Lie Symmetries of Differential Equations

A Mathematica *program to determine Lie–symmetries*

G. Baumann
Abteilung für Mathematische Physik,
Universität Ulm
Albert-Einstein-Allee 11,
D-7900 Ulm, Germany *

Abstract. A Mathematica program calculating Lie symmetry groups is presented. This program can be used for systems of partial as well as ordinary differential equations. Determining equations for systems of m differential equations of order k with p independent and q dependent variables can be calculated with this program. A list containing a minimal set of determining equations for the infinitesimals is produced. From the automatic solution of determining equations, the Lie symmetry group can be constructed. Some examples demonstrate the application of the program.

1 Introduction

The ideas and principles of symmetry play an important role in mathematical physics. It appears that cases of exact solutions of differential equations are based on the use of symmetry of these equations with respect to certain transformations. Symmetry analysis is one of the systematic and accurate ways to obtain solutions of differential equations. An alternative method especially for solving nonlinear partial differential equations is the inverse scattering method. This method is only applicable on completely integrable equations of motion whereas a symmetry analysis is also useful in the study of nonintegrable equations of motion. A major obstacle in the application of symmetry analysis solving differential equations is that a large number of tedious calculations is usually involved. The purpose of this article is to present a *Mathematica* program for computer assisted calculations of symmetry groups. Similar programs written in REDUCE, MACSYMA, SCRATCHPAD, etc. are available on large scale computers [5, 7, 6]. The most prominent of these programs is SPDE by F. Schwarz, originally written in REDUCE and ported to SCRATCHPAD later on [5]. A review of different programs and their programming languages is given in [6, 7].

The setting of our program in *Mathematica* is an entirely general one. Its algorithm is well known and described e.g. in the books by Olver [1] or Bluman and Kumei [2]. We consider the general case of a nonlinear system of differential equations for an arbitrary number q of unknown functions u^a which may depend on p independent variables x_i. We denote these sets of variables simply by $u = (u^1, u^2, ..., u^q)$ and $x = (x_1, x_2, ..., x_p)$, respectively. The general case is given by a

* e-mail: bau@theophys.physik.uni-ulm.de

system of m nonlinear differential equations

$$\Delta^i(x, u^{(k)}) = 0 \qquad i = 1, 2, ..., m \tag{1}$$

of order k. The term $u^{(k)}$ is understood as k-th derivative of u with respect to x. We note that m, k, p and q are arbitrary, positive integers. Consider further a one–parameter ϵ–Lie group of transformations

$$x^{'} = \Xi\ (x, u; \epsilon) \tag{2}$$

$$u^{'} = \Phi\ (x, u; \epsilon) \tag{3}$$

under which (1) must be invariant. Invariance of (1) under the action of (2,3) means that any solution $u = \Theta(x)$ of (1) maps into some other solution $v = \Psi(x; \epsilon)$ of (1). Let $u = \Theta(x)$ be a solution of (1). If we replace the dependent and independent variables u and x by v and $x' = \Xi$, equations (1) become

$$\Delta^i(x', v^{(k)}) = 0 \qquad i = 1, 2, ..., m. \tag{4}$$

Then $v = \Theta(x^{'})$ are solutions of (4). This implies that if (1) and (4) have a unique solution, then

$$\Theta(x^{'}) = \Phi(x, \Theta(x); \epsilon). \tag{5}$$

Hence Θ satisfies the one–paramenter functional equation

$$\Theta\big(\Xi(x; \epsilon)\big) = \Phi(x, \Theta; \epsilon). \tag{6}$$

Expanding equations (2) and (3) around the identity $\epsilon = 0$, one can generate the following infinitesimal transformations

$$x_i^{'} = x_i + \epsilon\ \xi_i(x, u) + O(\epsilon^2) \qquad i = 1, 2, ..., p \tag{7}$$

$$u^{'\alpha} = u^\alpha + \epsilon\ \phi_\alpha(x, u) + O(\epsilon^2) \qquad \alpha = 1, 2, ..., q. \tag{8}$$

The functions ξ_i and ϕ_α are the infinitesimals of the transformations for the independent and dependent variables, respectively. In order to find the unknown infinitesimals, we need to extend respectively prolong the transformation group to include the properties of the derivatives. It is an infinitesimal approach which considers the Lie–algebra $\mathcal{L}$ corresponding to the Lie group G. The infinitesimal transformation (7,8) can be put into the form of

$$\vec{v} = \sum_{i=1}^{p} \xi_i(x, u)\frac{\partial}{\partial x_i} + \sum_{\alpha=1}^{q} \phi_\alpha(x, u)\frac{\partial}{\partial u^\alpha} \tag{9}$$

where $\vec{v}$ represents a linear combination of the vector fields generating $\mathcal{L}$ which in turn is based on the characteristic quantities ξ_i and ϕ_α of the transformation (7,8).

The algorithm used in the *Mathematica* program for finding the infinitesimals ξ_i and ϕ_α is described below. We emphasize that the infinitesimals depend on this simple form only with respect to independent and dependent variables. A prolongation of the dependencies to derivatives extends the Lie symmetries to so–called generalized Lie symmetries. Transformations (7,8), together with the transformations for the first, second, ... derivatives of the u^α's, are called first, second, ... prolongations. Using these various extensions, the infinitesimal criterion for the invariance of (1) under the group (2,3) is derivable by

$$pr^{(k)}\vec{v}\Delta\Big|_{\Delta=0} = 0, \tag{10}$$

where the k-th prolongation of the vector field $\vec{v}$ is given by

$$pr^{(k)}\vec{v} = \vec{v} + \sum_{\alpha=1}^{q}\sum_{J} \phi_\alpha^J(x, u^{(k)})\frac{\partial}{\partial u_J^\alpha}\,. \tag{11}$$

The second summation extends over all multi–indices $J = (j_1, ..., j_l)$ with $1 \le j_l \le p, 1 \le l \le k$. The k-th prolongation coefficients ϕ_α^J are given by

$$\phi_\alpha^J(x, u^{(k)}) = D_J\left(\phi_\alpha - \sum_{i=1}^{p}\xi_i u_i^\alpha\right) + \sum_{i=1}^{p}\xi_i u_{J,i}^\alpha \tag{12}$$

where $u_i^\alpha = \frac{\partial u^\alpha}{\partial x_i}$ and $u_{J,i}^\alpha = \frac{\partial u_J^\alpha}{\partial x_i}$. Thus, the system of differential equations (1) is invariant under the transformation of a one–parameter group with the infinitesimal generator (9) if the ξ_i's and ϕ_α's are determined from equation (10). The result of the *Mathematica* program is a system of linear homogeneous partial differential equations (PDEs) for ξ_i and ϕ_α, in which x and u are independent variables. These are the so–called determining equations for the symmetries of the system. The solution algorithm for (1) thus is separated into two steps: first, derivation of the simplest form of determining equations and second, solution of the determining equations.

In Section 2, we describe the algorithm used to carry out the first and second step. The description of the actual *Mathematica* program is given in Section 3. An explanation of the use of the program and some examples are presented in Section 4. Section 5 discusses benchmarks for a family of Korteweg deVries equations. The final Section 6 is devoted to conclusions.

2 Algorithm used in *Mathematica*

The algorithm presented here follows the notation used by Olver [1]. The work of Olver is usable in a direct implementation for a computer program. Using his notations the algorithm involves the following steps:

i) Applying the prolongation operator $pr^{(k)}\vec{v}$ to each equation $\Delta^i(x, u^{(k)})$ and requiring that equation (10) is satisfied. The k–th prolongation of the vector field $\vec{v}$ in (9) is determined by means of the formula (11) where the coefficients ϕ_α^J are given by the direct formula (12). We note that we do not use a recursion formula to generate higher prolongations as the program of Champagne et al [7] preferst to do.

ii) Consider (1) as an algebraic system of equations where derivatives become the new variables. Provided this system of algebraic equations can be solved by the *Mathematica* function `Solve[]`, we can gain m solutions w^β which contain the rest of the equations Δ^i. The variables w^β with $\beta = 1, 2, \ldots, m$ are used as substitution rules or side conditions in equation (10).

iii) Obtaining the determining equations for the infinitesimals ξ_i and ϕ_α by equating to zero the coefficients of all functionally independent expressions in the remaining derivatives u_J^α.

iv) Simplifying the total number of equations by using the first order derivatives and homogeneous higher order derivatives for the infinitesimals. This step is repeated until the number of determining equations reach a certain stable value.

v) If the determining equations are known it may be possible to solve these equations by using a polynomial ansatz for the infinitesimals. We note that these step will not provide the most general solution if such one exists but delivers at least a subclass of it.

vi) If the infinitesimals are found automaticly or by using an iterative manual procedure, solving the determining equations partly and feeding back the obtained information into the program, one can use these information to derive and classify the coresponding Lie–algebra.

This procedure works expediently as long as the variables w^β are obtainable by `Solve[]` from the equations of motion (1) and if `Solve[]` can find a solution from the linear equations for the constants used in the ansatz of the infinitesimals. The manual procedure, feeding back some information on the infinitesimals gained by partly solving the determining equations is completely independent from the automatic solution procedure. If one is convinced that the ansatz used in `Lie[]` does not provide the general most solution one can use this manual and iterative procedure to find the solutions of the determining equations. The first and second step of the solution process can be completely automized if one creates two files containing the following commands

```
Lie["name",Ansatz->{order}]
```

to find the infinitesimals and

```
Lie["name",Info->True,CommutatorTable->True,-
        StructureConstants->True,
                LaTeX->True]
```

to verify the infinitesimals and do the classification of the algebra.

3 The Lie[] program

The flexibility of this program allow the user of small–scale computers to find symmetry groups of arbitrary, large and complicated systems of equations. We have used the Lie[] program extensively on a 386Pc, 486Pc, and on a NeXT computer. We also tested the program Lie[] on a HP 9000 computer. Our program in *Mathematica* is a about a factor 4 or 5 faster than a similar program in MACSYMA [7] running on a VAX 6440 computer under VMS. The MACSYMA program of [7] only provides the derivation of the determining equations but does not solve these equations automaticly it also gives no classification of the related Lie–algebra and cannot treate transzendental functions in the equations. These are the main justifications for presenting a new program in a field where several programs already exist [5]. However, up to now there is no such program available in *Mathematica*. The amount of interaction by the user is very small and can be reduced to a few key strokes if one uses two batch files to manage the calculations. Our experience from several calculations done by hand is that for systems of equations, the most time consuming part of the calculation is the derivation of the determining equations and the elimination of redundant equations from the system. The second step, the solution of the determining equations can be found by inspection, using elementary results from the theory of linear PDEs or using an ansatz for the infinitesimals. Our program is capable to manage both tasks. In the ansatz, we use a polynomial to reduce the system of linear PDEs to a linear system of algebraic equations.

The philosophy of the approach implemented in the present program is first to follow the path that would be taken in a pencil calculation or use an ansatz of polynomial form for the infinitesimals. The manual procedure is to obtain in a manner as simple as possibe all one–term equations, solve them and feed the found information back into the program. All necessary substitutions and simplifications leading to the new determining system–which are error-prone if done by hand–are carried out automatically and correctly by this program. The automatic procedure assumes that a solution of the infinitesimals exists and that this solution is of polynomial form in the dependent and independent variables. This assumption will lead at least to a subgroup of solutions but cannot handel the occurence of arbitrary functions in the group. However, in a great number of practical applications a polynomial ansatz is sufficient to find a solution for the infinitesimals.

The Lie[] program assumes that the u^α are real functions. If a system of differential equations contains complex valued dependent variables, the decomposition into real and imaginary parts must be made before using the program. In addition, the total information on the determining equations delivered by the program is logged in a file for further use at each run.

3.1 Description of the program

We used *Mathematica* 2.0 for the `Lie[]` program. This is a version of *Mathematica* which is available for many computers ranging from a Pc up to a mainframe computer. Information concerning the definition and principal features of *Mathematica* can be found in S. Wolfram's handbook [4].

The Program essentially consists of function definitions. In fact, an appropriate function definition is given in the program for each task in connection with the algorithm outlined in Section 2. The `Lie[]` function is the main program. Once called, it reads the data from an input file and goes through the algorithm by sequentially calling the necessary functions.

3.2 Main functions

`ProlongationF[]` determines the k–th prolongation of $\vec{v}$ given by equation (9) for a function depending on x and $u^{(k)}$.

`SearchCoefficients[]` Given an expression which is polynomial in the derivatives u_j^α, this function finds all the coefficients of the various partial derivatives of u and puts these coefficients in a list which is returned as a result. These coefficients are essentially the determining equations for the infinitesimals.

`SimplifyEquations[]` Given the list of determining equations in the infinitesimals ξ_i and ϕ_α, this function produces a unique list of determining equations named `Result2`, equivalent to the input list but free of repetition and differential redundancy. In detail:

i) Equations of the input list are ordered by increasing length relative to the operator `Plus`. Monomials are collected in the list `Result1` and polynomials in `Result2`.

ii) Equations of `Result1` will be factorized in a product form $A{\cdot}B$ where B is a differential equation in the infinitesimals and A is a multiplicative factor free of ξ_i and ϕ_α. The program assumes that all parameters or functions appearing in A are nonzero; hence division is made and actually, A will be one after the treatment. In a second step, all differential equations of monomial type are used to clear the redundancies in the polynomial equations.

`SubstituteInfinitesimals[]` This function is used to check wether the infinitesimals satisfy the determining equations or not. If the flag `Info` is set to `True` the information on the infinitesimals contained in the lists `ListXi` and

`ListPhi` is used to check the equations obtained by the function `Simplify-Equations[]`.

`Lie[ ]` This function stands for the main program. A call to this function initiates computation. One or more arguments enable the user to control the execution. The first argument `Param1` essentially controls which equation is treated and from where the information on that equation can be read. If `Param1` is a string denoting an existing file name, the prgoram will read data from this file. All further arguments are used as options. The following options for `Lie[]` exist: `ProlongationOrder, Warning, Info, EquationS, Trace, CommutatorTable, StructureConstants, Ansatz`, and `LaTeX`. A description of the options is given below.

- `Ansatz` is an option which can be used to find the infinitesimals by using a polynomial ansatz. The list given in the call contains the maximal order of the polynomial for the infinitesimals. `Ansatz` $\rightarrow \{2\}$ creates infinitesimals quadratic in the dependent and independent variables. The default value is `Ansatz` $\rightarrow \{0\}$.
- `CommutatorTable` if set to `True` in connection with `Info` set also to `True` the basis of the vectorfield related to the Lie–algebra is generated and a commutator table is calculated.
- `EquationS` a list of equation labels are used in the calculation. All equations included in this list are dealt with. The default value of `EquationS` is {`All`}.
- `Info` if set to `True` the information in the lists `ListXi` and `ListPhi` given in the input file is used; default is `False`.
- The option `LaTeX` allows to controll the output of the program in LaTeX form. If set to `True` all information on the equations and the obtained results are stored in a separate file.
- `ProlongationOrder` determines the highest order of derivatives in the calculation of the prolongation. `ProlongationOrder` $\rightarrow \{order\}$ default value is {`All`}.
- `StructureConstants` if set to `True` the structure constants of the Lie–algebra are calculated. This option is only useful in connection with the option `Info`. If `Info` is also set to `True` the structure constants are calculated.
- `Trace` if set to `True` information about the present step in the calculation is given to the terminal.
- `Warning` if set to `True` warnings are given if nonnumeric terms are eliminated in the determining equations; default value is `True`.

All these options are arbitrarily arrangable in the call of `Lie[]`.

`CommutatorTable[]` if the flags `Info` and `CommutatorTable` is set to `True` the information on the infinitesimals contained in the lists `ListXi` and `ListPhi` are used to create the basis of the vector field $\vec{v}$ and the commutator table is calculated. The result of the calculation is stored in the global variable `LieTable`.

`StructureConstants[]` if the flags `Info` and `StructureConstants` are `True` the information on the infinitesimals are used to construct the commutator table from which the structure constants are derivable. The structure constants are available in the global variable `LieStructure`.

`AnsatzInfinitesimals[]` creates a polynomial of a certain order in the dependent and independent variables. The order of the polynomial is transfered to the function by the option `Ansatz` $\to \{order\}$. The created polynomial is used by the function `SubstituteInfinitesimals[]`.

We remark that only a few of the discussed functions are globaly available. If you read in the `Mathematica` program `Lie[]` the global functions are denoted.

3.3 Input Data

The information needed by the program to perform the computation is contained in an input file. The contence of this file is written in conventional *Mathematica* notation. This file is read by `Lie[]`; provided, it exists in the current directory. If the file does not exist `Lie[]` is terminated with an error message stating that the filename given is not available in the current directory. The input file contains:

1. A list of independent variables with the name: `IndepVar`. The sequential order of the independent variables in this list should represent the sequential order of the variables used as arguments of the dependent variables.

2. A list of dependent variables with the name: `DependVar`

3. The left hand side of the equations of motion (1). The equations $\Delta^i, i = 1, ..., m$ must be given in real form in the list `EqList`. All dependencies and differentiations must be explicitly written out.

4. The variables w^β for the side conditions $|_{\Delta=0}$ in (10) are collected in the list `SubsList`. This list should contain the terms for which `EqList` can be solved by the *Mathematica* function `Solve`.

5. The list `ParameterS` contains parameters of the model which are treated as constants. This list is only used in the calculation of the Lie–algebra.

6. Lists `ListXi` and `ListPhi` contain information about the infinitesimals. The information is used by the program if the option `Info` is set `True` otherwise

no use of the contence in the lists is made by `Lie[]`. The order of infinitesimals within each list correspond to the order of independent and dependent variables in the lists `IndepVar` and `DependVar`.

A sample set of such input data is included in the online description of `Lie[]`. Calling for help with `?Lie` at the *Mathematica* prompt gives you a short descrition of the options and the input file. Contrary to other programs in REDUCE, MACSYMA, etc., the equations Δ^i are not converted to an artificial representation, denoting differentiation and dependent variables by a vector of cumbersome numbers. The input data of `Lie[]` are in a form actually used in pencil calculations. The representation of equations can be used in the form used in handwritten notation, i.e. $\frac{\partial u(x,t)}{\partial,t} - \frac{\partial^2 u(x,t)}{\partial x^2} = 0$ corresponds in *Mathematica* to `D[u[x,t]t]-D[u[x,t],{x,2}]`. The expressions used in the substitution list `SubsList` and the infinitesimal lists `ListXi` and `ListPhi` must be of the same explicite form. The input data is put into a file with the use of *Mathematica* assignment statements. This file is read by the program if `Lie[]` is called with the file name as first argument (`Param1 = filename`).

If `Lie[]` is called at the first time the program checks the existence of a file containing the determining equations. If this file exists `Lie[]` uses the information on the determining equations stored in this file, otherwise `Lie[]` will create the file after the calculation of the prolongation and simplification. It is important to notice that a change in the equations of motion will also change the determining equations and thus may result into errors in solving the determining equations. If one needs to chnage the equations of motion one shoul delete the existing file with extension `.deq` in the current directory.

3.4 Output Data

If `Lie[]` finished the computation in a regular way some information are printed on the screen. During the calculation results and messages concerning the state of the calculation are displayed. Some samples are given in the section Exaples. The results produced by `Lie[]` are also available for further manipulations in the list `FinalResult`. This list contains as elements several data of different structure. A collection of the listelements is given below.

`FinalResult[[1]]` A list containing the left hand side of the equations of motion.

`FinalResult[[2]]` A list of side conditions under which the invariance is calculated.

`FinalResult[[3]]` A list of infinitesimals of the independent and dependent variables. The order is ξ_i first and ϕ_α last.

`FinalResult[[4]]` A list of the determining equations for the infinitesimals.

`FinalResult[[5]]` A list of Casimier operators of the algebra.

`FinalResult[[6]]` A list of the basis and ideals of the Lie–algebra.

`FinalResult[[7]]` A list which contains the basis of the vector field $\vec{v}$ given in equation (9).

`FinalResult[[8]]` An array which contains the comutator table of the basic vector fiels.

`FinalResult[[9]]` An array with the structure constants of the Lie–algebra.

`FinalResult[[10]]` An array with the metric g_{ij} of the Lie–algebra.

The determining equations are printed on the screen and are available for further manipulation in the list `Result2`. While the program is running, the total output is simultaneously logged in a file whous first part of the name is derived from the name used in calling `Lie[]` but its extension is changed to `.lie`. If the options `CommutatorTable` and/or `StructureConstants` in connection with the option `Info` are set `True`, the basic vector fields of the vectorfield $\vec{v}$, the commutator table and/or the structure constants are printed. With these options the metric of the algebra is also calculated and displaied.

To speed up the recursive and automatic calculation of the infinitesimals, the determining equations are saved in a file with extension `.deq`. This file is only written when `Lie[]` does not find this file in the current directory. Should you change any constants or the equation itself, be sure that this file is created by `Lie[]` at the same time. Otherwise, your results may be wrong!

A prity print output in LaTeX form is available if the option `LaTeX` in the calling sequence of `Lie[]` is set to `True`. The LaTeX output is written into a file with extension `.tex`. It contains the same information as saved in the list `FinalResult`. This file can be used as an input file to the text system LaTeX.

