Sage Quick Reference: ODEs

Marshall Hampton and William Stein

Sage Version 4.5.2

Builtin constants and functions

Constants: $\pi =$ `pi` $\quad e =$ `e` $\quad i =$ `I` $=$ `i`

$\infty =$ `oo` $=$ `infinity` NaN=`NaN` $\log(2) =$`log2`

$\phi =$ `golden_ratio` $\quad \gamma =$ `euler_gamma`

Approximate: `pi.n(digits=18)` $= 3.14159265358979324$

Builtin functions: `sin cos tan sec csc cot sinh cosh tanh sech csch coth log ln exp` ...

Defining symbolic expressions

Create symbolic variables:

`var("t u theta")` or `var("t,u,theta")`

Use `*` for multiplication and `^` for exponentiation:

$2x^5 + \sqrt{2}$ = `2*x^5 + sqrt(2)`

Typeset: `show(2*theta^5 + sqrt(2))` $\longrightarrow 2\theta^5 + \sqrt{2}$

Symbolic functions

Symbolic function (can integrate, differentiate, etc.):

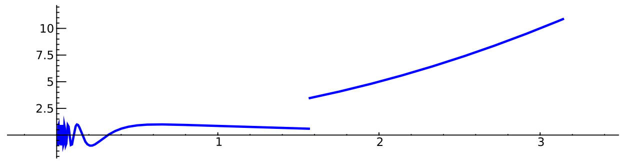
`f(a,b,theta) = a + b*theta^2`

Also, a "formal" function of theta:

`f = function('f',theta)`

Piecewise symbolic functions:

`Piecewise([[(0,pi/2),sin(1/x)],[(pi/2,pi),x^2+1]])`



Python functions

Defining:

```
def f(a, b, theta=1):
    c = a + b*theta^2
    return c
```

Inline functions:

```
f = lambda a, b, theta = 1: a + b*theta^2
```

Factorization

Factored form: `(x^3-y^3).factor()`

List of (factor, exponent) pairs:

`(x^3-y^3).factor_list()`

Simplifying and expanding

Below f must be symbolic (so **not** a Python function):

Simplify: `f.simplify_exp()`, `f.simplify_full()`, `f.simplify_log()`, `f.simplify_radical()`, `f.simplify_rational()`, `f.simplify_trig()`

Expand: `f.expand()`, `f.expand_rational()`

Equations

Relations: $f = g$: `f == g`, $f \neq g$: `f != g`,
$f \leq g$: `f <= g`, $f \geq g$: `f >= g`,
$f < g$: `f < g`, $f > g$: `f > g`

Solve $f = g$: `solve(f == g, x)`, and
`solve([f == 0, g == 0], x,y)`

`solve([x^2+y^2==1, (x-1)^2+y^2==1],x,y)`

Solutions:

`S = solve(x^2+x+1==0, x, solution_dict=True)`

`S[0]["x"]` `S[1]["x"]` are the solutions

Exact roots: `(x^3+2*x+1).roots(x)`

Real roots: `(x^3+2*x+1).roots(x,ring=RR)`

Complex roots: `(x^3+2*x+1).roots(x,ring=CC)`

Limits

$\lim_{x \to a} f(x) =$ `limit(f(x), x=a)`

`limit(sin(x)/x, x=0)`

Derivatives

$\frac{d}{dx}(f(x)) =$ `diff(f(x),x)` = `f.diff(x)`

$\frac{\partial}{\partial x}(f(x,y)) =$ `diff(f(x,y),x)`

`diff` = `differentiate` = `derivative`

`diff(x*y + sin(x^2) + e^(-x), x)`

Integrals

$\int f(x)dx =$ `integral(f,x)` = `f.integrate(x)`

`integral(x*cos(x^2), x)`

$\int_a^b f(x)dx =$ `integral(f,x,a,b)`

`integral(x*cos(x^2), x, 0, sqrt(pi))`

$\int_a^b f(x)dx \approx$ `numerical_integral(f(x),a,b)[0]`

`numerical_integral(x*cos(x^2),0,1)[0]`

Taylor and partial fraction expansion

Taylor polynomial, deg n about a:

`taylor(f,x,a,n)`$\approx c_0 + c_1(x-a) + \cdots + c_n(x-a)^n$

`taylor(sqrt(x+1), x, 0, 5)`

Partial fraction:

`(x^2/(x+1)^3).partial_fraction()`

Numerical roots and optimization

Numerical root: `f.find_root(a, b, x)`

`(x^2 - 2).find_root(1,2,x)`

Maximize: find (m, x_0) with $f(x_0) = m$ maximal

`f.find_maximum_on_interval(a, b, x)`

Minimize: find (m, x_0) with $f(x_0) = m$ minimal

`f.find_minimum_on_interval(a, b, x)`

Minimization: `minimize(f,` *start_point*`)`

`minimize(x^2+x*y^3+(1-z)^2-1, [1,1,1])`

Symbolic ODE solutions

First define y as an unknown function of x:

`y=function('y',x)`

Get the solution to $\frac{d^2y}{dx^2} + 3x = y$:

`desolve(diff(y,x,2)+3*x==y, y)`

With initial conditions $y(0) = 1$, $y'(0) = 2$:

`desolve(diff(y,x,2)+3*x==y, y, ics = [0,1,2])`

Numerical ODE solutions

Plot the solution to $y' = -y + \frac{1}{1+t}$, $y(0) = 1$ from $t = 0$ to $t = 10$:

```
T = ode_solver()
f = lambda t,y: [-y[0] + 1/(1+t)]
T.function = f
T.ode_solve([0,10], [1], num_points = 100)
plot(T.interpolate_solution(),0,10).show()
```