An Introductory Guide to Maple

Prepared By

Mark H. Holmes
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180

Table of Contents

1. Introductory Demonstration of Maple

The following is an example that demonstrates how to solve a mathematical problem using Maple. It is strongly recommended that those starting out use Maple to work through the example.

Problem Solve the equation $2x^2 - 13x - 24 = 0$.

The following is a Maple session that solves this problem. Note that each Maple command ends with a semicolon (;) and the computer does not execute the command until the line is entered (by hitting the return key). Also, any line beginning with a # symbol indicates to Maple that this is a comment line and it is therefore ignored. Finally, the symbol > beginning each line is the Maple prompt.

```
    |\^/|     Maple V Release 3 (Rensselaer Polytechnic Institute)
._|\|   |/|_. Copyright (c) 1981-1994 by Waterloo Maple Software and the
 \  MAPLE  /  University of Waterloo. All rights reserved. Maple and Maple V
 <____ ____>  are registered trademarks of Waterloo Maple Software.
      |       Type ? for help.
```

># We first enter the expression y = 2*x^2 – 13*x – 24 . Note the way powers and
># multiplication are expressed here.

> y := 2*x^2 – 13*x – 24;

$$y := 2\,x^2 - 13\,x - 24$$

># Maple has echoed our command. Also, note that in Maple one uses the symbols := to
># indicate assignment. In this example the name y has been assigned to the polynomial
># 2*x^2 – 13*x – 24 . Hereafter we can refer to the polynomial simply as y .

># Let's experiment with y a little before solving the problem. First we can try
># evaluating y at x = 0 . This can be accomplished as follows:

> x := 0;

$$x := 0$$

> y;

$$-24$$

># We get the expected result. However, we are now stuck with x set to zero. To return
># it to its old unevaluated self, use the following command:

> x := 'x';

$$x := x$$

> y;

$$2\,x^2 - 13\,x - 24$$

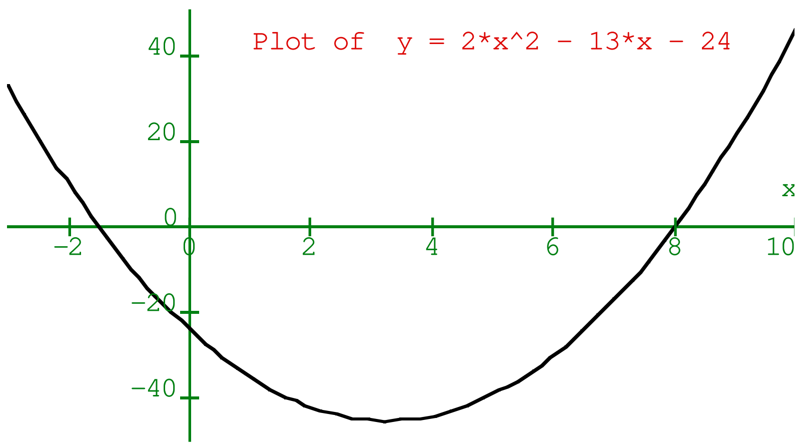
># There is another way to evaluate y which doesn't require us to unevaluate x afterwards.
># This is the substitution command (*subs*) and it is:

> subs(x = 0 , y);

$$-24$$

># Now let's plot y from x = −3 to x = 10 . This is done using the *plot* command...

> plot(y , x = −3..10 , title = `Plot of  y = 2*x^2 − 13*x − 24`);



># Perhaps the most unusual aspect of the plot command is the way the x interval is entered.
># Also, note we have used back quotes (``) around the title of the plot. If you forget the
># specifics of a command you can always use the on-line help, which is accessed using the ?
># symbol. To find out about *plot* we use the following (a semicolon isn't necessary here)...

> ?plot

># The Maple response hasn't been given here but you can find out what it is easily enough
># by simply entering this command.

