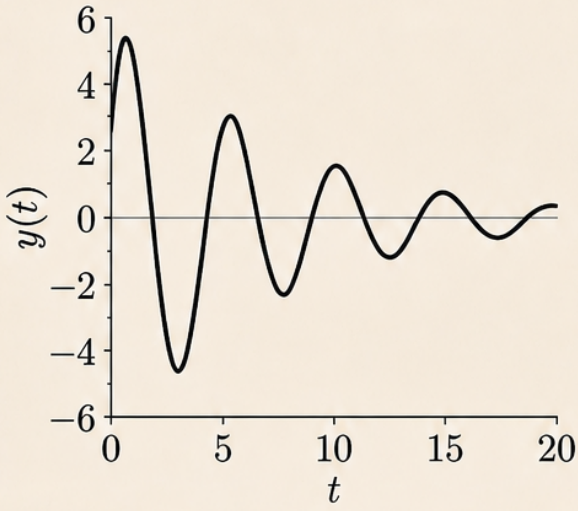
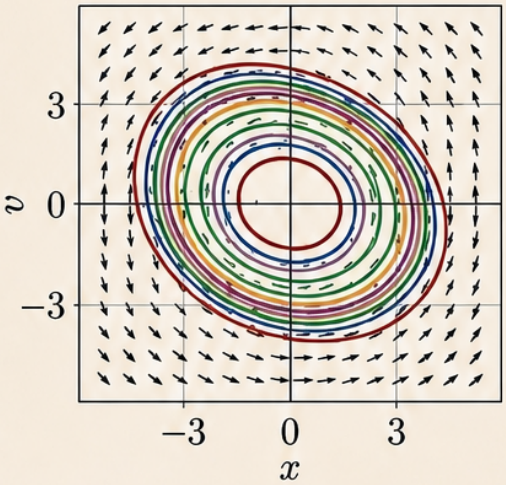
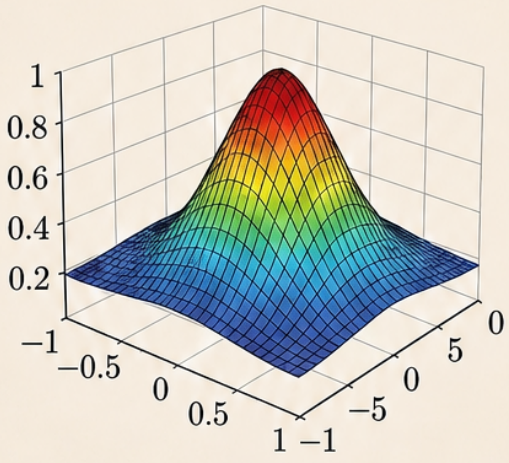
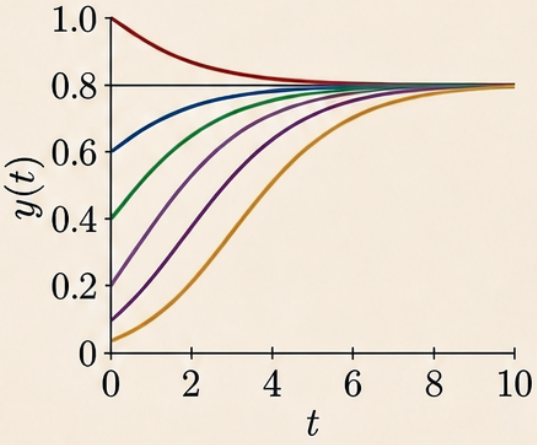
MAXIMA
VIA EXAMPLES

VOLUME 3
Guide to Differential Equations with Maxima









Guide to Differential Equations with Maxima

Contents

1 Introduction

This guide is the third volume in the "Maxima via Examples" collection. Its purpose is to show how Maxima can be used to solve differential equations, either symbolically or by means of numerical methods.
This document, together with the PDF and wxMaxima versions of this Guide to Differential Equations with Maxima, is available on the https://maxima-french-doc.fr website.

In addition to Maxima's standard commands for solving differential equations, Maxima also provides packages containing solution methods for particular types of differential equations. Some of these packages will be presented in this document.

This guide is based on Maxima version 5.49 and therefore also applies to all later versions. I would like to thank *Dr Wolfgang Lindner*, whose book Introductory Differential Equations with Maxima was a valuable source of inspiration in the preparation of this guide. I should also mention the manuals by *Edwin L. Woollett*, entitled Maxima by Example, in particular Chapter 3, Ordinary Differential Equation Tools. Finally, the manual wxMaxima for calculus volume 2, and especially Chapter 4, contains very useful information on this topic.

Throughout this guide, it is assumed that the functions under consideration are differentiable as many times as necessary on the intervals under study. The chapters of this booklet deal with *ordinary differential equations (ODEs)*, characterized by the fact that the unknown function depends on only one variable, often denoted by x or t. A differential equation relates this function to its successive derivatives with respect to that single variable. The other category of differential equations concerns functions of several variables, whose equations involve partial derivatives; these will be referred to as *PDEs (partial differential equations)*.

If you have an Internet connection, the red links entitled **Click here for Live Demo** (just after Result :) loads the Maxima commands presented on the Maxima server https://net124.reltub.ca/yamwi/ and enables you to run them online. This feature allows you to experiment and modify parameters or variables without having to install Maxima on your computer.

2 Defining an ODE with Maxima

If y is a function of the variable x, its successive derivatives $\frac{\mathrm{d}y}{\mathrm{d}x}$, $\frac{\mathrm{d}^2y}{\mathrm{d}x^2}$, ... are defined in Maxima respectively by `diff(y,x)` and `diff(y,x,2)`,...
These derivatives can therefore be used to write a differential equation, provided that `diff` is preceded by **an apostrophe '** in order to prevent Maxima from computing the derivative immediately.

Maxima command:

```
'diff(y,x) = -y;
'diff(y(x),x) = -y(x);
'diff(y,x) + (1/(2*x))*y = 2;
'diff(y,x,2) + y = 0;
'diff(f,x,2) + 3*'diff(f,x) = k*f;
```

Comment:

- The function in the differential equation can be denoted by y or $y(x)$. However, the notation $y(x)$ is essential when several variables are involved.
- An ODE can be assigned to a variable with the command: `eq:'diff(y,x)=-y`.

Result: Click here for Live Demo

(% i1) 'diff(y,x) = -y;

$$\frac{d}{dx}y = -y \qquad (\%\ o1)$$

(% i2) 'diff(y(x),x) = -y(x);

$$\frac{d}{dx}\,y(x) = -\,y(x) \tag{\% o2}$$

(% i3) 'diff(y,x) + (1/(2*x))*y = 2;

$$\frac{d}{dx}y + \frac{y}{2x} = 2 \tag{\% o3}$$

(% i6) 'diff(y,x,2) + y = 0;

$$\frac{d^2}{dx^2}y + y = 0 \tag{\% o6}$$

(% i7) 'diff(f,x,2) + 3*'diff(f,x) = k*f;

$$\frac{d^2}{dx^2}f + 3\left(\frac{d}{dx}f\right) = fk \tag{\% o7}$$

3 Checking whether a function is a solution of an ODE

Principle: replace the unknown function in the differential equation by the function to be tested.

