

MLC Lab Visit - Introduction to Maple

Mth 351 July 11 2002 Maple 6

Bent E. Petersen

File: 351u2002_lab_visit.mws

I use Maple 6 at present though I may upgrade to Maple 7 soon (though apparently 8 is out). Some of the commands below may not work correctly in Maple 5 and earlier, and some may not work correctly in Maple 7 or later

If you are viewing the MWS version of this document you will note all the Maple output has been removed. You will have to execute each command (by pressing Enter when the cursor is on the command line) to see the output. Take the opportunity to experiment! Change some things. If you are viewing the PDF version, most of Maple's output will be visible, though I may suppress some.

Click on the + signs below to open the corresponding sections. Click on the - signs to close the sections up again. The sections with blue titles are instructional text only. The other sections contain Maple commands. Each section is independent of the other sections and each nontext section starts with a `restart` command to make sure it starts fresh.

Introduction

Maple is a CAS, that is, a Computer Algebra System. It performs mathematical operations symbolically, but a large number of robust numerical routines are also built in. Maple can be used interactively as a rather fancy calculator, but it can also be used as a flexible programming language. Our emphasis in this introduction is on interactive use. Even so we barely scratch the surface.

One feature of Maple that may interest some of you is the possibility of executing Matlab commands from within Maple. In Maple 6 this feature requires that you have Matlab 5 or later installed on the same computer.

The workstations in the MLC lab are PCs running Windows NT 4.0. You must have an ONID account to use the PCs in the MLC lab.

When you logon to a machine in the MLC lab be sure the correct domain is selected - ONID.

When you logon to a machine in the MLC lab your ONID directory on the ONID server will be visible as drive Z: This is where you should keep your personal files. Then they will be available from any PC in the MLC lab (and many other labs) when you login.

Note there is a lab manual (currently much out-of-date) at

http://www.onid.orst.edu/~peterseb/pclab/lab_enchiridion.html

You are unlikely to need any of the information in the manual but you might want to look at it for interest sake. Note in spite of what it may say in the lab manual your user NT user profile is no longer saved (nor restored). Anything you want to save will have to be saved explicitly on your drive Z: Whatever you save on the local drive will be available only on the machine you are working on, and may be deleted periodically, so don't save locally. Use the Z: network drive.

- Login

The machines in the lab are normally left on, but the monitors may be turned off. If the monitor is off, then switch it on. Next press the Ctrl-Alt-Delete keys simultaneously. You should get a login prompt. Enter your ONID user name and press the Tab key (not the Enter key). Then enter your ONID password and press the Enter key.

Next you may see a message about a slow network connection. This message is bogus. Ignore it.

Next you may see a question about a default Novell server. Just answer "none" unless you have a reason to answer otherwise.

To start Maple (or Matlab, or Mathematica, ...) select the Start button (lower left corner of the screen), then Programs from the menu, etc. If you don't know the appropriate steps here ask for help. Writing all this out in detail produces an incredibly dull document.

- Logout

When you are done with your session save your work, shutdown the software you were using and logout. To logout select the Start button (this is a very strange Windows idiom), and then select "Shut Down ..." and finally select "Logon as another user."

Another way to logout is to press Ctrl-Alt-Delete. A menu will appear. Select logoff. That is the simplest way.

Do not select Restart or Shutdown unless you have a reason to do so and do not shut off the PC. There is no harm in shutting off the PC, but doing so causes the next user to have a long wait while the machine reboots. You may turn off the monitor if you wish. That will save power and will reduce the load on the air-conditioner.

Note: If you do not logout you leave your account open for the next person to come along. That person will have access to your personal files on the ONID server. Do not forget to logout!

If you plan to leave the lab, even just for a few minutes, save your work to the Z: drive and logoff.

When you return and logon, even to a different machine, your work will be available. Do not select "Lock Workstation." If you do, someone else wishing to use the workstation may power-cycle it in order to gain access and your unsaved work may be lost as a result. The same comments apply to relying on a password protected screensaver. Don't do it. Save your work and logoff. You have no claim on any workstation if you are not physically present.

Getting Started

Scroll down to the restart command below (in the section, Assignment and Ditto Operators). Position the cursor on the line containing the restart command and press Enter. Maple will execute the restart command and then move the cursor on the next command, skipping over all the intervening text. Now press Enter to execute the next command, etc., or be brave and edit it first. Experiment! Some of the commands depend on previous assignments, etc. If you skip around and something doesn't work you may just have to execute a few of the previous commands.

This entire document was written in Maple. The sample commands were selected to illustrate a few Maple features to get you started using Maple. To learn more you should make heavy use of the online help. The help is very good and usually includes a few examples.

The Worksheet

When you are using Maple in a window environment it is possible to move around on the worksheet by left-clicking the mouse. As a result, commands may end up being executed in a nonlinear order. This can cause some confusion, since there is no visual clue. One way to fix a mess is to have Maple re-execute the whole worksheet (look on the Edit menu). This works best if old expressions are cleaned up first, so it is a good idea to start each worksheet with the command `restart`. You do not need to do so of course

Maple commands are executed by pressing the Enter key when the mouse cursor (pointer, thumb) is in the line containing the commands. Note that Maple skips over the interpolated text comments (like this one). To execute the commands on this worksheet position the mouse cursor on the command line and press Enter. Edit the command first if you wish. Explore! Simply waiting for something to happen will not be productive.

Note some of the commands on this worksheet depend on previous commands. For best results you should probably execute the commands in order.

Note each Maple command must be terminated by a colon or a semicolon (except help commands preceded by a question mark). The effect of the colon is to suppress output from the corresponding command, though the command is still carried out. If you normally use Matlab you may experience some difficulties with the colons and semicolons!

You can spread the command over several lines by postponing the terminating colon or semicolon. You simply move to a new line by pressing Enter. There is no visible line continuation character (as in Matlab). Maple will chatter at you when you move to a new line in this manner if the previous command is unterminated. Ignore it, but keep in mind a command will not be executed before it is properly terminated.

You can also stack up several commands on one line by terminating them individually with colons or semicolons. All the commands on a line are executed when you press the Enter key (with the cursor anywhere on the line).

Here's a useful fact: You can open a new command line below the current one by pressing Ctrl-J, or above the current line, by pressing Ctrl-K. This is pretty useful when you realize you omitted something at a certain step.

Assignment and Ditto Operators

```
[ > restart;
```

The assignment operator in maple is := (colon and equals sign juxtaposed). The equals sign by itself does not perform assignment.

Note once you have assigned a value to a variable you can display the value by entering (properly terminated) the variable name at the prompt.

```
[ > var:=10;
```

```
var := 10
```

```
[ > var;
```

```
10
```

You can retrieve the name of a variable by enclosing it in single quotes. This is handy for producing equations. Thus

```
[ > 'var' = var;
```

```
var = 10
```

```
[ > 'log(3.72)' = log(3.72);
```

```
log(3.72) = 1.313723668
```

Maple has two ditto operators, % and %%. The value of % is the previously evaluated expression, the value of %% is the one before that. Since the Worksheet commands may be executed in any order, the ditto operators can cause a lot of confusion. It is probably best to restrict them to the same line as the expressions they refer to. Here is an example which also

illustrates the quoting mechanism discussed above:

```
> ''log(3.72)'': % = value(%);  
log(3.72) = 1.313723668
```

Note we had to quote twice because the outer set of quotes is stripped off during an assignment and here there is an implicit assignment to `%`. Try using just single quotes to see what happens.

You can also unassign variables. Right now `a` is 5. That would cause problems if we want to use `a` as a dummy variable of integration!

```
> unassign('a','b'); a; b;  
  
a  
b
```

You can pass any number of variables to the `unassign()` command.

A simpler way to unassign one variable is to assign it its name quoted by single quotes (this is a Maple idiom)

```
>  
> a:=5; a:='a': a;  
  
a := 5  
a
```

This is quite convenient, but sometimes the single quotes are hard to find on the keyboard and even harder to see on the monitor. Thus, even though it is more typing you may prefer to use the evaluate to a name function `evaln()` since it does not require the pesky single quotes.

```
> a:=5; b:=4;  
  
a := 5  
b := 4  
> unassign(evaln(a),evaln(b)); a; b;  
  
a  
b
```

Unfortunately, you can pass only one expression to `evaln()`, since it returns only one name.

Some Maple statements may have equal signs in them. It is important to remember that the the equals sign by itself (without a preceding colon) does not perform an assignment.

```
> 3=4; 3:=4;
```

```
3 = 4
```

```
Error, invalid left hand side of assignment
```

Here the error comes from trying to assign 4 to 3. The expression $3=4$ causes no problem though. It is simply an expression. The truth value of an expression may be evaluated by using the evaluate Boolean, `evalb()`, function.

```
> evalb(3=4); evalb(3=3);
```

```
false
```

```
true
```

- Constants

```
> restart;
```

Maple has built-in constants

```
> Pi = evalf(Pi,60); I; I^2; gamma = evalf(gamma,40);
```

```
 $\pi = 3.14159265358979323846264338327950288419716939937510582097494$ 
```

```
 $I$ 
```

```
-1
```

```
 $\gamma = .5772156649015328606065120900824024310422$ 
```

Note the upper case letter for Pi. If you enter pi you will just get the Greek letter pi, not the real number pi.

The `evalf()` function evaluates an expression to floating point. As you can see, the `evalf()` function takes a second parameter specifying the precision in decimal digits. This parameter is optional. If it is not specified then the global constant `Digits` is used (the default value is 10, but you can assign any positive integer value (up to many thousands)).

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

```
3.141592654
```

```
> evalf(Pi,200);
```

```
3.1415926535897932384626433832795028841971693993751058209749445923078164062\  
862089986280348253421170679821480865132823066470938446095505822317253594081\  
284811174502841027019385211055596446229489549303820
```

Note the use of the line continuation character `\` in Maple's response when the response will not fit

on a single line.

- Digits

```
[ > restart;
```

You set the Maple's floating point precision by assigning a value to Digits (the default is 10). Maple usually does exact calculations, but when floating point numbers are involved then Digits sets the precision. Here's an amusing example

```
[ > Digits:=4: convert(evalf(Pi),rational);
```

$$\frac{22}{7}$$

The conversion to a rational number makes use of Digits, rather than any precision specified in the evalf() command. You can easily find other rational approximations to pi

```
[ > Digits:=8: convert(evalf(Pi),rational);
```

$$\frac{355}{113}$$

The label "rational" is protected in Maple 6. You can not assign a value to it (which is just as well).

