Article

ReLie: A Reduce Program for Lie Group Analysis of Differential Equations

Francesco Oliveri

Department of Mathematical and Computer Sciences, Physical Sciences and Earth Sciences, University of Messina, Viale F. Stagno d’Alcontres 31, 98166 Messina, Italy; francesco.oliveri@unime.it

Abstract: Lie symmetry analysis provides a general theoretical framework for investigating ordinary and partial differential equations. The theory is completely algorithmic even if it usually involves lengthy computations. For this reason, along the years many computer algebra packages have been developed to automate the computation. In this paper, we describe the program ReLie, written in the Computer Algebra System Reduce, since 2008 an open source program for all platforms. ReLie is able to perform almost automatically the needed computations for Lie symmetry analysis of differential equations. Its source code is freely available too. The use of the program is illustrated by means of some examples; nevertheless, it is to be underlined that it proves effective also for more complex computations where one has to deal with very large expressions.

Keywords: Lie symmetries; differential equations; symbolic computation; Reduce Computer Algebra System

MSC: 34-04; 34A05; 35-04; 58J70; 58J72

1. Introduction

A general and powerful theoretical approach to deal with ordinary as well as partial differential equations is provided by the study of their continuous symmetries, i.e., transformations mapping the set of solutions of differential equations into itself. Symmetry analysis of differential equations originated in the nineteenth century with Sophus Lie [1,2], who realized that many special integration theories of differential equations are a consequence of the invariance under one-parameter continuous groups of transformations providing a diffeomorphism on the space where independent and dependent variables live. These mappings in turn determine a transformation of the derivatives such that the contact conditions are preserved.

Many textbooks and monographs, either elementary or advanced, some of them also focused on relevant applications, dealing with Lie group methods are available [3–18]. Lie groups of continuous transformations, besides their intrinsic theoretical interest, have many important applications such as

- implementation of algorithmic procedures for lowering the order of ordinary differential equations, or integrating them by quadrature [13,15,19];
- characterization of invariant solutions of initial and boundary value problems [5,8,9,20–27];
- derivation of conservation laws [28–30];
- algorithmic construction of invertible point transformations mapping the differential equations into equivalent forms easier to handle [17,31–39];
- design of efficient numerical schemes [40–42].

Furthermore, Lie’s classical theory represented a source for various generalizations. Among these generalizations there are the nonclassical symmetries first proposed by Bluman and Cole [43], and now part of the more general method of differential constraints [44,45], the potential symmetries [46], the nonlocal symmetries [47–49], the gen-
eralized symmetries [5], which in turn generalize contact symmetries introduced by Lie himself, the equivalence transformations [3,50–56], to quote a few. A further extension is represented by approximate symmetries [57–61] for differential equations containing small terms, often arising in concrete applied problems (see, for instance, [62] for an application of approximate Lie symmetries to Navier–Stokes equations).

The generality and effectiveness of Lie’s theory, and of its modern generalizations, determined in the last decades a rapid extension of its range of application. Besides the classical applications to problems arising in mathematical physics, in recent years, these techniques proved successful also in the investigation of socio–economic and financial models (see, for instance, [63–65], and references therein), as well as in biomathematics (see [66–68], and references therein), and in mathematical epidemiology (see [69–72], and references therein), to quote a few. In such contexts, a computer algebra package able to provide the symmetries admitted by mathematical models, and consequently help to characterize meaningful solutions, can prove effective.

The determination of Lie symmetries admitted by differential equations is algorithmic; nevertheless, it usually involves a lot of cumbersome and tedious calculations. The current availability of many powerful Computer Algebra Systems (CAS) (either commercial or open source) greatly helped the application of Lie group methods, so that most of the needed algebraic manipulations can now be done quickly and often automatically. In fact, many specific packages for performing symmetry analysis of differential equations are currently available in the literature [10,12,73–86].

In this paper, the package ReLie, written in the computer algebra system Reduce [87], is presented. The routines contained in ReLie allow the user to easily compute Lie point symmetries and conditional symmetries, as well as contact transformations, variational symmetries, and equivalence transformations for classes of differential equations containing arbitrary elements; ReLie is also able to compute approximate Lie symmetries according to the approach proposed in [60].

Reduce is a general purpose computer algebra system whose development was started by Anthony Hearn, and is written in a Lisp dialect (Portable Standard Lisp). Since December 2008, it is an open source program available for all operating systems at the url http://www.reduce-algebra.com (accessed on 9 August 2021), and continuously updated and supported. The source code of ReLie, as well as the user’s manual, can be found at the url http://mat521.unime.it/oliveri (accessed on 9 August 2021). Thus, ReLie can be freely used by researchers who do not have licenses of commercial Computer Algebra Systems like Wolfram Mathematica™ or Maple™.

The origin of this package dates back to 1994 when the author developed some routines useful to manage the lengthy expressions needed to determine the Lie point symmetries of differential equations; this set of procedures constantly grew through the years providing new capabilities, and now constitutes an extensively tested package able to perform almost automatically much of the work. Remarkably, the program can be used also in interactive mode mimicking the steps one has to do with pencil and paper but with the benefits of using a computer algebra system: this can represent a useful support in a higher course on symmetry analysis of differential equations or in all those situations (e.g., when one looks for conditional symmetries) where the determining equations can not be automatically solved and some special assumptions are needed.

Even if there exist many good packages working in different CAS and suitable for doing automatically the computation of Lie symmetries of differential equations, the opinion of the author is that this program, besides offering the possibility of being freely used without requiring the access to a commercial CAS, contains procedures for investigating problems in different areas of modern group analysis (classical point, contact and variational symmetries, group-classification problems, Q-conditional symmetries, Lie remarkable equations); in addition, it is the first program where the approach to approximate symmetries proposed in [60] can be automatically exploited. Last but not the least, its use is not limited to face only simple problems but allows the user to analyze differential
equations where the needed computations for finding the symmetries require to solve equations with a huge number of terms.

The plan of the paper is the following. In Section 2, we review the basic elements of Lie group theory of differential equations with reference to point symmetries (Section 2.1), conditional symmetries (Section 2.2), contact symmetries (Section 2.3), variational symmetries (Section 2.4), approximate point, conditional, contact and variational symmetries (Section 2.5), and equivalence transformations (Section 2.6); this Section is not intended as a tutorial for Lie group methods (the interested reader should refer to some of the monographs above quoted); it serves only to fix the notation and recall the very basic results in order to keep the paper almost self-contained. Then, in Section 3, we consider some examples in order to illustrate the use of ReLie, whereas Section 4 is devoted to a description of the global variables and the main functions contained in the package. In Sections 3 and 4, it is assumed that every potential user of ReLie is somehow familiar with Lie symmetry analysis of differential equations. Moreover, in order to fully benefit of the capabilities of ReLie, the user is required to have a minimal experience in working with the algebraic mode of Reduce [87]. Thus, in this paper, only the necessary details about the use of ReLie will be given. Finally, Section 5 contains some concluding remarks as well as possible future developments.

2. Basic Elements of the Theory

This Section reviews the main notions of modern group analysis, mainly covering the topics where the program ReLie may help the application of the theory to specific problems. First of all, we remark that the manifolds and maps introduced in the subsequent material are $C^\infty$. From a geometrical point of view, differential equations of order $r \geq 1$ characterize a submanifold of a suitable jet space (of order $r$) endowed with the equivalence relation given by the condition of contact of order $r$ [88]. Roughly speaking, the jet space is a manifold whose coordinates are interpreted as independent and dependent variables, and the derivatives of the latter with respect to the former up to the order $r$. In the following, we denote with $x \in \mathcal{X} \subseteq \mathbb{R}^n$ the independent variables, with $u \in \mathcal{U} \subseteq \mathbb{R}^m$ the dependent variables, and $u^{(r)}$ the set of all (partial) derivatives of the $u$’s with respect to the $x$’s up to the order $r \geq 1$, whereupon the dimension of the $r$-order jet space is $n + m(n+r)$. As a general rule, in the following, if needed, we use the Einstein convention of sums over repeated indices.

2.1. Lie Point Symmetries

Let

$$\Delta(x, u, u^{(r)}) = 0,$$

be a (system of) differential equation(s) of order $r$; it represents, on $n$-dimensional submanifolds of a manifold $E \subseteq \mathbb{R}^n \times \mathbb{R}^m$, a submanifold of the $r$-th order jet space [5,11,29].

Consider a one-parameter ($a$) Lie group of point transformations

$$x^\ast = X(x, u; a) = x + a \xi(x, u) + O(a^2),$$
$$u^\ast = U(x, u; a) = u + a \eta(x, u) + O(a^2),$$

where the functions $X$ and $U$ defined in $E \times S$ with values in $E$, where $S \subseteq \mathbb{R}$ is an interval, are $C^\infty$ with respect to the arguments $x$ and $u$, and analytic with respect to $a$; the components $\xi_i$ ($i = 1, \ldots, n$) of $\xi$, and $\eta_\alpha$ ($\alpha = 1, \ldots, m$) of $\eta$ are referred to as the infinitesimals of the Lie group; to the Lie group (2) it is associated the infinitesimal generator [3,5,6]

$$\Xi = \sum_{i=1}^n \xi_i(x, u) \frac{\partial}{\partial x_i} + \sum_{\alpha=1}^m \eta_\alpha(x, u) \frac{\partial}{\partial u_\alpha}.$$  

In dealing with differential equations, we need to prolong the action of the Lie group to the $r$-th order jet space. The transformations for derivatives are obtained by requiring that
the contact conditions are preserved by the transformation (roughly speaking, the transformed derivatives have to be the derivatives of the transformed dependent variables with respect to the transformed independent variables).

First order prolongation of the infinitesimal generator reads

$$\Xi^{(1)} = \Xi + \sum_{\alpha=1}^{m} \sum_{i=1}^{n} \eta_{[\alpha,i]} \frac{\partial}{\partial u_{\alpha,i}},$$

where

$$\eta_{[\alpha,i]} = \frac{D\eta_{[\alpha,i]}}{Dx_i} - \sum_{j=1}^{n} u_{\alpha,i} \frac{D\xi_j}{Dx_i},$$

and, in general, higher order extended infinitesimal generator has the form

$$\Xi^{(k)} = \Xi^{(k-1)} + \sum_{\alpha=1}^{m} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_k=1}^{n} \eta_{[\alpha,i_1 \cdots i_k]} \frac{\partial}{\partial u_{\alpha,i_1 \cdots i_k}},$$

with $\eta_{[\alpha,i_1 \cdots i_k]}$ defined recursively by the relation

$$\eta_{[\alpha,i_1 \cdots i_k]} = \frac{D\eta_{[\alpha,i_1 \cdots i_{k-1}]}_j}{Dx_{i_k}} - \sum_{j=1}^{n} u_{\alpha,i_1 \cdots i_{k-1}} \frac{D\xi_j}{Dx_{i_k}}.$$

the Lie derivative being

$$\frac{D}{Dx_i} = \frac{\partial}{\partial x_i} + \sum_{\alpha=1}^{m} \left( u_{\alpha,i} \frac{\partial}{\partial u_{\alpha}} + \sum_{j=1}^{n} u_{\alpha,ij} \frac{\partial}{\partial u_{\alpha,j}} + \cdots \right),$$

where $u_{\alpha,i} = \frac{\partial u_{\alpha}}{\partial x_i}, u_{\alpha,ij} = \frac{\partial^2 u_{\alpha}}{\partial x_i \partial x_j} \cdots$

According to Lie’s algorithm [3,5], which requires that the $r$-th order prolongation $\Xi^{(r)}$ acting on (1) is zero along the solutions of (1), i.e.,

$$\Xi^{(r)}(\Delta)\big|_{\Delta=0} = 0,$$

the generators of Lie groups of point transformations are found by solving a system of linear partial differential equations (the so called determining equations).

Remarkably, if all the derivatives occur polynomially in the differential equations, then the invariance condition (5) involves polynomials in the variables $u^{(r)}$ because the prolonged infinitesimals are always polynomials in the derivatives. Since we evaluate the invariance condition on the differential equations (so eliminating suitable leading derivatives), the derivatives appearing in the invariance condition are independent. Therefore, the latter vanishes if and only if all the coefficients of such polynomials (involving the infinitesimals, their derivatives, the independent and the dependent variables) are zero, whereupon one obtains an overdetermined system of linear and homogeneous partial differential equations (determining equations), whose integration provides the generators of Lie point symmetries admitted by (1). The infinitesimal generators of a Lie group, whose components are solutions of a linear homogeneous system of partial differential equations, span a vector space which can be finite or infinite-dimensional; by introducing an operation of commutation between two infinitesimal generators, this vector space gains the structure of a Lie algebra [89,90].

2.2. Q-Conditional Symmetries

It is well known that some differential equations of interest in the applications possess very few Lie symmetries. In 1969, Bluman and Cole [43] introduced a generalization of
classical Lie symmetries, and applied their method (called nonclassical) to the linear heat equation. The basic idea was that of imposing the invariance to a system made by the differential equation at hand, the invariant surface condition and the differential consequences of the latter \[20,21,91–97\]. This method leads to nonlinear determining equations whose general integration is usually difficult. Nonclassical symmetries are now part of conditional symmetries, i.e., symmetries of differential equations where some additional differential constraints are imposed to restrict the set of solutions; some recent applications of conditional symmetries to reaction–diffusion equations can be found in \[66,98–101\].

Given a differential equation
\[
\Delta \left( x, u, u^{(r)} \right) = 0, \tag{6}
\]
and considering the invariant surface conditions
\[
Q = \sum_{i=1}^{n} \xi_i(x, u) \frac{\partial u}{\partial x_i} - \eta(x, u), \quad \sum_{i=1}^{n} \xi_i^2 \neq 0, \tag{7}
\]
the Q-conditional symmetries are found by requiring
\[
\Xi^{(r)}(\Delta) \big|_{\mathcal{M}} = 0, \tag{8}
\]
where \( \mathcal{M} \) is the manifold of the jet space defined by the system of equations
\[
\Delta = 0, \quad Q = 0, \quad \frac{\partial^{[k]} Q}{\partial x_1^{k_1} \ldots \partial x_n^{k_n}} = 0, \tag{9}
\]
where \( 1 \leq |k| = k_1 + \ldots + k_n \leq r - 1 \).

