arXiv:gr-qc/9607037v1 16 Jul 1996

Universidade do Estado do Rio de Janeiro
Instituto de Física

IF-UERJ-27/96
Preprint
July 1996

Computer Algebra Solving of First Order ODEs Using Symmetry Methods

E.S. Cheb-Terrab[1], L.G.S. Duarte[2] and L.A.C.P. da Mota[1]

Abstract

A set of Maple V R.3/4 computer algebra routines for the analytical solving of 1^{st} order ODEs, using Lie group symmetry methods, is presented. The set of commands includes a 1^{st} order ODE-solver and routines for, among other things: the explicit determination of the coefficients of the infinitesimal symmetry generator; the construction of the most general invariant 1^{st} order ODE under given symmetries; the determination of the canonical coordinates of the underlying invariant group; and the testing of the returned results.

(Submitted to Computer Physics Communications)

[1]Symbolic Computation Group, Departamento de Física Teórica, IF-UERJ.
Available as http://dft.if.uerj.br/preprint/e6-27.tex

PROGRAM SUMMARY

Title of the software package: First order ODE tools.

Catalogue number: (supplied by Elsevier)

Software obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Licensing provisions: none

Operating systems under which the program has been tested: UNIX systems, Macintosh, DOS (AT 386, 486 and Pentium based) systems, DEC VMS, IBM CMS.

Programming language used: **Maple V** Release 3/4.

Memory required to execute with typical data: 8 Megabytes.

No. of lines in distributed program, including On-Line Help, etc.: 4183.

Keywords: 1^{st} order ordinary differential equations, symmetry methods, symbolic computation.

Nature of mathematical problem
Analytical solving of 1^{st} order ordinary differential equations.

Methods of solution
Computer algebra implementation of Lie group symmetry methods.

Restrictions concerning the complexity of the problem
Besides the inherent restrictions of the method (there is as yet no general scheme for solving the associated PDE for the coefficients of the infinitesimal symmetry generator), the present implementation does not work with systems of ODEs nor with higher order ODEs.

Typical running time
This depends strongly on the ODE to be solved, usually taking from a few seconds to a few minutes. In the tests we ran (with 466 1^{st} order ODEs from Kamke's book [5], see sec.4), the *average times* were: 6 sec. for a solved ODE and 20 sec. for an unsolved one, using a Pentium 133 with 64 Mb. RAM on a *Windows* 3.11 platform.

Unusual features of the program
The 1^{st} order ODE-solver here presented is an implementation of *all* the steps of the symmetry method solving scheme; i.e., when successful it returns a closed solution, not the symmetry generator. Also, this solver permits the user to (optionally) participate in the solving process by giving an advice (HINT option) concerning the *functional form* for the coefficients of the infinitesimal symmetry generator (infinitesimals). All the intermediate steps of the symmetry method solving scheme are available as user-level commands too. For instance, using the package's commands, it is possible to obtain the infinitesimals, the related canonical coordinates, and the most general 1^{st} order ODE invariant under a symmetry group. The package also includes a command for classifying ODEs (according to Kamke's' book[5]) popping up Help pages with Kamke's advice for solving them, facilitating the study of a given ODE and the use of the package with pedagogical purposes.

LONG WRITE-UP

Introduction

The automatic computation of symbolic solutions for ODEs has already been implemented in almost all symbolic computation systems available. However, though Lie's discovery of symmetry methods (SM) put most of the known solving methods under a common umbrella[1, 2], only a few of the computer algebra standard solvers use SM in their solving schemes[3], and almost none use them for solving 1^{st} order ODEs.

Generally speaking, one of the reasons for this is that the use of SM requires the solving of a system of partial differential equations (PDEs) for the coefficients of the infinitesimal symmetry generators (infinitesimals), whose solution is not obvious, thus involving the programming of semi-systematic *heuristic* methods for obtaining it[3]. Also, though in the specific case of 1^{st} order ODEs we just need to solve a single 1^{st} order PDE, one may think that SM are useless anyway[4], since to *guess* a solution for that PDE would not be simpler than to *guess* an equivalent integrating factor for the original ODE. Furthermore, there are many well known methods, based on a previous *classification* of the ODE, which already give good results without using heuristics[5]. So what would be the advantage of introducing heuristics in the solving process?

There are however two strong arguments in favor of heuristic methods: they are the only chance of solving *non-classifiable* ODEs, and they are the main way to find the underlying logic structures with which one could develop new methods.

Bearing all this in mind, we programmed an 1^{st} order ODE-solver based on SM, implementing a set of heuristic methods for solving the PDE which leads to the infinitesimals. In addition, in order to test ODE-solvers in general, we prepared a file with the Maple input of the 576 ODEs[2] of Kamke's book, and another one with 424 ODES, mainly of Riccati, Abel and non-classifiable types.

The set of user-level routines here presented also allows the user to obtain almost all the intermediate results involved in the SM scheme, to find the most general invariant ODE under a given symmetry, to test explicit/implicit results obtained by any (Maple) ODE-solver, and to classify ODEs, according to Kamke's book, including the optional pop up of Help pages with Kamke's *advice* about how to solve them.

