Maxima Reference Card (1)

General & Help	
end of a command	`;`
end without output	`$`
exit Maxima	`quit()`
describe command	`? <string>`
get an example	`example(<command>)`
previous computation	`%`
i-th previous computation	`%th(i)`
don't evaluate	`'<expr>`
Operators	
basic operations	`+, -, *, /, ^`
factorial	`!`
assignment	`:`
function definition	`:=`
equality	`=`
not equal	`#`
inequality	`<, >, >=, <=`
matrix multiplication	`.`
logical operators	`and`, `or`, `not`
Basic functions	
absolute value	`abs(<expr>)`
$\lfloor x \rfloor$	`fix(<num>)`
checks, if even / odd	`evenp(<expr>)`, `oddp(<expr>)`
checks, if a predicate holds	`is(<pred>)` (e.g. `is(x<y)`)
square root	`sqrt(<expr>)`
integer square root	`isqrt(<expr>)`
maximum, minimum	`max(<expr1>,<expr2>,...)`, `min(..)`
division remainder	`mod(<dd>,<ds>)`
random integer $r \in \{0, \ldots, n-1\}$	`random(n)`
signum	`sign(<expr>)`, `signum(<expr>)`
exponential function	`exp(<expr>)`
greatest common divider	`gcd(<expr1>,<expr2>)`
factors an expression	`factor(<expr>)`
numerator of a fraction	`num(<expr>)`
denominator of a fraction	`denom(<expr>)`
real part of a complex number	`realpart(<expr>)`
imaginary part of a complex number	`imagpart(<expr>)`
$\prod_{k=low}^{up} f(k)$	`product(f,k,low,up)`
$\sum_{k=low}^{up} f(k)$	`sum(f,k,low,up)`
sums a function over a list	`lsum(<expr>,<index>,<list>)`
Working with expressions	
sorting a list	`sort(<list>)`, `sort(<list>,<pred>)`
substitute x for y in z	`subst(x,y,z)`
combining all terms of an expression over a common denominator	`xthru(<expr>)`
sublist of a list, for which a boolean function returns `true`	`sublist(<list>,<boolfun>)`
expanding expressions	`expand(<expr>)`
Constants	
π	`%pi`
euler's number	`%e`
imaginary unit	`%i`
boolean constants	`true`, `false`
Plotting	
plotting an expression as a function of one variable	`plot2d(<expr>,[<var>,<min>,<max>])`
plotting an expression as a function of two variables (specify range as above)	`plot3d(<expr>,range1,range2)`
Input and Output	
loading a file	`load("<fullpath>")`
printing an expression	`print(<expr>)`
saving expressions to a file	`save("<fullpath>",expr1,...)`
loading a file (saved by maxima)	`loadfile("<fullpath>")`
TEX output	`tex(<expr>)`, `tex(<expr>,<file>)`
executing a system command	`system("<command>")`

Maxima is distributed under the GNU General Public License, visit http://maxima.sourceforge.net/ to get information about Maxima.

Maxima Reference Card (2)

Floating point operations	
converting into float/ bigfloat	`float(<expr>)`, `bfloat(<expr>)`
set floating point precision	`fpprec:<value>`
set digits to print	`fpprintprec:<value>`
rounding a float to an integer	`?round(<expr>)`
truncating a floats decimal places	`?truncate(<expr>)`
Polynomials	
gives quotient and remainder	`divide(<poly1>,<poly2>)`
converts into horner's scheme	`horner(<poly>,<var>)`
Logarithms and Trigonometry	
natural logarithm	`log(<expr>)`
principal branch of complex logarithm	`plog(<expr>)`
sinus, arcus sinus	`sin(<expr>)`, `asin(<expr)`
cosinus, arcus cosinus	`cos(<expr>)`, `acos(<expr)`
tangent, arcus tangent	`tan(<expr>)`, `atan(<expr>)`
Differential and Integral calculus	
infinity, negative infinity	`inf`, `minf`
$\lim_{x\to k} f(x)$	`limit(f,x,k)`
$\lim_{x\searrow k} f(x)$, $\lim_{x\nearrow k}$	`limit(f,x,k,PLUS)`, `..(..,MINUS)`
differentiate an expression	`diff(<expr>,<var>,<ntimes>)`
antiderivative of an expression	`integrate(<expr>,<var>)`
$\int_{x=low}^{up} f(x)\,dx$	`integrate(f(x),x,<low>,<up>)`
Equations	
find all roots of a real polynomial	`allroot(<poly>)`
solve a system of equations	`solve([<eq1>,..],[<var1>,...])`
solve an ordinary differential equation	`ode2(<equ>,<y-var>,<x-var>)`
finds the zero of $f(x)$ in $[a, b]$	`interpolate(f,x,a,b)`
Matrices and Linear Algebra	
creates a matrix (where row is a list)	`matrix(<row1>,<row2>,..)`
enter a $m \times n$ matrix value by value	`entermatrix(m,n)`
computes the adjoint of a matrix	`adjoint(<matrix>)`
gets column / row of a matrix	`col(<matr>,<num>)`, `row(<m>,<n>)`
gets element (i,j) from a matrix	`<matrix>[i,j]`
creates a copy of a matrix	`copymatrix(<matrix>)`
computes the charact. polynomial	`charpoly(<matrix>,<var>)`
computes the determinant	`determinant(<matrix>)`
computes the eigenvalues	`eigenvalues(<matrix>)`
computes eigenvectors	`eigenvectors<matrix>)`
creates an $n \times n$ identity matrix	`ident(n)`
finds an inverse matrix	`invert(<matrix>)`
maps a func. onto each matrix-elem.	`matrixmap(<fun>,<matrix>)`
computes the rank of a matrix	`rank(<matrix>)`
transposes a matrix	`transpose(<matrix>)`
computes an upper triangular form	`triangularize(<matrix>)`
Series	
expands into a powerseries	`powerseries(<expr>,<var>,<point>)`
expands into a truncated taylor series	`taylor(<fun>,<var>,<point>,<pot>)`
Number theory	
binomial coefficient	`binomial(<expr>,<expr>)`
converts into a continued fraction	`cf(<expr>)`
nth Fibonacci number	`fib(n)`
checks, if an expression is prime	`primep(<expr>)`
Euler's φ-function	`totient(<expr>)`
Lists	
creates a list	`[<expr1>,..]`
creates a copy of a list	`copylist(<list>)`
gets the ith element	`<list>[i]`
concatenates two lists	`append(<list1>,<list2>)`
Programming	
program block	`block(<epxr1>,..)`
for-loop	`for <decl> thru <num> step <num> do <block>`
while-loop	`for <decl> while <cond> do <block>`
if-then-else construct	`if <cond> then <block> else <block>`
returns a value	`return(<value>)`
function definition with block	`f(x):=block(<expr1>,..,return(<value>))`
Miscellaneous	
random number from normal distrib. with `mean` and standard `deviation`	`gauss(<mean>,<dev>)`