Maple Reference Sheets

Paul Ullrich

August 30th, 2008

Contents

1 Fundamental Commands

`restart;`
Resets all Maple variables and settings to their default values. Should be at the beginning of every program.

`?<command>;`
Brings up Maple help on the specified command.

`with(<package>);`
Include `<package>` in the list of packages to search for additional commands.
Example: `with(LinearAlgebra);`

`<variable> := <value>;`
Assignment of `<value>` to `<variable>`.
Example: `HalfPi := Pi/2;`

`Digits := <precision>;`
Sets the default precision for floating point operations to `<precision>`.
Default: `Digits := 10;`

`<func> := <var> -> <expression>;`
Constructs a single-variable function that maps `<variable>` to `<expression>`.
Example: `sinc := x -> sin(x)/x;`

`<func>(<var>);`
Evaluate a single-variable function.
Example: `sinc(5);`

`<func> := (<v1>,...,<vn>) -> <expression>;`
Constructs a multi-variable function that maps variables `<v1>`, ..., `<vn>` to `<expression>`.
Example: `F := (x,y) -> sin(x)*sin(y);`

`%`
A variable containing the output from the immediately previous evaluation.
Example: `x^2:  eval(%, x=5);`

`%%`
A variable containing the output from the second last evaluation.
Example: `x^2:  x^3:  eval(% + %%, x=5);`

`<func>(<v1>,...,<vn>);`
Evaluate a multi-variable function.
Example: `F(5,z);`

`eval(<expression>, <variable> = <value>);`
Evaluate the given expression containing variable `<variable>` at `<value>`.
Example: `eval(sin(x^2), x=5);`

`subs(<v1>=a, ..., <vn>=b, <expression>);`
Perform the given substitutions into the given expression.
Example: `subs(x=arcsin(z), tan(x));`

`evalf(<expression>);`
Evaluate the given expression in terms of floating point numbers (removes `Pi`, for instance).
Example: `evalf(sqrt(Pi));`

`simplify(<expression>);`
Attempt to simplify the given expression.
Example: `simplify(sin(arctan(x)));`

`expand(<expression>);`
Expand a factored expression.
Example: `expand((x+1)^5);`

`factor(<expression>);`
Collect factors in a given expression.
Example: `factor(x^3 + 3*x^2 + 3*x + 1);`

`{<a1>,...,<an>}`
Constructs a set (repeats are not allowed) with elements `<a1>`, ..., `<an>`.
Example: `Easy := {1,2,3};`

`[<a1>,...,<an>]`
Constructs a list with elements `<a1>`, ..., `<an>`.
Example: `Fib := [1,1,2,3,5,8];`

`<V>[<n>]`
Shorthand for the n^{th} element of object `<V>`.
Example: `S := [1,1,2,3,5]:  S[2];`

`op(<n>, <expression>);`
Extracts the n^{th} operand of the given expression. Used for extracting elements of a list, set, equality or other expression.
Example: `op(2, [1,2,3,4,5]);`

`nops(<expression>);`
Gives the number of operands in a given expression.
Example: `nops([1,2,3,4,5]);`

`map(<function>, <expression>);`
Apply the function `<function>` to each term or element of `<expression>`.
Example: `map(x -> x^2, {1,2,3});`

`unapply(<expression>, <list of vars>);`
Transform `<expression>` into a function with parameters given by `<list of vars>`.
Example: `unapply(x^2 + y^2, [x, y]);`

`solve(<set of eqs>[, <set of vars>]);`
Solve a set of equations for the given set of variables.
Example: `solve(x=y,x+y=2,x,y);`

`fsolve(<set of eqs>);`
Numerically solve the given set of equations for all variables.
Example: `fsolve(x^3+Pi*x^2+Pi^2*x+1=0;);`

2 The Assume Facility

`assume(<variable>, <property>);`
Impose assumptions given by `<property>` on `<variable>`.
Example: `assume(a, 'real');`

`assume(<expression>);`
Impose assumptions given by `<expression>`.
Example: `assume(c > 0);`

`additionally(<variable>, <property>);`
Impose additional assumptions on `<variable>`, without removing existing assumptions.
Example: `additionally(c, 'integer');`

