Cheatography

Sympy Cheat Sheet

by gloo13 via cheatography.com/185324/cs/39863/

Basic Operations

expr.subs([(x, 2), (y, 4), (z, 0)])	substitute x with 2 etc.
sympify(str_expr)	convert strings into SymPy expressions
expr.evalf(15, chop=True)	evaluate a numerical expression into a floating point number
lambdify(x, expr, "numpy")	converts the SymPy names to the names of the given numerical library
init_printing()	This will automatically enable the best printer available in your environment.
simplify(expr)	simplify mathematical expressions
expand(expr)	expand polynomial expressions
factor(expr)	takes a polynomial and factors it into irreducible factors over the rational numbers
factor_list(expr)	returns a list with the factors. More structured.
collect(expr, x)	collects common powers of a term in an expression
cancel(expr)	take any rational function and put it into the standard canonical form
apart(expr)	performs a partial fraction decomposition on a rational function

Matrices

Matrix([1, 2, 3])	matrix constructor(mutable matrix)
shape(expr)	shape of matrix
M.row(0)	get the first row
M.col(-1)	get the last column
M.col_del(0)	delete first column
M.row_del(1)	delete second row
M.row_insert(1, Matrix([[0, 4]]))	insert a row
M.col_insert(0, Matrix([1, -2]))	insert a column
M**-1	inverse of M
M.T	transpose of M
eye(n)	create a nxn identity matrix
zeros(n,m)	creates a nxm matrix of zeroes
ones(n,m)	creates a nxm matrix of ones
diag(expr)	creates a matrix with expr in the diagonal
M.det()	computes the determinant of M
M.rref()	put a matrix into reduced row echelon form
M.nullspace()	returns a list of column vectors that span the nullspace of the matrix
M.columnspace()	returns a list of column vectors that span the columnspace of the matrix
M.eigenvals()	eigenvals returns a dictionary of eigenvalue: algebraic_multiplicity pairs

Matrices (cont)

M.eigenvects()	returns a list of tuples of the form (eigenvalue, algebraic_multiplicity, [eigenvectors])
M.diagonalize()	returns a tuple (P, D), where D is diagonal and M = P DP **−1
M.charpoly(lamda)	return the characteristic polynomial

Trigonometric Simplification

trigsimp(expr)	simplify expressions using trigonometric identities
expand_trig(expr)	expand trigonometric functions

Powers

powsimp(expr)	use power identities
expand_power_exp(x**(a + b))	x**a * x**b
expand_power_base((xy)*a)	x**a * y**a
powdenest((x**a**)b)powdenest((x**a**)b)	x**(a*b)

Exponentials and logarithms

expand_log(expr)
logcombine(expr)

Special Functions

factorial(n)	return the factorial of n
binomial(n, k)	return the binomial coefficient of n and k
gamma(z)	return the gamma function
expr.rewrite(function)	rewrite expr in terms of function
expand_func(expr)	expand special functions

Special Functions (cont)

hypere-xpand(-expr)	rewrite hyper in terms of more standard functions
combsi-mp(-expr)	simplify combinatorial expres-sions
gammas imp-(expr)	simplify expressions with gamma functions or combin-atorial functions

Assumptions

positive	negative
real	complex
integer	
expr.a-ssu-mpt-ions0	The full set of known predicates for a symbol
posify-(expr)	replace all symbols in an expression with symbols that have the assumption positi-ve=True

Calculus

diff(expr, x, n)	nth order derivative of expr in terms of x
Derivative(expr, x, n)	create an unevaluated derivative
deriv.doit()	evaluate an unevaluated derivative
integrate(expr, x, a, b)	integrate expr from a to b
Integral(expr, x, n)	create an unevaluated integral
limit(expr, x, xo)	limit of expr to xo
Limit(expr, x, xo)	create an unevaluated limit
expr.series(x, x0, n)	nth order series expansion of expr around x0
expr.series(x, x0, n).rem-oveO()	remove O notation

Calculus (cont)

differentiat-e_finite(-expr)	differentiate using finite differences
expr.as_f-inite_differ-ence()	generate approximations of the derivative to arbitrary order

Solvers

solveset(expr, x, domain=S.Com-plexes, dict=False)	solve expr=0
linsolve([expr1, expr2, ...], (x, y, ...))	solve a linear system of equations
nonlinsolve([expr1, expr2, ...], [x, y, ...])	solve a non linear system of equations
dsolve(diffeq, f(x))	soves differential equation diffeq
roots(expr, x)	o get the solutions of a polynomial including multiplicity