4 Examples

4.1 The one dimensional diffusion equation

As a first example, let us consider the widely studied symmetries of the diffusion equations [1, 2]. In order to obtain the determining system, we need to apply the second prolongation $pr^{(2)}\vec{v}$ according to (10) to the diffusion equation

$$\frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} = 0. \tag{13}$$

Equation (13) and some information on the dependencies and substitutions are given in the listing of the input file named `diffu.dgl`. To read the data stored in this file, the program is called with `Lie["diffu.dgl"]`. This command initiates the calculation of the determining equations and writes these equations to a file called `diffu.deq`.

```
(* ---    Diffusion Equation      --- *)
(*----------------------------------*)
(* --- independent variables --- *)
IndepVar={x,t};
(* --- dependent variables --- *)
DependVar={u};
(* --- equations  --- *)
EqList = {D[u[x,t],t] - D[u[x,t],{x,2}]}
(* --- substitutions --- *)
SubsList = {D[u[x,t],t]};
(* --- parameters --- *)
ParameterS = {};
(* --- lists of the infinitesimals only treated if Info -> True --- *)
ListXi = {c1 + c4*x + 2*c5*t + 4*c6*x*t,
          c2 + 2*c4*t + 4*c6*t^2};
ListPhi = {(c3-c5*x-2*c6*t-c6*x^2)*u[x,t]+f[x,t]};
```

Functions ξ_i and ϕ_α are obtained by the following system of partial differential equations. The infinitesimals $\xi_i(x,u)$ and $\phi_\alpha(x,u)$ are denoted in the *Mathematica* output by `Xi[i][x,t,u[x,t]]` and `Phi[alpha][x,t,u[x,t]]` where $i = 1, 2, \ldots, q$ and $\alpha = 1, 2, \ldots, p$, respectively.

```
+----------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the         |
|            calculation of the symmetry group              |
|                          by                               |
|                      G. Baumann                           |
|                   University of Ulm                       |
+----------------------------------------------------------+


### Information: File LieSymmetry`Private`nmaeSystem not found.
                 Calculate the prolongation.
Loading data from  diffu.dgl


 -------------------------------------------------------------------
 Equation No. 1

  (0,1)          (2,0)
 u     [x, t] - u     [x, t] = 0

 -------------------------------------------------------------------
 Substitution No. 1

  (0,1)            (2,0)
 u     [x, t] -> u     [x, t]

 -------------------------------------------------------------------
```

```
You are using the 2-th order of derivatives
in the calculation of prolongations.

Working on equation 1
  Enter ProlongationF
  Applying constraints
  Enter expand
  Enter search of coefficients

  Enter simplifying equations

You are using only the equation: {All}
of the given system consisting of 1 equations in total.

**** Number of determining equations before simplification
**** N_before = 9

List of factors cancelled in the calculation:       {-1, 2}

**** Number of determining equations after simplification
**** N_after = 7.
**** These determining equations are stored in Result2.   ****

 Equation No. : 1
 --------------------------------------------------------------------
        (0,0,1)
 (xi[1])       [x, t, u[x, t]] = 0

 --------------------------------------------------------------------
 Equation No. : 2
 --------------------------------------------------------------------
        (0,0,1)
 (xi[2])       [x, t, u[x, t]] = 0

 --------------------------------------------------------------------
 Equation No. : 3
 --------------------------------------------------------------------
         (0,0,2)
 (phi[1])       [x, t, u[x, t]] = 0

 --------------------------------------------------------------------
 Equation No. : 4
 --------------------------------------------------------------------
        (1,0,0)
 (xi[2])       [x, t, u[x, t]] = 0

 --------------------------------------------------------------------
 Equation No. : 5
 --------------------------------------------------------------------
         (0,1,0)                            (1,0,1)
 -(xi[1])       [x, t, u[x, t]] - 2 (phi[1])       [x, t, u[x, t]] +

           (2,0,0)
>    (xi[1])       [x, t, u[x, t]] = 0
```

```
-------------------------------------------------------------------
Equation No. : 6
-------------------------------------------------------------------
        (0,1,0)                         (2,0,0)
(phi[1])       [x, t, u[x, t]] - (phi[1])       [x, t, u[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 7
-------------------------------------------------------------------
        (0,1,0)                          (1,0,0)
-(xi[2])       [x, t, u[x, t]] + 2 (xi[1])       [x, t, u[x, t]] = 0

-------------------------------------------------------------------

**** Number of determining equations after simplification
**** N_after = 7.
**** The determining equations are stored in Result2.   ****

 Results are collected in the list FinalResult.
```

Differentiation with respect to some of the independent variables is denoted in *Mathematica* by an upper index $u^{(2,0)}[x,t]$ which is identical with the mathematical notation $\frac{\partial^2 u(x,t)}{\partial x^2}$. First we discuss the iterative and manual procedure to solve the determining equations. Starting with a total of 9 equations, a reduced system of 7 equations follows if the redundancy is canceled. The time used to derive these equations is 21 sec on a 386Pc. The general solution of these equations determines the symmetry group of the diffusion equation. The next step in the calculation is to solve the single–term equations. From the first four equations, we obtain

$$\xi_1 = \xi_1(x,t) \tag{14}$$

$$\xi_2 = \xi_2(t) \tag{15}$$

$$\phi_1 = f_1(x,t)u(x,t) + f_2(x,t). \tag{16}$$

Feeding this information into the `Lie[]` program in a second run by changing the lists `ListXi` and `ListPhi` in the file `diffu.dgl` to `ListXi = { Xi1[x,t],Xi2[t]}` and `ListPhi = {f1[x,t]*u[x,t] +f2[x,t]}`, we will get a simplified system of equations. Calling `Lie[]` with the option `Info -> True` by

```
Lie["diffu.dgl",Info->True,Warning->False]
```

a new system of determining equations is obtained. The given command reads first the previously written file `diffu.deq` and then the altered file `diffu.dgl`. The information contained in both files are used to find the probably simpler determining system for the infinitesimals. The input file `diffu.dgl` and the results are given below.

```
(* ---    Diffusion Equation      --- *)
(*----------------------------------*)
(* --- independent variables --- *)
IndepVar={x,t};
(* --- dependent variables --- *)
DependVar={u};
(* --- equations  --- *)
EqList = {D[u[x,t],t] - D[u[x,t],{x,2}]}
(* --- substitutions --- *)
SubsList = {D[u[x,t],t]};
(* --- parameters --- *)
ParameterS = {};
(* --- lists of the infinitesimals only treated if Info -> True --- *)
ListXi = {Xi1[x,t],Xi2[t]};
ListPhi = {f1[x,t]*u[x,t] + f2[x,t]};
```

The results of the second run where partial information of the infinitesimals are feed in looks like the following output. In this listing the infinitesimals are printed in the form given in the input file `diffu.dgl`.

```
+-------------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the           |
|            calculation of the symmetry group                |
|                           by                                |
|                       G. Baumann                            |
|                     University of Ulm                       |
+-------------------------------------------------------------+


Loading previous calculations from  diffu.deq

Loading data from  diffu.dgl


 --------------------------------------------------------------------
 Equation No. 1

  (0,1)          (2,0)
 u     [x, t] - u     [x, t] = 0

 --------------------------------------------------------------------
 Substitution No. 1

  (0,1)           (2,0)
 u     [x, t] -> u     [x, t]

 --------------------------------------------------------------------
 --------------------------------------------------------------------
 Infinitesimals :
```

```
 Xi1 = Xi1[x, t]

 Xi2 = Xi2[t]

 Phi1 = f2[x, t] + f1[x, t] u[x, t]
 ---------------------------------------------------------------------

You are using the 2-th order of derivatives
in the calculation of prolongations.

You are using only the equation: {All}
of the given system consisting of 1 equations in total.


**** Number of determining equations after simplification
**** N_after = 3.
**** These determining equations are stored in Result2.   ****

 Equation No. : 1
 ---------------------------------------------------------------------
     (0,1)              (1,0)            (2,0)
 -Xi1     [x, t] - 2 f1     [x, t] + Xi1     [x, t] = 0

 ---------------------------------------------------------------------
 Equation No. : 2
 ---------------------------------------------------------------------
           (0,1)            (0,1)                   (2,0)
 u[x, t] f1     [x, t] + f2     [x, t] - u[x, t] f1     [x, t] -

      (2,0)
>   f2     [x, t] = 0

 ---------------------------------------------------------------------
 Equation No. : 3
 ---------------------------------------------------------------------
                 (1,0)
 -Xi2'[t] + 2 Xi1     [x, t] = 0

 ---------------------------------------------------------------------

**** Number of determining equations after simplification
**** N_after = 3.
**** The determining equations are stored in Result2.   ****

 Results are collected in the list FinalResult.
```

We additionally set the flag `Warning -> False` to suppress output from `Simplify-Equations[]`. A straightforward integration of these three equations delivers the given solutions in the form

$$\xi_1 = c_1 + c_4 x + 2tc_5 + 4xtc_6 \tag{17}$$

$$\xi_2 = c_2 + 2tc_4 + 4t^2 c_6 \tag{18}$$

$$\phi_1 = (c_3 - xc_5 - (2t + x^2)c_6)u(x,t) + f(x,t) \tag{19}$$

where $c_1, ..., c_6$ are arbitrary integration constants and f is a free function. This form of infinitesimals is given in the listing of the input file `diffu.dgl` used in the first run. The information on the infinitesimals did not affect the calculation at the first run since the option `Info` had its default value `False`. A third run now with `Info -> True` and with the known solutions provides the expected result that only one determining equation remains for the arbitrary function $f(x,t)$. This was to be expected as the symmetry group of a linear equation always contains a function which satisfies the original equation. The last run to verify the infinitesimals takes about 6 sec on a 386Pc.

```
+------------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the          |
|            calculation of the symmetry group               |
|                          by                                |
|                      G. Baumann                            |
|                  University of Ulm                         |
+------------------------------------------------------------+


Loading previous calculations from  diffu.deq

Loading data from  diffu.dgl


 --------------------------------------------------------------------
 Equation No. 1

  (0,1)           (2,0)
 u     [x, t] - u     [x, t] = 0

 --------------------------------------------------------------------
 Substitution No. 1

  (0,1)            (2,0)
 u     [x, t] -> u     [x, t]

 --------------------------------------------------------------------
 --------------------------------------------------------------------
 Infinitesimals :

 Xi1 = c1 + 2 c5 t + c4 x + 4 c6 t x

                           2
 Xi2 = c2 + 2 c4 t + 4 c6 t

                                               2
 Phi1 = f[x, t] + (c3 - 2 c6 t - c5 x - c6 x ) u[x, t]
 --------------------------------------------------------------------
```

```
You are using the 2-th order of derivatives
in the calculation of prolongations.

You are using only the equation: {All}
of the given system consisting of 1 equations in total.


**** Number of determining equations after simplification
**** N_after = 1.
**** These determining equations are stored in Result2.   ****

 Equation No. : 1
 ---------------------------------------------------------------------
  (0,1)          (2,0)
 f      [x, t] - f      [x, t] = 0

 ---------------------------------------------------------------------

**** Number of determining equations after simplification
**** N_after = 1.
**** The determining equations are stored in Result2.   ****

 Results are collected in the list FinalResult.
```

Knowing the infinitesimals ξ_1, ξ_2, and ϕ_1, we are able to calculate the corresponding discrete Lie–algebra. Running the program `Lie[]` again now with the options `Info`, `CommutatorTable`, and `StructureConstants` set to `True`, we get the commutator table and the structure constants of the Lie–algebra. The complete commandline for obtaining these results reads

```
Lie["diffu.dgl",Info->True,StructureConstants->True,-
                   CommutatorTable->True].
```

The output produced contains as already known the equations, substitution rules, infinitesimals, and determining equations. We create also the basis of the vector field by using the "algebraic" options `StructureConstants` and `CommutatorTable`. The derived algebras, the adjoint algebra, the commutator table, the structure constants, and the metric tensor of the algebra are also available.

```
+--------------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the            |
|            calculation of the symmetry group                 |
|                           by                                 |
|                       G. Baumann                             |
|                    University of Ulm                         |
+--------------------------------------------------------------+


Loading previous calculations from  diffu.deq
```

```
Loading data from  diffu.dgl


 -------------------------------------------------------------------
 Equation No. 1

  (0,1)          (2,0)
 u     [x, t] - u     [x, t] = 0

 -------------------------------------------------------------------
 Substitution No. 1

  (0,1)           (2,0)
 u     [x, t] -> u     [x, t]

 -------------------------------------------------------------------
 -------------------------------------------------------------------
 Infinitesimals :

 Xi1 = c1 + 2 c5 t + c4 x + 4 c6 t x

                             2
 Xi2 = c2 + 2 c4 t + 4 c6 t

                                             2
 Phi1 = f[x, t] + (c3 - 2 c6 t - c5 x - c6 x ) u[x, t]
 -------------------------------------------------------------------

You are using the 2-th order of derivatives
in the calculation of prolongations.

You are using only the equation: {All}
of the given system consisting of 1 equations in total.


**** Number of determining equations after simplification
**** N_after = 1.
**** These determining equations are stored in Result2.   ****

 Equation No. : 1
 -------------------------------------------------------------------
  (0,1)          (2,0)
 f     [x, t] - f     [x, t] = 0

 -------------------------------------------------------------------

**** Number of determining equations after simplification
**** N_after = 1.
**** The determining equations are stored in Result2.   ****

Calculation of the commutator table.

 Basis of the Lie algebra.
 -------------------------------------------------------------------
```

```
             (0,0,1)                   2       (0,0,1)
-2 t u Funcf        [x, t, u] - u x  Funcf        [x, t, u] +

       2       (0,1,0)                       (1,0,0)
>   4 t  Funcf        [x, t, u] + 4 t x Funcf        [x, t, u] -> V[6]

           (0,0,1)                       (1,0,0)
-(u x Funcf        [x, t, u]) + 2 t Funcf        [x, t, u] -> V[5]

         (0,1,0)                    (1,0,0)
2 t Funcf        [x, t, u] + x Funcf        [x, t, u] -> V[4]

       (0,0,1)
u Funcf        [x, t, u] -> V[3]

     (0,1,0)
Funcf        [x, t, u] -> V[2]

     (1,0,0)
Funcf        [x, t, u] -> V[1]

----------------------------------------------------------------------

Structure of the Lie algebra.
----------------------------------------------------------------------

Ideal 0 :  {V[1], V[2], V[3], V[4], V[5], V[6]}

Ideal 1 :  {nonsolvable algebra}

----------------------------------------------------------------------

Casimier(s) of the Lie algebra.

{V[3]}

----------------------------------------------------------------------

Commutator table of the Lie algebra.

----------------------------------------------------------------------
0                   0                   0                   -V[1]

>       V[3]                -2 V[5]

0                   0                   0                   -2 V[2]

>       -2 V[1]             2 V[3] - 4 V[4]

0                   0                   0                   0

>       0                   0

V[1]                2 V[2]              0                   0
```

```
>       -V[5]              -2 V[6]

-V[3]              2 V[1]             0                  V[5]

>       0                  0

2 V[5]             -2 V[3] + 4 V[4]   0                  2 V[6]

>       0                  0
--------------------------------------------------------------------

Commutator table is contained in the list LieTable.

Nonzero structure Constants.
--------------------------------------------------------------------

 C[1,4,1] = -1
 C[1,5,3] = 1
 C[1,6,5] = -2
 C[2,4,2] = -2
 C[2,5,1] = -2
 C[2,6,3] = 2
 C[2,6,4] = -4
 C[4,1,1] = 1
 C[4,2,2] = 2
 C[4,5,5] = -1
 C[4,6,6] = -2
 C[5,1,3] = -1
 C[5,2,1] = 2
 C[5,4,5] = 1
 C[6,1,5] = 2
 C[6,2,3] = -2
 C[6,2,4] = 4
 C[6,4,6] = 2

--------------------------------------------------------------------

Metric of the structure constants

            0     0     0     0     0     0

            0     0     0     0     0     -20

            0     0     0     0     0     0

            0     0     0     10    0     0

            0     0     0     0     0     0

            0     -20   0     0     0     0

Determinant of the metric

det = 0
```

```
Structure constants are contained in the list LieStructure.

Results are collected in the list FinalResult.
```

Setting additionally to the previous optins the option `LaTeX` to `True`, we can create a LaTeX output from `Lie[]`. The output obtained is separated by two bars from the running text

Results created by `Lie[]` © [1]

Equations of motion : **diffu.dgl**

$$0 = u^{(0,1)}(x,t) - u^{(2,0)}(x,t) \tag{20}$$

Substitutions in the equations of motion :

$$1 : u^{(0,1)}(x,t) \to u^{(2,0)}(x,t) \tag{21}$$

Infinitesimals for independent and dependent variables

$$\xi_1 = c1 + 2\,c5\,t + c4\,x + 4\,c6\,t\,x \tag{22}$$

$$\xi_2 = c2 + 2\,c4\,t + 4\,c6\,t^2 \tag{23}$$

$$\phi_1 = f(x,t) + \left(c3 - 2\,c6\,t - c5\,x - c6\,x^2\right)\,u(x,t) \tag{24}$$

Determining equations

$$1 : 0 = f^{(0,1)}(x,t) - f^{(2,0)}(x,t) \tag{25}$$

Casimier(s) of the vector fields

$$\mathcal{C}_1 = V(3) \tag{26}$$

Basis and Ideals of the Lie–algebra

$$I_0 = \{V(1), V(2), V(3), V(4), V(5), V(6)\} \tag{27}$$

$$I_1 = \{nonsolvablealgebra\} \tag{28}$$

[1] G. Baumann, University of Ulm , Albert-Einstein-Allee 11, D-7900 Ulm, Germany

Basis of the Lie–algebra

$$1 \; : \; -2\,t\,u\,Funcf^{(0,0,1)}(x,t,u) - u\,x^2\,Funcf^{(0,0,1)}(x,t,u) + 4\,t^2\,Funcf^{(0,1,0)}(x,t,u) + 4\,t\,x\,Funcf^{(1,0,0)}(x,t,u) \to V(6) \tag{29}$$

$$2 \; : \; -\Big(u\,x\,Funcf^{(0,0,1)}(x,t,u)\Big) + 2\,t\,Funcf^{(1,0,0)}(x,t,u) \to V(5) \tag{30}$$

$$3 \; : \; 2\,t\,Funcf^{(0,1,0)}(x,t,u) + x\,Funcf^{(1,0,0)}(x,t,u) \to V(4) \tag{31}$$

$$4 \; : \; u\,Funcf^{(0,0,1)}(x,t,u) \to V(3) \tag{32}$$

$$5 \; : \; Funcf^{(0,1,0)}(x,t,u) \to V(2) \tag{33}$$

$$6 \; : \; Funcf^{(1,0,0)}(x,t,u) \to V(1) \tag{34}$$

Commutator table

[,]	$V(1)$	$V(2)$	$V(3)$	$V(4)$	$V(5)$	$V(6)$
V(1)	0	0	0	$-V(1)$	$V(3)$	$-2\,V(5)$
V(2)	0	0	0	$-2\,V(2)$	$-2\,V(1)$	$2\,V(3) - 4\,V(4)$
V(3)	0	0	0	0	0	0
V(4)	$V(1)$	$2\,V(2)$	0	0	$-V(5)$	$-2\,V(6)$
V(5)	$-V(3)$	$2\,V(1)$	0	$V(5)$	0	0
V(6)	$2\,V(5)$	$-2\,V(3) + 4\,V(4)$	0	$2\,V(6)$	0	0