- Functions and Expressions

```
[ > restart;
```

Maple distinguishes between functions and expressions. Here's one way to define a function:

```
[ > f:=x->sin(3*x+x^2);
```

$$f := x \rightarrow \sin(3x + x^2)$$

We can also define a related expression:

```
[ > g:=sin(3*x+x^2);
```

$$g := \sin(3x + x^2)$$

Both of the examples above assume that x has not already been assigned a value. It needs to be an unassigned variable. In the definition of f the x is a dummy variable, a place marker. In g however, it is part of the expression, and one can refer to it.

To evaluate a function we use the usual function convention. To evaluate an expression one generally uses the subs() command (though it has other subtle uses).

```
> f(1); subs(x=1,g);
```

```
sin(4)
```

```
sin(4)
```

Note the subs() command above does not assign a value to x.

An expression can also be evaluated by using the eval() command, but do check help to make sure you don't have any surprises in more complicated situations. The commands eval() and subs() work in quite different ways. In the simple case that we illustrated here eval() is actually the preferred command to use.

```
> eval(g,x=1);
```

```
sin(4)
```

Note the eval() command above does not assign a value to x.

We can convert an expression into a function by using the unapply() command

```
> h:=unapply(g,x);
```

```
h := x → sin(3 x + x2)
```

You can think of unapply() as turning the indicated variable(s) into dummy variables or place markers. Thus f(x) is the the function f evaluated at x and unapply(f(x),x) ought to return the function f. Let's check that:

```
> ff:=unapply(f(x),x); (ff-f)(w);
```

```
ff := x → sin(3 x + x2)
```

```
0
```

Sure enough!

If you have an inquisitive nature you probably wonder if Maple has an apply() command. It does but the functional notation is usually more convenient.

```
> is(f(t) = apply(f,t));
```

```
true
```

Derivatives

```
> restart;
```

Some Maple commands work on expressions, some work on functions, and some on both.

```
> f:=x->exp(3*x^2)*cos(4*x); g:=exp(3*x^2)*cos(4*x);
```

$$f := x \rightarrow e^{(3x^2)} \cos(4x)$$

$$g := e^{(3x^2)} \cos(4x)$$

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

$$x \rightarrow 6x e^{(3x^2)} \cos(4x) - 4 e^{(3x^2)} \sin(4x)$$

$$6x e^{(3x^2)} \cos(4x) - 4 e^{(3x^2)} \sin(4x)$$

Here D is the operator for differentiating a function, whereas diff() is used for an expression. Note in diff() you have to specify the variable that you wish to differentiate with respect to.

Second derivatives are no problem

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

$$x \rightarrow -10 e^{(3x^2)} \cos(4x) + 36x^2 e^{(3x^2)} \cos(4x) - 48x e^{(3x^2)} \sin(4x)$$

$$-10 e^{(3x^2)} \cos(4x) + 36x^2 e^{(3x^2)} \cos(4x) - 48x e^{(3x^2)} \sin(4x)$$

but this notation can get out hand. Fortunately there is an alternative! Here are the fourth derivatives as an illustration:

```
> (D@@4)(f); diff(g,x$4);
```

$$x \rightarrow -212 e^{(3x^2)} \cos(4x) - 2160x^2 e^{(3x^2)} \cos(4x) - 192x e^{(3x^2)} \sin(4x)$$

$$+ 1296x^4 e^{(3x^2)} \cos(4x) - 3456x^3 e^{(3x^2)} \sin(4x)$$

$$-212 e^{(3x^2)} \cos(4x) - 2160x^2 e^{(3x^2)} \cos(4x) - 192x e^{(3x^2)} \sin(4x)$$

$$+ 1296x^4 e^{(3x^2)} \cos(4x) - 3456x^3 e^{(3x^2)} \sin(4x)$$

Here \$ is the sequence command. Thus

```
> y$10;
```

$$y, y, y, y, y, y, y, y, y, y$$

The @@ command is a composition command. It can be applied to other functions or operators

```
> g:=x->c*x*(1-x); (g@@2)(x);
```

$$g := x \rightarrow cx(1-x)$$

$$c^2 x(1-x)(1-cx(1-x))$$

Thus

`> solve (g@@2) (x) = x ;`

$$\left\{ x=0, c=c \right\}, \left\{ x=\frac{-1+c}{c}, c=c \right\}, \left\{ x=\frac{\frac{1}{2}c + \frac{1}{2} + \frac{1}{2}\sqrt{-3-2c+c^2}}{c}, c=c \right\},$$

$$\left\{ x=\frac{\frac{1}{2}c + \frac{1}{2} - \frac{1}{2}\sqrt{-3-2c+c^2}}{c}, c=c \right\}$$

gives us the points of period 2 for g.

Partial derivatives of expressions are also easily computed (here once relative to y and three times relative to x):

`> diff (x/(x^2+y^2), x$3, y) ;`

$$-288 \frac{x^2 y}{(x^2 + y^2)^4} + \frac{24 y}{(x^2 + y^2)^3} + \frac{384 x^4 y}{(x^2 + y^2)^5}$$

There is an inert version Diff() of diff(). An inert function returns unevaluated. That may seem strange, but sometimes one can save time by postponing evaluation, or one can prevent Maple from attempting a calculation that will fail at present, but can be carried out later in special cases or different contexts. The most important use of unevaluated commands is to prevent an attempt at a symbolic solution when that is impossible or very complicated. Unevaluated expressions may be evaluated by using the command value(), though there are other ways.

Inert functions, together with the ditto operator can be used to get nicely typeset expressions. See if you can sort out the following:

`> Diff (x/(x^2+y^2), x$3, y) : %=value(%);`

$$\frac{\partial^4}{\partial y \partial x^3} \frac{x}{x^2 + y^2} = -288 \frac{x^2 y}{(x^2 + y^2)^4} + \frac{24 y}{(x^2 + y^2)^3} + \frac{384 x^4 y}{(x^2 + y^2)^5}$$

- Integration

`> restart;`

Let's bring back some fond memories from calculus - the problem of integration. Here's an example to get you started: Once again I use postponed evaluation to get a nicely typeset equation. You don't need to do such trickery, of course, but it's nice to know how.

> `Int(1/(1+x^4),x): % = value(%);`

$$\int \frac{1}{1+x^4} dx = \frac{1}{8}\sqrt{2} \ln\left(\frac{x^2+x\sqrt{2}+1}{x^2-x\sqrt{2}+1}\right) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}+1) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}-1)$$

You can obtain the same effect by writing

> `Int(1/(1+x^4),x) = int(1/(1+x^4),x);`

$$\int \frac{1}{1+x^4} dx = \frac{1}{8}\sqrt{2} \ln\left(\frac{x^2+x\sqrt{2}+1}{x^2-x\sqrt{2}+1}\right) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}+1) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}-1)$$

if you don't mind writing the integrand twice. In both these examples the equals sign is just part of the expression. It is not an assignment. Another approach is

> `expr:=1/(1+x^4),x: Int(expr)=int(expr);`

$$\int \frac{1}{1+x^4} dx = \frac{1}{8}\sqrt{2} \ln\left(\frac{x^2+x\sqrt{2}+1}{x^2-x\sqrt{2}+1}\right) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}+1) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}-1)$$

If you are just interested in evaluating the integral then you can dispense with all the typesetting niceties:

> `int(1/(1+x^4),x);`

$$\frac{1}{8}\sqrt{2} \ln\left(\frac{x^2+x\sqrt{2}+1}{x^2-x\sqrt{2}+1}\right) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}+1) + \frac{1}{4}\sqrt{2} \arctan(x\sqrt{2}-1)$$

Note that `int()` function works on expressions, not functions. Thus to integrate a function you have to convert it to an expression by evaluating it at a dummy variable.

> `h:=t->t^3; int(h(u),u);`

$$h := t \rightarrow t^3$$

$$\frac{1}{4}u^4$$

I bet you wish you had a tool like this when you were studying calculus!

Naturally definite integrals are possible too.

```
> Int(2*x^2*log(x)^3+x^3*log(x),x=1..2): %=value(%);
```

$$\int_1^2 2x^2 \ln(x)^3 + x^3 \ln(x) dx = -\frac{853}{432} + \frac{16}{3} \ln(2)^3 - \frac{16}{3} \ln(2)^2 + \frac{68}{9} \ln(2)$$

If you want a floating point number you can simply use evalf(), but there is a subtle and important difference depending on how you do it.

```
> a:=int(2*x^2*log(x)^3+x^3*log(x),x=1..2): evalf(a,16);
```

```
2.476290396904212
```

```
> evalf(Int(2*x^2*log(x)^3+x^3*log(x),x=1..2),16);
```

```
2.476290396904210
```

In the first case we assign the symbolic expression for the integral to a and then evaluate that expression. In the second example, Maple detects that we want a numeric result and evaluates the integral numerically without first trying to obtain a symbolic solution. This is important. For example

```
> restart;
```

```
> st:=time();
```

```
> int(arctan(x)/log(x),x=Pi/8..Pi/4); evalf(%);
```

$$\int_{1/8\pi}^{1/4\pi} \frac{\arctan(x)}{\ln(x)} dx$$

```
-4.623890373
```

```
> TIME = time() - st;
```

```
TIME = 7.656
```

```
> restart;
```

```
> st:=time();
```

```
> evalf(Int(arctan(x)/log(x),x=Pi/8..Pi/4));
```

```
-4.623890373
```

```
> TIME = time() - st;
```

```
TIME = 2.171
```

Here, in the first case, Maple decided after a while (possibly a long while) that it can not return a symbolic value for the integral and so returned it unevaluated. Then evalf() called a numeric quadrature rule to get an answer. In the second case however, Maple wasted no time trying to find a nonexistent symbolic solution, but instead used a numeric quadrature method. This is an

important use of inert functions. You can grow noticeably older waiting for a symbolic solution to a complex problem.

There are refinements. For example, you can specify what quadrature method to use. Enter the command `?int[numeric]` for more information.

It is worth noting too that symbolic solutions may become very complex. If you end up with an expression with thousands of terms it may not be very useful and may take a long time to display, or even crash Maple. Always save your worksheet before executing a command which may lead to a very complex result, and consider requesting a numeric result when it makes sense to do so.

Here's a silly example of a moderately complex result:

```
> int((a+b*x+c*log(x)+d*x*log(x))^12,x): nops(%);  
3185
```

The output means that the expression for the integral has 3,185 terms in it. If you want to see them, just replace the colon with a semicolon and re-execute the command.

