ReLie User's Manual

Francesco Oliveri

MIFT Department, University of Messina
viale F. Stagno d'Alcontres 31, Messina, Italy
E-mail: foliveri@unime.it
http://mat521.unime.it/oliveri

Contents

1 Introduction

Lie symmetry analysis and its modern generalizations and extensions provide a general theoretical framework for investigating ordinary and partial differen-

tial equations. The theory is completely algorithmic even if it usually involves lengthy computations. For this reason, many computer algebra packages have been developed along the years to automate the computation.

A lot of programs, designed for automatically or interactively performing the Lie group analysis of a (system of) differential equation(s), are currently available, so one could ask: why another program? Essentially there are two reasons. The first one is historical: the author developed in 1994 some routines useful to manage the lengthy expressions needed to determine the Lie point symmetries of differential equations; this set of routines constantly grew through the years providing new capabilities, and now constitutes a package able to perform almost automatically much of the work. The second reason lies on the need of having a program that can be used also in interactive mode using a nice and light (at least in the author's opinion) notation. In fact, the program can be used also in interactive mode mimicking the steps one has to apply 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.

More in detail, ReLie is a Reduce program that may be useful for investigating Lie group properties of differential equations. Every potential user of ReLie is expected to know what Lie point symmetries of differential equations represent [27, 26, 14, 15, 16, 2, 3, 1, 22], so no detail of the underlying theory will be given. Also, it will be assumed that the user has a minimum experience in working with the algebraic mode of Reduce [11]. Thus, in this document only the necessary details of the use of ReLie will be presented.

The routines contained in ReLie allow the user to easily compute (exact and approximate, the latter with the approach proposed in [8]) Lie point symmetries and conditional symmetries as well as contact transformations, variational symmetries and equivalence transformations for classes of differential equations containing arbitrary elements.

Reduce is a general purpose computer algebra system whose development started by Anthony Hearn. It is written in a Lisp dialect (Portable Standard Lisp); since December 2008 it is an open source program (available for all platforms at the url `http://www.reduce-algebra.com`). Reduce is designed as an interactive system, but naturally it can also operate in a batch processing.

In Section 2, we illustrate how to use the package ReLie by considering some examples. In Section 3, we list alphabetically the global variables used by ReLie, and describe the main functions the user may call. In Section , we show how ReLie can be used interactively.

For further details, one can refer to the paper [23] and references therein quoted.

2 The program ReLie

The application of Lie's group theory (together its generalization) to differential equations usually involves a lot of lengthy and cumbersome computations, so that the implementation of the algorithms in Computer Algebra Systems (CAS) is highly desired [29, 30, 10, 12, 31, 16, 13, 1, 5, 33, 4, 6, 32, 7, 28, 17].

The package ReLie is a collection of (algebraic) routines that allow 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 of 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 [20, 19, 18, 9]). 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 available 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) - 2020
Last update August 9, 2020
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: 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 1 *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 3 for a list of internal global variables used by RELIE). The value of `jetorder` is **not** bounded by RELIE.

The first function one has to call is `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 `jetorder`, `xvar` and `uvar` have been assigned, then calling `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 `diffeqs` of differential equations and the list `leadders` of leading derivatives. Nevertheless, we can compute the `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 `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 1, the list `allinfinitesimals` is a list of two elements, each element being the list {`xi_t, xi_x, eta_u`} (this redundancy is used in order to have a unified coding for dealing also with other kinds of symmetries, see below). The dependencies of the elements in `allinfinitesimals` upon the independent and dependent variables (the elements in `xvar` and `uvar`, respectively) are automatically set by RELIE.

By calling the RELIE function `relieprol()`, as a result we get the list `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 `prolongation` contains the components of the prolonged vector field up to `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)

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} + u\frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^2 \partial t} = 0 \tag{1}$$

has to be represented as

```
u_t + u_x + u*u_x - u_txx .
```

If we want to compute the Lie symmetries admitted by Benjamin-Bono-Mahoney equation, after loading ReLie, it is sufficient the following code:

```
jetorder:=3 $
xvar:={t,x} $
uvar:={u} $
diffeqs:={u_t+u_x+u*u_x-u_txx} $
leadders:={u_t} $
relieinit() $
relieinv() $
reliedet() $
reliesolve() $
```

Besides defining the independent and dependent variables, and setting `jetorder`, we need yo define the differential equation(s) (the list `diffeqs`) and the derivative(s) to eliminate by solving the differential equations. The function `relieinit()` initializes all the objects necessary for the computation, `relieinv()` computes the invariance condition, `reliedet()` splits the invariance condition providing the determining equations, and `reliesolve()` solves them and gives the infinitesimals.

3 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).

3.1 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 `epsilon`; when looking for approximate symmetries the user has to define the rule `let epsilon**(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 `arbelem` depend on (only for equivalence transformations); if `zorder` is set to 0, the arbitrary elements depend on `xvar` and `uvar`; `zorder` cannot exceed `jetorder`.

3.2 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 `minors(m,k)`, where `m` is a matrix and `k` a positive integer, or by the function `inverselie(k)` that takes the `jetorder`-th distribution of `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 `reliegen()`);
- `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 `reliegen()`); the list `generators` may also been 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()`);

- `splitsymmetries`: list of lists: the k-th element is a list containing the infinitesimals corresponding to the k-th element of `generators` (produced by `reliegen()`); the list `splitsymmetries` may also been 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()`).

3.3 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);
- `solutiondedv`: 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: `relieinit()` 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).

3.4 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`;
- `newordering(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`;
- `nonzerocommutators(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 (used in the function `reliesolve()`) in order to display the steps needed for solving determining equations;
- `relieclaw(k)`: returns `fluxes`, a list of the components of the fluxes of the conservation law corresponding to the `k`-th Lie generator (obtained after calling to `reliegen()`);
- `reliedet()`: splits the invariant conditions providing the list `deteqs` of 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 `reliegen()`, 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`;

- `relieinit()`: 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; returns `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 polynomyals) 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 $\{1, 2, \ldots, n\}$;
- `combrep(n,k)`: returns the combinations with repetition of k elements chosen in $\{1, 2, \ldots, n\}$;
- `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 $k1 = k2$, 0 otherwise;
- `letterlist(obj,n)`: `obj` is a symbol, n a positive integer;
  for instance, `letterlist(x,4)` builds the list $\{x1, x2, x3, x4\}$;
- `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 $n \leq 0$.

4 Case studies

In this Section, we present some case studies for illustrating the use of ReLie; both the code required to start the computation and the log of the Reduce session are shown.

4.1 Symmetries of ODEs

Example 2 (Linear 2nd order ODE) *The symmetries of the linear 2nd order ODE*

$$\frac{d^2u}{dx^2} + u = 0.$$

are computed by using the code

```
off nat $ % Reduce switch to display expressions in an input compatible form
load_package relie $
jetorder:=2 $
xvar:={x} $
uvar:={u} $
diffeqs:={u_xx+u} $
leadders:={u_xx} $
relie(4) $
reliegen(1,{}) $
commutatortable(generators) $
;end;
```

Here is the log of the Reduce session.

```
**********************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

**********************************************************************

Calling relieinit()...$

You may compute exact point symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$
```

```
Calling reliedet()...$

'deteqs' have been computed (determining equations).$

Calling reliesolve()...$

'symmetries' have been computed (infinitesimals).$

'generators' have been computed (vector fields of symmetries).$

'commtable' has been computed (commutator table).$


symmetries;
{
{
{},
{eta_u= - cos(x)*sin(x)*k_6*u - cos(x)*k_2*u**2 - cos(x)*k_8
- sin(x)**2*k_5*u - sin(x)*k_1*u**2 - sin(x)*k_7 + k_4*u,
xi_x=(2*cos(x)*sin(x)*k_5 + 2*cos(x)*k_1*u - 2*sin(x)**2*k_6
- 2*sin(x)*k_2*u + k_3 + k_6)/2},
{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8},
{}
}
}$


generators;
{
{cos(x)*u, - sin(x)*u**2},
{ - sin(x)*u, - cos(x)*u**2},
{1/2,0},
{0,u},
{cos(x)*sin(x), - sin(x)**2*u},
{( - 2*sin(x)**2 + 1)/2, - cos(x)*sin(x)*u},
{0, - sin(x)},
{0, - cos(x)}
}$


commtable;
{{0, 0, ( - vf_2)/2, - vf_1, vf_1, vf_2/2, - vf_4 + vf_5, vf_3 + vf_6},
{0, 0, vf_1/2, - vf_2, 0, vf_1/2, - vf_3 + vf_6, - 2*vf_4 - vf_5},
{vf_2/2, ( - vf_1)/2, 0, 0, vf_6, ( - vf_4 - 2*vf_5)/2, vf_8/2, ( - vf_7)/2},
{vf_1, vf_2, 0, 0, 0, 0, - vf_7, - vf_8},
{ - vf_1,0, - vf_6, 0, 0, - vf_3, vf_7, 0},
{( - vf_2)/2, ( - vf_1)/2, (vf_4 + 2*vf_5)/2, 0, vf_3, 0, vf_8/2, vf_7/2},
{vf_4 - vf_5, vf_3 - vf_6, ( - vf_8)/2, vf_7, - vf_7, ( - vf_8)/2, 0, 0},
{ - (vf_3 + vf_6), 2*vf_4 + vf_5, vf_7/2, vf_8, 0, ( - vf_7)/2,0, 0}}$
```

```
quit;
```