Structure constants

$$C_{1,4,1} \;=\; -1 \tag{35}$$

$$C_{1,5,3} = 1 \tag{36}$$

$$C_{1,6,5} = -2 \tag{37}$$

$$C_{2,4,2} = -2 \tag{38}$$

$$C_{2,5,1} = -2 \tag{39}$$

$$C_{2,6,3} = 2 \tag{40}$$

$$C_{2,6,4} = -4 \tag{41}$$

$$C_{4,1,1} = 1 \tag{42}$$

$$C_{4,2,2} = 2 \tag{43}$$

$$C_{4,5,5} = -1 \tag{44}$$

$$C_{4,6,6} = -2 \tag{45}$$

$$C_{5,1,3} = -1 \tag{46}$$

$$C_{5,2,1} = 2 \tag{47}$$

$$C_{5,4,5} = 1 \tag{48}$$

$$C_{6,1,5} = 2 \tag{49}$$

$$C_{6,2,3} = -2 \tag{50}$$

$$C_{6,2,4} = 4 \tag{51}$$

$$C_{6,4,6} = 2 \tag{52}$$

$$\tag{53}$$

Metric tensor g_{ij}

$$g_{ij} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -20 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -20 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{54}$$

$$\det(g_{ij}) = 0 \tag{55}$$

4.2 Fisher's equation

The equation which describes the combination of diffusion and population growth is

$$u_t - u_{xx} - a_1(1-u)u(x,t) = 0 \tag{56}$$

where a_1 is a constant. The equation (56) describes, for example, the diffusive spread of a population along the x–axis. The diffusion is altered by a linear creation term and a quadratic annihiliation component. Thus, the population cannot grow to infinity. The quadratic term in (56) will counteract the linear growth if a certain threshold in the concentration u is reached. By adding a linear and a quadratic term to the original diffusion equation, the symmetries of the diffusion equation are drastically changed. First, we find a reduction of the symmetry group from an infinite dimension to a finite one. Second, all the scaling properties of the six–dimensional discrete subgroup of the diffusion equation disappear. The remaining symmetries are just translational symmetries of the independent variables. The infinitesimals for x, t and u read

$$\xi_1 = c_1 \tag{57}$$

$$\xi_2 = c_2 \tag{58}$$

$$\phi_1 = 0, \tag{59}$$

respectively. To find such symmetries automaticly, we called `Lie[]` by the command `Lie["fisher.dgl",Ansatz->{1}]`. The input file `fisher.dgl` given below contains in the parameter list `ParameterS` the constant $a1$ of the model. The inclusion of the parameter list is necessary to distinguish it from the parameters introduced in the ansatz. The calculation of the infinitesimals laseted 15 sec on 486Pc. The verification of this result took about 10 sec on a 486Pc. The input data and log files for the runs are listed below.

```
(* --- Fisher's equation    --- *)
(*----------------------------*)
(* --- independent variables ---*)
IndepVar = {x,t};
(* --- dependent variables --- *)
DependVar = {u};
(* --- equations --- *)
EqList = {D[u[x,t],t] - D[u[x,t],{x,2}] - a1*u[x,t]*(1-u[x,t])};
(* --- substitutions --- *)
SubsList = {D[u[x,t],t]};
(* --- parameters --- *)
ParameterS = {a1};
(* --- lists of the infinitesimals only treated if Info -> True --- *)
```

```
+----------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the         |
|            calculation of the symmetry group              |
|                          by                               |
|                      G. Baumann                           |
|                   University of Ulm                       |
+----------------------------------------------------------+


### Information: File LieSymmetry`Private`nmaeSystem not found.
                 Calculate the prolongation.
Loading data from  fisher.dgl


 --------------------------------------------------------------------
 Equation No. 1

                                 (0,1)          (2,0)
 -(a1 (1 - u[x, t]) u[x, t]) + u      [x, t] - u     [x, t] = 0

 --------------------------------------------------------------------
 Substitution No. 1

  (0,1)                                   2    (2,0)
 u     [x, t] -> a1 u[x, t] - a1 u[x, t]  + u     [x, t]

 --------------------------------------------------------------------

You are using the 2-th order of derivatives
in the calculation of prolongations.

Working on equation 1
  Enter ProlongationF
  Applying constraints
  Enter expand
  Enter search of coefficients

  Enter simplifying equations

You are using only the equation: {All}
of the given system consisting of 1 equations in total.

**** Number of determining equations before simplification
**** N_before = 9

List of factors cancelled in the calculation:      {-1, 2}

**** Number of determining equations after simplification
**** N_after = 7.
**** These determining equations are stored in Result2.  ****

 Equation No. : 1
 --------------------------------------------------------------------
        (0,0,1)
```

```
(xi[1])        [x, t, u[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 2
-------------------------------------------------------------------
       (0,0,1)
(xi[2])        [x, t, u[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 3
-------------------------------------------------------------------
        (0,0,2)
(phi[1])        [x, t, u[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 4
-------------------------------------------------------------------
       (1,0,0)
(xi[2])        [x, t, u[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 5
-------------------------------------------------------------------
        (0,1,0)                            (1,0,1)
-(xi[1])        [x, t, u[x, t]] - 2 (phi[1])        [x, t, u[x, t]] +

            (2,0,0)
>    (xi[1])        [x, t, u[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 6
-------------------------------------------------------------------
-(a1 phi[1][x, t, u[x, t]]) + 2 a1 u[x, t] phi[1][x, t, u[x, t]] +

                               (0,0,1)
>    a1 u[x, t] (phi[1])        [x, t, u[x, t]] -

                2          (0,0,1)
>    a1 u[x, t]  (phi[1])        [x, t, u[x, t]] +

             (0,1,0)
>    (phi[1])        [x, t, u[x, t]] -

                              (0,1,0)
>    a1 u[x, t] (xi[2])        [x, t, u[x, t]] +

                2         (0,1,0)
>    a1 u[x, t]  (xi[2])        [x, t, u[x, t]] -

             (2,0,0)
>    (phi[1])        [x, t, u[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 7
-------------------------------------------------------------------
```

```
        (0,1,0)                          (1,0,0)
-(xi[2])       [x, t, u[x, t]] + 2 (xi[1])       [x, t, u[x, t]] = 0

--------------------------------------------------------------------

**** Number of determining equations after simplification
**** N_after = 7.
**** The determining equations are stored in Result2.  ****

 Create an ansatz for the infinitesimals.
--------------------------------------------------------------------

 Calling Solve with 4 equations
 The number of unknowns is 6

 The calculated infinitesimals are:

--------------------------------------------------------------------

Xi[1] = k21

Xi[2] = k42

Phi[1] = 0


--------------------------------------------------------------------


 Results are collected in the list FinalResult.
```

The input file `fisher.dgl` was altered by `Lie[]` in the last two lines. It reads:

```
(* --- Fisher's equation    --- *)
(*----------------------------*)
(* --- independent variables ---*)
IndepVar = {x,t};
(* --- dependent variables --- *)
DependVar = {u};
(* --- equations --- *)
EqList = {D[u[x,t],t] - D[u[x,t],{x,2}] - a1*u[x,t]*(1-u[x,t])};
(* --- substitutions --- *)
SubsList = {D[u[x,t],t]};
(* --- parameters --- *)
ParameterS = {a1};
(* --- lists of the infinitesimals only treated if Info -> True --- *)
ListXi = {k21, k42}

ListPhi = {0}
```

Again, we can print out the results of the calculation with the option `LaTeX` stet to `True`. The listing is given below.

Results created by `Lie[]` © [2]

Equations of motion : **fisher.dgl**

$$0 \;=\; -(a1\;(1-u(x,t))\;u(x,t)) + u^{(0,1)}(x,t) - u^{(2,0)}(x,t) \qquad (60)$$

Substitutions in the equations of motion :

$$1\;:\;u^{(0,1)}(x,t) \rightarrow a1\;u(x,t) - a1\;u(x,t)^2 + u^{(2,0)}(x,t) \qquad (61)$$

Infinitesimals for independent and dependent variables

$$\xi_1 \;=\; k21 \qquad (62)$$

$$\xi_2 \;=\; k42 \qquad (63)$$

$$\phi_1 \;=\; 0 \qquad (64)$$

Casimier(s) of the vector fields

$$\mathcal{C}_1 \;=\; V(1) \qquad (65)$$

$$\mathcal{C}_2 \;=\; V(2) \qquad (66)$$

Basis and Ideals of the Lie–algebra

$$I_0 \;=\; \{V(1), V(2)\} \qquad (67)$$

$$I_1 \;=\; \{\} \qquad (68)$$

Basis of the Lie–algebra

$$1\;:\;Funcf^{(0,1,0)}(x,t,u) \rightarrow V(2) \qquad (69)$$

$$2\;:\;Funcf^{(1,0,0)}(x,t,u) \rightarrow V(1) \qquad (70)$$

Commutator table

[2] G. Baumann, University of Ulm , Albert-Einstein-Allee 11, D-7900 Ulm, Germany

[,]	$V(1)$	$V(2)$
V(1)	0	0
V(2)	0	0

Structure constants

$$ \tag{71}$$

Metric tensor g_{ij}

$$g_{ij} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{72}$$

$$\det(g_{ij}) = 0 \tag{73}$$

4.3 The nonlinear Schrödinger equation

The diffusion equation is a fairly simple example for determining the symmetries of a differential equation. The nonlinear Schrödinger equation is an equation for which the necessary calculations for obtaining the full symmetry group is more extensive. The equation plays an important role in several physical applications and has received considerable interest due to the existence of soliton solutions. For the complex function $\psi(x,t)$, it reads in 1 + 1 dimensions

$$i\psi_t(x,t) + \psi_{xx}(x,t) + 2|\psi(x,t)|^2\psi(x,t) = 0 \tag{74}$$

Separating the unknown function ψ in its real and imaginary parts by $\psi = u + iv$, we obtain the following system of partial differential equations

$$u_t(x,t) + v_{xx}(x,t) + 2(u(x,t)^2 + v(x,t)^2)v(x,t) = 0 \tag{75}$$

$$-v_t(x,t) + u_{xx}(x,t) + 2(u(x,t)^2 + v(x,t)^2)u(x,t) = 0 \tag{76}$$

Calling `Lie[]` by

```
Lie["nls.dgl"]
```

produces the following output:

```
+-------------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the           |
|            calculation of the symmetry group                |
|                            by                               |
|                        G. Baumann                           |
|                    University of Ulm                        |
+-------------------------------------------------------------+


### Information: File LieSymmetry`Private`nmaeSystem not found.
                Calculate the prolongation.
Loading data from  nls.dgl


 ---------------------------------------------------------------------
 Equation No. 1

                  2          2     (0,1)          (2,0)
 2 v[x, t] (u[x, t]  + v[x, t] ) + u     [x, t] + v     [x, t] = 0

 Equation No. 2

                  2          2     (0,1)          (2,0)
 2 u[x, t] (u[x, t]  + v[x, t] ) - v     [x, t] + u     [x, t] = 0

 ---------------------------------------------------------------------
 Substitution No. 1

  (0,1)                     2                    3    (2,0)
 u     [x, t] -> -2 u[x, t]  v[x, t] - 2 v[x, t]  - v     [x, t]

 Substitution No. 2

  (0,1)                    3                    2    (2,0)
 v     [x, t] -> 2 u[x, t]  + 2 u[x, t] v[x, t]  + u     [x, t]

 ---------------------------------------------------------------------

You are using the 2-th order of derivatives
in the calculation of prolongations.

Working on equation 1
  Enter ProlongationF
  Applying constraints
  Enter expand
  Enter search of coefficients
Working on equation 2
  Enter ProlongationF
  Applying constraints
  Enter expand
  Enter search of coefficients

  Enter simplifying equations
```

```
You are using the 2 equations of the system.

**** Number of determining equations before simplification
**** N_before = 46

List of factors cancelled in the calculation:       {-3, -2, -1, 2}

**** Number of determining equations after simplification
**** N_after = 20.
**** These determining equations are stored in Result2.   ****

 Equation No. : 1
 -------------------------------------------------------------------
        (0,0,0,1)
 (xi[1])         [x, t, u[x, t], v[x, t]] = 0

 -------------------------------------------------------------------
 Equation No. : 2
 -------------------------------------------------------------------
        (0,0,0,1)
 (xi[2])         [x, t, u[x, t], v[x, t]] = 0

 -------------------------------------------------------------------
 Equation No. : 3
 -------------------------------------------------------------------
         (0,0,0,2)
 (phi[1])         [x, t, u[x, t], v[x, t]] = 0

 -------------------------------------------------------------------
 Equation No. : 4
 -------------------------------------------------------------------
         (0,0,0,2)
 (phi[2])         [x, t, u[x, t], v[x, t]] = 0

 -------------------------------------------------------------------
 Equation No. : 5
 -------------------------------------------------------------------
        (0,0,1,0)
 (xi[1])         [x, t, u[x, t], v[x, t]] = 0

 -------------------------------------------------------------------
 Equation No. : 6
 -------------------------------------------------------------------
        (0,0,1,0)
 (xi[2])         [x, t, u[x, t], v[x, t]] = 0

 -------------------------------------------------------------------
 Equation No. : 7
 -------------------------------------------------------------------
         (0,0,1,1)
 (phi[1])         [x, t, u[x, t], v[x, t]] = 0

 -------------------------------------------------------------------
 Equation No. : 8
 -------------------------------------------------------------------
```

```
        (0,0,1,1)
(phi[2])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 9
-------------------------------------------------------------------
        (0,0,2,0)
(phi[1])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 10
-------------------------------------------------------------------
        (0,0,2,0)
(phi[2])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 11
-------------------------------------------------------------------
       (1,0,0,0)
(xi[2])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 12
-------------------------------------------------------------------
          (1,0,1,0)
2 (phi[1])         [x, t, u[x, t], v[x, t]] -

            (2,0,0,0)
>    (xi[1])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 13
-------------------------------------------------------------------
          (1,0,0,1)
2 (phi[2])         [x, t, u[x, t], v[x, t]] -

            (2,0,0,0)
>    (xi[1])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 14
-------------------------------------------------------------------
4 u[x, t] v[x, t] phi[1][x, t, u[x, t], v[x, t]] +

             2
>    2 u[x, t]  phi[2][x, t, u[x, t], v[x, t]] +

             2
>    6 v[x, t]  phi[2][x, t, u[x, t], v[x, t]] +

             3         (0,0,0,1)
>    2 u[x, t]  (phi[1])         [x, t, u[x, t], v[x, t]] +

                     2         (0,0,0,1)
>    2 u[x, t] v[x, t]  (phi[1])         [x, t, u[x, t], v[x, t]] -
```

```
                 2                  (0,0,1,0)
>    2 u[x, t]  v[x, t] (phi[1])             [x, t, u[x, t], v[x, t]] -

                 3          (0,0,1,0)
>    2 v[x, t]  (phi[1])             [x, t, u[x, t], v[x, t]] +

                (0,1,0,0)
>    (phi[1])             [x, t, u[x, t], v[x, t]] +

                 2                 (0,1,0,0)
>    2 u[x, t]  v[x, t] (xi[2])             [x, t, u[x, t], v[x, t]] +

                 3         (0,1,0,0)
>    2 v[x, t]  (xi[2])             [x, t, u[x, t], v[x, t]] +

                (2,0,0,0)
>    (phi[2])             [x, t, u[x, t], v[x, t]] = 0

  --------------------------------------------------------------------
  Equation No. : 15
  --------------------------------------------------------------------
              2
  6 u[x, t]  phi[1][x, t, u[x, t], v[x, t]] +

                 2
>    2 v[x, t]  phi[1][x, t, u[x, t], v[x, t]] +

>    4 u[x, t] v[x, t] phi[2][x, t, u[x, t], v[x, t]] -

                 3          (0,0,0,1)
>    2 u[x, t]  (phi[2])             [x, t, u[x, t], v[x, t]] -

                         2          (0,0,0,1)
>    2 u[x, t] v[x, t]  (phi[2])             [x, t, u[x, t], v[x, t]] +

                 2                  (0,0,1,0)
>    2 u[x, t]  v[x, t] (phi[2])             [x, t, u[x, t], v[x, t]] +

                 3          (0,0,1,0)
>    2 v[x, t]  (phi[2])             [x, t, u[x, t], v[x, t]] -

                (0,1,0,0)
>    (phi[2])             [x, t, u[x, t], v[x, t]] +

                 3         (0,1,0,0)
>    2 u[x, t]  (xi[2])             [x, t, u[x, t], v[x, t]] +

                         2         (0,1,0,0)
>    2 u[x, t] v[x, t]  (xi[2])             [x, t, u[x, t], v[x, t]] +

                (2,0,0,0)
>    (phi[1])             [x, t, u[x, t], v[x, t]] = 0

  --------------------------------------------------------------------
```

```
Equation No. : 16
-------------------------------------------------------------------
        (0,1,0,0)
-(xi[1])         [x, t, u[x, t], v[x, t]] +

             (1,0,1,0)
>   2 (phi[2])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 17
-------------------------------------------------------------------
       (0,1,0,0)
(xi[1])         [x, t, u[x, t], v[x, t]] +

             (1,0,0,1)
>   2 (phi[1])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 18
-------------------------------------------------------------------
         (0,0,0,1)
-(phi[2])         [x, t, u[x, t], v[x, t]] +

           (0,0,1,0)
>   (phi[1])         [x, t, u[x, t], v[x, t]] +

          (0,1,0,0)
>   (xi[2])         [x, t, u[x, t], v[x, t]] -

            (1,0,0,0)
>   2 (xi[1])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 19
-------------------------------------------------------------------
        (0,0,0,1)
(phi[2])         [x, t, u[x, t], v[x, t]] -

           (0,0,1,0)
>   (phi[1])         [x, t, u[x, t], v[x, t]] +

          (0,1,0,0)
>   (xi[2])         [x, t, u[x, t], v[x, t]] -

            (1,0,0,0)
>   2 (xi[1])         [x, t, u[x, t], v[x, t]] = 0

-------------------------------------------------------------------
Equation No. : 20
-------------------------------------------------------------------
        (0,0,0,1)
(phi[1])         [x, t, u[x, t], v[x, t]] +

           (0,0,1,0)
>   (phi[2])         [x, t, u[x, t], v[x, t]] = 0
```

```
 ---------------------------------------------------------------------

**** Number of determining equations after simplification
**** N_after = 20.
**** The determining equations are stored in Result2.   ****

 Create an ansatz for the infinitesimals.
 ---------------------------------------------------------------------

 Calling Solve with 25 equations
 The number of unknowns is 15

 The calculated infinitesimals are:

 ---------------------------------------------------------------------

Xi[1] = k55 + 2 k94 t - k89 x

Xi[2] = k70 - 2 k89 t

Phi[1] = k89 u[x, t] - k88 v[x, t] - k94 x v[x, t]

Phi[2] = k88 u[x, t] + k94 x u[x, t] + k89 v[x, t]


 ---------------------------------------------------------------------


 Results are collected in the list FinalResult.
```

The determining system contains 46 equations before simplification and 20 after the elimination of redundant terms. This is a factor two in the reduction of the determining equations. With the help of `Lie[]` the determining system is solved in an analogous way as demonstrated by the previous example using an ansatz. In handwritten calculations it may take a full day to obtain the infinitesimals without error which read for the real valued system (75,76)

$$\xi_1 = c_2 - 2c_4 t + c_5 x \tag{77}$$

$$\xi_2 = c_1 + 2c_5 t \tag{78}$$

$$\phi_1 = (c_3 + c_4 x)v(x,t) - c_5 u(x,t) \tag{79}$$

$$\phi_2 = -(c_3 + c_4 x)u(x,t) - c_5 v(x,t). \tag{80}$$

To derive the infinitesimals by calling `Lie["nls.dgl",Ansatz->{2}]`, we used about 1150 sec on a 386Pc. The input file `nls.dgl` reads after the run

```
(* --- NLS equation in real form --- *)
(*----------------------------*)
(* --- independent variables ---*)
IndepVar = {x,t};
(* --- dependent variables --- *)
DependVar = {u,v};
(* --- equations --- *)
EqList = {D[u[x,t],t] + D[v[x,t],{x,2}] + 2*(u[x,t]^2 + v[x,t]^2)*v[x,t],
          -D[v[x,t],t] + D[u[x,t],{x,2}] + 2*(u[x,t]^2 + v[x,t]^2)*u[x,t]};
(* --- substitutions --- *)
SubsList = {D[u[x,t],t],D[v[x,t],t]};
(* --- parameters --- *)
ParameterS ={};
(* --- lists of the infinitesimals only treated if Info -> True --- *)
ListXi = {k55 + 2*k94*t - k89*x, k70 - 2*k89*t}

ListPhi = {k89*u[x, t] - k88*v[x, t] - k94*x*v[x, t],
   k88*u[x, t] + k94*x*u[x, t] + k89*v[x, t]}
```

To verify the correctness of the infinitesimals (77-(80), and find the algebraic structure `Lie[]` takes about 90 sec on a 386Pc.

```
+------------------------------------------------------------+
|       Welcome to the MATHEMATICA program for the           |
|           calculation of the symmetry group                |
|                          by                                |
|                      G. Baumann                            |
|                   University of Ulm                        |
+------------------------------------------------------------+


Loading previous calculations from  nls.deq

Loading data from  nls.dgl


 --------------------------------------------------------------------
 Equation No. 1

                  2          2     (0,1)           (2,0)
 2 v[x, t] (u[x, t]  + v[x, t] ) + u     [x, t] + v     [x, t] = 0

 Equation No. 2

                  2          2     (0,1)           (2,0)
 2 u[x, t] (u[x, t]  + v[x, t] ) - v     [x, t] + u     [x, t] = 0

 --------------------------------------------------------------------
 Substitution No. 1

  (0,1)                   2                    3    (2,0)
 u     [x, t] -> -2 u[x, t]  v[x, t] - 2 v[x, t]  - v     [x, t]
```

```
 Substitution No. 2

  (0,1)                   3                      2    (2,0)
 v     [x, t] -> 2 u[x, t]  + 2 u[x, t] v[x, t]  + u     [x, t]

 -------------------------------------------------------------------
 -------------------------------------------------------------------
 Infinitesimals :

 Xi1 = k55 + 2 k94 t - k89 x

 Xi2 = k70 - 2 k89 t

 Phi1 = k89 u[x, t] - k88 v[x, t] - k94 x v[x, t]

 Phi2 = k88 u[x, t] + k94 x u[x, t] + k89 v[x, t]
 -------------------------------------------------------------------

You are using the 2-th order of derivatives
in the calculation of prolongations.

You are using the 2 equations of the system.


**** Number of determining equations after simplification
**** N_after = 0.

**** Number of determining equations after simplification
**** N_after = 0.

Calculation of the commutator table.

 Basis of the Lie algebra.
 -------------------------------------------------------------------

          (0,0,0,1)                        (0,0,1,0)
 u x Funcf         [x, t, u, v] - v x Funcf         [x, t, u, v] +

              (1,0,0,0)
>    2 t Funcf         [x, t, u, v] -> V[5]

        (0,0,0,1)                      (0,0,1,0)
 v Funcf         [x, t, u, v] + u Funcf         [x, t, u, v] -

              (0,1,0,0)                     (1,0,0,0)
>    2 t Funcf         [x, t, u, v] - x Funcf         [x, t, u, v] -> V[4]

        (0,0,0,1)                      (0,0,1,0)
 u Funcf         [x, t, u, v] - v Funcf         [x, t, u, v] -> V[3]

      (0,1,0,0)
 Funcf         [x, t, u, v] -> V[2]

      (1,0,0,0)
```

```
Funcf         [x, t, u, v] -> V[1]

--------------------------------------------------------------------

Structure of the Lie algebra.
--------------------------------------------------------------------

Ideal 0 :  {V[1], V[2], V[3], V[4], V[5]}

Ideal 1 :  {V[1], V[2], V[3], V[5]}

Ideal 2 :  {V[1], V[3]}

Ideal 3 :  {solvable algebra}

--------------------------------------------------------------------

Casimier(s) of the Lie algebra.

{V[3]}

--------------------------------------------------------------------

Commutator table of the Lie algebra.

--------------------------------------------------------------------
0         0         0         V[1]      -V[3]

0         0         0         2 V[2]    -2 V[1]

0         0         0         0         0

-V[1]     -2 V[2]   0         0         V[5]

V[3]      2 V[1]    0         -V[5]     0
--------------------------------------------------------------------

Commutator table is contained in the list LieTable.

Nonzero structure Constants.
--------------------------------------------------------------------

 C[1,4,1] = 1
 C[1,5,3] = -1
 C[2,4,2] = 2
 C[2,5,1] = -2
 C[4,1,1] = -1
 C[4,2,2] = -2
 C[4,5,5] = 1
 C[5,1,3] = 1
 C[5,2,1] = 2
 C[5,4,5] = -1

--------------------------------------------------------------------
```

```
Metric of the structure constants

          0   0   0   0   0

          0   0   0   0   0

          0   0   0   0   0

          0   0   0   6   0

          0   0   0   0   0

Determinant of the metric

det = 0

Structure constants are contained in the list LieStructure.

Results are collected in the list FinalResult.
```

4.4 Kadomtsev–Petviashvili equation (KP)

A further extension of the application of `Lie[]` is a 2 + 1 dimensional equation (two spatial and one time variable). A prominent example of this type is the KP equation [3]

$$\left(u_t + \frac{3}{2}uu_x + \frac{1}{4}u_{xx}\right)_x + \frac{3}{4}u_{yy} = 0\,. \tag{81}$$

The KP equation may be thought of as a two spatial, dimensional analog of the Korteweg de Vries equation. It is one of the classical prototype equations in the field of exactly solvable equations and arises in several physical contexts such as plasma physics and surface water waves.

