16.2 APPLYSYM: Infinitesimal symmetries of differential equations

This package provides programs APPLYSYM, QUASILINPDE and DETRAFO for applying infinitesimal symmetries of differential equations, the generalization of special solutions and the calculation of symmetry and similarity variables.

Author: Thomas Wolf.

In this paper the programs `APPLYSYM`, `QUASILINPDE` and `DETRAFO` are described which aim at the utilization of infinitesimal symmetries of differential equations. The purpose of `QUASILINPDE` is the general solution of quasilinear PDEs. This procedure is used by `APPLYSYM` for the application of point symmetries for either

- calculating similarity variables to perform a point transformation which lowers the order of an ODE or effectively reduces the number of explicitly occuring independent variables in a PDE(-system) or for
- generalizing given special solutions of ODEs / PDEs with new constant parameters.

The program `DETRAFO` performs arbitrary point- and contact transformations of ODEs / PDEs and is applied if similarity and symmetry variables have been found. The program `APPLYSYM` is used in connection with the program `LIEPDE` for formulating and solving the conditions for point- and contact symmetries which is described in [4]. The actual problem solving is done in all these programs through a call to the package `CRACK` for solving overdetermined PDE-systems.

16.2.1 Introduction and overview of the symmetry method

The investigation of infinitesimal symmetries of differential equations (DEs) with computer algebra programs attrackted considerable attention over the last years. Corresponding programs are available in all major computer algebra systems. In a review article by W. Hereman [1] about 200 references are given, many of them describing related software.

One reason for the popularity of the symmetry method is the fact that Sophus Lie's method [2],[3] is the most widely used method for computing exact solutions of non-linear DEs. Another reason is that the first step in this method, the formulation of the determining equation for the generators of the symmetries, can already be very cumbersome, especially in the case of PDEs of higher order and/or in case of many dependent and independent variables. Also, the formulation of the conditions is a straight forward task involving only differentiations and basic algebra - an ideal task for computer algebra systems. Less straight forward is the automatic solution of the symmetry conditions which is the strength of the program `LIEPDE` (for a comparison with another program see [4]).

The novelty described in this paper are programs aiming at the final third step: Applying symmetries for

- calculating similarity variables to perform a point transformation which lowers the order of an ODE or effectively reduces the number of explicitly occuring independent variables of a PDE(-system) or for
- generalizing given special solutions of ODEs/PDEs with new constant parameters.

Programs which run on their own but also allow interactive user control are indispensible for these calculations. On one hand the calculations can become quite lengthy, like variable transformations of PDEs (of higher order, with many variables). On the other hand the freedom of choosing the right linear combination of symmetries and choosing the optimal new symmetry- and similarity variables makes it necessary to 'play' with the problem interactively.

The focus in this paper is directed on questions of implementation and efficiency, no principally new mathematics is presented.

In the following subsections a review of the first two steps of the symmetry method is given as well as the third, i.e. the application step is outlined. Each of the remaining sections is devoted to one procedure.

The first step: Formulating the symmetry conditions

To obey classical Lie-symmetries, differential equations

$$H_A = 0 \tag{16.1}$$

for unknown functions $y^\alpha, \quad 1 \leq \alpha \leq p$ of independent variables $x^i, \quad 1 \leq i \leq q$ must be forminvariant against infinitesimal transformations

$$\tilde{x}^i = x^i + \varepsilon\xi^i, \qquad \tilde{y}^\alpha = y^\alpha + \varepsilon\eta^\alpha \tag{16.2}$$

in first order of ε. To transform the equations (16.1) by (16.2), derivatives of y^α must be transformed, i.e. the part linear in ε must be determined. The corresponding formulas are (see e.g. [10], [20])

$$\begin{aligned}
\tilde{y}^\alpha_{j_1\ldots j_k} &= y^\alpha_{j_1\ldots j_k} + \varepsilon\eta^\alpha_{j_1\ldots j_k} + O(\varepsilon^2) \\
\eta^\alpha_{j_1\ldots j_{k-1}j_k} &= \frac{D\eta^\alpha_{j_1\ldots j_{k-1}}}{Dx^k} - y^\alpha_{ij_1\ldots j_{k-1}}\frac{D\xi^i}{Dx^k}
\end{aligned} \tag{16.3}$$

where D/Dx^k means total differentiation w.r.t. x^k and from now on lower latin indices of functions y^α, (and later u^α) denote partial differentiation w.r.t. the independent variables x^i, (and later v^i). The complete symmetry condition then takes the form

$$XH_A = 0 \quad \text{mod} \quad H_A = 0 \tag{16.4}$$
$$X = \xi^i \frac{\partial}{\partial x^i} + \eta^\alpha \frac{\partial}{\partial y^\alpha} + \eta^\alpha_m \frac{\partial}{\partial y^\alpha_m} + \eta^\alpha_{mn} \frac{\partial}{\partial y^\alpha_{mn}} + \ldots + \eta^\alpha_{mn\ldots p} \frac{\partial}{\partial y^\alpha_{mn\ldots p}} \tag{16.5}$$

where mod $H_A = 0$ means that the original PDE-system is used to replace some partial derivatives of y^α to reduce the number of independent variables, because the symmetry condition (16.4) must be fulfilled identically in x^i, y^α and all partial derivatives of y^α.

For point symmetries, ξ^i, η^α are functions of x^j, y^β and for contact symmetries they depend on x^j, y^β and y^β_k. We restrict ourself to point symmetries as those are the only ones that can be applied by the current version of the program APPLYSYM (see below). For literature about generalized symmetries see [1].

Though the formulation of the symmetry conditions (16.4), (16.5), (16.3) is straightforward and handled in principle by all related programs [1], the computational effort to formulate the conditions (16.4) may cause problems if the number of x^i and y^α is high. This can partially be avoided if at first only a few conditions are formulated and solved such that the remaining ones are much shorter and quicker to formulate.

A first step in this direction is to investigate one PDE $H_A = 0$ after another, as done in [22]. Two methods to partition the conditions for a single PDE are described by Bocharov/Bronstein [9] and Stephani [20].

In the first method only those terms of the symmetry condition $XH_A = 0$ are calculated which contain at least a derivative of y^α of a minimal order m. Setting coefficients of these u-derivatives to zero provides symmetry conditions. Lowering the minimal order m successively then gradually provides all symmetry conditions.

