# Algorithms for Computing Exact Solutions for Ordinary and Partial Differential Equations

**Edgardo S. Cheb-Terrab**
**MITACS-CECM, Mathematics & Statistics, Simon Fraser University**
**Theoretical Physics Department, State University of Rio de Janeiro, Brazil**

This project aims at elaborating new algorithms for finding *Exact Solutions* to ODEs, PDEs and systems of them. The project includes implementing the algorithms in a Computer Algebra System (CAS) - we choose Maple. This work was supported by the MITACS NCE of Canada.

The determination of Exact solutions to differential equations is a challenge. Few algorithms are known, and they work mostly for linear equations; for the non-linear case, the methods based on finding point symmetries or restricted forms of integration factors cover a quite small portion of the problem (albeit, this portion is the one we see being used in modelling, most probably because the related differential equations are actually the only ones we know how to solve exactly). The main motivation here is that, having more general *Exact Solution* algorithms and implemented in a user-friendly CAS environment, will remove some of the present barriers researchers have when modelling real and theoretical problems.

Concerning the techniques involved in this project, the main ones are: symmetries, integrating factors, formulation of equivalence problems for the non-linear case and determination of (as general as possible) hypergeometric solutions for the linear case. All these methods are considered in the framework of both ODEs and PDEs. The related papers - some already published and some under consideration in Journals are [1,2,3,4].

The three sections below contain a couple of examples per section, related to the algorithms - already implemented - for computing:

- *Exact solutions to Partial Differential Equation systems* (program already integrated to the PDE solver of the upcoming version of Maple);
- *Exact solutions to Abel type ODEs*, unifying into a single approach the different algorithms found scattered in the literature, as well as turning new Abel ODE classes fully solvable (program already integrated into Maple 6 - released Feb/2000);
- *Exact solutions for a general class of Non-Linear First order ODEs*, based on performing linear transformations (program already integrated to the ODE-solver of the upcomming version of Maple);

**References**

[1]. E.S. Cheb-Terrab and T. Kolokolnikov, *"First order ODEs, Symmetries and Linear Transformations"*; submitted (Jul/2000) to European Journal of Applied Mathematics.

[2]. E.S. Cheb-Terrab and A.D. Roche, *"An Abel ODE class generalizing known integrable classes".* Accepted for publication (Apr/2000) to European Journal of Applied Mathematics.

[3]. E.S. Cheb-Terrab and A.D. Roche, *"Abel Equations: Equivalence and New Integrable Cases"*, Computer Physics Communications, 130 p.197 (2000).

[4]. E.S. Cheb-Terrab and A.D. Roche, *"Integrating Factors for Second Order ODEs"*, Journal of Symbolic Computation V. 27, No. 5, pp. 501-519
(1999).

# Exact solutions to Partial Differential Equation systems

Systems of partial differential equations usually arise in the modelling of phenomena in quite varied areas. The solving of these systems is a major problem. Typically, the exact solving of a PDE system - linear or non-linear - requires the solving of systems of ODEs, scalar non-trivial first or higher order PDEs and ODEs, as well as (maybe non-linear) systems of algebraic equations depending on none or many parameters. All of these intermediate "simpler" problems are in turn major problems, for which the research of related exact solution algorithms is the main activity of various groups around the world. It is then understandable why Computer Algebra Systems are not distributed with such a computational facility.

On the other hand, as a matter of fact, in various cases the exact solving of PDE systems can be mapped into the solving of subfamilies of problems which turn out to be solvable with the present CAS facilities; as example of these solvable "simpler" problems consider some types of linear ODE systems, varied types of non-linear ODEs or first order PDEs, or even higher order linear PDEs. In addition, the 6th version of the Maple CAS is already distributed with programs for performing differential elimination - an essential tool for "triangularizing" (equivalent to uncoupling) differential equations (DE) systems. Also the ODE and PDE solvers of Maple 6 are considered nowadays mature programs in that they have already incorporated an enormous amount of DE solving algorithms typically scattered in the literature.

