Symbolic solutions for a class of partial differential equations

J. Symbolic Computation (1996) **22**, 459–468

Symbolic Solutions for a Class of Partial Differential Equations

BRAM DE JAGER[†] AND BRAM VAN ASCH[‡]

[†]*Faculty of Mechanical Engineering,*
[‡]*Faculty of Mathematics and Computer Science,*

Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

(Received 12 June 1995)

An algorithm to generate solutions for members of a class of completely integrable partial differential equations has been derived from a constructive proof of Frobenius' Theorem. The algorithm is implemented as a procedure in the computer algebra system Maple. Because the implementation uses the facilities of Maple for solving sets of ordinary differential equations and for sets of nonlinear equations, and these facilities are limited, the problems that actually can be solved are restricted in size and complexity. Several examples, some derived from industrial practice, are presented to illustrate the use of the algorithm and to demonstrate the advantages and shortcomings of the implementation.

1. Introduction

Solving systems of differential equations is a difficult problem, except for some special classes. General purpose CAS (computer algebra systems) do not provide facilities that can be applied generally. Maple's command `dsolve`, for instance, will provide full solutions only to a restricted class of ODE's (ordinary differential equations). The same holds for other CAS.

A facility provided by Maple is the `liesym` package that computes Lie point symmetries of differential equations. This package is not integrated in the `dsolve` command. Most other CAS provide equivalent facilities. For ODE's these Lie point symmetries, or if desired other symmetries, can be used to reduce the order of the ODE. The following example is taken from Stephani (1989, p. 30).

Consider the second order differential equation

$$y'' = (x - y)(y')^3.$$

By considering the Lie point symmetry

$$\tilde{x} = x + \epsilon(x - y), \qquad \tilde{y} = y$$

[†] E-mail: `A.G.de.Jager@wfw.wtb.tue.nl`
[‡] E-mail: `wsinaa@win.tue.nl`

0747–7171/96/100459 + 10 $25.00/0

we arrive at a change of coordinates

$$t = y, \qquad s = \ln(x - y)$$

which transforms the ODE into

$$s'' + (s')^2 + 1 = 0.$$

So, we have reduced the order by 1, because this is a first order ODE in s'.

In general the following holds. If we have an ODE of order n, and there are n Lie point symmetries, we can reduce the problem to order 0, provided that the Lie group generated by the symmetries is solvable.

For PDE's (partial differential equations) the situation is different. Here, Lie point symmetries cannot be used to lower the order of the differential equation. Instead, they might be used to reduce the number of independent variables, and thus gain access to special classes of solutions. Eventually this might even lead to an ODE. In Stephani (1989), for example, it is shown how the wave equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} - \frac{\partial^2 u}{\partial t^2} = 0$$

can be reduced to the ODE

$$\sigma \frac{\mathrm{d}^2 w}{\mathrm{d}\sigma^2} + \frac{\mathrm{d}w}{\mathrm{d}\sigma} = 0, \qquad \sigma = \frac{x^2 + y^2}{t^2 - z^2}, \qquad w = u$$

using symmetries. However, in general the problem of finding Lie point symmetries for PDE's is of equal difficulty to solving the differential equation.

Having demonstrated the procedure to use symmetries in solving (partial) differential equations, the question arises if this technique has any practical value. As answers to this question several different points of view have been encountered in the literature. Stephani is rather pessimistic. In his book (Stephani, 1989) it is noted that Lie symmetries have hardly ever any value for obtaining explicit solutions for ODE's or PDE's. However, since the publication of Stephani's book, a number of articles appeared using certain kinds of symmetries in finding exact solutions of PDE's. In MacCallum (1995) it is remarked that Lie symmetries are at least an ordering principle for obtaining solutions, and results of a computer algebra program are summarized that could solve a large part of a set of reference ODE's using Lie symmetries. Also, Schwarz and Wolf (MacCallum, 1995) use Lie symmetries as an integral part of their programs to compute explicit solutions of ODE's and PDE's. A lot of other work in this area has been performed. Furthermore, Lie's symmetry method is the method that is generally applicable to generate exact solutions, so it certainly has its value.