The set of classical Lie symmetries of differential equations (spanning a vector space which is also a Lie algebra) is contained in the set of Q-conditional symmetries; however, the latter, in general, are not the elements of a Lie algebra. Moreover, it is simple to recognize from (7) that multiplying the infinitesimals of a Q-conditional symmetry by an arbitrary non-vanishing smooth function of dependent and independent variables produces still a Q-conditional symmetry. As a consequence, Q-conditional symmetries can be searched distinguishing \( n \) different cases by assuming
\[
\xi_j = 0 \quad \text{for} \quad 1 \leq j < i \leq n, \quad \text{and} \quad \xi_i = 1.
\]

When \( m > 1 \) (more than one dependent variable), one can look for conditional symmetries of \( q \)-th type \( (1 \leq q \leq m) \) \[66,100,101\], by requiring the condition
\[
\Xi^{(r)}(\Delta) \big|_{\mathcal{M}_q} = 0, \tag{10}
\]
where \( \mathcal{M}_q \) is the manifold of the jet space defined by the system of equations
\[
\Delta = 0, \quad Q_i = \cdots = Q_{iq} = 0, \quad \frac{\partial^{[k]} Q_{i_{1}}}{\partial x_1^{k_1} \ldots \partial x_n^{k_n}} = \cdots = \frac{\partial^{[k]} Q_{i_{q}}}{\partial x_1^{k_1} \ldots \partial x_n^{k_n}} = 0, \tag{11}
\]
where \( 1 \leq |k| = k_1 + \cdots + k_n \leq r - 1, 1 \leq i_1 < \cdots < i_q \leq m \). It is obvious that we may have in principle \( \binom{m}{q} \) different sets of Q-conditional symmetries of \( q \)-th type, whereas for \( q = 0 \) we simply have classical Lie symmetries. Moreover, when \( q < m \), it may result easier to find solutions to the nonlinear determining equations. Q-conditional symmetries allow for symmetry reductions of differential equations, and may provide explicit solutions not obtainable with classical symmetries.
2.3. Contact Transformations

Besides point transformations whose infinitesimals depend at most on independent and dependent variables, contact transformations \[5,102,103\] play an important role in many areas. Nevertheless, true contact transformations exist for differential equations involving only one dependent variable (for more than one dependent variable, they are prolongations of point transformations), and their infinitesimal generator has the form

\[
\Xi = \sum_{i=1}^{n} \xi_i(x, u, u^{(1)}) \frac{\partial}{\partial x_i} + \eta(x, u, u^{(1)}) \frac{\partial}{\partial u} + \sum_{i=1}^{n} \eta_{[i]}(x, u, u^{(1)}) \frac{\partial}{\partial u_{[i]}}. \tag{12}
\]

The operator (12) characterizes a group of contact transformations if and only if there exists a function \(\Omega(x, u, u^{(1)})\) (characteristic function) such that

\[
\xi_i = \frac{\partial \Omega}{\partial u_{[i]}}, \quad \eta = \Omega - \sum_{i=1}^{n} u_{[i]} \frac{\partial \Omega}{\partial u_{[i]}}, \quad \eta_{[i]} = \frac{\partial \Omega}{\partial x_i} + u_{[i]} \frac{\partial \Omega}{\partial u}. \tag{13}
\]

Prolongations of contact transformations for higher order derivatives are computed with the usual formulas.

Proper contact transformations (i.e., transformations that are not prolongations of point transformations) are the ones determined by a characteristic function \(\Omega\) that is nonlinear in the first order derivatives.

2.4. Variational Symmetries

Differential equations expressed as conservation laws play an important role in mathematical physics. From the physical viewpoint, they express conservation of physical quantities (mass, momentum, angular momentum, energy, etc.) whereas from a mathematical perspective are relevant in problems related to the integrability, existence, uniqueness and stability of solutions, as well as in the implementation of efficient numerical methods of integration \[104,105\].

The link between Lie symmetries admitted by differential equations and conservation laws has been established in 1918 by Emmy Noether \[28\] with her famous theorem applicable to differential equations derived by looking for the extrema of the action integral of a given Lagrangian function. The essence of Noether theorem is that to a Lie symmetry of the action integral (variational symmetry) there corresponds a local conservation law (i.e., a divergence expression) through an explicit formula that involves the infinitesimals of the point symmetry and the Lagrangian itself.

Consider a functional given by an integral over a domain \(\Omega \subset \mathbb{R}^n\), say

\[
\mathcal{J} = \int_{\Omega} \mathcal{L}(x, u, u^{(r)}) \, dx, \tag{14}
\]

where \(\mathcal{L}(x, u, u^{(r)})\) is a sufficiently smooth Lagrangian function \[106\].

By imposing the functional \(\mathcal{J}\) to be stationary for suitable variations of \(u\) and \(u^{(r)}\) vanishing on the boundary of the domain \(\Omega\) (Hamilton principle), we derive the Euler–Lagrange equations

\[
E_{u_{[a]}}(\mathcal{L}) = 0, \quad a = 1, \ldots, m, \tag{15}
\]

where

\[
E_{u_{[a]}}(\cdot) = \frac{\partial(\cdot)}{\partial u_{[a]}} - D \left( \frac{\partial(\cdot)}{\partial u_{[a,j]}} \right) + \frac{D^2}{Dx_j Dx_{j'}} \left( \frac{\partial(\cdot)}{\partial u_{[a,j,j']}} \right) - \cdots + (-1)^{r} \frac{D^r}{Dx_{j_1} \cdots Dx_{j_r}} \left( \frac{\partial(\cdot)}{\partial u_{[a,j_1 \cdots j_r]}} \right) \tag{16}
\]
is the Euler operator with respect to \( u_{ij} \), whereas \( \frac{D}{Dx_i} \) is the Lie derivative defined in (4), and

\[
\frac{D^r}{Dx_{i_1} \cdots Dx_{i_r}} = \frac{D}{Dx_j} \left( \frac{D^{r-1}}{Dx_j \cdots Dx_{j_r-1}} \right).
\]

Noether’s theorem [28] states that if we know a Lie group of point transformations with the generator (3) leaving the action integral invariant, and this holds true if the condition

\[
\Xi^{(r)}(\mathcal{L}) + \mathcal{L} \sum_{i=1}^{n} \frac{D\xi_i}{Dx_i} = \sum_{i=1}^{n} \frac{D\phi_i}{Dx_i}
\]

for some functions \( \phi_i(x, u, u^{(r-1)}) \) \( (i = 1, \ldots, n) \) is satisfied, then the conservation law

\[
\sum_{i=1}^{n} \frac{D}{Dx_i} (\xi_i \mathcal{L} + W_i - \phi_i) = 0,
\]

where

\[
W_i = (\eta_i - \xi_j u_{ij}) \left( \frac{\partial \mathcal{L}}{\partial u_{ij}} + \cdots + (-1)^{r-1} \frac{D^{r-1}}{Dx_{i_1} \cdots Dx_{i_r-1}} \left( \frac{\partial \mathcal{L}}{\partial u_{ij_1 \cdots j_{r-1}}} \right) \right)
\]

and \( \delta (\eta_i - \xi_j u_{ij}) \)

holds for any solution \( u(x) \) of Euler–Lagrange equations. For \( n = 1 \) (in such a case the independent variable is the time), and for the classical Lagrangians of Analytical Mechanics (sum of kinetic energy and potential of the active forces) the conservation law provides a first integral of ordinary differential equations.

There are some extensions of Noether theorem, for instance the one by Boyer [107] that can be used for constructing conservation laws using the invariance with respect to generalized symmetries [5], i.e., symmetries with infinitesimals depending on higher order derivatives (see [17]), as well as direct methods [108,109] applicable also to differential equations not derived from Lagrangian functions (see also [110,111]).

2.5. Approximate Symmetries

Let

\[
\Delta \left( x, u, u^{(r)}; \varepsilon \right) = 0
\]

be a differential equation of order \( r \) involving a small parameter \( \varepsilon \). By perturbing an equation, even adding terms involving coefficients with very small order of magnitude, the consequences are often dramatic since many symmetries are lost; this inconvenient restricts the range of applicability of Lie group methods to differential equations arising in some concrete applications. Since perturbation techniques [112] are widely used in the literature for investigating differential equations containing small terms, it is important to develop methods for combining Lie group theory with perturbation analysis. In the literature, two main approaches for establishing an approximate symmetry theory of such differential equations do exist and have been widely applied: One has been proposed in 1988 by Baikov, Gazizov and Ibragimov [57] (see also [58]) and another one in 1989 by Fushchich and Shtelen [59]. Both approaches have pros and cons [60]. To overcome these problems, recently, an alternative approach has been proposed [60], that is consistent with perturbation theory, and allows to extend all the relevant features of Lie group analysis to an approximate context; in some sense, this approach mixes the ideas in [57] with the ones in [59]. In what follows this latter approach is briefly described.
By looking for Lie point symmetries of Equation (19), the infinitesimal generators of the admitted Lie symmetries do not necessarily depend on the parameter $\varepsilon$. Nevertheless, it is a typical experience that some symmetries admitted by the unperturbed equation

$$\Delta \left( x, u, u^{(r)}; 0 \right) = 0 \quad (20)$$

may disappear. Limiting ourselves to solutions in the form

$$u(x; \varepsilon) = \sum_{k=0}^{p} \varepsilon^k u_{(k)}(x) + O(\varepsilon^{p+1}), \quad (21)$$

the differential equation writes as

$$\Delta \approx \sum_{k=0}^{p} \varepsilon^k \tilde{\Delta}_{(k)} \left( x, u_{(0)}, u^{(r)}_{(0)}, \ldots, u_{(k)}, u_{(k)}^{(r)} \right) = 0, \quad (22)$$

where, once fixed the integer $p$, the notation $f \approx g$ stands for $f = g + O(\varepsilon^{p+1})$. Now, let us consider a Lie generator

$$\Xi = \sum_{i=1}^{n} \xi_i(x, u; \varepsilon) \frac{\partial}{\partial x_i} + \sum_{\alpha=1}^{m} \eta_{\alpha}(x, u; \varepsilon) \frac{\partial}{\partial u_{\alpha}}, \quad (23)$$

and assume that the infinitesimals $\xi_i$ and $\eta_{\alpha}$ depend on the small parameter $\varepsilon$ and can be expanded in power series of $\varepsilon$ around $\varepsilon = 0$, whereupon, using (21), we have

$$\tilde{\xi}_i \approx \sum_{k=0}^{p} \varepsilon^k \tilde{\xi}_{(k)i}, \quad \eta_{\alpha} \approx \sum_{k=0}^{p} \varepsilon^k \tilde{\eta}_{(k)\alpha}, \quad (24)$$

where

$$\tilde{\xi}_{(0)i} = \xi_{(0)i} = \tilde{\xi}_i(x, u_{(0)}, 0), \quad \tilde{\eta}_{(0)\alpha} = \eta_{(0)\alpha} = \eta_{\alpha}(x, u_{(0)}, 0),$$

$$\tilde{\xi}_{(k+1)i} = \frac{1}{k+1} \mathcal{R}[\tilde{\xi}_{(k)i}], \quad \tilde{\eta}_{(k+1)\alpha} = \frac{1}{k+1} \mathcal{R}[\tilde{\eta}_{(k)\alpha}], \quad (25)$$

\(\mathcal{R}\) being a linear recursion operator satisfying product rule of derivatives and such that

$$\mathcal{R} \left[ \frac{\partial^{|\tau|} f_{(k)}(x, u_{(0)})}{\partial u_{(0)}^{i_1} \cdots \partial u_{(0)}^{i_m}} \right] = \frac{\partial^{|\tau|} f_{(k+1)}(x, u_{(0)})}{\partial u_{(0)}^{i_1} \cdots \partial u_{(0)}^{i_m}} + \sum_{i=1}^{m} \frac{\partial}{\partial u_{(0)}^{i}} \left( \frac{\partial^{|\tau|} f_{(k)}(x, u_{(0)})}{\partial u_{(0)}^{i_1} \cdots \partial u_{(0)}^{i_m}} \right) u_{(1)i_{\tau}}, \quad (26)$$

$$\mathcal{R}[u_{(k)}] = (k+1)u_{(k+1)},$$

where $k \geq 0, j = 1, \ldots, m, |\tau| = \tau_1 + \cdots + \tau_m$.

Thence, we have an approximate Lie generator

$$\tilde{\Xi} \approx \sum_{k=0}^{p} \varepsilon^k \tilde{\Xi}_{(k)}, \quad (27)$$

where

$$\tilde{\Xi}_{(k)} = \sum_{i=1}^{n} \tilde{\xi}_{(k)i}(x, u_{(0)}, \ldots, u_{(k)}) \frac{\partial}{\partial x_i} + \sum_{\alpha=1}^{m} \tilde{\eta}_{(k)\alpha}(x, u_{(0)}, \ldots, u_{(k)}) \frac{\partial}{\partial u_{\alpha}}. \quad (28)$$

In order to find the approximate generators admitted by a differential equation, it is necessary to prolong the Lie generator to manage the transformation of derivatives. This is done naturally by imposing the derivatives to be transformed in such a way the contact conditions are preserved; the only required trick is that the prolongations must be
computed taking into account the expansions of \( \xi, \eta, u \), and dropping the \( O(\epsilon^{p+1}) \) terms.

**Example 1.** Let \( p = 1 \), and consider the approximate Lie generator

\[
\Xi \approx \sum_{i=1}^{n} \left( \xi(0)_i + \epsilon \left( \xi(1)_i + \sum_{\beta=1}^{m} \frac{\partial \xi(0)_i}{\partial u(0)_{\beta}} u(1)_{\beta} \right) \right) \frac{\partial}{\partial x_i} + \sum_{a=1}^{m} \left( \eta(0)_a + \epsilon \left( \eta(1)_a + \sum_{\beta=1}^{m} \frac{\partial \eta(0)_a}{\partial u(0)_{\beta}} u(1)_{\beta} \right) \right) \frac{\partial}{\partial u_a},
\]

(29)

where \( \xi(0)_i, \xi(1)_i, \eta(0)_a \) and \( \eta(1)_a \) depend on \( (x, u(0)) \). The first order prolongation is

\[
\Xi^{(1)} \approx \Xi + \sum_{a=1}^{m} \sum_{j=1}^{n} \eta_{[a,j]} \frac{\partial}{\partial u_{a,j}},
\]

(30)

where

\[
\eta_{[a,j]} = \frac{\tilde{D}}{\partial x_i} \left( \eta(0)_a + \epsilon \left( \eta(1)_a + \sum_{\beta=1}^{m} \frac{\partial \eta(0)_a}{\partial u(0)_{\beta}} u(1)_{\beta} \right) \right) + \sum_{j=1}^{n} \frac{\tilde{D}}{\partial x_i} \left( \xi(0)_j + \epsilon \left( \xi(1)_j + \sum_{\beta=1}^{m} \frac{\partial \xi(0)_j}{\partial u(0)_{\beta}} u(1)_{\beta} \right) \right) \left( u(0)_{a,j} + \epsilon u(1)_{a,j} \right),
\]

(31)

along with the Lie derivative

\[
\frac{\tilde{D}}{\partial x_i} = \frac{\partial}{\partial x_i} + \sum_{k=0}^{p} \sum_{a=1}^{m} \left( u(k)_{a,j} \frac{\partial}{\partial u(k)_{a}} + \sum_{j=1}^{n} u(k)_{a,j} \frac{\partial}{\partial u(k)_{a,j}} + \cdots \right).
\]

(32)

Similar reasonings lead to higher order prolongations.