The goals of such a work can be summarized as:

1. to build an ODE-solver complementing the standard **dsolve**, mainly for tackling non-classifiable ODEs,

2. to build a research environment for discussing possible connections between patterns of infinitesimals and patterns of ODEs, mainly using the HINT option of the solver (see sec.2) in connection with the *advisor* (**odeadv**) command,

3. to build a friendly educational environment for studying ODEs and their solving methods in general.

The exposition is organized as follows. In sec.1, the SM scheme for solving ODEs is briefly reviewed and discussed. In sec.2 the package of routines is presented. Sec.3 contains a more detailed explanation of each of the heuristic methods implemented. Sec.4 is devoted to testing the package, mainly the solver, against the aforementioned set of 1,000 ODEs. These tests allow an evaluation of the solver from different angles, and include a comparison of the performances of the new solver and the Maple **dsolve** in solving Kamke's ODEs. Finally, the conclusions contain a brief discussion about this work and its possible extensions.

1 Symmetry methods for 1^{st} order ODEs

Generally speaking, the key point of Lie's method for solving ODEs is that the knowledge of a (Lie) group of transformations which leaves a given ODE invariant may help in reducing the problem of finding its solution to quadratures[1, 2]. Despite the subtleties which arise when considering different cases, we can summarize the computational task of using SM for solving a given ODE, say,

[2] From here on, we will use the abbreviate ODE for representing 1^{st} order ODEs.

$$\frac{dy}{dx} = \Phi(y, x), \tag{1}$$

as *the finding of the infinitesimals* of a one-parameter Lie group which leaves Eq.(1) invariant, i.e., a pair of functions $\{\xi(y,x),\ \eta(y,x)\}$ satisfying

$$\frac{\partial \eta}{\partial x} + \left(\frac{\partial \eta}{\partial y} - \frac{\partial \xi}{\partial x}\right) \Phi - \frac{\partial \xi}{\partial y} \Phi^2 - \xi \frac{\partial \Phi}{\partial x} - \eta \frac{\partial \Phi}{\partial y} = 0 \tag{2}$$

followed by either

a) the determination of the canonical coordinates, say $\{r, s\}$, of the associated Lie group, to be used in a change of variables which will reduce Eq.(1) to a quadrature; or, alternatively,

b) the setting up of an integrating factor for Eq.(1) using the explicit expressions found for $\{\xi(y,x),\ \eta(y,x)\}$.

A first look at the symmetry scheme may lead to the conclusion that finding solutions to Eq.(2) would be much more difficult than solving the original Eq.(1). However, what we are really looking for is a particular solution to Eq.(2), as *simple as possible* (for instance two constants), and for a non-classifiable ODE this particular solution may be the simpler one to be obtained.

As an example, consider

$$\frac{\partial y}{\partial x} = \frac{(y - x\ \ln(x))^2}{x^2} + \ln(x) \tag{3}$$

This equation is of Riccati type and cannot be solved easily[3], but a polynomial *guess* for the infinitesimals (made by one of **symgen**'s internal routines) leads to[4]:

```
> symgen(");        # input = ODE,  output = the infinitesimals
```

$$\_\xi(x,y) = x, \qquad \_\eta(x,y) = x + y$$

Passing Eq.(3) directly to **odsolve** (the solver), it will call **symgen**, use the result above to determine the canonical coordinates, change the variables, and solve Eq.(3) as follows:

```
> odsolve("");
```

$$y = \frac{x\sqrt{5}}{2}\ \tanh\left(-\frac{\sqrt{5}\ln(x)}{2} + \frac{\sqrt{5}\_C1}{2}\right) + x\ \ln(x) + \frac{x}{2}$$

What is amazing, and characteristic of symmetry methods is that changing $(y - x\ \ln(x))/x$ to $F((y - x\ \ln(x))/x)$ in Eq.(3), where F is an arbitrary function, the symmetry remains unchanged and the solving scheme succeeds as well:

$$\frac{dy}{dx} = F\left(\frac{y - x\ \ln(x)}{x}\right) + \ln(x) \tag{4}$$

```
> odsolve(");
```

$$\ln(x) = -\int^{\frac{y - x\ \ln(x)}{x}} \frac{d\_a}{(1 + \_a - F(\_a))} + \_C1$$

The integral above is expressed using the new **intat** command (of the new version of *PDEtools* [6]) which represents the *integral evaluated at a point* -analogous to the derivative evaluated at a point. **intat**

[3] An analysis of Eq.(3) can be obtained by sending it in REDUCE format (to be tackled with the CONVODE program) to convode@riemann.physmath.fundp.ac.be

[4] In what follows, the *input* can be recognized by the Maple prompt `>`.

displays the *evaluation point* as an upper limit of integration. For Eq.(4), Maple V R.4 returned an *explicit* result actually equivalent to the above (see Eq.(15)).