Bearing this in mind, this project aimed at preparing a program for determining ***Exact Solutions*** for PDE systems, relying on the capabilities of Maple for solving the intermediate "simpler" problems mentioned on top. The strategy implemented consists of - mainly - two big steps, which can be understood with ease, for instance, considering the case of a *PDE system with just two unknowns* $\xi$ and $\eta$:

```
> restart; with(ODEtools): with(PDEtools): declare( f(x,y,z,t),
  (xi,eta)(x,y), quiet ):
> sys:=[diff(xi(x,y),`$`(y,2))=0,diff(eta(x,y),`$`(y,2))-2*diff(xi(x,y)
  ,x,y)=0,-3*(x+y)*diff(xi(x,y),y)+2*diff(eta(x,y),x,y)-diff(xi(x,y),`$
  `(x,2))=0,-eta(x,y)-2*(x+y)*diff(xi(x,y),x)+(x+y)*diff(eta(x,y),y)-xi
  (x,y)+diff(eta(x,y),`$`(x,2)) = 0]:
> for _eq in sys do _eq; od;
```

$$\xi_{y, y} = 0$$

$$\eta_{y, y} - 2\, \xi_{x, y} = 0$$

$$-3\, (x+y)\, \xi_y + 2\, \eta_{x, y} - \xi_{x, x} = 0$$

$$-\eta - 2\, (x+y)\, \xi_x + (x+y)\, \eta_y - \xi + \eta_{x, x} = 0$$

The two steps mentioned are related to:

1. *Assume that the system above could be rewritten as two other PDE systems: one involving* ***only*** $\xi$, *and another one involving* ***both*** $\{\xi, \eta\}$ ***or*** *just* $\eta$.

2. *Assume now that we could solve the PDE system involving* ***only*** $\xi$.

If both 1. and 2. were true, our strategy would be: First solve the sub-system involving only one unknown (here $\xi$), then substitute this value of $\xi$ into the other sub-system for $\{\xi, \eta\}$, which becomes a system in one unknown again (now $\eta$). Since we assumed we can solve systems in one unknown, the original problem has been solved. The scheme applies straightforwardly to the case of more than two variables, so all amounts to "how could one perform these steps 1. and 2".

*Concerning step 1.*

When the system is polynomial in the unknown functions - linear or non-linear case - there is theory for assuring whether or not a solution exists, and, in the positive case, actually splitting the system in the way just explained. The corresponding Maple packages are mainly ***diffalg*** and ***RIF*** - the latter being one of the MITACS contributions by Allan Wittkopf and Greg Reid. These packages, however, do not tell what would be the convenient *"**ordering**"* (do we solve first for $\xi$ or for $\eta$?). Moreover, there is no theory for telling that ***"for sure this ordering is better than other ones"***. For determining this ordering, we use a deterministic technique, which however is heuristic in that other orderings may be better.

**Examples**

For the given system

```
> sys;
```

$$[\xi_{y,y}=0, \eta_{y,y}-2\,\xi_{x,y}=0, -3\,(x+y)\,\xi_y+2\,\eta_{x,y}-\xi_{x,x}=0, -\eta-2\,(x+y)\,\xi_x+(x+y)\,\eta_y-\xi+\eta_{x,x}=0]$$

below casesplit uses diffalg to obtain the two PDE susbsystems mentioned:

```
> casesplit(sys):
> sys2 := selectremove(u -> has(u,eta),op(1,%));
```

$$sys2 := \left[\eta_{x,x}=\eta+2\,\xi_x\,x+2\,\xi_x\,y-\eta_y\,x-\eta_y\,y+\xi, \eta_{x,y}=\frac{3}{2}\,\xi_y\,x+\frac{3}{2}\,\xi_y\,y+\frac{1}{2}\,\xi_{x,x}, \eta_{y,y}=2\,\xi_{x,y}\right],$$
$$[\xi_{x,x,x}=-3\,\xi_{x,y}\,x-3\,\xi_{x,y}\,y+4\,\xi_x-\xi_y, \xi_{x,x,y}=\xi_y, \xi_{y,y}=0]$$

A detail here is also that these packages do not accept input when the system is non-polynomial **-** for instance when it involves trigonometric or exponential functions, or fractional powers etc. For that case Maple already incorporated ***casesplit*** (developed under MITACS at CECM during 1999), which extends the ***diffalg*** and ***RIF*** domains of computation to include the usual transcendental functions and some others (e.g. hypergeometric functions).