The second method is even more selective. If H_A is of order n then only terms of the symmetry condition $XH_A = 0$ are generated which contain n'th order derivatives of y^α. Furthermore these derivatives must not occur in H_A itself. They can therefore occur in the symmetry condition (16.4) only in $\eta^\alpha_{j_1\ldots j_n}$, i.e. in the terms

$$\eta^\alpha_{j_1\ldots j_n} \frac{\partial H_A}{\partial y^\alpha_{j_1\ldots j_n}}.$$

If only coefficients of n'th order derivatives of y^α need to be accurate to formulate preliminary conditions then from the total derivatives to be taken in (16.3) only that part is performed which differentiates w.r.t. the highest y^α-derivatives. This means, for example, to form only $y^\alpha_{mnk}\partial/\partial y^\alpha_{mn}$ if the expression, which is to be differentiated totally w.r.t. x^k, contains at most second order derivatives of y^α.

The second method is applied in `LIEPDE`. Already the formulation of the remaining conditions is speeded up considerably through this iteration process. These methods can be applied if systems of DEs or single PDEs of at least second order are investigated concerning symmetries.

The second step: Solving the symmetry conditions

The second step in applying the whole method consists in solving the determining conditions (16.4), (16.5), (16.3) which are linear homogeneous PDEs for ξ^i, η^α. The complete solution of this system is not algorithmic any more because the solution of a general linear PDE-system is as difficult as the solution of its non-linear characteristic ODE-system which is not covered by algorithms so far.

Still algorithms are used successfully to simplify the PDE-system by calculating its standard normal form and by integrating exact PDEs if they turn up in this simplification process [4]. One problem in this respect, for example, concerns the optimization of the symbiosis of both algorithms. By that we mean the ranking of priorities between integrating, adding integrability conditions and doing simplifications by substitutions - all depending on the length of expressions and the overall structure of the PDE-system. Also the extension of the class of PDEs which can be integrated exactly is a problem to be pursuit further.

The program `LIEPDE` which formulates the symmetry conditions calls the program `CRACK` to solve them. This is done in a number of successive calls in order to formulate and solve some first order PDEs of the overdetermined system first and use their solution to formulate and solve the next subset of conditions as described in the previous subsection. Also, `LIEPDE` can work on DEs that contain parametric constants and parametric functions. An ansatz for the symmetry generators can be formulated. For more details see [4] or [17].

The procedure `LIEPDE` is called through
`LIEPDE(`*problem,symtype,flist,inequ*`);`
All parameters are lists.

The first parameter specifies the DEs to be investigated:
problem has the form {*equations, ulist, xlist*} where

equations is a list of equations, each has the form `df(ui,..)=...` where the LHS (left hand side) `df(ui,..)` is selected such that
- The RHS (right h.s.) of an equations must not include the derivative on the LHS nor a derivative of it.
- Neither the LHS nor any derivative of it of any equation may occur in any other equation.
- Each of the unknown functions occurs on the LHS of exactly one equation.

ulist	is a list of function names, which can be chosen freely
xlist	is a list of variable names, which can be chosen freely

Equations can be given as a list of single differential expressions and then the program will try to bring them into the 'solved form' `df(ui,..)=...` automatically. If equations are given in the solved form then the above conditions are checked and execution is stopped it they are not satisfied. An easy way to get the equations in the desired form is to use

`FIRST SOLVE({`*eq1,eq2,...*`},{`*one highest derivative for each function u*`})`
(see the example of the Karpman equations in `LIEPDE.TST`). The example of the Burgers equation in `LIEPDE.TST` demonstrates that the number of symmetries for a given maximal order of the infinitesimal generators depends on the derivative chosen for the LHS.

The second parameter *symtype* of `LIEPDE` is a list { } that specifies the symmetry to be calculated. *symtype* can have the following values and meanings:

`{"point"}`	Point symmetries with $\xi^i = \xi^i(x^j, u^\beta),\ \eta^\alpha = \eta^\alpha(x^j, u^\beta)$ are determined.
`{"contact"}`	Contact symmetries with $\xi^i = 0,\ \eta = \eta(x^j, u, u_k)$ are determined ($u_k = \partial u/\partial x^k$), which is only applicable if a single equation (16.1) with an order > 1 for a single function u is to be investigated. (The *symtype* `{"contact"}` is equivalent to `{"general",1}` (see below) apart from the additional checks done for `{"contact"}`.)
`{"general",`*order*`}`	where *order* is an integer > 0. Generalized symmetries $\xi^i = 0$, $\eta^\alpha = \eta^\alpha(x^j, u^\beta, \ldots, u^\beta_K)$ of a specified order are determined (where $_K$ is a multiple index representing *order* many indices.) NOTE: Characteristic functions of generalized symmetries ($= \eta^\alpha$ if $\xi^i = 0$) are equivalent if they are equal on the solution manifold. Therefore, all dependences of characteristic functions on the substituted derivatives and their derivatives are dropped. For example, if the heat equation is given as $u_t = u_{xx}$ (i.e. u_t is substituted by u_{xx}) then `{"general",2}` would not include characteristic functions depending on u_{tx} or u_{xxx}. THEREFORE: If you want to find *all* symmetries up to a given order then either - avoid using $H_A = 0$ to substitute lower order derivatives by expressions involving higher derivatives, or - increase the order specified in *symtype*. For an illustration of this effect see the two symmetry determinations of the Burgers equation in the file

LIEPDE.TST.

`{xi!_x1 =...,...,`
` eta!_u1=...,...}`

It is possible to specify an ansatz for the symmetry. Such an ansatz must specify all ξ^i for all independent variables and all η^α for all dependent variables in terms of differential expressions which may involve unknown functions/constants. The dependences of the unknown functions have to be declared in advance by using the DEPEND command. For example,

```
DEPEND f, t, x, u$
```

specifies f to be a function of t, x, u. If one wants to have f as a function of derivatives of $u(t, x)$, say f depending on u_{txx}, then one *cannot* write

```
DEPEND f, df(u,t,x,2)$
```

but instead must write

```
DEPEND f, u!`1!`2!`2$
```

assuming *xlist* has been specified as `{t,x}`. Because t is the first variable and x is the second variable in *xlist* and u is differentiated oncs wrt. t and twice wrt. x we therefore use `u!`1!`2!`2`. The character `!` is the escape character to allow special characters like ‘ to occur in an identifier.