Even when the built-in heuristics fail, in the framework of SM, it is possible to provide computational routines permitting users to test their own conjectures concerning the *functional form* of the infinitesimals (HINT option, see sec.2). As an example of this, consider

$$\frac{dy}{dx} = \frac{x + \cos(\mathrm{e}^{y} + (x+1)\,\mathrm{e}^{-\mathrm{x}})}{\mathrm{e}^{(y+x)}} \tag{5}$$

This ODE is not classifiable nor is it solved by the automatic heuristics of **odsolve**. An example of a user-conjecture concerning the infinitesimals would be:

$$\xi(x,y) = f(x), \qquad \eta(x,y) = x\,g(y) \tag{6}$$

where $f(x)$ and $g(y)$ are unknown functions of their arguments. The `HINT` algorithm we programmed, given such an ansatz, will try to determine f and g, in this case leading to:

```
> symgen(",HINT=[f(x),x*g(y)]);
```

$$\_\xi = e^{x}, \qquad \_\eta = x\,e^{-y}$$

```
> odsolve("",HINT=[f(x),x*g(y)]);
```

$$y = \ln\left(2\,\arctan\left(\frac{e^{-\left(e^{-x}+\_C1\right)}-1}{e^{-\left(e^{-x}+\_C1\right)}+1}\right) - (x+1)\,e^{-x}\right)$$

2 The package's commands

A detailed description of the package's commands, with examples and explanations concerning their calling sequences, is found in the On-Line Help. Therefore, we have restricted this section to a brief summary, followed by a detailed description of the solver, **odsolve**, and the advisor, **odeadv**, and a description of just one paragraph for each of the other routines[5]. Simple *input/output* examples can be seen in sec.3.

2.1 *Summary*

A brief review of the commands of the package is as follows:

Command	Purpose:
odsolve	solve ODEs using the symmetry method scheme
fatint	look for an integrating factor
canoni	look for a pair of canonical coordinates
symgen	look for a pair of infinitesimals
equinv	look for the most general ODE invariant under a given symmetry
buisym	look for the infinitesimals given the solution of an ODE
odepde	return the PDE for the infinitesimals
odetest	test explicit/implicit results obtained by any Maple ODE-solver
symtest	test a given symmetry
odeadv	classify ODEs and pop up a Help-page with Kamke's solving advice

[5]This section contains some information already presented in the previous section; this was necessary to produce a complete description of the package.

2.2 *Description*

Command name: odsolve

Feature: 1st order ODE-solver based on symmetry methods
Calling sequence:

```
> odsolve(ode);}
> odsolve(ode, y(x), way=xxx, HINT=[expr1, expr2], int_scheme);}
```

Parameters:

`ode`	- a 1st order ODE.
`y(x)`	- the dependent variable (required when not obvious).
`way=xxx`	- optional, forces the use of only one (`xxx`) of the 6 internal algorithms {abaco1, 2, 3, 4, 5, abaco2} for determining the coefficients of the infinitesimal symmetry generator (infinitesimals).
`HINT = [e1,e2]`	- optional, `e1` and `e2` indicate a possible *functional* form for the infinitesimals.
`int_scheme`	- optional, one of: `fat`, `can`; meaning respectively: use an integrating factor; use canonical coordinates.

Optional parameters can be given alone or in conjunction, and in any order.
Synopsis:

Given a 1st order ODE, **odsolve**'s main goal is to solve it in two steps: first, determine the infinitesimals of a 1-parameter symmetry group which leaves the ODE invariant, and then use these infinitesimals to integrate the ODE, by building an integrating factor or reducing the ODE to a quadrature using the canonical coordinates of that group. To determine the infinitesimals, **odsolve** makes calls to **symgen**, another command of the package, and to make the change of variables introducing the canonical coordinates, it makes calls to **dchange**, from the *PDEtools* package [6].

odsolve does not *classify* the ODE before tackling it, and is mainly concerned with *non-classifiable* ODEs, for which the standard **dsolve** fails. By default, **odsolve** starts off trying to isolate the derivative in the given ODE, then sequentially uses all six internal algorithms of **symgen** to try to determine the infinitesimals (see sec.3), and finally tries the two integration schemes sequentially.

When **odsolve** succeeds in solving the ODE, it returns, in order of preference: an *explicit* closed solution, an *implicit* closed solution, or a *parametric* solution (only when the derivative cannot be isolated[7]).

All these defaults can be changed by the user. The main options she/he has are:

1. to request the use of only one of the six internal algorithms for determining the infinitesimals (`way=xxx` option; different algorithms might lead to different symmetries for the same ODE, perhaps turning the integration step easier);

2. to enforce the use of only one of the two alternative schemes for integrating a given ODE after finding the infinitesimals (`can` or `fat` optional arguments, useful to select the best integration strategy for each case);

3. to indicate a possible *functional form* for the infinitesimals (`HINT=xxx` option). This option is valuable when the solver fails, or to study the possible connection between the algebraic pattern of the given ODE and that of the infinitesimals.