`additionally(<expression>);`
Impose additional assumptions given by `<expression>`, without removing existing assumptions.
Example: `additionally(a < 5);`

`is(<variable>, <property>);`
Determine if `<variable>` has the specified property given by `<property>`.
Example: `is(a, 'real');`

`is(<expression>);`
Determine if `<expression>` is always true.
Example: `is(a^2 >= 0);`

`coulditbe(<variable>, <property>);`
Determine if `<variable>` could have the specified property given by `<property>`.
Example: `coulditbe(a, 'integer');`

`coulditbe(<expression>);`
Attempt to determine if `<expression>` could hold.
Example: `coulditbe(a^2 = 1.0);`

`about(<variable>);`
Give information on `<variable>`, including assumptions made.
Example: `about(a);`

3 Calculus

`diff(y, x);`
Calculates dy/dx.
Example: `diff(sin(x^2), x);`

`diff(y, x_1, ..., x_n);`
Calculates derivative of y with respect to all variables `x_1`, ..., `x_n`. Repetition of variables is allowed.
Example: `diff(sin(y)*cos(x),x,x,y);`

`limit(y, x=a [, left|right]);`
Evaluate the limit of the given expression y at the point $x = a$. Whether to use the left or right limit may also be specified.
Example: `limit(x/abs(x), x=0, right);`

`int(y, x);`
Calculates the indefinite integral of y with respect to x.
Example: `int(sin(x)*tan(x),x);`

`Int(y, x);`
Equivalent to '`int`', except does not evaluate the resulting integral (known as the *inert form*).

`int(y, x=a..b);`
Calculates the definite integral of y with respect to x over the interval $[a, b]$.
Example: `int(sin(x)*tan(x),x=0..Pi/4);`

`Int(y, x=a..b);`
Equivalent to '`int`', except does not evaluate the resulting integral (known as the *inert form*).

`evalf(Int(y, x=a..b));`
Numerical integration of y with respect to x over the given interval. Note the usage of the inert integration command `Int`.
Example: `evalf(Int(exp(-x^2), x=0..1));`

`sum(<expression>, n=a..b);`
Evaluate the given summation.
Example: `sum(k^2, k=1..n);`

`Sum(<expression>, n=a..b);`
Equivalent to '`sum`', except does not evaluate the result (known as the *inert form*).

`product(<expression>, n=a..b);`
Evaluate the given product.
Example: `product(k+c,k=1..n);`

`Product(<expression>, n=a..b);`
Equivalent to '`product`', except does not evaluate the result (known as the *inert form*).

`series(y, x=a [, <order>]);`
Calculate the Taylor series expansion of y about the point $x = a$ up to `<order>` terms.
Example: `series(ln(x), x=1, 4);`

4 Graphics

`plot(<expr>, <var>=a..b);`
Plot the expression `<expr>` containing variable `<var>` over the range `a..b`.
Example: `plot(BesselJ(1,x), x=-10..10);`

`plot3d(<expr>, <var1>=a..b, <var2>=c..d);`
Plot, in 3D, the expression `<expr>` containing variables `<var1>` and `<var2>` over the range `a..b` and `c..d`.
Example: `plot3d(x^2+sqrt(y), x=-1..1, y=0..2);`

5 Linear Algebra

`with(LinearAlgebra);`
Include standard linear algebra functionality.

`?LinearAlgebra`
Help on the full set of linear algebra functionality.

`Vector(<list of values>);`
Construct a Vector object from a list of values.
Example: `V := Vector([1,2,3]);`

`Matrix(<list of lists of values>);`
Construct a Matrix object from a list of lists of values.
Example: `M := Matrix([[1,1],[0,1]]);`

`.`
Matrix or vector multiplication.
Example: `N := M.M;`

`Determinant(<Matrix>);`
Calculate the determinant of the given matrix.
Example: `Determinant(M);`

`Eigenvalues(<Matrix>);`
Calculate the eigenvalues of the given matrix.
Example: `Eigenvalues(M);`

`Eigenvectors(<Matrix>);`
Calculate the eigenvectors and associated eigenvalues of the given matrix.
Example: `Eigenvectors(M);`

`Transpose(<Matrix>);`
Perform the matrix transpose operation.
Example: `Transpose(M);`