Some Plots

```
> restart;
```

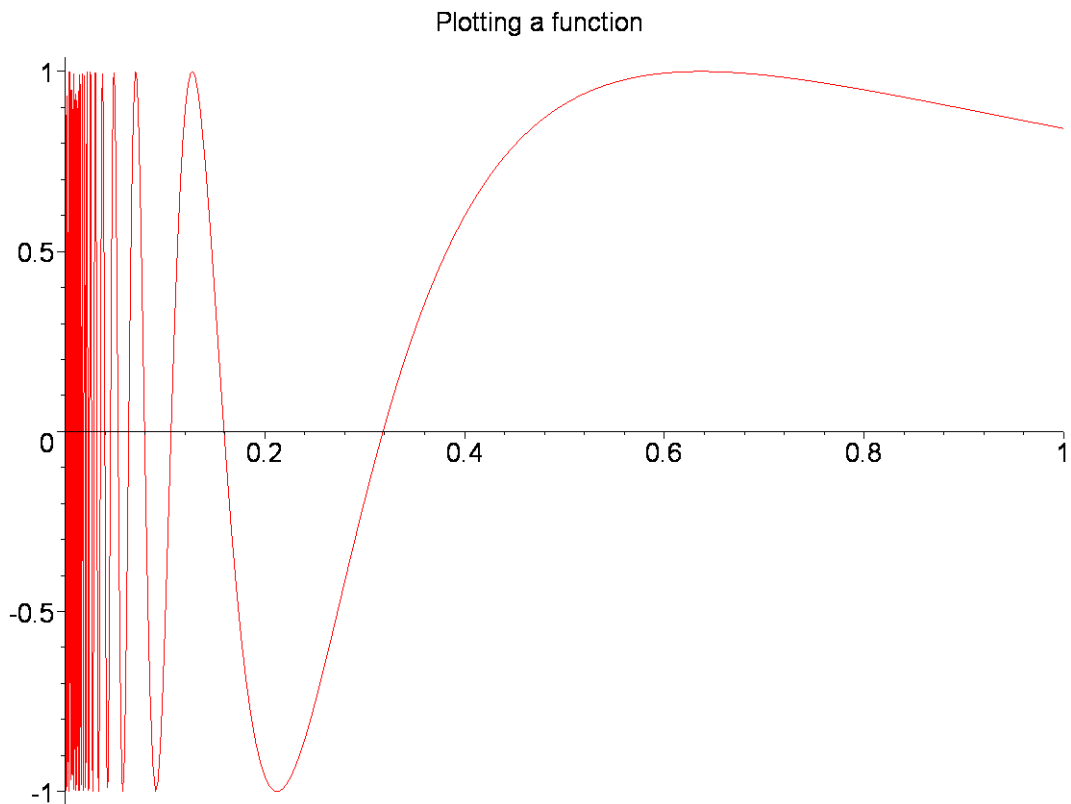
Functions and expressions can be plotted. There are numerous plot variations. Check the help facility, `?plot`, for details.

```
> f:=x->sin(1/x); g:=sin(1/x);
```

$$f := x \rightarrow \sin\left(\frac{1}{x}\right)$$
$$g := \sin\left(\frac{1}{x}\right)$$

```
> plot(g,x=0..1,numpoints=200,title="Plotting an expression");
```

```
> plot(f,0..1,numpoints=200,title="Plotting a function");
```



Replace the colon above by semicolons and press Enter to generate the plot.

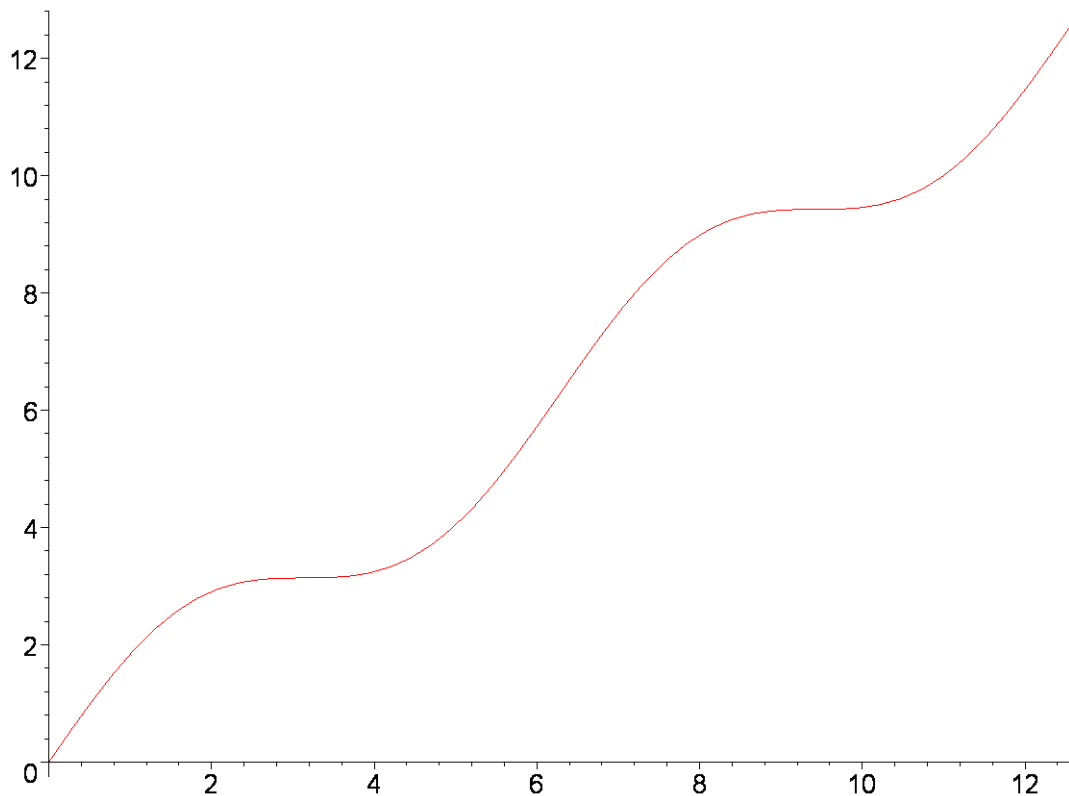
We can convert a function into an expression simply by evaluating it, so one can also do

```
> plot(f(x), x=0..1):
```

Replace the colon above by a semicolon and press Enter to generate the plot.

You can also plot anonymous functions, or expressions, that is, plot them without first assigning them to a variable:

```
> plot(x->x+sin(x), 0..4*Pi);
```



```
> plot(x+sin(x), x=0..4*Pi):
```

Replace the colon above by semicolons and press Enter to generate the plot.

Maple has many plot types. Some 3D plots are demonstrated below, but if you need something else, you will have to explore Maple help to see if you can find what you want..

- More Plots

```
> restart:
```

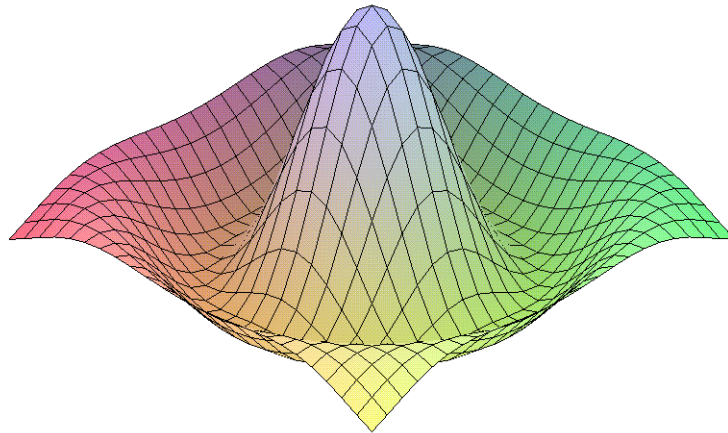
```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

Maple has a number of built-in plot commands. Additional commands are made available by loading the plots package (by means of the with(plots) command).

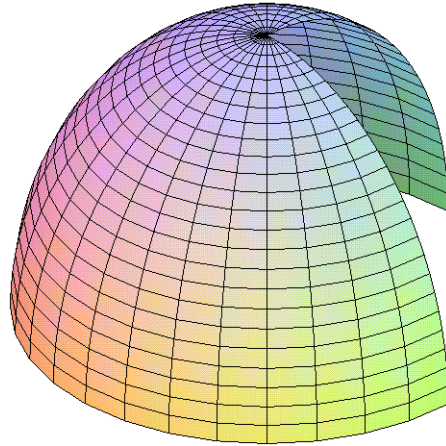
Here is a well-know plot.

```
> plot3d(sin(sqrt(x^2+y^2))/sqrt(x^2+y^2), x=-7..7, y=-7..7);
```



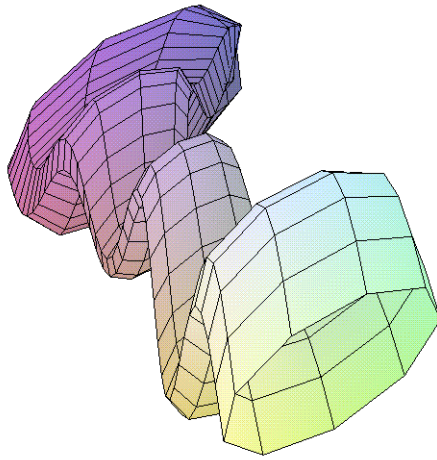
We can also do parametric plots. We will use parameters t and p , so let's make sure first they have not been assigned to some other expressions (otherwise we will get incomprehensible error messages).