4. to enforce the use of **dsolve** at first and **symgen**'s algorithms only when it fails (assign the global variable `sym := false`);

A brief description of how the `HINT=xxx` option can be used (see examples in sec.1 and 3) is as follows:

- `HINT=[e1,e2]` indicates to the solver that it should take e1 and e2 as the infinitesimals and determine the form of any (a maximum of two) indeterminate functions entering e1 and/or e2, such as to solve the problem (e.g., Eq.(5)).

- `HINT=[e1,'*']` where e1 represents the user's guess for the first infinitesimal, ξ, and '*' indicates to the solver that it must consider the other infinitesimal, η, as a product of two indeterminate functions of one variable each: the independent and the dependent variables, respectively (e.g., Eq.(16)).

- `HINT=[e1,'+']` works as in the previous case, but replacing the product of two indeterminate functions by a sum of them .
- `HINT=parametric` indicates to the solver that it should look for a parametric solution for the given ODE. This solving scheme may be useful even when the system succeeds in isolating the derivative [7].

Finally, there are three global variables managing the solving process, which are automatically set by internal routines but can also be set by the user, as desired. They are {*dgun, ngun, sgun*}, for setting, respectively, the maximum *d*egree of some polynomials entering ansätze for the infinitesimals, the maximum *n*umber of subproblems into which the original ODE should be mapped, and the maximum *s*ize permitted for such subproblems. The *dgun* variable is automatically set each time **symgen** is called, according to the given ODE, whereas, by default, *ngun* and *sgun* have their values respectively fixed to 2 and 1 . Although the setting of these variables by the user might increase the efficacy of the algorithms significantly, increasing by one unit the *ngun* or *sgun* variables, depending on the given ODE, may geometrically slow down the solving process.

Apart from the solver, the package includes an analyzer, **odeadv**, which can play a pedagogical role, or be a comfortable tool for classifying ODEs while studying the possible links between symmetry methods and standard ones. A detailed description of it is as follows:

Command name: odeadv

Feature: 1st order ODE-analyzer and solving advisor

Calling sequence:

```
> odeadv(ode);
> odeadv(ode, y(x),{type1,type2,type3,...},help);
```

Parameters:

`ode`	- a 1st order ODE
`y(x)`	- the indeterminate function (necessary when not obvious)
`{type1,type2,...}`	- optional, a subset of ODE classification types to be checked
`help`	- optional, to request the pop-up of a Help-page indicating Kamke's book advice for solving the given ODE.

Synopsis:

Given a 1st order ODE, **odeadv**'s main goal is to *classify* it according to Kamke's book and pop up a Help-page indicating Kamke's advice for solving it when required (by entering the word `help` as an extra argument). The Help-pages include examples and the Maple input lines implementing the advice, so as to allow the user to adapt them to her/his problem.

When used without extra arguments, **odeadv** will try to classify the given ODE into one or more of the following types:

> *quadrature; separable; linear: classes A, B, and C; homogeneous: classes A, C and D; exact; rational; Clairaut; Bernoulli; Riccati, Riccati special; Abel: 1st type, 2nd type subclasses A, B and C; d'Alembert; and "patterns": $y = g(y')$, $x = g(y')$, $0 = G(x, y')$, $0 = G(y, y')$, $y = G(x, y')$, $x = G(y, y')$*

The matching of the types is checked sequentially, in the order displayed above. **odeadv** may return more than one type when the ODE is of type homogeneous, exact or rational; otherwise, the first matching of a pattern will interrupt the process and a classification will be returned.

As an option, the user can indicate the checking of just a subset of the types mentioned above, by giving this subset as an extra argument.

A compact description of the purpose of the other routines of the package is as follows:

- **symgen** looks for the infinitesimals $\xi(x,y)$ and $\eta(x,y)$ of a 1-parameter Lie group which leave the received ODE invariant, as a pair of functions satisfying Eq.(2). The options available for this command are the same `way=xxx` and `HINT=xxx` options of **odsolve**.
- **fatint** and **canoni** respectively return the integrating factor for the given ODE or a set of transformations from the original coordinates to the canonical coordinates of the Lie group mentioned above.
- **equinv** takes as arguments a list of two algebraic expressions, to be seen as the infinitesimals of a 1-parameter Lie group, and returns, within the possibilities of the system, the most general ODE invariant under that group.
- **buisym** takes as arguments the solution of an ODE and an indication of which are the dependent and independent variables, and looks for a pair of infinitesimals for the ODE which generated the problem.
- **odetest** tests explicit and implicit solutions found by *any* Maple ODE-solver, such as **dsolve** or **odsolve**, by making a careful simplification of the ODE with respect to that solution.
- **symtest** tests the results returned by **symgen**, returning 0 when these results check *ok*, or an algebraic expression obtained after simplifying the PDE for the infinitesimals according to **symgen**'s result.

buisym and **equinv** may be useful in connection with the **odeadv** command and the `HINT` option of **odsolve**, in order to study the relation between symmetry patterns and ODE patterns. Also, it is worth mentioning that, since any *solver* will make extensive use of the whole Maple library, testing commands such as **odetest** and **symtest** would also be useful in detecting possible wrong results of the Maple library.