For generalized symmetries one usually sets all $\xi^i = 0$. Then the η^α are equal to the characteristic functions.

The third parameter *flist* of LIEPDE is a list { } that includes

- all parameters and functions in the equations which are to be determined such that symmetries exist (if any such parameters/functions are specified in *flist* then the symmetry conditions formulated in LIEPDE become non-linear conditions which may be much harder for CRACK to solve with many cases and subcases to be considered.)
- all unknown functions and constants in the ansatz `xi!_..` and `eta!_..` if that has been specified in *symtype*.

The fourth parameter *inequ* of LIEPDE is a list { } that includes all non-vanishing expressions which represent inequalities for the functions in flist.

The result of LIEPDE is a list with 3 elements, each of which is a list:

$$\{\{con_1, con_2, \ldots\}, \{\texttt{xi\_}_{\ldots} = \ldots, \ldots, \texttt{eta\_}_{\ldots} = \ldots, \ldots\}, \{flist\}\}.$$

The first list contains remaining unsolved symmetry conditions con_i. It is the empty list {} if all conditions have been solved. The second list gives the symmetry generators, i.e. expressions for ξ_i and η_j. The last list contains all free constants and functions occuring in the first and second list.

The third step: Application of infinitesimal symmetries

If infinitesimal symmetries have been found then the program APPLYSYM can use them for the following purposes:

1. Calculation of one symmetry variable and further similarity variables. After transforming the DE(-system) to these variables, the symmetry variable will not occur explicitly any more. For ODEs this has the consequence that their order has effectively been reduced.
2. Generalization of a special solution by one or more constants of integration.

Both methods are described in the following section.

16.2.2 Applying symmetries with APPLYSYM

The first mode: Calculation of similarity and symmetry variables

In the following we assume that a symmetry generator X, given in (16.5), is known such that ODE(s)/PDE(s) $H_A = 0$ satisfy the symmetry condition (16.4). The aim is to find new dependent functions $u^\alpha = u^\alpha(x^j, y^\beta)$ and new independent variables $v^i = v^i(x^j, y^\beta), \quad 1 \leq \alpha, \beta \leq p, \; 1 \leq i, j \leq q$ such that the symmetry generator $X = \xi^i(x^j, y^\beta)\partial_{x^i} + \eta^\alpha(x^j, y^\beta)\partial_{y^\alpha}$ transforms to

$$X = \partial_{v^1}. \tag{16.6}$$

Inverting the above transformation to $x^i = x^i(v^j, u^\beta), y^\alpha = y^\alpha(v^j, u^\beta)$ and setting $H_A(x^i(v^j, u^\beta), y^\alpha(v^j, u^\beta), \ldots) = h_A(v^j, u^\beta, \ldots)$ this means that

$$\begin{aligned} 0 &= XH_A(x^i, y^\alpha, y^\beta_j, \ldots) \quad \text{mod} \quad H_A = 0 \\ &= Xh_A(v^i, u^\alpha, u^\beta_j, \ldots) \quad \text{mod} \quad h_A = 0 \\ &= \partial_{v^1} h_A(v^i, u^\alpha, u^\beta_j, \ldots) \quad \text{mod} \quad h_A = 0. \end{aligned}$$

Consequently, the variable v^1 does not occur explicitly in h_A. In the case of an ODE(-system) ($v^1 = v$) the new equations $0 = h_A(v, u^\alpha, du^\beta/dv, \ldots)$ are then of lower total order after the transformation $z = z(u^1) = du^1/dv$ with now $z, u^2, \ldots u^p$ as unknown functions and u^1 as independent variable.

The new form (16.6) of X leads directly to conditions for the symmetry variable v^1 and the similarity variables $v^i|_{i\neq 1}, u^\alpha$ (all functions of x^k, y^γ):

$$Xv^1 = 1 \quad = \quad \xi^i(x^k, y^\gamma)\partial_{x^i} v^1 + \eta^\alpha(x^k, y^\gamma)\partial_{y^\alpha} v^1 \tag{16.7}$$

$$Xv^j|_{j\neq 1} = Xu^\beta = 0 \quad = \quad \xi^i(x^k, y^\gamma)\partial_{x^i} u^\beta + \eta^\alpha(x^k, y^\gamma)\partial_{y^\alpha} u^\beta \tag{16.8}$$

The general solutions of (16.7), (16.8) involve free functions of $p+q-1$ arguments. From the general solution of equation (16.8), $p+q-1$ functionally independent special solutions have to be selected ($v^2, \ldots, v^p$ and $u^1, \ldots, u^q$), whereas from (16.7) only one solution v^1 is needed. Together, the expressions for the symmetry and similarity variables must define a non-singular transformation $x, y \to u, v$.

Different special solutions selected at this stage will result in different resulting DEs which are equivalent under point transformations but may look quite differently. A transformation that is more difficult than another one will in general only complicate the new DE(s) compared with the simpler transformation. We therefore seek the simplest possible special solutions of (16.7), (16.8). They also have to be simple because the transformation has to be inverted to solve for the old variables in order to do the transformations.

The following steps are performed in the corresponding mode of the program `APPLYSYM`:

- The user is asked to specify a symmetry by selecting one symmetry from all the known symmetries or by specifying a linear combination of them.
- Through a call of the procedure `QUASILINPDE` (described in a later section) the two linear first order PDEs (16.7), (16.8) are investigated and, if possible, solved.
- From the general solution of (16.7) 1 special solution is selected and from (16.8) $p+q-1$ special solutions are selected which should be as simple as possible.
- The user is asked whether the symmetry variable should be one of the independent variables (as it has been assumed so far) or one of the new functions (then only derivatives of this function and not the function itself turn up in the new DE(s)).
- Through a call of the procedure `DETRAFO` the transformation $x^i, y^\alpha \to v^j, u^\beta$ of the DE(s) $H_A = 0$ is finally done.
- The program returns to the starting menu.