*Concerning step 2.*

For the solving of a PDE system in one unknown, consider for instance the subsystem for $\xi$

```
> for _eq in sys2[2] do _eq; od;
```

$$\xi_{x,x,x}=-3\,\xi_{x,y}\,x-3\,\xi_{x,y}\,y+4\,\xi_x-\xi_y$$
$$\xi_{x,x,y}=\xi_y$$
$$\xi_{y,y}=0$$

Suppose we first solve the PDE which looks like simplest of the ones above:

```
> _eq;
```

$$\xi_{y,y}=0$$

```
> a1 := pdsolve(_eq);
```

$$a1 := \xi=\_F1(x)\,y+\_F2(x)$$

The problem is easy because the two arbitrary functions *_F1* and *_F2* actually depend on the original symbol variables (x is here a differential invariant). So in this case if we just substitute this value of $\xi$ in the other two PDEs of the system, the problem transforms into an ODE system for *_F1* and *_F2*:

```
> declare((_F1,_F2)(x), prime=x );
```

*derivatives with respect to: x of functions of one variable will now be displayed with '*

*_F1(x) will now be displayed as _F1*

*_F2(x) will now be displayed as _F2*

```
> eval(sys2[2][1..2], a1):
  for _eq in % do _eq; od;
```

$$\_F1'''\,y+\_F2'''=-3\,\_F1'\,x+\_F1'\,y+4\,\_F2'-\_F1$$
$$\_F1''=\_F1$$

If we now solve the last equation for *_F1* and substitute into the first one

```
> dsolve(_eq);
```

$$\_F1 = \_C1\,\mathbf{e}^{(-x)} + \_C2\,\mathbf{e}^{x}$$

```
> eval(%%%,%);
```

$$(-\_C1\,\mathbf{e}^{(-x)} + \_C2\,\mathbf{e}^{x})\,y + \_F2''' = \\ -3\,(-\_C1\,\mathbf{e}^{(-x)} + \_C2\,\mathbf{e}^{x})\,x + (-\_C1\,\mathbf{e}^{(-x)} + \_C2\,\mathbf{e}^{x})\,y + 4\,\_F2' - \_C1\,\mathbf{e}^{(-x)} - \_C2\,\mathbf{e}^{x}$$

which is a single equation for *_F2*, solvable using the ODE solver

```
> dsolve(%);
```

$$\_F2 = x\,\_C1\,\mathbf{e}^{(-x)} + x\,\_C2\,\mathbf{e}^{x} - \frac{1}{2}\_C3\,\mathbf{e}^{(-2\,x)} + \frac{1}{2}\_C4\,\mathbf{e}^{(2\,x)} + \_C5$$

Once we determined *_F1* and *_F2* we have solved the system for our single unknown $\xi$.

The program for doing all this - actually both Steps 1 and 2 - is already incorporated into Maple's PDE solver and performs all these tasks automatically.
For example, for our sub-system for $\xi$ solved above in interactive manner, ***pdsolve*** returns

```
> sys2[2];
```

$$[\xi_{x,x,x} = -3\,\xi_{x,y}\,x - 3\,\xi_{x,y}\,y + 4\,\xi_x - \xi_y,\ \xi_{x,x,y} = \xi_y,\ \xi_{y,y} = 0]$$

```
> pdsolve(%);
```

$$\{\xi = y\,\_C4\,\mathbf{e}^{(-x)} + y\,\_C5\,\mathbf{e}^{x} + \mathbf{e}^{(-x)}\,\_C4\,x + \mathbf{e}^{x}\,\_C5\,x - \frac{1}{2}\_C1\,\mathbf{e}^{(-2\,x)} + \frac{1}{2}\_C2\,\mathbf{e}^{(2\,x)} + \_C3\}$$