3 The *heuristic* methods

3.1 General remarks

All the implemented *heuristic* methods assume that the system was successful in isolating the derivative, as in Eq.(1). These methods consist of six computational *algorithms* for building and testing an explicit algebraic expression for the infinitesimals ξ and η. By heuristic we mean that the possible success of the scheme cannot be determined *a priori*.

Two of the six algorithms are in turn subdivided into eight and four subalgorithms respectively; thus, in all, the given ODE is actually tackled by trying sixteen different *functional forms* for ξ and η. Three of these sixteen schemes involve different types of polynomial ansätze for $\{\xi,\ \eta\}$, whose form, apart from the polynomial degree, is virtually independent of the ODE input. In the other thirteen schemes, the ansätze for ξ and η depend on the received problem, and are not *a priori* of polynomial type, but rather are obtained by solving auxiliary ODEs.

Four of the six algorithms directly tackle Eq.(2), while the other two reformulate the problem taking into account that Eq.(2) admits $\eta = \xi\,\Phi$ as a general (useless) solution. Therefore, it is always possible to introduce

$$\eta = \xi\,\Phi + \chi \tag{7}$$

in Eq.(2), where $\chi(x,y)$ is an unknown function of its arguments, in order to map the original problem into that of solving

$$\frac{\partial \chi}{\partial x} + \Phi \frac{\partial \chi}{\partial y} - \left(\frac{\partial \Phi}{\partial y}\right)\chi = 0 \tag{8}$$

for χ. The knowledge of χ, in turn, allows one to set ξ and η as desired using Eq.(7).

For the case in which all these schemes fail, we programmed a HINT option to allow the user to try her/his own heuristics for finding the infinitesimals, as explained in sec.2.2.

The six algorithms

The 1st algorithm, called *abaco*$_1$, consists of 2 sets of 4 schemes each, leading to 8 different trials. The first 4 trials look for $\{\xi,\ \eta\}$ taking one of the infinitesimals as 0 and the other one as a function of only one variable, namely:

$$\{\xi = 0,\, \eta = f(x)\}, \quad \{\xi = 0,\, \eta = f(y)\}, \quad \{\xi = f(x),\, \eta = 0\}, \quad \{\xi = f(y),\, \eta = 0\} \tag{9}$$

The possible success of each case is determined by algebraic factorization of Eq.(2), and when this possibility is confirmed, an auxiliary ODE is built to determine the explicit resulting form for f.

The second sequence of 4 trials looks for $\{\xi,\ \eta\}$ taking one of the infinitesimals as 0 and the other one as an expression containing both x and y, namely:

$$\begin{aligned} &\{\xi = 0,\, \eta = f(x)\, g(y)\}, \qquad \{\xi = f(x)\, g(y),\, \eta = 0\}, \\ &\{\xi = 0,\, \eta = f(y)\, g(x)\}, \qquad \{\xi = f(y)\, g(x),\, \eta = 0\} \end{aligned} \tag{10}$$

where g is an algebraic expression built by extracting factors from the numerator and denominator of the ODE to be solved, and f is an unknown function to be determined by solving auxiliary ODEs as in the first 4 trials.

Example[6]*:* Kamke's ODE 120.

$$x\frac{dy}{dx} - y\left(x\ln\left(\frac{x^2}{y}\right) + 2\right) = 0 \tag{11}$$

```
> symgen(");
```

$$\_\xi = 0, \quad \_\eta = -e^{-x}y \qquad (g(y) = y \text{ and } f(x) \text{ was determined as } -e^{-x})$$

```
> odsolve("");
```

$$y = x^2 e^{(\_C1\, e^{-x})}$$

As a general benchmark, *abaco*$_1$ can determine a pair of infinitesimals for 151 of the 466 non-quadratures found in Kamke's first 500 ODEs, spending on each an average time of 0.26 sec. when successful and 1.1 sec. otherwise.

The 2nd of the set of six main algorithms consists of a bivariate polynomial in (x, y) ansatz for χ to solve Eq.(8). The maximum degree of the polynomial can be set by the user by assigning the global *dgun* variable; otherwise, it will be determined automatically by an internal routine. This second algorithm can determine a pair of infinitesimals for 96 of Kamke's 466 non-quadrature ODEs mentioned above, spending an average time of 0.08 sec. when successful and 0.5 sec. otherwise.

The 3rd and 4th of the set of six algorithms respectively consist of bivariate polynomial and rational ansätze in (x, y) for $\{\xi, \eta\}$, to solve Eq.(2).

Example: Kamke's ODE 236 (Abel, 2nd type, class B),

$$x\,(y+4)\,\frac{dy}{dx} - y^2 - 2\,y - 2\,x = 0 \tag{12}$$

```
> symgen(");
```

$$\_\xi = 4\,x + x^2, \quad \_\eta = xy + 4\,x$$

In solving the aforementioned 466 Kamke's non-quadrature ODEs, these two algorithms can determine a pair of infinitesimals in 234 cases, spending an average time of 0.7 sec. in each success and 1.2 sec otherwise.

