

Introduction into Maple 6 *

1	Introduction	2
2	Maple Basics	2
2.1	Worksheet	2
2.2	Statements	3
2.3	Comments	3
2.4	Simple Arithmetic	4
2.5	Variables	4
2.6	Ditto Operator	6
2.7	Functions	7
2.8	Basic Types and Conversions	9
2.9	Packages	11
2.10	Saving, Restoring, Ending and Restarting and Sessions	12
3	Manipulation of Expressions	13
4	Summation	15
5	Differentiation	16
6	Integration	17
7	Taylor Series	18
8	Limits	19
9	Solving Equations	20
10	Assignment and Substitution of Solutions	24
11	Graphics	25

* This Manual draws on *Maple on Athena (AC-72)*, copyright (c) 1995 by Massachusetts Institute of Technology (MIT), permission granted.

1 Introduction

Maple 6 is a Symbolic Computation System. It can manipulate mathematical expressions in an algebraic manner, i.e. it does not need numerical values for variables.

This manual introduces the basic commands of Maple 6. It shows how to use Maple interactively. Maple also allows the user to write programs. However, this is not treated in this guide.

2 Maple Basics

2.1 Worksheet

The worksheet is an integrated environment in which you interactively solve problems and document your work. Worksheets contain not only text and conventional document-processing information, but live mathematical commands and their automatically generated results. Maple starts with a new worksheet.

2.2 Statements

To let you know that it is waiting for input, Maple prints a prompt:

```
>
```

The key format for your interaction with Maple is the *statement*. You type in statements for Maple and press **Return**; it evaluates them and gives you the results.

```
> statement;
```

```
response-to-statement
```

Your statement must be terminated by a semicolon (;) or a colon (:) for Maple to realize that you have completed it. The semicolon tells Maple to print out the result of evaluating the statement, while the colon tells it not to print the result. For example

```
> 4+5;
```

```
9
```

2.3 Comments

Any line or portion of a line beginning with a # is treated as a comment. Comments are ignored in processing. For example:

```
> 4+5;#This is an example of a comment
```

```
9
```

Since a comment is not a statement, it does not need to be terminated. It only causes the remaining input on the line to be ignored.

2.4 Simple Arithmetic

You can use Maple to evaluate arithmetic expressions. Useful operators are:

Expression	Operator
+	add
-	subtract
*	multiply
/	divide
^ or **	raise to the power
!	factorial

Arithmetic expressions are evaluated according to typical precedence rules:

```
> 4+3*2^3; # this means 4+(3*(2^3))
```

```
28
```

2.5 Variables

Maple Variables are names that generally begin with a letter and are followed by up to 498 alphanumeric or underscore characters; case is significant (eg: **x** is different from **X**). Maple distinguishes between two kinds of variables. A Maple *programming variable* or *assigned variable* is a variable that you have assigned a result to, generally with an assignment statement:

```
> x := 2 + 5;
```

```
x := 7
```

```
> x;
```

```
7
```

The programming variable **x** now is a label for the result of 2+5.

The other kind of Maple 6 Variable is a *mathematical variable* or an *unassigned variable*. These exist in the sense of algebraic unknowns, as in the case of:

```
> z := 2+y;
z := 2 + y
> y;
y
> z;
2 + y
```

Here, y is a mathematical variable, and z is a programming variable because an expression has been assigned to it. If we now assign a value to y, y becomes a programming variable, and is no longer an algebraic unknown:

```
> y := 5;
y := 5
> y;
5
> z;
7
```

Unassigned variables have their own name as their value. To turn an assigned variable (i.e., a programming variable) back into an unassigned variable (i.e., a mathematical variable), you must assign it its own name. As an example:

```
> r := 2;
r := 2
```

```
> r;
2
> r := 'r';
r := r
> r;
r
```

You can also use the **unassign** command:

```
> unassign('r');
```

2.6 Ditto Operator

In Maple, percentage signs are used to refer to previously computed expressions. Specifically, the following operators are defined:

```
%      last expression
%%     second last expression
%%%    third last expression
```