And hence, for our original system in two variables

```
> for _eq in sys do _eq; od;
```

$$\xi_{y,y} = 0$$

$$\eta_{y,y} - 2\,\xi_{x,y} = 0$$

$$-3\,(x+y)\,\xi_y + 2\,\eta_{x,y} - \xi_{x,x} = 0$$

$$-\eta - 2\,(x+y)\,\xi_x + (x+y)\,\eta_y - \xi + \eta_{x,x} = 0$$

pdsolve returns

```
> pdsolve( sys );
```

$$\{\eta = \left(\frac{1}{2}x + \frac{1}{2} + \frac{1}{2}y\right)\_C1\,\mathbf{e}^{(-2\,x)} + \left(-\frac{1}{2} + \frac{1}{2}y + \frac{1}{2}x\right)\_C2\,\mathbf{e}^{(2\,x)} \\ + \left(\_C7 + \left(-y^2 + (-2\,x-1)\,y - \frac{1}{2} - x^2 - x\right)\_C4\right)\mathbf{e}^{(-x)} + \left(\_C6 + \left(y^2 + (2\,x-1)\,y - x + \frac{1}{2} + x^2\right)\_C5\right)\mathbf{e}^{x} \\ + (x+y)\,\_C8 - \_C3,$$

$$\xi = \mathbf{e}^{(-x)}\,\_C4\,x + y\,\_C4\,\mathbf{e}^{(-x)} - \frac{1}{2}\_C1\,\mathbf{e}^{(-2\,x)} + \frac{1}{2}\_C2\,\mathbf{e}^{(2\,x)} + \mathbf{e}^{x}\,\_C5\,x + y\,\_C5\,\mathbf{e}^{x} + \_C3\}$$

The program has no restrictions to the number of independent variables (although depending on the problem and the computer being used, it may become too difficult to handle).

# Exact solutions to Abel type ODEs

From some point of view, after Riccati type equations, the simplest first order ordinary differential equations (ODEs) are those having as right hand side (RHS) a third degree polynomial in the dependent variable, also called *Abel type ODEs*:

$$y' = f_3\,y^3 + f_2\,y^2 + f_1\,y + f_0$$

where $\{f_1, f_2, f_3\}$ are arbitrary functions of $x$. Abel equations appear in the reduction of order process related to finding exact solutions to many second and higher order ODE families, and hence are frequently found in the modelling of real problems in varied areas. That has for a long time motivated people to study their integrable cases and related solving methods. The implementation of methods for Abel ODEs here presented appeared as a new feature in Maple 6 and follows what is shown in the paper *Abel Equations: Equivalence and New Integrable Classes*, by E.S. Cheb-Terrab and A.D. Roche, Computer Physics Communications, 130 p.197 (2000).

A general "exact integration" strategy for these ODEs was first formulated by Liouville, in the 19th century, and is based on the concepts of classes, invariants and the solving of the *equivalence* problem. Generally speaking, two Abel ODEs belong to the same equivalence class if and only if one can be obtained from the other by means of a transformation - say *TR* - of the form

$$\{x = \mathrm{F}(t),\ y = \mathrm{P}(t)\,\mathrm{u}(t) + \mathrm{Q}(t)\}$$

where $t$ and $\mathrm{u}(t)$ are respectively the new independent and dependent variables, and *F, P, Q* are arbitrary functions of $t$ satisfying

$$F' \neq 0,\ P \neq 0$$

Each class has infinitely many members and there are infinitely many classes. To each class there corresponds a different set of values of the *invariants* (built with the coefficients $\{f_1, f_2, f_3\}$and their derivatives), and actually any one of them (we shall pick one and call it the *invariant*) is enough to characterize a class. A simple integrable case happens when the invariant *is constant*; the solution to the ODE then follows straightforwardly in terms of quadratures, as explained in textbooks (this simple case of constant invariant Abel ODEs was implemented in Maple R5). But when the invariant is *not constant*, just a few integrable cases are known and the formulation of solving strategies is entirely based on the equivalence between two such Abel ODEs (one of which is integrable) under the transformation *TR*.