The second mode: Generalization of special solutions

A second application of infinitesimal symmetries is the generalization of a known special solution given in implicit form through $0 = F(x^i, y^\alpha)$. If one knows a symmetry variable v^1 and similarity variables v^r, u^α, $\quad 2 \le r \le p$ then v^1 can be shifted by a constant c because of $\partial_{v^1} H_A = 0$ and therefore the DEs $0 = H_A(v^r, u^\alpha, u^\beta_j, \ldots)$ are unaffected by the shift. Hence from

$$0 = F(x^i, y^\alpha) = F(x^i(v^j, u^\beta), y^\alpha(v^j, u^\beta)) = \bar{F}(v^j, u^\beta)$$

follows that

$$0 = \bar{F}(v^1 + c, v^r, u^\beta) = \bar{F}(v^1(x^i, y^\alpha) + c, v^r(x^i, y^\alpha), u^\beta(x^i, y^\alpha))$$

defines implicitly a generalized solution $y^\alpha = y^\alpha(x^i, c)$.

This generalization works only if $\partial_{v^1}\bar{F} \neq 0$ and if $\bar{F}$ does not already have a constant additive to v^1.

The method above needs to know $x^i = x^i(u^\beta, v^j),\ y^\alpha = y^\alpha(u^\beta, v^j)$ <u>and</u> $u^\alpha = u^\alpha(x^j, y^\beta), v^\alpha = v^\alpha(x^j, y^\beta)$ which may be practically impossible. Better is, to integrate x^i, y^α along X:

$$\frac{d\bar{x}^i}{d\varepsilon} = \xi^i(\bar{x}^j(\varepsilon), \bar{y}^\beta(\varepsilon)), \qquad \frac{d\bar{y}^\alpha}{d\varepsilon} = \eta^\alpha(\bar{x}^j(\varepsilon), \bar{y}^\beta(\varepsilon)) \tag{16.9}$$

with initial values $\bar{x}^i = x^i, \bar{y}^\alpha = y^\alpha$ for $\varepsilon = 0$. (This ODE-system is the characteristic system of (16.8).)

Knowing only the finite transformations

$$\bar{x}^i = \bar{x}^i(x^j, y^\beta, \varepsilon), \quad \bar{y}^\alpha = \bar{y}^\alpha(x^j, y^\beta, \varepsilon) \tag{16.10}$$

gives immediately the inverse transformation $\bar{x}^i = \bar{x}^i(x^j, y^\beta, \varepsilon), \quad \bar{y}^\alpha = \bar{y}^\alpha(x^j, y^\beta, \varepsilon)$ just by $\varepsilon \to -\varepsilon$ and renaming $x^i, y^\alpha \leftrightarrow \bar{x}^i, \bar{y}^\alpha$.

The special solution $0 = F(x^i, y^\alpha)$ is generalized by the new constant ε through

$$0 = F(x^i, y^\alpha) = F(x^i(\bar{x}^j, \bar{y}^\beta, \varepsilon), y^\alpha(\bar{x}^j, \bar{y}^\beta, \varepsilon))$$

after dropping the $^-$.

The steps performed in the corresponding mode of the program `APPLYSYM` show features of both techniques:

- The user is asked to specify a symmetry by selecting one symmetry from all the known symmetries or by specifying a linear combination of them.
- The special solution to be generalized and the name of the new constant have to be put in.
- Through a call of the procedure `QUASILINPDE`, the PDE (16.7) is solved which amounts to a solution of its characteristic ODE system (16.9) where $v^1 = \varepsilon$.
- `QUASILINPDE` returns a list of constant expressions

  $$c_i = c_i(x^k, y^\beta, \varepsilon), \quad 1 \leq i \leq p + q \tag{16.11}$$

  which are solved for $x^j = x^j(c_i, \varepsilon), \quad y^\alpha = y^\alpha(c_i, \varepsilon)$ to obtain the generalized solution through

  $$0 = F(x^j, y^\alpha) = F(x^j(c_i(x^k, y^\beta, 0), \varepsilon), y^\alpha(c_i(x^k, y^\beta, 0), \varepsilon)).$$

- The new solution is availabe for further generalizations w.r.t. other symmetries.

If one would like to generalize a given special solution with m new constants because m symmetries are known, then one could run the whole program m times, each time with a different symmetry or one could run the program once with a linear combination of m symmetry generators which again is a symmetry generator. Running the program once adds one constant but we have in addition $m - 1$ arbitrary constants in the linear combination of the symmetries, so m new constants are added. Usually one will generalize the solution gradually to make solving (16.9) gradually more difficult.

Syntax

The call of `APPLYSYM` is `APPLYSYM`({*de*, *fun*, *var*}, {*sym*, *cons*});

- *de* is a single DE or a list of DEs in the form of a vanishing expression or in the form $\ldots = \ldots$.
- *fun* is the single function or the list of functions occuring in *de*.
- *var* is the single variable or the list of variables in *de*.
- *sym* is a linear combination of all symmetries, each with a different constant coefficient, in form of a list of the ξ^i and η^α: {xi_...=...,...,eta_...=...,...}, where the indices after 'xi_' are the variable names and after 'eta_' the function names.
- *cons* is the list of constants in *sym*, one constant for each symmetry.

The list that is the first argument of `APPLYSYM` is the same as the first argument of `LIEPDE` and the second argument is the list that `LIEPDE` returns without its first element (the unsolved conditions). An example is given below.

What `APPLYSYM` returns depends on the last performed modus. After modus 1 the return is
{{*newde*, *newfun*, *newvar*}, *trafo*}
where

- *newde* lists the transformed equation(s)
- *newfun* lists the new function name(s)
- *newvar* lists the new variable name(s)
- *trafo* lists the transformations $x^i = x^i(v^j, u^\beta), y^\alpha = y^\alpha(v^j, u^\beta)$

After modus 2, `APPLYSYM` returns the generalized special solution.