The approximate (at the order \( p \)) invariance condition of a differential equation reads

\[
\Xi^{(p)}(\Delta) \bigg|_{\Delta=O(\epsilon^{p+1})} = O(\epsilon^{p+1}).
\]

(33)

Some simple consequences can be immediately proved [60]. The Lie generator \( \tilde{\Xi}(0) \) is always a symmetry admitted by the unperturbed equation, and the terms \( \sum_{k=0}^{p} \epsilon^k \tilde{\Xi}(k) \) represent the deformation of the symmetry because of the terms involving \( \epsilon \). Nevertheless, not all symmetries of the unperturbed equations are admitted as the zeroth terms of approximate symmetries (those admitted are called stable symmetries [57]). Moreover, if \( \tilde{\Xi} \) is the generator of an approximate Lie symmetry, then \( \epsilon \tilde{\Xi} \) is a generator of an approximate Lie point symmetry too.

By introducing an approximate Lie bracket [57,58], it can be proved that the approximate Lie point symmetries of a differential equation are the elements of an approximate Lie algebra (see also [60]).

Of course, if one considers differential equations containing small terms, it is possible to construct approximate conditional symmetries [113,114], the only difference with respect to exact symmetries being the structure of Lie generator which follows the approach described above. In analogy with the exact case, for \( p \)-th order approximate conditional symmetries we have \( n \) different cases according to the choices

\[
\tilde{\xi}(0)_j = \tilde{\xi}(1)_j = \cdots = \tilde{\xi}(p)_j = 0 \quad \text{for} \quad 1 \leq j < i \leq n,
\]

\[
\tilde{\xi}(0)_i = 1, \quad \tilde{\xi}(1)_i = \cdots = \tilde{\xi}(p)_i = 0.
\]
We may also consider approximate contact transformations for scalar differential equations containing small terms by assuming the characteristic function $\Omega$ to depend on the small parameter $\varepsilon$. As an example, limiting to first order approximate contact symmetries, the expansion for $\Omega$ reads

$$
\Omega = \Omega_0(x, u(0), u(0)_\mu) + \varepsilon \left( \Omega_1(x, u(0), u(0)_\mu) + \sum_{l=1}^n \Omega_{l0} \frac{\partial u(0)_\mu}{\partial u(1)_\mu} \right).
$$

Finally, also approximate variational symmetries for Lagrangians containing small terms can be defined.

2.6. Equivalence Transformations

There are situations where the differential equations at hand contain unspecified functions (arbitrary elements); therefore, it is convenient to consider a class $\mathcal{E}(p)$ of differential equations involving a vector $p$ of arbitrary continuously differentiable functions $p_k$ ($k = 1, \ldots, \ell$) that can depend on some variables.

Let us consider a class of differential equations

$$
\Delta \left( x, u, u^{(s)}; p, p^{(s)} \right) = 0,
$$

whose elements are given once we fix the functions $p_k$, where $p^{(s)}$ denotes the set of all derivatives up to the order $s$ of the $p$’s with respect to their arguments; these arguments may be the independent variables, the dependent ones, and also the derivatives of the latter with respect to the former up to a fixed order $q$. Let us denote the arguments of the arbitrary elements $p$ as $z \equiv (z_1, \ldots, z_N)$.

In such a framework, it is possible to consider equivalence transformations, i.e., transformations that may change the form of the arbitrary parameters but preserve the differential structure of the equations [3,50–55].

By generalizing Lie groups of point transformations, equivalence transformations [3] of a family $\mathcal{E}(p)$ of differential equations may be expressed as

$$
x^* = X(x, u, p; a) = x + a\xi(x, u, p) + O(a^2),
$$

$$
u^* = U(x, u, p; a) = u + a\eta(x, u, p) + O(a^2),
$$

$$
p^* = P(x, u, p; a) = p + a\mu(x, u, p) + O(a^2),
$$

$a$ being the parameter; they are locally $C^\infty$ diffeomorphisms mapping a class $\mathcal{E}(p)$ of differential equations into itself.

A common assumption consists in considering the transformations (36) where $\xi$ and $\eta$ are assumed independent of the arbitrary elements $p$.

By introducing an augmented space $\mathcal{A}$ [3,50], where the independent variables, the dependent variables and the arbitrary functions live, respectively, the generator of the equivalence transformation

$$
\Xi = \sum_{i=1}^m \xi_i(x, u) \frac{\partial}{\partial x_i} + \sum_{a=1}^n \eta_a(x, u) \frac{\partial}{\partial u_a} + \sum_{j=1}^\ell \mu_j(x, u, p) \frac{\partial}{\partial p_j},
$$

involves also the infinitesimals $\mu_j(x, u, p)$ accounting for the arbitrary functions $p_j$.

The first prolongation of $\Xi$ writes as

$$
\Xi^{(1)} = \Xi + \sum_{a=1}^n \sum_{i=1}^m \eta_{[a_i]} \frac{\partial}{\partial u_{a_j}} + \sum_{j=1}^\ell \sum_{\beta=1}^N \mu_{[j]} \frac{\partial}{\partial p_{j,\beta}}.
$$
with

$$\eta_{[\alpha,\beta]} = \frac{D\eta_{\alpha}}{Dx_i} - \sum_{j=1}^{n} u_{\gamma j} \frac{D\xi_j}{Dx_i}, \quad \mu_{[\gamma,\beta]} = \frac{D\mu_{\gamma}}{Dz_{\beta}} - \sum_{\gamma=1}^{N} p_{\gamma,\gamma} \frac{D\xi_{\gamma}}{Dz_{\beta}} \tag{39}$$

where $p_{\gamma,\gamma} = \frac{\partial p_{\gamma}}{\partial z_{\gamma}}$ and $\xi_{\gamma}$ is the infinitesimal generator of $z_{\gamma}$, along with the Lie derivatives $\frac{D}{Dx_i}$ already defined in (4) and

$$\tilde{D}_{\beta} = \tilde{D}_{\beta \gamma} = \frac{\partial}{\partial z_{\beta}} + \sum_{k=1}^{\ell} p_{k,\beta} \frac{\partial}{\partial p_{k}} \tag{40}$$

Higher order prolongations are immediately obtained by recursion,

$$\Xi^{(k)} = \Xi^{(k-1)} + \sum_{\alpha=1}^{m} \sum_{\beta_1=1}^{n} \sum_{\beta_2=1}^{n} \cdots \sum_{\beta_{k-1}=1}^{n} \eta_{[\alpha,\beta_1...\beta_{k-1}]} \frac{\partial}{\partial u_{\alpha,\beta_1...\beta_{k-1}}} + \sum_{j=1}^{\ell} \sum_{\beta_1=1}^{n} \sum_{\beta_2=1}^{n} \cdots \sum_{\beta_{k-1}=1}^{n} \mu_{[\gamma,\beta_1...\beta_{k-1}]} \frac{\partial}{\partial p_{\gamma,\beta_1...\beta_{k-1}}} \tag{41}$$

where

$$\mu_{[\gamma,\beta_1...\beta_{k-1}]} = \frac{\tilde{D}_{\beta} \mu_{[\beta_1...\beta_{k-1}]}}{Dz_{\beta_k}} - \sum_{\gamma=1}^{N} p_{\gamma,\gamma} \frac{D\xi_{\gamma}}{Dz_{\beta_k}} \tag{41}$$

As proposed by Ovsiannikov [3], the search for continuous equivalence transformations can be exploited by using the Lie’s infinitesimal criterion, i.e., imposing the condition

$$\Xi^{(\text{max}(r,\sigma + 1))}(\Delta)|_{\Delta=0} = 0. \tag{42}$$

When the infinitesimals $\xi_i$ and $\eta_\alpha$ are assumed to be independent of $p$, it is possible to project the symmetries on the space $Z \equiv \mathcal{X} \times \mathcal{U}$ of the independent and dependent variables, so obtaining a transformation changing an element of the class of differential equations to another element in the same class (see [115–117] for some applications of these ideas).

If some arbitrary elements do not depend upon some variables, the differential equations at hand have to be supplemented with auxiliary conditions. For instance, the auxiliary conditions for the equivalence transformations of the class of differential equations [52]

$$\frac{\partial^2 u}{\partial t^2} - f(x, \frac{\partial u}{\partial x}) \frac{\partial^2 u}{\partial x^2} + g(x, \frac{\partial u}{\partial x}) = 0, \tag{42}$$

where $f$ and $g$ are unspecified functions of their arguments, are:

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial u}, \quad \frac{\partial f}{\partial (\partial u/\partial t)} = \frac{\partial g}{\partial t} = \frac{\partial g}{\partial u} = \frac{\partial g}{\partial (\partial u/\partial t)} = 0. \tag{43}$$

More in general, taking the transformations of the independent and dependent variables as functions of the arbitrary elements $p$ too [118], in the expression of prolongation we have to replace the Lie derivative $\frac{D}{Dx_i}$ with

$$\frac{D}{Dx_i} = \frac{\partial}{\partial x_i} + \sum_{a=1}^{m} \left( u_{a,i} \frac{\partial}{\partial u_a} + \sum_{j=1}^{n} u_{a,j} \frac{\partial}{\partial u_{a,j}} + \cdots \right) + \sum_{k=1}^{\ell} \left( \frac{\partial p_k}{\partial x_i} + \sum_{\beta=n+1}^{N} \frac{\partial p_k}{\partial z_{\beta}} \frac{\partial}{\partial x_i} \right) \frac{\partial}{\partial p_k} \tag{43}$$
2.7. Lie Remarkable Equations

In [119], within the framework of inverse Lie problem, strongly and weakly Lie remarkable differential equations have been defined; relevant examples of such equations have been studied in [120–122]. Their analysis involves the study of the rank of the distribution of prolongations of a Lie algebra of generators.

3. The Program ReLie

The application of Lie group theory (together with its generalizations) to differential equations usually involves a lot of lengthy and cumbersome computations, so that the implementation of the algorithms in a CAS is highly desired [10,12,73–86].

In this Section, we illustrate the use of the Reduce [87] package ReLie by considering illustrative examples; though some of them are simple, we underline that more complex applications can be efficiently faced as well. The package ReLie is a collection of (algebraic) procedures allowing the user to investigate ordinary and partial differential equations within the framework of Lie symmetry analysis. By using the program, it is possible to compute almost automatically: Lie point symmetries, conditional symmetries, contact symmetries, variational symmetries (all these symmetries may be either exact or approximate) of differential equations, and equivalence transformations for classes of differential equations containing arbitrary elements. Moreover, the program implements functions for computing Lie brackets, the commutator table of a list of Lie generators, and the distribution of an algebra of Lie symmetries (useful in the context of inverse Lie problem [119–122]). Remarkably, the program can be used interactively in all the cases where the determining equations are not automatically solved (for instance, when one looks for conditional symmetries or in some group classification problems).

After entering Reduce, assuming that we are in the directory containing the source file relie.red, the set of routines implemented in the package ReLie becomes available after the statement

```
in "relie.red" $
```

The loading of ReLie is successful if the following output is displayed:

```
ReLie, version 3.0
A Reduce program for Lie group analysis of differential equations
(c) Francesco Oliveri (foliveri@unime.it) - 2021
Last update 9 August 2021
http://mat521.unime.it/oliveri/
```

Alternatively, one may include the package in the Reduce image by issuing in a Reduce session the statements

```
faslout 'relie $
in "relie.red" $
faslend $
```

If all works, ReLie is loaded through the command

```
load_package relie $
```

According to the task we are interested to, some input data have to be provided to the program. The minimal required set of data is given by a positive integer value for the global variable jetorder (the maximum order of prolongation of the Lie generator), the list xvar of the independent variables, and the list uvar of the dependent variables.

**Example 2.** The assignments

```
jetorder:=2 $ 
xvar:={t,x} $ 
uvar:={u} $
```
refer to a second order partial differential equation for the unknown \( u(t,x) \).

The number of independent as well as dependent variables is arbitrary; moreover, the identifiers of independent and dependent variables are arbitrary provided that no conflict arises with reserved words of Reduce or identifiers used by ReLie (see Section 4 for a list of internal global variables used by ReLie). The value of \( \text{jetorder} \) is not bounded by ReLie.

The first function one has to call is \text{relieinit()}, that defines and initializes all the needed objects required to perform Lie group analysis. It checks the input and possibly displays some warnings. If only \( \text{jetorder}, xvar \) and \( uvar \) have been assigned, then calling \text{relieinit()} produces the output:

The list ‘diffeqs’ of differential equation(s) is missing!
The list ‘leadders’ of leading derivative(s) is missing!
Check ‘diffeqs’ and/or ‘leadders’!!
You may only call relieprol() or generatealgebra(k) (k = 1,2,3).

In fact, we did not define the list \text{diffeqs} of differential equations and the list \text{leadders} of leading derivatives (or the list \text{lagrangian} for variational symmetries). Nevertheless, we can compute the \text{jetorder}-th prolongation of the vector field generating a Lie group of point transformation. The infinitesimal generators of the independent and dependent variables are automatically defined by the program (therefore, the user is not requested to set them), and stored in the list \text{allinfinitesimals}. The infinitesimals of the independent variables are denoted by \( xi_\_ \) followed by the identifiers in \( xvar \); analogously, the infinitesimals of the dependent variables are denoted by \( eta_\_ \) followed by the identifiers in \( uvar \). According to the Example 2, the list \text{allinfinitesimals} is a list of two elements, each element being in this case the list \( \{xi_t, xi_x, eta_u\} \) (this redundancy is used in order to have a unified representation for dealing also with other kinds of symmetries, see below). The dependence of the elements in \text{allinfinitesimals} upon the independent and dependent variables (the elements in \( xvar \) and \( uvar \), respectively) are automatically set by ReLie.

By calling the ReLie function \text{relieprol()}, as a result we get the list \text{prolongation}. This list has two elements: The first one is a list containing the coordinates of the jet space, the second one the list of the corresponding infinitesimal generators. Thus, the second element of \text{prolongation} contains the components of the prolonged vector field up to \text{jetorder}-th order.

ReLie internally represents the derivatives of the dependent variables upon the independent variables by appending to the identifiers of the dependent variable the underscore _ followed by the identifiers of the involved independent variables. For mixed derivatives the order of the independent variables in the internal representation of derivatives reflects the order in the list \( xvar \). For instance, if \( xvar \) is \( \{t,x\} \) and \( uvar \) is \( \{u\} \), the left-hand side of the equation (Benjamin–Bono–Mahoney equation)

\[
\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} + u \frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^2 \partial t} = 0
\]  

(44)

has to be represented as

\[ u_t + u_x + uu_x - u_{txx} . \]

3.1. Computing Lie Point Symmetries of Differential Equations

To compute the Lie point symmetries admitted by differential equations, in addition to \text{jetorder}, \( xvar \) and \( uvar \), we have to provide:

1. the list \text{diffeqs} of the left-hand sides of the differential equations to be studied which are assumed with zero right-hand sides;
2. the list leadders of some derivatives appearing in the differential equations: When computing the invariance conditions, the elements in the list leadders are removed by solving the differential equations with respect to them.