We do not delve into this further, but conclude that other approaches than Lie symmetries to solve certain classes of PDE's are worthwhile avenues, so other ways to obtain solutions have to be searched for. This comes down, in most cases, to restrict the class of PDE's one wants to consider, and to develop rather specific methods for this class.

For special types of PDE's, as the ones to be considered here, the method of characteristics is an alternative, because it is particularly well suited for coupled systems of first order (Kevorkian, 1993; Sneddon, 1957). In this paper, however, yet another method, based on a constructive proof of Frobenius' Theorem, is discussed. The method is suitable for sets of first order PDE's that are completely integrable, or, in other words, for which a certain distribution is involutive, as stated by Frobenius' Theorem. Note that

this type of PDE is frequently encountered in, at least, nonlinear system theory in a differential geometry setting, and has therefore great practical significance. For applications in this area we are content with finding a single closed form solution, but without user intervention.

2. Frobenius' Theorem

This section and the next one draws from Isidori (1989). Starting with sets of first order PDE's of the following form

$$\frac{\partial h(x)}{\partial x}\begin{bmatrix} f_1(x) & \cdots & f_d(x) \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} \tag{2.1}$$

with $x \in \mathbb{R}^n$, $\frac{\partial h(x)}{\partial x}$ the row vector $[\frac{\partial h}{\partial x_1} \cdots \frac{\partial h}{\partial x_n}]$, and f_i a smooth vector field ($i = 1, \ldots, d$), we want $n - d$ independent solutions $h(x)$ of this set of PDE's. This is equivalent to the complete integrability of the distribution $\Delta(x)$ spanned by the columns f_i, $i = 1, \ldots, d$, with Δ assumed to be nonsingular. Frobenius' Theorem now states

> A nonsingular distribution Δ is completely integrable if and only if it is involutive.

The constructive proof of the sufficiency part of this Theorem in Isidori (1989, pp. 26–30) is used to solve the PDE's.

3. Algorithm

The solution algorithm used can be described by the following steps.

1. Test the involutiveness of the distribution $\Delta(x) = \text{span}\{f_1(x), \ldots, f_d(x)\}$.
2. Extend the distribution Δ with vector fields $f_{d+1}, \ldots, f_n$ such that the n vector fields $f_1, \ldots, f_n$ span $\mathbb{R}^n$.
3. Compute the flows $\Phi_t^{f_i}(x^\circ)$ for the vector fields f_i, $i = n, \ldots, 1$. The flow $x(t) = \Phi_t^f(x^\circ)$ of a vector field f is by definition the solution of the initial value problem $\frac{\mathrm{d}x}{\mathrm{d}t} = f$ with initial condition $x(0) = x^\circ$.
4. Determine the mapping $F\colon \mathbb{R}^n \mapsto \mathbb{R}^n$ defined by composing the flows:
$$F(z_1, \ldots, z_n) = \Phi_{z_1}^{f_1} \circ \cdots \circ \Phi_{z_n}^{f_n}(x^\circ).$$
Here $\circ$ denotes composition with respect to x. So
$$\Phi_{z_{n-1}}^{f_{n-1}} \circ \Phi_{z_n}^{f_n}(x^\circ) = \Phi_{z_{n-1}}^{f_{n-1}}(\Phi_{z_n}^{f_n}(x^\circ))$$
etc.
5. The last $n - d$ components of the inverse mapping F^{-1} are independent solutions of the PDE's.

4. Implementation

The steps of the solution algorithm are implemented in the new procedure `psolve` as follows.

1. The involutiveness of the distribution is tested by checking if the rank of the matrices

$$\left[\begin{array}{cccc} f_1 & \cdots & f_d & [f_i, f_j] \end{array}\right]$$

for $1 \leq i < j \leq d$, where $[f_i, f_j] = \frac{\partial f_j}{\partial x} f_i - \frac{\partial f_i}{\partial x} f_j$ denotes the Lie bracket of f_i and f_j, does not increase and is still equal to $\operatorname{rank}[\, f_1 \ \cdots \ f_d \,] = d$.