The discrete symmetry group corresponding to (81) is a six parameter group. Symmetries of this equation correspond to translations with respect to all spatial coordinates and time. Additionally, there exists scaling in all varables. The infinitesimals follow from a set of 21 equations which are the result of the simplification in `SimplifyEquations[]` from 89 equations derivable from (10). The infinitesimals are given by

$$\xi_1 = c_2 + 2yc_4 + 3tc_5 + xc_6 \tag{82}$$

$$\xi_2 = c_3 - 3tc_4 + 2yc_6 \tag{83}$$

$$\xi_3 = c_1 + 3tc_6 \tag{84}$$

$$\phi_1 = 2c_5 - 2u(x,t)c_6\,. \tag{85}$$

The infinitesimals ξ_1, ξ_2 and ξ_3 are related to the independent variables x, y, t, respectively. With `Lie[]`, the whole manual calculation of the infinitesimals takes about five iterative steps. Real computer time of the automatic solution is about 131 sec on a 486Pc. The input and log files for the last calculation with `CommutatorTable` and `StructureConstants` set to `True` are listed below.

```
(*******************************************************************
*                                                                  *
*   Kadomtsev Petviashvili equation                                *
*   two-dimensional KdV equation                                   *
*                                                                  *
*   (U_xxx/4 - 3*U*U_x/2 + U_t)_x + 3 U_yy/4 = 0                   *
*                                                                  *
*******************************************************************)
(* --- independent variables --- *)
IndepVar={x,y,t};
(* --- dependent variables --- *)
DependVar={u};
(* --- equations  --- *)
EqList = {D[ D[u[x,y,t],{x,3}]/4 +
             3*u[x,y,t]*D[u[x,y,t],x]/2 + D[u[x,y,t],t],x] +
             3*D[u[x,y,t],{y,2}]/4}
(* --- substitutions --- *)
SubsList = {D[u[x,y,t],t,x]};
(* --- parameters --- *)
ParameterS ={};
(* --- lists of the infinitesimals only treated if Info -> True --- *)
ListXi = {c2 + 2y*c4 + 3*t*c5 + x*c6,
          c3 -3*t*c4+2*y*c6,
          c1 + 3*t*c6};
ListPhi = {2*c5 - 2*c6*u[x,y,t]};
```

```
+-----------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the          |
|            calculation of the symmetry group               |
|                          by                                |
|                      G. Baumann                            |
|                  University of Ulm                         |
+-----------------------------------------------------------+


Loading previous calculations from  kp.deq

Loading data from  kp.dgl


 -------------------------------------------------------------------
 Equation No. 1
```

```
    (0,2,0)              (1,0,0)         2
3 u       [x, y, t]   3 u       [x, y, t]      (1,0,1)
------------------- + -------------------- + u       [x, y, t] +
         4                     2

                 (2,0,0)            (4,0,0)
   3 u[x, y, t] u       [x, y, t]   u       [x, y, t]
>  ----------------------------- + ----------------- = 0
                 2                          4

--------------------------------------------------------------------
Substitution No. 1

 (1,0,1)
u       [x, y, t] ->

        (0,2,0)              (1,0,0)         2
>   -(3 u       [x, y, t] + 6 u       [x, y, t]  +

                    (2,0,0)            (4,0,0)
>       6 u[x, y, t] u       [x, y, t] + u       [x, y, t]) / 4

--------------------------------------------------------------------
--------------------------------------------------------------------
Infinitesimals :

Xi1 = c2 + 3 c5 t + c6 x + 2 c4 y

Xi2 = c3 - 3 c4 t + 2 c6 y

Xi3 = c1 + 3 c6 t

Phi1 = 2 c5 - 2 c6 u[x, y, t]
--------------------------------------------------------------------

You are using the 4-th order of derivatives
in the calculation of prolongations.

You are using only the equation: {All}
of the given system consisting of 1 equations in total.


**** Number of determining equations after simplification
**** N_after = 0.

**** Number of determining equations after simplification
**** N_after = 0.

Calculation of the commutator table.

 Basis of the Lie algebra.
--------------------------------------------------------------------

          (0,0,0,1)                        (0,0,1,0)
-2 u Funcf         [x, y, t, u] + 3 t Funcf         [x, y, t, u] +
```

```
>                 (0,1,0,0)                              (1,0,0,0)
      2 y Funcf          [x, y, t, u] + x Funcf          [x, y, t, u] -> V[6]

           (0,0,0,1)                              (1,0,0,0)
   2 Funcf          [x, y, t, u] + 3 t Funcf          [x, y, t, u] -> V[5]

              (0,1,0,0)                              (1,0,0,0)
   -3 t Funcf          [x, y, t, u] + 2 y Funcf          [x, y, t, u] -> V[4]

         (0,1,0,0)
   Funcf          [x, y, t, u] -> V[3]

         (1,0,0,0)
   Funcf          [x, y, t, u] -> V[2]

         (0,0,1,0)
   Funcf          [x, y, t, u] -> V[1]

   -------------------------------------------------------------------

   Structure of the Lie algebra.
   -------------------------------------------------------------------

   Ideal 0 :  {V[1], V[2], V[3], V[4], V[5], V[6]}

   Ideal 1 :  {V[1], V[2], V[3], V[4], V[5]}

   Ideal 2 :  {V[2], V[3]}

   Ideal 3 :  {solvable algebra}

   -------------------------------------------------------------------

   Casimier(s) of the Lie algebra.

   {}

   -------------------------------------------------------------------

   Commutator table of the Lie algebra.

   -------------------------------------------------------------------
   0         0         0         3 V[3]    -3 V[2]   -3 V[1]

   0         0         0         0         0         -V[2]

   0         0         0         -2 V[2]   0         -2 V[3]

   -3 V[3]   0         2 V[2]    0         0         V[4]

   3 V[2]    0         0         0         0         2 V[5]

   3 V[1]    V[2]      2 V[3]    -V[4]     -2 V[5]   0
   -------------------------------------------------------------------
```

```
 Commutator table is contained in the list LieTable.

Nonzero structure Constants.
 --------------------------------------------------------------------

  C[1,4,3] = 3
  C[1,5,2] = -3
  C[1,6,1] = -3
  C[2,6,2] = -1
  C[3,4,2] = -2
  C[3,6,3] = -2
  C[4,1,3] = -3
  C[4,3,2] = 2
  C[4,6,4] = 1
  C[5,1,2] = 3
  C[5,6,5] = 2
  C[6,1,1] = 3
  C[6,2,2] = 1
  C[6,3,3] = 2
  C[6,4,4] = -1
  C[6,5,5] = -2

 --------------------------------------------------------------------

 Metric of the structure constants

             0    0    0    0    0    0

             0    0    0    0    0    0

             0    0    0    0    0    0

             0    0    0    0    0    0

             0    0    0    0    0    0

             0    0    0    0    0    19

 Determinant of the metric

 det = 0

 Structure constants are contained in the list LieStructure.

 Results are collected in the list FinalResult.
```

4.5 System of coupled nonlinear Schrödinger equations

A rather extensive system of equations is given by the coupled nonlinear Schrödinger equations

$$i\psi_{1t}(x,t) = -\frac{1}{2}\psi_{1xx} + \kappa_1\psi_1^*\psi_1^2 + \lambda\psi_2^*\psi_2\psi_1 \tag{86}$$

$$i\psi_{2t}(x,t) = -\frac{1}{2}\psi_{2xx} + \kappa_2\psi_2^*\psi_2^2 + \lambda\psi_1^*\psi_1\psi_2\,. \tag{87}$$

These equations are used to describe the interaction of two copropagating light pulses in an optical fiber [8, 9]. Instead of investigating the complex system (86,87), we will study the corresponding system of Madelung equations [10, 11]. We first transform the complex valued fields $\psi_i(x,t), (i = 1,2)$ to real valued Madelung fields $r_i = r_i(x,t)$ and $s_i = s_i(x,t)$ by $\psi_i = \sqrt{r_i}\exp(is_i)$. This transformation gives $r_i = \psi_i^*\psi_i$ as density of the Schrödinger fields and s_i as the real phases of the wave function ψ_i. The transformation to densities and phases leads from (86,87) to a set of equations closely related to hydrodynamical equations

$$\partial_t r_i + \partial(r_i s_{ix}) = 0 \tag{88}$$

$$\partial_t s_i + \frac{1}{2}(s_{ix})^2 + \Gamma_i r_i + \frac{1}{8}\left(\frac{r_{ix}}{r_i}\right)^2 - \frac{1}{4}\left(\frac{r_{ixx}}{r_i}\right) + r_j = 0 \tag{89}$$

$(j, i = 1,2; j \neq i)$ where $\Gamma_i = \kappa_i/|\lambda|$ is the ratio of the coupling constants. We also introduced scaled variables $|\lambda|t \to t$ and $|\lambda|^{1/2}x \to x$ to reduce the number of free parameters in the model. The first two of these four equations are continuity equations for the densities r_i. The second two are generalizations of Euler equations describing the velocity fields of a two component fluid by potentials s_i.

The infinitesimal group transformations of equations (88,89) are derivable from a set of 408 determining equations before simplification. These equations collapse to 85 equations by cancelation of the single term equations. On a 486Pc we need about 750 sec to do the calculations. After several runs of `Lie[]` with different relations for the infinitesimals obtained from earlier runs, we derive a six parameter symmetry group for the equations (88,89) which reads

$$\xi_1 = c_4 + c_3 t + c_1 x \tag{90}$$

$$\xi_2 = c_2 + 2c_1 t \tag{91}$$

$$\phi_1 = -2c_1 r_1(x,t) \tag{92}$$

$$\phi_2 = c_5 + c_3 x \tag{93}$$

$$\phi_3 = -2c_1 r_2(x,t) \tag{94}$$

$$\phi_4 = c_6 + c_3 x. \tag{95}$$

The infinitesimals $\xi_1, \xi_2, \phi_1, \phi_2, \phi_3$ and ϕ_4 correspond to the independent and dependent variables x, t and r_1, s_1, r_2, s_2, respectively. The six arbitrary parameters c_n determine the transformation properties of (88,89). They reflect, for example, the invariance of translations with respect to t, x and s_i, the scaling of r_i, x, and t, and the Galilei invariance. The representation of the infinitesimals can be used to derive explicit solutions of (88) and (89) via a reduction to ordinary differential equations [12]. The derivation of the determining equations and the verification of (90-95) is listed below.

```
(**********************************************************************
*                                                                     *
*   System of nonlinear Schroedinger equations in fluid form          *
*                                                                     *
*   r1_t + r1_x*s1_x + r1*s1_xx = 0                                   *
*   r1^2*s1_t + (r1*s1_x)^2/2 + g1*r1^3 + r2*r1^2 +                   *
*                              r1_x^2/8 - r1*r1_xx/4 = 0              *
*   r2_t + r2_x*s2_x + r2*s2_xx = 0                                   *
*   r2^2*s2_t + (r2*s2_x)^2/2 + g2*r2^3 + r1*r2^2 +                   *
*                              r2_x^2/8 - r2*r2_xx/4 = 0              *
*                                                                     *
**********************************************************************)
(* --- independent variables --- *)
IndepVar={x,t};
(* --- dependent variables --- *)
DependVar={r1,s1,r2,s2};
(* --- equations  --- *)
EqList = {D[r1[x,t],t] + D[r1[x,t],x]*D[s1[x,t],x] + r1[x,t]*D[s1[x,t],{x,2}],
          D[s1[x,t],t] + (D[s1[x,t],x])^2/2 + g1*r1[x,t]+
          r2[x,t] + (D[r1[x,t],x]/r1[x,t])^2/8 - D[r1[x,t],{x,2}]/(4*r1[x,t]),
          D[r2[x,t],t] + D[r2[x,t],x]*D[s2[x,t],x] + r2[x,t]*D[s2[x,t],{x,2}],
          D[s2[x,t],t] + (D[s2[x,t],x])^2/2 + g2*r2[x,t] +
          r1[x,t] + (D[r2[x,t],x]/r2[x,t])^2/8 - D[r2[x,t],{x,2}]/(4*r2[x,t])
}
(* --- substitutions --- *)
SubsList = {D[r1[x,t],t],D[s1[x,t],t],D[r2[x,t],t],D[s2[x,t],t]};
(* --- parameters --- *)
ParameterS = {};
(* --- lists of the infinitesimals only treated if Info -> True --- *)
ListXi = {c1*x + c3*t + c4,
          2*c1*t + c2};
ListPhi = {-2*c1*r1[x,t],
           c3*x + c5,
           -2*c1*r2[x,t],
           c3*x + c6};
```

```
+----------------------------------------------------------+
|        Welcome to the MATHEMATICA program for the        |
|            calculation of the symmetry group             |
```

```
|                              by                              |
|                          G. Baumann                          |
|                      University of Ulm                       |
+--------------------------------------------------------------+


### Information: File LieSymmetry`Private`nmaeSystem not found.
                 Calculate the prolongation.
Loading data from  mad2.dgl


 --------------------------------------------------------------------
 Equation No. 1

   (0,1)           (1,0)         (1,0)                   (2,0)
 r1      [x, t] + r1      [x, t] s1      [x, t] + r1[x, t] s1      [x, t] = 0

 Equation No. 2

                                                     (1,0)      2     (1,0)      2
                                         (0,1)     r1      [x, t]    s1      [x, t]
 g1 r1[x, t] + r2[x, t] + s1      [x, t] + -------------- + -------------- -
                                                            2           2
                                                      8 r1[x, t]

      (2,0)
    r1      [x, t]
>   ------------- = 0
     4 r1[x, t]

 Equation No. 3

   (0,1)           (1,0)         (1,0)                   (2,0)
 r2      [x, t] + r2      [x, t] s2      [x, t] + r2[x, t] s2      [x, t] = 0

 Equation No. 4

                                                     (1,0)      2     (1,0)      2
                                         (0,1)     r2      [x, t]    s2      [x, t]
 r1[x, t] + g2 r2[x, t] + s2      [x, t] + -------------- + -------------- -
                                                            2           2
                                                      8 r2[x, t]

      (2,0)
    r2      [x, t]
>   ------------- = 0
     4 r2[x, t]

 --------------------------------------------------------------------
 Substitution No. 1

   (0,1)              (1,0)         (1,0)                      (2,0)
 r1      [x, t] -> -(r1      [x, t] s1      [x, t]) - r1[x, t] s1      [x, t]

 Substitution No. 2
```

```
   (0,1)
 s1     [x, t] ->

                   3             2              (1,0)     2
>   -(8 g1 r1[x, t]  + 8 r1[x, t]  r2[x, t] + r1     [x, t]  +

                  2   (1,0)     2                 (2,0)
>       4 r1[x, t]  s1     [x, t]  - 2 r1[x, t] r1     [x, t]) /

                2
>     (8 r1[x, t] )

 Substitution No. 3

   (0,1)              (1,0)         (1,0)                    (2,0)
 r2     [x, t] -> -(r2     [x, t] s2     [x, t]) - r2[x, t] s2     [x, t]

 Substitution No. 4

   (0,1)
 s2     [x, t] ->

                         2                3     (1,0)     2
>   -(8 r1[x, t] r2[x, t]  + 8 g2 r2[x, t]  + r2     [x, t]  +

                  2   (1,0)     2                 (2,0)
>       4 r2[x, t]  s2     [x, t]  - 2 r2[x, t] r2     [x, t]) /

                2
>     (8 r2[x, t] )

 --------------------------------------------------------------------
 --------------------------------------------------------------------
 Infinitesimals :

 Xi1 = c4 + c3 t + c1 x

 Xi2 = c2 + 2 c1 t

 Phi1 = -2 c1 r1[x, t]

 Phi2 = c5 + c3 x

 Phi3 = -2 c1 r2[x, t]

 Phi4 = c6 + c3 x
 --------------------------------------------------------------------

You are using the 2-th order of derivatives
in the calculation of prolongations.

Working on equation 1
  Enter ProlongationF
  Applying constraints
```

```
  Enter expand
  Enter search of coefficients
Working on equation 2
  Enter ProlongationF
  Applying constraints
  Enter expand
  Enter search of coefficients
Working on equation 3
  Enter ProlongationF
  Applying constraints
  Enter expand
  Enter search of coefficients
Working on equation 4
  Enter ProlongationF
  Applying constraints
  Enter expand
  Enter search of coefficients

  Enter simplifying equations

You are using the 4 equations of the system.

**** Number of determining equations before simplification
**** N_before = 408


List of factors cancelled in the calculation:

                2          2            3             3              4
>   {-r1[x, t] , r1[x, t] , -r1[x, t] , 2 r1[x, t] , -4 r1[x, t] ,

             4             2                    2
>    8 r1[x, t] , -2 r1[x, t]  r2[x, t], r1[x, t]  r2[x, t],

             2                     2                       3
>    2 r1[x, t]  r2[x, t], 4 r1[x, t]  r2[x, t], -4 r1[x, t]  r2[x, t],

             3                       4                      4
>    2 r1[x, t]  r2[x, t], -16 r1[x, t]  r2[x, t], 8 r1[x, t]  r2[x, t],

            2          2                       2                  2
>    -r2[x, t] , r2[x, t] , -2 r1[x, t] r2[x, t] , r1[x, t] r2[x, t] ,

                       2                     2            2         2
>    2 r1[x, t] r2[x, t] , 4 r1[x, t] r2[x, t] , -8 r1[x, t]  r2[x, t] ,

              2          2              2          2            2         2
>    -4 r1[x, t]  r2[x, t] , -2 r1[x, t]  r2[x, t] , 4 r1[x, t]  r2[x, t] ,

             2          2              3          2              3         2
>    8 r1[x, t]  r2[x, t] , -32 r1[x, t]  r2[x, t] , -16 r1[x, t]  r2[x, t] ,

              3          2              3          2            3         2
>    -8 r1[x, t]  r2[x, t] , -4 r1[x, t]  r2[x, t] , 4 r1[x, t]  r2[x, t] ,
```

```
               3          2              3          2              3          2
>    8 r1[x, t]  r2[x, t] , 16 r1[x, t]  r2[x, t] , 32 r1[x, t]  r2[x, t] ,

                3          2              4          2
>    48 r1[x, t]  r2[x, t] , -64 r1[x, t]  r2[x, t] ,

                 4          2              4          2
>    -32 r1[x, t]  r2[x, t] , -16 r1[x, t]  r2[x, t] ,

                4          2              4          2              4          2
>    -8 r1[x, t]  r2[x, t] , 16 r1[x, t]  r2[x, t] , 32 r1[x, t]  r2[x, t] ,

                4          2           3             3                         3
>    64 r1[x, t]  r2[x, t] , -r2[x, t] , 2 r2[x, t] , -4 r1[x, t] r2[x, t] ,

                         3               2          3              2          3
>    2 r1[x, t] r2[x, t] , -32 r1[x, t]  r2[x, t] , -16 r1[x, t]  r2[x, t] ,

                2          3              2          3             2          3
>    -8 r1[x, t]  r2[x, t] , -4 r1[x, t]  r2[x, t] , 4 r1[x, t]  r2[x, t] ,

               2          3              2          3              2          3
>    8 r1[x, t]  r2[x, t] , 16 r1[x, t]  r2[x, t] , 32 r1[x, t]  r2[x, t] ,

                2          3              3          3              3          3
>    48 r1[x, t]  r2[x, t] , -8 r1[x, t]  r2[x, t] , 16 r1[x, t]  r2[x, t] ,

                 4          3              4          3           4
>    -32 r1[x, t]  r2[x, t] , 64 r1[x, t]  r2[x, t] , -4 r2[x, t] ,

               4                          4                        4
>    8 r2[x, t] , -16 r1[x, t] r2[x, t] , 8 r1[x, t] r2[x, t] ,

                 2          4              2          4
>    -64 r1[x, t]  r2[x, t] , -32 r1[x, t]  r2[x, t] ,

                 2          4             2          4              2          4
>    -16 r1[x, t]  r2[x, t] , -8 r1[x, t]  r2[x, t] , 16 r1[x, t]  r2[x, t] ,

                2          4              2          4               3          4
>    32 r1[x, t]  r2[x, t] , 64 r1[x, t]  r2[x, t] , -32 r1[x, t]  r2[x, t] ,

                3          4
>    64 r1[x, t]  r2[x, t] }

**** Number of determining equations after simplification
**** N_after = 0.

**** Number of determining equations after simplification
**** N_after = 0.

Calculation of the commutator table.

 Basis of the Lie algebra.
 --------------------------------------------------------------------
```

```
        (0,0,0,0,0,1)
x Funcf             [x, t, r1, s1, r2, s2] +

            (0,0,0,1,0,0)
>     x Funcf             [x, t, r1, s1, r2, s2] +

            (1,0,0,0,0,0)
>     t Funcf             [x, t, r1, s1, r2, s2] -> V[3]

            (0,0,0,0,1,0)
-2 r2 Funcf             [x, t, r1, s1, r2, s2] -

               (0,0,1,0,0,0)
>     2 r1 Funcf             [x, t, r1, s1, r2, s2] +

              (0,1,0,0,0,0)
>     2 t Funcf             [x, t, r1, s1, r2, s2] +

            (1,0,0,0,0,0)
>     x Funcf             [x, t, r1, s1, r2, s2] -> V[1]

      (0,0,0,0,0,1)
Funcf             [x, t, r1, s1, r2, s2] -> V[6]

      (0,0,0,1,0,0)
Funcf             [x, t, r1, s1, r2, s2] -> V[5]

      (1,0,0,0,0,0)
Funcf             [x, t, r1, s1, r2, s2] -> V[4]

      (0,1,0,0,0,0)
Funcf             [x, t, r1, s1, r2, s2] -> V[2]

--------------------------------------------------------------------

Structure of the Lie algebra.
--------------------------------------------------------------------

Ideal 0 :  {V[1], V[2], V[3], V[4], V[5], V[6]}

Ideal 1 :  {V[2], V[3], V[4], V[5], V[6]}

Ideal 2 :  {V[4], V[5], V[6]}

Ideal 3 :  {solvable algebra}

--------------------------------------------------------------------

Casimier(s) of the Lie algebra.

{V[5], V[6]}

--------------------------------------------------------------------
```

```
Commutator table of the Lie algebra.

-------------------------------------------------------------------
0              2 V[2]         -V[3]          V[4]           0

>       0

-2 V[2]        0              -V[4]          0              0

>       0

V[3]           V[4]           0              V[5] + V[6]    0

>       0

-V[4]          0              -V[5] - V[6]   0              0

>       0

0              0              0              0              0

>       0

0              0              0              0              0

>       0
-------------------------------------------------------------------

Commutator table is contained in the list LieTable.

Nonzero structure Constants.
-------------------------------------------------------------------

 C[1,2,2] = 2
 C[1,3,3] = -1
 C[1,4,4] = 1
 C[2,1,2] = -2
 C[2,3,4] = -1
 C[3,1,3] = 1
 C[3,2,4] = 1
 C[3,4,5] = 1
 C[3,4,6] = 1
 C[4,1,4] = -1
 C[4,3,5] = -1
 C[4,3,6] = -1

-------------------------------------------------------------------

Metric of the structure constants

            6   0   0   0   0   0

            0   0   0   0   0   0

            0   0   0   0   0   0
```

```
        0   0   0   0   0   0

        0   0   0   0   0   0

        0   0   0   0   0   0

Determinant of the metric

det = 0

Structure constants are contained in the list LieStructure.

Results are collected in the list FinalResult.
```