4.2 Symmetries of PDEs

Example 3 (Burgers equation) *To compute the symmetries of Burgers equation,*

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

use the code

```
off nat $
load_package relie $
jetorder:=2 $
xvar:={t,x} $
uvar:={u} $
diffeqs:={u_t+u*u_x-u_xx} $
leadders:={u_t} $
relie(4) $
reliegen(1,{}) $
commutatortable(generators) $
```

The log of the REDUCE *session is below.*

```
*****************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

*****************************************************************

Calling relieinit()...$

You may compute exact point symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$

Calling reliedet()...$

'deteqs' have been computed (determining equations).$
```

```
Calling reliesolve()...$

'symmetries' have been computed (infinitesimals).$

'generators' have been computed (vector fields of symmetries).$

'commtable' has been computed (commutator table).$


symmetries;
{
{
{},
{eta_u= - k_1 - k_2*t*u + k_2*x - k_3*u,
xi_x=( - 2*k_1*t + 2*k_2*t*x + 2*k_3*x - k_5)/2,
xi_t=k_2*t**2 + 2*k_3*t - k_4},
{k_1, k_2, k_3, k_4, k_5},
{}
}
}$


generators;
{
{0, - t,-1},
{t**2,t*x, - t*u + x},
{2*t,x, - u},
{-1,0,0},
{0,( - 1)/2,0}
}$


commtable;
{{0, 0, - vf_1, 2*vf_5, 0},
{0, 0, - 2*vf_2, vf_3, ( - vf_1)/2},
{vf_1, 2*vf_2,0, - 2*vf_4, - vf_5},
{ - 2*vf_5, - vf_3, 2*vf_4, 0, 0},
{0, vf_1/2, vf_5, 0, 0}}$


quit;
```

4.3 Group classification problems

Example 4 (Variable coefficient KdV equation) *Consider the KdV equation with variable coefficients*

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} + \frac{k}{t}u = 0,$$

and look to the admitted symmetries. In the code below, we include the statement `freepars:=k` *in order to check if there are special values of constant k providing different sets of symmetries:*

```
off nat $
load_package relie $
jetorder:=3 $
xvar:={t,x} $
uvar:={u} $
freepars:={k} $
diffeqs:={u_t+u*u_x+u_xxx+k*u/t} $
leadders:={u_xxx} $
relie(4) $
```

The log of the REDUCE *session is as follows (note that* REALIE *computes four solution sets for the infinitesimals according to various choices of k:*

```
**********************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

**********************************************************************

Calling relieinit()...$

You may compute exact point symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$

Calling reliedet()...$

'deteqs' have been computed (determining equations).$

Calling reliesolve()...$

'symmetries' have been computed (infinitesimals).$


length(symmetries);
4$
```

```
symmetries;
{
 {
  {},
  {k=1,
  eta_u=(k_2 + 2*k_3*t*u)/t,
  xi_x=log(t)*k_2 - k_1 - k_3*x,
  xi_t= - 3*k_3*t},
  {k_1,k_2,k_3},
  {}
 },
 {
  {},
  {k=0,
  eta_u=2*k_4*u - k_5,
  xi_x= - k_4*x - k_5*t - k_6,
  xi_t= - 3*k_4*t - k_7},
  {k_4,k_5,k_6,k_7},
  {}
 },
 {
  {},
  {k=1/2,
  eta_u=(2*sqrt(t)*k_4*u + k_10 - 4*k_8*t*u + k_8*x)/sqrt(t),
  xi_x=( - sqrt(t)*k_4*x - sqrt(t)*k_9 + 2*k_10*t + 2*k_8*t*x)/sqrt(t),
  xi_t=( - 3*sqrt(t)*k_4*t + 4*k_8*t**2)/sqrt(t)},
  {k_4,k_8,k_9,k_10},
  {k}
 },
 {
  {},
  {eta_u=(2*t**k*k_4*u + k_12)/t**k,
  xi_x=( - t**k*k*k_11 - t**k*k*k_4*x + t**k*k_11 + t**k*k_4*x - k_12*t)/(t**k*k - t**k),
  xi_t= - 3*k_4*t},
  {k,k_4,k_11,k_12},
  {2*k - 1,k - 1,k}
 }
}$


quit;
```

Remark 1 *The list* `freepars` *may include also functions; in this case the user has to set the dependence of these functions upon their arguments. Furthermore, the user may define the list* `nonzeropars` *of parameters involved in the equations that can not be vanishing.*

Example 5 (3D Gas dynamics equations) *Consider the 3D Euler equations*

of ideal gas dynamics,

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

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

```
load_package relie $
off nat $
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_10*x2+k_12*v2-k_2*v1-k_3*v3-k_8*v2,
   eta_v1=k_10*t*v1-k_10*x1-k_11+k_12*v1+k_2*v2+k_5*v3-k_8*v1,
   eta_rho=3*k_10*rho*t-2*k_12*rho-k_14*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**2-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
```

```
  },
  {}
 },
 {
  {},
  {
   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*v1-k_3*v3-k_8*v2,
   eta_v1=-k_11+k_12*v1+k_2*v2+k_5*v3-k_8*v1,
   eta_rho=-2*k_12*rho-k_14*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_2*x1-k_3*x3-k_4-k_8*x2,
   xi_x1=-k_11*t+k_2*x2+k_5*x3-k_8*x1-k_9,
   xi_t=-k_12*t-k_13
  },
  {
    gamma, k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_11, k_12, k_13, k_14
  },
  {
    gamma, 3*gamma - 5
  }
 }
}
```

In this case, `symmetries` *is a list of 2 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* Γ *different from 0 and* $5/3$*. By calling*

```
reliegen(2,{-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, - x1,0,0,v2, - v1,0,0},
 {0,x3,0, - x1,0,v3,0, - v1,0},
 {0,0,x3, - x2,0,0,v3, - v2,0},
 {0,t,0,0,0,1,0,0,0},
 {0,0,t,0,0,0,1,0,0},
 {0,0,0,t,0,0,0,1,0},
 {t,0,0,0,2*rho,-v1,-v2,-v3,0},
 {0,x1,x2,x3,-2*rho,v1,v2,v3,0},
 {0,0,0,0,rho,0,0,0,p}
```

```
}
```

so that for Γ *unspecified we have a 13–dimensional Lie algebra of point symmetries spanned by*

$$\Xi_1 = \frac{\partial}{\partial t}, \qquad \Xi_{1+k} = \frac{\partial}{\partial x_k}, \qquad \Xi_{4+k} = t\frac{\partial}{\partial x_k} + \frac{\partial}{\partial v_k}, \qquad k = 1, 2, 3,$$

$$\Xi_{5+i+j} = x_j\frac{\partial}{\partial x_i} - x_i\frac{\partial}{\partial x_j} + v_j\frac{\partial}{\partial v_i} - v_i\frac{\partial}{\partial v_j}, \qquad 1 \le i < j \le 3,$$

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

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

Calling

```
reliegen(1,{-1,1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1}) $
generators:=newordering(generators,{13,9,4,6,2,5,3,11,1,7,12,8,14,10}) $
```

we have that for $\Gamma = 5/3$ *there is a 14th symmetry generated by*

$$\Xi_{14} = t^2\frac{\partial}{\partial t} + \sum_{k=1}^{3}\left(x_k t\frac{\partial}{\partial x_k} + (x_k - v_k t)\frac{\partial}{\partial v_k}\right) - 3\rho t\frac{\partial}{\partial \rho} - 5pt\frac{\partial}{\partial p}.$$

4.4 Differential equations with derivatives not in polynomial form

Example 6 *To correctly compute the determining equations for the infinitesimals of the equation*