># We now return to the original problem of solving the equation. In the above plot we can
># see that one root is between −2 and 0 while the other is very near 8 . To find out exactly
># what they are we use the *solve* command...

> solve(y = 0 , x);

$$8, -3/2$$

># So, Maple has given us answers we expected based on the above plot. Note that in this
># command we have specified the equation (y = 0) and the variable to solve for (x).

># We now end this Maple session using the command...
> quit

2. Maple Commands

The following pages contain a listing of the Maple commands you will need for calculus (plus a few extras). However, this listing is not intended to replace the manual but only serve as a guideline, and perhaps a quick reference, for what needs to be used in the labs. For more extensive assistance it is recommended that you use the on-line help (see the ? command described below).

As a reminder, a semicolon (;) or colon (:) must follow every command. Also, anything on a line following the symbol # is ignored by Maple (this is one of the ways comments are added to a Maple session).

Operators (?+)

addition: + subtraction: −
multiplication: * division: /
exponentiation: ^ or **
repetition: $

Example: diff(x^5 , x$3) yields $60\,x^2$

Constants (?Pi)

π: Pi ∞: infinity $\sqrt{-1}$: I

Elementary Functions (?exp)

exponential: exp(x) natural logarithm: ln(x)
absolute value: abs(x) square root: sqrt(x) or x^(1/2)
trigonometric: sin(x), cos(x), tan(x), sec(x), cot(x), csc(x)

Note the trigonometric functions in Maple require angles measured in radians.

inverse trigonometric functions: arcsin(x), arccos(x), arctan(x), arctan(y,x)
hyperbolic functions: sinh(x), cosh(x), tanh(x), sech(x), csch(x), coth(x)

Commands Useful in Math 1

> ? Provides very useful information and examples about Maple commands. The ? can be used for help for any Maple command and package.

```
?               # gives info about the help facility
?library        # gives the standard commands and functions
?index          # gives an index of help descriptions
?plot           # gives info on the plot command
?plot[polar]    # gives info on the plot option of using polar coordinates
```

> "; This gives the previously computed result. Maple remembers the previous three (i.e., you can use " and "" and """).

f := sin(2);
g := 6*"; # gives 6 sin(2)
h := "*""; # gives $6\ \sin(2)^2$

> x := 'x'; Returns x to a variable (note back quotes are **not** used here).

> f:=x−> F; Arrow notation to define f as a function of x . For help on this use ?-> .

f := x −> 3*x + 5;
f(2); # gives 11
f := 'f'; # unassigns the above definition for f
g := (x,y) −> x*y^2;

> diff(f,x$n); Finds the n<u>th</u> derivative of f with respect to x .

diff(f, x); # this gives the first derivative f'
diff(f, x$3); # this gives the third derivative f'''
diff(f,x,x,x); # this also gives the third derivative f'''
diff(x^3, x); # gives $3\ x^2$
diff(t^3, t$3); # gives 6
f := x −> 2*x^3 + 5;
diff(f(x), x); # gives $6\ x^2$

> Digits := n; Sets the number of digits used for floating point numbers to n (the default is 10). Note the capital D in this command.

> evalf(f); Evaluates the expression f using floating point arithmetic.

evalf(−1/4 + sqrt(33)/4); # gives 1.186140662
evalf(Pi, 20); # gives 3.1415926535897932385

> expand(f); Expands the expression f using the laws of algebra and trigonometry.

expand(ln((x + 2)/x^2)); # gives ln(x + 2) − 2 ln(x)
expand((s + 1)*(s + 3)); # gives $s^2 + 4\ s + 3$

> factor(f); Factors the given expression.

factor(x^2 + 5*x + 6); # gives (x + 3) (x + 2)

> fsolve(f = a, x); Solves the equation f = a for x . The answer is given in decimal form. Usually *fsolve* returns a single real root, but for some polynomial and transcendental equations it will find all real roots.

```
fsolve(x^2 − 3*x + 2, x);       # gives  1.000000000, 2.000000000
fsolve(r^3 + 4 = 45, r);        # gives  3.448217240
fsolve(f = a, x, x1..x2);       # This solves  f(x) = a  for  x1 < x < x2 .
```