Both lists, diffeqs and leadders, need to have the same length, otherwise a warning is displayed; moreover, the differential equations must be solvable with respect to the leading derivatives. The user should be careful in the choice of leadders, especially in the case of systems of differential equations, in order to guarantee that the differential equations stored in the list diffeqs can be solved with respect to the elements in leadders.

Example 3. Once ReLie has been loaded, the statements

\[
\text{jetorder}:=3 \$
\text{xvar}:=\{x\} \$
\text{uvar}:=\{u\} \$
\text{diffeqs}:={u_{xxx} + u*u_{xx}/2} \$
\text{leadders}:={u_{xxx}} \$
\text{relieinit()} \$
\text{relieinv()} \$
\text{reliedet()} \$
\text{reliesolve()} \$
\]

allow to compute the Lie point symmetries of Blasius equation [123]

\[
\frac{d^3 u}{dx^3} + \frac{1}{2} \frac{d^2 u}{dx^2} = 0.
\]

The call to the function relieinit() initializes the needed objects, then the call to the function relieinv() provides the invariance condition of the differential equation defined in diffeqs; in other terms, the jetorder-th prolongation of the vector field of the Lie group is applied to the differential equation and the equation itself is used to eliminate \(\frac{d^3 u}{dx^3}\). As a result, the list invcond is provided:

\[
\text{invcond} -> \\
\{0 \times \}
\]

Since we have a scalar ordinary differential equation, the list invcond contains just one element, which is a polynomial in the variables \(u_x\) and \(u_{xx}\). Invoking the function reliedet(), one gets the list deteqs of the coefficients of such (list of) polynomial(s); thus, we have the determining equations, i.e., a system of linear partial differential equations for the infinitesimals \(xi_x\) and \(eta_u\)

\[
\text{deteqs} -> \\
\{0 \times \}
\]

- 2*df(xi_x,u,x)*u,
- 12*df(xi_x,u,2),
2*df(eta_u,u,3) - 6*df(xi_x,u,2,x) - df(xi_x,u,2)*u,
- 2*df(xi_x,u,3)}

We may automatically solve this set of determining equations by calling the function `reliesolve()`, and the result turns out to be contained in the list `symmetries`:

```
symmetries ->
{
{
},
{eta_u= - k_1*u,
xi_x=k_1*x + k_2},
{k_1,k_2},
{} }
```

The function `reliesolve()` uses the Reduce package CRACK by S. Wolf [80], specifically suitable to solve (overdetermined) systems of linear partial differential equations.

The list `symmetries` contains an element (when more than one solution set is possible for particular choices of the involved parameters this list has one element for each solution set, see later) which in turn is a list of four elements:

1. the first one is a list of conditions that are still unsolved (in this case the list is empty since no condition remains unsolved);
2. the second one is a list giving the solution to the determining equations, i.e., the expressions of the infinitesimals;
3. the third one is a list containing the parameters involved in the solution (in this case \( k_1 \) and \( k_2 \));
4. the fourth one is a list of expressions which can not vanish (in this case the list is empty).

The parameters involved in the symmetries of Blasius equation are constant; symmetries of partial differential equations may involve arbitrary functions that ReLie denotes as \( f_1, f_2, \ldots \); we can check the variables such parameters depend on by passing the name of the function to the procedure `fargs()` (described in CRACK [80]).

Since the infinitesimals are solutions of linear homogeneous differential equations, they are linear combinations of fundamental solutions. We can compute these fundamental solutions through the use of the function `reliegen()`. This function requires two arguments: An integer and a list. If `reliesolve()` returns only one solution this integer is 1; when more than one solution is available (this may occur in group classification problems or in the search of conditional symmetries), the integer selects the solution. The second argument is a list of values. If the length of this list corresponds to the number of arbitrary (constant) parameters involved in the chosen symmetries, then a list of vector fields (generators) is returned where the \( i \)-th parameter involved in the symmetries is replaced by the \( i \)-th element in the list and the other parameters are replaced by 0; otherwise, the \( i \)-th parameter is replaced by 1 and the other parameters are replaced by 0. For instance, invoking `reliegen(1,\{\})` (the same results could be obtained by writing also `reliegen(1,\{1,1\})`, but the first form is clearly simpler):

```
generators ->
{
{x, -u},
{1,0} }
```

Split `symmetries` ->

```
splitsymmetries ->
{
{xi_x=x, eta_u=-u},
{xi_x=1, eta_u=0} }
```
We have the list `generators` where each element is the list of the components of the vector field generating a symmetry (the components of the infinitesimals of the independent and dependent variables), and the list `splitsymmetries` whose elements are the expressions of the infinitesimals; the vector fields here characterize a scaling and a translation of the independent variable, respectively:

\[ \Xi_1 = x \frac{\partial}{\partial x} - u \frac{\partial}{\partial u}, \quad \Xi_2 = \frac{\partial}{\partial x}. \]

**Remark 1.** By calling `reliegen(1,{-1})`, the lists `generators` and `splitsymmetries` contain only one element representing the components of the infinitesimals in their general form, i.e., the linear combinations of all admitted generators. For Blasius equation, the call `reliegen(1,{-1})` provides

```latex
generators -> 
{ 
{k_1*x + k_2,- k_1*u} 
}
splitsymmetries -> 
{ 
{xi_x=k_1*x + k_2, eta_u=-k_1*u} 
}
```

**Remark 2.** Notice that `reliegen()`, unless its second parameter is \{-1\}, returns the generators of a finite-dimensional Lie algebra even when the admitted symmetries are infinite-dimensional!

In the above example, we detailed all the steps leading to the computation of the point symmetries of a differential equation. Nevertheless, we have a general function, called `relie()`, requiring an integer argument:

1. `relie(1)` is equivalent to calling `relieinit()`;
2. `relie(2)` is equivalent to calling in sequence `relieinit()` and `relieinv()`;
3. `relie(3)` is equivalent to calling in sequence `relieinit()`, `relieinv()` and `reliedet()`;
4. `relie(4)` is equivalent to calling in sequence `relieinit()`, `relieinv()`, `reliedet()` and `reliesolve()`.

**Example 4.** After loading ReLie, the statements

```latex
jetorder:=2 $ 
xvar:={t,x} $ 
uvar:={u} $ 
diffeqs:={u_t-u_xx} $ 
leadders:={u_t} $ 
relie(4) $ 
```

allow one to easily compute the Lie point symmetries of the linear heat equation

\[ \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0. \]

The output produced by ReLie is

```latex
symmetries -> 
{ 
{df(f_1,t)-df(f_1,x,2)}, 
{ 
eta_u=(-8*f_1+2*k_1*u*x+2*k_4*t*u+k_4*u*x**2+2*k_6*u)/8, 
xi_x=(-k_1*t-k_2-k_3*x-k_4*t*x)/2, 
xi_t=(-2*k_3*t-k_4*t**2-2*k_5)/2
```

The infinitesimals of the point symmetries of the heat equation involve 7 parameters. The parameter $f_1$ is a function, whereas the other parameters are constant. By calling \texttt{fargs(f_1)}, we have as a result \{t, x\}, denoting that the function $f_1$ is a function of $t$ and $x$; moreover, this function satisfies the heat equation (the first element of the list \texttt{symmetries}). The symmetry corresponding to this parameter is responsible for the linear superposition principle of the solutions (typical of linear equations).

Calling the function \texttt{reliegen(1,\{-2, -2, -2, -2, -1, 4\})}, the result is the list \texttt{generators} of vector fields of the finite subalgebra of the Lie algebra of symmetries of linear heat equation:

\begin{verbatim}
generators -> 
{ 
{0, t, (-u*x)/2}, 
{0, 1, 0}, 
{2*t, x, 0}, 
{t**2, t*x, (u*(-2*t - x**2))/4}, 
{1, 0, 0}, 
{0, 0, u} 
} 
\end{verbatim}

We can reorder the vector fields in \texttt{generators} by using the function \texttt{newordering}()

\begin{verbatim}
generators:=newordering(generators,\{5, 2, 6, 3, 1, 4\}) $ 
so obtaining

\begin{verbatim}
generators -> 
{ 
{1, 0, 0}, 
{0, 1, 0}, 
{0, 0, u}, 
{2*t, x, 0}, 
{0, t, -u*x/2}, 
{(t**2, t*x, u*(-2*t - x**2))/4}, 
} 
\end{verbatim}

i.e.,

\begin{align*}
\Xi_1 &= \frac{\partial}{\partial t}, \\
\Xi_2 &= \frac{\partial}{\partial x}, \\
\Xi_3 &= u \frac{\partial}{\partial u}, \\
\Xi_4 &= 2t \frac{\partial}{\partial t} + x \frac{\partial}{\partial x}, \\
\Xi_5 &= t \frac{\partial}{\partial x} - \frac{ux}{2} \frac{\partial}{\partial u}, \\
\Xi_6 &= t^2 \frac{\partial}{\partial t} + xt \frac{\partial}{\partial x} - \frac{u(2t + x^2)}{4} \frac{\partial}{\partial u},
\end{align*}

spanning a 6-dimensional Lie algebra. The symmetry corresponding to the function $f_1$,

\begin{align*}
\Xi_f = f(t, x) \frac{\partial}{\partial u} \quad \text{with} \quad \frac{\partial f}{\partial t} - \frac{\partial^2 f}{\partial x^2} = 0,
\end{align*}

renders the algebra of symmetries of linear heat equation infinite-dimensional.

The function \texttt{reliedet()} assumes that the invariance conditions are polynomials in the derivatives; this is true if all derivatives appear polynomially in the differential equations at hand. In the case where the differential equations we are investigating contain some derivatives not in polynomial form, ReLie constructs correctly the determining equations if we inform the program about the derivatives not occurring in polynomial form. These derivatives have to be the elements of the list \texttt{nonpolyders}.
Example 5 (Differential equations not polynomial in some derivatives). The following code allows the user to compute the Lie symmetries of the equation

$$\frac{\partial^2 u}{\partial t \partial x} + \exp \left( \frac{\partial u}{\partial x} \right) - u = 0.$$

```plaintext
jetorder:=2 $ 
jetorder:=2 $ 
xvar:={t,x} $ 
xvar:={t,x} $ 
uvar:={u} $ 
uvar:={u} $ 
diffeqs:={u抽象(u)+exp(u抽象(u)-u)} $ 
diffeqs:={u抽象(u)+exp(u抽象(u)-u)} $ 
leadders:={u抽象(u)} $ 
leadders:={u抽象(u)} $ 
nonpolyders:={u抽象(u)} $ 
onpolyders:={u抽象(u)} $ 
relie(4) $ 
relie(4) $ 
```

Remark 3. The use of ReLie is not limited to simple cases. For instance, consider the Navier-Stokes-Fourier equations of a gas in a rotating (with constant angular velocity $\omega$ along the vertical axis) frame reference and subject to gravity,

$$\frac{\partial \rho}{\partial t} + 3 \sum_{k=1}^{3} \frac{\partial (\rho v_k)}{\partial x_k} = 0,$$

$$\frac{\partial (\rho v_i)}{\partial t} + \sum_{k=1}^{3} \frac{\partial}{\partial x_k} \left\{ \rho v_i v_k - \frac{2}{\alpha} \left( \frac{K}{m} \right) \frac{T}{2} \left( \frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} \right) - \frac{1}{3} \frac{\partial v_i}{\partial x_k} \delta_{ik} \right\} = \rho F_i, \quad i = 1, 2, 3,$$

$$\frac{\partial (\rho T)}{\partial t} + \frac{\partial}{\partial x_k} \left\{ \rho T v_k - \frac{5}{2\alpha} \left( \frac{K}{m} \right) T \frac{\partial T}{\partial x_k} \right\} + \sum_{i,k=1}^{3} \left\{ \frac{2}{3} \rho T \delta_{ik} - \frac{4}{3\alpha} \left[ \frac{1}{2} \left( \frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} \right) - \sum_{\ell=1}^{3} \frac{1}{3} \frac{\partial v_\ell}{\partial x_k} \delta_{ik} \right] \right\} \frac{\partial v_i}{\partial x_k} = 0,$$

where $\rho$ is the mass density, $v \equiv (v_1, v_2, v_3)$ the velocity, $T$ the temperature, $F_i$ the $i$-th component of the sum of external and inertial force,

$$F = (\rho(2\omega v_2 + \omega^2 x_1), \rho(-2\omega v_1 + \omega^2 x_2), -\rho g),$$

$g$ the gravitational acceleration, $K$ the Boltzmann constant, $m$ the mass of a single particle, and the coefficient $\alpha$ is an appropriate constant which describes the interaction between the Maxwellian molecules. The computation of the Lie symmetries [124], spanning a 12th dimensional Lie algebra, is done automatically in few seconds by means of ReLie.

3.1.1. An Example of Group Classification

ReLie is able to face the problem of group classification of differential equations. In fact, in certain cases it may be interesting to check if special instances of arbitrary constants and/or functions involved in the differential equations lead to different sets of Lie symmetries. This problem is faced by including the parameters we want to check in the list `freepars`, as illustrated by the following example. Moreover, we may define the list `nonzeropars` containing parameters (constants or functions involved in the differential equations) or expressions that can not be vanishing.
Example 6. Consider the 3D Euler equations of ideal gas dynamics,

$$\frac{\partial \rho}{\partial t} + \sum_{k=1}^{3} v_k \frac{\partial \rho}{\partial x_k} + \rho \sum_{k=1}^{3} \frac{\partial v_k}{\partial x_k} = 0,$$

$$\rho \left( \frac{\partial v_i}{\partial t} + \sum_{k=1}^{3} v_k \frac{\partial v_i}{\partial x_k} \right) + \frac{\partial p}{\partial x_i} = 0, \quad i = 1, 2, 3,$$

$$\frac{\partial p}{\partial t} + \sum_{k=1}^{3} v_k \frac{\partial p}{\partial x_k} + \Gamma \rho \sum_{k=1}^{3} \frac{\partial v_k}{\partial x_k} = 0,$$

where $\rho(t, x)$ is the mass density, $v_i(t, x)$ the components of velocity, $p(t, x)$ the pressure, and $\Gamma$ the adiabatic exponent. The following code performs the required task:

```text
jetorder:=1 $
$xvar:={t,x1,x2,x3} $
$uvar:={rho,v1,v2,v3,p} $
$freepars:={gamma} $
$nonzeropars:={gamma} $
$diffeqs:={rho_t+v1*rho_x1+v2*rho_x2+v3*rho_x3+rho*(v1_x1+v2_x2+v3_x3),
 rho*(v1_t+v1*v1_x1+v2*v1_x2+v3*v1_x3)+p_x1,
 rho*(v2_t+v1*v2_x1+v2*v2_x2+v3*v2_x3)+p_x2,
 rho*(v3_t+v1*v3_x1+v2*v3_x2+v3*v3_x3)+p_x3,
 p_t+v1*p_x1+v2*p_x2+v3*p_x3+gamma*p*(v1_x1+v2_x2+v3_x3),}$
$leadders:={rho_t,v1_t,v2_t,v3_t,p_t}$ $
$relie(4) $
```