Maxima command:

```
eq: (1+x^2)*'diff(y,x) + 2*x*y = 4*x^3
sol: y = (x^4+1)/(1+x^2);
verif: ev(eq, sol, diff);
ratsimp(verif);
```

Comment: The first command defines the ODE and the second a candidate solution. The third command replaces y in the ODE by the expression defined in sol, then computes the derivatives. The final command simplifies the result in order to determine whether the differential equation is satisfied.

Result: Click here for Live Demo

(% i6) eq: (1+x^ 2)*'diff(y,x) + 2*x*y = 4*x^ 3;

$$\left(x^2+1\right)\left(\frac{d}{dx}y\right) + 2xy = 4x^3 \tag{eq}$$

(% i7) sol: y = (x^ 4+1)/(1+x^ 2);

$$y = \frac{x^4+1}{x^2+1} \tag{sol}$$

(% i8) verif: ev(eq, sol, diff);

$$\left(x^2+1\right)\left(\frac{4x^3}{x^2+1} - \frac{2x\,\left(x^4+1\right)}{\left(x^2+1\right)^2}\right) + \frac{2x\,\left(x^4+1\right)}{x^2+1} = 4x^3 \tag{verif}$$

(% i9) ratsimp(verif);

$$4x^3 = 4x^3 \tag{\% o9}$$

Maxima command:

```
f(x):= (x^4+1)/(1+x^2);
(1+x^2)*diff(f(x),x)+2*x*f(x);
ratsimp(%);
```

Comment: A standard method is used in this example. We define the function f whose status as a solution is to be checked, then explicitly compute the left-hand side of the ODE by replacing y with $f(x)$ and evaluating the derivative. It only remains to simplify the result in order to draw the conclusion.

Result: Click here for Live Demo

(% i10) f(x):= (x^ 4+1)/(1+x^ 2);

$$f(x) := \frac{x^4+1}{1+x^2} \qquad (\%\ o10)$$

(% i11) (1+x^ 2)*diff(f(x),x)+2*x*f(x);

$$\left(x^2+1\right)\left(\frac{4x^3}{x^2+1}-\frac{2x\ \left(x^4+1\right)}{\left(x^2+1\right)^2}\right)+\frac{2x\ \left(x^4+1\right)}{x^2+1} \qquad (\%\ o11)$$

(% i12) ratsimp(%);

$$4x^3 \qquad (\%\ o12)$$

Maxima command:

```
test(fun, equation, var) := block(
    [r],
    r : subst(fun, y, equation),
    r : ev(r, diff),
    is(fullratsimp(lhs(r)-rhs(r)) = 0)
)$
eq2 : 'diff(y,x,2) + y = 0;
f(x):=k*sin(x);
test(f(x),eq2,x);
test(x*sin(x),eq2,x);
```

Comment: To automate the preceding calculation, we define a function that tests whether a function is a solution of an ODE. Its parameters are the function to be tested, the differential equation, and the differentiation variable.
For the calculation, we define a new differential equation eq2, a function f defined by $f(x) = k\sin(x)$, and apply the test procedure. We do the same with the function $x \longmapsto x\sin(x)$.

Result: Click here for Live Demo

(% i18) eq2 : 'diff(y,x,2) + y = 0;

$$\frac{d^2}{dx^2}y+y=0 \qquad (\text{eq2})$$

(% i19) f(x):=k*sin(x);

$$f(x) := k\sin\left(x\right) \qquad (\%\ o19)$$

(% i20) test(f(x),eq2,x);

$$true \qquad (\%\ o20)$$

(% i21) test(x*sin(x),eq2,x);

$$false \qquad (\%\ o21)$$

4 Solving an ODE with ode2, Maxima's built-in solver

Without considering special cases, theory shows that an ordinary differential equation has a family of solution functions, called the general solution. For a first-order ODE (an equation defined by the first derivative of y), the family of functions is determined up to one constant. For a second-order equation, it is determined up to two constants. By default, Maxima denotes these constants by %c, %k1, ...

The built-in Maxima command for finding the general solution of a first- or second-order ODE is

`ode2(differential equation,unknown function,variable).`

Maxima command:

```
eq1 : 'diff(y,x) + 2*y = 6*x;
sol1 : ode2(eq1, y, x);
```

Comment: Direct application of the solving command for a first-order ODE.

Result: Click here for Live Demo

(% i2) eq1 : 'diff(y,x) + 2*y = 6*x;
sol1 : ode2(eq1, y, x);

$$\frac{d}{dx}y + 2y = 6x \tag{eq1}$$

$$y = \%e^{-(2x)}\left(\frac{3\%e^{2x}\,(2x-1)}{2} + \%c\right) \tag{sol1}$$

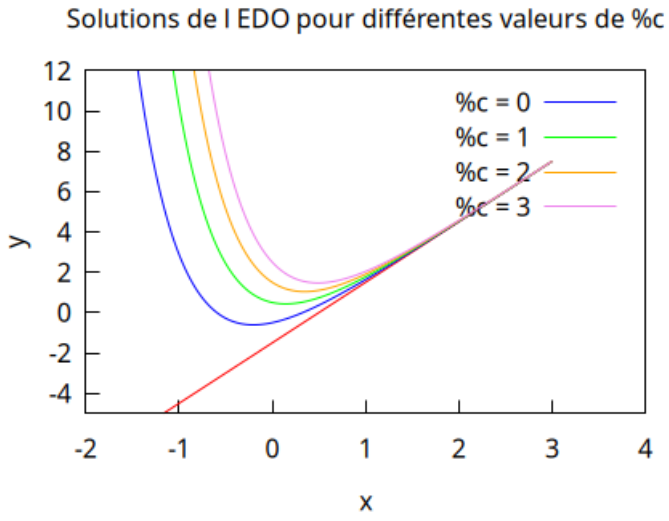
Let us plot several curves from this family of solutions:

Maxima command:

```
s1 : rhs(sol1)$
wxdraw2d(
    xrange = [-2, 4],
    yrange = [-5, 12],
    xlabel = "x",
    ylabel = "y",
    title = "Solutions of the ODE for different values of %c",
    color = red,
    explicit(subst(0, %c, s1), x, -3, 3),    key = "%c = 0",
    color = blue,
    explicit(subst(1, %c, s1), x, -3, 3),    key = "%c = 1",
    color = green,
    explicit(subst(2, %c, s1), x, -3, 3),    key = "%c = 2",
    color = orange,
    explicit(subst(3, %c, s1), x, -3, 3),    key = "%c = 3",
    color = violet,
    explicit(subst(4, %c, s1), x, -3, 3),    key = "%c = 4"
)$
```

Comment: The command `rhs(sol1)` extracts the right-hand side of the general expression for the solution sol1, defined by $y = ...$

Result: Click here for Live Demo



Maxima command:
```
eq2 : 'diff(y,x,2) - 3*'diff(y,x) + 2*y = 0;
sol2 : ode2(eq2, y, x);
```
Comment: Solving a second-order ODE.