For example:

```
> z1 := z2 + z3;
z1 := z2 + z3;
> a1 := a2 * a3;
a1 := a2 a3;
> % + %%;
a2 a3 + z2 + z3
```

2.7 Functions

Maple has a large library of functions for you to use. Syntactically, functions are a type of expression. They have a name, a sequence of zero or more arguments, and they return a value as a result of calling the function. For example:

```
> abs(-2);
                2

> b := abs(a);
                b := |a|

> b;
                |a|

> a := -2;
                a := -2

> b;
                2

> a;
                -2
```

The result of taking the absolute value of "a" is returned as the result of the function **abs(...)**. The variable "a" itself is not modified as a side effect. If you wanted to change the value of the variable, you would use:

```
> a := abs(a);
                a := 2
```

Some important functions are:

Expression	Operator
$\log(x)$ or $\ln(x)$	natural logarithm
\sqrt{x}	square root
$\text{abs}(x)$	absolute value
$\exp(x)$	exponential
$\sin(x)$	sine

To get the help on a function you can use one of the help commands:

```
> help(abs);
> ?abs;
```

2.8 Basic Types and Conversions

Maple has a large number of object types. Typical types of numeric values are integer, float (floating-point or pseudo-real), fraction, rational, and boolean. Some of the more complex object types include string, polynom (polynomial expression), series, matrix, vector, set, list, and procedure.

Integers are expressed as a string of one or more digits with an optional sign, like "-2" or "3870".

```
> -2;
-2
```

Rational numbers are a signed ratio of unsigned integers and will be simplified by Maple:

```
> 2/3;
2/3
> 4/6;
2/3
```

Floating point numbers contain an explicit decimal point, and any integer or rational expression that contains such a decimal point will be evaluated as floating point:

```
> 2.0/3.0;
.666666667
```

Maple has a global variable named "Digits" that you use to control the accuracy of floating-point operations. The initial setting for Digits is 10 but you can change it by assigning it some other value:

```
> Digits:=5;
Digits := 5
> 2.0/3.0;
.66667
```

Notice that the value of Digits does not control the display of floating point values, it controls their calculation. If you calculate something with one value for Digits and afterwards change Digits to a new value, only future floating-point operations will reflect the change.

To convert a rational expression to floating point, you can use the **evalf** function:

```
> evalf(2/3);
.666666667
```

evalf also provides a way to temporarily control the accuracy of floating-point calculations.

```
> evalf(2/3, 5);
.66667
```

A more general way to convert between types is to use the **convert** function.

2.9 Packages

A Maple "*package*" is a collection of related functions. An example of a package is **linalg**, the Maple linear algebra package. To use a package, you initially read it into memory using the **with** command:

```
> with(linalg):
Warning, the protected names norm and trace have been
redefined and unprotected
```

When Maple loads a package, it checks to see whether any of the new function names will replace an existing function. If so, it gives you a warning. For instance, since **linalg** has a function to compute the trace of a matrix that is called "**trace**", which is the same name as the debugging command "**trace**", a warning was generated. The new function is available for use, but the previous function of the same name is no longer accessible.

Among the available packages are:

Package	Description
DEtools	differential equations tools
Matlab	Matlab Link
Algcurves	Algebraic Curves
finance	financial mathematics
linalg	Linear algebra
plots	graphics package
simplex	linear optimization
stats	statistics
student	student calculus

2.10 Saving, Restoring, Ending and Restarting and Sessions

You can save the active worksheet using the **File/Save** or **File/Save...** choices from the menu. **File/Open...** will open a previously save worksheet. However, it will not restore the assigned names. The Maple Kernel will remain unchanged until you recalculate the commands. You can recalculate the entire Worksheet using the menu option **Edit/Execute/Worksheet**.

