Poprvé s CAS

Systémy CAS (Computer Algebra Systems) jsou určeny především k provádění symbolických algebraických a diferenciálních úprav výrazů a analytickému řešení algebraických a diferenciálních rovnic. K výbavě patří maticové operace, aritmetika neomezené přesnosti ad. Soustředíme se na CAS balíčky v Pythonu (knihovna SymPy), MATLABu (Symbolic Math Toolbox) a GNU Octavu (balíček symbolic). Prostřední z nich je komerční, ostatní dva dostupné zdarma. Dalšími významnými konkurenty v oblasti CAS jsou Mathematica a Maple, oba komerční; Wikipedia shrnuje další systémy na stránce List of computer algebra systems.

Python: SymPy

Volně dostupný balíček k Pythonu 2 i 3, aktuálně ve verzi 1.3. Dokumentace: https://docs.sympy.org, včetně odkazu na Tutorial. Dostupná je i online verze: https://live.sympy.org. Obvyklou úvodní akcí je:

```
from sympy import *
```

případně:

```
import sympy; sympy.init_session()
```

což provede následující import a definici symbolů:

```
from __future__ import division
from sympy import *
x,y,z,t = symbols('x y z t')
k,m,n = symbols('k m n', integer=True)
f,g,h = symbols('f g h', cls=Function)
init_printing()
```

Nápověda: např. help(symbols). Definovat lze i takto: x=Symbol('x'); f=Function('f'). Symboly jsou z komplexního oboru, neřekne-li se jinak, např: x=Symbol('x',real=True,positive=True). Reálná nekonečna jsou -oo a oo, na výstupu i -inf a inf, komplexní nekonečno je zoo, nečíslo je nan. SymPy používá syntaxi procedurální i objektovou.

Základní postupy, úpravy, tisk

zjednodušení: Integer(2)/4 pro 1/2, Float(10)**1000. pro 1e1000, sqrt(8) pro 2*sqrt(2), sqrt(-4) pro 2*I, atan(1)*4 pro pi, exp(1) pro E, Float(1)/Float(0) pro inf, Float(1)/0 pro zoo, Float(0)/0 pro nan

vyčíslení: sqrt(2).evalf() pro 1.41421356237310 (15 číslic), pi.evalf(1000) pro 1000 číslic

výrazy: vytvoření ze symbolů: expr=x+2*y; vytvoření z řetězce: expr=sympify('x+2*y')
str a lambdify pro převod ze SymPy na řetězec nebo lambda funkci

speciální funkce: factorial(n), binomial(n,k), gamma, erf, besselj aj., legendre, Ynm, DiracDelta(x), Heaviside(x) aj.

úpravy: černá skříňka simplify(x*expr) nebo specificky expand(x*expr) nebo factor(x*expr+y**2)
trigsimp,expand_trig, powsimp,expand_power_base,expand_power_exp, logcombine,expand_log, expand_func, hyperexpand, combsimp, gammasimp aj.
rozepsání funkcí: Ynm(2,0,th,ph).expand(func=True)
přepsání pomocí jiné funkce: cos(x).rewrite(sin); factorial(n).rewrite(gamma)

dosazování: expr.subs([(x,1),(y,2)]) nebo expr.subs(x,1).subs(y,2) nebo expr.evalf(subs={x:1,y:2})

pretty printing: init_printing() pro LaTeX, Unicode, ASCII aj., init_printing(pretty_print=False) pro vypnutí
expr=Integral(sqrt(1/x),(x,0,oo)); pprint(expr); latex(expr); ccode(pi) a fcode(pi) pro C a Fortran