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

ReLie *needs to know that not all derivatives appear in polynomial form. This case can be faced with the following code*

```
load_package relie $
off nat $
jetorder:=2 $
xvar:={t,x} $
uvar:={u} $
diffeqs:={u_tx+exp(u_x)-u} $
leadders:={u_tx} $
nonpolyders:={u_x} $
relie(4) $
reliegen(1,{-1,-1,-1,-1,-1}) $
commutatortable(generators) $
```

Here is the log of the Reduce *session.*

```
****************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

****************************************************************

Calling relieinit()...$

You may compute exact point symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$

Calling reliedet()...$

'deteqs' have been computed (determining equations).$

Calling reliesolve()...$

'symmetries' have been computed (infinitesimals).$


symmetries;
{{{},
{eta_u=f_1 - k_1*u + k_3*x - k_5,
xi_x= - k_1*x - k_2,
xi_t= - k_3*t - k_4},
{f_1, k_1, k_2, k_3, k_4, k_5},
{}}}$


fargs(f_1);
{t}$

% f_1 is an arbitrary function of t

reliegen(1,{-1,-1,-1,-1,-1});
'generators' have been computed (vector fields of symmetries).$


generators;
```

```
{{0,x,u},
{0,1,0},
{t,0, - x},
{1,0,0},
{0,0,1}}$


commutatortable(generators);
'commtable' has been computed (commutator table).$

{{0, - vf_2,0,0, - vf_5},
{vf_2,0, - vf_5,0,0},
{0,vf_5,0, - vf_4,0},
{0,0,vf_4,0,0},
{vf_5,0,0,0,0}}$


quit;
```

Notice that `reliegen(1,{-1,-1,-1,-1,-1})` *provides a basis for the finite-dimensional Lie algebra only, unless we do not call* `reliegen(1,{-1})`*. In this case, we have only one generator which is the linear combinations of the generators of the finite-dimensional Lie algebra and of the generator corresponding to arbitrary function.*

Example 7 *To compute the Lie point symmetries of the linear heat equation*

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

we need the code

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

The log of the REDUCE *session is below.*

```
**********************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020
```

```
Last update August 9, 2020

http://mat521.unime.it/oliveri/

****************************************************************

Calling relieinit()...$

You may compute exact point symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$

Calling reliedet()...$

'deteqs' have been computed (determining equations).$

Calling reliesolve()...$

'symmetries' have been computed (infinitesimals).$


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},
{f_1, k_1, k_2, k_3, k_4, k_5, k_6},
{}}}$


fargs(f_1);
{t,x}$

% f_1 depends on t and x and is a solution of the linear heat equation

reliegen(1,{-1});
'generators' have been computed (vector fields of symmetries).$

generators;
{{( - 2*k_3*t - k_4*t**2 - 2*k_5)/2,
( - (k_1*t + k_2 + k_3*x + k_4*t*x))/2,
( - 8*f_1 + 2*k_1*u*x + 2*k_4*t*u + k_4*u*x**2 + 2*k_6*u)/8}}$


quit;
```

4.5 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 `nonclassical` and define the list `qcond` (a list of distinct integers chosen in the range from 1 to the number of dependent variables). Suppose that `xvar:={x1,x2,x3}` and `uvar:={u,v}`. The components of the Lie generator of the conditional symmetries are

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

The list `qcond` tells to the program which invariant surface conditions have to be used to restrict the solution manifold:

- `qcond:={1,2}` if the invariant surface conditions of both dependent variables have to be used;
- `qcond:={1}` if the invariant surface condition of the first dependent variable has to be used;
- `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 `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 To compute the conditional symmetries generated by the operator*

$$\Xi = \frac{\partial}{\partial t} + \xi(t,x,u)\frac{\partial}{\partial x} + \eta(t,x,u)\frac{\partial}{\partial u}, \tag{2}$$

of the linear heat equation

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0. \tag{3}$$

the following code can be used.

```
load_package relie $
off nat $
jetorder:=2 $
xvar:={t,x} $
uvar:={u} $
nonclassical:=1 $
qcond:={1} $
diffeqs:={u_t-u_xx} $
leadders:={u_xx} $
relie(4) $
```

The log of the REDUCE *session is as follows.*

```
****************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

****************************************************************

Calling relieinit()...$

You may compute exact conditional symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$

Calling reliedet()...$

'deteqs' have been computed (determining equations).$

Calling reliesolve()...$

'symmetries' have been computed (infinitesimals).$


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},
{}}}$


fargs(f_1);
{t,x}$


fargs(f_2);
{t,x}$
```

```
quit;
```

Therefore, we have:

$$\xi = -\frac{\partial f_2}{\partial x}, \qquad \eta = -f_1 + \frac{1}{2}\left(\frac{\partial f_2}{\partial t} - \frac{\partial^2 f_2}{\partial x^2} - \left(\frac{\partial f_2}{\partial x}\right)^2\right) u,$$

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

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

Remark 2 *When computing conditional symmetries we have to be careful in the choice of* `leadders`*, since also the invariant surface conditions and their differential consequences are solved with respect to some derivatives.*

4.6 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.

```
load_package relie $
off nat $
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_x*x, ( - u_x**2*x)/2, ( - u_x**2)/2},
 {(x*(2*u - u_x*x))/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.

4.7 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 the variable `variational` to 1. The functions that we have to call do not change, as the following examples show.

Example 10 *The code*

```
load_package relie $
off nat $
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

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

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 ϕ 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 ϕ 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*

$$\frac{1}{6}\left(t^2\left(tu^6 + 3t\left(\frac{du}{dt}\right)^2 + 3u\frac{du}{dt}\right)\right) = \textit{constant}.$$

Example 11 (Klein–Gordon equation (see [3])) *The class of Klein–Gordon wave equations,*

$$\frac{\partial^2 u}{\partial t \partial x} + g(u) = 0 \tag{4}$$

with a general nonlinear interaction term $g(u)$, can be derived from a variational principle given by the action functional with Lagrangian

$$\mathcal{L} = -\frac{1}{2}\frac{\partial u}{\partial t}\frac{\partial u}{\partial x} + h(u), \qquad h'(u) = g(u). \tag{5}$$

For an arbitrary $g(u)$, the generators of these symmetries, are

$$\Xi_1 = t\frac{\partial}{\partial t} - x\frac{\partial}{\partial x}, \qquad \Xi_2 = \frac{\partial}{\partial t}, \qquad \Xi_3 = \frac{\partial}{\partial x}, \tag{6}$$

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:

$$\begin{aligned}
&\frac{D}{Dt}\left(\frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^2\right) + \frac{D}{Dx}\left(h(u)\right) = 0,\\
&\frac{D}{Dt}\left(h(u)\right) + \frac{\partial}{\partial x}\left(\frac{1}{2}\left(\frac{\partial u}{\partial t}\right)^2\right) = 0,\\
&\frac{D}{Dt}\left(\frac{1}{2}x\left(\frac{\partial u}{\partial x}\right)^2 - th(u)\right) + \frac{D}{Dx}\left(-\frac{1}{2}t\left(\frac{\partial u}{\partial t}\right)^2 + xh(u)\right) = 0.
\end{aligned} \tag{7}$$

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.*

4.8 Computation of approximate symmetries

Approximate Lie point symmetries of differential equations containing small terms are computed according to the approach described in [8]. The small parameter must be denoted by `epsilon`. In addition to the inputs described before, the user has to define the integer `approxorder` for the order of approximation,

and a rule, say `let epsilon**(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*

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} - \varepsilon\frac{\partial^2 u}{\partial x^2} = 0 \tag{8}$$

```
load_package relie $
off nat $
jetorder:=3 $
approxorder:=1 $
let epsilon^2 = 0 $
xvar:={t,x} $
uvar:={u} $
diffeqs:={u_t+u*u_x+u_xxx-epsilon*u_xx} $
leadders:={u_t} $
relie(4) $
```

and the log of the REDUCE *session follows.*

```
****************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

****************************************************************

Calling relieinit()...$

You may compute approximate point symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$

Calling reliedet()...$

'deteqs' have been computed (determining equations).$

Calling reliesolve()...$
```

```
'symmetries' have been computed (infinitesimals).$


symmetries;
{{{},
{eta1_u=k_3 - k_4*u0,
eta0_u= - k_1,
xi1_x=(6*k_3*t + 3*k_4*x - 2*k_6)/6,
xi1_t=(3*k_4*t - 2*k_5)/2,
xi0_x= - k_1*t - k_2,
xi0_t= - k_7},
{k_1, k_2, k_3, k_4, k_5, k_6, k_7},
{}}}$


reliegen(1,{-1,-1,-1,-1,-1,-1,-1});
'generators' have been computed (vector fields of symmetries).$


generators;
{{0,t,1},
{0,1,0},
{0, - epsilon*t, - epsilon},
{( - 3*epsilon*t)/2,( - epsilon*x)/2,epsilon*u0},
{epsilon,0,0},
{0,epsilon/3,0},
{1,0,0}}$

quit;
```

so that a basis of the approximate Lie algebra of symmetries of equation (8) *is (after a reordering)*

$$\begin{aligned}
&\Xi_1 = \frac{\partial}{\partial t}, \qquad \Xi_2 = \frac{\partial}{\partial x}, \qquad \Xi_3 = t\frac{\partial}{\partial x} + \frac{\partial}{\partial u},\\
&\Xi_4 = \varepsilon\left(t\frac{\partial}{\partial t} + \frac{x}{3}\frac{\partial}{\partial x} - \frac{2}{3}u_0\frac{\partial}{\partial u}\right), \qquad \Xi_5 = \varepsilon\left(t\frac{\partial}{\partial x} + \frac{\partial}{\partial u}\right) = \varepsilon\Xi_3, \qquad (9)\\
&\Xi_6 = \varepsilon\frac{\partial}{\partial t} = \varepsilon\Xi_1, \qquad \Xi_7 = \varepsilon\frac{\partial}{\partial x} = \varepsilon\Xi_2.
\end{aligned}$$