Example: A second order ODE

Weyl's class of solutions of Einsteins field equations consists of axialsymmetric time independent metrics of the form

$$\mathrm{d}s^2 = e^{-2U}\left[e^{2k}\left(\mathrm{d}\rho^2 + \mathrm{dz}^2\right) + \rho^2\mathrm{d}\varphi^2\right] - e^{2U}\mathrm{dt}^2, \qquad (16.12)$$

where U and k are functions of ρ and z. If one is interested in generalizing these solutions to have a time dependence then the resulting DEs can be transformed such that one longer third order ODE for U results which contains only ρ derivatives [23]. Because U appears not alone but only as derivative, a substitution

$$g = dU/d\rho \qquad (16.13)$$

lowers the order and the introduction of a function

$$h = \rho g - 1 \qquad (16.14)$$

simplifies the ODE to

$$0 = 3\rho^2 h\, h'' - 5\rho^2\, h'^2 + 5\rho\, h\, h' - 20\rho\, h^3 h' - 20\, h^4 + 16\, h^6 + 4\, h^2. \qquad (16.15)$$

where $' = d/d\rho$. Calling `LIEPDE` through

```
depend h,r;
prob:={{-20*h**4+16*h**6+3*r**2*h*df(h,r,2)+5*r*h*df(h,r)
         -20*h**3*r*df(h,r)+4*h**2-5*r**2*df(h,r)**2},
        {h}, {r}};
sym:=liepde(prob, {"point"},{},{});
end;
```

gives

```
                                3                            2
sym := {{}, {xi_r= - c10*r   - c11*r, eta_h=c10*h*r }, {c10,c11}}.
```

All conditions have been solved because the first element of `sym` is $\{\}$. The two existing symmetries are therefore

$$-\rho^3\partial_\rho + h\rho^2\,\partial_h \qquad \text{and} \qquad \rho\partial_\rho. \qquad (16.16)$$

Corresponding finite transformations can be calculated with `APPLYSYM` through

```
newde:=applysym(prob,rest sym);
```

The interactive session is given below with the user input following the prompt '`Input:3:`' or following '?'. (Empty lines have been deleted.)

```
Do you want to find similarity and symmetry variables (enter `1;')
or generalize a special solution with new parameters  (enter `2;')
or exit the program                                   (enter  `;')
Input:3: 1;
```

We enter '1;' because we want to reduce dependencies by finding similarity variables and one symmetry variable and then doing the transformation such that the symmetry variable does not explicitly occur in the DE.

```
----------------------   The 1.  symmetry is:
        3
xi_r= - r
        2
eta_h=h*r
----------------------   The 2.  symmetry is:
xi_r= - r
----------------------
Which single symmetry or linear combination of symmetries
do you want to apply?
Enter an expression with `sy_(i)' for the i'th symmetry.
sy_(1);
```

We could have entered 'sy_(2);' or a combination of both as well with the calculation running then differently.

```
The symmetry to be applied in the following is
          3         2
{xi_r= - r ,eta_h=h*r }
Enter the name of the new dependent variables:
Input:3: u;
Enter the name of the new independent variables:
Input:3: v;
```

This was the input part, now the real calculation starts.

```
The ODE/PDE (-system) under investigation is :
                   2            2  2               3
0 = 3*df(h,r,2)*h*r  - 5*df(h,r) *r  - 20*df(h,r)*h *r
                           6       4      2
     + 5*df(h,r)*h*r + 16*h  - 20*h  + 4*h
for the function(s) : h.
It will be looked for a new dependent variable u
and an independent variable v such that the transformed
de(-system) does not depend on u or v.
1. Determination of the similarity variable
                           2
The quasilinear PDE:  0 = r *(df(u_,h)*h - df(u_,r)*r).
The equivalent characteristic system:
               3
```

```
0= - df(u_,r)*r
        2
0= - r *(df(h,r)*r + h)
for the functions: h(r)  u_(r).
```

The PDE is equation (16.8).

```
The general solution of the PDE is given through
0 = ff(u_,h*r)
with arbitrary function ff(..).
A suggestion for this function ff provides:
0 =  - h*r + u_
Do you like this choice? (Y or N)
?y
```

For the following calculation only a single special solution of the PDE is necessary and this has to be specified from the general solution by choosing a special function `ff`. (This function is called `ff` to prevent a clash with names of user variables/functions.) In principle any choice of `ff` would work, if it defines a non-singular coordinate transformation, i.e. here r must be a function of $u\_$. If we have q independent variables and p functions of them then `ff` has $p+q$ arguments. Because of the condition 0 =`ff` one has essentially the freedom of choosing a function of $p+q-1$ arguments freely. This freedom is also necessary to select $p+q-1$ different functions `ff` and to find as many functionally independent solutions $u\_$ which all become the new similarity variables. q of them become the new functions u^α and $p-1$ of them the new variables $v^2,\ldots,v^p$. Here we have $p=q=1$ (one single ODE).

Though the program could have done that alone, once the general solution `ff(..)` is known, the user can interfere here to enter a simpler solution, if possible.

```
2. Determination of the symmetry variable
                                     2              3
The quasilinear PDE:  0 = df(u_,h)*h*r  - df(u_,r)*r  - 1.
The equivalent characteristic system:
             3
0=df(r,u_) + r
               2
0=df(h,u_) - h*r
for the functions: r(u_)  h(u_)   .
New attempt with a different independent variable
The equivalent characteristic system:
             2
0=df(u_,h)*h*r  - 1
   2
0=r *(df(r,h)*h + r)
for the functions: r(h)  u_(h)   .
The general solution of the PDE is given through
```

```
                  2  2       2
             - 2*h *r *u_ + h
0 = ff(h*r,--------------------)
                    2
with arbitrary function ff(..).
A suggestion for this function ff(..) yields:
      2       2
     h *( - 2*r *u_ + 1)
0 = ---------------------
               2
Do you like this choice? (Y or N)
?y
```

Similar to above.

```
The suggested solution of the algebraic system which will
do the transformation is:
                        sqrt(v)*sqrt(2)
{h=sqrt(v)*sqrt(2)*u,r=-----------------}
                             2*v
Is the solution ok? (Y or N)
?y
In the intended transformation shown above the dependent
variable is u and the independent variable is v.
The symmetry variable is v, i.e. the transformed expression
will be free of v.
Is this selection of dependent and independent variables ok? (Y or N)
?n
```

We so far assumed that the symmetry variable is one of the new variables, but, of course we also could choose it to be one of the new functions. If it is one of the functions then only derivatives of this function occur in the new DE, not the function itself. If it is one of the variables then this variable will not occur explicitly.