When a given Abel ODE belongs to one of a set of the solvable *classes* collected in the aforementioned CPC paper, the program (Maple's dsolve) first determines this fact, without solving any differential equations, and uses it to return a closed form solution without requiring further participation from the user.

Apart from bringing to the surface full parameterized new solvable Abel ODE classes, the ODE families that are covered include, as particular cases, all the Abel solvable cases presented in Kamke's and Murphy's books and various other references.

### Examples

As the first example we consider here the one presented in Kamke's book with number **185**

```
> restart; with(PDEtools):with(ODEtools): interface(imaginaryunit=i);
  declare(y(x), f(x), prime=x);
```

$\mathrm{f}(x)$, *will now be displayed as*, $f$

$\mathrm{y}(x)$, *will now be displayed as*, $y$

*derivatives with respect to:*, $x$, *of functions of one variable will now be displayed with* '

```
> ode[185] := x^7*diff(y(x),x)+2*(x^2+1)*y(x)^3+5*x^3*y(x)^2 = 0;
```

$$ode_{185} := x^7\, y' + 2\,(x^2+1)\, y^3 + 5\, x^3\, y^2 = 0$$

What the program does is to tackle the equivalence problem between this ODE and the known solvable Abel ODE classes, determining when a transformation of variables solving the problem exists, and, in the positive case, determining also the transformation of variables itself. In this case, ODE 185 belongs to a parameterized solvable class discovered by Abel in the 19th century:

$$Class_A := y' = -\frac{1\, C\,(x^2-1)\, y^3}{2\, x} + (C-1)\, y^2 - \frac{1\, C\, y}{2\, x}$$

where C is the parameter. In fact, for $C = -4$, the change of variables

$$\{x = \frac{i}{t}, y = \frac{i\,\mathrm{u}(t)}{t^2}\}$$

transforms the solvable class by Abel into Kamke's example 185. The program first notices that the conditions for the existence of a solution are fulfilled, then determines the value of C for which a solution (change of variables) exists and finally finds the change of variables above and uses it to construct the implicit exact solution:

```
> sol[185] := dsolve( ode[185], implicit, useInt);
```

$$sol_{185} := \_C1 + \frac{x}{\left(\left(\frac{1}{x}+\frac{x^2}{y}\right)^2+1\right)^{(1/4)}} + \frac{1}{2}\int^{\frac{1}{x}+\frac{x^2}{y}} \frac{1}{(\_a^2+1)^{(5/4)}}\, d\_a = 0$$

The integrals above can be evaluated in terms of hypergeometric functions.

A much more complicated example is given by the ODE presented in Kamke's book with number 43:

```
> ode[43] := diff(y(x),x)+(3*x^2*a+4*x*a^2+b)*y(x)^3+3*x*y(x)^2 = 0;
```

$$ode_{43} := y' + (3\,x^2\,a + 4\,x\,a^2 + b)\,y^3 + 3\,x\,y^2 = 0$$

This ODE belongs to a parameterized class discovered by Liouville. So, ODE 43 can be obtained from

$$Class_B := y' = 2\,(x^2 - C)\,y^3 + 2\,(x+1)\,y^2$$

by changing variables

$$\{x = -\frac{1\,(2\,a+3\,t)}{2\,a}, \mathrm{y}(x) = -\frac{2\,a^2\,\mathrm{u}(t)}{3}\}$$

and setting the parameter $C$ in $Class_B$ as *a rational function of the symbols a and b appearing in ODE 43:*

$$C = \frac{1\,(-3\,b + 4\,a^3)}{4\,a^3}$$

The difficulty resides in that a solution exists *only for this value of the parameter* $C$ and finding its value is highly non-trivial. But the functional dependence of $C$ in $a$ and $b$ is rational so that - computing recursively and interpolating - the program succeeds in finding this value as well as the transformation of variables shown above, finally using this information to build this implicit solution

```
> alpha=sqrt((y*b+4*y*x*a^2+3*y*x^2*a-2*a)/(a^3*y)), beta =
  sqrt((-3*b+4*a^3)/(a^3)); alias(%):
```