5 Benchmark for a family of Korteweg de Vries equations

A comparison of the time used by the programs `Lie[]` and `SYMMGRP.MAC` [7] for obtaining the determining equations in its simplest form is given for a family of Korteweg de Vries (KdV) equations. The program of Champagne et al [7] runs on a VAX 6440 under VMS 5.0. The `Lie[]` program in *Mathematica* was tested on a 386Pc, 486Pc, a NeXT computer, and on a Hewlett Packard 9000 computer. The time used to calculate the infinitesimals was measured by MACSYMA and *Mathematica* functions. We determined the symmetries for the original KdV equation, for the modified KdV equation, for cylindrical KdV equation, and for the λ-KdV equation. The corresponding equations of motion read

$$\begin{aligned} u_t + u_{xxx} + uu_x &= 0 \qquad \text{KdV} \\ u_t + u_{xxx} + u^2 u_x &= 0 \qquad \text{mKdV} \\ u_t + u_{xxx} + uu_x + \frac{u}{2t} &= 0 \qquad \text{cKdV} \\ u_t + u_{xxx} - 6u^2 u_x + 6\lambda u_x &= 0 \qquad \lambda\text{-KdV} \end{aligned} \tag{96}$$

The infinitesimals of the models are given in Table I. In Table II the times for deriving the determining equations are given. It is obvious that the time on a specific computer is nearly the same for the different models. The slowest production of results was by the VAX6440 computer on which the MACSYMA program `SYMMGRP.MAC` of Champagne et al [7] was used. A factor of about two faster is the *Mathematica* program `Lie[]` on the slowest computer a 386Pc/20MHz. The time needed to derive the equations is reduced by a factor five for a NeXT computer. We note that the infinitesimals given for the cKdV are only partly derivable by using `Lie[]`. Especially the symmetries related to c_3 and c_4 do not follow from the ansatz used in `Lie[]`.

6 Conclusions

We presented a *Mathematica* program that can be used for the calculation of the symmetry group of a system of differential equations. The program has the following essential features:

1. It is applicable to a system of m equations of order k, with q unknown and p independent variables.

2. It computes a system of determining equations that is free from redundancy and partially solves it in the sense that higher order equations which are consequences of lower order equations are automatically eliminated.

3. It needs very few data and is simple to use.

4. One can find a specific solution for the infinitesimals by using either a manual procedure or a polynomial ansatz in the dependent and independent variables.

5. It allows a classification of the Lie–agebra.

References

1. P.J. Olver, Applications of Lie Groups to Differential Equations. (Springer, New York, 1986)
2. G.W. Bluman and S. Kumei, Symmetries and Differential Equations. (Springer, New York, 1989)
3. B.B. Kadomtsev and V.I. Petviashvili, Sov. Phys. Dokl. **15**, 539 (1970)
4. S. Wolfram, Mathematica: A System Doing Mathematics by Computer. (Addison-Wesley, Redwood City, 1991)
5. F. Schwarz, Comput. Phys. Comm. **39**, 285 (1986)
6. F. Schwarz, SIAM Review **30**, 450 (1988)
7. B. Champagne, W. Hereman and P. Winternitz, CRM-1689, Université de Montréal, (1990)
8. S.V. Manakov, Soviet Phys. JETP **38**, 248 (1974)
9. V.E.Zakharov and E.I. Schulman, Physica **D4**, 270 (1982)
10. T.F. Nonnenmacher, G. Dukek and G. Baumann, Lett. Nuovo Cim. **36**, 453 (1983)
11. G. Baumann and T.F. Nonnenmacher, J. Math. Phys. **28**, 1250, (1987)
12. G. Baumann, W.G. Glöckle and T.F. Nonnenmacher, Proc. R. Soc. Lond. A **434**, 263 (1991)