In our case we prefer (without strong reason) to have the function as symmetry variable. We therefore answered with 'no'. As a consequence, u and v will exchange names such that still all new functions have the name u and the new variables have name v:

```
Please enter a list of substitutions. For example, to
make the variable, which is so far call u1, to an
independent variable v2 and the variable, which is
so far called v2, to an dependent variable u1,
enter: `{u1=v2, v2=u1};'
Input:3: {u=v,v=u};

The transformed equation which should be free of u:
                             3  6            2  3
```

```
0=3*df(u,v,2)*v - 16*df(u,v) *v  - 20*df(u,v) *v  + 5*df(u,v)
Do you want to find similarity and symmetry variables (enter `1;')
or generalize a special solution with new parameters  (enter `2;')
or exit the program                                   (enter  `;')
Input:3: ;
```

We stop here. The following is returned from our APPLYSYM call:

```
                                 3   6                2   3
{{{3*df(u,v,2)*v - 16*df(u,v) *v  - 20*df(u,v) *v + 5*df(u,v)},
  {u},
  {v}},
     sqrt(u)*sqrt(2)
 {r=-----------------, h=sqrt(u)*sqrt(2)*v }}
           2*u
```

The use of APPLYSYM effectively provided us the finite transformation

$$\rho = (2\,u)^{-1/2}, \qquad h = (2\,u)^{1/2}\,v. \tag{16.17}$$

and the new ODE

$$0 = 3u''v - 16u'^3v^6 - 20u'^2v^3 + 5u' \tag{16.18}$$

where $u = u(v)$ and $' = d/dv$. Using one symmetry we reduced the 2. order ODE (16.15) to a first order ODE (16.18) for u' plus one integration. The second symmetry can be used to reduce the remaining ODE to an integration too by introducing a variable w through $v^3 d/dv = d/dw$, i.e. $w = -1/(2v^2)$. With

$$p = du/dw \tag{16.19}$$

the remaining ODE is

$$0 = 3\,w\,\frac{dp}{dw} + 2\,p\,(p+1)(4\,p+1)$$

with solution

$$\tilde{c}w^{-2}/4 = \tilde{c}v^4 = \frac{p^3(p+1)}{(4\,p+1)^4}, \quad \tilde{c} = const.$$

Writing (16.19) as $p = v^3(du/dp)/(dv/dp)$ we get u by integration and with (16.17) further a parametric solution for ρ, h:

$$\rho = \left(\frac{3c_1^2(2p-1)}{p^{1/2}(p+1)^{1/2}} + c_2\right)^{-1/2} \tag{16.20}$$

$$h = \frac{(c_2 p^{1/2}(p+1)^{1/2} + 6c_1^2 p - 3c_1^2)^{1/2} p^{1/2}}{c_1(4p+1)} \tag{16.21}$$

where $c_1, c_2 = const.$ and $c_1 = \tilde{c}^{1/4}$. Finally, the metric function $U(p)$ is obtained as an integral from (16.13),(16.14).

Limitations of `APPLYSYM`

Restrictions of the applicability of the program `APPLYSYM` result from limitations of the program `QUASILINPDE` described in a section below. Essentially this means that symmetry generators may only be polynomially non-linear in x^i, y^α. Though even then the solvability can not be guaranteed, the generators of Lie-symmetries are mostly very simple such that the resulting PDE (16.22) and the corresponding characteristic ODE-system have good chances to be solvable.

Apart from these limitations implied through the solution of differential equations with `CRACK` and algebraic equations with `SOLVE` the program `APPLYSYM` itself is free of restrictions, i.e. if once new versions of `CRACK, SOLVE` would be available then `APPLYSYM` would not have to be changed.

Currently, whenever a computational step could not be performed the user is informed and has the possibility of entering interactively the solution of the unsolved algebraic system or the unsolved linear PDE.

16.2.3 Solving quasilinear PDEs

The content of `QUASILINPDE`

The generalization of special solutions of DEs as well as the computation of similarity and symmetry variables involve the general solution of single first order linear PDEs. The procedure `QUASILINPDE` is a general procedure aiming at the general solution of PDEs

$$a_1(w_i,\phi)\phi_{w_1} + a_2(w_i,\phi)\phi_{w_2} + \ldots + a_n(w_i,\phi)\phi_{w_n} = b(w_i,\phi) \qquad (16.22)$$

in n independent variables $w_i, i = 1\ldots n$ for one unknown function $\phi = \phi(w_i)$.

1. The first step in solving a quasilinear PDE (16.22) is the formulation of the corresponding characteristic ODE-system

$$\frac{dw_i}{d\varepsilon} = a_i(w_j,\phi) \qquad (16.23)$$

$$\frac{d\phi}{d\varepsilon} = b(w_j,\phi) \qquad (16.24)$$

   for ϕ, w_i regarded now as functions of one variable ε.

   Because the a_i and b do not depend explicitly on ε, one of the equations (16.23),(16.24) with non-vanishing right hand side can be used to divide all others through it and by that having a system with one less ODE to solve. If the equation to divide through is one of (16.23) then the remaining system would be

$$\frac{dw_i}{dw_k} = \frac{a_i}{a_k}, \quad i = 1,2,\ldots k-1,k+1,\ldots n \qquad (16.25)$$

$$\frac{d\phi}{dw_k} = \frac{b}{a_k} \qquad (16.26)$$

   with the independent variable w_k instead of ε. If instead we divide through equation (16.24) then the remaining system would be

$$\frac{dw_i}{d\phi} = \frac{a_i}{b}, \quad i = 1,2,\ldots n \qquad (16.27)$$

with the independent variable ϕ instead of ε.

The equation to divide through is chosen by a subroutine with a heuristic to find the "simplest" non-zero right hand side (a_k or b), i.e. one which

- is constant or
- depends only on one variable or
- is a product of factors, each of which depends only on one variable.

One purpose of this division is to reduce the number of ODEs by one. Secondly, the general solution of (16.23), (16.24) involves an additive constant to ε which is not relevant and would have to be set to zero. By dividing through one ODE we eliminate ε and lose the problem of identifying this constant in the general solution before we would have to set it to zero.

2. To solve the system (16.25), (16.26) or (16.27), the procedure `CRACK` is called. Although being designed primarily for the solution of overdetermined PDE-systems, `CRACK` can also be used to solve simple not overdetermined ODE-systems. This solution process is not completely algorithmic. Improved versions of `CRACK` could be used, without making any changes of `QUASILINPDE` necessary.

   If the characteristic ODE-system can not be solved in the form (16.25), (16.26) or (16.27) then successively all other ODEs of (16.23), (16.24) with non-vanishing right hand side are used for division until one is found such that the resulting ODE-system can be solved completely. Otherwise the PDE can not be solved by `QUASILINPDE`.