$$\alpha = \sqrt{\frac{y\,b + 4\,y\,x\,a^2 + 3\,y\,x^2\,a - 2\,a}{a^3\,y}}, \beta = \sqrt{\frac{-3\,b + 4\,a^3}{a^3}}$$

```
> dsolve( ode[43] );
```

$$\_C1 + \frac{-\left(\frac{1}{2}\beta - \frac{1}{2}\frac{2\,a+3\,x}{a}\right)\mathrm{K}\left(\frac{1}{2}\beta, -\frac{1}{2}\sqrt{3}\,\alpha\right) - \frac{1}{2}\,\mathrm{K}\left(\frac{1}{2}\beta+1, -\frac{1}{2}\sqrt{3}\,\alpha\right)\sqrt{3}\,\alpha}{-\left(\frac{1}{2}\beta - \frac{1}{2}\frac{2\,a+3\,x}{a}\right)\mathrm{I}\left(\frac{1}{2}\beta, -\frac{1}{2}\sqrt{3}\,\alpha\right) + \frac{1}{2}\,\mathrm{I}\left(\frac{1}{2}\beta+1, -\frac{1}{2}\sqrt{3}\,\alpha\right)\sqrt{3}\,\alpha}$$

where K and I are respectively the Bessel functions of first and second kind. The database of solvable classes implemented for the program distributed with Maple 6 has representatives for 25 classes including all those available in the main references as well as new ones presented in [2,3].

# Linear transformations for Non-Linear First order ODEs

The main motivation of this project consisted in:

*"To find a class of transformations such that the class forms a group, the related Equivalence problem for*

*first order ODEs is fully solvable, and the resulting algorithm covers a wide variety of typical ODEs arising in mathematical-physics"*

This project follows the presentation we made in the *"Rhine Workshop for Computer Algebra - Austria, 2000"*, as well as the paper presented in the European Journal of Applied Mathematics: E.S. Cheb-Terrab and T. Kolokolnikov, *"First order ODEs, Symmetries and Linear Transformations"*. The main idea here consists of developing an algorithm for first order ODEs, to determine both the existence and the symmetries themselves when they are of the form

$$[\xi = \mathrm{F}(x),\ \eta = \mathrm{P}(x)\, y + \mathrm{Q}(x)]$$

To this symmetry pattern one can associate an ODE class, which mainly embraces all first order ODEs which can be mapped into separable ones through linear transformations. This class includes as members 78 % of Kamke's 576 first order ODE examples and as such it represents the widest first order algorithmically solvable class we are aware of.

The computational strategy we used can be understood relatively easily by giving a look at the most general ODE class admitting such a symmetry. This class can be determined in Maple as follows. First, this is the general transformation defining the ODE class

$$\{x = \mathrm{F}(t),\ \mathrm{y}(x) = \mathrm{P}(t)\,\mathrm{u}(t) + \mathrm{Q}(t)\}$$

If we now depart from an autonomous ODE,

$$autonomous\_ODE := y' = \psi(y)$$

the general form of the ODE class we are looking for is obtained by directly changing variables in this autonomous ODE, arriving at

$$ODE\_class := y' = -\frac{P'\,y}{P} - \frac{Q' - \psi(P\,y + Q)\,F'}{P}$$

This is a non-linear family depending on four arbitrary functions: $\{F, P, Q, \psi\}$ and the non-linearity is due to the arbitrary function $\psi(P\,y + Q)$. An expression representing a closed form solution for *the whole ODE class* is obtained in the same way, by changing variables in the answer to the autonomous ODE

$$Solution\_of\_autonomous\_ODE := x - \int^{y} \frac{1}{\psi(\_a)}\, d\_a + \_C1 = 0$$

so that the general solution of the whole ODE class is

$$Solution\_of\_ODE\_class := F - \int^{P\,y+Q} \frac{1}{\psi(\_a)}\, d\_a + \_C1 = 0$$