As a result, we get

```
symmetries ->
{
{
{gamma=5/3, eta_p=5*k_10*p*t -k_14*p, eta_v3=k_10*t*v3-k_10*x3+k_12*v3+k_3*v2-k_5*v1-k_7-k_8*v3, eta_v2=-k_1+k_10*t*v2-k_2*x1-k_3*x3-k_4-k_8*v2, eta_v1=k_10*t*v1-k_11+k_12*v1+k_2*x2+k_5*v3-k_8*v1, eta_rho=3*k_10*p*t-2*k_12*rho-2*k_8*rho, xi_x3=-k_10*t*x3+k_3*x2-k_5*x1-k_6-k_7*t-k_8*x3, xi_x2=-k_1*t-k_10*t*x2-k_2*x1-k_3*x3-k_4-k_8*x2, xi_x1=-k_10*t*x1-k_11+t+k_2*x2+k_5*x3-k_8*x1-k_9, xi_t=-k_10*t**22-k_12*t-k_13,}
{
k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_10, k_11, k_12, k_13, k_14}
},
{gamma=5/3, eta_p=-k_14*p, eta_v3=k_12*v3+k_3*v2-k_5*v1-k_7-k_8*v3, eta_v2=-k_1+k_12*v2+k_2*x1+k_3*x3+k_4-k_8*x2, eta_v1=k_11+k_12*v1+k_2*v2+k_5*v3-k_8*v1, eta_rho=-2*k_12*rho-2*k_8*rho, xi_x3=k_3*x2-k_5*x1-k_6-k_7*t-k_8*x3, xi_x2=-k_1*t-k_10*t*x2-k_2*x1-k_3*x3-k_4-k_8*x2, xi_x1=-k_10*t*x1-k_11+t+k_2*x2+k_5*x3-k_8*x1-k_9, xi_t=-k_10*t**2-k_12*t-k_13,
```

As a result, we get
In this case, symmetries is a list of two elements since we have different solutions to the determining equations according to the value of \( \gamma \) (included in the list freepars). Looking at the results provided by ReLiE, the first solution set refers to \( \Gamma = 5/3 \), the second one to a value of \( \Gamma \) different from 0 and 5/3. By calling
```
reliegen(2,{-1,1,-1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1})
generators:=newordering(generators,{12,9,4,6,2,5,3,10,1,7,11,8,13})
```
we obtain the following result
```
generators ->
{1,0,0,0,0,0,0,0,0},
{0,1,0,0,0,0,0,0,0},
{0,0,1,0,0,0,0,0,0},
{0,0,0,1,0,0,0,0,0},
{0,x2, - x1,0,v2, - v1,0,0,0},
{0,x3,0, - x1,v3,0, - v1,0,0},
{0,0,x3, - x2,0,v3,0, - v2,0},
{t,0,0,0,1,0,0,0,0},
{0,0,0,0,1,0,0,0,0},
{t,0,0,0,2*rho,-v1,-v2,-v3,0},
{o,x1,x2,x3,-2*rho,v1,v2,v3,0},
{0,0,0,0,rho,0,0,0,p}
```
so that for \( \Gamma \) unspecified we have a 13-dimensional Lie algebra of point symmetries spanned by

\[
\Xi_1 = \frac{\partial}{\partial t}, \quad \Xi_{1+k} = \frac{\partial}{\partial x_k}, \quad \Xi_{4+k} = t \frac{\partial}{\partial x_k} + \frac{\partial}{\partial v_k}, \quad k = 1,2,3,
\]

\[
\Xi_{5+i+j} = x_i \frac{\partial}{\partial x_j} - x_j \frac{\partial}{\partial x_i} + v_i \frac{\partial}{\partial v_j} - v_j \frac{\partial}{\partial v_i}, \quad 1 \leq i < j \leq 3,
\]

\[
\Xi_{11} = t \frac{\partial}{\partial t} + 2\rho \frac{\partial}{\partial \rho} - \sum_{k=1}^3 v_k \frac{\partial}{\partial v_k}, \quad \Xi_{12} = 3 \sum_{k=1}^3 \left( x_k \frac{\partial}{\partial x_k} + v_k \frac{\partial}{\partial v_k} \right) - 2\rho \frac{\partial}{\partial \rho},
\]

\[
\Xi_{13} = \rho \frac{\partial}{\partial \rho} + p \frac{\partial}{\partial p}.
\]

Calling
```
reliegen(1,{-1,1,-1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1})
generators:=newordering(generators,{13,9,4,6,2,6,3,11,1,7,12,8,14,10})
```
we have that for \( \Gamma = 5/3 \) there is a fourteenth symmetry generated by

\[
\Xi_{14} = t^2 \frac{\partial}{\partial t} + \sum_{k=1}^3 \left( x_k \frac{\partial}{\partial x_k} + (x_k - v_k) \frac{\partial}{\partial v_k} \right) - 3pt \frac{\partial}{\partial p} - 5pt \frac{\partial}{\partial p}.
\]
Remark 4. The differential equations can contain arbitrary functions (in this case we need to declare in Reduce the variables these functions depend on), and they can be elements of the list freepars and/or nonzeropars.

For instance, if we want to classify the symmetries admitted by the Korteweg–deVries equation with variable coefficients [125]

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} + f(t)u = 0,$$

where $f(t)$ is an unspecified function of time, the needed code is the following one:

jetorder:=3 $
\text{xvar:}\{t,x\} $
\text{uvar:}\{u\} $
\text{depend f,t} $
\text{freepars:}\{f\} $
\text{diffeqs:}\{u_t+u*u_x+u_xxx+f*u\} $
\text{leadders:}\{u_t\} $
\text{relie(4)}$

3.1.2. Commutator Table

ReLie is able to compute the commutator table of a set of infinitesimal generators spanning a Lie algebra. Let us illustrate how to do with an example.

Example 7. Consider the Burgers’ equation

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

To compute its Lie symmetries, we provide to ReLie the following input:

jetorder:=2 $
\text{xvar:}\{t,x\} $
\text{uvar:}\{u\} $
\text{diffeqs:}\{u_t+u*u_x-u_xx\} $
\text{leadders:}\{u_t\} $
\text{relie(4)}$

We get quickly the output

symmetries ->
{
{}
{},
{0, - k_1*t, - k_1},
{0, - k_2,0},
{ - k_3*t,(-k_3*x)/2,(k_3*u)/2},
{( - k_4*t**2)/2,( - k_4*t*x)/2,(k_4*(t*u - x))/2},
{ - k_5,0,0}
},
{}
} $}

Therefore, we have a 5-dimensional Lie algebra of point symmetries. After extracting and reordering the components of the infinitesimal generators,

reliegen(1,{-1,-1,-2,-2,-1}) $
generators:=newordering(generators,\{5,2,1,3,4\})$
we have

\[
\text{generators} \to \\
\{1, 0, 0\}, \\
\{0, 1, 0\}, \\
\{0, t, 1\}, \\
\{2*t, x, -u\}, \\
\{t**2, t*x, -t*u + x\}
\]

that is a 5-dimensional Lie algebra spanned by

\[
\begin{align*}
\Xi_1 &= \frac{\partial}{\partial t}, \\
\Xi_2 &= \frac{\partial}{\partial x}, \\
\Xi_3 &= t \frac{\partial}{\partial x} + \frac{\partial}{\partial u}, \\
\Xi_4 &= 2t \frac{\partial}{\partial t} + x \frac{\partial}{\partial x} - u \frac{\partial}{\partial u}, \\
\Xi_5 &= t^2 \frac{\partial}{\partial t} + xt \frac{\partial}{\partial x} + (x - tu) \frac{\partial}{\partial u}.
\end{align*}
\] (45)

The commutator table of the generators (45) is obtained by calling

\[
\text{commsortable(generators)};
\]

that produces the output

\[
\text{commsortable} \to \\
\{0, 0, -vf_1, -vf_5, 0\}, \\
\{0, 0, -2*vf_2, -vf_3, -vf_1\}, \\
\{vf_1, 2*vf_2, 0, -2*vf_4, -vf_5\}, \\
\{vf_5, vf_3, 2*vf_4, 0, 0\}, \\
\{0, vf_1, vf_5, 0, 0\}
\]

where the five elements of the list \text{generators} have been referred to as \text{vf} _1, \text{vf} _2, \text{vf} _3, \text{vf} _4, \text{vf} _5, respectively.

3.2. Computation of Conditional Symmetries

Conditional symmetries are computed by using the same functions as above; we only need to assign a non-zero value (in the range between 1 and the number of independent variables) to the variable \text{nonclassical} and define the list \text{qcond} (a list of distinct integers chosen in the range from 1 to the number of dependent variables). Suppose that \text{xvar:}={x1,x2,x3} and \text{uvar:}={u,v}. The components of the Lie generator of the conditional symmetries are

- \{1, xi_x2, xi_x3, eta_u, eta_v\} if \text{nonclassical} = 1;
- \{0, 1, xi_x3, eta_u, eta_v\} if \text{nonclassical} = 2;
- \{0, 0, 1, eta_u, eta_v\} if \text{nonclassical} = 3.

The list \text{qcond} tells to the program which invariant surface conditions have to be used to restrict the solution manifold:

- \text{qcond:}={1,2} if the invariant surface conditions of both dependent variables have to be used;
- \text{qcond:}={1} if the invariant surface condition of the first dependent variable has to be used;
- \text{qcond:}={2} if the invariant surface condition of the second dependent variable has to be used.

Looking for conditional symmetries leads to nonlinear determining equations, whereupon it is not unusual that \text{reliesolve()} may fail to recover the solution. In such cases, it is necessary to make some special assumptions on the infinitesimals and/or try to solve interactively the determining equations.
Example 8. Nonclassical symmetries of linear heat equation

The code

\begin{verbatim}
jetorder:=2 $
xvar:={t,x} $
uvar:={u} $
nonclassical:=1 $
qcond:={1} $
diffeqs:={u_t-u_xx} $
leadders:={u_xx} $
relie(4) $
\end{verbatim}

immediately computes the conditional symmetries, corresponding to the operator

\begin{equation}
\Xi = \frac{\partial}{\partial t} + \xi(t, x, u) \frac{\partial}{\partial x} + \eta(t, x, u) \frac{\partial}{\partial u}, \tag{46}
\end{equation}

of the linear heat equation

\begin{equation}
\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0. \tag{47}
\end{equation}

As a result, we get

\begin{verbatim}
symmetries ->
{
{
{df(f_1,t) - df(f_1,x,2) - 2*df(f_2,x,2)*f_1,
2*df(f_2,t,x,2) + 2*df(f_2,t,x)*df(f_2,x) - df(f_2,t,2)
+ 2*df(f_2,t)*df(f_2,x,2) - df(f_2,x,4)
- 2*df(f_2,x,3)*df(f_2,x) - 4*df(f_2,x,2)**2
- 2*df(f_2,x,2)*df(f_2,x)**2
},
{eta_u=(df(f_2,t)*u-df(f_2,x,2)*u-df(f_2,x)**2*u-2*f_1)/2,
xi_x= - df(f_2,x)
},
{f_1,f_2
}
}
}
\end{verbatim}

Therefore, we have:

\begin{equation}
\xi = -\frac{\partial f_2}{\partial x}, \quad \eta = -f_1 + \frac{1}{2} \left( \frac{\partial f_2}{\partial t} - \frac{\partial^2 f_2}{\partial x^2} - \left( \frac{\partial f_2}{\partial x} \right)^2 \right) u, \tag{48}
\end{equation}

where $f_1(t, x)$ and $f_2(t, x)$ satisfy the differential equations

\begin{equation}
\begin{aligned}
\frac{\partial f_1}{\partial t} - \frac{\partial^2 f_1}{\partial x^2} - 2 f_1 \frac{\partial^2 f_2}{\partial x^2} &= 0,
2 \frac{\partial^3 f_2}{\partial t \partial x^2} + 2 \frac{\partial^2 f_2}{\partial t \partial x} \frac{\partial f_2}{\partial x} - \frac{\partial^2 f_2}{\partial t^2} + 2 \frac{\partial f_2}{\partial t} \frac{\partial^2 f_2}{\partial x^2} - \frac{\partial^4 f_2}{\partial x^4} - 2 \frac{\partial^2 f_2}{\partial x^2} \frac{\partial^2 f_2}{\partial x^3} \\
&+ 4 \left( \frac{\partial^2 f_2}{\partial x^2} \right)^2 - 2 \left( \frac{\partial f_2}{\partial x} \right)^2 \frac{\partial^2 f_2}{\partial x^3} = 0.
\end{aligned} \tag{49}
\end{equation}

Remark 5. When computing conditional symmetries we have to be careful in the choice of leaders, since also the invariant surface conditions and their differential consequences are solved with respect to some derivatives.
3.3. Computation of Contact Symmetries

For computing contact symmetries of a scalar differential equation we need to assign the value 1 to the variable contact. The functions that we have to call do not change, as the following example shows.

**Example 9.** Let us compute the contact symmetries of the ordinary differential equation

$$\frac{d^3u}{dx^3} = 0.$$  

The code is as follows.

```math
jetorder:=3 $  
xvar:={x} $  
uvar:={u} $  
contact:=1 $  
diffeqs:={u_xxx} $  
leadders:={u_xxx} $  
relie(4) $
```

As a result we get:

```
symmetries ->  
{
    {},  
    {omega=(-2*k_1*x**2-4*k_10*u_x-4*k_2*x-4*k_3*u-4*k_4+4*k_5*u*u_x-2*k_5*u_x**2*x-4*k_6*u**2+4*k_6*u*u_x*x-k_6*u_x**2*x**2-2*k_7*u_x**2-4*k_8*u_x*x+4*k_9*u_x-2*k_9*u_x*x**2)/4},  
    {k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8,k_9,k_10}  
}  
```

i.e., the expression of the characteristic function \( \Omega(x, u, du/dx) \) involves 10 arbitrary constants. A basis of the 10-dimensional Lie algebra of contact symmetries is obtained by invoking

```
reliegen(1,{-1,-1,-1,-1,-1,-1,1,1,1,1}) $
```

providing

```
generators ->  
{
    {0, x**2/2, x},  
    {0, x, 1},  
    {0, u, u_x},  
    {0, 1, 0},  
    {u - u_xx, ( - u_x**2*x)/2, ( - u_x**2)/2},  
    {(x*(2*u - u_xx))/2, (4*u**2 - u_x**2*x**2)/4, (u_x*(2*u - u_x*x))/2},  
    {u_x, u_x**2/2, 0},  
    {x, 0, - u_x},  
    {x**2/2, u*x, u},  
    {1, 0, 0}  
}
```

Note that only the fifth, sixth and seventh generators correspond to proper contact transformations; the remaining ones are prolongations of point symmetries.

3.4. Computation of Variational Symmetries and Associated Conservation Laws

For computing variational symmetries, besides assigning `jetorder`, `xvar` and `uvar` we need to assign the list `lagrangian` with only one element, and set to 1 the variable
variational. The functions that we have to call do not change, as the following example shows.