7 Program Listing

File lie.m

```
(*:Mathematica Version: 2.0 *)

(*:Name: Calculus`LieSymmetry` *)

(*:Title: Lie Symmetries *)

(*:Author: Gerd Baumann
           Abteilung fuer Mathematische Physik
           Universitaet Ulm
           Albert-Einstein-Allee 11
           D-7900 Ulm
           Germany

      phone:    +731-502-2996
      fax:      +731-502-3003
      e-mail:   bau@theophys.physik.uni-ulm.de
*)

(*:Summary:

The program allows the calculation of infinitesimal symmetries for systems
of partial and ordinary differential equations. If the symmetries are known
the related Lie-algebra is derivable and a clssification of the algebra is
performed.

*)

(*:Keywords:

Lie;
Lie symmetries;
symmetries;
transformations;
differential equations;
```

```
partial differential equations;
ordinary differential equations;
Lie algebre
derived algebras

*)

(*:Requirements: none. *)

(*:Sources:
      P. Olver, Applications of Lie Groups to Differential Equations
               Springer, New York, 1986
      G.W. Bluman and S. Kumei, Symmetries and Differential Equations
               Springer, New York, 1989
      G. Reid, Euro. Jnl. of Applied Mathematics 2, 293-318, (1991)
      B. Champagne, W. Hereman, and P. Winternitz, CRM-1689 Report,
               Centre de Recherches Mathematiques, Universite de Montreal
               (1990)
*)

(*:History:
      Version 1.0 : without options

      Version 2.0 : uses options as arguments
                    solves the determining equations
                    calculates the Lie-algebra and related things
*)


(*:Discussion:
      The purpos of this program is to derive the determining equations for
      Lie-point symmetries. The determining equations are derived in a form
      simplified as far as possible. The program allows an automatic solution
      of the determining equations by using a polynomial ansatz in the
      dependent and independent variables. If the infinitesimals (symmetries)
      of the symmetry transformation is known a classification of the
      related Lie-algebra is possible. The Lie-algebra is directly related
      to the symmetry group of the equations which can be used to construct
      an explicit solution of the equations of motion considered. Lie's
      symmetry method is usable to construct solutions for linear and
      especially for nonlinear systems of differential equations. The program
      also provides a classification of the Lie-algebre by derived algebras.

*)

BeginPackage["LieSymmetry`"]

Clear[CommentBatch,Lie,ProlongationF,MaxD,
      SearchCoefficients,SimplifyEquationS,SubstituteInfinitesimals,
      CloseLog,OpenLog,LTest,DerivativeElimination,Dxt,PowerT,
      CancelPreFactor,SubstRule,
      SearchDeri,Simple1,TresseOrder,OrdDeri,Substitution,
      Df,SubstRule,groupOrder,Commutator,CommutatorTable,
      StructureConstants,VectorBasis,MetricStructure,
      LateX,AnsatzInfinitesimals,InfinitesimalsAnsatz,
```

```
      Adjungation,KommutFunction,AdAlgebra,ParameterQ,
      EliminFactors,SingleTermQ]


Print["      + ------------------------------------------------------- +"];
Print["      |           Version 2.0 of Lie for Mathematica 2.0        |"];
Print["      |                  Copyright 1992 G. Baumann              |"];
Print["      |                                                         |"];
Print["      | ------------------------------------------------------- |"];
Print["      | *** The functions defined in the package LieSymmetry are |"];
Print["      | ------------------------------------------------------- |"];
Print["      |                                                         |"];
Print["      | Lie            : calculates the determining equations   |"];
Print["      | ProlongationF  : determines the k-th prolongation       |"];
Print["      | CloseLog       : close the log file                     |"];
Print["      | OpenLog        : open the log file                      |"];
Print["      |                                                         |"];
Print["      | ------------------------------------------------------- |"];
Print["      |                                                         |"];
Print["      | To get more information type ?name                      |"];
Print["      |                                                         |"];
Print["      + ------------------------------------------------------- +"];
Print[" "];

Lie::usage="Lie[Param1_,options__] is the main function which calculates the
determining equations for symmetries of a given system of partial or ordinary
differential equations. The parameters Param1 and options are used to
control the programm execution. If \n
         Param1 = name : all data and informations are \n
                         read from file name \n
\n
Options of the program are:
\n
--------------------------
\n
\n
ProlongationOrder  : specifies the largest derivative used in the
calculation of the prolongation
\n
Warning : if set True gives warnings if a term is eliminated in the
determining equations
\n
Info : if set True the program uses the information for the
characteristics given in the lists ListXi and ListPhi
\n
EquationsS : specifies the equations used in the calculation if the list
contains All (default) all equations are used
\n
Trace : if set to True the calculation is traced (default) otherwise no
tracing is given
\n
Tresse : if set to True a standard form of equations is calculated. Default
value is False.
\n
CommutatorTable: if set to True the table of the Lie algebre is calculated.
```

```
Default is False.
\n
StructureConstants: if set to True the structure constants of the Lie algebra
are calculated. Default is False.
\n
LaTeX: if set to True an output in LaTeX form is stored in a file.
All information on infinitesimals, structure constants and the commutator
table is written in LaTeX. Default is False.
\n
Ansatz: Specifies the polynomial order in the dependent and
independent variables for an ansatz of the infinitesimals. The default
value is Ansatz->{0}.
\n
The organization of input file for batch mode is demonstated by the
diffusion equation \n
\n
\n
Example: \n
(* ---    Diffusion Equation      --- *)                           \n
(*-----------------------------------*)                             \n
(* --- independent variables --- *)                                 \n
IndepVar={x,t};                                                     \n
(* --- dependent variables --- *)                                   \n
DependVar={u};                                                      \n
(* --- equations  --- *)                                            \n
EqList = {D[u[x,t],t] - D[u[x,t],{x,2}]}                            \n
(* --- substitutions --- *)                                         \n
SubsList = {D[u[x,t],t]};                                           \n
(* --- parameters --- *)                                            \n
ParameterS = {};                                                    \n
(* --- lists of the infinitesimals are only treated if Info -> True --- *)
\n
ListXi = {c1 + c4*x + 2*c5*t + 4*c6*x*t, \n
          c2 + 2*c4*t + 4*c6*t^2};        \n
ListPhi = {(c3-c5*x-2*c6*t-c6*x^2)*u[x,t]+f[x,t]};\n
"

ProlongationF::usage = "ProlongationF[delta_,dependVar_List,indVar_List,order_]
determines the prolongation of DELTA to the order ORDER for a set of
dependent and independent variables. This procedure assumes that
infinitesimals Xi and Phi are given in a list."

(* --- declare global --- *)

IndepVar::usage="List of independent variables."
DependVar::usage="List of dependent variables."
Sublisteqs::usage="List containing the equations treated in the calculation."
Warnings::usage="Give warnings on the elimination of coeffficients
if set True or suppress warnings if set False."
InfoGiven::usage="Serves as a flag in treating the infinitesimals. If set
True the given information in the lists ListXi and ListPhi is used in the
calculation of the determining equations."
EqList::usage="List contains the differential equations."
SubsList::usage="List contains the substitutions for the equations."
ParameterS::usage="List contains the parameters treated as a constant."
```

```
HighestDerivatives::usage="Greatest order of diferentiation in the equations.
If set to All the greatest derivative from the set of equations is
used in the calculation of prolongations."
ListXi::usage="List contains the infinitesimals for
independent variables."
ListPhi::usage="List contains the infinitesimals for dependent
variables."
Result2::usage="List contains the determining equations of the
infinitesimals."
FinalResult::usage="List contains all information on the equation in list
form."
xi::usage="Internal variables to represent the infinitesimals of the
independent variables."
phi::usage="Internal variables to represent the infinitesimals of the
dependent variables."
LieTable::usage="Contains the commutator table of the Lie algebra."
LieStructure::usage="Contains the structure constants of the Lie algebra."
VBasis::usage="Represents the basis of the Lie algebra."
CommTable::usage="Commutator table of the Lie algebra."

(* --- global options --- *)

ProlongationOrder::usage="Specifies the largest derivative used
in the calculation of the prolongation."
Warning::usage="If set True gives warnings if a term is eliminated
in the determining equations."
Info::usage="If set True the program uses the information for the
characteristics given in the lists ListXi and ListPhi."
EquationS::usage="Specifies the equations used in the calculation
if the list contains All (default) all equations are used."
Trace::usage="If set to True the calculation is traced (default)
otherwise no tracing is given."
Tresse::usage="If set to True a standard form of equations is
calculated. Default value is False."
CommutatorTable::usage="If set to True the table of the Lie algebre
is calculated. Default is False."
StructureConstants::usage="If set to True the structure constants of
the Lie algebra are calculated. Default is False."
LaTeX::usage="If set to True an output in LaTeX form is stored in a file.
All information on infinitesimals, structure constants and the commutator
table is written in LaTeX. Default is False."
Ansatz::usage="Specifies the polynomial order in the dependent and
independent variables for an ansatz of the infinitesimals. The default
value is Ansatz->{0}."


Begin["`Private`"]

CommentBatch[]:=Block[{},
      Print[" "];
      Print[" +----------------------------------------------------------+"];
      Print[" |        Welcome to the MATHEMATICA program for the        |"];
      Print[" |            calculation of the symmetry group             |"];
      Print[" |                            by                            |"];
      Print[" |                        G. Baumann                        |"];
```

```
      Print[" |                    University of Ulm                     |"];
      Print[" +----------------------------------------------------------+"];
      Print[" "]
      ]

(* --- substitution rule --- *)
SubstRule[x_,y_]:=x->y

(* --- Default values for options --- *)
Options[Lie] = {ProlongationOrder->All,
                Warning->True,
                Info->False,
                EquationS->{All},
                Trace->True,
                Tresse->False,
                CommutatorTable->False,
                StructurConstants->False,
                LaTeX->False,
                Ansatz->{0}}


(* --- main program --- *)
Lie[Param1_,options___]:=Block[
      {filetest,fileTestSystem,teXfilename,logFilename},
      Listexpr1 = {};
      FinalResult = {};

(* --- filename creation --- *)
      nameHelp = StringTake[Param1,-4];
      nameSystem = StringReplace[Param1,nameHelp->".deq"];
      teXfilename = StringReplace[Param1,nameHelp->".tex"];
      logFilename = StringReplace[Param1,nameHelp->".lie"];

      OpenLog[logFilename];
      CommentBatch[];

(* --- look for calculations done at an earlier run --- *)
      fileTestSystem = FileNames[nameSystem];
      If[Not[fileTestSystem === {}],
         Print["   "];
         Print[" Loading previous calculations from  ",nameSystem];
         Print["  "];
         Clear[IndepVar,DependVar,SubsList,ListPhi,ListXi];
         Get[nameSystem],
            Print["   "];
            Print[" ### Information: File ",nmaeSystem," not found."];
            Print["                  Calculate the prolongation."];
            prevInfo = False
            ];


(* --- Options --- *)

{HighestDerivatives,Warnings,InfoGiven,Sublisteqs,Param2,
standarD,tablE,structurE,laTeX,aNsatz} =
```

```
{ProlongationOrder,Warning,Info,EquationS,Trace,
Tresse,CommutatorTable,StructureConstants,LaTeX,Ansatz} /. {options}
                                                        /. Options[Lie];

(* --- Instructions which serve to read in the data --- *)

      filetest = FileNames[Param1];
      If[Not[filetest === {}],
         Print[" Loading data from  ",Param1];
         Print["  "];
         Get[Param1],
            Print[" ### Error File ",Param1," not found"];
            CloseLog[];
            Return["Done"]];
      Liste = {};
      P = Length[IndepVar];
      Q = Length[DependVar];
      M = Length[EqList];
      Do[AppendTo[Liste,EqList[[i]]],{i,1,M}];
      Listv = {};
      Do[AppendTo[Listv,SubsList[[i]]],{i,1,M}];
      If[InfoGiven,
         XiL = {};
         Do[AppendTo[XiL,ListXi[[i]]],{i,1,P}];
         PhiL = {};
         Do[AppendTo[PhiL,ListPhi[[i]]],{i,1,Q}]
      ];
      MaxDeri = {};
      Do[Do[AppendTo[MaxDeri,MaxD[Liste[[i]],DependVar[[j]]]],{i,1,M}],
      {j,1,M}];
      MaxDeri = Max[MaxDeri];
      If[HighestDerivatives === All,HighestDerivatives = MaxDeri];

(* --- Instructions used to set up the environment --- *)

      If[Sublisteqs[[1]] === All,
         R = M;
         Subliste = Liste,
           R = Length[Sublisteqs];
               If[R <= M, Subliste = {};
                          Scase = Cases[Sublisteqs,e_[n_]->{n}];
                          Do[AppendTo[Subliste,Liste[[Scase[[i]] ]]],
                          {i,1,R}];
                          Subliste = Flatten[Subliste],
                  Print[" ### Error in Sublisteqs ? ### "];
                  Return[" ### Execution terminated !!!!"]
               ]
      ];
(* --- Creation of dependent and independent variables --- *)
      Listx = IndepVar;
      Listu = {};
      Do[AppendTo[Listu,DependVar[[i]][Delete[Listx,0]]],{i,1,Q}];
      Listxu = {};
      Listxu = Flatten[Append[Listx, Listu]];
      Xi ={};
```

```
      Do[AppendTo[Xi,xi[k][Delete[Listxu,0]]],{k,1,P}];
      Phi = {};
      Do[AppendTo[Phi,phi[k][Delete[Listxu,0]]],{k,1,Q}];
      If[M == 0, Return["Done"]];
(* --- Solve equations to find the substitutions --- *)
      Algsols = Solve[Liste == 0,Listv];

(* --- Output the equations, substitutions, and infinitesimals --- *)
      Print[" "];
Print["  -------------------------------------------------------------------"];
      Do[
      Print["  Equation No. ",i];
      Print["   "];
      Print["  ",Liste[[i]]," = 0"];
      Print["   "],
      {i,1,Length[Liste]}];
Print["  -------------------------------------------------------------------"];
      Do[
      Print["  Substitution No. ",i];
      Print["   "];
      Print["  ",Algsols[[1,i]] ];
      Print["   "],
      {i,1,Length[Algsols[[1]]]}];
Print["  -------------------------------------------------------------------"];
      If[InfoGiven,
Print["  -------------------------------------------------------------------"];
Print["  Infinitesimals :"];
      Do[
      Print["   "];
      Print["  Xi",i," = ",XiL[[i]]],{i,1,Length[XiL]}];
      Do[
      Print["   "];
      Print["  Phi",i," = ",PhiL[[i]]],{i,1,Length[PhiL]}];
Print["  -------------------------------------------------------------------"]
       ];
   Print["  "];
   Print[" You are using the ",HighestDerivatives,"-th order of derivatives "];
   Print[" in the calculation of prolongations."];
   Print[" "];

(* --- save some results in the list FinalResults --- *)
(* --- equations of motion are the first [[1]] --- *)
      AppendTo[FinalResult,EqList];
(* --- side condititons are the second [[2]] --- *)
      AppendTo[FinalResult,Algsols];
(* --- infinitesimals are the third [[3]] the order is Xi first Phi
       last --- *)
      AppendTo[FinalResult,Flatten[Prepend[ListPhi,ListXi]]];




(* --- Instructions which control the execution of the computation --- *)
      Denum = {};
      ListExpr = {};
```

```
If[Not[prevInfo],

If[Not[Param2],
   Do[
      Expr = Subliste[[i]];
      If[Not[Expr === 0],Expr =
      ProlongationF[Expr,DependVar,IndepVar,HighestDerivatives]];
      If[ Not[Expr === 0], Expr = Expr /. Algsols[[1]]];
      If[ Not[Expr === 0], Expr = Expand[Expr]];
      If[ Not[Expr === 0], Expr = Together[Expr]];
      If[ Not[Expr === 0], Expr = Cancel[Expr]];
      If[ Not[Expr === 0], AppendTo[Denum,Denominator[Expr]]];
      If[ Not[Expr === 0], Expr = Expand[Numerator[Expr]]];
      AppendTo[ListExpr,SearchCoefficients[Expr]];
      ListExpr = Flatten[ListExpr],
   {i,1,R}],
   Do[
      Print[" Working on equation ",i];
      Expr = Subliste[[i]];
      Print["   Enter ProlongationF "];
      Expr = ProlongationF[Expr,DependVar,IndepVar,HighestDerivatives];
      Print["   Applying constraints "];
      Expr = Expr /. Algsols[[1]];
      Print["   Enter expand "];
      Expr = Expand[Expr];
      Expr = Together[Expr];
      Expr = Cancel[Expr];
      AppendTo[Denum,Denominator[Expr]];
      Expr = Expand[Numerator[Expr]];
      Print["   Enter search of coefficients "];
      AppendTo[ListExpr,SearchCoefficients[Expr]];
      ListExpr = Flatten[ListExpr],
   {i,1,R}];
Print[" "];
Print["   Enter simplifying equations "];
Print[" "]
]

];

If[R == M,
      If[M >= 2,
         Print[" You are using the ",M," equations of the system."];
         Print["  "]]];
If[R < M,
   If[R >= 2,
    Print[" You are using only the ",R," equations:
          ",Sublisteqs];
    Print[" of the given system consisting of ",M," equations in total."];
    Print["   "]],

If[R == 1,
   Print[" You are using only the equation: ",Sublisteqs];
   Print[" of the given system consisting of ",M," equations in total."];
   Print["   "]]
```

```
 ];

Len = Length[ListExpr];

If[Len != 0,
Print[" **** Number of determining equations before simplification"];
Print[" **** N_before = ",Len];
Print["  "];
             SimplifyEquationS[ListExpr];
             prevInfo = True;
             Save[nameSystem,prevInfo,Result2];
             If[InfoGiven,Result2 = Substitution[]];
             If[standarD,Result2 = Simple1[Result2]],

             If[prevInfo,Result2 = Substitution[]];
             ListExpr = {}];

If[Listexpr1 != {},
               Print[" List of factors cancelled in the calculation:
                     ",Union[Listexpr1]]];
Print[" "];
Print[" **** Number of determining equations after simplification"];
Print[" **** N_after = ",Length[Result2],"."];

If[Result2 != {},
   Print[" **** These determining equations are stored in Result2.
   ****"];
   Print["  "];
   Do[Print["  Equation No. : ",i];
Print["  -------------------------------------------------------------------"];
      Print["  ",Result2[[i]]," = 0" ];
      Print[" "];
Print["  -------------------------------------------------------------------"],
   {i,1,Length[Result2]}]
];

Print[" "];
Print[" **** Number of determining equations after simplification"];
Print[" **** N_after = ",Length[Result2],"."];
(* --- determining equations are the fourth [[4]] --- *)
AppendTo[FinalResult,Result2];
If[Result2 != {},
   Print[" **** The determining equations are stored in Result2.
   ****"]];

(* --- make an ansatz for the infinitesimals of polynomial form --- *)
If[aNsatz[[1]]  > 0 && Not[InfoGiven],
   InfinitesimalsAnsatz[Param1,Result2] ];

(* --- calculate algebraic properties --- *)
  If[InfoGiven && tablE,
     LieTable = CommutatorTable[];

(*    Print[" "];
Print[" Adjoint algebra of the Lie algebra."];
```

```
      Print[" "];
      Print["  -------------------------------------------------------------------"];
      Print["  ",MatrixForm[AdAlgebra[4]]];
      Print["  -------------------------------------------------------------------"];
*)
      Print[" "];
      Print[" Commutator table of the Lie algebra."];
      Print[" "];
      Print["  -------------------------------------------------------------------"];
      Print["  ",MatrixForm[LieTable]];
      Print["  -------------------------------------------------------------------"];
        Print["  "];
        Print["  Commutator table is contained in the list LieTable."]
        ];
(* --- commutatortable is the sixt [[6]] --- *)
      AppendTo[FinalResult,LieTable];

(* --- calculate the structure constants --- *)
        If[InfoGiven && structurE,
           LieStructure = StructureConstants[];
           Metric = MetricStructure[LieStructure];
        Print["  "];
        Print["  Metric of the structure constants"];
        Print["  "];
        Print["              ",MatrixForm[Metric]];
        Print["  "];
        Print["  Determinant of the metric"];
        Print["  "];
        Print["  det = ",Det[Metric]];
        Print["  "];
        Print["  Structure constants are contained in the list LieStructure."]
        ];
(* --- structure constants are the seventh [[7]] --- *)
      AppendTo[FinalResult,LieStructure];
      AppendTo[FinalResult,Metric];

(* --- output the results in LaTeX --- *)
      If[laTeX,LateX[teXfilename,FinalResult]];

        Print["  "];
        Print["  Results are collected in the list FinalResult."];
(*      Save[nameSystem,FinalResult]; *)
      CloseLog[];
      Return["Done"]
      ];



(* --- Determines maximal order of differentiation --- *)
MaxD[expr_,dependVar_]:=Block[{lengthExpr,level,case,order,x,y},
            lengthExpr = Depth[expr];
            level = Level[expr,lengthExpr];
            level = AppendTo[level,expr];
            case = Cases[level,Derivative[x___][dependVar][y___]->{x}];
            case = Apply[Plus,case,2];
```

```
            If[case == {},order = 0,order = Max[case]]
            ]



(* --- Applies the k-th prolongation of the vector field  --- *)
(* --- v to a function depending on the independent and   --- *)
(* --- dependent variables.                               --- *)
(* --- Formulas are identical with (2.38) and (2.39) of   --- *)
(* --- P. Olver (1986) p. 113                             --- *)

ProlongationF[delta_,dependVar_List,indVar_List,order_] := Block[
{lind,indepVar,lenin,lendep,dlist,d1list,Prlist,denslist,
 Jcase,Jlist,erg1,erg2,Phij},

      DeriList[x_,y_]:={x,y};

(* --- Length of lists for independent and dependent variables --- *)
      lind = indVar;
      indepVar = Delete[lind,0];
      lenin = Length[lind];
      lendep = Length[dependVar];
      erg = 0;

(* --- Calculation of the prolongation; sum over dependent variables --- *)
      Do[
      dlist = {dependVar[[ndep]][indepVar]};
      d1list = {delta};
(* --- Loop over orders --- *)
      Do[
      Prlist = {};
(* --- Determine all derivatives of the dependent variables up   --- *)
(* --- to order n and cancel all multiple terms                  --- *)
            dlist = Outer[D,dlist,{indepVar}];
            dlist = Union[Flatten[dlist]];
(* --- Calculate the derivatives with respect to the gradients --- *)
      denslist = Flatten[Outer[D,d1list,dlist]];
         Do[If[ Not[ denslist[[i]] === 0],
           Jcase = Cases[{dlist[[i]]},
                        Derivative[x___][dependVar[[ndep]]][y___]->{{y},{x}}];