Remark 3 *If the differential equation involves some of the dependent variables or some of their derivatives not in explicit and definite form the user needs to expand these functions in power series of ε up to the desired order of approximation. For instance, to compute the first order approximate symmetries of the equation*

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

we have to write

```
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 ε, but this is not strictly required.

Example 13 *The following code computes the first order approximate symmetries of the equation*

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

```
load_package relie $
off nat $
jetorder:=2 $
xvar:={t,x} $
uvar:={u} $
depend f,u0 $
freepars:={f} $:
nonzeropars:={f,df(f,u0)} $ % f is not zero and not a constant
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} $
leadders:={u_tt} $
approxorder:=1 $
let epsilon^2 = 0 $
relie(4) $
```

Here is the log of the REDUCE *session.*

```
*****************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

*****************************************************************

Calling relieinit()...$

You may compute approximate point symmetries.$

Calling relieinv()...$

'invcond' has been computed (invariance conditions).$
```

```
Calling reliedet()...$

'deteqs' have been computed (determining equations).$

Calling reliesolve()...$

'symmetries' have been computed (infinitesimals).$


length(symmetries);
53$


symmetries;
{{{},
{f=k_9/u0**((4*k_8)/(k_1 + k_8)),
eta1_u=( - k_1*t*u0 - k_8*t*u0)/2,
eta0_u=( - k_1**2*u0 + k_1*k_6*u0 - k_1*k_8*u0 + k_6*k_8*u0)/(2*k_8),
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x= - k_6*x - k_7,
xi0_t= - k_1*t + k_3 + k_4},
{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9},
{f, k_9, k_1 + k_8, k_8, k_1 - k_6}},
{{df(f,u0)*k_10*k_8*u0 + 2*df(f,u0)*k_11**2 + df(f,u0)*k_11*k_6*u0 + df(f,u0)*
k_11*k_8*u0 + 4*f*k_11*k_8},
{eta1_u=( - k_10*k_8*t*u0 - 2*k_11**2*t - k_11*k_6*t*u0 - k_11*k_8*t*u0)/(2*k_11
),
eta0_u=( - k_10**2*k_8*u0 - 2*k_10*k_11**2 - k_10*k_11*k_6*u0 - k_10*k_11*k_8*u0
)/(2*k_11**2),
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x= - k_6*x - k_7,
xi0_t=( - k_10*k_8*t + k_11*k_3 + k_11*k_4 - k_11*k_6*t)/k_11},
{f, k_2, k_3, k_4, k_5, k_6, k_7, k_10, k_8, k_11},
{k_10, k_11, f, df(f,u0), k_10*k_8*u0 + 2*k_11**2 + k_11*k_6*u0 + k_11*k_8*u0, k_8}},
{{},
{f=k_14/(2*k_12 + k_13*u0)**(4/3),
eta1_u=( - 6*k_12*k_8*t - 3*k_13*k_8*t*u0)/(2*k_13),
eta0_u=( - 2*k_12*k_13*x + 6*k_12*k_6 - 12*k_12*k_8 - k_13**2*u0*x + 3*k_13*k_6*
u0 - 6*k_13*k_8*u0)/(2*k_13),
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_6*x - 6*k_7)/6,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_12, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_13, k_14},
{k_8, k_13, f, k_14, 2*k_12 + k_13*u0, k_6 - 2*k_8}},
{{df(f,u0)*k_1*u0 + 2*df(f,u0)*k_11 + df(f,u0)*k_8*u0 + 4*f*k_8},
{eta1_u=( - k_1*t*u0 - 2*k_11*t - k_8*t*u0)/2,
```

```
eta0_u=0,
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x= - k_1*x - k_7,
xi0_t= - k_1*t + k_3 + k_4},
{f, k_1, k_8, k_2, k_11, k_3, k_4, k_5, k_7},
{f, df(f,u0), k_1*u0 + 2*k_11 + k_8*u0}},
{{},
{f=k_15/(2*k_12 + k_13*u0)**(4/3),
eta1_u=( - 6*k_12*k_8*t - 3*k_13*k_8*t*u0)/(2*k_13),
eta0_u=( - 2*k_12*x - k_13*u0*x)/2,
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_7 - 12*k_8*x)/6,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_12, k_2, k_3, k_4, k_5, k_7, k_8, k_13, k_15},
{2*k_12 + k_13*u0, k_15, f, k_13, k_8}},
{{},
{f=k_19/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_8*t*u0)/2,
eta0_u=(3*k_16*u0 - 6*k_8*u0)/2,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x= - k_16*x - k_17,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_2, k_3, k_4, k_5, k_16, k_17, k_18, k_8, k_19},
{k_16 - 2*k_8, f, k_19, k_18, k_8, c_37}},
{{},
{f=k_21/(2*k_11 + 3*k_8*u0)**(4/3),
eta1_u=( - 2*k_11**2*t - 2*k_11*k_20*x - 3*k_11*k_8*t*u0 - 3*k_20*k_8*u0*x)/(2*k_11),
eta0_u=( - 2*k_10*k_11 - 3*k_10*k_8*u0)/(2*k_11),
xi1_x=( - 2*k_11*k_4*x - 2*k_11*k_5 + k_20*k_8*x**2)/(2*k_11),
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x=(k_10*k_8*x - k_11*k_17 - 2*k_11*k_8*x)/k_11,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_2, k_3, k_4, k_5, k_17, k_10, k_20, k_8, k_11, k_21},
{f, k_21, k_10, k_8, k_11, k_20, 2*k_11 + 3*k_8*u0}},
{{},
{f=k_22/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_8*t*u0)/2,
eta0_u=( - k_13*u0*x + 3*k_16*u0 - 6*k_8*u0)/2,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_16*x - 6*k_17)/6,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_2, k_3, k_4, k_5, k_16, k_17, k_8, k_13, k_18, k_22},
{k_16 - 2*k_8, f, k_22, k_13, k_18, k_8, k_1}},
{{},
{f=k_23/(k_18*u0 + 2*k_20)**(4/3),
eta1_u=( - k_11*k_18*t*u0 - 2*k_11*k_20*t - k_18*k_20*u0*x - 2*k_20**2*x)/(2*k_20),
```