3. If the characteristic ODE-system (16.23), (16.24) has been integrated completely and in full generality to the implicit solution

$$0 = G_i(\phi, w_j, c_k, \varepsilon), \quad i, k = 1, \ldots, n+1, \quad j = 1, \ldots, n \tag{16.28}$$

   then according to the general theory for solving first order PDEs, ε has to be eliminated from one of the equations and to be substituted in the others to have left n equations. Also the constant that turns up additively to ε is to be set to zero. Both tasks are automatically fulfilled, if, as described above, ε is already eliminated from the beginning by dividing all equations of (16.23), (16.24) through one of them.

   On either way one ends up with n equations

$$0 = g_i(\phi, w_j, c_k), \quad i, j, k = 1 \ldots n \tag{16.29}$$

   involving n constants c_k.

   The final step is to solve (16.29) for the c_i to obtain

$$c_i = c_i(\phi, w_1, \ldots, w_n) \qquad i = 1, \ldots n. \tag{16.30}$$

   The final solution $\phi = \phi(w_i)$ of the PDE (16.22) is then given implicitly through

$$0 = F(c_1(\phi, w_i), c_2(\phi, w_i), \ldots, c_n(\phi, w_i))$$

   where F is an arbitrary function with n arguments.

Syntax

The call of `QUASILINPDE` is
`QUASILINPDE`(*de*, *fun*, *varlist*);

- *de* is the differential expression which vanishes due to the PDE *de* $= 0$ or, *de* may be the differential equation itself in the form $\ldots = \ldots$.
- *fun* is the unknown function.
- *varlist* is the list of variables of *fun*.

The result of `QUASILINPDE` is a list of general solutions

$$\{sol_1, sol_2, \ldots\}.$$

If `QUASILINPDE` can not solve the PDE then it returns $\{\}$. Each solution sol_i is a list of expressions

$$\{ex_1, ex_2, \ldots\}$$

such that the dependent function (ϕ in (16.22)) is determined implicitly through an arbitrary function F and the algebraic equation

$$0 = F(ex_1, ex_2, \ldots).$$

Examples

Example 1:
To solve the quasilinear first order PDE

$$1 = xu_{,x} + uu_{,y} - zu_{,z}$$

for the function $u = u(x, y, z)$, the input would be

```
depend u,x,y,z;
de:=x*df(u,x)+u*df(u,y)-z*df(u,z) - 1;
varlist:={x,y,z};
QUASILINPDE(de,u,varlist);
```

In this example the procedure returns

$$\{\{x/e^u, ze^u, u^2 - 2y\}\},$$

i.e. there is one general solution (because the outer list has only one element which itself is a list) and u is given implicitly through the algebraic equation

$$0 = F(x/e^u, ze^u, u^2 - 2y)$$

with arbitrary function F.
Example 2:
For the linear inhomogeneous PDE

$$0 = yz_{,x} + xz_{,y} - 1, \quad \text{for} \quad z = z(x, y)$$

QUASILINPDE returns the result that for an arbitrary function F, the equation

$$0 = F\left(\frac{x+y}{e^z}, e^z(x-y)\right)$$

defines the general solution for z.
Example 3:
For the linear inhomogeneous PDE (3.8) from [15]

$$0 = xw,_x + (y+z)(w,_y - w,_z), \quad \text{for} \quad w = w(x,y,z)$$

QUASILINPDE returns the result that for an arbitrary function F, the equation

$$0 = F\left(w,\ y+z,\ \ln(x)(y+z) - y\right)$$

defines the general solution for w, i.e. for any function f

$$w = f\left(y+z,\ \ln(x)(y+z) - y\right)$$

solves the PDE.

Limitations of QUASILINPDE

One restriction on the applicability of QUASILINPDE results from the program CRACK which tries to solve the characteristic ODE-system of the PDE. So far CRACK can be applied only to polynomially non-linear DE's, i.e. the characteristic ODE-system (16.25),(16.26) or (16.27) may only be polynomially non-linear, i.e. in the PDE (16.22) the expressions a_i and b may only be rational in w_j, ϕ.

The task of CRACK is simplified as (16.28) does not have to be solved for w_j, ϕ. On the other hand (16.28) has to be solved for the c_i. This gives a second restriction coming from the REDUCE function SOLVE. Though SOLVE can be applied to polynomial and transzendential equations, again no guarantee for solvability can be given.

16.2.4 Transformation of DEs

The content of DETRAFO

Finally, after having found the finite transformations, the program APPLYSYM calls the procedure DETRAFO to perform the transformations. DETRAFO can also be used alone to do point- or higher order transformations which involve a considerable computational effort if the differential order of the expression to be transformed is high and if many dependent and independent variables are involved. This might be especially useful if one wants to experiment and try out different coordinate transformations interactively, using DETRAFO as standalone procedure.

To run `DETRAFO`, the old functions y^α and old variables x^i must be known explicitly in terms of algebraic or differential expressions of the new functions u^β and new variables v^j. Then for point transformations the identity

$$\begin{aligned} dy^\alpha &= \left(y^\alpha{}_{,v^i} + y^\alpha{}_{,u^\beta}\, u^\beta{}_{,v^i}\right) dv^i & (16.31)\\ &= y^\alpha{}_{,x^j}\, dx^j & (16.32)\\ &= y^\alpha{}_{,x^j} \left(x^j{}_{,v^i} + x^j{}_{,u^\beta}\, u^\beta{}_{,v^i}\right) dv^i & (16.33) \end{aligned}$$

provides the transformation

$$y^\alpha{}_{,x^j} = \frac{dy^\alpha}{dv^i} \cdot \left(\frac{dx^j}{dv^i}\right)^{-1} \tag{16.34}$$

with $det\big(dx^j/dv^i\big) \neq 0$ because of the regularity of the transformation which is checked by `DETRAFO`. Non-regular transformations are not performed.

`DETRAFO` is not restricted to point transformations. In the case of contact- or higher order transformations, the total derivatives dy^α/dv^i and dx^j/dv^i then only include all v^i- derivatives of u^β which occur in

$$\begin{aligned} y^\alpha &= y^\alpha(v^i, u^\beta, u^\beta{}_{,v^j}, \ldots) \\ x^k &= x^k(v^i, u^\beta, u^\beta{}_{,v^j}, \ldots). \end{aligned}$$

Syntax

The call of `DETRAFO` is

DETRAFO({*ex*$_1$, *ex*$_2$, …, *ex*$_m$},
{*ofun*$_1$ =*fex*$_1$, *ofun*$_2$ =*fex*$_2$, …,*ofun*$_p$ =*fex*$_p$},
{*ovar*$_1$ =*vex*$_1$, *ovar*$_2$ =*vex*$_2$, …, *ovar*$_q$ =*vex*$_q$},
{*nfun*$_1$, *nfun*$_2$, …, *nfun*$_p$},
{*nvar*$_1$, *nvar*$_2$, …, *nvar*$_q$});

where m, p, q are arbitrary.