              Jlist = {};
              AppendTo[Jlist, Thread[ DeriList[Jcase[[1,1]],Jcase[[1,2]] ] ]];
              Jlist = Flatten[Jlist,1];
              Jlist = Delete[Jlist,0];
(* --- Equation (2.39) of Olver --- *)
           Phij = D[
                    Phi[[ndep]] -
                    Sum[Xi[[k]]*D[dependVar[[ndep]][indepVar],lind[[k]]],
                       {k,1,lenin}],Jlist] +
                    Sum[Xi[[k]]*D[D[dependVar[[ndep]][indepVar],Jlist],
                       lind[[k]]],{k,1,lenin}];
           AppendTo[Prlist,Phij*D[ delta ,
                               D[dependVar[[ndep]][indepVar],Jlist ] ]]
           ],
         {i,1,Length[denslist]}];
```

```
      erg = erg + Apply[Plus,Prlist],
      {n,1,order}],
      {ndep,1,lendep}];
(* --- Equation (2.38) of Olver --- *)
      erg1 = Sum[Xi[[k]]*Dxt[delta,lind[[k]]],{k,1,lenin}] ;
      erg2 = Sum[Phi[[ndep]]*D[delta,dependVar[[ndep]][indepVar]],
             {ndep,1,lendep}];
      erg = erg + erg1 + erg2;
      erg = Expand[erg]
      ];

(* --- Special derivative for independent variables --- *)
Dxt[expr_,varin_]:= Block[{level,d1term,dterms,sub1,sub2,ExprH},

      SearchCoefficients[expr];
      dterms = terms;
      d1term = {};
      Do[If[FreeQ[dterms[[i]],Times],
            AppendTo[d1term,dterms[[i]]],
            level = Level[dterms[[i]],1];
            AppendTo[d1term,level]
         ],
      {i,1,Length[dterms]}];
      sub1 = Flatten[d1term];
      sub2 = {};
      Do[AppendTo[sub2,Unique["derIabl"]],{i,1,Length[sub1]}];

      ExprH = expr /. Thread[SubstRule[sub1,sub2]];
      ExprH = ExprH /. Thread[SubstRule[Listu,DependVar]];
      ExprH = D[ExprH,varin];
      ExprH = ExprH /. Thread[SubstRule[DependVar,Listu]];
      ExprH = ExprH /. Thread[SubstRule[sub2,sub1]]
      ]



(* --- Search for the coefficients --- *)

SearchCoefficients[expr_]:=Block[{level,lenlev,cf1,ExprH},
      ExprH = Expand[expr];
(* --- Test if single term or sum --- *)
      If[FreeQ[ExprH,Plus],level = {ExprH},
         level = Level[ExprH,1]
      ];
      lenlev = Length[level];
(* --- Put the derivatives of the dependent variables in a list --- *)
      terms = {};
      Do[AppendTo[terms,Searchf2[ level[[i]] ] ], {i,1,lenlev}];
      terms = Flatten[terms];
      terms = Reverse[Union[terms]];
      terms = Sort[terms,PowerT[#1] > PowerT[#2]&];
(* --- Determine the coefficients --- *)
      listcof = {};
      Do[cf1 = Coefficient[ ExprH,terms[[i]] ];
        AppendTo[listcof,cf1];
```

```
       ExprH = Expand[ExprH - cf1*terms[[i]] ],
     {i,1,Length[terms]}];
     If[Not[ExprH === 0],AppendTo[listcof,{ExprH}]];
     listcof = DeleteCases[listcof,0];
     listcof = Flatten[listcof]
     ]

(* --- Match the derivatives --- *)
Searchf1[expr_] :=Block[{case,var,arg,der},
    fs1[x_,y_]:={x,y};
    case = Cases[{expr}, (mult_.)*Derivative[x___][u_][y___]^(n_.) ->
    {u, mult, n, {y}, {x}}];
    If[case != {} && Not[Intersection[DependVar,{case[[1,1]]}] == {}],
    var = Delete[Thread[fs1[case[[1,4]],case[[1,5]] ] ] ,0];
    arg = case[[1,1]][Delete[case[[1,4]],0]];
    der = D[arg,var]^case[[1,3]],
    {}]
    ]

(* --- Extract the derivatives in u,v,... ---*)
Searchf2[expr_]:=Block[{level,lenlev,ExprHi,ExprH,result},
      level = Level[expr,1];
      lenlev = Length[level];
      ExprHi = expr;
      result = 1;
      Do[ExprH = Searchf1[ExprHi];
         If[Not[ExprH === {}],
         ExprHi = Expand[ExprHi/ExprH];
         result = result*ExprH
         ],
      {i,1,lenlev}];
      result
      ]


(* --- Power of derivatives --- *)
PowerT[expr_] := Block[{ExprH,caseP,levelP},
   ExprH = expr;
   caseP ={};
     If[FreeQ[ExprH, Times], levelP = {ExprH},
        levelP = Level[ExprH, 1]];
        Do[AppendTo[caseP,
        Cases[{levelP[[i]]}, (mult_.)*Derivative[x___][u_][y___]^(exp_.) ->
          {exp}]], {i, 1, Length[levelP]}];
      caseP = Flatten[caseP];
      Apply[Plus, caseP]]


(* --- Simplification of determining equations --- *)

SimplifyEquationS[LExpr_]:=Block[
      {lExpr,lenExpr,L1,L2},
(* --- Unify the list LExpr --- *)
      lExpr = Union[LExpr];
(* --- Searching monomials in the list lExpr and put them in Result1 --- *)
```

```
(* --- Elements containing  Plus are collected in list Result2      --- *)

      lenExpr = Length[lExpr];
      Result1 = {};
      Result2 = {};
      Do[If[FreeQ[lExpr[[i]], Plus], AppendTo[Result1,lExpr[[i]] ],
                                     AppendTo[Result2,lExpr[[i]] ]],
      {i,1,lenExpr}];
(* --- Extract factors from the elements in list Result1 and create --- *)
(* --- substitution rules in auxiliary list L1                      --- *)
      L1 = {};
      Result1 = CancelPreFactor[Result1];
      L1 = Table[0,{i,1,Length[Result1]}];
      L1 = Thread[SubstRule[Result1,L1]];
(* --- Apply the substitutions on list Result2 --- *)
      Result2 = Result2 /. L1;
      AppendTo[Result2,Result1];
      Result2 = DeleteCases[Result2,0];
      Result2 =Flatten[Result2];

      Result1 = {};
      Do[If[FreeQ[Result2[[i]],Plus] && Head[Result2[[i]] ] == Times,
            AppendTo[Result1,Result2[[i]]]
          ],
      {i,1,Length[Result2]}];

      L1 = {};
      Result1 = CancelPreFactor[Result1];
      L1 = Table[0,{i,1,Length[Result1]}];
      L1 = Thread[SubstRule[Result1,L1]];
(* --- Apply the substitutions on list Result2 --- *)
      Result2 = Result2 /. L1;
      AppendTo[Result2,Result1];
      Result2 = DeleteCases[Result2,0];
      Result2 =Flatten[Result2];

      Result2 = Union[Result2];
      Result2 = Sort[Result2,FreeQ[#1,Plus]&];

(* --- Elimination of derivatives --- *)
      Do[ DerivativeElimination[i],
      {i,1,HighestDerivatives}]
      ]


(* --- Check the given infinitesimals --- *)
      Substitution[]:=Block[{lenl2},
      If[!InfoGiven,Result2,
           lenl2 = Length[Result2];
           L2 = {};
           Do[AppendTo[L2,SubstituteInfinitesimals[Result2[[i]]] ],{i,1,lenl2}];
           Result2 = DeleteCases[L2,0];
           Result2 = DeleteCases[Simplify[Result2],0]

      ]
```

```
      ]


(* --- Extract factors from the elements in list expr  --- *)
CancelPreFactor[expr_List]:=Block[
                   {Result1,Lhelp,levels,levels1,fac1,lh},
      Result1 = expr;
      Lhelp = {};
      Do[If[FreeQ[Result1[[i]], Times],
            AppendTo[Lhelp,Result1[[i]] ],

            levels = Level[Result1[[i]],1];
            levels1 = {};
          Do[
          If[FreeQ[levels[[j]],xi[_] ] && FreeQ[levels[[j]],phi[_] ],
             AppendTo[levels1,levels[[j]] ] ],
          {j,1,Length[levels]}];

          fac1 = Apply[Times,levels1];


          AppendTo[Listexpr1,fac1];
          If[fac1 === 0, lh = Result1[[i]],
                         lh = Result1[[i]]/fac1
          ];


(* --- give WARNINGS --- *)
              If[Warnings && Not[NumberQ[fac1]],
                Print[" -- WARNING ! -- "];
                Print[" We eliminated the factor         ",fac1];
                Print[" which was the coefficient of     ",lh ];
                Print["  "]
              ];
          AppendTo[Lhelp,lh]
        ],
      {i,1,Length[Result1]}];
      Lhelp]



(* --- Replacement of higher derivatives if a monomial exists ---*)
DerivativeElimination[der_]:=Block[
      {LE1,LE2,LE3,sum1,sum,icount,level,case1,var1,var2,arg,lenRes2old,
      Result1},
      lenRes2old = 0;
      While[Length[Result2] != lenRes2old,
      lenRes2old = Length[Result2];
      Result1 = {};
      LE2 = {};
      LE3 = {};
      Do[If[FreeQ[Result2[[i]],Plus], AppendTo[Result1,Result2[[i]] ] ],
      {i,1,Length[Result2]}];

      Result1 = CancelPreFactor[Result1];
```

```
      LE1 = Cases[Result1,Derivative[x___][u_[k_]][y___]->{u,k,{y},{x}}];
      Do[   nderi = der;
            sum1 = LE1[[i,4]];
            icount = 0;
            Do[If[sum1[[k]] > 0, icount = icount + 1],{k,1,Length[sum1]}];

        If[icount == 1,

            Do[If[FreeQ[Result2[[k]],Plus], level = {Result2[[k]]},
                  level = Level[Result2[[k]],1]];
                  case1 = Cases[level,
                          mult_.*Derivative[x___][u_[l_]][y___]->
                          {u,l,mult,{y},{x}}];
                Do[If[LTest[ LE1[[i]],case1[[m]] ],
                     var1 = Delete[case1[[m,5]],0];
                     var2 = Delete[case1[[m,4]],0];
                     arg = case1[[m,1]][ case1[[m,2]] ];
                     AppendTo[LE3,Result1[[i]]];
                     AppendTo[LE2,Derivative[var1][arg][var2]->0]],
                {m,1,Length[case1]}],
            {k,1,Length[Result2]}],
          Result2 = Result2 /. Result1[[i]]->0;
          AppendTo[Result2,Result1[[i]]]
        ],
      {i,1,Length[LE1]}];
      LE2 = Union[LE2];
      Result2 = Result2 /. LE2;
      Result2 = AppendTo[Result2,LE3];
      Result2 = Flatten[Result2];
      Result2 = DeleteCases[Result2,0];
      Result1 = {};
      Do[If[FreeQ[Result2[[i]],Plus],
            AppendTo[Result1,Result2[[i]] ]
          ],
      {i,1,Length[Result2]}];
      Result1 = CancelPreFactor[Result1];
      LE2 = Table[0,{i,1,Length[Result1]}];
      LE2 = Thread[SubstRule[Result1,LE2]];
      Result2 = Result2 /. LE2;
      AppendTo[Result2,Result1];
      Result2 = Flatten[Result2];
      Result2 = DeleteCases[Result2,0];
      Result2 = Union[Result2];
      Result2 = Sort[Result2,FreeQ[#1,Plus]&]
      ]
      ]

(* --- Test derivatives if identical  --- *)
LTest[x_List, y_List] := Block[{t1},
  t1 = False;
(* --- Test same variables --- *)
     If[x[[1]] === y[[1]] && x[[2]] === y[[2]],
         Do[If[x[[4,i]] == nderi && x[[4,i]] <= y[[5,i]],
         t1 = True],
```

```
          {i, 1, Length[x[[4]]]}]
        ];
       t1]


(* --- Calculate the determining equations if information   --- *)
(* --- of infinitesimals are given                          --- *)

SubstituteInfinitesimals[expr_]:=Block[
      {ListS,XiS,PhiS,level,simpli,case,var,ExprH},

      DeriList[x_,y_]:={x,y};
(* --- Declare substitutions for dependent variables --- *)
      ListS = Thread[SubstRule[Listu,DependVar]];
      XiS = XiL /. ListS;
      PhiS = PhiL /. ListS;
      Phi = Phi /. ListS;
      Xi = Xi /. ListS;
      ExprH = Expand[expr] /. ListS;
(* --- Determine the cases of ExprH --- *)
      If[FreeQ[ExprH,Plus],level = {ExprH},
                           level = Level[ExprH,1]];
      lenlev = Length[level];
(* --- Find the derivatives in ExprH --- *)
      simpli = 0;
      Do[
         If[FreeQ[level[[i]],Derivative], simpli = simpli + level[[i]],
            case = Cases[{level[[i]]},mlt_.*Derivative[x___][u_[k_]][y___]->
                         {u,k,mlt,{y},{x}}];
           If[Length[case] > 0,
            If[case[[1,1]] === xi,dep = XiS[[case[[1,2]]]] ];
            If[case[[1,1]] === phi,dep = PhiS[[case[[1,2]]]] ];
            var = Thread[DeriList[ case[[1,4]], case[[1,5]] ] ];
            var = Delete[var,0];
(* --- Determine the result from ExprH --- *)
            simpli = simpli + case[[1,3]]*D[dep,var]
           ]
         ],
      {i,1,lenlev}];
(* --- Do substitutions --- *)
      simpli = simpli /. Thread[SubstRule[Xi,XiS]];
      simpli = simpli /. Thread[SubstRule[Phi,PhiS]];
      simpli = simpli /. Thread[SubstRule[DependVar,Listu]];
      simpli = Expand[simpli]
      ]


(* --- Simplification  using Reid's method ---*)
       Simple1[expr_List]:=Block[{Result3,LPHI,LPHI1,lh1},
(* --- list for substituting phi[i] by xi[p+i] as new variables --- *)
      LPHI = {};
      LPHI1 = {};
      lh1 = {};
      Print["Length of input = ",Length[expr]];
      Do[
```

```
          AppendTo[LPHI1, xi[ Length[Xi]+i ] ];
          AppendTo[LPHI,Head[ Phi[[i]] ]],{i,1,Length[Phi]}];

      Result3 = expr /. Thread[SubstRule[Listu,DependVar]];
      Result3 = Result3 /. Thread[SubstRule[LPHI,LPHI1]];
      Result1 = {};
(* --- look for equations containing more than one term --- *)
      Do[
        If[Not[FreeQ[Result3[[i]],Plus]],
           AppendTo[Result1,Result3[[i]] ],
           AppendTo[lh1,Result3[[i]] ]
          ],
      {i,1,Length[Result3]}];
      listvar = {};

(* --- work as long as Result1 is not empty --- *)
      While[Length[Result1] > 0,
      Print["Length of List 1 = ",Length[Result1]];

(* --- find all the derivatives in the equations --- *)
      listderi = {};
      Do[AppendTo[listderi,SearchDeri[ Result1[[i]] ]],
      {i,1,Length[Result1]}];
(* --- Tresse ordering of the derivatives ---- *)
      listderi = Union[Flatten[listderi]];
      listderi = Sort[listderi,TresseOrder[#1,#2]&];
(* --- count the number of a specific derivative --- *)
      lder = {};
      listerg = {};
      Do[
        ider = 0;
        Do[If[ Not[ FreeQ[Result1[[i]],listderi[[j]] ]],ider = ider + 1
             ],
        {i,1,Length[Result1]}];
        AppendTo[lder,ider],
      {j,1,Length[listderi]}];

      Do[ If[ Apply[Plus,lder] > Length[lder],test = 1, test = 0];
         If[lder[[i]] > test,
(* --- find the first occurance of a specific derivative --- *)
             Do[ If[Not[FreeQ[Result1[[j]],listderi[[i]] ]],
                    equation = Result1[[j]];
                    Break[]
                   ],
             {j,1,Length[Result1]}];
(* --- solve for the derivatives --- *)
             erg = Solve[equation==0,listderi[[i]] ];
             erg = Flatten[erg];
             AppendTo[listvar,{listderi[[i]],equation}];
             Result1 = Result1 /. erg;
             Result1 = Simplify[Result1];
             Break[]
            ],
      {i,1,Length[listderi]}];
      Result1 = DeleteCases[Result1,0];
```

```
(* --- listvar contains the results --- *)
      listvar = Sort[listvar,OrdDeri[#1,#2]&];
      ];
      Do[AppendTo[lh1,listvar[[i,2]]],{i,1,Length[listvar]}];
      lh1 = lh1 /. Thread[SubstRule[LPHI1,LPHI]];
      lh1
      ]

(* --- derivative ordering --- *)
OrdDeri[expr1_List,expr2_List]:=Block[{},
      case1 = Cases[{expr1[[1]]},Derivative[x___][u_[k_]][y___]^(n_.) ->
              {u, k, {y}, {x}, n}];
      case2 = Cases[{expr2[[1]]},Derivative[x___][u_[k_]][y___]^(n_.) ->
              {u, k, {y}, {x}, n}];
      If[Apply[Plus,case1[[1,4]]] <
         Apply[Plus,case2[[1,4]]], True, False]
      ]


(* --- Tresse ordering --- *)
TresseOrder[expr1_,expr2_]:=Block[{},
    case1 = Cases[{expr1}, (mult_.)*Derivative[x___][u_[k_]][y___]^(n_.) ->
    {u, mult, n, {y}, {x}, k}];
    case2 = Cases[{expr2}, (mult_.)*Derivative[x___][u_[k_]][y___]^(n_.) ->
    {u, mult, n, {y}, {x}, k}];
    orda = Apply[Plus,case1[[1,5]]];
    ordb = Apply[Plus,case2[[1,5]]];
    If[orda > ordb, t1 = True;
                    Return[t1],
                    t1 = False];
       If[orda == ordb && case1[[1,6]] < case2[[1,6]], t1 = True;
                                                       Return[t1],
                                                       t1 = False ];
         ca1 = case1[[1,5]];
         ca2 = case2[[1,5]];
         Do[test = ca1[[i]]-ca2[[i]];
            If[orda == ordb && case1[[1,6]] == case2[[1,6]] && test >0,
               t1 = True;
               Return[t1],
               t1 = False ],
         {i,1,Length[ca1]}];
      t1
      ]





(* -- search the derivatives --- *)
SearchDeri[expr_]:=Block[{ExprH,level,der,case,var,arg},
      fs1[x_,y_]:={x,y};
      ExprH = Expand[expr];

(* --- test if single term or sum --- *)
```

```
      If[FreeQ[ExprH,Plus],level = {ExprH},
         level = Level[ExprH,1]
      ];
    der = {};
    Do[
    case = Cases[{level[[i]]}, (mult_.)*Derivative[x___][u_[k_]][y___]^(n_.) ->
    {u, mult, n, {y}, {x}, k}];
    If[case != {} ,
    var = Delete[Thread[fs1[case[[1,4]],case[[1,5]] ] ] ,0];
    arg = case[[1,1]][case[[1,6]]][Delete[case[[1,4]],0]];
    AppendTo[der,D[arg,var]^case[[1,3]] ],
    {}],
    {i,1,Length[level]}];
    der
    ]
```

```
(* --- Logging input and output --- *)

OpenLog[filename_String]:=(
      logfile = OpenWrite[filename, FormatType->OutputForm];
      If[logfile === $Failed, Return[] ];
      AppendTo[$Echo, logfile];
      AppendTo[$Output, logfile]
      )
CloseLog[]:=(
      $Echo = Complement[$Echo, {logfile}];
      $Output = Complement[$Output, {logfile}];
      Close[logfile];
      logfile = " "
      )
```

```
(* --- Determine the group order and group constants --- *)
(* --- Arbitrary functions in the group are set to 0 --- *)
groupOrder[indVar_,depVar_,listXi_,listPhi_]:=Block[
      {listfunctions,IndVar,listXI,listPHi,listGroup1},

(* --- Substitute the dependent variables --- *)
      listfunctions ={};
      IndVar = indVar;
      listVar = indVar;

      AppendTo[listVar,depVar];
      listVar = Flatten[listVar];

      Do[
          AppendTo[listfunctions,depVar[[i]][Delete[IndVar,0]]],
          {i,1,Length[depVar]}];
```

```
      substitute = Thread[SubstRule[listfunctions,depVar]];
      listXI = listXi /. substitute;
      listPHi = listPhi /. substitute;


(* --- determin the group constants --- *)
      listGroup = Variables[listXI];
      listGroup1 = Variables[listPHi];
      listGroup = Complement[Union[listGroup,listGroup1],ParameterS];
      Do[
      listGroup = DeleteCases[listGroup,indVar[[i]] ],
      {i,1,Length[indVar]}];
      Do[
      listGroup = DeleteCases[listGroup,depVar[[i]] ],
      {i,1,Length[depVar]}];
      listGroup = {Length[listGroup],listGroup};
      listGroup1 = {};
(* --- separate the group constants from arbitrary functions --- *)
      Do[
         Do[
            If[Not[FreeQ[listGroup[[2,i]],listVar[[j]] ]],
               AppendTo[listGroup1,listGroup[[2,i]] ]
              ],
         {j,1,Length[listVar]}],
      {i,1,listGroup[[1]]}];
      listGroup1 = Union[listGroup1];
      listGroup[[2]] = Complement[listGroup[[2]],listGroup1];
      listGroup[[1]] = Length[listGroup[[2]] ];
(* --- arbitrary functions are set to 0 --- *)
            listG1 = Table[0,{i,1,Length[listGroup1]}];
            rule = Thread[SubstRule[listGroup1,listG1]];
      AppendTo[listGroup,rule]
      ]


(* --- Create the vector field of the symmetry --- *)
Df[xin_,indVar_,depVar_,listXi_,listPhi_,number_]:=Block[
  {operator,IndVar,substitute,listXI,listPHi,xiN,listGroup,
  listGroup1,listG1,rule},

(* --- Substitute the dependent variables --- *)
      listfunctions ={};
      IndVar = indVar;

      Do[
          AppendTo[listfunctions,depVar[[i]][Delete[IndVar,0]]],
          {i,1,Length[depVar]}];

      substitute = Thread[SubstRule[listfunctions,depVar]];
      listXI = listXi /. substitute;
      listPHi = listPhi /. substitute;
      xiN = xin /. substitute;

(* --- determin the group constants --- *)
```

```
      listGroup = groupOrder[IndepVar,DependVar,listXi,listPhi];
      rule = listGroup[[3]];
      listGroup = listGroup[[2]];

(* --- Check the number of group-parameters --- *)
      If[number <= Length[listGroup],

(* --- Setup the substitutions for the group constants --- *)
      Do[
      listGroup = DeleteCases[listGroup,depVar[[i]] ],
      {i,1,Length[depVar]}];
      listGroup1 = DeleteCases[listGroup,listGroup[[number]]];
      AppendTo[rule,listGroup[[number]]->1];
      listG1 = Table[0,{i,1,Length[listGroup1]}];
      AppendTo[rule,Thread[SubstRule[listGroup1,listG1]]];
      rule = Flatten[rule];

(* --- Calculate the vectorfield --- *)
      operator = Sum[listXI[[i]]*D[xiN,indVar[[i]] ],
                {i,1,Length[indVar]}];

      operator = operator +
                 Sum[listPHi[[i]]*D[xiN,depVar[[i]] ],
                 {i,1,Length[depVar]}];
      operator = operator /. rule;
      operator = Expand[operator],
(* --- Print a warning --- *)
      Print[" #### Error: number of group-parameters ",Length[listGroup],"
      is less than ",number]
      ]

      ]


(* --- Create the vector basis of the algebra --- *)
VectorBasis[indVar_,depVar_,listXi_,listPhi_,order_]:=Block[
     {listBasis,listBAsis,IndVar,rule,rulePlus,ruleTimes},
(* --- Create the basis of the algebra --- *)
      listBasis = {};
      listBAsis = {};

      IndVar = indVar;
      AppendTo[IndVar,depVar];
      IndVar = Flatten[IndVar];
      IndVar = Delete[IndVar,0];

      Do[AppendTo[listBasis,
                  Df[Global`Funcf[IndVar],indVar,depVar,listXi,listPhi,k]
                 ];
         AppendTo[listBAsis,Global`V[k]],
      {k,1,order}];
      rule = Thread[SubstRule[listBasis,listBAsis] ];
      rule = Reverse[rule];
      rulePlus = {};
      ruleTimes = {};
```

```
      Do[
         If[FreeQ[rule[[k]],Plus],
            AppendTo[ruleTimes,rule[[k]] ],
            AppendTo[rulePlus,rule[[k]] ]
            ],
      {k,1,Length[rule]}];
      rule = {rulePlus};
      AppendTo[rule,ruleTimes]
      ]

(* --- Calculate the commutator of the Lie algebra --- *)
Commutator[i_,j_,rule_,order_]:=Block[
      {IndVar,komm,rulePlus,ruleTimes,Komm,
       levelKomm,level1,level2,listVar},

      IndVar = IndepVar;
      AppendTo[IndVar,DependVar];
      IndVar = Flatten[IndVar];
      IndVar = Delete[IndVar,0];

(* --- substitution rules --- *)
      rulePlus = rule[[1]];
      ruleTimes = rule[[2]];

(* --- Calculate the commutator --- *)
      komm = Expand[ Df[
                  Df[Global`Funcf[IndVar],IndepVar,DependVar,ListXi,ListPhi,i],
                  IndepVar,DependVar,ListXi,ListPhi,j] -
                  Df[
                  Df[Global`Funcf[IndVar],IndepVar,DependVar,ListXi,ListPhi,j],
                  IndepVar,DependVar,ListXi,ListPhi,i]];
(* --- Give a representation in the basis of the algebra --- *)
      rulePlus = rulePlus /. ruleTimes;
      komm = komm /. ruleTimes;
      komm = -Factor[-komm] /. rulePlus;
      komm = Factor[komm]  /. rulePlus;
      komm = -(-komm /. rulePlus);
(* --- more sophisticated matching --- *)
      If[FreeQ[komm,Plus], komm = komm,
            levelKomm = Level[Expand[komm],1];
            level1 = {};
            level2 = {};
            listVar = IndepVar;
            AppendTo[listVar,DependVar];
            listVar = Flatten[listVar];
            Do[
               Do[
               If[ Not[FreeQ[levelKomm[[l]], listVar[[k]] ]],
                  AppendTo[level2,levelKomm[[l]] ]
                  ],
                {k,1,Length[listVar]}],
            {l,1,Length[levelKomm]}];
            level1 = Complement[levelKomm,level2];
            Komm = Apply[Plus,level2];
            Komm = -Factor[-Komm] /. rulePlus;
```

```
            Komm = Factor[Komm] /. rulePlus;
            AppendTo[level1,Komm];
            Apply[Plus,level1]
      ]
      ]


(* --- Create a commutator table --- *)
CommutatorTable[]:=Block[{listGroup,order,tableh,rule,lenVbasis,
      Classes,class,table1,table2,lenClass,Solvable},
      Classes = {};
      Print["  "];
      Print[" Calculation of the commutator table."];
(* --- determin the order of the algebra --- *)
      listGroup = groupOrder[IndepVar,DependVar,ListXi,ListPhi];
      order = listGroup[[1]];
(* --- create the basis of the algebra --- *)
      VBasis = VectorBasis[IndepVar,DependVar,ListXi,ListPhi,order];
(* --- elements of the complete basis --- *)
      Classes = {Table[Global`V[i],{i,1,order}]};
(* --- calculate the commutator table --- *)
      tableh = Table[Commutator[i,j,VBasis,order],{i,1,order},{j,1,order}];
      class = Complement[Variables[Union[Flatten[tableh]]],ParameterS];
(* --- assume first a solvable algebra --- *)
      lenClass = Length[class];
      If[ lenClass == order,
            Solvable = False;
            AppendTo[Classes,{"nonsolvable algebra"}],
            Solvable = True;
            class = Complement[Variables[tableh],ParameterS];
            lenClass = Length[class];
            AppendTo[Classes,class]
      ];
(* --- find the subalgebras of the discret algebra --- *)