**Example 10.** The code

```plaintext
jetorder:=1 $
variational:=1 $
xvar:={t} $
uvar:={u} $
lagrangian:={t^2*u_t^2/2-t^2*u^6/6} $
relie(4) $
```

allows the user to compute the variational symmetries of the Lagrangian

\[ \mathcal{L} = \frac{t^2}{2} \left( \frac{du}{dt} \right)^2 - \frac{t^2 u^6}{6}, \]

corresponding to the Emden–Fowler equation

\[ \frac{d^2u}{dt^2} + \frac{2}{t} \frac{du}{dt} + u^5 = 0. \]

We get

```
symmetries ->
{
{"},
{\phi_t=k_2/6, \eta_u=(k_1*u)/2, \xi_t=-k_1*t},
{k_1,k_2},
{"}
}
```

where, besides the expressions of the infinitesimal generators, we have also the function \( \phi \) entering the definition of variational symmetries.

Calling `reliegen(1,{})`, we obtain the two lists

```
generators ->
{
{-t, u/2}
}
```

and

```
cogenerators ->
{
{0}
}
```

i.e., the (list of) Lie generator(s) of the variational symmetries and the (list of) function(s) entering the invariance condition of the Lagrangian action.

We have only a Lie generator, whereas the function \( \phi \) can be taken without loss of generality equal to zero.

Finally, calling `relieclaw(1)`, we get

```
fluxes ->
{
(t**2*(t*u**6 + 3*t*u_t**2 + 3*u*u_t))/6
}
```

i.e., the first integral

\[ \frac{1}{6} \left( t^2 u^6 + 3 t \left( \frac{du}{dt} \right)^2 + 3 u \frac{du}{dt} \right) = \text{constant}. \]
Example 11 (Klein–Gordon equation (see [17])). The class of Klein–Gordon wave equations,
\[ \frac{\partial^2 u}{\partial t^2} + g(u) = 0 \] (50)
with a general nonlinear interaction term \( g(u) \), can be derived from a variational principle given by requiring the stationarity of the action functional with Lagrangian
\[ \mathcal{L} = -\frac{1}{2} \frac{\partial u}{\partial t} \frac{\partial u}{\partial x} + h(u), \quad h'(u) = g(u). \] (51)
For an arbitrary \( g(u) \), the generators of these symmetries are
\[ \Xi_1 = t \frac{\partial}{\partial t} - x \frac{\partial}{\partial x}, \quad \Xi_2 = \frac{\partial}{\partial t}, \quad \Xi_3 = \frac{\partial}{\partial x}, \] (52)
and it is straightforward to verify that they are variational symmetries of the action functional.

Applying Noether’s theorem, the following conservation laws are easily derived:
\[ \frac{D}{Dt} \left( \frac{1}{2} \left( \frac{\partial u}{\partial x} \right)^2 \right) + \frac{D}{Dx} (h(u)) = 0, \]
\[ \frac{D}{Dt} (h(u)) + \frac{\partial}{\partial x} \left( \frac{1}{2} \left( \frac{\partial u}{\partial t} \right)^2 \right) = 0, \] (53)
\[ \frac{D}{Dt} \left( \frac{1}{2} x \left( \frac{\partial u}{\partial x} \right)^2 - th(u) \right) + \frac{D}{Dx} \left( -\frac{1}{2} \left( \frac{\partial u}{\partial t} \right)^2 + xh(u) \right) = 0. \]

The Reduce code for this computation is
```
jetorder:=1 
xvar:={t,x} 
uvar:={u} 
depend h,u 
lagrangian:={-u_t*u_x/2+h} 
variational:=1 
relie(4) 
reliegen(1,{-1,-1,-1})
```
The fluxes entering the conservation laws associated to the three Lie generators are obtained with the calls `relieclaw(1)`, `relieclaw(2)`, and `relieclaw(3)`, respectively.

3.5. Computation of Approximate Symmetries

Approximate Lie point symmetries of differential equations containing small terms are computed according to the approach described in [60]. The small parameter must be denoted by \( \epsilon \). In addition to the input data described before, the user has to define the integer \( \text{approxorder} \) for the order of approximation, and a rule, say \( \text{let} \epsilon^{\text{approxorder}+1} = 0 \). The same functions providing the necessary computations for exact Lie point symmetries work in this case too.

Example 12. The following code shows how to compute first order approximate symmetries of the Korteweg–deVries–Burgers equation
\[ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} - \epsilon \frac{\partial^2 u}{\partial x^2} = 0 \] (54)
```
jetorder:=3 
approxorder:=1 
let epsilon**2 = 0 
xvar:={t,x}
```
As a result, we get:

symmetries ->
{
{
{},
{eta1_u=( - 3*k_4 + 2*k_6*u0)/3,
eta0_u= - k_1,
xi1_x=( - 3*k_4*t - 3*k_5 - k_6*x)/3,
xi1_t= - k_6*t - k_7,
xi0_x= - k_1*t - k_2,
xi0_t= - k_3},
{k_1,k_2,k_3,k_4,k_5,k_6,k_7},
{}}
}

Invoking reliegen(1,{-1,-1,-1,-1,-1,-1,-1}) provides
generators ->
{
{0,t,1},
{0,1,0},
{1,0,0},
{0,epsilon*t,epsilon},
{0,epsilon,0},
{epsilon*t,(epsilon*x)/3,( - 2*epsilon*u0)/3},
{epsilon,0,0}
}

so that a basis of the approximate Lie algebra of symmetries of Equation (54) is (after a reordering)

\[ \Xi_1 = \frac{\partial}{\partial t}, \quad \Xi_2 = \frac{\partial}{\partial x}, \quad \Xi_3 = t \frac{\partial}{\partial x} + \frac{\partial}{\partial u}, \quad \Xi_4 = \epsilon \left( t \frac{\partial}{\partial t} + \frac{x}{3} \frac{\partial}{\partial x} - \frac{2}{3} u_0 \frac{\partial}{\partial u} \right), \quad \Xi_5 = \epsilon \left( t \frac{\partial}{\partial x} + \frac{\partial}{\partial u} \right) = \epsilon \Xi_3, \quad \Xi_6 = \epsilon \frac{\partial}{\partial t} = \epsilon \Xi_1, \quad \Xi_7 = \epsilon \frac{\partial}{\partial x} = \epsilon \Xi_2. \]

Remark 6. ReLie is able to expand automatically in power series of \( \epsilon \) a differential equation or a Lagrangian involving small terms. The only case where the user is forced to expand in power series of \( \epsilon \) up to the desired order of approximation some parts of the differential equation (or of the Lagrangian) is when these parts involve unspecified functions depending on the dependent variables and/or their derivatives. This can be illustrated by the following example of the computation of the first order approximate symmetries of the equation

\[ \frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x} \left( f(u) \frac{\partial u}{\partial x} \right) + \epsilon \frac{\partial u}{\partial t} = 0, \]

where we have to write the code

depend f,u0 $
diffeqs:=(u_tt-(f+epsilon*df(f,u0)*u1)*u_xx-(df(f,u0)
+epsilon*df(f,u0,2)*u1)*u_x^2+epsilon*u_t) $
Of course, the user may insert a differential equation completely expanded in power series of $\varepsilon$, but this is not strictly required if no unspecified functions of dependent variables and/or their derivatives occur.

Finally, we notice that in order to compute approximate conditional, contact or variational symmetries the only difference with respect to the corresponding exact case consists in setting a positive value to approxorder and defining the rule

\[ \text{let } \epsilon^{(\text{approxorder}+1)} = 0 \]

3.6. Computation of Equivalence Transformations

If we consider a class of differential equations, and want to determine equivalence transformations some more input data are necessary. In particular, in addition to the data already discussed for the computation of Lie point symmetries, we need to set:

1. the list arbelem of the arbitrary elements involved in the differential equations;
2. the integer arborder denoting the highest order of the derivatives of the arbitrary elements with respect to their arguments;
3. the integer zorder characterizing the variables the arbitrary elements depend on; for instance, if zorder is 0, then the arbitrary elements depend at most on the independent and dependent variables; if zorder is 1, then the arbitrary elements depend at most on the independent, dependent variables and first order derivatives, ….

If we want to remove the dependence of some arbitrary elements on some variables, we need to add auxiliary conditions to the differential equations at hand.

The infinitesimal generators of the arbitrary elements are automatically computed by the program (so the user is not requested to set them), and stored, together with the infinitesimals of independent and dependent variables in the list allinfinitesimals.

The infinitesimal generator of an arbitrary element is denoted by $\mu_{\text{name}}$ followed by the name of the arbitrary element. Let us illustrate with two different examples how ReLie works in such a situation.

Example 13 ([116]). Consider the class $\mathcal{E}(p)$ with $p = (p_1, p_2, p_3, p_4, p_5, p_6)$ of systems

\[
\begin{align*}
\frac{\partial u_1}{\partial t} + \frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} &= 0, \\
\frac{\partial u_2}{\partial t} + \frac{\partial p_1}{\partial x} + \frac{\partial p_2}{\partial y} - p_5 &= 0, \\
\frac{\partial u_3}{\partial t} + \frac{\partial p_3}{\partial x} + \frac{\partial p_4}{\partial y} - p_6 &= 0,
\end{align*}
\]

(56)

where $t, x$ and $y$ are the independent variables, $u_1, u_2$ and $u_3$ the dependent variables, whereas $p_i \equiv p_i(t, x, y, u_1, u_2, u_3)$ ($i = 1, \ldots, 6$) stand for arbitrary continuously differentiable functions of the indicated arguments.

ReLie determines the equivalence transformations with the code

\[
\text{jetorder}:=1 \\
\text{uvar}:={u_1,u_2,u_3} \\
\text{xvar}:={t,x,y} \\
\text{arbelem}:={p_1,p_2,p_3,p_4,p_5,p_6} \\
\text{arborder}:=1 \\
\text{zorder}:=0 \\
\text{diffeqs}:={u_1_t+u_2_x+u_3_y,} \\
\text{leadders}:={u_1_t,u_2_t,u_3_t} \\
\text{relie}(4) \\
\]
As a result we have the infinitesimals:

\[
\begin{align*}
\text{symmetries} & \rightarrow \\
& \{ \\
& \{ \\
& \text{df}(f_3,y) + \text{df}(f_6,t) + \text{df}(f_7,x) \\
& \}, \\
& \{ \\
& \text{eta}_u_3 = -\text{df}(f_1,t) * u_1 - \text{df}(f_1,x) * u_2 + \text{df}(f_10,t) * u_3 + \text{df}(f_2,x) * u_3 + f_3 - k_1 * u_3, \\
& \text{eta}_u_2 = \text{df}(f_1,y) * u_2 + \text{df}(f_10,t) * u_2 - \text{df}(f_2,t) * u_1 - \text{df}(f_2,y) * u_3 + f_7 - k_1 * u_2, \\
& \text{eta}_u_1 = \text{df}(f_1,y) * u_1 + \text{df}(f_2,x) * u_1 + f_6 - k_1 * u_1, \\
& \text{xi}_y = -f_1, \\
& \text{xi}_x = -f_2, \\
& \text{xi}_t = -f_10, \\
& \text{mu}_{p6} = -2 * \text{df}(f_1,t,x) * u_2 - 2 * \text{df}(f_1,t,y) * u_3 - \text{df}(f_1,t,2) * u_1 - \text{df}(f_1,x,y) * p_2 \\
& - \text{df}(f_1,x,y) * p_3 - \text{df}(f_1,x,2) * p_1 - \text{df}(f_1,x) * p_5 - \text{df}(f_1,y,2) * p_4 \\
& + \text{df}(f_10,t,2) * u_3 + 2 * \text{df}(f_10,t) * p_6 + \text{df}(f_2,x) * p_6 + \text{df}(f_3,t) \\
& + \text{df}(f_4,x) - \text{df}(f_5,y) - k_1 * p_6, \\
& \text{mu}_{p5} = \text{df}(f_1,y) * p_5 - \text{df}(f_10,t,2) * u_2 + 2 * \text{df}(f_10,t) * p_5 - 2 * \text{df}(f_2,t,x) * u_2 \\
& - 2 * \text{df}(f_2,t,y) * u_3 + \text{df}(f_2,t,2) * u_1 - \text{df}(f_2,x,y) * p_2 \\
& - \text{df}(f_2,x,y) * p_3 - \text{df}(f_2,x,2) * p_1 - \text{df}(f_2,y,2) * p_4 - \text{df}(f_2,y) * p_6 \\
& + \text{df}(f_7,t) + \text{df}(f_8,y) + \text{df}(f_9,x) - k_1 * p_5, \\
& \text{mu}_{p4} = -2 * \text{df}(f_1,t,x) * u_3 - \text{df}(f_1,x) * p_2 - \text{df}(f_1,x) * p_3 - \text{df}(f_1,y) * p_4 \\
& + 2 * \text{df}(f_10,t) * p_4 - \text{df}(f_2,x) * p_4 - f_5 - k_1 * p_4, \\
& \text{mu}_{p3} = \text{df}(f_1,t) * u_2 - \text{df}(f_1,x) * p_1 + 2 * \text{df}(f_10,t) * p_3 - \text{df}(f_2,t) * u_3 \\
& - \text{df}(f_2,t) * p_4 - f_4 - k_1 * p_3, \\
& \text{mu}_{p2} = \text{df}(f_1,t) * u_2 - \text{df}(f_1,x) * p_1 + 2 * \text{df}(f_10,t) * p_2 - \text{df}(f_2,t) * u_3 \\
& - \text{df}(f_2,t) * p_4 - f_8 - k_1 * p_2, \\
& \text{mu}_{p1} = \text{df}(f_1,t) * u_2 - \text{df}(f_1,t) * u_2 - \text{df}(f_2,t) * u_2 - \text{df}(f_2,x) * p_1 \\
& - \text{df}(f_2,y) * p_2 - \text{df}(f_2,y) * p_3 - f_9 - k_1 * p_1 \\
& \}, \\
& \{ \\
& f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_10, k_1 \\
& \}, \\
& \{ \\
& \}
\}
\]

where $f_1, \ldots, f_{10}$ are functions of $t, x, y$, along with the constraint

\[
\frac{\partial f_6}{\partial t} + \frac{\partial f_3}{\partial x} + \frac{\partial f_7}{\partial y} = 0. \tag{57}
\]

**Example 14.** The equivalence transformations of the equation

\[
\frac{\partial^2 u}{\partial t^2} - f \left( x, \frac{\partial u}{\partial x} \right) \frac{\partial^2 u}{\partial x^2} - g \left( x, \frac{\partial u}{\partial x} \right) = 0 \tag{58}
\]

are immediately found in ReLie with the following code:

```
jetorder:=2 $ 
ivar:=[t,x] $ 
uvar:=[u] $ 
zorder:=1 $ 
arborder:=1 $ 
arbelem:=[f,g] $ 
diffseqs:=[u_{tt} - f*u_{xx} - g, f_t, g_t, f_u, g_u, f_{u_t}, g_{u_t}] $ 
leadders:=[u_{tt}, f_t, g_t, f_u, g_u, f_{u_t}, g_{u_t}] $ 
 relie(4) $
```
Notice that we have added in \texttt{diffeqs} the auxiliary conditions; the variable \texttt{zorder} has been set to 1 since the functions $f$ and $g$ depend on first order derivatives; moreover, the variable \texttt{arborder} is set to 1 since in the auxiliary conditions the derivatives of $f$ and $g$ with respect to some of their arguments occur.