The 5th algorithm builds an explicit expression for $\chi(x, y)$, to solve Eq.(8), as follows:

[6]In all the following examples, an alias has been set for y, via `> alias(y=y(x));`

1. a basis of functions and algebraic objects is built by taking, from the given ODE, all the *known* functions[7] and composite algebraic objects, together with their derivatives, as well as all the *unknown* functions;

2. a polynomial of degree 2 in such objects is built; its coefficients, in turn, are polynomials of degree d (the *dgun* variable mentioned above) in $\{x, y\}$ with undetermined coefficients.

This ansatz for χ is introduced in Eq.(8) resulting in a system of algebraic equations for the undetermined coefficients mentioned above.
Example: Kamke's ODE 357 (no classification)

$$x\left(\frac{dy}{dx}\right)\ln(x)\sin(y)+\cos(y)\left(1-x\cos(y)\right)=0 \tag{13}$$

```
> symgen(");
```

$$\_\xi = 0, \qquad \_\eta = \left(\frac{\cos(2\,y)x}{2}+\frac{x}{2}\right)\frac{1}{2\,x\ln(x)\sin(y)}$$

```
> odsolve("");
```

$$y = \arccos\left(\frac{2\,\ln(x)}{2\,x+\_C1}\right)$$

This algorithm can find a pair of infinitesimals for 218 of Kamke's 466 non-quadratures mentioned above, spending on each an average time of 2.1 sec. when successful and 1.8 sec. otherwise.

The 6$^{\text{th}}$ algorithm, called *abaco*$_2$, leads to a sequence of 4 trials for the infinitesimals $\{\xi, \eta\}$, making use of the HINT option of **symgen**, with two indeterminate functions of only one variable each, namely:

$$\begin{aligned}&\{\xi = g(x),\, \eta = f(x)\}, \qquad \{\xi = g(x),\, \eta = f(y)\},\\ &\{\xi = g(y),\, \eta = f(x)\}, \qquad \{\xi = g(y),\, \eta = f(y)\}\end{aligned} \tag{14}$$

The possible success of these trials *cannot* be determined by algebraic factorization of Eq.(2), as in *abaco*$_1$, except in a few cases. Hence, the alternative scheme we implemented for managing such HINTs consists of determining f and g as follows:

1. subdivide Eq.(2) into subexpressions involving only one of $\{f,\, g\}$;

2. build a list of *candidates* for f and for g with the solutions to these subexpressions;

3. build a list of *pairs of candidates* by taking one candidate from each list.

To avoid spending much time, the number of *candidates* of each list in step (2) is restricted, by default, to $n = 2$, leading to a maximum of 4 *pairs of candidates* for each functional form in Eq.(14). This default can be changed by the user by assigning the global variable *ngun*.
Examples: ODE 593 (file 6) with an arbitrary function $F(x, y)$

$$\frac{dy}{dx} = \frac{e^x}{\sqrt{y}}\,F\left(\sqrt{y^3}-\frac{3\,e^x}{2}\right) \tag{15}$$

```
> symgen(");
```

$$\_\xi = \frac{1}{e^x}, \qquad \_\eta = \frac{1}{\sqrt{y}} \qquad\qquad \left(g(y) = \frac{1}{\sqrt{y}},\ f(x) = \frac{1}{e^x}\text{ - see Eq.(14)}\right)$$

```
> odsolve("");          # Maple V R.4 output using intat and RootOf
```

[7] By *known* function we mean a function whose derivative rule is known by the Maple system.

$$y = \left(\text{RootOf}(\int^{\frac{2\_Z^3}{3} - e^x} \frac{d\_a}{(-1 + F(\frac{3\_a}{2}))} - e^x + \_C1) \right)^2$$

```
> odetest("""","); # explicit and implicit results can be tested with odetest
```

$$0$$

For this ODE, Maple V R.3 returned an *implicit* (equivalent) result, as in Eq.(4).

abaco$_2$ can determine $\{\xi, \eta\}$ for 114 of Kamke's 466 non-quadratures mentioned above, spending on each an average time of 0.4 sec. when successful and 8 sec. otherwise.

To conclude this section, a curious result illustrating the HINT option is given by the infinitesimals found by **symgen** for the general form of Bernoulli's equation,

$$\frac{dy}{dx} + f(x)\, y + g(x)\, y^{\alpha} = 0, \tag{16}$$

where f and g are arbitrary functions of their arguments, and α is a symbolic power. The giving of a HINT, asking for a *separation of variables by product* for one of the infinitesimals (see subsec.2.2) leads to:

```
> symgen(",HINT=[0,`*`]);
```

$$\_\xi = 0, \quad \_\eta = y^{\alpha}\, e^{\int f(x)dx\,(\alpha - 1)}$$

```
> factor(odsolve("",HINT=[0,`*`]));
```

$$y = \left(e^{\left(\int f(x)dx\,(\alpha - 1) \right)} (\alpha - 1) \left(\int^{x} g(\_a) e^{\left(- \int f(\_a)d\_a\,(\alpha - 1) \right)} d\_a - \_C1 \right) \right)^{\frac{1}{1-\alpha}}$$

4 Tests

We tested the set of routines here presented having two different ideas in mind: to confirm the correctness of the returned results, and to evaluate the new routines' *performance* in solving Kamke's ODEs. The performance test was done with the intention of comparing the efficacy of a *SM-based* solver such as **odsolve** with that of a *classification-based* solver as is Maple's **dsolve**.