Result: Click here for Live Demo

(% i4) eq2 : 'diff(y,x,2) - 3*'diff(y,x) + 2*y = 0;
sol2 : ode2(eq2, y, x);

$$\frac{d^2}{dx^2}y - 3\left(\frac{d}{dx}y\right) + 2y = 0 \tag{eq2}$$

$$y = \%e^{x}\%\mathrm{k2} + \%e^{2x}\%\mathrm{k1} \tag{sol2}$$

Maxima command:
```
eq3 : 'diff(y,x,2) + y = sin(x);
sol3 : ode2(eq3, y, x);
```
Comment: A second-order ODE with a nonzero right-hand side.

Result: Click here for Live Demo

(% i6) eq3 : 'diff(y,x,2) + y = sin(x);
sol3 : ode2(eq3, y, x);

$$\frac{d^2}{dx^2}y + y = \sin(x) \tag{eq3}$$

$$y = \%\mathrm{k1}\sin(x) - \frac{x\cos(x)}{2} + \%\mathrm{k2}\cos(x) \tag{sol3}$$

5 Particular solution of a first-order ODE

If an additional condition is given (the value of the function at a point, the value of its derivative at a point, ...), it is then possible to determine a unique function in the family of solutions of the first-order ODE that satisfies this condition. This function is the particular solution satisfying the given condition.

Principle. First, solve the differential equation in order to obtain its general solution, which contains one integration constant. Then apply the initial condition to this general solution in order to determine the value of this constant. This gives the particular solution satisfying the given initial condition.

5.1 Computing a particular solution with Maxima

Find the particular solution fp of the ODE $\frac{dy}{dx} + 2y = x$ satisfying $y(0) = 1$. (In this case, we say that we are solving an initial value problem.)

Maxima command:

```
eq : 'diff(y,x) + 2*y = x$
sol:ode2(eq,y,x);
define(f(x),rhs(sol));
const:solve(f(0)=1,%c);
define(fp(x),subst(rhs(const[1]),%c,f(x)));
expand(fp(x));
```

Comment: We define the ODE, solve it with ode2, and define the general solution f. The constant %c is found by solving an equation that takes the initial condition into account; then the particular solution fp is defined.

Result: Click here for Live Demo

(% i2) eq : 'diff(y,x) + 2*y = x;
sol:ode2(eq,y,x);

$$\frac{d}{dx}y + 2y = x \tag{eq}$$

$$y = \%e^{-(2x)}\left(\frac{\%e^{2x}\,(2x-1)}{4} + \%c\right) \tag{sol}$$

(% i3) define(f(x),rhs(sol));

$$f(x) := \%e^{-(2x)}\left(\frac{\%e^{2x}\,(2x-1)}{4} + \%c\right) \tag{\% o3}$$

(% i4) const:solve(f(0)=1,%c);

$$\left[\%c = \frac{5}{4}\right] \tag{const}$$

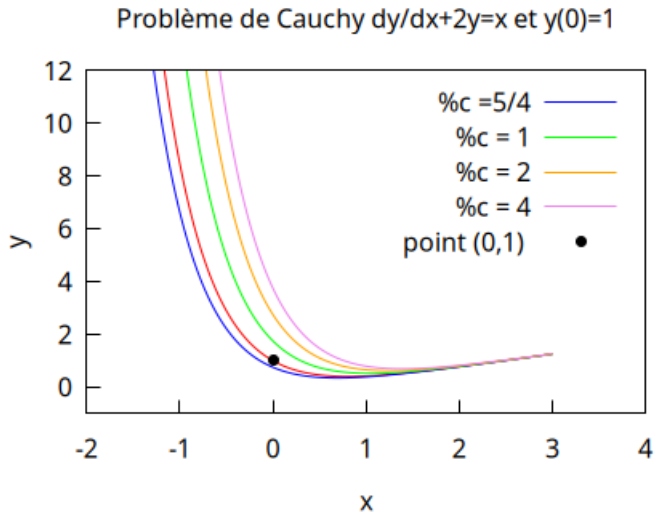
(% i5) define(fp(x),subst(rhs(const[1]),%c,f(x)));

$$fp(x) := \%e^{-(2x)}\left(\frac{\%e^{2x}\,(2x-1)}{4} + \frac{5}{4}\right) \tag{\% o5}$$

(% i6) expand(fp(x));

$$\frac{x}{2} + \frac{5\%e^{-(2x)}}{4} - \frac{1}{4} \tag{\% o6}$$

Let us represent this initial value problem graphically. To do so, let us plot several solution functions and the particular solution passing through the point with coordinates $(0, 1)$.
Result : Click here for Live Demo



5.2 Using Maxima's built-in ic1 command

In order for Maxima to compute directly the expression of the solution of the first-order differential equation satisfying $y(x_0) = y_0$, the general solution must first be found with `ode2`. Then the command `ic1` is used with the following syntax:

```
ic1(general_solution, x=x0, y=y0)
```

Maxima command:

```
eq2 : 'diff(y,x) - 3*y = 6*exp(x);y(0)=2;
solgen : ode2(eq2,y,x);
solpart : ic1(solgen,x=0,y=2);
```

Comment: Finding the particular solution of $\dfrac{dy}{dx} - 3y = 6e^x$ satisfying $y(0) = 2$.

Result: Click here for Live Demo

(% i29) eq2 : 'diff(y,x) - 3*y = 6*exp(x);y(0)=2;

$$\frac{d}{dx}y - 3y = 6\%e^x \tag{eq2}$$

$$y(0) = 2 \tag{\% o29}$$

(% i34) solgen : ode2(eq2,y,x);

$$y = \%e^{3x}\left(\%c - 3\%e^{-(2x)}\right) \tag{solgen}$$

(% i35) solpart : ic1(solgen,x=0,y=2);

$$y = 5\%e^{3x} - 3\%e^x \tag{solpart}$$

6 Particular solution of a second-order ODE

The general solution of a second-order ODE is defined by two parameters. If two conditions are imposed (values of the function at certain points or values of its derivative at certain points, ...), it is then possible to determine a unique function in the family of solutions of the second-order ODE that satisfies these two conditions.

Principle. First, solve the differential equation with `ode2` in order to obtain its general solution, which contains two integration constants. Then apply the initial conditions to this general solution in order to determine the values of these constants. This gives the particular solution satisfying the initial value problem for this type of ODE.