> int(f, x); Finds the indefinite integral of f with respect to x . The arbitrary constant of integration is not included in the answer.

```
int(x^2, x);            # gives  1/3 x^3
```

> int(f, x = a..b); Finds the definite integral of f from a to b . When this command is followed by evalf(") the integral is evaluated numerically.

```
int(x^2, x = 0..2);             # gives  8/3
int(exp(−z^2), z = 0..1);       # gives  1/2 Pi^(1/2) erf(1)
evalf(");                       # gives  .7468241330
```

> leftbox(f, x = a..b, n); Plots rectangular boxes used to approximate the definite integral of f over $a \le x \le b$. Height of each box is determined by value of f at left end of each subinterval; n specifies the number of boxes.

```
with(student):
leftbox(sin(x),x=0..Pi/2,10);
```

> leftsum(f, x = a..b, n); Finds the approximation of definite integral of f over $a \le x \le b$ when leftboxes are used; n specifies number of boxes).

```
with(student):
leftsum(sin(x),x=0..Pi/2,10);
evalf(");                              # gives  .9194031700
```

> limit(f, x = a); Finds the limit of f as $x \to a$. On some expressions it is best to use the *expand* command first before using *limit*.

```
limit( sin(x)/x, x = 0);               # gives  1
limit( exp(b), b = infinity);          # gives  infinity
limit( −1/x, x = 0, right);            # gives  −infinity
```

> middlebox(f, x = a..b, n); Plots rectangular boxes used to approximate definite integral of f over $a \le x \le b$. Height of each box determined by value of the function in middle of each subinterval; n specifies the number of boxes.

```
with(student):
middlebox(sin(x), x = 0..Pi/2, 10);
```

> middlesum(f, x = a..b, n); Finds approximation of definite integral of f over $a \le x \le b$ using middleboxes; n specifies the number of boxes (or subintervals).

```
with(student):
middlesum(sin(x), x = 0..Pi/2, 10);
evalf(");                          # gives  1.001028825
```

> plot(f, x = x1..x2, title = `Example`); Plots the graph of f for $x1 \le x \le x2$. Note the backquotes around the plot title.

```
plot(cos(x), x = 0..Pi);                 # this plots  y = cos(x)  for  0 ≤ x ≤ π
plot(cos(x), x = −Pi..Pi, title=`y=cos(x)`, color=YELLOW);
                                         # this plots  y = cos(x)  for  −π ≤ x ≤ π
plot(f,x=2..4,0..1, title=`y=f(x)`);     # this restricts the plot to 2 ≤ x ≤ 4, 0 ≤ y ≤ 1
```

> plot({f, g}, x = x1..x2, title=`TEST`); This is used for plotting 2 (or more) graphs on the same axes.

```
plot({x^2,sin(x)}, x=0..Pi, −1..3);      # this plots y = x^2 and y = sin(x)
                                         # for  0 ≤ x ≤ π ,  −1 ≤ y ≤ 3
```

> plot([f, g, t = a..b], title=`test`); Plots parametric equations $x = f(t), y = g(t)$ for $a \le t \le b$.

```
plot([sin(t),cos(t),t= −Pi..Pi], title = `A Circle`);
plot([f, g, t = 0..1], −2..3);                  # this restricts  −2 ≤ x ≤ 3
plot([f, g, t = 0..1], −2..3, 0..5);            # now  −2 ≤ x ≤ 3, 0 ≤ y ≤ 5
plot([r, t, t = 0..1],coords=polar);            # plots the polar equation  r = r(t)
plot({[f,g,t=0..1], [F,G,t=−1..3]}, title=`A Couple of Curves`);
                                         # this plots two parametric curves
```

> plot(L, x = x1..x2, title=`TEST`); The plot command can also be used to plot point plots.

```
L1 := [0,0,1,1,2,1,2,0,1,-1,0,0];
plot(L1, x=0..2, style=point);
L2 := [[ n, sin(n)] $n=1..10];
plot(L2, x=0..15, style=line,symbol=circle);
```

> quit — Quit Maple. If this doesn't work try the combination ;;;quit .