We used 10 test files, containing a set of 1,000 ODEs distributed in: files 1 to 6 containing the 576 ODEs of Kamke's book, and files 7 to 10 containing 424 ODEs of Riccati, Abel and *non-classifiable* types, for which the *classification-based* **dsolve** (R.3/4) fails. The ODEs of files 7-10 were built using the **equinv** command, departing from symmetries of Kamke's ODEs, and were used to test the correctness of the results returned by **symgen** and the integration subroutines of **odsolve**. The "*fail* by **dsolve**" condition for these 424 ODEs was proposed to evaluate **odsolve** in solving the type of ODEs for which it was really designed, i.e., non-classifiable or usually not solved by **dsolve**.

The purpose of preparing files with such a big set of ODEs was to set up what would be a convenient test-file for evaluating the performance of computational implementations in solving *classifiable/non-classifiable* ODEs. The 10 files, with the input code in Maple, Reduce and Mathematica format, are available at our site: http://dft.if.uerj.br/symbcomp.html.

A summary with the number of ODEs of each file belonging to each *classification* (we used our **odeadv** classification command) is as follows:

Class:	File: 1	2	3	4	5	6	7	8	9	10	Totals:
quadrature	10	0	1	12	10	13	0	0	0	0	46
separable	19	10	3	8	10	7	0	0	0	0	57
linear	14	15	16	1	1	0	0	0	0	0	47
homogeneous	1	19	15	11	28	4	14	19	19	8	138
exact	0	0	14	14	0	0	0	0	0	0	28
rational	5	48	68	28	26	1	17	64	54	40	351
Clairaut	0	0	0	7	12	8	0	0	0	0	27
Bernoulli	0	1	0	0	0	0	0	0	0	0	1
Riccati	23	46	1	0	0	0	13	27	19	21	150
Abel	15	8	30	0	0	0	12	19	25	42	151
d'Alembert	3	2	15	7	36	10	1	0	0	0	74
none	16	11	6	27	23	36	91	27	25	42	304
has $F(x,y)$	21	4	8	8	1	13	43	0	0	7	105
Total of ODEs	100	100	100	100	100	76	124	100	100	100	1,000

Table 1. *Classification of the 1,000 ODEs of files 1-10*

The quadratures pointed out in the table were found after isolating the derivative, receiving a rhs not containing x or $y(x)$ (the independent and dependent variables all along the 1,000 ODEs). The next-to-last line of the table above indicates the number of ODEs having an arbitrary function $F(x,y)$; e.g. $F(x^2 - e^y/y)$. Also, in many cases, **odeadv** returned more than one classification for a given ODE; all these possible classifications are reflected above.

4.1 Test of symgen

As explained in sec.1, the SM approach for solving ODEs involves two main steps: the determination of a pair of infinitesimals and their subsequent use in the integration process. We therefore divided the tests into the same two steps.

The first *performance* test, thus, concerned the explicit determination, by the **symgen** command, of infinitesimals for Kamke's ODEs (only the 466 non-quadratures of the first 500 ODEs were used). The table below displays the total number of successes, the average time spent with each solved/fail case, and the number of successes of *each* of **symgen**'s six algorithms when the whole set of ODEs was tackled using only one of them. The second column of the table indicates the number of non-quadrature ODEs per file.

File	ODEs	Successes	Average time[8]		ξ and η can be determined by					
			ok	*fail*	$abaco_1$	2	3	4	5	$abaco_2$
1	90	44	1.7 sec.	4.2 sec.	33	13	21	25	29	23
2	100	71	1.2 sec.	2.0 sec	30	13	47	46	39	27
3	99	81	0.7 sec.	3.6 sec.	35	36	55	46	40	15
4	88	72	3.8 sec.	4.1 sec.	38	28	44	45	49	21
5	89	75	1.7 sec.	5.1 sec.	15	6	67	71	61	28
Totals:	466	343	$\approx$ 8 sec.	$\approx$ 12 sec.	151	96	234	233	218	114

Table 2. *Kamke's ODEs for which the infinitesimals were determined by* **symgen***: 73%*

4.2 Test of odsolve's integration schemes and comparison of performances

The two integration alternatives we programmed for the second step of the solving scheme consist of either using an integration factor or making a change of variables (canonical coordinates) reducing the ODE to a quadrature. Of course, the corresponding routines work *only if* **symgen** succeeds in determining the infinitesimals.