2. The extension is performed iteratively, by augmenting the matrix $[\, f_1 \ \cdots \ f_d \,]$ with a single unit column (the simplest vector field), testing the rank, if the rank increases this column is kept and the next unit column is added until the rank is n.
3. The flow of the vector fields is computed with the `dsolve` procedure.
4. The construction of the mapping F, i.e., the composition of the flows, is performed by backward substitution of the computed flows.
5. The inverse mapping F^{-1} (the solution of a set of nonlinear algebraic equations) is computed with the `solve` procedure and the solution of the PDE's is extracted from this mapping.

Several checks are built in to test if the input data is correct, e.g., if the distribution Δ is nonsingular, to test if the computations are performed without errors, and finally to test the solution $h(x)$. The rank determination in the input check and in steps 1 and 2 is done with Gauss elimination techniques and thus gives a generic rank, i.e., for a lower dimensional subset in $\mathbb{R}^n$ the rank may drop.

We remark that the solution for (2.1) may not be unique. In general, `psolve` will only compute a single closed form solution, which is all we require in our use of this procedure, that may not be the most simple or insightful one. This is illustrated in the examples.

It is also evident from this sketch of `psolve` that critical parts in the implementation are the `dsolve` and `solve` procedures that should perform some of the hard work. The `dsolve` function is (partly) based on an implementation of the Risch integration algorithm (Geddes *et al.*, 1992). The main reasons to set up the implementation in this way are (1) ample use is made of existing code, (2) any improvement made to the `dsolve` and `solve` procedures will also improve `psolve`.

The procedures `dsolve` and `solve` sometimes fail to give an appropriate answer. The capabilities and limitations of the `dsolve` procedure are, for instance, discussed in Postel and Zimmermann (1996). This review also shows that, although not perfect, the `dsolve` procedure of Maple is reasonably powerful compared with the facilities of other CAS. The limitation of the `solve` procedure to solve conveniently sets of nonlinear equations is illustrated in example 5 in the next section.

5. Examples

This section presents five examples. Four for which an explicit solution can be found, and a last one for which an explicit solution is known to exist but cannot be computed directly by `psolve`. The first and last example are computed with Maple V Release 3, the other examples with Maple V. For all examples it is straightforward to check the answers by hand.

EXAMPLE 1. The single PDE that has to be solved for a nontrivial $h = h(x_1, x_2)$ is (Isidori, 1989, Example 1.4.1)

$$\frac{\partial h(x)}{\partial x} \begin{bmatrix} \mathrm{e}^{x_2} \\ 1 \end{bmatrix} = 0. \tag{5.1}$$

Note that the partial differential equation is linear in $h(x)$ but the coefficients of the derivatives are not all polynomial in x. The solution $h(x)$ is computed by Maple as follows.

```
> n := 2:
> x := array(1..n):
> f := array(1..n,1..1,[ [exp(x[2])], [1] ]):
> xinit := {seq (x[‘i‘]=0,i=1..n)}:
> psolve(f,x,xinit);
                              [ x[1] - exp(x[2]) + 1 ]
```

Note that the command `psolve` needs a description of the differential equation in `f`, the name of the independent variable x in `x`, and the initial condition x° in `xinit`. The last line of the output contains the desired solution,

$$h(x) = x_1 - \mathrm{e}^{x_2}, \tag{5.2}$$

where the constant 1 can be neglected. The correctness of the solution can be verified directly by substitution of (5.2) in the differential equation (5.1).

EXAMPLE 2. The set of PDE's is (Isidori, 1989, Example 1.4.3)

$$\frac{\partial h(x)}{\partial x} \begin{bmatrix} 2x_3 & -x_1 \\ -1 & -2x_2 \\ 0 & x_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}. \tag{5.3}$$

The solution $h(x)$ is computed by `psolve` as follows.

```
> n := 3:
> x := array(1..n):
> f := array(1..n,1..2,[[2*x[3], -x[1]], [-1, -2*x[2]], [0, x[3]]]):
> xinit := {seq (x[‘i‘]=0,i=1..n-1), x[n]=1}:
> psolve(f,x,xinit);
                                                 2
                          [ x[1] x[3] + 2 x[3]  x[2] ]
```

The solution generated is correct, as can be checked by substitution in (5.3) of the solution $h(x) = x_1x_3 + 2x_2x_3^2$.