> rightbox(f, x = a..b, n); — Plots rectangular boxes used to approximate the definite integral of f over $a \le x \le b$. Height of each box determined by the value of the function at the right side of each subinterval; n specifies the number of boxes.

```
with(student):
rightbox(sin(x), x = 0..Pi/2, 10);
```

> rightsum(f, x = a..b, n); — Finds the approximation of the definite integral of f over $a \le x \le b$ when rightboxes are used; n specifies the number of boxes.

```
with(student):
rightsum(sin(x), x = 0..Pi/2, 10);
evalf(");                              # gives  1.076482803
```

> simplify(f); — Simplifies the expression f . On some expressions it is best to first use the *expand* command before using *simplify*.

```
simplify(16^(1/2) + 6);                # gives  10
simplify( exp(a) +  ln(b*exp(c)) );    # gives  exp(a) + ln(b) + c
```

> simpson(f, x = a..b, n); — Finds approximation of the definite integral of f over $a \le x \le b$ using Simpson's rule; n specifies number of subintervals (it must be even).

```
with(student):
simpson(sin(x), x = 0..Pi/2, 10);
evalf(");                              # gives  1.000003392
```

> solve(f = a, x); — This solves $f(x) = a$ for x . This command produces exact solutions, if available, while *fsolve* produces numerical answers.

```
solve(sin(x) + y = 2, x);              # gives  − arcsin(y − 2)
solve(x^2 + 2*x*y = 1, x);             # gives  −y + (y^2 + 1)^(1/2),  −y − (y^2 + 1)^(1/2)
sol := solve(x^2 − 9 = 0, x);
sol[1];                                # gives  3
sol[2];                                # gives  −3
solve({x + y = 1, 2*x + y = 3}, {x,y});        # gives {y = −1, x = 2}
```

> subs(x = x0, f); Substitutes x0 for x in the expression f . Note that x0 can be either a numerical value or an algebraic expression. It is not necessary to use x:='x' after this command.

```
subs(x = y^3, x^2 + 9*x);                        # gives  y^6 + 9 y^3
subs(x = 0, y = −1, z = Pi, x + y + cos(z));     # gives  −1 + cos(Pi)
```

> sum(f, i = m..n); Calculates the sum of f from i = m to i = n (m,n may be negative).

```
sum(i^3, i =1..3);          # gives  36
f := 2*i + 1;
sum(f, i = −1..4);          # gives 24
```

> trapezoid(f, x = a..b, n); Finds the approximation of the definite integral of f over $a \leq x \leq b$ using trapezoidal rule; n specifies the number of subintervals.

```
with(student):
trapezoid(sin(x), x = 0..Pi/2, 10);
evalf(");                              # gives  .9979429868
```

Commands Useful in Math 2

> angle(u, w); Gives the angle between the vectors u and w .†

```
angle(vector([1,0,0]), vector([1,1,1]));    # arccos(1/3 3^(1/2) )
```

> convert(T , polynom); Converts a Taylor series T to a polynomial.

```
s := taylor(sin(x), x, 5);          # gives s := x − 1/6 x^3 + O(x^5)
p := convert(s, polynom);           # gives p := x − 1/6 x^3
```

> convert(v , list); Converts a vector v to a list that can then be used in *plot3d*.