Diferenciální počet

derivování: funkce diff(x**2+sin(x),x) i metoda (x**2+sin(x)).diff(x,x)
odložené vyčíslení: D=Derivative(x**2*sin(y),x,y); D.doit()

integrování: funkce integrate(x**2+sin(x),x) i metoda (x**2+sin(x)).integrate(x)
odložené vyčíslení: I=Integral(x**2+sin(x),x); latex(I); I.doit()
určité integrály: integrate(sin(x**2),(x,-oo,oo)); (sin(x)*sin(y)).integrate((x,0,pi),(y,0,pi))

limity: limit(sin(x)/x,x,0); limit(exp(-x),x,oo); exp(x).limit(x,oo); L=Limit(1/x,x,0,'+'); latex(L); L.doit()

rozvoje: x0=0; n=15; f=sin(x); f.series(x,x0,n);

konečné dif.: f=Function('f'); h=Symbol('h'); df=f(x).diff(x); df.as_finite_difference([0,h])
df2=f(x).diff(x,x); df2.as_finite_difference([-h,0,h]); též finite_diff_weights, apply_finite_diff

Řešiče soustav rovnic

Rovnice se zapíše jako Eq(lhs,rhs) a řeší se solve(Eq(lhs,rhs)) nebo solve(Eq(lhs-rhs,0)) nebo solve(lhs-rhs).
Řešič solve zastarává a nahrazují ho solvery solveset, linsolve a nonlinsolve.

algebraické rovnice: solveset(x**2-1,x); solveset(x**3-1,x); solveset(x**3-1,x,domain=S.Reals)
solveset(sin(x)-1,x); solveset(sin(x)/x,x); solveset(1/x,x); solveset(0,x)

lineární rovnice: a,b,c,d,p,q,x,y=symbols('a b c d p q x y'); linsolve([a*x+b*y-p,c*x+d*y-q],[x,y])

totéž A=Matrix([[a,b],[c,d]]); v=Matrix([p,q]); linsolve([A,v],x,y)

zatím totéž solve([a*x+b*y-p,c*x+d*y-q],[x,y])

zatím ne solveset([a*x+b*y-p,c*x+d*y-q],[x,y])

nelineární rovnice: nonlinsolve([2*x*y+3*x*z-93,4*x*y+5*y*z-235,6*x*z+7*y*z-371],(x,y,z))

diferenciální rovnice: t=Symbol('t'); y=Function('y'); eq=Eq(y(t).diff(t),-y(t)); dsolve(eq,y(t))
s poč. podmínkou dsolve(eq,y(t),ics={y(0):1})
vyššího řádu eq=Eq(y(t).diff(t,t),-y(t)); dsolve(eq,y(t))
s poč. podmínkou dsolve(eq,y(t),ics={y(0):0,y(t).diff(t).subs(t,0):1})
jiný zápis t=Symbol('t'); y=Function('y')(t); dy=Derivative(y,t); dy2=Derivative(y,t,t);
dsolve(dy2+y,ics={y.subs(t,0):0, dy.subs(t,0):1})
soustava y1=Function('y1')(t); y2=Function('y2')(t); dy1=Derivative(y1,t); dy2=Derivative(y2,t);
dsolve([dy1-y2, dy2+y1],ics={y1.subs(t,0):0, y2.subs(t,0):1})
s okraj. podmínkou dsolve([dy1-y2, dy2+y1],ics={y1.subs(t,0):1, y2.subs(t,pi):-1})

Matice

Matice se zapíše jako seznam řádků, M=Matrix([[0,1],[1,0]]). Sloupcový vektor je v=Matrix([1,2]), řádkový se získá transpozicí, v.T. Připraveny jsou konstruktory zeros, ones, eye, diag aj. K dispozici jsou mj. metody row, col, det a operátory +, -, *, **; inverzní matice je M**-1. Vlastní čísla a vektory počítají metody eigenvals a eigenvects, charakteristický polynom vrací metoda charpoly. (Pro zachování klíčového slova Pythonu lambda lze v Sympy použít symbol lamda, zobrazovaný shodně jako lambda.)

```
a,b=symbols('a b'); M=Matrix([[0,a],[b,0]]); M.T; M**-1; M.det(); M.eigenvals()
```

MATLAB: Symbolic Math Toolbox

Původní systém MuPAD se stal součástí MATLABu jako komerční Symbolic Math Toolbox.