```
> t:=evaln(t): p:=evaln(p):  
> plot3d([4*cos(t)*sin(p), 4*sin(t)*sin(p), 4*cos(p)], t=-Pi..Pi/2, p  
=0..Pi/2);
```



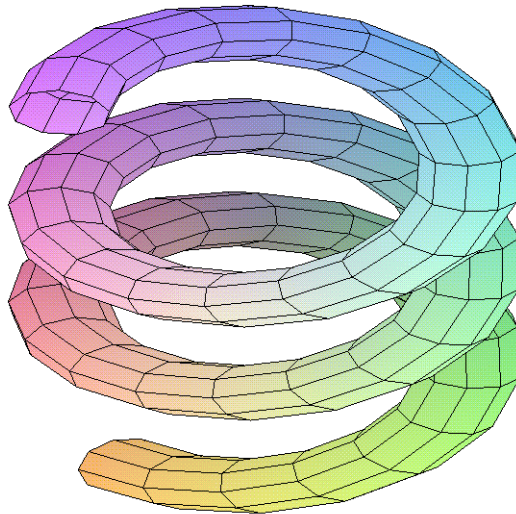
Many other plot commands are available. Check `?plots`. A nice plot to experiment with is the `tubeplot`

```
> with(plots):  
> tubeplot([t, t^2, t*sin(t)], t=-1..2.2, radius=6*(2+cos(t/4)));
```

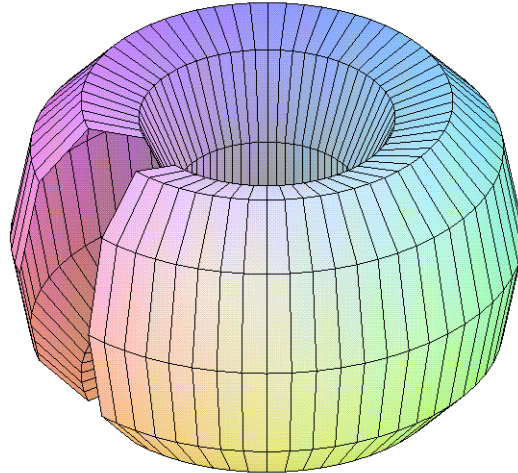


Note you can drag the plot around with the mouse to see the surface from different view points.

```
> tubeplot([4*cos(t), 4*sin(t), 4*t], t=0..18, radius=1);
```



```
> tubeplot([2*cos(t), 2*sin(t), 1], t=0..2*Pi-1/2, radius=1, numpoints=60);
```



- Taylor Series and Taylor Polynomials

```
[ > restart;
```

Maple can compute Taylor and interpolation polynomials. Actually the Taylor polynomial is a special case of an interpolation polynomial, with all the nodes equal. but Maple's interpolation routine requires distinct nodes (we can get around that restriction by using limits).

Let's start with an example of Taylor polynomials:

```
[ > tay1:=taylor(exp(2*sin(x)), x=0, 10);
```

$$tay1 := 1 + 2x + 2x^2 + x^3 - \frac{23}{60}x^5 - \frac{4}{15}x^6 - \frac{19}{360}x^7 + \frac{2}{45}x^8 + \frac{7057}{181440}x^9 + O(x^{10})$$

Note `taylor()` works on expressions. The second argument specifies the center. The third argument specifies the order of the terms omitted. Parameters may be included:

```
[ > a:=evaln(a);
```

$a := a$

```
[ > tay2:=taylor(exp(a*sin(x)), x=0, 5);
```

$$\text{tay2} := 1 + a x + \frac{1}{2} a^2 x^2 + \left(-\frac{1}{6} a + \frac{1}{6} a^3\right) x^3 + \left(-\frac{1}{6} a^2 + \frac{1}{24} a^4\right) x^4 + O(x^5)$$

Note I unevaluated `a` first, because we left it assigned to some number above. If I had not unevaluated it then Maple would have substituted the value of `a` in this expression.

The data type returned by `taylor()` is a series, not a polynomial. If you want a polynomial to play with you need to do a conversion:

```
> taylor(tan(x), x=0, 10) : p:=convert(%, polynomial);
```

$$p := x + \frac{1}{3} x^3 + \frac{2}{15} x^5 + \frac{17}{315} x^7 + \frac{62}{2835} x^9$$

In Maple 6 `polynom` is a reserved name (for a data type) so you do not have to worry that you might have assigned a value to it.

You can specify a different center, even a symbolic one

```
> c:=evaln(c) :
```

```
> taylor(exp(x), x=c, 4) : pc:=convert(%, polynomial);
```

$$pc := e^c + e^c (x - c) + \frac{1}{2} e^c (x - c)^2 + \frac{1}{6} e^c (x - c)^3$$

- Interpolation Polynomials

```
> restart;
```

Maple provides a built-in command for computing interpolation polynomials.

```
> q1:=interp([1,3,4,2], [2,1,3,1], x);
```

$$q1 := \frac{1}{6} x^3 - \frac{1}{2} x^2 - \frac{2}{3} x + 3$$

The first parameter we pass to `interp()` is the list of (distinct) abscissas, the second is the list of ordinates and the third is a name, the name for the variable to be used in the polynomial.

If you want a polynomial function rather than a polynomial expression in some variable, you can use `unapply()`:

```
> q2:=unapply(interp([1,3,4,2], [2,1,3,1], x), x);
```

$$q2 := x \rightarrow \frac{1}{6} x^3 - \frac{1}{2} x^2 - \frac{2}{3} x + 3$$

Let's check that it worked:

```
> q2(1); q2(3); q2(4); q2(2);  
  
2  
1  
3  
1
```

If you have a list of points you want to interpolate you can extract the abscissas and ordinates by using the op() command (it lists the operands in its argument):

```
> L:= [ [1,2], [2,-1], [3,-2], [-1,1], [-2,7], [8,6], [7,5] ];  
L := [[1, 2], [2, -1], [3, -2], [-1, 1], [-2, 7], [8, 6], [7, 5]]
```

We start by declaring two empty lists, XX and YY, and then push the abscissas on XX and the ordinates on YY:

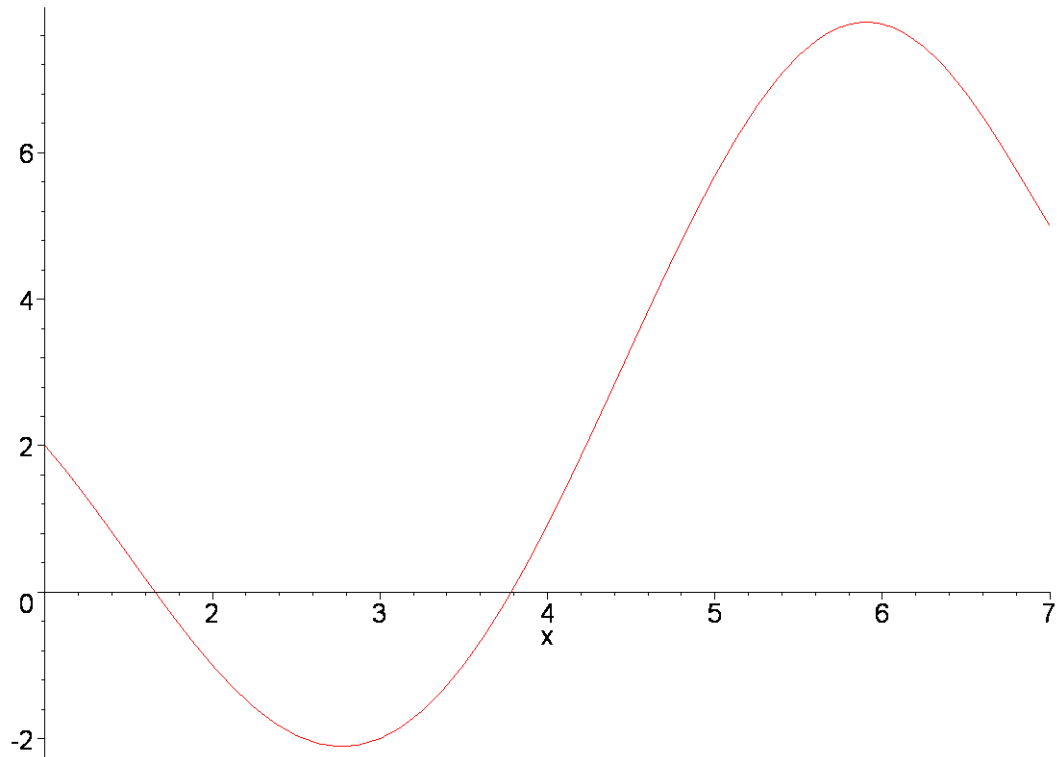
```
> XX:=[]; YY:=[]; for pnt in L do XX:=[op(XX),pnt[1]];  
YY:=[op(YY),pnt[2]]; od;
```

Here pnt is a list with two entries (it represents one of our points), pnt[1] is the first entry in pnt (think the x coordinate), and pnt[2] is the second entry (think the y coordinate). Before we use XX and YY let's check that they look alright

```
> XX; YY;  
  
[1, 2, 3, -1, -2, 8, 7]  
[2, -1, -2, 1, 7, 6, 5]
```

```
> p3:=interp(XX,YY,x);  
  

$$p3 := \frac{79}{16800}x^6 - \frac{521}{6048}x^5 + \frac{23567}{50400}x^4 - \frac{2435}{6048}x^3 - \frac{24403}{12600}x^2 + \frac{1495}{1512}x + \frac{667}{225}$$
  
> plot(p3,x=1..7,title="p3");
```



A convenient way to construct an interpolation polynomial for a function is to use the `map()` command to evaluate the function at each abscissa. Let's consider the sine function on $[0,4]$:

```
> XX := [0, 1/2, 1, 3/2, 2, 5/2, 3, 7/2, 4];
```

$$XX := \left[0, \frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}, 3, \frac{7}{2}, 4 \right]$$

```
> YY := evalf (map (sin, XX) );
```

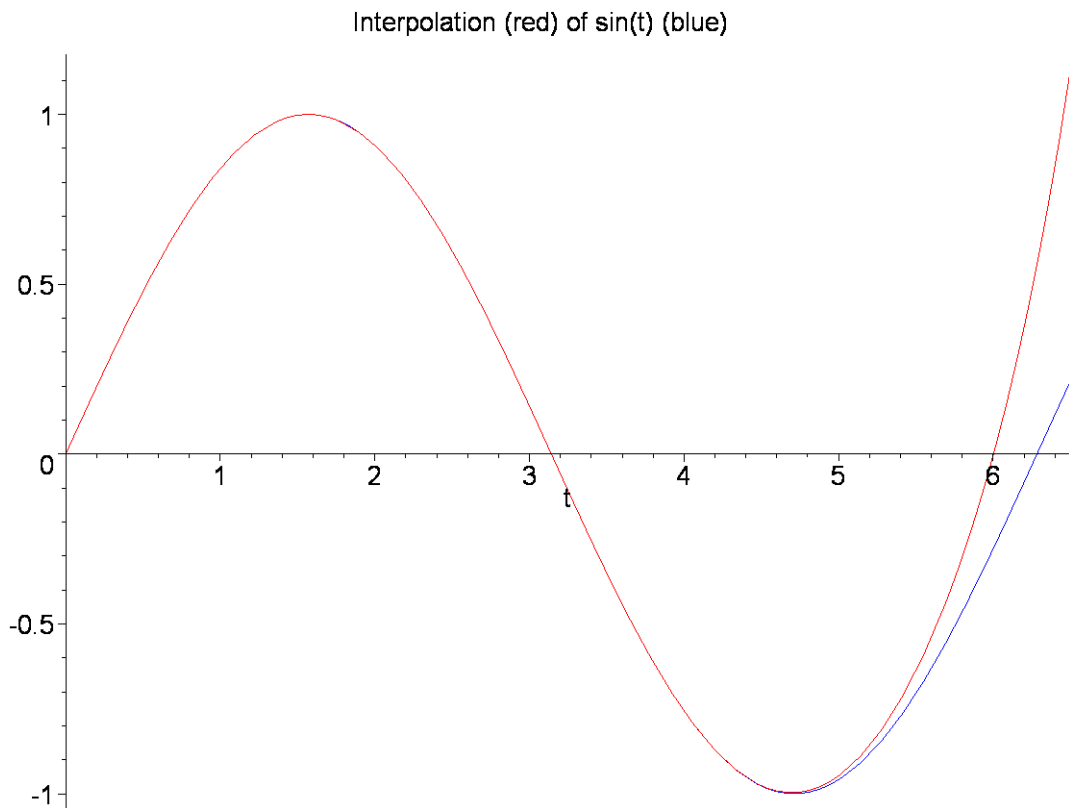
```
YY := [0., .4794255386, .8414709848, .9974949866, .9092974268, .5984721441,
       .1411200081, -.3507832277, -.7568024953]
```

Here we used `evalf()` to force (approximate) evaluation of the sine. Otherwise we will get an (painfully) exact answer. Try it.