Also, as mentioned, though **odsolve** was thought as a complement to the Maple **dsolve**, we prepared a comparative test of performances to have an idea of the possible efficacy of a *SM-based* solver, if compared to that of a *classification-based* solver as **dsolve**. With this purpose, we tested the standard **dsolve** with the non-quadrature Kamke's ODEs too. The results we found in testing the integration schemes and in comparing **odsolve**'s performance with that of **dsolve** can be summarized as follows:

[8]Due to the large number of ODEs being tested, we interrupted and computed as *fail* all calculations consuming more than a few minutes. This was the case in Kamke's ODEs numbers 51, 340, 347, 373, 394, 460 and 479; they were recalculated separately, receiving an *answer* in $5 \sim 20$ min. for all but ODE 479 .

File	ODEs	Solved by		Successful integration via	
		dsolve	**odsolve/symgen**	`can`	`fat`
1	90	47	44 / 44	41	40
2	100	71	71 / 71	66	70
3	99	75	79 / 81	61	78
4	88	63	70 / 72	51	68
5	89	69	70 / 75	69	55
Totals:	466	325	334 / 343	288	311

Table 4. *Comparative performance*

It was a surprise to us that, even when more than 80 % of the ODEs used in the test are of *classifiable* type (see Table 1.), the performance of **odsolve**, which does not classify the received ODE before tackling it, was almost the same as that of **dsolve**[9]. Also, **odsolve**'s integration schemes succeeded in 97 % (334 of 343) of the cases in which **symgen** determined a pair of infinitesimals, convincing us in that the main problem is, actually, the determination of those infinitesimals, not their posterior use in the integration process.

5 Conclusions

This paper presented a computer algebra implementation of symmetry methods for solving 1st order ODEs. Despite the additional complications introduced by the necessity of solving an auxiliary 1st order PDE, this method proved to be a valuable tool for solving non-classifiable (or even classifiable) ODEs, as shown in sec.4, resulting in the extension to Maple's **dsolve** we were looking for.

Moreover, we would like to remark the *interactive* character of the solver too; i.e., the user is given the possibility of participating in the solving process (the HINT option, see sec.2.2), achieving in this way a significant extension of the solving capabilities of the scheme.

Also, the computational routines presented here were designed not only for solving ODEs: the **odeadv** command and the HINT option of **odsolve** were thought as a combination for investigating new solving methods, and the set of commands here presented was intended to be complete from the educational point of view. Actually, with them it is possible to tackle an ODE in the framework of symmetry methods, directly or step by step, it being also possible to *go back* using the **equinv** and **buisym** commands, completing the conceptual cycle.

On the other hand, this is a first version of the package and many things can be improved. To mention but a few, the classification abilities of **odeadv** can be extended, and used by **symgen** in order to determine the most efficient ordering of the algorithms for tackling the given ODE (at present this order is unmovable); also, a study of the fail examples can be used as a basis for building new algorithms, mainly for ODEs of Riccati and Abel types. Such new algorithms can be implemented straightforwardly using the *HINT* option of **symgen**. Moreover, for instance, it is possible to extend the ideas presented here to tackle higher order ODEs too. These subjects are part of our present interests and we expect to be able to report related work in the near future[7].

Acknowledgments

This work was supported by the State University of Rio de Janeiro (UERJ), Brazil. The authors would like to thank J.E.F. Skea[10] and K. von Bülow[9] for useful discussions and a careful reading of this paper; and Prof. M.A.H. MacCallum[11] for kindly sending us a copy of his paper [4], as well as valuable references.

References

[1] G.W. Bluman and S. Kumei, *Symmetries and Differential Equations*, Applied Mathematical Sciences **81**, Springer-Verlag, (1989).

[2] P.J. Olver, *Applications of Lie Groups to Differential Equations*, Springer-Verlag, (1986).

[9] The possible success in solving Kamke's ODEs by introducing simple *hypotheses* for the infinitesimals has already been pointed out in a previous work by B. Char [8].

[10] Symbolic Computation Group of the Theoretical Physics Department at UERJ - Brazil.

[11] Queen Mary and Westfield College, University of London - U.K.

[3] W. Hereman, *Review of Symbolic Software for Lie Symmetry Analysis*, Mathematical and Computer Modeling, Vol. 20, Special Issue on Algorithms for Nonlinear Systems (1995).

[4] MacCallum, M.A.H., *Using computer algebra to solve ordinary differential equations*, Computer algebra in industry 2: Problem solving in practice, ed. A.M. Cohen, L.J. van Gastel and S.M. Verduyn Lunel, pp. 19-41, John Wiley and Sons, Chichester (1995).

[5] Kamke, E., *Differentialgleichungen: Lösungsmethoden und Lösungen*. Chelsea Publishing Co, New York (1959).

[6] E.S. Cheb-Terrab and K. von Bülow, Comp. Phys. Comm., 90 (1995) 102-116.

[7] E.S. Cheb-Terrab, L.G.S. Duarte and J.F.E. Skea, *Computer Algebra solving of Riccati and Abel 1st Order ODEs*, in preparation.

[8] B. Char, *Using Lie transformations to find closed form solutions to first order ordinary differential equations*, *SYMSAC 81*, ACM, New York (1981) 44-50.