + x = 27.0, x )`. In the first case Maple will report an exact but difficult-to-read answer. In the second it will give the answer in decimal form.

We can also plot graphs:

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

Thus Maple can greatly extend the ease and range of what we are able to compute.

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.

For more information about Maple see the references in the annotated bibliography at the end of these notes.

Exercises

If you have access to a Maple system, you can already try the following exercises.

1. Compute $2^{100}-1$, then find its prime factors using `ifactor`. Example: `ifactor(2^31 - 1)`.
2. Compute $(a+2b+3c)^4$. Use `expand` get the result you expect. What is the coefficient of ab^2c?
3. Find the derivative of $sin(x^2)$. Check Maple's computation by hand.
4. Find the indefinite integral of $1/(x^2+5)$.
5. Compute $\int_0^1 \frac{dx}{x^2+5}$. Example: `int(x^2, x = 0..1)` computes $\int_0^1 x^2 dx$. Use `evalf( " )` to convert the result to decimal form.
6. Solve the cubic equation $x^3 = 27$. Then solve $x^3 + 0.1 * x = 27$. Notice that the coefficients of the the second equation are close to the coefficients of the first one. So are the solutions.
7. 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.

Chapter 2. Basics

To start Maple, type the command `maple` or `xmaple` on the command line. The secons version, if available, will give better graphics. On a windowing system it is better to say `xmaple &`. Leave Maple by typing `quit`. Below is a sample session, in which we do some simple arithmetic:

```
% maple
     > 2 + 2;
                                    4
     > quit
%
```

Note the 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 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. It is the semicolon and not the return key that tells Maple you have finished a command. 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` press the `interrupt` button in the Maple window.

Addition, subtraction, etc., are standard, and parentheses are used in the usual way. Be sure to use an asterix for multiplication: $2 * 3 = 6$. To take powers, use the caret symbol:

```
     > 2^10;
```

Note that Maple carries out exact rational arithmetic. 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 stands for the result of the immediately preceding computations. Here are some examples (remember — do them now):

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

Note that *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

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.

Functions work in Maple as they do in mathematics. For example, to compute $2 + 14 * 8 + |-14| - \sqrt{\sin(1)}$, say

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

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 you need additional information on something, e.g., arctan, try this:

```
> ?arctan
```

Note that there is no semicolon this time.

Let's now take a big step forward and begin using variables. We assign integer values to variables a and b and then do some computations:

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

Note the special symbol used to assign a value to a variable. You can read $a := 25$ as "a gets 25". To get a decimal approximation for a result just computed, recall that we use $evalf$. 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 examples. 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.

Exercises

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

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.

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.

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

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

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 unit. 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 $collect(p, x)$.

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 $-\pi \leq x\pi$, we say

```
> plot( sin(x*3), 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 $y = \sin(3x)$ 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.

Multiple 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 root of $x^3 + x - 27 = 0$ near there. 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 of $\tan(x) = x$. 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 );
```

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), a..b ], coords = polar );
```

For an easy example, consider a circle of radius one. Its polar equations are $r(t) = 1$, $\theta(t) = t$, so we use

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

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, 0..2*Pi ], coords = polar );
```

More examples:

Consider a circle of radius one with a circle of radius a which rolls inside it. The curve traced by a fixed point of the second circle is called a *cardiod*, and is given by the polar equation $r = a(1 + \cos\theta)$. 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 $t = 0$.

§ 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.

§ 5. More information

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]
```

etc.

§ 6. Printing plots

This is easy or difficult, depending on the design of the system. For Maple on Macintosh computers, make sure the plot window is active (click on it), then select **print** in the **file** menu. For other systems, consult your local guru or documentation.

Exercises

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. 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?
4. Plot the curve $r = 1 + 2\cos(\theta)$, $\theta \in [0, 2\pi]$.

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 out 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., "appears twice as in a factor." 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 more useful. We can get it like this:

```
> solve (p = 1.0);

      - .6672857431 + .08764824247 I,
      - .6672857431 - .08764824247 I,
        1.502146598,
      -4.000908446
```

Note that two of the roots are complex.

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

Another way to solve equations is with the function `fsolve`. It gives 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

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

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. For example, 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 look 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* 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`.

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);
                                   {x < 0}, {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);
Error, (in solve/ineqs) not implemented yet
```

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. 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:

$$2x + 3y = 1$$
$$3x + 2y = 2$$

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) relational plots not implemented yet, 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, which gives us what we want:

```
> 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 tangent line 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]`.

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

Exercises

1. Solve $x^2 + 2 * x + 1 = a$ 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 $x^3 - 27 = 0$, $x^3 + x - 27 = 0$, and 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 the roots of 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 $x^n + a * x + 1 = 0$ 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. Functions

With Maple, just as in mathematics, one can define new functions. For example, to define the function

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

we say this:

```
> 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 $18 * x^4 + 69 * x^3 - 40 * x^2 - 124 * x - 48$. 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(")`.