```
> ps := interp (XX, YY, t);
```

$$ps := .00002074516762 t^8 - .0002575581726 t^7 + 1.000090277 t + .0000235057535 t^6 \\ - .00045081425 t^2 + .008609497669 t^5 - .1658254022 t^3 - .000739266491 t^4$$

```
> plot ([ps, sin(t)], t=0..6.5, title="Interpolation (red) of sin(t)
      (blue)", color=[red,blue]);
```



Note the previous example shows one way of plotting two functions on one graph.

- Interpolation Splines

```
[ > restart;
```

Maple computes splines of all degrees - check the help. Here we will look only at linear and (natural) cubic splines. A linear spline is just a piecewise linear function. The parameters are much the same as for `interp()`, but the abscissas must be in increasing order.

```
[ > XX := [1, 2, 5/2, 3, 13/4, 15/4, 5];
```

```
          XX := [1, 2, 5/2, 3, 13/4, 15/4, 5]
```

```
[ > YY := [1, 1, 2, 1, -1, -1, 3];
```

```
          YY := [1, 1, 2, 1, -1, -1, 3]
```

```
[ > sp1 := spline(XX, YY, x, linear);
```

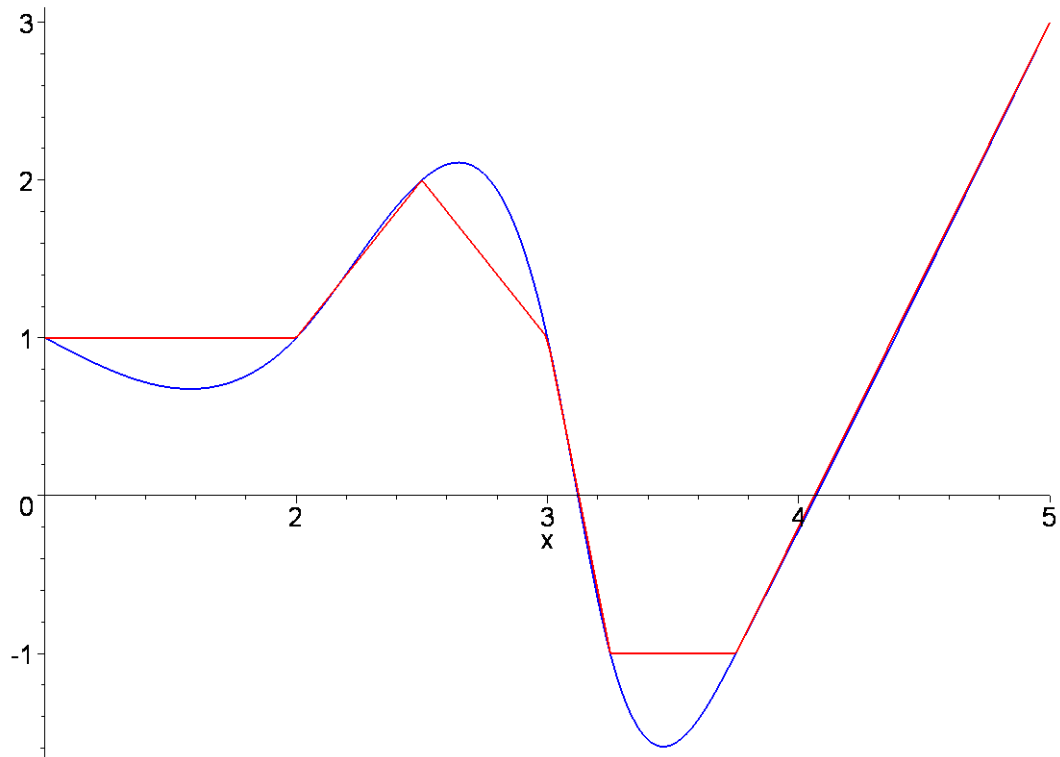
$$sp1 := \begin{cases} 1 & x < 2 \\ -3 + 2x & x < \frac{5}{2} \\ 7 - 2x & x < 3 \\ 25 - 8x & x < \frac{13}{4} \\ -1 & x < \frac{15}{4} \\ -13 + \frac{16}{5}x & otherwise \end{cases}$$

> `sp3:=spline(XX,YY,x,cubic);`

$$sp3 := \begin{cases} 1 + \frac{41008}{24395}x - \frac{61512}{24395}x^2 + \frac{20504}{24395}x^3 & x < 2 \\ \frac{184719}{4879} - \frac{1307792}{24395}x + \frac{612888}{24395}x^2 - \frac{13128}{3485}x^3 & x < \frac{5}{2} \\ \frac{450319}{4879} - \frac{2901392}{24395}x + \frac{1250328}{24395}x^2 - \frac{176888}{24395}x^3 & x < 3 \\ -\frac{4412539}{3485} + \frac{30237976}{24395}x - \frac{9796128}{24395}x^2 + \frac{1050496}{24395}x^3 & x < \frac{13}{4} \\ \frac{3062588}{4879} - \frac{12408836}{24395}x + \frac{3325968}{24395}x^2 - \frac{59072}{4879}x^3 & x < \frac{15}{4} \\ -\frac{43627}{4879} + \frac{16024}{24395}x + \frac{12672}{24395}x^2 - \frac{4224}{121975}x^3 & otherwise \end{cases}$$

> `plot([sp1,sp3],x=1..5,color=[red,blue],thickness=2,numpoints=200,title="piecewise linear (red) and cubic spline (blue) interpolation");`

piecewise linear (red) and cubic spline (blue) interpolation



You can see how the cubic spline smoothens out the graph without introducing too much oscillation. If we compare the piecewise linear spline and the interpolation polynomial we see unreasonable oscillation unsupported by the data:

```
> pp:=interp(XX,YY,x);
```

```
pp :=
```

$$-\frac{128488}{51975}x^6 + \frac{730628}{17325}x^5 - \frac{29835503}{103950}x^4 + \frac{68943997}{69300}x^3 - \frac{95898871}{51975}x^2 + \frac{7969177}{4620}x - \frac{41341}{66}$$

```
> plot([sp1,pp],x=1..5,color=[red,blue],thickness=2,numpoints=200,
,title="piecewise linear (red) and polynomial interpolation
(blue)");
```

Replace the colon above by a semicolon and press Enter to generate the plot.

Note the vertical scales are different in the two graphs. We can plot all three function in one graph for a more convincing demonstration of how well the cubic spline follows the piecewise interpolation.

```
> plot([sp1,sp3,pp],x=1..5,-3..9,color=[red,blue,black],thickness
=2,numpoints=200,title="red=piecewise linear, blue=cubic
spline, black=interpolation polynomial");
```

Replace the colon above by a semicolon and press Enter to generate the plot.

Note how I restricted the vertical range to $-3..9$ so we would be able to see the details (otherwise the piecewise linear and the cubic spline just about merge on the graph).

- Trapezoidal and Simpson's Rules

```
> restart;
```

Maple has numerous high-power quadrature methods built in, but if one simply wants to experiment with the trapezoidal rule or Simpson's rule, these are available in the student package, accessed through the command `with(student)`.

It is also fairly easy to roll your own, even to write high order Newton-Cotes methods, if you wish. There are some example on my web page. For now, let's use the student package.

```
> with(student):
```

```
> trapezoid(f(x), x=a..b, 6);
```

$$\frac{1}{2} \left(\frac{1}{6} b - \frac{1}{6} a \right) \left(f(a) + 2 \left(\sum_{i=1}^5 f \left(a + i \left(\frac{1}{6} b - \frac{1}{6} a \right) \right) \right) + f(b) \right)$$

```
> simpson(f(x), x=a..b, 6);
```

$$\frac{1}{3} \left(\frac{1}{6} b - \frac{1}{6} a \right)$$

$$\left(f(a) + f(b) + 4 \left(\sum_{i=1}^3 f \left(a + (2i-1) \left(\frac{1}{6} b - \frac{1}{6} a \right) \right) \right) + 2 \left(\sum_{i=1}^2 f \left(a + 2i \left(\frac{1}{6} b - \frac{1}{6} a \right) \right) \right) \right)$$

The first command above loads the student package. This package contains the code for the trapezoidal, Simpson's rule and many other things. It is a standard part of Maple.

Let's try an actual function, say $\exp(x) \cdot \cos(x)$.