6.1 Computing a particular solution with Maxima

This method applies in all cases where the particular function is defined by two conditions, unlike the built-in commands, which apply only in certain cases.

Find the particular solution of $\frac{d^2y}{dx^2} + y = 0$ with $y(0) = 2$ and $y'\left(\frac{\pi}{4}\right) = 1$

Maxima command:

```
eq: 'diff(y,x,2)+y = 0;y(0)=2;yprime(%pi/4)=1;
sol_gen: ode2(eq, y, x);
define(f(x),rhs(sol_gen));
define(fprime(x),diff(f(x),x));
constantes:solve([f(0)=2,fprime(%pi/4)=1],[%k1,%k2]);
subst(constantes,f(x));
```

Comment: Defining f' as a function subsequently simplifies the solution of the system of equations used to determine the integration constants.

Result: Click here for Live Demo

(% i20) eq: 'diff(y, x, 2) + y = 0;y(0)=2;yprime(%pi/4)=1;
sol_gen: ode2(eq, y, x);

$$\frac{d^2}{dx^2}y + y = 0 \qquad \text{(eq)}$$

$$y(0) = 2 \qquad \text{(\% o18)}$$

$$yprime\left(\frac{\pi}{4}\right) = 1 \qquad \text{(\% o19)}$$

$$y = \%\text{k1}\sin(x) + \%\text{k2}\cos(x) \qquad \text{(sol\_ gen)}$$

(% i5) define(f(x),rhs(sol_gen));

$$f(x) := \%\text{k1}\sin(x) + \%\text{k2}\cos(x) \qquad \text{(\% o5)}$$

(% i6) define(fprime(x),diff(f(x),x));

$$fprime(x) := \%\text{k1}\cos(x) - \%\text{k2}\sin(x) \qquad \text{(\% o6)}$$

(% i9) constantes:solve([f(0)=2,fprime(%pi/4)=1],[%k1,%k2]);

$$\left[\left[\%\text{k1} = \sqrt{2} + 2\,,\ \%\text{k2} = 2\right]\right] \qquad \text{(constantes)}$$

(% i16) subst(constantes,f(x));

$$\left(\sqrt{2} + 2\right)\sin(x) + 2\cos(x) \qquad \text{(\% o16)}$$

6.2 Using Maxima's built-in ic2 command

The command `ic2` determines the particular solution when the two conditions give the values $y(x_0) = y_0$ and $y'(x_0) = yp_0$, both known at the same point x_0. The command syntax is:
`ic2(general_solution, x=x0, y=y0, 'diff(y,x)=yp0)`

Find the particular solution of $\frac{d^2y}{dx^2} + 3\frac{dy}{dx} + 2y = 0$ with $y(0) = 1$ and $y'(0) = 0$. In this case, there are two initial conditions.

Maxima command:
```
eq: 'diff(y,x,2) + 3*'diff(y,x) + 2*y = 0;
sol_gen: ode2(eq,y,x);
sol_part: ic2(sol_gen, x=0, y=1, 'diff(y,x)=0);
```

Result: Click here for Live Demo

(% i16) eq: 'diff(y, x, 2) + 3*'diff(y, x) + 2*y = 0;y(0)=1;yprime(0)=0;
sol_gen: ode2(eq, y, x);
sol_part: ic2(sol_gen, x=0, y=1, 'diff(y,x)=0);

$$\frac{d^2}{dx^2}y + 3\left(\frac{d}{dx}y\right) + 2y = 0 \qquad \text{(eq)}$$

$$y(0) = 1 \qquad (\%\ \text{o13})$$

$$yprime(0) = 0 \qquad (\%\ \text{o14})$$

$$y = \%e^{-(2x)}\%\text{k2} + \%e^{-x}\%\text{k1} \qquad \text{(sol\_ gen)}$$

$$y = 2\%e^{-x} - \%e^{-(2x)} \qquad \text{(sol\_ part)}$$

6.3 Using Maxima's built-in bc2 command

The command `bc2` solves a boundary value problem. In this case, the two conditions used to find the particular solution are given by the values of y at two points x_1 and x_2: $y(x_1) = y_1$ and $y(x_2) = y_2$. The command syntax is:
`bc2(solution, x=x1, y=y1, x=x2, y=y2)`

Find the particular solution of $x^2\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + 2y = x^3$ with $y(1) = 1$ and $y(2) = 4$.

Maxima command:
```
eq: x^2*'diff(y,x,2) - 2*x*'diff(y,x) + 2*y = x^3;
sol_gen: ode2(eq,y,x);
sol_bord: bc2(sol_gen, x=1, y=1, x=2, y=4);
```

Result: Click here for Live Demo

(% i29) eq: x^ 2*'diff(y, x, 2) - 2*x*'diff(y, x) + 2*y = x^ 3;y(1)=1;y(2)=4;
sol_gen: ode2(eq, y, x);
sol_bord: bc2(sol_gen, x=1, y=1, x=2, y=4);

$$x^2\left(\frac{d^2}{dx^2}y\right)-2x\left(\frac{d}{dx}y\right)+2y=x^3 \qquad \text{(eq)}$$

$$y(1)=1 \qquad \text{(\% o26)}$$

$$y(2)=4 \qquad \text{(\% o27)}$$

$$y=\frac{x^3}{2}+\%\text{k1}\,x^2+\%\text{k2}\,x \qquad \text{(sol\_ gen)}$$

$$y=\frac{x^3}{2}-\frac{x^2}{2}+x \qquad \text{(sol\_ bord)}$$

7 Study of some types of differential equations

7.1 Directly integrable differential equation: $y' = f(x)$

Solve the differential equation $y' = x^2 + \cos(x)$

Maxima command:

```
f(x):=x^2+cos(x);
eq: 'diff(y, x) = f(x);
sol_gen: ode2(eq,y,x);
y=integrate(f(x),x)+%c;
```

Comment: Define the function f and solve the equation with `ode2`. Another method consists of integrating the function f directly, without forgetting to add the integration constant.

Result: Click here for Live Demo

(% i1) f(x):=x^ 2+cos(x);

$$f(x):=x^2+\cos(x) \qquad \text{(\% o1)}$$

(% i3) eq: 'diff(y, x) = f(x);
sol_gen: ode2(eq, y, x);

$$\frac{d}{dx}y=\cos(x)+x^2 \qquad \text{(eq)}$$

$$y=\sin(x)+\frac{x^3}{3}+\%\text{c} \qquad \text{(sol\_ gen)}$$

(% i4) y=integrate(f(x),x)+%c;

$$y=\sin(x)+\frac{x^3}{3}+\%\text{c} \qquad \text{(\% o4)}$$

7.2 Autonomous differential equation $y' = f(y)$

In this case, the variable x does not appear explicitly in the equation. The equation is usually solved with `ode2`; then y is isolated by solving the resulting equation for y. It is often necessary to transform the result in order to obtain a simplified expression.