For arbitrary initial conditions, generated by

```
> xinit := {seq (x[‘i‘]=x0[‘i‘],i=1..n)}:
```

the computed solution becomes

```
                          2                2
    x[1] x[3] + 2 x[3]  x[2] - 2 x0[3]  x0[2] - x0[3] x0[1]
[ ------------------------------------------------------- ]
                               x0[3]
```

which shows that for $x_3^\circ = 0$ this solution is not valid.
If in the set `xinit` the condition for x_3° is set to 0 the following error results.

```
Warning, (in psolve) MAPLE failed in testing the inverse mapping of F,

    [ locz3 exp(- locz2), - locz1, 0 ], for component, 1, 2, 3

    although this inverse was computed by solve.
```

This is a limitation of the implementation, that does not account for restrictions on the initial conditions. It is easy to see that in fact a unique inverse does *not* exist, because the Jacobian of the mapping F is singular for $x_3^\circ = 0$.

Example 3. The problem considered stems from the analysis of the exact linearizability by state feedback and coordinate change of the dynamics of a DC motor driving a load with moment of inertia J. The relevant set of PDE's is given by

$$\frac{\partial h(x)}{\partial x}\begin{bmatrix} 1/L_s & R_s/L_s^2 \\ 0 & Kx_3/L_r \\ 0 & -Kx_2/J \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

that has to be solved for $h(x)$. The indexed L, R, and the constant K are the inductance, resistance, and motor constant respectively. The motor inertia is lumped with the load inertia J. The indices $_s$ and $_r$ stand for stator and rotor. The components of the state x are stator and rotor currents and rotor speed. The solution can be computed by `psolve` according to the following session log.

```
> n := 3:
> x := array(1..n):
> f := array(1..n,1..2,[[1/Ls, Rs/Ls^2], [0, K*x[3]/Lr], [0, -K*x[2]/J]]):
> xinit := {seq (x[‘i‘]=0,i=1..n)}:
> psolve(f,x,xinit);
                                    2           2  1/2
                              (J x[3]  + Lr x[2]  )
                            [ ----------------------- ]
                                       1/2
                                      J
```

It is easy to see that, e.g., $h(x) = L_r x_2^2 + J x_3^2$, an energy type function, is also a solution, so the "simplest" form for h is not necessarily found.

Example 4. This set of PDE's is derived from the input matrix of a model for a robot with one translational and one rotational joint and occurs in computing the transformation to a normal form of the equations of motion of the robot.

$$\frac{\partial h(x)}{\partial x}\begin{bmatrix} 1 & 0 \\ \sin x_2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}.$$

Note that here a non-polynomial function in x is involved. Initially, a solution cannot be computed, as shown by the following log.

```
> n := 3:
> x := array(1..n):
```

```
> f := array(1..n,1..2,[[1, 0], [sin(x[2]), 0], [0, 1]]):
> xinit := {seq (x[`i`]=0,i=1..n)}:
> psolve(f,x,xinit);
         psolve:, unable to compute coeff,

             MAPLE procedure dsolve failed in solving the flow of: ,

             [ 1, sin(locx2(locz1)), 0 ], continuing with extdsolve
```

The message is due to limitations of the `dsolve` procedure, used to compute the flows, to solve sets of ODE's. To remedy this, an extension named `extdsolve` was written, that is able to solve the flow in this case (this extension was also needed to solve Example 1). Now the following appears later on in the computation.

```
Warning, (in psolve) MAPLE failed in testing the inverse mapping of F,

                          exp(locz1) (- 1 + cos(x0[2] + locz3))
locz1+x0[1], -2 arctan(-------------------------------------), x0[3]+locz2
                                  sin(x0[2] + locz3)

for component, 2, although this inverse was computed by solve.

                                (- 1 + cos(x[2])) exp(- x[1] + x0[1])
        [ - x0[2] - 2 arctan(-------------------------------------) ]
                                             sin(x[2])
```

The last line contains the desired answer

$$h(x) = 2\arctan\left(\tan\frac{x_2}{2}\mathrm{e}^{-x_1+x_1^\circ}\right) - x_2^\circ \qquad \text{because} \quad \frac{-1+\cos x_2}{\sin x_2} = -\tan\frac{x_2}{2}.$$

The warning is due to limitations in testing if the inverse mapping F^{-1} as computed by the `solve` procedure is correct. In this case it has no consequences, but the warning has the intention to generate some caution with respect to the correctness of the answer.