```
> trapezoid(exp(x) * cos(x), x=0..3, 12): test:=evalf(%);
```

```
test := -9.148761413
```

```
> simpson(exp(x) * cos(x), x=0..3, 12): sest:=evalf(%);
```

```
sest := -9.024261903
```

```
> int(exp(x) * cos(x), x=0..3); evalf(%);
```

$$\frac{1}{2} e^3 \cos(3) + \frac{1}{2} e^3 \sin(3) - \frac{1}{2}$$

```
-9.025029854
```

As we expected, Simpson's rule performs much better here.

- Sums and Products

```
[ > restart;
```

```
[ Maple can compute quite a few standard sums and products:
```

```
[ > expr:=k^2,k=1..n: Sum(expr)=sum(expr); expand(%);
```

$$\sum_{k=1}^n k^2 = \frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

$$\sum_{k=1}^n k^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$$

```
[ > Sum(k^3,k=1..n): %=value(%); expand(%);
```

$$\sum_{k=1}^n k^3 = \frac{1}{4}(n+1)^4 - \frac{1}{2}(n+1)^3 + \frac{1}{4}(n+1)^2$$

$$\sum_{k=1}^n k^3 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2$$

```
[ > Sum(k^4,k=1..n) = sum(k^4,k=1..n); expand(%);
```

$$\sum_{k=1}^n k^4 = \frac{1}{5}(n+1)^5 - \frac{1}{2}(n+1)^4 + \frac{1}{3}(n+1)^3 - \frac{1}{30}n - \frac{1}{30}$$

$$\sum_{k=1}^n k^4 = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n$$

These three examples illustrate different ways of achieving the same typographical effect. Note that `sum()` computes a sum, whereas `Sum()` simply returns unevaluated. The first example also shows that you can assign any kind of expression to a label.

Here's an obvious product

```
[ > product(k/(k+1),k=1..n); simplify(%);
```

$$\frac{\Gamma(n+1)}{\Gamma(n+2)}$$
$$\frac{1}{n+1}$$

- Number Theory

```
[ > restart;
```

```
[ Maple has a some simple number theory support built-in. Additional resources are available in the (standard) numtheory library (loaded by with(numtheory) if needed).
```


$k = 11, false$

If you just want to list known Mersenne primes (Maple has a built-in list) you can use a modified `mersenne()` command:

```
> for k from 1 to 11 do mersenne([k]); od;  
3  
7  
31  
127  
8191  
131071  
524287  
2147483647  
2305843009213693951  
618970019642690137449562111  
162259276829213363391578010288127
```

If you want to find the k 's which gives rise to the Mersenne primes above, you could do the following:

```
> for k from 1 to 11 do 2^n-1 = mersenne([k]), 'n' =  
round(evalf(log[2](mersenne([k])+1))); od;  
2^n - 1 = 3, n = 2  
2^n - 1 = 7, n = 3  
2^n - 1 = 31, n = 5  
2^n - 1 = 127, n = 7  
2^n - 1 = 8191, n = 13  
2^n - 1 = 131071, n = 17  
2^n - 1 = 524287, n = 19  
2^n - 1 = 2147483647, n = 31  
2^n - 1 = 2305843009213693951, n = 61  
2^n - 1 = 618970019642690137449562111, n = 89  
2^n - 1 = 162259276829213363391578010288127, n = 107
```

There's probably a better way to do it!

The Mersenne primes grow pretty quickly. When you get to thousands of digits perhaps you are satisfied just knowing how many digits without seeing the actual number.

```
> for k from 25 to 31 do "digits in k-th mersenne prime", 'k'=k,
  'digits' = length(mersenne([k])); od;
      "digits in k-th mersenne prime", k = 25, digits = 6533
      "digits in k-th mersenne prime", k = 26, digits = 6987
      "digits in k-th mersenne prime", k = 27, digits = 13395
      "digits in k-th mersenne prime", k = 28, digits = 25962
      "digits in k-th mersenne prime", k = 29, digits = 33265
      "digits in k-th mersenne prime", k = 30, digits = 39751
      "digits in k-th mersenne prime", k = 31, digits = 65050
```

A positive integer is a perfect number if it is the sum of its proper divisors. Euclid showed that an even number M is perfect if and only if M is of the form $M = 2^q(q-1)(2^q-1)$ where 2^q-1 is prime (so a Mersenne prime). No one knows if there are any odd perfect numbers. Here's a list of the first few perfect numbers

```
> for k from 1 to 12 do
  n:=mersenne([k])*2^(round(evalf(log[2](mersenne([k])+1)))-1);
  od;
      n := 6
      n := 28
      n := 496
      n := 8128
      n := 33550336
      n := 8589869056
      n := 137438691328
      n := 2305843008139952128
      n := 2658455991569831744654692615953842176
      n := 191561942608236107294793378084303638130997321548169216
      n := 13164036458569648337239753460458722910223472318386943117783728128
n := 14474011154664524427946373126085988481573677491474835889066354349131199\
152128
```

The first 4 have been known since antiquity.

Here is the Euler polynomial that produces so many primes

```
> p:=x->x^2+x+41;
```

$$p := x \rightarrow x^2 + x + 41$$

```
> for k from 1 to 40 do; k, ifactor(p(k)), isprime(p(k)); od;
```

```
1, (43), true  
2, (47), true  
3, (53), true  
4, (61), true  
5, (71), true  
6, (83), true  
7, (97), true  
8, (113), true  
9, (131), true  
10, (151), true  
11, (173), true  
12, (197), true  
13, (223), true  
14, (251), true  
15, (281), true  
16, (313), true  
17, (347), true  
18, (383), true  
19, (421), true  
20, (461), true  
21, (503), true  
22, (547), true  
23, (593), true  
24, (641), true  
25, (691), true  
26, (743), true  
27, (797), true  
28, (853), true  
29, (911), true  
30, (971), true  
31, (1033), true  
32, (1097), true
```

```

33, (1163), true
34, (1231), true
35, (1301), true
36, (1373), true
37, (1447), true
38, (1523), true
39, (1601), true
40, (41)2, false

```

- Solving Equations

```
[ > restart;
```

We can solve a few equations symbolically with solve(), and many more numerically with fsolve(). Note we can specify the range in which to look for a solution.

```
[ > fsolve(tan(x)=3*x,x,avoid={x=0},0..1.4);
1.324194450
```

We didn't actually need the avoid={x=0} here, but it is one way to make sure the trivial solution x=0, is not the one found.

```
[ > fsolve(tan(x)=3*x,x,1.4..5);
4.640683631
```

```
[ > solve(x^4+1=0,x);
```

$$\frac{1}{2}\sqrt{2} + \frac{1}{2}I\sqrt{2}, \frac{1}{2}I\sqrt{2} - \frac{1}{2}\sqrt{2}, -\frac{1}{2}\sqrt{2} - \frac{1}{2}I\sqrt{2}, -\frac{1}{2}I\sqrt{2} + \frac{1}{2}\sqrt{2}$$

```
[ > solve(sin(x)=cos(x),x);
```

$$\frac{1}{4}\pi$$

We can also solve some systems

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

$$\left\{ y = \frac{1}{2}, x = 2 \right\}$$

For serious work with linear equations use the linear algebra packages, linalg or LinearAlgebra.

- Linear Algebra (linalg package)

```
[ > restart;
```

We will be doing some linear algebra, so we load the linear algebra library, `linalg`. This library defines some of the commands and data structures that we will use. When we load a library it announces all of the commands that it defines. We can suppress this output by using a colon, in place of the usual command-terminating semicolon. Here I used a semicolon so you can see the list of commands defined by the `linalg` library.

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

```
Warning, the protected names norm and trace have been redefined and unprotected
```

```
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol,
  addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat,
  charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto,
  crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues,
  eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci,
  forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite,
  hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith, issimilar,
  iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix, minor, minpoly,
  mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent, pivot, potential,
  randmatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan, rref, scalarmul,
  singularvals, smith, stackmatrix, submatrix, subvector, subbasis, swapcol, swaprow, sylvester,
  toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector, wronskian]
```

Matrices can be entered by listing their rows (lists are denoted by `[]` in Maple) or by giving the size and then listing the entries in the matrix row-wise. Here I enter the matrix `A` in both ways.

```
> A:=matrix([[1,2,3,4],[0,-1,3,4],[2,-1,2,3]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -1 & 3 & 4 \\ 2 & -1 & 2 & 3 \end{bmatrix}$$

```
> A:=matrix(3,4,[1,2,3,4,0,-1,3,4,2,-1,2,3]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -1 & 3 & 4 \\ 2 & -1 & 2 & 3 \end{bmatrix}$$

It is also possible to build a matrix by specifying its columns (as vectors or lists). Thus

```
> a[1]:=[1,0,2]; a[2]:=[2,-1,-1]; a[3]:=[3,3,2]; a[4]:=[4,4,3];
```

$$a_1 := [1, 0, 2]$$

$$a_2 := [2, -1, -1]$$

$$a_3 := [3, 3, 2]$$

$$a_4 := [4, 4, 3]$$

Now we use the augment command to build the matrix:

```
> A:=augment(a[1],a[2],a[3],a[4]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -1 & 3 & 4 \\ 2 & -1 & 2 & 3 \end{bmatrix}$$

Alternately we can build the matrix by defining its rows and then stacking them on top of each other with the stackmatrix command.

```
> b[1]:=[1,2,3,4]; b[2]:=[0,-1,3,4]; b[3]:=[2,-1,2,3];
```

$$b_1 := [1, 2, 3, 4]$$

$$b_2 := [0, -1, 3, 4]$$

$$b_3 := [2, -1, 2, 3]$$

```
> A:=stackmatrix(b[1],b[2],b[3]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -1 & 3 & 4 \\ 2 & -1 & 2 & 3 \end{bmatrix}$$

There is also a command called entermatrix which may be used to build a matrix. It is a special purpose command though and not very convenient unless you enjoy typing numerous semicolons.

Diagonal matrices can be entered in a natural way.