The **save** statement will write all assigned names in the current Maple session into the file specified by filename

```
> save "filename.m";
```

When you specify a filename with the extension ".m", Maple saves your work in "Maple internal format". This is the most useful format for Maple when you restore your work later on. You restore the information with **read**:

```
> read "filename.m";
```

Once you have saved your work, you can terminate your Maple session with any of the following commands (without semicolon):

```
> quit
> done
> stop
```

The **restart** command will cause the Maple kernel to clear its internal memory so that it acts as if you had just started Maple:

```
> restart
```

3 Manipulation of Expressions

For the most part, Maple leaves expressions the way you entered them or the way they were created from some computation. However, there are some obvious simplifications that it automatically performs. An example of this would be:

```
> x^2 + y*x - x*y - y^2;
```

$$x^2 - y^2$$

There are two reasons why only a limited number of automatic simplifications are performed:

Cost: some simplifications, such as the factoring of expressions, can be reasonably time-consuming to perform. The Maple solution to this is to automatically perform only certain low-cost simplifications, but to provide ways of forcing more involved methods.

Preference: different users will consider different styles of expression as being simpler. The Maple solution is to provide ways to specify specific simplification techniques.

The **simplify** command provides typical expression simplifications:

```
> f := sin(x)^2 + ln(2*y) + cos(x)^2;
```

$$f := \sin(x)^2 + \ln(2y) + \cos(x)^2$$

```
> simplify(f);
```

$$1 + \ln(2) + \ln(y)$$

By using **simplify(expr, variety)** you can control what simplifications are performed:

```
> simplify( f, trig );
```

$$1 + \ln(2y)$$

The **expand** command forces distribution of multiplication over addition. For instance:

```
> f := (x + y)*(x - y);
```

$$f := (x + y)(x - y)$$

```
> expand( f );
```

$$x^2 - y^2$$

Since **expand** turns **f** into $x^2 + y * x - x * y - y^2$, the automatic simplifications turn the expression into $x^2 - y^2$.

The **factor** command factors polynomial expressions:

```
> factor( x^2-x-6 );
```

$$(x + 2)(x - 3)$$

Factor cancels common terms of both the numerator and the denominator:

```
> f := (x^2 - y^2)/(x-y)^2;
```

$$f := \frac{x^2 - y^2}{(x - y)^2}$$

```
> factor(f);
```

$$\frac{x + y}{x - y}$$

The **convert** command converts expressions between different forms:

```
> convert( 0.375*x, rational );
```

$$\frac{3}{8}x$$

```
> convert( Pi/2, degrees );
```

90 degrees

The **combine** command combines terms in sums, products and powers into a single term:

```
> combine( exp(x)^2*exp(y) );
```

$$e^{(2x+y)}$$

```
> combine( (x^a)^2 );
```

$$x^{(2a)}$$

4 Summation

Maple allows you to specify both definite and indefinite symbolic summations using the **sum** command. Be sure that the index variable is a mathematical (unassigned) variable. This is an example of a definite sum:

```
> sum( x^i, i = 0..3 );
```

$$1+x+x^2+x^3$$

This is an example of an indefinite sum:

```
> sum( x^i, i=0..infinity );
```

$$-\frac{1}{x-1}$$

5 Differentiation

Differentiation is performed using the **diff** command.

Take the derivative of an univariate expression w.r.t. x:

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

$$6x$$

Take the partial derivative of a multivariate expression w.r.t. x:

```
> diff( y*x^2 + y, x );
```

$$2xy$$

Take partial derivatives by specifying a sequence of variables. Derivatives are taken in the same order as the variable sequence:

```
> diff( y*x^2 + y, x, y );
```

$$2x$$

The capitalized function name **Diff** is the inert diff function, which simply returns unevaluated:

```
> Diff( y * x^2 + y, x, y );
```

$$\frac{\partial^2}{\partial y \partial x} (y x^2 + y)$$

6 Integration

Definite and indefinite integration are performed using the **int** command.