> convert(x , degrees); Converts x from radians to degrees.

```
convert( Pi, degrees);        # gives 180 degrees
```

> crossprod(v, w); Computes the cross-product of the vectors v and w .†

```
v1 := vector([1, 2, 3]);   v2 := vector([2, 3, 4]);
crossprod(v1, v2);            # gives [ −1, 2, −1 ]
```

> display({F, G}); Command used to display multiple plots on the same axes.

```
with(plots):
C := plot3d([ cos(t), sin(t), t ], t = 0..Pi, s=0..1, grid = [35, 2]):
P := plot3d([x, y, x + Pi/2], x = −1..1,y = 0..2):
S := plot3d( sin(x + y), x = 0..2, y = −1 .. 1):
display( {C, P, S}, title = `A Curve and a Couple of Surfaces` );
```

> dotprod(v, w); Calculates the dot product of the vectors v and w .†

```
v:=vector([ 1, x, y ]);   w:=vector( [1, 0, 2] );
dotprod(v, w);                            # gives 1 + 2 y
dotprod(vector([1,2]),vector([a,b]));     # gives a + 2 b
```

> evalm(a*v + b*w); Calculates av + bw where a,b are scalars and v,w are vectors.†

```
v:=vector([ 1, x, y ]);  w:=vector( [1, 0, 2] );  u:=vector( [−1, 0, q] );
evalm(2*v − 3*w);             # gives [ −1, 2 x, 2 y − 6 ]
evalm(v + w − u/2);           # gives [ 5/2, x, y + 2 − 1/2 q ]
```

---

† Command *with(linalg):* must appear somewhere in the Maple session before this command is used.

> grad(f, [x, y, z]);   Finds the gradient of f .†

```
grad( x*y*z, [x,y,z] );          # gives [ y z, x z, x y ]
```

> map(diff, v, x);   Differentiates the vector v with respect to x .†

```
map(diff, vector( [3*x, cos(x^2) ]), x);     # gives [3, −2 x sin(x^2) ]
```

> mtaylor(f, [x, y], n);   Computes the multivariate Taylor series of f to order $n - 1$, in the variables x and y .

```
readlib(mtaylor):
mtaylor(sin(x + y), [x, y], 2);      # gives x + y
```

> norm(v, 2);   Calculates the length of the vector v .†

```
v := vector( [1, −2, 3] );
norm( v, 2 );             # gives 14^(1/2)
```

> plot3d(f, x = a..b, y = c..d);   Plots $z = f(x, y)$ for $a \le x \le b$ and $c \le y \le d$. You can modify and print the plot using menus on the plot window.

```
plot3d( sin(x  + y ), x = −1..1, y =−1..1 );
plot3d( [ r*sin(s), cos(s), r + s ], r = 0..1, s = −Pi..Pi );
        # The above is used when surface is described parametrically.
plot3d( cos(x + y), x = −12..12, y = −12..12, grid = [35, 35],
        orientation = [85, 30],  axis = BOXED, title = `Example`);
plot3d( [t*s, exp(s), s], t = −2..1, s=0..3, view = [−1..1, 0..3, −1..2] );
```

> taylor(f, x = a, n);   Computes the Taylor series, up to degree $n - 1$, of f with respect to x about the point $x = a$. The defaults are $a = 0$ and $n = 6$.

```
taylor( sin(x) , x );         # gives  x − 1/6 x^3 + 1/120 x^5 + O(x^6)
taylor( exp(x) , x , 4 );     # gives  1 + x + 1/2 x^2 + 1/6 x^3 + O(x^4)
taylor( ln(x) , x = 1 , 3 );  # gives x − 1 − 1/2 (x − 1)^2 + O((x − 1)^3)
```

> vector([x1, ..., xn]);   Defines vector with n elements.†

```
v := vector( [5, 4, 6, 3] );
v[2];                   # gives  4
```

† Command *with(linalg):* must appear somewhere in the Maple session before this command is used.