```
> diag(1,2,3,4);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

The diag command gives you a simple way to define your own n by n identity matrix, IE(n):

```
> IE:=n->diag(seq(1,k=1..n));
```

$$IE := n \rightarrow \text{diag}(\text{seq}(1, k = 1 .. n))$$

```
> IE(2); IE(3);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

It would have been more natural to use I for the identity matrix, but Maple uses that for the square root of minus one.

Maple has a built-in command, `rref`, for computing the row reduced echelon form of a matrix. For example, for the matrix A above we have

`> rref(A);`

$$\begin{bmatrix} 1 & 0 & 0 & \frac{3}{19} \\ 0 & 1 & 0 & \frac{-1}{19} \\ 0 & 0 & 1 & \frac{25}{19} \end{bmatrix}$$

Note we still have A available. Ordinarily in Maple an expression may be evaluated (or the value recalled) by entering the name followed by a semicolon.

`> A;`

A

As we see, Maple makes an exception for matrices. The reason is they can be quite large and you may not wish to waste time displaying them, nor to clutter up your worksheet. You can always use the `evalm` function to force matrix evaluation and display.

`> evalm(A);`

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -1 & 3 & 4 \\ 2 & -1 & 2 & 3 \end{bmatrix}$$

Now suppose we want to row reduce A ourselves. We'd want to add -2 times the first row to the third row:

`> A1:=addrow(A,1,3,-2);`

$$A1 := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & -1 & 3 & 4 \\ 0 & -5 & -4 & -5 \end{bmatrix}$$

Note I assigned the result to A1 so I would be able to refer to it in the next step. An alternative approach is to use the ditto operator, %, but it is risky to use except within the same line, because it refers to the previous expression evaluated, which may not be the previous expression on your worksheet if you have done a lot of editing. Here's an example of using the % operator. I stack up two commands on one line, so I will not have to worry what % points to.

> **addrow(A1,2,1,2); A2:=addrow(%,2,3,-5);**

$$\begin{bmatrix} 1 & 0 & 9 & 12 \\ 0 & -1 & 3 & 4 \\ 0 & -5 & -4 & -5 \end{bmatrix}$$

$$A2 := \begin{bmatrix} 1 & 0 & 9 & 12 \\ 0 & -1 & 3 & 4 \\ 0 & 0 & -19 & -25 \end{bmatrix}$$

> **mulrow(A2,2,-1); A3:=mulrow(%,3,-1/19);**

$$\begin{bmatrix} 1 & 0 & 9 & 12 \\ 0 & 1 & -3 & -4 \\ 0 & 0 & -19 & -25 \end{bmatrix}$$

$$A3 := \begin{bmatrix} 1 & 0 & 9 & 12 \\ 0 & 1 & -3 & -4 \\ 0 & 0 & 1 & \frac{25}{19} \end{bmatrix}$$

> **addrow(A3,3,1,-9); A4:=addrow(%,3,2,3);**

$$\begin{bmatrix} 1 & 0 & 0 & \frac{3}{19} \\ 0 & 1 & -3 & -4 \\ 0 & 0 & 1 & \frac{25}{19} \end{bmatrix}$$

$$A4 := \begin{bmatrix} 1 & 0 & 0 & \frac{3}{19} \\ 0 & 1 & 0 & \frac{-1}{19} \\ 0 & 0 & 1 & \frac{25}{19} \end{bmatrix}$$

We did not have to swap any rows here, but the command for that is swaprow. Here is an example using again our matrix A.

```
> swaprow(A, 1, 3);
```

$$\begin{bmatrix} 2 & -1 & 2 & 3 \\ 0 & -1 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Consider now a system of the form $Ax=b$, using again our matrix A .

```
> b := [2, 7, -5];
```

$$b := [2, 7, -5]$$

```
> soln := linsolve(A, b);
```

$$\text{soln} := \left[-t_1, -\frac{5}{3} - \frac{1}{3}t_1, \frac{25}{3}t_1 + \frac{128}{3}, -\frac{19}{3}t_1 - \frac{92}{3} \right]$$

Maple returns a list of values. The underscore t sub 1 is a parameter. You assign it arbitrary values to obtain all the solutions. If you find the expression difficult to read you can always substitute something more agreeable, for example

```
> soln := linsolve(A, b, rnk, s);
```

$$\text{soln} := \left[s_1, -\frac{5}{3} - \frac{1}{3}s_1, \frac{25}{3}s_1 + \frac{128}{3}, -\frac{19}{3}s_1 - \frac{92}{3} \right]$$

Note the parameter name has to appear as the fourth variable, so we had to insert a third variable, rnk , in the function call. It turns out that `linsolve` stuffs the rank of A into the third variable. Thus

```
> rnk;
```

3

One can pick out the individual components easily

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

$$\begin{aligned} & s_1 \\ & -\frac{5}{3} - \frac{1}{3}s_1 \\ & \frac{25}{3}s_1 + \frac{128}{3} \\ & -\frac{19}{3}s_1 - \frac{92}{3} \end{aligned}$$

You can also use the command `op()` to extract operands:

```
> op(soln)[1]; op(soln)[2]; op(soln)[3]; op(soln)[4];
```

$$\begin{aligned} & s_1 \\ & -\frac{5}{3} - \frac{1}{3}s_1 \\ & \frac{25}{3}s_1 + \frac{128}{3} \\ & -\frac{19}{3}s_1 - \frac{92}{3} \end{aligned}$$

We can also solve directly by row reduction of course -

```
> M:=augment(A,b); R:=rref(M);
```

$$M := \begin{bmatrix} 1 & 2 & 3 & 4 & 2 \\ 0 & -1 & 3 & 4 & 7 \\ 2 & -1 & 2 & 3 & -5 \end{bmatrix}$$
$$R := \begin{bmatrix} 1 & 0 & 0 & \frac{3}{19} & \frac{-92}{19} \\ 0 & 1 & 0 & \frac{-1}{19} & \frac{-1}{19} \\ 0 & 0 & 1 & \frac{25}{19} & \frac{44}{19} \end{bmatrix}$$

Here we can read the solution easily. Moreover, we get the canonical form of the solution, with the free variable as the parameter. This is not the form that Maple returned above (though it is equivalent, of course).

```
> for k from 1 to 3 do x[k]:=R[k,5]-R[k,4]*s; od; x[4]:=s;
```

$$\begin{aligned} x_1 & := -\frac{92}{19} - \frac{3}{19}s \\ x_2 & := -\frac{1}{19} + \frac{1}{19}s \\ x_3 & := \frac{44}{19} - \frac{25}{19}s \\ x_4 & := s \end{aligned}$$

For more complicated situations this could get out of hand and `linsolve` may be preferable, even if it returns a non-canonical solution.

Note we do not have to use `linsolve` to find the rank of a matrix. Maple has a command called `rank`. Let's generate a random matrix to illustrate

```
> B:=randmatrix(5,7);
```

$$B := \begin{bmatrix} -85 & -55 & -37 & -35 & 97 & 50 & 79 \\ 56 & 49 & 63 & 57 & -59 & 45 & -8 \\ -93 & 92 & 43 & -62 & 77 & 66 & 54 \\ -5 & 99 & -61 & -50 & -12 & -18 & 31 \\ -26 & -62 & 1 & -47 & -91 & -47 & -61 \end{bmatrix}$$

```
> rank(B);
```

5

Actually, it turns out that for a 5 by 7 random matrix the probability of rank 5 is 1. In other words, it is nearly certain!

Consider now the vectors

```
> u:=[2,1,-3,1]; v:=[-1,3,5,0]; w:=[2,-1,1,-3]; b:=[-16,17,37,3];
```

$$u := [2, 1, -3, 1]$$

$$v := [-1, 3, 5, 0]$$

$$w := [2, -1, 1, -3]$$

$$b := [-16, 17, 37, 3]$$

We wish to write b as a linear combination of u , v , w if possible. First we form the matrix with columns u , v and w . Then we solve the system of linear equations with this matrix as the coefficient matrix and with b as the inhomogeneous term.

```
> B:=augment(u,v,w); linsolve(B,b);
```

$$B := \begin{bmatrix} 2 & -1 & 2 \\ 1 & 3 & -1 \\ -3 & 5 & 1 \\ 1 & 0 & -3 \end{bmatrix}$$
$$[-3, 6, -2]$$

We see $b = -3u + 6v - 2w$. What happens if there is no solution? Here's an example:

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

$$C := \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$b := [1, 2]$$

```
> linsolve(C,b);
```

```
> rref(augment(C,b));
```

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

There is no solution and therefore Maple returns no response! It's logical, but may take getting used to.

We have discussed matrix multiplication a little bit in class already. In Maple the notation for matrix multiplication is `&*`. It looks strange, but the ampersand warns Maple's symbolic simplification routines that this multiplication is non-commutative.

Here's a few examples:

```
> A:=randmatrix(3,3); B:=randmatrix(3,3);
```

$$A := \begin{bmatrix} 41 & -58 & -90 \\ 53 & -1 & 94 \\ 83 & -86 & 23 \end{bmatrix}$$

$$B := \begin{bmatrix} -84 & 19 & -50 \\ 88 & -53 & 85 \\ 49 & 78 & 17 \end{bmatrix}$$

```
> evalm(A &* B); evalm(B &* A);
```

$$\begin{bmatrix} -12958 & -3167 & -8510 \\ 66 & 8392 & -1137 \\ -13413 & 7929 & -11069 \end{bmatrix}$$
$$\begin{bmatrix} -6587 & 9153 & 8196 \\ 7854 & -12361 & -10947 \\ 7554 & -4382 & 3313 \end{bmatrix}$$

Notice I had to force Maple to evaluate these expressions. The default behavior is to return the product unevaluated. This is very useful behavior in some cases where intermediate results are inconveniently large or not actually needed.

Consider now three vectors:

```
> c1:=vector([1,2,3]); c2:=vector([0,1,4]); c3:=vector([1,3,7]);
```

$$c1 := [1, 2, 3]$$

$$c2 := [0, 1, 4]$$

$$c3 := [1, 3, 7]$$

Are they linearly independent? We row reduce the matrix with columns `c1`, `c2` and `c3`.

```
> C:=augment(c1,c2,c3); rref(C);
```

$$C := \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 3 & 4 & 7 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Since C has only 2 pivotal columns the vectors c1, c2 and c3 are linearly dependent. Suppose we want to solve the system $Cx=b$. We know that there will be a compatibility condition on b. Let's try it with b symbolic.

```
> b:=vector([b1,b2,b3]);
```

$$b := [b1, b2, b3]$$

```
> linsolve(C,b);
```

No response! Wise, but not useful. Let's try another approach.

```
> rref(augment(C,b));
```

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

That's strange. This result implies no solutions at all, yet there certainly is a solution if $b = 0$ for example. The problem is that Maple simplifies expressions such as z/z to 1 when z is symbolic. Thus Maple's answer is actually correct in the sense that there is no solution which is a rational expression in b_1 , b_2 and b_3 .

One way out of the impasse is to note that we can keep track of the coefficients of b_1 , b_2 and b_3 during row reduction by keeping them in separate columns. Thus we augment C by an identity matrix and row reduce.

```
> augment(C,IE(3)); R:=rref(%);
```

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 2 & 1 & 3 & 0 & 1 & 0 \\ 3 & 4 & 7 & 0 & 0 & 1 \end{bmatrix}$$

$$R := \begin{bmatrix} 1 & 0 & 1 & 0 & \frac{4}{5} & \frac{-1}{5} \\ 0 & 1 & 1 & 0 & \frac{-3}{5} & \frac{2}{5} \\ 0 & 0 & 0 & 1 & \frac{-4}{5} & \frac{1}{5} \end{bmatrix}$$

There is no pivot in the first 3 columns in the last row, so the remaining part of the last row yields a compatibility condition on b

```
> evalm(submatrix(R,3..3,4..6) &* b) = 0;
```

$$\left[b_1 - \frac{4}{5}b_2 + \frac{1}{5}b_3 \right] = 0$$

Now we can read of the solutions easily.

One comment - Maple commands are polymorphic. Their behavior depends on the number and type of parameters that you feed to them. However, even though lists and vectors are different data types in Maple, the linear algebra routines for the most part do not care which you use and return the appropriate solution.

The type() function may be used to check data-types. This is useful in your own code if you want it to behave reasonably with different inputs.