An example of indefinite integration:

```
> int( x^2, x );
```

$$\frac{1}{3}x^3$$

Example of definite integrations:

```
> int( x^2, x=0..2 );
```

$$\frac{8}{3}$$

```
> int( 1/x^2, x=1..infinity );
```

$$1$$

The capitalized function name **Int** is the inert version of the int function, which simply returns unevaluated:

```
> Int( x^2, x=0..2 );
```

$$\int_0^2 x^2 dx$$

When Maple is unable to find the solution to a problem, it returns unevaluated. Sometimes this will be because Maple doesn't know how to solve the problem, but more often it will be because either no solution exists or the original command/query was not posed properly.

An Example:

```
> int( exp(-x^2)*ln(x), x );
```

$$\int e^{-x^2} \ln(x) dx$$

7 Taylor Series

Maple knows how to generate any finite number of terms of a Taylor series. The Maple function is **taylor**. You specify the number of terms by specifying the "order of truncation". If the terms up to the $x^{(n-1)}$ do not make the complete expression, an order term of $O(x^n)$ is added to the end. You can specify the point of expansion for the series.

Compute the Taylor series for e^x at $x = 0$ and truncate after 4 terms:

```
> taylor( exp(x), x=0, 4 );
```

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + O(x^4)$$

8 Limits

Maple can compute limits of a limited range of expressions. The expression in question must be an expression in one variable. For instance:

```
> limit( 1/x, x=infinity );
0
```

determines the limit of $1/x$ w.r.t. x at the limit point plus infinity.

The **limit** command will assume that any unassigned variables in the expression are real and nonzero. In the following example a is assumed real and nonzero:

```
> limit( a/x, x=infinity );
0
```

The capitalized function name **Limit** is the inert limit function, which simply returns unevaluated.

```
> Limit( a/x, x=infinity );

$$\lim_{x \rightarrow \infty} \frac{a}{x}$$

```

If Maple is not able to find the solution it will return unevaluated (if the limit doesn't exist) or a range (if the limit solution has a range of values).

```
> limit( a/x, x=0 );

$$\lim_{x \rightarrow 0} \frac{a}{x}$$

```

9 Solving Equations

There are many Maple functions for solving systems of different kinds of equations. The simplest and most commonly used of these is **solve**, which solves algebraic systems of equations:

Solve a single equation with one unknown and one solution:

```
> solve( exp(x)=1 );
0
```

Solve for an unknown variable x in a single equation with an unknown constant a and two solutions:

```
> solve( x^2=a, x );

$$\sqrt{a}, -\sqrt{a}$$

```

Maple returns the solution as a set, i.e. denoted by enclosing braces {}, if the equation and the unknown variable is given as a set. This is usually the most convenient form:

```
> solve( {x^2=a}, {x} );

$$\{x = \sqrt{a}, \{x = -\sqrt{a}\}$$

```

Solve a linear system of equations simultaneously. Here, the equations and the unknown variables have to be given as a set:

```
> solve( {a*x+b*y=c, d*x+e*y=f}, {x,y} );

$$\left\{ x = \frac{bf - ce}{-ae + db}, y = -\frac{af - dc}{-ae + db} \right\}$$

```

Sometimes it is better to define the equation system before solving it. Note the difference between “:=” and “=”.

```
> eqn1:= a*x+b*y=c;
      eqn1 := a x + b y = c
> eqn2:= d*x+e*y=f;
      eqn2 := d x + e y = f
> solve( {eqn1, eqn2}, {x,y} );
      { x =  $\frac{bf - ce}{-ae + db}$ , y =  $-\frac{af - dc}{-ae + db}$  }
```

In some cases, implicit solutions are given in terms of RootOfs. The **RootOf(expr)** represents the roots (‘Nullstellen’) of the expr with respect to its single variable *_Z*. Solve this non-linear equation system:

```
> solve( {x+y=0, x*y=b}, {x,y} );
      { y = RootOf(  $_Z^2 + b$ , label =  $_L2$  ), x = -RootOf(  $_Z^2 + b$ , label =  $_L2$  ) }
```