```
eta0_u=( - 2*k_11*k_18**2*u0 - 4*k_11*k_18*k_20 - k_12*k_18**2*u0*x - 2*k_12*
k_18*k_20*x + 3*k_16*k_18*k_20*u0 + 6*k_16*k_20**2)/(2*k_18*k_20),
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - k_11*k_18*t**2 - 6*k_2*k_20 - 6*k_20*k_4*t)/(6*k_20),
xi0_x=(k_12*k_18*x**2 - 6*k_16*k_20*x - 6*k_17*k_20)/(6*k_20),
xi0_t=( - 2*k_11*k_18*t + 3*k_20*k_3 + 3*k_20*k_4)/(3*k_20)},
{k_2, k_3, k_4, k_5, k_16, k_17, k_18, k_11, k_12, k_20, k_23},
{f, k_23, k_18, k_11, k_18*u0 + 2*k_20, k_20, k_12, c_37, - 2*k_11*k_18 + 3*k_16*k_20}},
{{},
{f=k_24/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_8*t*u0)/2,
eta0_u=0,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x= - k_17 - 2*k_8*x,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_2, k_3, k_4, k_5, k_17, k_18, k_8, k_24},
{k_8, k_18, k_24, f}},
{{},
{f=k_25/(2*k_11 + 3*k_8*u0)**(4/3),
eta1_u=( - 2*k_11**2*t - 2*k_11*k_20*x - 3*k_11*k_8*t*u0 - 3*k_20*k_8*u0*x)/(2*k_11),
eta0_u=0,
xi1_x=( - 2*k_11*k_4*x - 2*k_11*k_5 + k_20*k_8*x**2)/(2*k_11),
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x= - k_17 - 2*k_8*x,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_2, k_3, k_4, k_5, k_17, k_20, k_8, k_11, k_25},
{f, k_25, k_20, k_11, k_8, 2*k_11 + 3*k_8*u0, k_1}},
{{},
{f=k_26/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_8*t*u0)/2,
eta0_u=( - k_13*u0*x)/2,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_4*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_17 - 12*k_8*x)/6,
xi0_t=k_3 + k_4 - 2*k_8*t},
{k_2, k_3, k_4, k_5, k_17, k_8, k_13, k_18, k_26},
{k_1, k_8, k_18, k_13, k_26, f}},
{{},
{f=k_27/(k_18*u0 + 2*k_20)**(4/3),
eta1_u=( - k_11*k_18*t*u0 - 2*k_11*k_20*t - k_18*k_20*u0*x - 2*k_20**2*x)/(2*k_20),
eta0_u=( - k_12*k_18*u0*x - 2*k_12*k_20*x)/(2*k_20),
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - k_11*k_18*t**2 - 6*k_2*k_20 - 6*k_20*k_4*t)/(6*k_20),
xi0_x=( - 4*k_11*k_18*x + k_12*k_18*x**2 - 6*k_17*k_20)/(6*k_20),
xi0_t=( - 2*k_11*k_18*t + 3*k_20*k_3 + 3*k_20*k_4)/(3*k_20)},
{k_2, k_3, k_4, k_5, k_17, k_18, k_11, k_12, k_20, k_27},
{k_12, k_20, k_18*u0 + 2*k_20, k_11, k_18, k_27, f}},
{{},
{f=k_30/(k_18*u0 + 2*k_20)**(4/3),
```

```
eta1_u=( - k_11*k_18*t*u0 - 2*k_11*k_20*t - k_18*k_20*u0*x - k_18*k_29*u0
- 2*k_20**2*x - 2*k_20*k_29)/(2*k_20),
eta0_u=( - k_12*k_18*u0*x - 2*k_12*k_20*x)/(2*k_20),
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - k_11*k_18*t**2 - 2*k_18*k_29*t - 6*k_2*k_20 - 6*k_20*k_4*t)/(6*k_20),
xi0_x=( - 4*k_11*k_18*x + k_12*k_18*x**2 - 6*k_20*k_28)/(6*k_20),
xi0_t=( - 2*k_11*k_18*t - 2*k_18*k_29 + 3*k_20*k_3 + 3*k_20*k_4)/(3*k_20)},
{k_2, k_3, k_4, k_5, k_28, k_29, k_18, k_11, k_12, k_20, k_30},
{f, k_30, k_18, k_11, k_18*u0 + 2*k_20, k_20, k_12, k_29}},
{{},
{f=k_32/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_4*u0 - 3*k_8*t*u0)/2,
eta0_u=( - k_13*u0*x)/2,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_28 - 12*k_8*x)/6,
xi0_t=k_3 - 2*k_31 + 3*k_4 - 2*k_8*t},
{k_31, k_2, k_3, k_4, k_5, k_28, k_8, k_13, k_18, k_32},
{k_31 - k_4, f, k_32, k_13, k_18, k_8, k_1}},
{{},
{f=k_33/(2*k_11 + 3*k_8*u0)**(4/3),
eta1_u=( - 2*k_11**2*t - 2*k_11*k_20*x - 2*k_11*k_29 - 3*k_11*k_8*t*u0 - 3*k_20*
k_8*u0*x - 3*k_29*k_8*u0)/(2*k_11),
eta0_u=0,
xi1_x=( - 2*k_11*k_4*x - 2*k_11*k_5 + k_20*k_8*x**2)/(2*k_11),
xi1_t=( - 2*k_11*k_2 - 2*k_11*k_4*t - k_11*k_8*t**2 - 2*k_29*k_8*t)/(2*k_11),
xi0_x= - k_28 - 2*k_8*x,
xi0_t=(k_11*k_3 + k_11*k_4 - 2*k_11*k_8*t - 2*k_29*k_8)/k_11},
{k_2, k_3, k_4, k_5, k_28, k_29, k_20, k_8, k_11, k_33},
{k_1, 2*k_11 + 3*k_8*u0, k_8, k_11, k_20, k_33, f, k_29}},
{{},
{f=k_34/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_4*u0 - 3*k_8*t*u0)/2,
eta0_u=0,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x= - k_28 - 2*k_8*x,
xi0_t=k_3 - 2*k_31 + 3*k_4 - 2*k_8*t},
{k_31, k_2, k_3, k_4, k_5, k_28, k_18, k_8, k_34},
{k_31 - k_4, f, k_34, k_18, k_8}},
{{},
{f=k_36/(k_18*u0 + 2*k_20)**(4/3),
eta1_u=( - k_11*k_18*t*u0 - 2*k_11*k_20*t - k_18*k_20*u0*x - k_18*k_29*u0 - 2*
k_20**2*x - 2*k_20*k_29)/(2*k_20),
eta0_u=( - 2*k_11*k_18**2*u0 - 4*k_11*k_18*k_20 - k_12*k_18**2*u0*x - 2*k_12*
k_18*k_20*x + 3*k_18*k_20*k_35*u0 + 6*k_20**2*k_35)/(2*k_18*k_20),
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - k_11*k_18*t**2 - 2*k_18*k_29*t - 6*k_2*k_20 - 6*k_20*k_4*t)/(6*k_20),
xi0_x=(k_12*k_18*x**2 - 6*k_20*k_28 - 6*k_20*k_35*x)/(6*k_20),
xi0_t=( - 2*k_11*k_18*t - 2*k_18*k_29 + 3*k_20*k_3 + 3*k_20*k_4)/(3*k_20)},
```

```
{k_2, k_3, k_4, k_5, k_35, k_28, k_29, k_18, k_11, k_12, k_20, k_36},
{c_37, k_12, k_20, k_18*u0 + 2*k_20, k_11, k_18, k_36, f,
2*k_11*k_18 - 3*k_20*k_35, k_29}},
{{},
{f=k_37/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_4*u0 - 3*k_8*t*u0)/2,
eta0_u=( - k_13*u0*x + 3*k_35*u0 - 6*k_8*u0)/2,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_28 - 6*k_35*x)/6,
xi0_t=k_3 - 2*k_31 + 3*k_4 - 2*k_8*t},
{k_31, k_2, k_3, k_4, k_5, k_35, k_28, k_8, k_13, k_18, k_37},
{k_31 - k_4, k_1, k_8, k_18, k_13, k_37, f, - k_35 + 2*k_8}},
{{},
{f=k_38/(2*k_11 + 3*k_8*u0)**(4/3),
eta1_u=( - 2*k_11**2*t - 2*k_11*k_20*x - 2*k_11*k_29 - 3*k_11*k_8*t*u0 - 3*k_20*
k_8*u0*x - 3*k_29*k_8*u0)/(2*k_11),
eta0_u=( - 2*k_10*k_11 - 3*k_10*k_8*u0)/(2*k_11),
xi1_x=( - 2*k_11*k_4*x - 2*k_11*k_5 + k_20*k_8*x**2)/(2*k_11),
xi1_t=( - 2*k_11*k_2 - 2*k_11*k_4*t - k_11*k_8*t**2 - 2*k_29*k_8*t)/(2*k_11),
xi0_x=(k_10*k_8*x - k_11*k_28 - 2*k_11*k_8*x)/k_11,
xi0_t=(k_11*k_3 + k_11*k_4 - 2*k_11*k_8*t - 2*k_29*k_8)/k_11},
{k_2, k_3, k_4, k_5, k_28, k_29, k_10, k_20, k_8, k_11, k_38},
{f, k_38, k_29, k_20, k_11, k_8, k_10, 2*k_11 + 3*k_8*u0}},
{{},
{f=k_39/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_4*u0 - 3*k_8*t*u0)/2,
eta0_u=(3*k_35*u0 - 6*k_8*u0)/2,
xi1_x=(k_18*x**2 - 6*k_4*x - 6*k_5)/6,
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x= - k_28 - k_35*x,
xi0_t=k_3 - 2*k_31 + 3*k_4 - 2*k_8*t},
{k_31, k_2, k_3, k_4, k_5, k_35, k_28, k_18, k_8, k_39},
{k_31 - k_4, k_8, k_18, k_39, f, c_37, - k_35 + 2*k_8}},
{{},
{f=k_41/(2*k_12 + k_13*u0)**(4/3),
eta1_u=( - 6*k_12*k_31 + 6*k_12*k_4 - 6*k_12*k_8*t - 3*k_13*k_31*u0
+ 3*k_13*k_4*u0 - 3*k_13*k_8*t*u0)/(2*k_13),
eta0_u=( - 2*k_12*x - k_13*u0*x)/2,
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_40 - 12*k_8*x)/6,
xi0_t=k_3 - 2*k_31 + 3*k_4 - 2*k_8*t},
{k_12, k_31, k_2, k_3, k_4, k_5, k_40, k_8, k_13, k_41},
{k_8, k_13, f, k_41, 2*k_12 + k_13*u0, k_31 - k_4}},
{{df(f,u0)*k_1*u0 + 2*df(f,u0)*k_11 + df(f,u0)*k_8*u0 + 4*f*k_8},
{eta1_u=( - k_1*k_11*t*u0 - k_1*k_29*u0 - 2*k_11**2*t - 2*k_11*k_29 - k_11*k_8*t
*u0 - k_29*k_8*u0)/(2*k_11),
eta0_u=0,
xi1_x= - k_4*x - k_5,
```