Solve the differential equation $y' = y(1-y)$

Maxima command:

```
eq : 'diff(y,x) = y*(1-y);
sol : ode2(eq, y, x);
sol_simp : logcontract(sol);
sol_exp  : exp(lhs(sol_simp)) = exp(rhs(sol_simp));
sol_y : solve(sol_exp, y);
```

Comment: Maxima must be guided through the required transformations. To do this, apply the exponential manually to both sides of the obtained solution, then solve using the `solve` command.

Result: Click here for Live Demo

(% i44)
```
eq : 'diff(y,x) = y*(1-y);
sol : ode2(eq, y, x);
sol_simp : logcontract(sol);
sol_exp  : exp(lhs(sol_simp)) = exp(rhs(sol_simp));
sol_y : solve(sol_exp, y);
```

$$\frac{d}{dx}y = (1-y)\ y \tag{eq}$$

$$\log(y) - \log(y-1) = x + \%c \tag{sol}$$

$$\log\left(\frac{y}{y-1}\right) = x + \%c \tag{sol\_ simp}$$

$$\frac{y}{y-1} = \%e^{x+\%c} \tag{sol\_ exp}$$

$$\left[y = \frac{\%e^{x+\%c}}{\%e^{x+\%c}-1}\right] \tag{sol\_ y}$$

This result can be transformed using properties of the exponential function by replacing $\%e^{x+\%c}$ with $K\%e^{x}$, where K is an arbitrary constant. Maxima must be guided manually to obtain this result, for example with the command `subst(K*%e^x,%e^(x+%c),sol_y);`

(% i10)
```
subst(K*%e^ x,%e^ (x+%c),sol_y);
```

$$\left[y = \frac{\%e^{x}K}{\%e^{x}K-1}\right] \tag{\% o10}$$

7.3 Separable differential equation: $y' = f(y)g(x)$

Two methods can be used for this type of differential equation:

1. *Method 1:* solve the equation with `ode2`, then express y as a function of x whenever possible.
2. *Method 2:* separate the variables (y on one side and x on the other), integrate each side, then try to express y as a function of x. In this case, Maxima often needs assistance because the standard commands are not suited to the situation.

When solving this type of differential equation, one often obtains an equality between an expression involving y and an expression involving x, but it is often impossible to isolate y in a simple way. Numerical methods can nevertheless be used to study the resulting solution.

Example 1: Solve the differential equation $y' = yx$.

Maxima command:
```
eq: 'diff(y,x) = y*x;
sol: ode2(eq, y, x);
```
Comment: The command `ode2` is used here.

Result: Click here for Live Demo

```
(% i2)  eq: 'diff(y,x) = y*x;
        sol: ode2(eq, y, x);
```

$$\frac{d}{dx}y = x\,y \tag{eq}$$

$$y = \%e^{\frac{x^2}{2}}\,\%c \tag{sol}$$

Maxima command:
```
eq/y;
sol1:integrate(1/y,y)=integrate(x,x)+K;
solve(sol1,y);
```
Comment: The equation is solved by separating the variables. The command `eq/y:` identifies the left- and right-hand sides to be integrated. The same result is obtained, with a different notation for the integration constant.

Result: Click here for Live Demo

```
(% i3)  eq/y;
```

$$\frac{\frac{d}{dx}y}{y} = x \tag{\% o3}$$

```
(% i4)  sol1:integrate(1/y,y)=integrate(x,x)+K;
```

$$\log(y) = \frac{x^2}{2} + K \tag{sol1}$$

```
(% i5)  solve(sol1,y);
```

$$\left[y = \%e^{\frac{x^2}{2}+K}\right] \tag{\% o5}$$

Example 2: Solve the differential equation $y' = \dfrac{x}{y^2+1}$

Maxima command:
```
eq: 'diff(y,x) = x/(1 + y^2);
sol: ode2(eq, y, x);
sol2:solve(sol,y)$sol2[3];
```
Comment: The result is a cubic equation in y^3. The `solve` command gives three roots, two of which are complex. The real root, which is the third one obtained, is selected.

Result: Click here for Live Demo

(% i7) eq: 'diff(y,x) = x/(1 + y^ 2);
sol: ode2(eq, y, x);

$$\frac{d}{dx}y = \frac{x}{y^2+1} \tag{eq}$$

$$\frac{y^3+3y}{3} = \frac{x^2}{2} + \%c \tag{sol}$$

(% i9) sol2:solve(sol,y)$sol2[3];

$$y = \left(\frac{\sqrt{9x^4+36\%cx^2+36\%c^2+16}}{4} + \frac{6\%c+3x^2}{4}\right)^{\frac{1}{3}} - \frac{1}{\left(\frac{\sqrt{9x^4+36\%cx^2+36\%c^2+16}}{4} + \frac{6\%c+3x^2}{4}\right)^{\frac{1}{3}}} \tag{\% o9}$$

Maxima command:
```
eq*(y^2+1);
sol3:integrate(y^2+1,y)=integrate(x,x)+K;
```
Comment: To separate the variables, transform the differential equation and integrate each side manually.

Result: Click here for Live Demo

(% i10) eq*(y^ 2+1);

$$\left(y^2+1\right)\left(\frac{d}{dx}y\right) = x \tag{\% o10}$$

(% i11) sol3:integrate(y^ 2+1,y)=integrate(x,x)+K;

$$\frac{y^3}{3} + y = \frac{x^2}{2} + K \tag{sol3}$$

7.4 Linear differential equation $y' = f(x)y + g(x)$

Solve the differential equation $y' = xy + x - 1$

Maxima command:
```
eq: 'diff(y,x) = x*y +x-1;
ode2(eq,y,x);
```
Comment: The solution is expressed in terms of the error function, which is mathematically defined by $\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,dt$

Result: Click here for Live Demo

(% i19) eq: 'diff(y,x) = x*y +x-1;
ode2(eq,y,x);

$$\frac{d}{dx}y = xy + x - 1 \tag{eq}$$

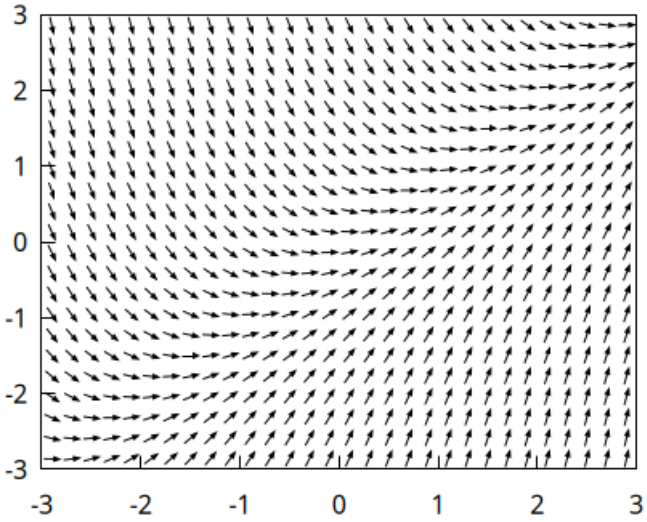
$$y = \%e^{\frac{x^2}{2}}\left(-\left(\frac{\sqrt{\pi}\,erf\left(\frac{x}{\sqrt{2}}\right)}{\sqrt{2}}\right) + \%c - \%e^{-\left(\frac{x^2}{2}\right)}\right) \tag{\% o19}$$