The procedure **allvalues** will attempt to evaluate expressions involving RootOfs exactly. The roots of nth degree polynomial equations where $n \leq 4$ can be obtained exactly:

```
> allvalues(%);
      { y =  $\sqrt{-b}$ , x =  $-\sqrt{-b}$  }, { y =  $-\sqrt{-b}$ , x =  $\sqrt{-b}$  }
```

By setting the global variable `_EnvExplicit` to true, solve will return explicit solutions for polynomials up to 4th degree directly.

```
> _EnvExplicit:=true;
> solve( {x+y=0, x*y=b}, {x,y} );
      { y =  $\sqrt{-b}$ , x =  $-\sqrt{-b}$  }, { y =  $-\sqrt{-b}$ , x =  $\sqrt{-b}$  }
```

In many cases Maple is not able to find explicit solutions. For example in the case of a polynomial of degree 5 it returns

```
> allvalues(solve(  $x^5 - a x^4 + b x^2 - c x - d = 0$ , x ));
      RootOf(  $_Z^5 - a _Z^4 + _Z^2 b - c _Z - d$ , index = 1 ),
      RootOf(  $_Z^5 - a _Z^4 + _Z^2 b - c _Z - d$ , index = 2 ),
      RootOf(  $_Z^5 - a _Z^4 + _Z^2 b - c _Z - d$ , index = 3 ),
      RootOf(  $_Z^5 - a _Z^4 + _Z^2 b - c _Z - d$ , index = 4 ),
      RootOf(  $_Z^5 - a _Z^4 + _Z^2 b - c _Z - d$ , index = 5 )
```

Consider the above example with all the parameters but *x* known. Again Maple cannot find the explicit solutions:

```
> allvalues(solve(  $x^5 - 2 x^4 + 30 x^2 - 20 x - 40 = 0$ , x ));
      RootOf(  $_Z^5 - 2 _Z^4 + 30 _Z^2 - 20 _Z - 40$ , index = 1 ),
      RootOf(  $_Z^5 - 2 _Z^4 + 30 _Z^2 - 20 _Z - 40$ , index = 2 ),
      RootOf(  $_Z^5 - 2 _Z^4 + 30 _Z^2 - 20 _Z - 40$ , index = 3 ),
      RootOf(  $_Z^5 - 2 _Z^4 + 30 _Z^2 - 20 _Z - 40$ , index = 4 ),
      RootOf(  $_Z^5 - 2 _Z^4 + 30 _Z^2 - 20 _Z - 40$ , index = 5 )
```

However, Maple will find the numerical solutions:

```
> evalf(%);
      1.570890168, 1.973060122 + 2.639460154 I, -0.8938693625, -2.623141050,
      1.973060122 - 2.639460154 I
```

For purely floating-point solutions use **fsolve**. For a general equation, **fsolve** attempts to compute a single real root.

```
> fsolve( sin(x), x );
0
```

However, for polynomials it will compute all real (non-complex) roots, although exceptionally ill-conditioned polynomials may cause **fsolve** to miss some roots.

```
> fsolve( x^5-2*x^4+30*x^2-20*x-40= 0, x );
-2.623141050, -.8938693625, 1.570890168
```

To compute all roots of a polynomial over the field of complex numbers, use the **complex** option.

```
> fsolve( x^5-2*x^4+30*x^2-20*x-40=0, x, complex);
-2.623141050, -.8938693625, 1.570890168, 1.973060122 - 2.639460154 I,
1.973060122 + 2.639460154 I
```

You can search for roots in a given interval only:

```
> fsolve( x^5-2*x^4+30*x^2-20*x-40=0, x,
x=0..infinity )
1.570890168
```

You can specify starting values for the variables to solve for:

```
> fsolve( sin(x), x=3.1 );
3.141592654
```

Note that an **fsolve** computation may fail to find a root even though one exists, in which case specifying appropriate range information or starting values may result in a successful computation.