```
xi1_t=( - 2*k_11*k_2 - 2*k_11*k_4*t - k_11*k_8*t**2 - 2*k_29*k_8*t)/(2*k_11),
xi0_x= - k_1*x - k_40,
xi0_t=( - k_1*k_11*t - k_1*k_29 + k_11*k_3 + k_11*k_4)/k_11},
{f, k_1, k_2, k_3, k_4, k_5, k_40, k_29, k_8, k_11},
{k_29, k_11, f, df(f,u0), k_1*u0 + 2*k_11 + k_8*u0, k_8}},
{{},
{f=k_43/u0**((4*k_8)/(k_1 + k_8)),
eta1_u=( - k_1*t*u0 - k_3*u0 - k_31*u0 - k_42*u0 - k_8*t*u0)/2,
eta0_u=0,
xi1_x=( - k_1*k_31*x - k_1*k_5 + k_3*k_8*x + k_42*k_8*x - k_5*k_8)/(k_1 + k_8),
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x= - k_1*x - k_40,
xi0_t= - k_1*t - k_42},
{k_1, k_42, k_31, k_2, k_3, k_5, k_40, k_8, k_43},
{k_8, f, k_43, k_3 + k_31 + k_42, k_1 + k_8}},
{{},
{f=k_45/(2*k_12 + k_13*u0)**(4/3),
eta1_u=( - 6*k_12*k_31 + 6*k_12*k_4 - 6*k_12*k_8*t - 3*k_13*k_31*u0 + 3*k_13*k_4
*u0 - 3*k_13*k_8*t*u0)/(2*k_13),
eta0_u=( - 2*k_12*k_13*x + 6*k_12*k_44 - 12*k_12*k_8 - k_13**2*u0*x + 3*k_13*
k_44*u0 - 6*k_13*k_8*u0)/(2*k_13),
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x=(k_13*x**2 - 6*k_40 - 6*k_44*x)/6,
xi0_t=k_3 - 2*k_31 + 3*k_4 - 2*k_8*t},
{k_12, k_31, k_2, k_3, k_4, k_5, k_44, k_40, k_8, k_13, k_45},
{2*k_12 + k_13*u0, k_45, f, k_13, k_8, k_44 - 2*k_8, k_31 - k_4}},
{{df(f,u0)*k_10*k_8*u0 + 2*df(f,u0)*k_11**2 + df(f,u0)*k_11*k_44*u0 + df(f,u0)*
k_11*k_8*u0 + 4*f*k_11*k_8},
{eta1_u=( - k_10*k_11*k_8*t*u0 - k_10*k_29*k_8*u0 - 2*k_11**3*t - 2*k_11**2*k_29
 - k_11**2*k_44*t*u0 - k_11**2*k_8*t*u0 - k_11*k_29*k_44*u0 - k_11*k_29*k_8*u0)/
(2*k_11**2),
eta0_u=( - k_10**2*k_8*u0 - 2*k_10*k_11**2 - k_10*k_11*k_44*u0 - k_10*k_11*k_8*
u0)/(2*k_11**2),
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_11*k_2 - 2*k_11*k_4*t - k_11*k_8*t**2 - 2*k_29*k_8*t)/(2*k_11),
xi0_x= - k_40 - k_44*x,
xi0_t=( - k_10*k_11*k_8*t - k_10*k_29*k_8 + k_11**2*k_3 + k_11**2*k_4 - k_11**2*
k_44*t - k_11*k_29*k_44)/k_11**2},
{f, k_2, k_3, k_4, k_5, k_44, k_40, k_29, k_10, k_8, k_11},
{k_8, k_10*k_8*u0 + 2*k_11**2 + k_11*k_44*u0 + k_11*k_8*u0,
f, df(f,u0), k_11, k_10, k_29}},
{{},
{f=k_46/u0**((4*k_8)/(k_1 + k_8)),
eta1_u=( - k_1*k_31*u0 + k_1*k_4*u0 - k_1*k_8*t*u0 - k_31*k_8*u0 + k_4*k_8*u0 -
k_8**2*t*u0)/(2*k_8),
eta0_u=( - k_1**2*u0 + k_1*k_44*u0 - k_1*k_8*u0 + k_44*k_8*u0)/(2*k_8),
xi1_x= - k_4*x - k_5,
xi1_t=( - 2*k_2 - 2*k_31*t - k_8*t**2)/2,
xi0_x= - k_40 - k_44*x,
```

```
xi0_t=( - k_1*k_31 + k_1*k_4 - k_1*k_8*t + k_3*k_8 + k_4*k_8)/k_8},
{k_1, k_31, k_2, k_3, k_4, k_5, k_44, k_40, k_8, k_46},
{ - k_1 + k_44, k_8, k_1 + k_8, k_46, f, k_31 - k_4}},
{{},
{f=k_50/(2*k_12 + k_13*u0)**(4/3),
eta1_u=( - 6*k_12*k_31 + 6*k_12*k_47 - 3*k_13*k_31*u0 + 3*k_13*k_47*u0)/(2*k_13),
eta0_u=( - 2*k_12*x - k_13*u0*x)/2,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_31*t,
xi0_x=(k_13*x**2 - 6*k_49)/6,
xi0_t=k_3 - 2*k_31 + 3*k_47},
{k_12, k_31, k_2, k_3, k_47, k_48, k_49, k_13, k_50},
{2*k_12 + k_13*u0, k_50, f, k_13, - k_31 + k_47}},
{{},
{f=k_52/u0**(4/3),
eta1_u=( - 3*k_31*u0 + 3*k_47*u0)/2,
eta0_u=( - k_13*u0*x + 3*k_51*u0)/2,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_31*t,
xi0_x=(k_13*x**2 - 6*k_49 - 6*k_51*x)/6,
xi0_t=k_3 - 2*k_31 + 3*k_47},
{k_31, k_2, k_3, k_47, k_48, k_49, k_13, k_51, k_52},
{ - k_31 + k_47, k_51, k_13, k_52, f}},
{{},
{f=k_53/(2*k_10 - 3*k_51*u0)**(4/3),
eta1_u=( - 2*k_10*k_29 + 3*k_29*k_51*u0)/(2*k_10),
eta0_u=( - 2*k_10**2 - 2*k_10*k_12*x + 3*k_10*k_51*u0 + 3*k_12*k_51*u0*x)/(2*
k_10),
xi1_x= - k_47*x - k_48,
xi1_t=( - k_10*k_2 - k_10*k_47*t + k_29*k_51*t)/k_10,
xi0_x=( - 2*k_10*k_49 - 2*k_10*k_51*x - k_12*k_51*x**2)/(2*k_10),
xi0_t=(k_10*k_3 + k_10*k_47 + 2*k_29*k_51)/k_10},
{k_2, k_3, k_47, k_48, k_49, k_29, k_12, k_51, k_10, k_53},
{k_54, 2*k_10 - 3*k_51*u0, k_12, k_10, k_51, k_53, f, k_29}},
{{},
{f=u0**((4*k_51)/k_54)*k_55,
eta1_u=(k_31*k_54*u0 - k_47*k_54*u0)/(2*k_51),
eta0_u=( - k_54*u0)/2,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_31*t,
xi0_x= - k_49 - k_51*x,
xi0_t=(k_3*k_51 + k_31*k_51 + k_31*k_54 - k_47*k_54)/k_51},
{k_31, k_2, k_3, k_47, k_48, k_49, k_51, k_54, k_55},
{f, k_55, k_51, k_54, - k_31 + k_47}},
{{2*df(f,u0)*k_29 + df(f,u0)*k_3*u0 + df(f,u0)*k_31*u0 + df(f,u0)*k_42*u0 + 4*f*
k_31 - 4*f*k_47},
{eta1_u=( - 2*k_29 - k_3*u0 - k_31*u0 - k_42*u0)/2,
eta0_u=( - 2*k_10*k_29 - k_10*k_3*u0 - k_10*k_31*u0 - k_10*k_42*u0)/(2*k_29),
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_31*t,
```