      If[Solvable,
            table1 = tableh;
            While[lenClass > 0,
                  index = Flatten[Cases[Flatten[table1],m_*Global`V[k_]->{k}]];
                  index = Union[index];
               If[ Length[index] > 0,
                  table2 = Table[0,{i,1,Length[index]},{j,1,Length[index]}];
                  Do[
                     Do[
                        table2[[i,j]] = tableh[[index[[i]],index[[j]]]],
                     {i,1,Length[index]}],
                  {j,1,Length[index]}];
               class = Complement[Variables[table2],ParameterS];
               lenClass = Length[class];
                  If[lenClass === 0,AppendTo[Classes,{"solvable algebra"}],
                     AppendTo[Classes,class]
                  ];
               table1 = table2,
               lenClass = 0
               ]
            ];
```

```
      ];

(* --- determine the Casimier operators --- *)

      Casimier = {};
      Do[
      tabsum = 1;
      tabsum = Apply[Plus,tableh[[i]] ];
      If[tabsum === 0, AppendTo[Casimier,Global`V[i] ] ],
      {i,1,order}];

(* --- append the results to FinalResult --- *)
      AppendTo[FinalResult,Casimier];
      AppendTo[FinalResult,Classes];
      VBasis = Flatten[VBasis];
      AppendTo[FinalResult,VBasis];

(* --- output the information on screen --- *)
      Print["  "];
      Print["  Basis of the Lie algebra."];
      Print["  --------------------------------------------------------------------"];
      Print[" "];
      Do[Print["  ",VBasis[[k]] ];
         Print["                        "],
      {k,1,Length[VBasis]}];
      Print["  --------------------------------------------------------------------"];
      Print["  "];
      Print["  Structure of the Lie algebra."];
      Print["  --------------------------------------------------------------------"];
      Print[" "];
      Do[Print["  Ideal ",k-1," :  ",Classes[[k]] ];
         Print["                        "],
      {k,1,Length[Classes]}];
      Print["  --------------------------------------------------------------------"];
      Print["  "];
      Print["  Casimier(s) of the Lie algebra."];
      Print["  "];
      Print["  ",Casimier];
      Print["  "];
      Print["  --------------------------------------------------------------------"];
      Print[" "];
      CommTable = tableh
      ]


(* ---Determine the structure constants of the algebra --- *)
StructureConstants[]:=Block[{listGroup,order,structure,comm},
(* --- determin the order of the algebra --- *)
      listGroup = groupOrder[IndepVar,DependVar,ListXi,ListPhi];
      order = listGroup[[1]];
(* --- inizialize the structure constants by 0 --- *)
      structure = Table[0,{i,1,order},{j,1,order},{k,1,order}];
(* --- calculate the commutatot table --- *)
      If[CommTable == {},comm = CommutatorTable[],comm = CommTable];
(* --- find the structure constants --- *)
```

```
      Print["  "];
      Print[" Nonzero structure Constants."];
      Print["  ----------------------------------------------------------------------"];
      Print[" "];
      Do[
         Do[
            Do[
                 structure[[i,j,k]] = Coefficient[comm[[i,j]],Global`V[k]];
                 If[structure[[i,j,k]] != 0,
                 Print["   C[",i,",",j,",",k,"] = ",structure[[i,j,k]]]
                 ],
               {k,1,order}],
         {j,1,order}],
      {i,1,order}];
      Print[" "];
      Print["  ----------------------------------------------------------------------"];
      structure
      ]

(* --- Calculation of the metric tensor from structure constants --- *)
MetricStructure[inlist_List] :=
  Block[{lieStructure,leninlist,h1,g},
(* --- length of the list --- *)
          lieStructure = inlist;
          leninlist = Length[inlist];
(* --- initialize the metric matrix g --- *)
          g = Table[0,{i,1,leninlist},{i,1,leninlist}];
          h1 = 0;
(* --- do the summation over m and k C_im^k*C_jk^m --- *)
          Do[
            Do[
              Do[
                Do[
                  h1 = h1 + lieStructure[[i,m,k]]*lieStructure[[j,k,m]],
                {m, 1, leninlist}],
              {k, 1, leninlist}];
            g[[i,j]] = h1;
            h1 = 0,
            {i, 1, leninlist}],
          {j, 1, leninlist}];
          g]




(* --- Output for LaTex text system --- *)
LateX[filename_,finalResult_]:=Block[

      {stream,nameHelp,text,texth,texth2,fResult},
(* --- open the straem file --- *)
      stream = OpenWrite[filename];

      nameHelp = StringTake[filename,-4];
      text = StringReplace[filename,nameHelp->".dgl}"];
```

```
(* --- write the header of the file --- *)
      Write[stream,TeXForm["\\documentstyle[12pt]{article}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm["\\begin{document} "]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm["\\begin{center}"]];
      Write[stream,TeXForm["Results produced by
      {\\tt Lie[]} \\copyright"]];
      Write[stream,TeXForm[" \\footnote{
      G. Baumann, University of Ulm,"]];
      Write[stream,TeXForm["Albert-Einstein-Allee 11,
      D-7900 Ulm, Germany}"]];
      Write[stream,TeXForm["\\end{center}"]];
      Write[stream,TeXForm["\\vspace{0.75cm}"]];
      Write[stream,TeXForm[" "]];

(* --- write the equations of motion --- *)
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Equations of motion : {\\bf "],TeXForm[text]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Do[
      text = StringJoin[" 0 "," &=& "];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[finalResult[[1,i]] ]];
      If[i < Length[finalResult[[1]]], Write[stream,TeXForm["\\\\[2mm]"]]],
      {i,1,Length[finalResult[[1]]]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]];

(* --- write the substitutions --- *)
      fResult = Flatten[finalResult[[2]] ];
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Substitutions in the equations of motion : "]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Do[
      text = StringJoin[ToString[i],"   &:& "];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[fResult[[i]] ]];
      If[i < Length[fResult], Write[stream,TeXForm["\\\\[2mm]"]]],
      {i,1,Length[fResult]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]];

(* --- write the infinitesimals Xi and Phi --- *)
      If[Length[finalResult[[3]]] > 0,
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Infinitesimals for independent and dependent
      variables"]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
```

```
      Do[
      If[i > Length[IndepVar],
      text = StringJoin["\phi_",ToString[i-Length[IndepVar]]," &=& "],
      text = StringJoin["\xi_",ToString[i]," &=& "]
      ];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[finalResult[[3,i]] ]];
      If[i < Length[finalResult[[3]]], Write[stream,TeXForm["\\\\[2mm]"]]],
      {i,1,Length[finalResult[[3]]]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]]
      ];

(* --- write the determining equations --- *)
      If[Length[finalResult[[4]]] > 0,
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Determining equations "]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Do[
      text = StringJoin[ToString[i]," : 0 &=& "];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[finalResult[[4,i]] ]];
      If[i < Length[listEquation], Write[stream,TeXForm["\\\\[2mm]"]]],
      {i,1,Length[finalResult[[4]]]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]]
      ];

(* --- write the Casimiers --- *)
      If[Length[finalResult[[5]]] > 0,
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Casimier(s) of the vector fields "]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Do[
      text = StringJoin["{\cal C}_",ToString[i]," &=& "];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[finalResult[[5,i]] ]];
      If[i < Length[finalResult[[5]]], Write[stream,TeXForm["\\\\[2mm]"]]],
      {i,1,Length[finalResult[[5]]]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]]
      ];

(* --- write the Ideals --- *)
      If[Length[finalResult[[6]]] > 0,
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Basis and Ideals of the Lie-algebra "]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Do[
```

```
      text = StringJoin["{I}_",ToString[i-1]," &=& "];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[finalResult[[6,i]] ]];
      If[i < Length[finalResult[[6]]], Write[stream,TeXForm["\\\\[2mm]"]]],
      {i,1,Length[finalResult[[6]]]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]]
      ];

(* --- write the vector basis of the algebra --- *)
      If[Length[finalResult[[7]]] > 0,
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Basis of the Lie-algebra "]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Do[
      text = StringJoin[ToString[i],"   &:& "];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[finalResult[[7,i]] ]];
      If[i < Length[finalResult[[7]]], Write[stream,TeXForm["\\\\[2mm]"]]],
      {i,1,Length[finalResult[[7]]]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]]
      ];

(* --- write the commutator table --- *)
      If[Length[finalResult[[8]]] > 0,
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Commutator table"]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{center}"]];
      text = "\begin{tabular}{c";
      texth = "$[,]$";
      texth2 = "\hline ";
      Do[text = StringJoin[text,"|c"];
         texth2 = StringJoin[texth2," & "];
         texth = StringJoin[texth," & $V(",ToString[i],")$"],
      {i,1,Length[finalResult[[8]]]}];
      text = StringJoin[text,"}"];
      texth = StringJoin[texth,"\\\\","[2mm] ","\hline"];
      texth2 = StringJoin[texth2,"\\\\","[2mm]"];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[texth]];
      Write[stream,TeXForm[texth2]];
      Do[
         texth = StringJoin["V(",ToString[i],")"];
         Do[
         texth = StringJoin[texth," & $",
                          ToString[TeXForm[finalResult[[8,i,j]] ]],
                          "$"],
         {j,1,Length[finalResult[[8]]]}
         ];
         Write[stream,TeXForm[texth] ];
```

```
      Write[stream,TeXForm["\\\\[2mm]"]];
      Write[stream,TeXForm["\hline"]];
      Write[stream,TeXForm[texth2]],
      {i,1,Length[finalResult[[8]]]}
      ];
      Write[stream,TeXForm["\end{tabular}"]];
      Write[stream,TeXForm["\end{center}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]]
      ];

(* --- write the nonzero structure constants --- *)
      If[Length[finalResult[[9]]] > 0,
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Structure constants"]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Do[
        Do[
           Do[
      If[finalResult[[9,i,j,k]] != 0,
      text = StringJoin["C_{",ToString[i],",",ToString[j],
                        ",",ToString[k],"}"," &=& "];
      Write[stream,TeXForm[text]];
      Write[stream,TeXForm[finalResult[[9,i,j,k]] ]];
      Write[stream,TeXForm["\\\\[2mm]"]]
      ],
           {k,1,Length[finalResult[[9]]]}],
        {j,1,Length[finalResult[[9]]]}],
      {i,1,Length[finalResult[[9]]]}];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]];

(* --- write the metric table g_ij --- *)
      Write[stream,TeXForm["\\noindent "]];
      Write[stream,TeXForm["Metric tensor $g_{ij} $"]];
      Write[stream,TeXForm["% "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Write[stream,TeXForm["g_{ij} = \left ( "]];
      text = "\begin{array}{";
      Do[text = StringJoin[text,"c"],
      {i,1,Length[finalResult[[8]]]}];
      text = StringJoin[text,"}"];
      Write[stream,TeXForm[text]];
      Do[
         texth = " ";
         Do[
         If[j < Length[finalResult[[8]]],
         texth = StringJoin[texth,ToString[TeXForm[finalResult[[10,i,j]]]]," & " ],
         texth = StringJoin[texth,ToString[TeXForm[finalResult[[10,i,j]]]] ]
         ],
         {j,1,Length[finalResult[[8]]]}
         ];
         Write[stream,TeXForm[texth] ];
```

```
      Write[stream,TeXForm["\\\\[1mm]"]],
      {i,1,Length[finalResult[[8]]]}
      ];
      Write[stream,TeXForm["\end{array}"]];
      Write[stream,TeXForm["\\right ) "]];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm["\begin{eqnarray}"]];
      Write[stream,TeXForm[StringJoin["\\mbox{det}(g_{ij}) = ",
                                ToString[Det[finalResult[[10]]]]] ]];
      Write[stream,TeXForm["\end{eqnarray}"]];
      Write[stream,TeXForm[" "]];

      ];

      Write[stream,TeXForm[" "]];
      Write[stream,TeXForm["\\end{document} "]];

      Close[stream];
      ]


(* --- make an ansatz for the infinitesimals of polynomial form --- *)
InfinitesimalsAnsatz[Param1_,Result2_]:=Block[
      {Vars,AnsatzInfi,exponentS,coeffiC,AnsatzTerms,EquationsAnsatz,
      ResultSubst,ResultAnsatz,ErgabSubst,SingleFactor,SolutionRules},

      Print[" "];
      Print["  Create an ansatz for the infinitesimals of ",aNsatz[[1]],". order."];
      Print["  -------------------------------------------------------------------"];
      Print[" "];
(* --- create the ansatz in independent and dependent variables --- *)
      AnsatzInfi = AnsatzInfinitesimals[IndepVar,DependVar,aNsatz[[1]]];
(* --- save the ansatz in the lists for the infinitesimals --- *)
      XiL = Table[0,{i,1,Length[IndepVar]}];
      PhiL = Table[0,{i,1,Length[DependVar]}];
      Do[
      If[i <= Length[IndepVar],
      XiL[[i]] = AnsatzInfi[[1,i]],
      PhiL[[i-Length[IndepVar]]] = AnsatzInfi[[1,i]]
      ],
      {i,1,Length[AnsatzInfi[[1]] ]}];
(* --- substitute the ansatz into the determining equations --- *)
      ResultSubst = {};
      Do[AppendTo[ResultSubst,SubstituteInfinitesimals[Result2[[i]]] ],
       {i,1,Length[Result2]}];
      ResultSubst = DeleteCases[Simplify[ResultSubst],0];

(* --- separate the equations of the ansatz --- *)
      Vars = IndepVar;
(* --- creat a list of variables --- *)
      AppendTo[Vars,Table[DependVar[[i]][Delete[IndepVar,0]],
                   {i,1,Length[DependVar]}] ];
      Vars = Flatten[Vars];
```

```
(* --- find the exponents and the terms depending on independent and
       dependent variables --- *)
      exponentS = {};
      Do[
         AppendTo[exponentS,Exponent[ResultSubst,Vars[[i]],List]],
      {i,1,Length[Vars]}];
      AnsatzTerms = {};
      Do[
      AppendTo[
      AnsatzTerms,
        Table[Vars[[i]]^exponentS[[i,k,l]],{k,1,Length[exponentS[[i]]]},
                                         {l,1,Length[exponentS[[i,k]]]}]
            ],
      {i,1,Length[Vars]}];
      AnsatzTerms = Union[Flatten[Apply[Times,AnsatzTerms]]];
      AnsatzTerms = Reverse[Sort[AnsatzTerms]];
(* --- extract the equations for the constants --- *)
      EquationsAnsatz = {};
      Do[
         Do[
         coeffiC = Coefficient[ResultSubst[[j]],AnsatzTerms[[i]] ];
         AppendTo[EquationsAnsatz,{coeffiC}];
         ResultSubst[[j]] = Expand[ResultSubst[[j]] -
                                   coeffiC*AnsatzTerms[[i]] ],
         {i,1,Length[AnsatzTerms]}];
      coeffiC = ResultSubst[[j]];
      AppendTo[EquationsAnsatz,{coeffiC}],
      {j,1,Length[ResultSubst]}];
(* --- solve for the constants --- *)
      EquationsAnsatz = Flatten[EquationsAnsatz];
      EquationsAnsatz = DeleteCases[EquationsAnsatz,0];
      EquationsAnsatz = Union[EquationsAnsatz];
      SolutionRules = {};
      SingleFactor = {};
      Print["  elimination of single terms"];
      While[SingleTermQ[EquationsAnsatz],
(* --- separate single term equations from equations with several terms --- *)
        singleTerm ={};
        multiTerm ={};
        Do[
           If[FreeQ[EquationsAnsatz[[i]],Plus],
              AppendTo[singleTerm,EquationsAnsatz[[i]] ],
              AppendTo[multiTerm,EquationsAnsatz[[i]] ]
           ],
        {i,1,Length[EquationsAnsatz]}];
(* --- eliminate numeric factors in single term equations --- *)
        singleFactor = {};
        multiFactor ={};
        Do[
            If[FreeQ[singleTerm[[i]],Times],
               AppendTo[singleFactor,singleTerm[[i]] ],
               level = Level[singleTerm[[i]],1];
                  Do[
                     If[NumberQ[level[[j]]] || ParameterQ[level[[j]]],
                     singleTerm[[i]] = Expand[singleTerm[[i]]/level[[j]] ]
```

```
                  ],
               {j,1,Length[level]}];
           AppendTo[singleFactor,singleTerm[[i]] ]
           ],
     {i,1,Length[singleTerm]}];
   singleFactor = Union[singleFactor];
   AppendTo[SingleFactor,singleFactor];
(* --- create the substitutions for single terms --- *)
   singleZero = Table[0,{i,1,Length[singleFactor]}];
   solutionRules = Thread[SubstRule[singleFactor,singleZero]];
   AppendTo[SolutionRules,solutionRules];
(* --- simplify the equations --- *)
   EquationsAnsatz = EquationsAnsatz /. solutionRules;
   EquationsAnsatz = DeleteCases[EquationsAnsatz,0]
   ];
(* --- simplyfy the equations by elimination of common factors --- *)
   Print["  elimination of common factors "];
   EquationsAnsatz = EliminFactors[EquationsAnsatz,ParameterS];

   If[Length[EquationsAnsatz] >= 1,
      substEquation[x_,y_]:=x==y;
      EquationZero = Table[0,{i,1,Length[EquationsAnsatz]}];
      EquationsAnsatz = Thread[substEquation[EquationsAnsatz,EquationZero]];
      AnsatzVariables = Complement[AnsatzInfi[[2]],Flatten[SingleFactor]];

(* --- solve the reduced system of equations --- *)
      Print["  Calling Solve with ",Length[EquationsAnsatz]," equations "];
      Print["  The number of unknowns is ",Length[AnsatzVariables]];
      Print[" "];

      ResultAnsatz = Solve[EquationsAnsatz , AnsatzVariables ],
      ResultAnsatz = {}
   ];

   AppendTo[ResultAnsatz,SolutionRules];
   ResultAnsatz = Flatten[ResultAnsatz];
(* --- substitute the constants back into the ansatz --- *)
   ErgabSubst = AnsatzInfi[[1]] /. ResultAnsatz;
(* --- save the results in the input file --- *)
   ErgabSubst =Flatten[ErgabSubst];
   ListXi = Table[0,{i,1,Length[IndepVar]}];
   ListPhi = Table[0,{i,1,Length[DependVar]}];
   Do[
   If[i <= Length[IndepVar],
   ListXi[[i]] = Simplify[ErgabSubst[[i]]],
   ListPhi[[i-Length[IndepVar]]] = Simplify[ErgabSubst[[i]]]
   ],
   {i,1,Length[ErgabSubst ]}];
   Save[Param1,ListXi,ListPhi];
   Print["  The calculated infinitesimals are:"];
   Print[" "];
   Print["  -------------------------------------------------------------------"];
   Print[" "];
   Do[
      Print[" Xi[",i,"] = ",ListXi[[i]] ];
```

```
        Print[" "],
      {i,1,Length[ListXi]}];
      Do[
        Print[" Phi[",i,"] = ",ListPhi[[i]] ];
        Print[" "],
      {i,1,Length[ListPhi]}];
      Print[" "];
      Print["  ------------------------------------------------------------------"];
      Print[" "];
      ];

(* --- check input as an element in the list ParameterS --- *)
ParameterQ[input_]:=Block[{},
      If[Length[Complement[ParameterS,{input}]] < Length[ParameterS],
        True,
        False
      ]
      ]


(* --- elimination of common factors in equations --- *)
EliminFactors[equations_List,parametrs_List]:=Block[{equation,summs,level},
(* --- factorice the equations --- *)
      equation = Factor[equations];
      summs = {};
      Do[
(* --- seperate sums and products --- *)
        If[FreeQ[equation[[i]],Times,1],
          AppendTo[summs,equation[[i]]],
          level = Level[equation[[i]],1];
(* --- cancel common factors --- *)
                Do[
                  If[NumberQ[level[[j]]] || ParameterQ[level[[j]]],
                  equation[[i]] = Expand[equation[[i]]/level[[j]] ]
                  ],
                {j,1,Length[level]}];
          AppendTo[summs,equation[[i]]]
         ],
       {i,1,Length[equation]}];
(* --- unify the equations --- *)
       Union[summs]
       ]

(* --- test on single terms in the equations --- *)
SingleTermQ[equations_List]:=Block[{equation},
      equation = equations;
      meassage = False;
      Do[
        If[FreeQ[equation[[i]],Plus],
          meassage = True
        ],
      {i,1,Length[equation]}];
      meassage
      ]
```

```
(* --- Ansatz for the infinitesimals --- *)
AnsatzInfinitesimals[indepVar_List,dependentVar_List,orderDepend_]:=
      Block[{indVar,variab,listconst,erg,listpoly,
      lVariables,lResult,poly,ph,termS},

(* --- initialize the variables --- *)
      listconst = {};
      indVar = indepVar;
      variab = dependentVar;
      erg = 0;
      lVariables = {};
(* --- create the list of polynomila variables --- *)
         Do[
            AppendTo[lVariables,variab[[iv]][Delete[indVar,0]] ],
         {iv,1,Length[variab]}];
      AppendTo[lVariables,indVar];
      lVariables = Flatten[lVariables];
      listpoly = {};
(* --- create the polynomial --- *)
      Do[
         poly = 0;
         lResult = Table[1,{i,1,Length[lVariables]}];
         termS = {};
         Do[
            lResult = Union[Flatten[Outer[Times,lResult,lVariables]]];
            AppendTo[termS,lResult];
            lconstants = {};
            Do[AppendTo[lconstants,Unique["k"]],{j,1,Length[lResult]}];
            AppendTo[listconst,lconstants];
            poly = poly + lconstants.lResult,
         {i,1,orderDepend}];
        ph = Unique["k"];
        AppendTo[listconst,ph];
        poly = poly + ph;
        AppendTo[listpoly,poly],
      {k,1,Length[lVariables]}];
      listpoly = Partition[listpoly,Length[lVariables]];
      termS = Reverse[Union[Flatten[termS]]];
      erg = Flatten[listpoly];
      {erg,Flatten[listconst],termS}
 ]

(* --- calculate the adjungate algebra --- *)
AdAlgebra[order_]:=Block[{},
                  Table[Adjungation[Global`V[i],Global`V[j],order],
                  {i,1,Length[LieTable]},{j,1,Length[LieTable]}]
                  ]

(* --- adjungation of the algebra --- *)
Adjungation[v1_, v2_, order_] :=
  Block[{listTerms,listCoeff,listHelp, map1},
    listTerms = {v1};
    listCoeff = {1};
```

```
    map1 = Flatten[MapThread[KommutFunction, {{v1}, {v2}}]];
    AppendTo[listTerms, map1];
    AppendTo[listCoeff, -Global`eps];
    Do[map1 = Flatten[MapThread[KommutFunction, {{v1}, map1}]];
       AppendTo[listCoeff,(-Global`eps)^(i+1)/(i+1)!];
       AppendTo[listTerms, map1],
    {i, 1, order}];
  listHelp = Flatten[listTerms*listCoeff];
  Apply[Plus,listHelp]
  ]


(* --- commutation of two vectorfields --- *)
KommutFunction[a_,b_]:=Block[{v1,v2,level,tableNew},
       If[NumberQ[a],Return[a],
           If[ FreeQ[a,Plus], level = {a},
               level = Level[a,1]
           ];
       v1 = Flatten[Cases[level,m_.*Global`V[i_]->{i}]]
       ];
       If[NumberQ[b],Return[b],
           If[ FreeQ[b,Plus], level = {b},
               level = Level[b,1]
           ];
       v2 = Flatten[Cases[level,m_.*Global`V[j_]->{j}]]
       ];
       tableNew = {};
       Do[
         Do[AppendTo[tableNew,
                     LieTable[[ v1[[i]],v2[[j]] ]]
                    ],
         {i,1,Length[v1]}],
       {j,1,Length[v2]}];
       Apply[Plus,tableNew]
       ]



Print["     Program loaded. Use  Lie[filename,Options]  to run the program."];
End[]
EndPackage[]
```

model	ξ_1	ξ_2	ϕ_1
KdV	$c_4 + c_3 t + \frac{c_1}{3} x$	$c_2 + c_1 t$	$c_3 - \frac{2c_1}{3} u$
mKdV	$c_3 + \frac{c_1}{3} x$	$c_2 + c_1 t$	$-\frac{c_1}{3} u$
cKdV	$c_2 + c_1 x + 2c_3\sqrt{t} + 2c_4 x\sqrt{t}$	$3c_1 t + 4c_4 t\sqrt{t}$	$-2c_1 u + c_3/\sqrt{t} + c_4(x/\sqrt{t} - 4\sqrt{t}u)$
λ-KdV	$c_1 - c_3(12\lambda t + x)$	$c_2 - 3c_3 t$	$c_3 u$

TABLE I
Infinitesimals of the KdV-family.

model	386Pc	486Pc	HP-9000/700	NeXT	VAX6440
KdV	45	21	21	12	72
mKdV	42	21	21	12	88
cKdV	64	31	30	14	95
λ-KdV	49	24	25	14	86

TABLE II
Time measured in seconds needed to derive the determining equations for the infinitesimals of the KdV-family. On a NeXT computer we get times by a factor of about 5 smaller than on a VAX 6440 using MACSYMA.