3. Editing Commands in a Maple Session

Because Maple is used interactively, the ability to edit commands and then reissue them is a skill worth acquiring. To explain, suppose you issue the command

```
> plot(x*sin(x), x = 0..2*Pi);
```

After looking at the plot you decide the interval $0 \le x \le 2\pi$ is too large and it would be better to use $\pi/2 \le x \le 3\pi/2$. Well, you can either 1) retype a completely new plot command, 2) copy the old command, paste it onto the new line, and then edit it, or 3) move the cursor up to the old command and edit it there. In the case of the latter, if you go back more than one command, you need to be careful about hidden variables (e.g., see Section 6). It is recommended that you experiment using (2) and (3) because you will find them very useful when using Maple.

4. Idiosyncrasies and Features of Maple

As for any computer program, there are certain aspects of Maple that take some getting used to. For example, when should you use forward quotes and when should you use backward quotes? There are also questions, such as "do I want to use *fsolve* here or would I really be better off with *solve*?". Well, some of these issues are discussed below, but again, you should also consider using Maple's on-line help.

fsolve vs *solve*
The *solve* command will solve equations with unevaluated constants (e.g., $x^2 = 2k$) and it will, in some cases, find all of the solutions (real and complex). This is good but this also limits what it can solve (actually, in some ways, it is limited by what equations mathematicians have been able to find formulas for). On the other hand, *fsolve* will solve many different types of equations but in this case there can be no arbitrary constants in the equation (e.g., it will solve $x^2 = 2$ but not $x^2 = 2k$ unless k has been specified). There is also an important difference between these commands even for equations they both will solve. For example, *solve* will give the exact solutions to an equation such as $x^2 = 2$ (i.e., it will find the two solutions $\pm 2^{1/2}$) whereas *fsolve* will give decimal solutions (e.g., ±1.414213562). The latter are not exact but are very accurate approximations. The fact that they are not exact, however, brings us to the next topic.

evalf, *fsolve* and *Digits*
There are certain things we are forced to worry about even if we don't want to, and one example of this is roundoff error. Anytime you tell a computer to try to evaluate something numerically (say, $\sqrt{2}$), it can only use a finite number of digits. The difference between the exact result and the numerically evaluated quantity is known as roundoff error. If this seems to be causing a problem you can increase the value of *Digits*, although you should be aware that this can increase

the computing time. You may now be wondering how you will be able to tell if you should be worrying about roundoff error. Well, there is no way to answer this that will cover all situations but if you are suspicious of the accuracy of the answer just increase the value of Digits to see if the answer is affected. For example, suppose you get an answer like $1 + 0.2\mathrm{x}10^{-10}\,\mathrm{I}$. If you think the answer should be real valued then you would be suspicious of this result. You might also be suspicious because Maple normally keeps 10 digits and this answer says the solution is complex but it is the eleventh digit where the I appears. In either case it would be worth increasing Digits to see what happens to the result. One other thing, if you believe roundoff may be a problem try entering the numbers as fractions rather than in decimal form (e.g., use 13/10 instead of 1.3). This is because in certain situations fractions are treated differently than numbers in decimal form.

:= vs –>

In Maple you can enter $f(x) = x^2$ as f:=x^2 or as f:=x–>x^2 . The first is called an expression and the second a "function". Note that if you use the expression form then you refer to the formula using just f whereas for the "function" form you must refer to it as f(x) . To illustrate the differences we have the following:

Definition:	f := x^3 + 1;	f := x –> x^3 + 1;
Evaluate at x = 2:	subs(x = 2, f);	f(2);
Differentiate:	diff(f , x);	diff(f(x) , x);
Integrate:	int(f , x = 0..1);	int(f(x) , x = 0..1);

; vs :

If you want Maple to display the response then use a semicolon (;) after the command. If you don't want to see the response (e.g., maybe it is so long that it is unintelligible to a human) then use a colon (:) after the command.