Example 5. Consider the single PDE which is a generalization of the first PDE in Example 4

$$\omega(x_1)\frac{\partial h}{\partial x_1} + \psi(x_2)\frac{\partial h}{\partial x_2} = 0$$

i.e.,

$$f_1(x) = \begin{bmatrix}\omega(x_1)\\ \psi(x_2)\end{bmatrix}.$$

Choose

$$f_2(x) = \begin{bmatrix}1\\ 0\end{bmatrix}.$$

It is obvious that

$$\Phi_t^{f_2} = \begin{bmatrix}t + x_1^\circ\\ x_2^\circ\end{bmatrix}.$$

The computation of $\Phi_t^{f_1}(x^\circ)$ comes down to solving the two ODE's

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = \omega(x_1) \qquad \text{and} \qquad \frac{\mathrm{d}x_2}{\mathrm{d}t} = \psi(x_2).$$

In both cases the variables can be separated. A first problem may arise when we try to express

$$\int \frac{1}{\omega(x_1)}\mathrm{d}x_1 \qquad \text{and} \qquad \int \frac{1}{\psi(x_2)}\mathrm{d}x_2$$

in terms of elementary functions. This may not be possible, but if we are using CAS this problem can be solved by adding the integrals of $1/\omega(x_1)$ and $1/\psi(x_2)$ to the class of functions we are using. So, let us assume we can find functions $\Omega(x_1)$ and $\Psi(x_2)$ which are the integrals of $1/\omega(x_1)$ and $1/\psi(x_2)$ respectively. Let us assume furthermore that Ω and Ψ have inverses Ω^{-1} and Ψ^{-1}. Then it follows that

$$\Phi_t^{f_1}(x^\circ) = \begin{bmatrix} \Omega^{-1}(t + \Omega(x_1^\circ)) \\ \Psi^{-1}(t + \Psi(x_2^\circ)) \end{bmatrix}.$$

An easy calculation, following steps 4 and 5 of the solution algorithm, yields

$$h(x_1, x_2) = \Omega^{-1}(\Omega(x_1) - \Psi(x_2) - \Psi(x_2^\circ)) - x_1^\circ \tag{5.4}$$

and the problem seems to be solved.

However, if we take $\omega(x_1) = 1$ and $\psi(x_2) = 1/(1 + x_2^2)$ and use `psolve` we get

```
> n := 2:
> x := array(1..n):
> f := array(1..n,1..1,[ [1], [1/(1+x[2]**2)] ]):
> xinit := {seq (x[‘i‘]=x0[‘i‘],i=1..n)}:
> psolve(f,x,xinit);
    Warning, (in psolve) MAPLE failed in testing the inverse mapping of F,
                                  ...
        for component, 2, although this inverse was computed by solve.

              3       2                        2             3
   [ RootOf(_Z  + 3 _Z  x0[2] + (3 + 3 x0[2] ) _Z + x0[2]  + 3 x[1]

                                          3
     + 3 x0[2] - 3 x0[1] - 3 x[2] - x[2] ) ]
```

The last two lines form the answer, in a kind of implicit form containing the dummy variable `_Z` instead of in the desired explicit form.