As a result we obtain:

\begin{verbatim}
symmetries -> {
{
\{eta_u=(2*f_1-2*k_1*u-k_6*t**2-2*k_7*t-2*k_8)/2,
x1_x=-k_4*x-k_5,
x1_t=-k_2*t-k_3,
mu_g=-df(f_1,x,2)*f-g*k_1+2*g*k_2-k_6,
mu_f=2*f*k_2-2*f*k_4 \},
\{f_1,k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8\},
{}\}
}
\end{verbatim}

where the function $f_1$ is arbitrary and depends on $x$.

Remark 7. By default, the infinitesimals for the independent and dependent variables do not depend on arbitrary elements; if we are interested to general equivalence transformations where also the infinitesimal generators of the independent and dependent variables depend on the arbitrary elements \cite{118}, then we have to add the statement

\begin{verbatim}
generalequiv:=1 
\end{verbatim}

3.7. Inverse Lie Problem

Here we show with a simple example how ReLie can be used for investigating the Lie remarkable of differential equations \cite{119–122}. The following code shows how one can easily verify that Monge–Ampère equation describing a surface in $\mathbb{R}^3$ with zero Gaussian curvature,

$$
\frac{\partial^2 u}{\partial x^2} \frac{\partial^2 u}{\partial y^2} - \left( \frac{\partial^2 u}{\partial x \partial y} \right)^2 = 0,
$$

is strongly Lie remarkable \cite{119}:

\begin{verbatim}
jetorder:=2 
xvar:={x,y} 
uvar:={u} 
relieinit() 
generatealgebra(2) 
\end{verbatim}

After setting the necessary objects (\texttt{jetorder}, \texttt{xvar} and \texttt{uvar}), and calling \texttt{relieinit()}, we generate the vector fields spanning the affine Lie algebra in $\mathbb{R}^3$. Then, by the call to the function \texttt{testrank(generators)}, we ascertain that the rank of the second order distribution is maximal (equal to the dimension of second order jet space); the call \texttt{inverselie(8)} computes the list \texttt{allminors} of all minors of order 8 of such a distribution; all minors vanish by evaluating them on the differential equation, that can be verified by writing

\begin{verbatim}
allminors:=sub(u_yy=u_xy^2/u_xx,allminors) 
\end{verbatim}

As a last check,

\begin{verbatim}
rank(sub(u_yy=u_xy^2/u_xx,distribution));
\end{verbatim}

returns 7, which is the dimension of the submanifold in the second order jet space characterized by Monge–Ampère equation.
4. Inside ReLie: Global Variables and Routines

ReLie uses some global variables to perform the various tasks: They can be distinguished among input variables (that the user needs to set before starting computation), output variables (computed by ReLie and of interest to the user), and intermediate variables (computed by ReLie and in general not of interest to the common user).

4.1. Input Variables

- **approxorder**: Maximum order of approximate symmetries of equations containing a small parameter; by default it is 0, i.e., exact symmetries; the small parameter involved in the approximate symmetries must be denoted by \( \text{epsilon} \); when looking for approximate symmetries the user has to define the rule \( \text{let } \text{epsilon}^{(\text{approxorder}+1)}=0 \);
- **arbelem**: List of the arbitrary elements (only for equivalence transformations); by default it is an empty list;
- **arborder**: Maximum order of derivatives of arbitrary elements (only for equivalence transformations); by default it is \(-1\), i.e., point symmetries;
- **contact**: Set to 1 for contact symmetries; by default it is 0;
- **diffeqs**: List of the left-hand sides of differential equations (with vanishing right-hand sides);
- **freepars**: List of arbitrary constants or functions involved in the differential equations (for group classification problems); by default it is an empty list;
- **generalequiv**: Set to 1 for general equivalence transformations where all infinitesimals depend on independent and dependent variables and arbitrary elements; the default value is 0, meaning that the infinitesimals of independent and dependent variables do not depend upon the arbitrary elements;
- **jetorder**: Maximum order of derivatives in differential equations;
- **lagrangian**: A list with only one element corresponding to the Lagrangian (it is necessary to set variational to 1);
- **leadders**: List of the leading derivatives; diffeqs are solved with respect to them;
- **nonclassical**: Set to a value between 1 and the number of independent variables (only for conditional symmetries); by default it is 0, i.e., classical symmetries;
- **nonpolyders**: List of derivatives not occurring in polynomial form in the differential equations; by default it is an empty list;
- **nonzeropars**: List of arbitrary constants or functions involved in the differential equations that can not vanish; by default it is an empty list;
- **qcond**: List of indexes of dependent variables whose invariant surface conditions have to be used for computing conditional symmetries;
- **uvar**: List of the dependent variables;
- **variational**: Set to 1 if variational symmetries of a Lagrangian are needed; by default it is 0;
- **xvar**: List of the independent variables;
- **zorder**: Maximum order of derivatives of \( uvar \) with respect to \( xvar \) the elements in \( \text{arbelem} \) depend on (only for equivalence transformations); if \( \text{zorder} \) is set to 0, the arbitrary elements depend on \( xvar \) and \( uvar \); \( \text{zorder} \) cannot exceed \( \text{jetorder} \).

4.2. Output Variables

- **allinfinitesimals**: List of two lists; the first sublist is the list of the infinitesimals, in order, of the independent variables, dependent variables and arbitrary elements (the latter in the case of equivalence transformations); the second sublist is the list of various terms (constants and functions) involved in the expression of infinitesimals;
- **allminors**: List of minors of a given order extracted from a matrix; returned by the function \( \text{minors}(m,k) \), where \( m \) is a matrix and \( k \) a positive integer, or by the function \( \text{inverselie}(k) \) that takes the \( \text{jetorder} \)-th distribution of \( \text{generators} \) as the matrix from which the minors of order \( k \) are extracted;
- **arbconst**: List of arbitrary constants involved in the expression of the symmetries;
arbfun: List of arbitrary functions involved in the symmetries;
cogenerators: List of the functions entering the definition of variational symmetries and corresponding to the infinitesimal generators (produced by reliesgen());
commtable: Table of commutators of a list of vector fields;
deteqs: List of the determining equations (produced by reliedet());
distribution: Matrix of the jetorder-th distribution of a list of generators (produced by reliedistrib()), i.e., a matrix where each row is the prolonged vector field evaluated on one of the provided infinitesimal generators;
fluxes: List of the fluxes of the conservation law corresponding to a Lie generator (computed by relieclaw());
generators: List of the infinitesimal generators of the finite Lie algebra admitted by the differential equations at hand (produced by reliesgen()); the list generators may also be obtained by calling generatealgebra(k), where k can be 1 (algebra of isometries), 2 (algebra of affine transformations) or 3 (algebra of projective transformations); of course, it is necessary to set jetorder, xvar and uvar before calling generatealgebra(k);
invcond: List of the invariance conditions of the differential equations at hand (produced by relieinv());
nzcomm: List of non-zero commutators of a list of vector fields;
prolongation: List of two lists: The first one is the list of the coordinates of the jet space, the second one the list of the corresponding infinitesimals (produced by relieprol());
plitsymmetries: List of lists: The k-th element is a list containing the infinitesimals corresponding to the k-th element of generators (produced by reliesgen()); the list splitsymmetries may also be obtained by calling generatealgebra(k), where k can be 1 (algebra of isometries), 2 (algebra of affine transformations) or 3 (algebra of projective transformations); of course, it is necessary to set jetorder, xvar and uvar before calling generatealgebra(k); the list splitsymmetries is used internally by the functions reliedistrib(), inverselie() and testrank();
symmetries: List of the infinitesimals admitted by the differential equations at hand (produced by reliesolve());

4.3. Intermediate Variables

jet: List of three lists: Indices for computing the infinitesimals and their prolongations, coordinates of jet space and their internal representation;
jetapprox: List of two lists: List of independent variables and expansions of dependent variables and their derivatives, and list of their internal representation (only for approximate symmetries);
jetequiv: List of three lists: Indices for computing the infinitesimals and their prolongations for arbitrary elements, arbitrary elements, and their internal representation (only for equivalence transformations);
jetsplit: List of two lists: Indices for computing the infinitesimals and their prolongations, list of independent variables, zeroth order dependent variables and their derivatives (only for approximate symmetries);
solutiondev: Solution of the differential equations specified in diffeqs with respect to leadders; for conditional symmetries, the invariant surface conditions and their needed differential consequences are solved too;
steprelie: Integer that stores the status of the computation; 0: No computation done; 1: reliesinit() has been called; 2: relieinv() has been called; 3: reliedet() has been called; 4: reliesolve() has been called;
zvar: List of two lists: The first one is the list of the variables arbelem depend on, the second one the corresponding infinitesimals (only for equivalence transformations).
4.4. Functions

A short description of the main functions the user may call is as follows:

- **abelian(gens)**: Checks if the generators gens span an Abelian Lie algebra;
- **commutatorTable(gens)**: Returns the commutator table of the generators gens;
- **essentialpars(gens,vars)**: Takes a list of generators gens of a multiparameter Lie group of transformations for the variables vars, and returns the generators which are not linearly independent;
- **generatealgebra(k)**: Once jetorder, xvar and uvar have been properly assigned, this function returns a list of generators spanning the algebra of isometries (for \( k = 1 \)), affine algebra (for \( k = 2 \)), projective algebra (for \( k = 3 \));
- **inverselie(k)**: Computes all the minors of order \( k \) of the jetorder-th distribution generated by the list of vector fields contained in generators;
- **liebracket(gen1,gen2)**: Returns the Lie bracket of the generators gen1 and gen2;
- **lieordering(lis,ind)**: Returns a list of the elements in lis reordered according to the permutation ind of the integers \( \{1, \ldots, n\} \), where \( n \) is the length of list lis;
- **nonzeroocommutators(gens)**: Returns nzcomm, a list of non-zero commutators of generators gens;
- **offprintcrack()**: Prevents reliesolve() to display the steps needed for solving determining equations (this is the default configuration);
- **onprintcrack()**: Sets a variable used in CRACK package (in turn used in the function reliesolve()) in order to display the steps needed for solving determining equations;
- **reliedistrib()**: Returns the matrix distribution, i.e., a matrix whose rows are the prolonged vector fields evaluated in the list splitsymmetries (computed by the function reliesgen(), or by the function generatealgebra(), or suitably assigned by the user);
- **reliegen(k,lis)**: Returns the list generators; \( k \) is an integer (less or equal to the length of symmetries) and lis a list that can be empty; if lis is made by as many values as the number of arbitrary constants occurring in symmetries, generators consists of a list of vector fields, where each vector field is obtained replacing the \( i \)-th parameter by the \( i \)-th element in the list lis (or 1 if lis is empty or its length is different from the number of arbitrary constants entering symmetries) and the other parameters are replaced by 0; if the list lis is \( \{-1\} \), then generators is a list with only one element containing the components of the infinitesimals in their general form, i.e., the linear combinations of all admitted generators; the function produces also the list splitsymmetries whose \( k \)-th element is a list containing the infinitesimals corresponding to the \( k \)-th element of generators;
- **relieinitt()**: If input data have been correctly defined, the function initializes the objects for doing the computation;
- **relieinv()**: Computes the invariance conditions; returns invcond;
- **relieprol()**: Returns the prolongation of a general vector field;
- **reliesolve()**: Solves the determining equations for the infinitesimals, and returns the list symmetries; in group classification problems (but also in the case of conditional symmetries), the list symmetries may contain different solutions for the infinitesimals according to the values of freepars; as a default reliesolve() does not display the steps made to obtain the solution of determining equations; the user can see these steps by calling onprintcrack(); this is suggested when reliesolve() seems to use too much time to complete its execution; the list symmetries contains a list of the sets of solutions of determining equations; each element of this list in turn is a list of four elements: the first one is a list of conditions (possibly empty) that remained unsolved; the second one is a list giving the solution to the determining equations, i.e., the
expressions of the infinitesimals; the third one is a list containing the parameters involved in the solution; the fourth one is a list of expressions which can not vanish (this list can be empty);

- **solvable(gens)**: Checks if the generators gens span a solvable Lie algebra;
- **testrank(gens)**: Returns the rank of the jetorder-th distribution generated by the generators gens.

Notice that ReLie contains some companion functions, used by the main procedures. Here we list some of them that can be useful in interactive sessions.

- **allcoeffs(lis1,lis2)**: Returns the list of coefficients of lis1 (list of polynomials) with respect to the variables in list lis2;
- **bincoeff(n,k)**: Returns \( \binom{n}{k} \);
- **combnorep(n,k)**: Returns the combinations without repetition of \( k \) elements chosen in \( \{1,2,...,n\} \);
- **combrep(n,k)**: Returns the combinations with repetition of \( k \) elements chosen in \( \{1,2,...,n\} \);
- **delzero(lis)**: Returns a list containing all non-zero elements in the list lis;
- **dependence(lis1,lis2)**: Declares that the elements in the list lis1 depend on the variables in the list lis2;
- **dlie(obj,var)**: Returns the usual Lie derivative of obj with respect to var;
- **dlieapprox(obj,var)**: Returns the Lie derivative of obj with respect to var in the context of approximate symmetries;
- **dliestar(obj,var)**: Returns the additional Lie derivative used for equivalence transformations;
- **kroneckerdelta(k1,k2)**: Returns 1 if \( k_1 = k_2 \), 0 otherwise;
- **letterlist(obj,n)**: obj is a symbol, n a positive integer; for instance, letterlist(x,4) builds the list \( \{x_1, x_2, x_3, x_4\} \);
- **letterlistvar(obj,lis)**: obj is a symbol, lis a list; for instance, letterlistvar(xi_,{x,y}) builds the list \( \{xi_x, xi_y\} \);
- **listletter(lis,ch)**: lis is a list, ch a symbol; for instance, listletter({u_,v_},x) builds the list \( \{u_x, v_x\} \);
- **membership(elem,lis)**: Returns the number of occurrences of the element elem in the list lis;
- **minors(m,k)**: m is a matrix and k is a positive integer: Returns the list of minors of order \( k \) of the matrix \( m \);
- **nodependence(lis1,lis2)**: Removes the dependence of the objects in the list lis1 upon the variables in the list lis2;
- **removeelement(lis,elem)**: Removes the element elem from the list lis;
- **removemultiple(lis1,lis2)**: Removes from the list lis1 the elements of the list lis2;
- **scalarmult(obj,lis)**: Returns a list whose \( k \)-th element is the product of the scalar obj and the \( k \)-th element of list lis;
- **scalarproduct(lis1,lis2)**: Returns the sum of the products element by element of two lists with the same number of elements;
- **sumlist(lis1,lis2)**: Returns a list summing element by element the two lists with the same length;
- **zerolist(n)**: Returns a list of \( n \) zeros, the empty list if \( n \leq 0 \).