Back Quotes vs Quotes

Back quotes (` ) are used to identify text to go into a command, such as the title for a plot (e.g., title = `Plot`). Quotes (') are used, among other things, to return a variable to unevaluated status and to undefine a "function" (e.g., x := 'x').

sqrt(•)

One source of potential trouble is the square root function. To illustrate, if you enter the expression sqrt(f^2) then Maple will return f . This is OK but what if f is negative and you expect sqrt(f^2) to be positive? For example, Maple states that sqrt(sin(x)^2) = sin(x) . This is positive if sin(x) is positive and negative if sin(x) is negative. A simple way to make sure the result is nonnegative is to use something like abs(sqrt(f^2)) . However, it is recommended that you do **not** do this unless absolutely necessary. The reason is that integration formulas and solutions for equations are limited if the abs(•) function is present.

I, Pi, O, etc

Like any good computing system, Maple has a certain collection of well used mathematical constants and functions available. These have special symbols and you cannot use them for other tasks. So, for example, Maple reserves I to represent $\sqrt{-1}$ and Pi for π . You can get a listing of these special symbols by using the help commands ?I or ?Pi .

5. A Common Error

It is inevitable that you will occasionally have trouble getting your ideas across to Maple and this can quickly lead to a very frustrating situation. It is impossible to anticipate every possibility but there is one that is quite common. It occurs when a variable has already been assigned and you try to use it again in another context. To illustrate this, consider the following short Maple session...

> f := x^3 + 1;

$$f := x^3 + 1$$

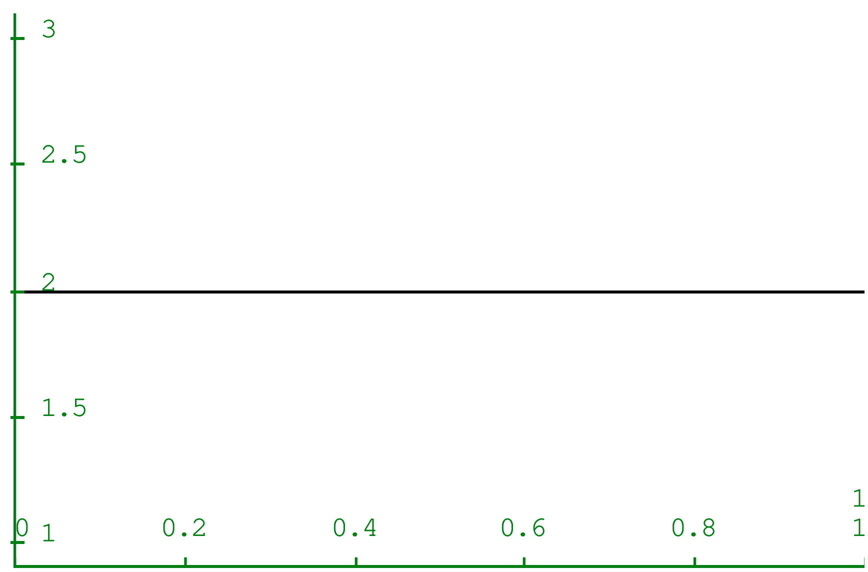
> x := 1;

$$x := 1$$

> f;

$$2$$

> plot(f, x = 0..1);



> g := x^2 – 1;

$$g := 0$$

> quit

The assignment x := 1 appearing in this session has a profound effect on the last two commands. In the plot, we get the constant (f=2) and not the original cubic. Also, in the last command, g is not a quadratic but simply a constant. If this is what was desired then fine. However, if you expected to see a cubic in the plot and a quadratic for g then x has to be unassigned its earlier value. This is accomplished by issuing the command (before the plot command):

```
> x := 'x';
```

If you are not sure whether or not you have assigned a value to a variable you can always check. For example, for the variable x you issue the command:

```
> x;
```

If the reply is just x:=x then it is unassigned. This simple test will work unless x has been defined earlier to be a vector or an array. In this case you can get Maple to tell you about x by using the command:

```
> print(x);
```

This also works to find out if x has been given a numerical value (such as 6).