8 Direction field of a first-order ODE

For a differential equation $y' = f(x, y)$, the derivative y' represents, at the point (x, y), the slope of the tangent to the solution curve passing through that point. At each point (x, y) in the plane, a short segment of slope $f(x, y)$ is drawn. These segments indicate the direction that any solution passing through that point must follow. The graphical representation of these segments is the direction field of the ODE. Maxima has a built-in command to plot this direction field:
`load(drawdf)$` : load the dedicated package
`wxdrawdf(f(x,y),[x,-10,10],[y,-10,10]);` plots the field between -10 and 10 on each axis.

Maxima command:
```
load(drawdf)$
wxdrawdf(x - y, [x,-3,3], [y,-3,3])$
```
Comment: Representation of the direction field of $y' = x - y$

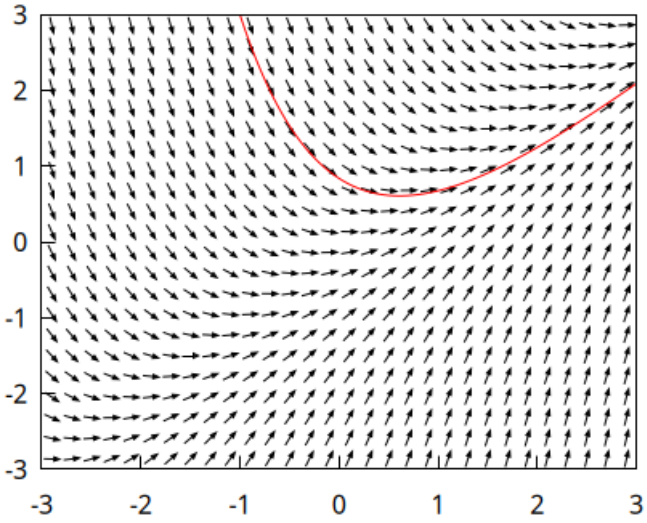
Result: Click here for Live Demo



A particular solution can easily be added to this direction field using the option `[trajectory_at,a,b]`, which plots the solution satisfying $y(a) = b$.

Maxima command:
```
load(drawdf)$
wxdrawdf(x-y,[x,-3,3],[y,-3,3],[trajectory_at,-1,3])$
```

Result: Click here for Live Demo



9 Numerical approximation of a particular solution of an ODE

9.1 Euler's method for a first-order ODE

Principle: for a first-order differential equation $y' = f(x, y)$, a numerical approximation of the solution defined by the initial condition (x_0, y_0) can be obtained.

$y'_0 = f(x_0, y_0)$ is the slope of this particular solution at the point with abscissa x_0. Knowing the starting point and the slope, the curve can therefore be approximated by a straight-line segment on the interval $[x_0, x_0 + \Delta x]$, where Δx is a small value. This gives

$$x_1 = x_0 + \Delta x \qquad y_1 = y_0 + y'_0 \times \Delta x$$

This procedure is then repeated by replacing (x_0, y_0) with (x_1, y_1), and so on. An algorithm is thus constructed that yields the value of y at a given point c. In practice, the interval $[x_0, c]$ is divided into n subintervals of width Δx. The accuracy of the approximation is directly related to the chosen number n and increases with it.

9.2 Approximating a value of the particular solution

Consider the particular solution of the differential equation $y' = x - y$ satisfying $y(0) = 1$. Let us find an approximation of $y(1)$ using a step size of $\frac{1}{100}$.

Maxima command:

```
/* Euler's method for y' = x - y, y(0) = 1 */
/* Parameters */
x0: 0$
y0: 1$
h: 1/100$
nsteps: 100$
/* Function f(x,y) = x - y */
f(x, y) := x - y$
/* Initialization */
x: x0$
y: y0$
/* Euler loop */
for i: 1 thru nsteps do (
    y: y + h * f(x, y),
    x: x + h
)$
/* Result: approximation of y(1) */
print("Approximation de y(1) par Euler :", float(y))$
```

Result: Click here for Live Demo

"Approximation de y(1) par Euler :"

0.732064682546459

Maxima command:

```
kill(x, y)$
eq: 'diff(y,x) = x - y;
sol_exacte: ode2(eq, y, x);
sol_ci: ic1(sol_exacte, x=0, y=1);
y_exact: rhs(solve(sol_ci, y)[1]);
print("Valeur exacte y(1) :", ev(y_exact, x=1, numer))$
```

Comment: This is compared with the exact solution. To do so, solve the equation with `ode2`, find the

. particular solution with `ic1`, and compute the value of this function at $x = 1$.

Result: Click here for Live Demo

```
(% i135) /* Comparison with the exact solution */
         kill(x, y)$
         eq: 'diff(y,x) = x - y;
         sol_exacte: ode2(eq, y, x);
         sol_ci: ic1(sol_exacte, x=0, y=1);
         y_exact: rhs(solve(sol_ci, y)[1]);
         print("Exact value y(1) :", ev(y_exact, x=1, numer))$
```

$$\frac{d}{dx}y = x - y \qquad \text{(eq)}$$

$$y = \%e^{-x}\left(\%e^{x}\left(x-1\right)+\%c\right) \qquad \text{(sol\_ exacte)}$$

$$y = \%e^{-x}\left(\%e^{x}x-\%e^{x}+2\right) \qquad \text{(sol\_ ci)}$$

$$\%e^{-x}\left(\%e^{x}x-\%e^{x}+2\right) \qquad \text{(y\_ exact)}$$

"Exact value y(1) :"

$$0.7357588823428847$$

9.3 Plotting a curve approximating a particular solution

Principle: to plot, on a given interval, a curve approximating a particular solution of a first-order ODE by Euler's method, the interval is subdivided into n parts, and an approximate value of the particular solution y is computed at each point of these subintervals using the previous method. It then remains to place these points on a graph and join them.

Plot a curve approximating the particular solution on $[0, 2\pi]$ of the differential equation

$$y' = \cos(x) + y \quad \text{with } y(0) = 1$$

The interval is divided into 100 subintervals.