```
xi0_x=(k_10*k_31*x - k_10*k_47*x - k_29*k_49)/k_29,
xi0_t= - k_42},
{f, k_42, k_31, k_2, k_3, k_47, k_48, k_49, k_10, k_29},
{k_10, k_29, f, df(f,u0),
2*k_29 + k_3*u0 + k_31*u0 + k_42*u0, - k_31 + k_47}},
{{2*df(f,u0)*k_29 + df(f,u0)*k_3*u0 + df(f,u0)*k_31*u0 + df(f,u0)*k_42*u0 + 4*f*
k_31 - 4*f*k_47},
{eta1_u=( - 2*k_29 - k_3*u0 - k_31*u0 - k_42*u0)/2,
eta0_u=0,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_31*t,
xi0_x= - k_49,
xi0_t= - k_42},
{f, k_42, k_31, k_2, k_29, k_3, k_47, k_48, k_49},
{f, df(f,u0), 2*k_29 + k_3*u0 + k_31*u0 + k_42*u0}},
{{},
{f=k_57/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_47*u0)/2,
eta0_u=( - k_13*u0*x)/2,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_31*t,
xi0_x=(k_13*x**2 - 6*k_56)/6,
xi0_t=k_3 - 2*k_31 + 3*k_47},
{k_31, k_2, k_3, k_47, k_48, k_56, k_13, k_18, k_57},
{ - k_31 + k_47, k_18, k_13, k_57, f}},
{{},
{f=k_58/(k_18*u0 + 2*k_20)**(4/3),
eta1_u=( - k_18*k_20*u0*x - k_18*k_29*u0 - 2*k_20**2*x - 2*k_20*k_29)/(2*k_20),
eta0_u=( - k_12*k_18*u0*x - 2*k_12*k_20*x)/(2*k_20),
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t=( - k_18*k_29*t - 3*k_2*k_20 - 3*k_20*k_47*t)/(3*k_20),
xi0_x=(k_12*k_18*x**2 - 6*k_20*k_56)/(6*k_20),
xi0_t=( - 2*k_18*k_29 + 3*k_20*k_3 + 3*k_20*k_47)/(3*k_20)},
{k_2, k_3, k_47, k_48, k_56, k_29, k_18, k_12, k_20, k_58},
{f,
k_58, k_18, k_12, k_20, k_18*u0 + 2*k_20, k_29}},
{{},
{f=k_60/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_47*u0)/2,
eta0_u=( - k_13*u0*x + 3*k_59*u0)/2,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_31*t,
xi0_x=(k_13*x**2 - 6*k_56 - 6*k_59*x)/6,
xi0_t=k_3 - 2*k_31 + 3*k_47},
{k_31, k_2, k_3, k_47, k_48, k_56, k_13, k_18, k_59, k_60},
{ - k_31 + k_47, k_54, k_13, f, k_60, k_59, k_18}},
{{},
{f=k_61/(2*k_10 - 3*k_59*u0)**(4/3),
eta1_u=( - 2*k_10*k_20*x - 2*k_10*k_29 + 3*k_20*k_59*u0*x + 3*k_29*k_59*u0)/(2*
k_10),
```

```
eta0_u=(2*k_10*k_13*x - 6*k_10*k_59 - 3*k_13*k_59*u0*x + 9*k_59**2*u0)/(6*k_59),
xi1_x=( - 2*k_10*k_47*x - 2*k_10*k_48 - k_20*k_59*x**2)/(2*k_10),
xi1_t=( - k_10*k_2 - k_10*k_47*t + k_29*k_59*t)/k_10,
xi0_x=(k_13*x**2 - 6*k_56 - 6*k_59*x)/6,
xi0_t=(k_10*k_3 + k_10*k_47 + 2*k_29*k_59)/k_10},
{k_2, k_3, k_47, k_48, k_56, k_29, k_13, k_20, k_59, k_10, k_61},
{k_13, k_59, k_10, k_20, k_61, f, 2*k_10 - 3*k_59*u0, k_54, k_29}},
{{},
{f=k_62/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_47*u0)/2,
eta0_u=(3*k_59*u0)/2,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_31*t,
xi0_x= - k_56 - k_59*x,
xi0_t=k_3 - 2*k_31 + 3*k_47},
{k_31, k_2, k_3, k_47, k_48, k_56, k_18, k_59, k_62},
{ - k_31 + k_47, f, k_62, k_59, k_18}},
{{},
{f=k_63/(2*k_10 - 3*k_59*u0)**(4/3),
eta1_u=( - 2*k_10*k_20*x - 2*k_10*k_29 + 3*k_20*k_59*u0*x + 3*k_29*k_59*u0)/(2*
k_10),
eta0_u=( - 2*k_10 + 3*k_59*u0)/2,
xi1_x=( - 2*k_10*k_47*x - 2*k_10*k_48 - k_20*k_59*x**2)/(2*k_10),
xi1_t=( - k_10*k_2 - k_10*k_47*t + k_29*k_59*t)/k_10,
xi0_x= - k_56 - k_59*x,
xi0_t=(k_10*k_3 + k_10*k_47 + 2*k_29*k_59)/k_10},
{k_2, k_3,
k_47, k_48, k_56, k_29, k_20, k_59, k_10, k_63},
{k_54, 2*k_10 - 3*k_59*u0, k_59, k_10, k_20, k_63, f, k_29}},
{{},
{f=k_64/u0**(4/3),
eta1_u=( - k_18*u0*x - 3*k_31*u0 + 3*k_47*u0)/2,
eta0_u=0,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_31*t,
xi0_x= - k_56,
xi0_t=k_3 - 2*k_31 + 3*k_47},
{k_31, k_2, k_3, k_47, k_48, k_56, k_18, k_64},
{ - k_31 + k_47, k_18, k_64, f}},
{{},
{f=k_65/( - 2*k_29 - 3*k_31*u0 + 3*k_47*u0)**(4/3),
eta1_u=( - 2*k_20*k_29*x - 3*k_20*k_31*u0*x + 3*k_20*k_47*u0*x - 2*k_29**2 - 3*
k_29*k_31*u0 + 3*k_29*k_47*u0)/(2*k_29),
eta0_u=0,
xi1_x=(k_20*k_31*x**2 - k_20*k_47*x**2 - 2*k_29*k_47*x - 2*k_29*k_48)/(2*k_29),
xi1_t= - k_2 - k_31*t,
xi0_x= - k_56,
xi0_t=k_3 - 2*k_31 + 3*k_47},
{k_31, k_2, k_3, k_47, k_48, k_56, k_20, k_29, k_65},
{f, k_65, k_20, k_29, - k_31 + k_47, - 2*k_29 - 3*k_31*u0 + 3*k_47*u0}},
```