5. Conclusions

The program ReLie allows the user to perform almost automatically much of the computations needed for investigating ordinary and partial differential equations by means of Lie group methods. The package can also be used in interactive computations where some special assumptions need to be made. Remarkably, the program works in a CAS like Reduce which is open source and freely available for all operating systems. The various routines of ReLie have been tested along the years in many different situations.
and hopefully are sufficiently reliable. At the url http://mat521.unime.it/oliveri (accessed on 9 August 2021) the source code of the package, together with the user’s manual, can be found.

We illustrated the use of the package by choosing some rather simple examples in different areas of Lie symmetry analysis of differential equations; nevertheless, this does not mean that only simple problems can be faced. In fact, ReLie proves effective also when one has to investigate systems of partial differential equations that require to deal with very long expressions. As a last remark, we observe that this is the first program able to compute approximate Lie symmetries of differential equations containing small terms according to the theory recently proposed [60], as well as approximate Q-conditional symmetries [113,114].

Possible future developments could consist in adding functions for computing canonical variables associated to a Lie generator (useful for instance to lower the order of an ordinary differential equation), or computing differential invariants, or deriving the representation of invariant solutions and the corresponding reduced equations (some of these problems could be faced interfacing ReLie with other packages, like APPLYSYM, CDE, CDIFF, LIEPDE, that are present in the current distribution of Reduce [87], as well as procedures for computing higher order or nonlocal symmetries [47]. A further project consists in developing a package for classifying Lie algebras and determining optimal systems of Lie subalgebras.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** The study did not involve humans or animals.

**Informed Consent Statement:** The study did not involve humans.

**Data Availability Statement:** The study did not report any data.

**Acknowledgments:** Work supported by G.N.F.M. of “Istituto Nazionale di Alta Matematica”. The author is grateful to colleagues and friends that used along the years part of this program in their research activity so contributing to fix bugs; in particular, the author warmly thanks Matteo Gorgone for many useful suggestions and criticisms. The author thanks the anonymous referees for their comments, useful to improve the quality and clarity of the paper.

**Conflicts of Interest:** The author declares no conflict of interest.

**References**

1. Lie, S.; Engel, F. *Theorie der Transformationsgruppen*; Teubner: Leipzig, Germany, 1888.
2. Lie, S. *Vorlesungen über Differentialgleichungen mit Bekannten Infinitesimalen Transformationen*; Teubner: Leipzig, Germany, 1891.
3. Ovsiannikov, L.V. *Group Analysis of Differential Equations*; Academic Press: New York, NY, USA, 1982.
4. Ibragimov, N.H. *Transformation Groups Applied to Mathematical Physics*; Reidel Publishing Company: Dordrecht, The Netherlands, 1985.
5. Olver, P.J. *Applications of Lie Groups to Differential Equations*; Springer: New York, NY, USA, 1986.
6. Bluman, G.W.; Kumei, S. *Symmetries and Differential Equations*; Springer: New York, NY, USA, 1989.
7. Stephani, H. *Differential Equations. Their Solutions Using Symmetries*; Cambridge University Press: Cambridge, UK, 1989.
8. Ibragimov, N.H. (Ed.) *Symmetries, Exact Solutions, and Conservation Laws*. In *CRC Handbook of Lie Group Analysis of Differential Equations*; CRC Press: Boca Raton, FL, USA, 1994; Volume 1.
9. Ibragimov, N.H. (Ed.) *Applications in Engineering and Physical Sciences*. In *CRC Handbook of Lie Group Analysis of Differential Equations*; CRC Press: Boca Raton, FL, USA, 1995; Volume 2.
10. Ibragimov, N.H. (Ed.) *New Trends in Theoretical Developments and Computational Methods*. In *CRC Handbook of Lie Group Analysis of Differential Equations*; CRC Press: Boca Raton, FL, USA, 1996; Volume 3.
11. Olver, P.J. *Equivalence, Invariants, and Symmetry*; Cambridge University Press: Cambridge, UK, 1995.
12. Baumann, G. *Symmetry Analysis of Differential Equations with Mathematica*; Springer: New York, NY, USA, 2000.
13. Hydon, P.E. *Symmetry Methods for Differential Equations. A Beginner’s Guide*; Cambridge University Press: Cambridge, UK, 2000.
14. Cantwell, B.J. *Introduction to Symmetry Analysis*; Cambridge University Press: Cambridge, UK, 2002.
15. Bluman, G.W.; Anco, S.C. *Symmetry and Integration Methods for Differential Equations*; Springer: New York, NY, USA, 2002.
16. Meleshko, S.V. *Methods for Constructing Exact Solutions of Partial Differential equations*; Springer: New York, NY, USA, 2005.
17. Bluman, G.W.; Cheviakov, A.F.; Anco, S.C. *Applications of Symmetry Methods to Partial Differential Equations*; Springer: New York, NY, USA, 2009.
18. Bordag, L.A. Geometrical Properties of Differential Equations. Applications of Lie Group Analysis in Financial Mathematics; World Scientific Publishing; Singapore, 2015.
19. Bluman, G. A reduction algorithm for an ordinary differential equation admitting a solvable Lie group. SIAM J. Appl. Math. 1990, 50, 1689–1705. [CrossRef]
20. Olver, P.J.; Rosenau, P. The construction of special solutions to partial differential equations. Phys. Lett. 1986, 144A, 107–112. [CrossRef]
21. Olver, P.J.; Rosenau, P. Group invariant solutions of differential equations. SIAM J. Appl. Math. 1987, 47, 263–278. [CrossRef]
22. Rogers, C.; Ames, W.F. Nonlinear Boundary Value Problems in Science and Engineering; Vol. 183, Mathematics in Science and Engineering; Academic Press: San Diego, CA, USA, 1989.
23. Oliveri, F.; Speciale, M.P. Exact solutions to the equations of ideal gas–dynamics by means of the substitution principle. Int. J. Non-Linear Mech. 1998, 33, 585–592. [CrossRef]
24. Oliveri, F.; Speciale, M.P. Exact solutions to the equations of perfect gases through Lie group analysis and substitution principles. Int. J. Non-Linear Mech. 1999, 34, 1077–1087. [CrossRef]
25. Oliveri, F.; Speciale, M.P. Exact solutions to the unsteady equations of perfect gases through Lie group analysis and substitution principles. Int. J. Non-Linear Mech. 2002, 37, 257–274. [CrossRef]
26. Olver, P.J. Geometric foundations of numerical algorithms and symmetry. Appl. Anal. Discret. Math. 2007, 1, 150–171.
81. Butcher, J.; Carminati, J.; Vu, K.T. A comparative study of some computer algebra packages which determine the Lie point symmetries of differential equations. *Comput. Phys. Commun.* **2003**, *155*, 92–114. [CrossRef]

82. Cheviakov, A.F. GeM software package for computation of symmetries and conservation laws of differential equations. *Comput. Phys. Commun.* **2007**, *176*, 48–61. [CrossRef]

83. Steeb, W.H. *Continuous Symmetries, Lie algebras, Differential Equations and Computer Algebra*, 2nd ed.; World Scientific Publishing: Singapore, 2007.

84. Cheviakov, A.F. Symbolic computation of local symmetries of nonlinear and linear partial and ordinary differential equations. *Math. Comput. Sci.* **2010**, *4*, 203–222. [CrossRef]

85. Rocha Filho, T.M.; Figueiredo, A. [SADE] a Maple package for the symmetry analysis of differential equations. *Comput. Phys. Commun.* **2010**, *182*, 467–476. [CrossRef]

86. Jefferson, G.F.; Carminati, J. ASP: Automated symbolic computation of approximate symmetries of differential equations. *Comput. Phys. Commun.* **2013**, *184*, 1045–1063. [CrossRef]

87. Hearn, A.C; Schöpf, R. Reduce User’s Manual. Free Version. 2021. Available online: https://reduce-algebra.sourceforge.io/ (accessed on 9 August 2021).

88. Saunders, D.J. *The Geometry of Jet Bundles*; Cambridge University Press: Cambridge, UK, 1989.

89. de Graaf, W.A. *Lie Algebras: Theory and Algorithms*; North-Holland Mathematical Library 56; Elsevier: Amsterdam, The Netherlands, 2000.

90. Erdmann, K.; Wildon, M.J. *Introduction to Lie Algebras*; Springer Undergraduate Mathematics Series; Springer: New York, NY, USA, 2006.

91. Clarkson, P.A.; Kruskal, M. New similarity reductions of the Boussinesq equation. *J. Math. Phys.* **1989**, *30*, 2201–2213. [CrossRef]

92. Levi, D.; Winternitz, P. Nonclassical symmetry reduction: Example of the Boussinesq equation. *J. Phys. A Math. Gen.* **1989**, *22*, 2915–2924. [CrossRef]

93. Nucci, M.C.; Clarkson, P.A. The nonclassical method is more general than the direct method for symmetry reductions. An example of the FitzHugh–Nagumo equation. *Phys. Lett.* **1992**, *164*, 49–56. [CrossRef]

94. Arrigo, D.J.; Broadbridge, P.; Hill, J.M. Nonclassical symmetry solutions and the methods of Bluman–Cole and Clarkson–Kruskal. *J. Math. Phys.* **1993**, *34*, 4692–4703. [CrossRef]

95. Nucci, M.C. Nonclassical symmetries and Bäcklund transformations. *J. Math. Anal. Appl.* **1993**, *178*, 294–300. [CrossRef]

96. Clarkson, P.A.; Mansfield, E.L. Symmetry reductions and exact solutions of nonlinear heat equations. *Physica D* **1994**, *10*, 250–288. [CrossRef]

97. Saccomandi, G. A personal overview on the reduction methods for partial differential equations. *Note di Matematica* **2004**, *23*, 217–248.

98. Cherniha, R. New Q–conditional symmetries and exact solutions of some reaction–diffusion–convection equations arising in mathematical biology. *J. Math. Anal. Appl.* **2007**, *326*, 783–799. [CrossRef]

99. Cherniha, R.; Pliukhin, O. New conditional symmetries and exact solutions of reaction–diffusion systems with power diffusivities. *J. Phys. A Math. Theor.* **2008**, *41*, 185208. [CrossRef]

100. Cherniha, R. Conditional symmetries for systems of PDEs: New definitions and their application for reaction–diffusion systems. *J. Phys. A Math. Theor.* **2010**, *43*, 405207. [CrossRef]

101. Cherniha, R.; Davydovych, V. Conditional symmetries and exact solutions of nonlinear reaction–diffusion systems with non–constant diffusivities. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 3177–3188. [CrossRef]

102. Eisenhart, L.P. Contact transformations. *Ann. Math.* **1928**, *30*, 211–249. [CrossRef]

103. Kushner, A.; Lychagin, V.; Rubtsov, V. *Contact Geometry and Nonlinear Differential Equations*; Cambridge University Press: Cambridge, UK, 2007.

104. Hairer, E.; Lubich, C.; Wanner, G. *Geometric Numerical Integration*; Springer: New York, NY, USA, 2002.

105. Iserles, A.; Munthe-Kaas, H.; Nørsett, S.; Zanna, A. Lie–group methods. *Acta Numer.* **2000**, *9*, 215–365. [CrossRef]

106. Marsden, J.E.; Ratiu, T.; Abraham, R. *Manifolds, Tensor Analysis and Applications*; Springer: New York, NY, USA, 2001.

107. Boyer, T.H. Continuous symmetries and conserved currents. *Ann. Phys.* **1967**, *42*, 445–466. [CrossRef]

108. Anco, S.C.; Bluman, G.W. Direct construction method for conservation laws of partial differential equations. Part I: Examples of conservation law classifications. *Eur. J. Appl. Math.* **2002**, *13*, 545–566. [CrossRef]

109. Anco, S.C.; Bluman, G.W. Direct construction method for conservation laws of partial differential equations. Part II: General treatment. *Eur. J. Appl. Math.* **2002**, *13*, 567–585. [CrossRef]

110. Ibragimov, N.I. A new conservation theorem. *J. Math. Anal. Appl.* **2007**, *333*. [CrossRef]

111. Ibragimov, N.H. Nonlinear self–adjointness and conservation laws. *J. Phys. A Math. Theor.* **2011**, *44*, 432002. [CrossRef]

112. Nayfeh, A.H. *Introduction to Perturbation Techniques*; Wiley: New York, NY, USA, 1981.

113. Gorgone, M.; Oliveri, F. Approximate Q-conditional symmetries of partial differential equations. *Electron. J. Differ. Equ.* **2018**, *25*, 133–147.

114. Gorgone, M.; Oliveri, F. Consistent approximate Q-conditional symmetries of PDEs: application to a hyperbolic reaction-diffusion-convection equation. *Z. Angew. Math. Phys.* **2021**, *72*, 119. [CrossRef]

115. Oliveri, F; Speciale, M.P. Equivalence transformations of quasilinear first order systems and reduction to autonomous and homogeneous form. *Acta Appl. Math.* **2012**, *122*, 447–460. [CrossRef]
116. Oliveri, F.; Speciale, M.P. Reduction of balance laws to conservation laws by means of equivalence transformations. *J. Math. Phys.* 2013, 54, 041506. [CrossRef]

117. Gorgone, M.; Oliveri, F.; Speciale, M.P. Reduction of balance laws in $3 + 1$ dimensions to autonomous conservation laws by means of equivalence transformations. *Acta Appl. Math.* 2014, 132, 333–345. [CrossRef]

118. Meleshko, S.V. Generalization of the equivalence transformations. *Nonlinear Math. Phys.* 1996, 3, 170–174. [CrossRef]

119. Manno, G.; Oliveri, F.; Vitolo, R. On differential equations characterized by their Lie point symmetries. *J. Math. Anal. Appl.* 2007, 332, 767–786. [CrossRef]

120. Manno, G.; Oliveri, F.; Vitolo, R. Differential equations uniquely determined by algebras of point symmetries. *Theor. Math. Phys.* 2007, 151, 843–850. [CrossRef]

121. Manno, G.; Oliveri, F.; Saccomandi, G.; Vitolo, R. Ordinary differential equations described by their Lie symmetry algebra. *J. Geom. Phys.* 2014, 85, 2–15. [CrossRef]

122. Gorgone, M.; Oliveri, F. Lie remarkable partial differential equations characterized by Lie algebras of point symmetries. *J. Geom. Phys.* 2019, 144, 314–323. [CrossRef]

123. Blasius, H. Grenzschichten in Flüssigkeiten mit kleiner Reibung. *Z. Angew. Math. Phys.* 1908, 56, 1–37.

124. Oliveri, F. Lie symmetries of differential equations: Direct and inverse problems. *Note di Matematica* 2004, 23, 195–216.

125. Oliveri, F. On the similarity solutions of the KdV equation with variable coefficients. *Int. J. Non-Linear Mech.* 1987, 22, 467–474. [CrossRef]