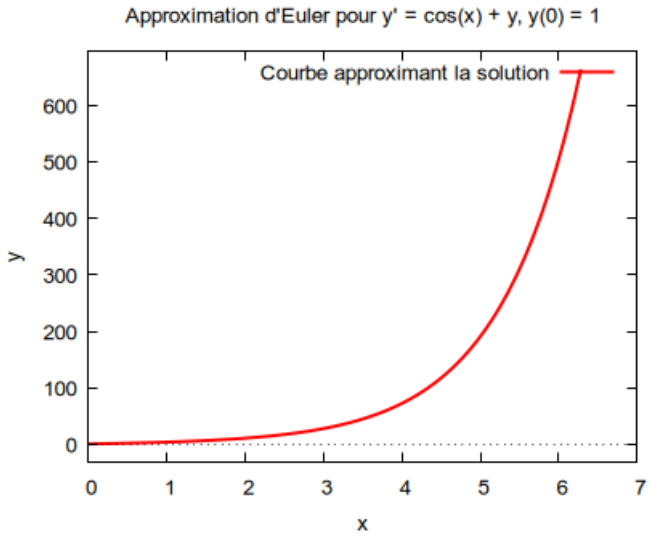
Maxima command:

```
x0: 0.0$y0: 1.0$x_final: 2.0*%pi$nsteps: 100$
h: float((x_final - x0) / nsteps)$
f(x, y) := float(cos(x) + y)$
x: x0$y: y0$
/* Lists to store the points */
X_list: [x]$
Y_list: [y]$
/* Euler loop */
for i: 1 thru nsteps do (
    y: y + h * f(x, y),    x: x + h,
    X_list: endcons(x, X_list),
    Y_list: endcons(y, Y_list)
)$
wxplot2d([discrete, X_list, Y_list],
    [style, [lines, 2, red]],
    [xlabel, "x"],
    [ylabel, "y"],
```

```
    [title, "Euler approximation for y' = cos(x) + y, y(0) = 1"],
    [legend, "Curve approximating the solution"]
)$
```

Comment: To reduce Maxima's computational load, approximate values of the computed points are used via the `float` command, because exact arithmetic is very slow.

Result: Click here for Live Demo



9.4 Approximating a solution with the built-in rk command

For a first-order ODE, the built-in Maxima command `rk` applies the classical fourth-order Runge–Kutta method to compute successive numerical approximations of the solution on a given interval. This interval is divided into n subintervals, on each of which four slope calculations are performed. This method is therefore more accurate than Euler's method presented previously. The syntax of this command is:
`rk(equation, function, initial_value, [variable, start, end, step]);`

Find an approximation of the solution of $y' = x - y$ with $y(0) = 1$ on $[0, 1]$ with a step size of 0.01.

Maxima command:

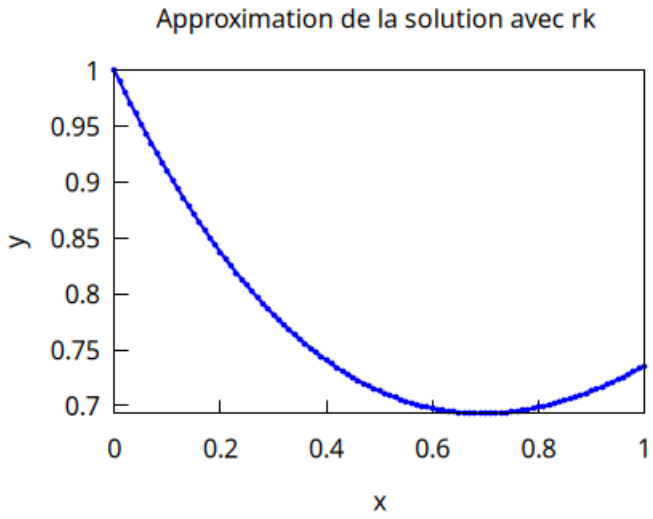
```
rk( x - y,  y, 1,[x, 0, 1, 0.4]);
listepoints:rk( x - y,  y, 1,[x, 0, 1, 0.01])$
wxdraw2d(
    title = "Approximation de la solution avec rk ",
    xlabel = "x",
    ylabel = "y",
    color = blue,
    point_type = filled_circle,
    point_size = 0.3,
    line_width = 2,
    points_joined = true,
    points(listepoints)
)$
```

Comment: `rk` returns a list of points. The first command illustrates the list obtained when the step size is 0.25. The list of points is then generated for the desired subdivision, and the points are plotted and joined to obtain the curve approximating the particular solution of the differential equation.

Result: Click here for Live Demo

(% i19) rk(x - y, y, 1,[x, 0, 1, 0.4]);

$$[[0.0\,,1.0]\,,[0.4\,,0.7408]\,,[0.8\,,0.69887232]\,,[1.0\,,0.7359367307946667]] \qquad (\%\ o19)$$



10 The built-in desolve command

There is another native Maxima command for solving differential equations: `desolve`, which uses the Laplace transform for this purpose. This command is more general than `ode2` because it can solve systems of differential equations. Regarding the syntax of this command, the unknown functions must be written in functional form ($y(x)$ instead of y).
`desolve([eq1, eq2, ...], [y(x), z(x), ...]);`

Example 1: Solve $y' + y = e^{-x}$ with $y(0) = 2$.

Maxima command:
```
de: 'diff(y(x), x) + y(x) = exp(-x);
sol: desolve(de, y(x));
sol_ci: ev(sol, y(0) = 2);
```
Comment: The general solution is expressed in terms of the initial value at 0. If the particular solution is defined by this initial value at 0, it can be obtained directly by replacing $y(0)$ with its value. Otherwise, $y(0)$ is treated as a constant to be determined from the given condition, as in Chapter 5.

Result: Click here for Live Demo

```
(% i11)  de: 'diff(y(x), x) + y(x) = exp(-x);
         sol: desolve(de, y(x));
         sol_ci: ev(sol, y(0) = 2);
```

$$\frac{d}{dx}\,y(x) + y(x) = \%e^{-x} \tag{de}$$

$$y(x) = \%e^{-x}x + \%e^{-x}\,y(0) \tag{sol}$$

$$y(x) = \%e^{-x}x + 2\%e^{-x} \tag{sol\_ ci}$$

Example 2: Solve $y'' - y = e^{x}$ with $y(0) = 1$ and $y'(0) = 0$.
Here, the `atvalue` command is used to define the initial values that will be used by `desolve` to find the particular solution.
`atvalue(y(x), x=0, 1);` and `atvalue('diff(y(x),x), x=0, 0);`

Maxima command:

```
atvalue(y(x), x = 0, 1)$
atvalue('diff(y(x), x), x = 0, 0)$
de2: 'diff(y(x), x, 2) - y(x) = exp(x);
sol2: desolve(de2, y(x));
```

Comment: The particular solution is obtained directly.