The whole solving strategy, including determining whether or not a given ODE belongs to this class, is then based on an analysis of the form of the symmetry of this ODE class. This symmetry is obtained from the symmetry of an autonomous ODE

$$Symmetry\_of\_autonomous\_ODE := [\xi = 1,\ \eta = 0]$$

by change variables in the same way, leading to

$$Symmetry\_of\_ODE\_class := \left[\frac{1}{F'}, -\frac{P'\,y + Q'}{F'\,P}\right]$$

So that the whole problem can be formulated as:

*"To determine when a given first order ODE has a symmetry of the form above (with arbitrary F, P and Q), and when this symmetry exists, to determine { F, P, Q } to solve the equivalence problem, mapping the ODE into an autonomous one"*

The details of the algorithm for determining F, P and Q, when such a transformation exist, and so directly leading to the solution of the given ODE, are described in [1]. A single key observation to develope such a solving algorithm is that

*"The determination of just "P" up to a constant - say* $\kappa$ *- suffices to map the ODE into another ODE admitting a symmetry of the form* $[\xi = \mathrm{M}(x), \eta = \mathrm{N}(x)]$, *that is, both infinitesimals depending on just x, the independent variable. In turn the latter problem is fully solvable using a method we presented in 1998 (see [6])".*

So that the algorithm basically consists of a method for finding *P* up to a constant factor.

**Examples**

A case involving abstract powers and other symbols: Kamke's first order example number 128

```
> restart;with(ODEtools): with(PDEtools): declare(y(x),u(t),
  (F,P,Q)(x),prime=x,quiet):
> ode := x*diff(y(x),x)+a*y(x)-f(x)*g(x^a*y(x))=0;
```

$$ode := x\, y' + a\, y - \mathrm{f}(x)\, \mathrm{g}(x^a\, y) = 0$$

This ODE has symmetries of the form mentioned, which are determined by the new program as

```
> symgen(ode,way=patterns);
```

$$\left[\_\xi = \frac{x^{(-a+1)}}{\mathrm{f}(x)}, \_\eta = -\frac{x^{(-a)}\, a\, y}{\mathrm{f}(x)}\right]$$

Here the algorithm first checked that the existence conditions for a solution of the problem are satisfied, then proceeded with determining the symmetry above. This subroutine for determining the symmetry is used automatically by the ODE solver to finally construct the implicit exact solution

```
> dsolve(ode,[sym_pat], implicit);
```

$$\int x^{(a-1)}\, \mathrm{f}(x)\, dx - \int^{x^a\, y} \frac{1}{\mathrm{g}(\_a)}\, d\_a - \_C1 = 0$$

where the integrals could only be computed if both $\mathrm{f}(x)$ and $\mathrm{g}(x^a\, y)$ are given. In the example above, from the form of the symmetry we see that $Q = 0$. As an example where all of F, P and Q are different from zero consider

```
> ode := x*(y(x)^2+x^2*y(x)+x^2)*diff(y(x),x)-2*y(x)^3-2*x^2*y(x)^2+x^4
  = 0;
```

$$ode := x\, (y^2 + x^2\, y + x^2)\, y' - 2\, y^3 - 2\, x^2\, y^2 + x^4 = 0$$

This ODE admits the symmetry (of the form identifying the Class related to this work):

```
> collect(symgen(ode,way=patterns),y);
```

$$[\_\xi = -4\, x + x^3, \_\eta = (2\, x^2 - 4)\, y + 2\, x^2]$$

from where two general solutions - in this case algebraic and explicit - are given by

```
> dsolve(ode,[sym_pat]);
```

$$y = -\frac{1}{2} x^2 + \frac{1}{2} \_C1\, x^2 + \frac{1}{2} x \sqrt{-x^2 + \_C1^2\, x^2 + 4}, y = -\frac{1}{2} x^2 + \frac{1}{2} \_C1\, x^2 - \frac{1}{2} x \sqrt{-x^2 + \_C1^2\, x^2 + 4}$$