- The *ex*$_i$ are differential expressions to be transformed.
- The second list is the list of old functions *ofun* expressed as expressions *fex* in terms of new functions *nfun* and new independent variables *nvar*.
- Similarly the third list expresses the old independent variables *ovar* as expressions *vex* in terms of new functions *nfun* and new independent variables *nvar*.

- The last two lists include the new functions *nfun* and new independent variables *nvar*.

Names for *ofun, ovar, nfun* and *nvar* can be arbitrarily chosen.

As the result `DETRAFO` returns the first argument of its input, i.e. the list

$$\{ex_1, ex_2, \ldots, ex_m\}$$

where all ex_i are transformed.

Limitations of `DETRAFO`

The only requirement is that the old independent variables x^i and old functions y^α must be given explicitly in terms of new variables v^j and new functions u^β as indicated in the syntax. Then all calculations involve only differentiations and basic algebra.

Bibliography

[1] W. Hereman, Chapter 13 in vol 3 of the CRC Handbook of Lie Group Analysis of Differential Equations, Ed.: N.H. Ibragimov, CRC Press, Boca Raton, Florida (1995). Systems described in this paper are among others:
DELiA (Alexei Bocharov et.al.) Pascal
DIFFGROB2 (Liz Mansfield) Maple
DIMSYM (James Sherring and Geoff Prince) REDUCE
HSYM (Vladimir Gerdt) Reduce
LIE (V. Eliseev, R.N. Fedorova and V.V. Kornyak) Reduce
LIE (Alan Head) muMath
Lie (Gerd Baumann) Mathematica
LIEDF/INFSYM (Peter Gragert and Paul Kersten) Reduce
Liesymm (John Carminati, John Devitt and Greg Fee) Maple
MathSym (Scott Herod) Mathematica
NUSY (Clara Nucci) Reduce
PDELIE (Peter Vafeades) Macsyma
SPDE (Fritz Schwarz) Reduce and Axiom
SYM_DE (Stanly Steinberg) Macsyma
Symmgroup.c (Dominique Berube and Marc de Montigny) Mathematica
STANDARD FORM (Gregory Reid and Alan Wittkopf) Maple
SYMCAL (Gregory Reid) Macsyma and Maple
SYMMGRP.MAX (Benoit Champagne, Willy Hereman and Pavel Winternitz) Macsyma
LIE package (Khai Vu) Maple

Toolbox for symmetries (Mark Hickman) Maple
Lie symmetries (Jeffrey Ondich and Nick Coult) Mathematica.

[2] S. Lie, Sophus Lie's 1880 Transformation Group Paper, Translated by M. Ackerman, comments by R. Hermann, Mathematical Sciences Press, Brookline, (1975).

[3] S. Lie, Differentialgleichungen, Chelsea Publishing Company, New York, (1967).

[4] T. Wolf, An efficiency improved program `LIEPDE` for determining Lie - symmetries of PDEs, Proceedings of the workshop on Modern group theory methods in Acireale (Sicily) Nov. (1992)

[5] C. Riquier, Les systèmes d'équations aux dérivées partielles, Gauthier–Villars, Paris (1910).

[6] J. Thomas, Differential Systems, AMS, Colloquium publications, v. 21, N.Y. (1937).

[7] M. Janet, Leçons sur les systèmes d'équations aux dérivées, Gauthier–Villars, Paris (1929).

[8] V.L. Topunov, Reducing Systems of Linear Differential Equations to a Passive Form, Acta Appl. Math. 16 (1989) 191–206.

[9] A.V. Bocharov and M.L. Bronstein, Efficiently Implementing Two Methods of the Geometrical Theory of Differential Equations: An Experience in Algorithm and Software Design, Acta. Appl. Math. 16 (1989) 143–166.

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

[11] G.J. Reid, A triangularization algorithm which determines the Lie symmetry algebra of any system of PDEs, J.Phys. A: Math. Gen. 23 (1990) L853-L859.

[12] F. Schwarz, Automatically Determining Symmetries of Partial Differential Equations, Computing 34, (1985) 91-106.

[13] W.I. Fushchich and V.V. Kornyak, Computer Algebra Application for Determining Lie and Lie–Bäcklund Symmetries of Differential Equations, J. Symb. Comp. 7 (1989) 611–619.

[14] E. Kamke, Differentialgleichungen, Lösungsmethoden und Lösungen, Band 1, Gewöhnliche Differentialgleichungen, Chelsea Publishing Company, New York, 1959.

[15] E. Kamke, Differentialgleichungen, Lösungsmethoden und Lösungen, Band 2, Partielle Differentialgleichungen, 6.Aufl., Teubner, Stuttgart:Teubner, 1979.

[16] T. Wolf, An Analytic Algorithm for Decoupling and Integrating systems of Nonlinear Partial Differential Equations, J. Comp. Phys., no. 3, 60 (1985) 437-446 and, Zur analytischen Untersuchung und exakten Lösung von Differentialgleichungen mit Computeralgebrasystemen, Dissertation B, Jena (1989).

[17] T. Wolf, A. Brand, The Computer Algebra Package `CRACK` for Investigating PDEs, Manual for the package `CRACK` in the REDUCE network library and in Proceedings of ERCIM School on Partial Differential Equations and Group Theory, April 1992 in Bonn, GMD Bonn.

[18] M.A.H. MacCallum, F.J. Wright, Algebraic Computing with REDUCE, Clarendon Press, Oxford (1991).

[19] M.A.H. MacCallum, An Ordinary Differential Equation Solver for REDUCE, Proc. ISAAC'88, Springer Lect. Notes in Comp Sci. 358, 196–205.

[20] H. Stephani, Differential equations, Their solution using symmetries, Cambridge University Press (1989).

[21] V.I. Karpman, Phys. Lett. A 136, 216 (1989)

[22] B. Champagne, W. Hereman and P. Winternitz, The computer calculation of Lie point symmetries of large systems of differential equations, Comp. Phys. Comm. 66, 319-340 (1991)

[23] M. Kubitza, private communication