Result: Click here for Live Demo

```
(% i50) atvalue(y(x), x = 0, 1)$
        atvalue('diff(y(x), x), x = 0, 0)$
        de2: 'diff(y(x), x, 2) - y(x) = exp(x);
        sol2: desolve(de2, y(x));
```

$$\frac{d^2}{dx^2}\,y(x)-y(x)=\%e^{x} \tag{de2}$$

$$y(x)=\frac{\%e^{x}\,x}{2}+\frac{\%e^{x}}{4}+\frac{3\%e^{-x}}{4} \tag{sol2}$$

Example 3:
Solve the system of differential equations:

$$\begin{cases}\dfrac{dx}{dt}=x+y\\[2ex]\dfrac{dy}{dt}=4x-2y\end{cases}$$

with initial conditions $x(0)=1$ and $y(0)=0$.

Maxima command:

```
atvalue(x(t), t=0, 1)$
atvalue(y(t), t=0, 0)$
eq1: 'diff(x(t),t) = x(t) + y(t);
eq2: 'diff(y(t),t) = 4*x(t) - 2*y(t);
sol3: desolve([eq1,eq2], [x(t),y(t)]);
```

Comment: in this example, x and y denote functions, while the independent variable is t.

Result: Click here for Live Demo

```
(% i79) atvalue(x(t), t = 0, 1)$
        atvalue(y(t), t = 0, 0)$
        eq1: 'diff(x(t), t) = x(t) + y(t);
        eq2: 'diff(y(t), t) = 4*x(t) - 2*y(t);
        sol3: desolve([eq1, eq2], [x(t), y(t)]);
```

$$\frac{d}{dt}\,x(t)=y(t)+x(t) \tag{eq1}$$

$$\frac{d}{dt}\,y(t)=4\,x(t)-2\,y(t) \tag{eq2}$$

$$\left[x(t)=\frac{4\%e^{2t}}{5}+\frac{\%e^{-(3t)}}{5}\,,\,y(t)=\frac{4\%e^{2t}}{5}-\frac{4\%e^{-(3t)}}{5}\right] \tag{sol3}$$

11 Going further: the contrib_ode package

This package extends the capabilities of `ode2` for certain ODEs that `ode2` cannot solve, in particular some nonlinear first-order equations and some homogeneous linear second-order equations with variable coefficients, which may have several families of solutions. For first-order ODEs, `contrib_ode` first calls `ode2`, then tries other methods (factorization, Clairaut equations, Lagrange equations, Riccati equations, Abel equations, ...).
The essential commands of this package are:
`load(contrib_ode);` load the package
`contrib_ode(equation, function, variable);` attempts to solve the differential equation
`ode_check(equation, candidate_solution);` tests whether the function "candidate_solution" satisfies the differential equation or not.

Example 1: Solve $xy'' - (1+xy)y' + y = 0$

Maxima command:
```
load(contrib_ode)$
eq: x*'diff(y,x)^2 - (1+x*y)*'diff(y,x) + y = 0;
ode2(eq,y,x);
contrib_ode(eq,y,x);
ode_check(eq,[y=log(x)]);
```
Comment: First try `ode2`, which fails to solve the differential equation, then use `contrib_ode` for comparison. The last command tests whether the function log is indeed a solution; this is the case if the result is 0.

Result: Click here for Live Demo

```
(% i29)  eq: x*'diff(y,x)^ 2 - (1 + x*y)*'diff(y,x) + y = 0;
         ode2(eq,y,x);
         contrib_ode(eq,y,x);
         ode_check(eq,[y=log(x)]);
```

$$x\left(\frac{d}{dx}y\right)^2-(xy+1)\left(\frac{d}{dx}y\right)+y=0 \qquad \text{(eq)}$$

$$x\left(\frac{d}{dx}y\right)^2-(xy+1)\left(\frac{d}{dx}y\right)+y=0 \qquad (\%\ \text{t27})$$

"first order equation not linear in y'"

$$false \qquad (\%\ \text{o27})$$

$$x\left(\frac{d}{dx}y\right)^2-(xy+1)\left(\frac{d}{dx}y\right)+y=0 \qquad (\%\ \text{t28})$$

"first order equation not linear in y'"

$$[y=\log(x)+\%\text{c}\,,\ y=\%e^{x}\%\text{c}] \qquad (\%\ \text{o28})$$

$$0 \qquad (\%\ \text{o29})$$

Example 2: Solve $y' = y^2 - 2xy + x^2 + 1$

Maxima command:

```
eq2: 'diff(y,x) = y^2 - 2*x*y + x^2 + 1;
contrib_ode(eq2, y, x);
```

Comment: This is a Riccati equation.

Result: Click here for Live Demo

(% i34) eq2: 'diff(y,x) = y^ 2 - 2*x*y + x^ 2 + 1;
contrib_ode(eq2, y, x);

$$\frac{d}{dx}y = y^2 - 2xy + x^2 + 1 \tag{eq2}$$

$$\left[\left[x = \frac{1}{\sqrt{\%t - 1}} + \%c\,, y = x - \sqrt{\%t - 1}\right], \left[x = \%c - \frac{1}{\sqrt{\%t - 1}}\,, y = x + \sqrt{\%t - 1}\right]\right] \tag{\% o34}$$

This package also includes the command `odelin`, which solves certain homogeneous linear second-order equations with variable coefficients. The syntax is:
`odelin(equation, function, variable);`

Example 3: Solve $(x^2+1)y'' + xy' - y = 0$

Maxima command:

```
eq3: (x^2 + 1)*'diff(y,x,2) + x*'diff(y,x) - y = 0;
sol3: odelin(eq3, y, x);
```

Comment: `odelin` returns a list of functions forming a basis of the solution space. The general solution is therefore generated by a linear combination of the functions returned by `odelin`.

Result: Click here for Live Demo

(% i9) eq3: (x^ 2 + 1)*'diff(y,x,2) + x*'diff(y,x) - y = 0;
sol3: odelin(eq3, y, x);

$$(x^2+1)\left(\frac{d^2}{dx^2}y\right) + x\left(\frac{d}{dx}y\right) - y = 0 \tag{eq3}$$

$$\left\{x\,, \sqrt{x^2+1}\right\} \tag{sol3}$$

Maxima command:

```
sol4:listify(sol3);
y=%k1*sol4[1]+%k2*sol4[2];
```

Comment: The result, which is a set, is transformed into a list using the `listify` command, then the general solution of the differential equation is generated.

Result: Click here for Live Demo

(% i13) sol4:listify(sol3);

$$\left[x\,, \sqrt{x^2+1}\right] \tag{sol4}$$

(% i14) y=%k1*sol4[1]+%k2*sol4[2];

$$y = \%k2\sqrt{x^2+1} + \%k1x \tag{\% o14}$$

For any remarks or comments on this document, you can send an email to michel.gosse@free.fr.

Index