```
> a := [1,2,3]; b := vector([1,2,3]); c := matrix(1,3,[1,2,3]);
```

```
      a := [1, 2, 3]
```

```
      b := [1, 2, 3]
```

```
      c := [1  2  3]
```

```
> type(a,vector); type(a,list); type(a, array); type(a,matrix);
```

```
      false
```

```
      true
```

```
      false
```

```
      false
```

```
> type(b,vector); type(b,list); type(b,array); type(b,matrix);
```

```
      true
```

```
      false
```

```
      true
```

```
      false
```

```
> type(c,vector); type(c,list); type(c,array); type(c,matrix);
```

false

false

true

true

If you get really mysterious results try thinking hard about your data types. Sometimes a lack of clarity in thought will bite you!

Another thing to watch out for is 0. When Maple multiplies a symbolic expression by 0 the result is 0. But which zero?

```
> 0*p;
```

0

This behavior can cause problems. For example

```
> N:=matrix(2,2,[1,2,3,4]);
```

$$N := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> 0*N; evalm(3*N);
```

$$0 \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

Ouch! Multiplying by 3 works fine, but our 2 by 2 matrix became a scalar when we multiplied by 0. Some very strange error messages could result later! Here the solution is to use the scalar multiplication function

```
> scalarmul(N,0); scalarmul(N,3);
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

Of course, it is rare for people to be so careful all the time. The key is to think about your results, to understand what is going on and to add the right level of precision to guarantee your results. Even with computer algebra systems there is no substitute for understanding (or thought).

For help with the linalg package try:

```
[ > ?linalg
```

Maple (at least version 6 and higher) has another linear algebra package called LinearAlgebra. It duplicates some of the features of linalg but is better suited for numeric calculations with large matrices. For more information try

```
[ > ?LinearAlgebra
```

- Limits

```
[ > restart;
```

Maple has a limited understanding of limits

```
[ > limit((exp(x)-1-x)/x^2,x=0);
```

$$\frac{1}{2}$$

As above, we can use the inert (unevaluated) form of limit(), i.e., Limit(), to do fancy typography.

```
[ > Limit((exp(x)-1-x)/x^2,x=0) = limit((exp(x)-1-x)/x^2,x=0);
```

$$\lim_{x \rightarrow 0} \frac{e^x - 1 - x}{x^2} = \frac{1}{2}$$

Or with less typing (and less chance for errors)

```
[ > Limit((exp(x)-1-x)/x^2,x=0): % = value(%);
```

$$\lim_{x \rightarrow 0} \frac{e^x - 1 - x}{x^2} = \frac{1}{2}$$

Here are some more limits

```
[ > k:=evaln(k): Sum(k^(-4),k=1..infinity): %=value(%);
```

$$\sum_{k=1}^{\infty} \frac{1}{k^4} = \frac{1}{90} \pi^4$$

```
[ > sum(1/k^1.0002,k=1..infinity);
```

5000.577230

```
[ > Int(exp(-x^4),x=0..infinity): %=value(%);
```

$$\int_0^{\infty} e^{-x^4} dx = \frac{1}{4} \frac{\pi \sqrt{2}}{\Gamma\left(\frac{3}{4}\right)}$$

- Recursion and remember

```
[ > restart;
```

We can define sequences and functions recursively

```
[ > T := n->
> if n=1 then 3;
> elif n=2 then 1;
> else 2*T(n-2)+T(n-1);
> fi;
```

and then compute any part of the sequence

```
[ > L:=[]: for k from 11 to 15 do L:=[op(L),T(k)]: od: L;
[ 1367, 2729, 5463, 10921, 21847 ]
```

Note here `op(L)` returns the operands in `L`, then we tack on `T(k)` and form a list by enclosing everything in square brackets, i.e., we push `T(k)` on the list.

This calculation is actually very inefficient. If we write it as a procedure with the `remember` option, then Maple will remember results from previous incantations of the procedure (there are many since it calls itself) and therefore run quicker (but use more RAM).

```
[ > TT:=proc (n)
> option remember;
> if n=1 then 3;
> elif n=2 then 1;
> else 2*TT(n-2)+TT(n-1);
> fi;
> end;
```

Let's check the run-times (in seconds).

```
[ > tm:=time(): T(25); time()-tm;
[ 22369623
[ 9.766
[ > tm:=time(): TT(25); time()-tm;
[ 22369623
```

0.

Your times will differ from mine, but you will see that `TT` is hundreds of times faster than `T`. Keep the "option remember" in mind when doing recursion. If you try to compute `T(1000)` you will grow very much older, whereas `TT(1000)` is very quick.

```
> tm:=time(): TT(2000); time()-tm;
765420463516169682821888800785121322681545134725796800318428491217177507594\
 913542571106324211004179945630643092658308515631412640575370220950620904111\
 102123594199927635415200004591860988266965809526179933756574965210769757642\
 589092982113506936442359806033310387749325392964197370822077663574428146468\
 884542294459393229723370682529898719427059339565176669526684778044172203480\
 054214908541945075119222114710659969309454180145199056026947732354555010272\
 596226013946137033051890597813594400546381444203885035869503891506770188991\
 909462880363392768505225884167836236858672152285138786436358414565674326862\
 49
```

.562

If you try to compute `TT(n)` for too large an `n` Maple will return an error, "Too many levels of recursion." It is not always a good idea to define a function recursively, even when it is slick.

Set Theory

```
[ > restart;
```

Maple has a few set theory commands built in.

```
[ > U:={1,2,3,4,5,6,7,8,9,10,11,12}; # small universe
```

```
U := {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
```

```
[ > A:={3,5,7,9}; B:={2,3,5,6,7}; C:={2,4,6,8};
```

```
A := {3, 5, 7, 9}
```

```
B := {2, 3, 5, 6, 7}
```

```
C := {2, 4, 6, 8}
```

```
[ > A union B;
```

```
{2, 3, 5, 6, 7, 9}
```

```
[ > B intersect C;
```

```
{2, 6}
```

```
[ > B minus A;
```

```
{2, 6}
```

```
[ > A minus B;
```

{9}

We can define our own absolute complement function (relative to our selected universe):

```
> complement:=X->U minus X;
                                complement := X → U minus X
> complement(A);
                                {1, 2, 4, 6, 8, 10, 11, 12}
> complement(B);
                                {1, 4, 8, 9, 10, 11, 12}
> complement(C);
                                {1, 3, 5, 7, 9, 10, 11, 12}
> (A union B) minus C;
                                {3, 5, 7, 9}
> complement(A union B) intersect complement(B union C);
                                {1, 10, 11, 12}
> (A union C) minus (C minus A);
                                {3, 5, 7, 9}
```

The symmetric difference measures by how much two sets differ. We can easily define our own symmetric difference function.

```
> symdiff:=(X,Y)->(X minus Y) union (Y minus X);
                                symdiff := (X, Y) → (X minus Y) union (Y minus X)
> symdiff(A,B);
                                {2, 6, 9}
> symdiff(A,C);
                                {2, 3, 4, 5, 6, 7, 8, 9}
> symdiff(B,C);
                                {3, 4, 5, 7, 8}
```

We can easily test membership

```
> member(3,A);
                                true
> member(4,A);
                                false
```

Here's a simple way to define a procedure `carp()` to compute the Cartesian product. This is a bit more complicated than the arrow notation for functions, but is much more flexible.

```

> carp:=proc (X,Y)
>   local Z,x,y;
>   Z:={};
>   for x in X do
>     for y in Y do
>       Z:=Z union {[x,y]};
>     od;
>   od;
>   return Z;
> end:
> carp(A,A);
{[3, 5], [3, 9], [3, 7], [5, 5], [5, 7], [5, 9], [5, 3], [7, 3], [7, 5], [7, 7], [7, 9], [9, 3],
  [9, 5], [9, 7], [9, 9], [3, 3]}
> carp(A,B);
{[3, 6], [5, 2], [7, 2], [7, 6], [5, 6], [9, 2], [9, 6], [3, 5], [3, 7], [5, 5], [5, 7], [5, 3],
  [7, 3], [7, 5], [7, 7], [9, 3], [9, 5], [9, 7], [3, 2], [3, 3]}
> carp(A,C);
{[9, 4], [9, 8], [3, 4], [3, 8], [7, 8], [5, 4], [5, 8], [7, 4], [3, 6], [5, 2], [7, 2], [7, 6],
  [5, 6], [9, 2], [9, 6], [3, 2]}
> carp(A,B) minus carp(A,C);
{[3, 5], [3, 7], [5, 5], [5, 7], [5, 3], [7, 3], [7, 5], [7, 7], [9, 3], [9, 5], [9, 7], [3, 3]}

```

We can devise our own test for subsets

```

> subset:=proc (X,Y)
>   local x,s;
>   s:=true;
>   for x in X do
>     s:= s and member(x,Y);
>   od;
> end:
> subset(A,B);
false
> subset({9,5},A);
true

```

Here's a slicker way to check for a subset. Here evalb() means evaluate Boolean, that is, find the truth value of a statement.

```

> subset2:=(X,Y)->evalb((X minus Y)={});
subset2 := (X, Y) → evalb(X minus Y = { })
> subset2(A,B);

```

```
> subset2({9,5},A);  
false  
true
```

The power set may be computed by using the Maple function `choose()` from the `combinat` package. To use it we have either to load the `combinat` package by issuing the command `with(combinat)` or we have to call the function by using its so called long name `combinat[choose]()`. We will use the second method but we will rename the function to the more convenient `powset()`.

```
> alias(powset=combinat[choose]);  
powset  
> powset(A);  
{ {5,9}, { }, {9}, {3,5,7}, {7,9}, {3,7,9}, {3,9}, {3,5,9}, {5,7,9}, {3}, {5},  
  {3,5}, {7}, {3,7}, {5,7}, {3,5,7,9} }
```

Try to be reasonable. If you decide to compute `powset(carp(A,B))` you will have a long wait. This set has 2^{20} elements!

```
> 2^20;  
1048576
```

It will take a while to list over 1 million elements.

We have barely scratched the surface of Maple. There are many other things Maple can do. Try exploring the help facility. Above all, experiment!