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);
```

Although we don't really need it now, we should mention that there is another way of defining functions. For example, to define the function $f(x,y) = e^x \sin(x)$, we could say this.

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

Remarks:

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. We can also make lists this way:

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

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

Exercises

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. 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 10 do sqrt(i) od;
```

We modify this 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
```

Results can be accumulated inside a loop. We illustrate this by computing the sum of the first ten cubes,

$$S = 1^3 + 2^3 + \cdots + 10^3$$

```
> S := 0:
  for i from 1 to 10
    do
      S := S + i^3
    od;
```

If you don't want to see all the intermediate results, do this:

```
>  S := 0:
   for i from 1 to 10
     do
      S := S + i^3
     od:
   S;
```

Recall that the colon is used in place of the semicolon to suppress output.

For more about loops, use

```
?for
```

There we find out about the *while* construction. As an example of how to use it we find the number of terms such that the sum

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

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.

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 encapsulate the loop used above in the proc style function definition discussed in the previous chapter:

```
> 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]);
```

Recall from the remarks at the end of the last chapter that map applies a function to a list of elements and returns a list of values.

There is something we can lear 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 bettter, so see what the composite function sqrt@cubes does:

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

Recall 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);
```

Exercises

1. In the previous example we looked at the sum

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

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 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?

3. 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. 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?

Chapter 8. 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 );     # partial with respect to y
```

§ 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 integral 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 — the functions you make by doing the usual operations of algebra, together with extraction of roots on polynomials, logs, exponentials, trig functions, and their inverses. However, we can certainly find an approximate value for the definite integral above, accurate to any precision we want:

```
> 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 );  # limit
```

It can also do taylor series and do 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)      # returns -x/(x-1)
```

§ 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, do series, numerical, and Laplace transform solutions.

There is a special package called ODE which allows you to plot line fields, vector fields integral curves, etc. This package is courtesy of Daniel Schwalbe. The main routines are

```
fieldplot:   create a direction field plot for a first order ODE

phaseplot:   create a phase plot with flow lines for two first order
             ODE's

firsteuler:   solves a first order ODE using Eulers method

impeuler:    solves a first order ODE using the "improved Euler method"

rungekuta:   numerical solution to a system of first order ODEs using
             a 4'th order Runge-Kutta method.  sion of rungekutta
```

To use the package (which on our system resides in /u/cl/maple), type

```
> read `/u/cl/maple/ODE`;
```

Once this is done you can get more information about the above routines by typing ?*fieldplot*, ?*phaseplot*, etc. Pay special heed to the examples at the end of each help screen. They may help you more than the general description does.

Note: On Unix systems there you can make a file called `.mapleinit` which tells Maple to do things before it starts up. On our system a worthwhile line in your `.mapleinit` file is

```
read(`/u/cl/maple/iread`):
```

This causes Maple to define a function called `iread` which gives you easy access to whatever our standard Maple library is. Thus, to use the `ODE` package, you just say

```
> iread(ODE);
```

Chapter 9. Linear Algebra: Equations, Vectors, and Matrices

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 conside 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. Finally, note the semicolon at the end of of the `solve` command. Maple is quite finicky about such details of punctuation. 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. The quote sign stands for the most recently computed result — in this case the result of solving equation (a). A phrase like `a := b` means "store the value of `b` in `a`." 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 the result of the most recently computed expression to decimal form we use the `evalf` function, which means "evaluate in floating point 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.

3. For more information about `solve` use the the Maple command

```
> ?solve
```

Note that there is no semicolon.

§ 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 & 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));
  # create random matrix
  > 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 $x = B^{-1}v$.
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. There are companion functions for column operations. To row reduce a matrix, use *gausselim* or *gaussjord*. To illustrate these, we work out an example. It illustrates the standard procedure for computing the inverse of a matrix by row reduction.

```
#  Set things up:
>  A := matrix( [[3, 2, 2], [-3, -1, -2], [1, 1, 1]] );
>  B := diag(1,1,1); # make 3x3 identity matrix
>  C := augment(A,B); # make 3x6 matrix with A as left block, B as right block

    #  Clean part on and below diagonal
>  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

    #  Clean part above diagonal
    >  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
    >  D := submatrix(C, 1..3, 4..6); # extract right-hand block

    #  Verify that D is inverse of A
>  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
```

Chapter 10. Saving your work

You can save your work to a file using the function "save", and you can read it from a file using "read". Type

```
> ?save
> ?read
```

for instructions on how to use these.

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.