syms a, a=sym(a) zavedení symbolů pro proměnné
syms a b c d x, [a b]*[c d]'
syms x, diff(x^2), int(x^2)
solve, dsolve string řešení algebraických a diferenciálních rovnic symbolicky
syms a b c x, solve a*x^2+b*x+c=0
syms a b c d p q x, solve('a*x+b*y=p','c*x+d*y=q'); [ans.x ans.y]
syms x y, result=dsolve('Dx=y', 'Dy=-x'); result.x, result.y
vpa string precision vyčíslení výrazu s volitelnou přesností (variable-precision arithmetic), př. vpa pi 100
ezplot(f,[x0 x1]) analytická funkce f(x) na <x0,x1> (easy plot), lze implicitní i parametrický zápis
syms t x y; ezplot(@(x) x.^2,[0 1]); ezplot(x^2+y^2-1,[-1 1 -1 1]); ezplot(cos(t),sin(t))
ezpolar(f,[ph0 ph1]) pro funkci r=f(phi), př. ezpolar(1/t,[1 pi*6])
ezplot3(x,y,z,[t0 t1]) 3D parametrická funkce s možností animace
syms t x y z; x=cos(t); y=sin(t); z=t; ezplot3(x,y,z,[0 pi*6],'animate')
ezsurfc(f,[x0 x1 y0 y1]) plochy a izočáry f(x,y) (surface & contours), př. ezsurfc(x*y)
ezcontour, ezcontourf, ezmesh, ezmeshc, ezsurf

GNU Octave: symbolic

Volně dostupný GNU Octave se drží syntaxe MATLABu, což platí i pro balíček symbolic. Ten se pro realizaci zadání obrací na Python se SymPy. Octave v aktuální verzi 4.4.1 má balíček symbolic ve verzi 2.7.1, který spolupracuje se SymPy 1.3. Octave je třeba spustit v prostředí s dostupným Pythonem, např. ve Windows se nejprve spustí Anaconda Prompt, pak Octave příkazem C:\Octave\octave.vbs --force-gui a v něm se zavede balíček příkazem pkg load symbolic. Není-li instalován (viz pkg list), nainstaluje se příkazem pkg install -forge symbolic. Nápověda: https://octave.sourceforge.io/symbolic, offline: help @sym/solve, help vpa apod.

definice symbolů: syms x y f(x) neboli x=sym('x') atd., více help sym, výpis symbolů: syms
úpravy: expr=x+2*y, simplify(x*expr), expand(x*expr), factor(x*expr+y**2)
vyčíslení: vpa(sqrt(2)), vpa(pi,1000)
pretty printing: disp(A,'flat')
derivování: diff(x**2+sin(x)), diff(x**2+sin(x),2)
integrování: int(x**2+sin(x)), int(sin(x**2),-inf,inf), int(int(sin(x)*sin(y),x,[0,pi]),y,[0,pi])
limity: limit(sin(x)/x,x,0), limit(exp(-x),x,inf), limit(exp(x),x,inf), limit(1/x,x,0,'+')
algebraické rovnice: syms x; solve(x**2-1,x), solve(x**3-1,x)
syms x a b c; res=solve(a*x^2+b*x+c==0,x)
syms x y a b c d p q; res=solve([a*x+b*y;c*x+d*y]==[p;q],[x,y]); res.x,res.y
[resx,resy]=solve(a*x+b*y==p,c*x+d*y==q, x,y)
diferenciální rovnice: syms y(t); eq=diff(y,t)==-y(t); dsolve(eq); dsolve(eq,y(0)==1)
syms y(t); eq=diff(y,2)==-y(t); dsolve(eq); dsolve(eq,y(0)==0,diff(y)(0)==1)
syms y1(t) y2(t); eq=[diff(y1)==y2;diff(y2)==-y1]; res=dsolve(eq); res{1,1:2}
res=dsolve(eq,y1(0)==0,y2(0)==1); res{1,1:2}