It is easy to see where things go awry. The solution of

$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = \frac{1}{1 + x_2^2}, \qquad x_2(0) = x_2^\circ$$

is given by

$$\frac{1}{3}x_2^3 + x_2 = t + \frac{1}{3}(x_2^\circ)^3 + x_2^\circ.$$

To compute Ψ^{-1} the cubic equation

$$\frac{1}{3}x_2^3 + x_2 - (t + \frac{1}{3}(x_2^\circ)^3 + x_2^\circ) = 0$$

has to be solved for x_2. It has three solutions, and only one of those is real, namely

$$x_2 = \left(\frac{\sqrt{4+9p^2}+3p}{2} \right)^{\frac{1}{3}} - \left(\frac{\sqrt{4+9p^2}-3p}{2} \right)^{\frac{1}{3}},$$

where

$$p = t + \frac{1}{3}(x_2^\circ)^3 + x_2^\circ.$$

The cubic equation for Ψ^{-1} leads to the cubic root in the answer generated by Maple. The program does not choose one particular root of the equation and therefore does not generate an explicit answer. The obstruction can be removed in this particular case. One could add a few extra statements to select the real solution, using the `allvalues` command to get the three explicit solutions. This would be an *ad hoc* remedy, enlarging the class of solvable problems only slightly. Also, although the real root of the answer given above by Maple is correct, it is more complicated than the answer, $h(x_1, x_2) = x_1 - \frac{1}{3}x_2^3 - x_2 - \frac{1}{3}(x_2^\circ)^3 - x_2^\circ - x_1^\circ$, given by (5.4). Here too, the simplest answer is not generated.

6. Conclusions and Recommendations

An algorithm has been devised and implemented as a procedure in Maple to solve some partial differential equations.

The main conclusion is that rather elementary and some more involved partial differential equations can be solved, but larger and more complicated ones cannot be completely tackled yet.

Possible approaches to remedy this situation, at least for the users of CAS, are to:

1. improve Maple's `dsolve` and `solve` procedures;
2. investigate if other algorithms for solving the PDE's, e.g., based on the method of characteristics, are amenable to implementation and check if they provide more powerful tools or are less taxing for the CAS;
3. provide a standard Maple procedure for solving as many PDE's as possible.

After a quick look into the capabilities of some other CAS we expect that our experiences and main conclusion do hold for those systems as well. Probably, they show the same type of shortcomings and may need the same remedies as suggested for Maple. This is also evident in Postel and Zimmermann (1996).

Acknowledgments

Harm van Essen implemented an initial version of `psolve` in the NONCON package† (van Essen, 1992; van Essen and de Jager, 1993; de Jager, 1995). Some crucial points in the implementation have been modified significantly, however, since then.

† Available on request from the first author.

References

de Jager, B. (1995). The use of symbolic computation in nonlinear control: Is it viable?, *IEEE Trans. Automat. Control* **40**(1), 84–89.

Geddes, K.O., Czapor, S.R., Labahn, G. (1992). *Algorithms for Computer Algebra*, Boston, Kluwer Academic Publishers.

Isidori, A. (1989). *Nonlinear Control Systems: An Introduction*, 2nd edn, Berlin, Springer-Verlag.

Kevorkian, J. (1993). *Partial Differential Equations: Analytical Solution Techniques*, New York, Chapman & Hall. Second printing.

MacCallum, M.A.H. (1995). Using computer algebra to solve ordinary differential equations. In Cohen, A.M., van Gastel, L. and Verduyn Lunel, S. (eds), *Computer Algebra in Industry 2: Problem Solving in Practice*, Chichester, John Wiley & Sons, pp. 19–41.

Postel, F., Zimmermann, P. (1996). A review of the ODE solvers of Axiom, Derive, Macsyma, Maple, Mathematica, MuPAD and Reduce. Preprint, presented at the 5th Rhine Workshop on Computer Algebra.

Sneddon, I. (1957). *Elements of Partial Differential Equations*, New York, McGraw-Hill Book Company.

Stephani, H. (1989). *Differential equations: Their solution using symmetries*, Cambridge, Cambridge University Press.

van Essen, H. (1992). *Symbols speak louder than numbers: Analysis and design of nonlinear control systems with the symbolic computation system MAPLE*, MSc thesis, Eindhoven University of Technology, Fac. of Mechanical Engineering. Report WFW 92.061.

van Essen, H., de Jager, B. (1993). Analysis and design of nonlinear control systems with the symbolic computation system Maple. In Nieuwenhuis, J.W., Praagman, C. and Trentelman, H.L. (eds), *Proc. of the second European Control Conf.*, Vol. 4, Groningen, The Netherlands, pp. 2081–2085.