10 Assignment and Substitution of Solutions

The function **assign** is designed to take solution sets of the type produced by **solve** and actually perform the variable assignments. To show this applied to a previous example:

Start with x and y being mathematical, i.e. unassigned, variables:

```
> x,y;
x,y
```

Solve the equation system and assign a name to the two sets of solutions:

```
> soln := allvalues(solve({x+y, x*y=b}, {x,y}));
soln := { x = -sqrt(-b), y = sqrt(-b) }, { x = sqrt(-b), y = -sqrt(-b) }
```

You can check the first set of the solutions by substituting it into the equation:

```
> subs(soln[1],x+y);
0
> subs(soln[1],x*y);
b
```

Assign the first set of solutions to the independent variables x and y :

```
> assign(soln[1]);
```

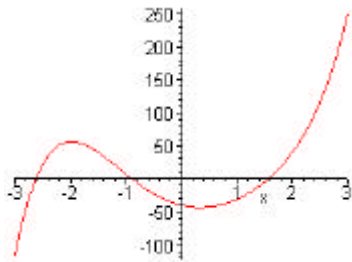
x and y are now programming, i.e. assigned, variables

```
> x,y;
-sqrt(-b), sqrt(-b)
```

11 Graphics

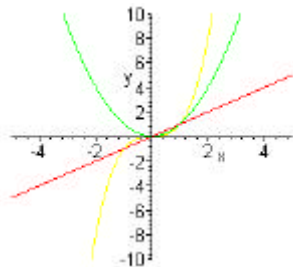
You can plot expressions in Maple. Two-dimensional plots are generated using the **plot** command:

```
> plot( x^5-2*x^4+30*x^2-20*x-40, x=-3..3 );
```



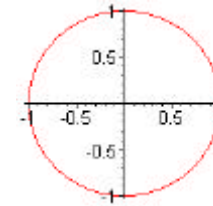
To plot more than one function in the same plot, give plot a list of functions

```
> plot([x,x^2,x^3],x=-10..10, y=-10..10);
```



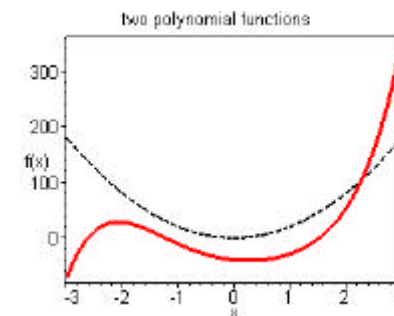
Sometimes a two-dimensional plot is defined implicitly: both the x-coordinate and the y-coordinate are functions of a parameter. These graphs are called parametric plots

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



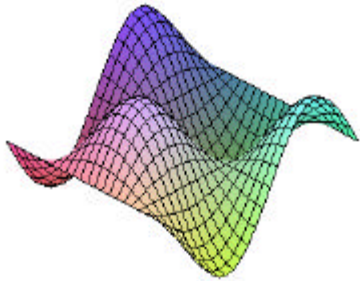
You can use the menu items style, color, axes and projection to change the appearance of the graph. You can do even more using options in the plot command: see **help(plot)** for details.

```
> plot( [x^5+20*x^2-10*x-40, 20*x^2], x=-3..3,
  axes=box, linestyle=[1, 4], thickness=[3, 2],
  color=[red, black], title="two polynomial
  functions", labels=["x","f(x)],
  tickmarks=[6,4]);
```



To generate 3-D plots you use the **plot3d** command:

```
> plot3d( sin(x)*cos(y), x=0..2*Pi, y=0..2*Pi);
```



Show a contourplot of the above 3-dimensional function. Along the contour curves, the function value is constant, as in a topographical map. The **contourplot** command is part of the **plots** package.

```
> with(plots):  
> contourplot( sin(x)*cos(y), x=0..2*Pi,  
y=0..2*Pi);
```