```
{{},
{f=k_67/(k_18*u0 + 2*k_20)**(4/3),
eta1_u=( - k_18*u0*x - 2*k_20*x)/2,
eta0_u=0,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_47*t,
xi0_x= - k_66,
xi0_t=k_3 + k_47},
{k_2, k_20, k_3, k_47, k_48, k_66, k_18, k_67},
{k_18,f,k_67,k_18*u0 + 2*k_20}},
{{},
{f=k_69/(2*k_10 - 3*k_68*u0)**(4/3),
eta1_u=( - 2*k_10*k_20*x + 3*k_20*k_68*u0*x)/(2*k_10),
eta0_u=( - 2*k_10 + 3*k_68*u0)/2,
xi1_x=( - 2*k_10*k_47*x - 2*k_10*k_48 - k_20*k_68*x**2)/(2*k_10),
xi1_t= - k_2 - k_47*t,
xi0_x= - k_66 - k_68*x,
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_66, k_20, k_68, k_10, k_69},
{f, k_69, k_20, k_10, k_68, 2*k_10 - 3*k_68*u0, k_54}},
{{},
{f=k_70/u0**(4/3),
eta1_u=( - k_18*u0*x)/2,
eta0_u=(3*k_68*u0)/2,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_47*t,
xi0_x= - k_66 - k_68*x,
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_66, k_68, k_18, k_70},
{k_68, k_18, k_70, f}},
{{},
{f=k_71/(2*k_10 - 3*k_68*u0)**(4/3),
eta1_u=( - 2*k_10*k_20*x + 3*k_20*k_68*u0*x)/(2*k_10),
eta0_u=(2*k_10*k_13*x - 6*k_10*k_68 - 3*k_13*k_68*u0*x + 9*k_68**2*u0)/(6*k_68),
xi1_x=( - 2*k_10*k_47*x - 2*k_10*k_48 - k_20*k_68*x**2)/(2*k_10),
xi1_t= - k_2 - k_47*t,
xi0_x=(k_13*x**2 - 6*k_66 - 6*k_68*x)/6,
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_66, k_13, k_20, k_68, k_10, k_71},
{k_54, 2*k_10 - 3*k_68*u0, f, k_71, k_20, k_10, k_68, k_13}},
{{},
{f=k_72/u0**(4/3),
eta1_u=( - k_18*u0*x)/2,
eta0_u=( - k_13*u0*x + 3*k_68*u0)/2,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_47*t,
xi0_x=(k_13*x**2 - 6*k_66 - 6*k_68*x)/6,
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_66, k_13, k_68, k_18, k_72},
{k_54, k_68, k_18, k_72, f, k_13}},
```

```
{{},
{f=k_73/(k_18*u0 + 2*k_20)**(4/3),
eta1_u=( - k_18*u0*x - 2*k_20*x)/2,
eta0_u=( - k_12*k_18*u0*x - 2*k_12*k_20*x)/(2*k_20),
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_47*t,
xi0_x=(k_12*k_18*x**2 - 6*k_20*k_66)/(6*k_20),
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_66, k_18, k_12, k_20, k_73},
{k_18*u0 + 2*k_20, k_20, k_12, k_18, k_73, f}},
{{},
{f=k_74/u0**(4/3),
eta1_u=( - k_18*u0*x)/2,
eta0_u=( - k_13*u0*x)/2,
xi1_x=(k_18*x**2 - 6*k_47*x - 6*k_48)/6,
xi1_t= - k_2 - k_47*t,
xi0_x=(k_13*x**2 - 6*k_66)/6,
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_66, k_13, k_18, k_74},
{f, k_74, k_13, k_18}},
{{},
{eta1_u=0,
eta0_u=0,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_47*t,
xi0_x= - k_75,
xi0_t=k_3 + k_47},
{f, k_2, k_3, k_47, k_48, k_75},
{f,df(f,u0)}},
{{2*df(f,u0)*k_10 + df(f,u0)*k_54*u0 - 4*f*k_76},
{eta1_u=0,
eta0_u=( - 2*k_10 - k_54*u0)/2,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_47*t,
xi0_x= - k_75 - k_76*x,
xi0_t=k_3 + k_47},
{f, k_10, k_54, k_2, k_3, k_47, k_48, k_76, k_75},
{2*k_10 + k_54*u0, f, df(f,u0)}},
{{},
{f=k_77/(2*k_10 - 3*k_76*u0)**(4/3),
eta1_u=0,
eta0_u=( - 2*k_10**2 - 2*k_10*k_12*x + 3*k_10*k_76*u0 + 3*k_12*k_76*u0*x)/(2*
k_10),
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_47*t,
xi0_x=( - 2*k_10*k_75 - 2*k_10*k_76*x - k_12*k_76*x**2)/(2*k_10),
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_75, k_12, k_76, k_10, k_77},
{f, k_77, k_76, k_10, k_12, 2*k_10 - 3*k_76*u0, k_54}},
{{},
```

```
{f=k_78/u0**(4/3),
eta1_u=0,
eta0_u=( - k_13*u0*x + 3*k_76*u0)/2,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_47*t,
xi0_x=(k_13*x**2 - 6*k_75 - 6*k_76*x)/6,
xi0_t=k_3 + k_47},
{k_2, k_3, k_47, k_48, k_75, k_13, k_76, k_78},
{f, k_78, k_13, k_76}},
{{},
{f=k_79/(2*k_12 + k_13*u0)**(4/3),
eta1_u=0,
eta0_u=( - 2*k_12*x - k_13*u0*x)/2,
xi1_x= - k_47*x - k_48,
xi1_t= - k_2 - k_47*t,
xi0_x=(k_13*x**2 - 6*k_75)/6,
xi0_t=k_3 + k_47},
{k_12, k_2, k_3, k_47, k_48, k_75, k_79, k_13},
{f, k_79, k_13, 2*k_12 + k_13*u0}}}$


quit;
```

The funcion `reliesolve()` *returns 53 solution sets for the symmetries, according to the different forms of the function* f*. An analysis of these solutions is required to the user in order to eliminate duplicates and/or subcases.*

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

```
let epsilon^(approxorder+1) = 0 $
```

4.9 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 fix:

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` denoting 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_` followed by the name of the arbitrary element. Let us illustrate with two different examples how ReLie works in such a situation.

Example 14 *[24] Consider the class* $\mathcal{E}(\mathbf{p})$ *with* $\mathbf{p} = (p_1, p_2, p_3)$ *of (*2×2*) systems*

$$\begin{aligned} &\partial_t u_1 - \partial_x u_2 = 0, \\ &\partial_t u_2 + p_1(x,t,u_1,u_2)\partial_x u_1 + p_2(x,t,u_1,u_2)\partial_x u_2 = p_3(x,t,u_1,u_2), \end{aligned} \tag{10}$$

where $p_1(x,t,u_1,u_2)$, $p_2(x,t,u_1,u_2)$ *and* $p_3(x,t,u_1,u_2)$ *are arbitrary continuously differentiable functions of the indicated arguments. The* ReLie *code needed to compute the equivalence transformations of such a system is as follows.*

```
load_package relie $
off nat $
jetorder:=1 $
uvar:={u1,u2} $
xvar:={t,x} $
arbelem:={p1,p2,p3} $
arborder:=0 $
zorder:=0 $
diffeqs:=
{u1_t-u2_x, u2_t+p1*u1_x+p2*u2_x-p3} $
leadders:={u1_t,u2_t}$
relie(4) $
```

The symmetries which are found are given below.

```
symmetries ->
{
 {
  {},
  {mu_p3= - df(c_12,t,x)*p2 - df(c_12,t,2) - df(c_12,x,2)*p1
         - df(c_3,t,x)*p2*u1 - df(c_3,t,2)*u1 - df(c_3,x,2)*p1*u1
         - df(c_4,t,x)*p2*u2 - df(c_4,t,2)*u2 - 2*df(c_4,t)*p3
         - df(c_4,x,2)*p1*u2 - df(c_4,x)*p2*p3 + c_11*p3,
   mu_p2=2*df(c_3,t) + df(c_3,x)*p2 - df(c_4,t)*p2
         + 2*df(c_4,x)*p1 - df(c_4,x)*p2**2,
   mu_p1=df(c_3,t)*p2 + 2*df(c_3,x)*p1
          - 2*df(c_4,t)*p1 - df(c_4,x)*p1*p2,
   eta_u2= - df(c_12,t) - df(c_3,t)*u1 - df(c_4,t)*u2 + c_11*u2,
   eta_u1= - df(c_12,x) - df(c_3,x)*u1 - df(c_4,x)*u2 + c_11*u1,
   xi_x=c_3,
```

```
   xi_t=c_4},
  {c_3,c_4,c_12,c_11},
  {}
 }
}
```

The dependencies of the parameters involved in the infinitesimals can be determined calling the function `fargs()`.

Example 15 *[25] Consider the class* $\mathcal{E}(\mathbf{p})$ *with* $\mathbf{p} = (p_1, p_2, p_3, p_4, p_5, p_6)$ *of systems*

$$\begin{aligned} &\frac{\partial u_1}{\partial t} + \frac{\partial u_2}{\partial x} + \frac{\partial u_3}{\partial y} = 0, \\ &\frac{\partial u_2}{\partial t} + \frac{\partial p_1}{\partial x} + \frac{\partial p_2}{\partial y} - p_5 = 0, \\ &\frac{\partial u_3}{\partial t} + \frac{\partial p_3}{\partial x} + \frac{\partial p_4}{\partial y} - p_6 = 0, \end{aligned} \tag{11}$$

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*

```
load_package relie $
off nat $
jetorder:=1 $
uvar:={u1,u2,u3} $
xvar:={t,x,y} $
arbelem:={p1,p2,p3,p4,p5,p6} $
arborder:=1 $
zorder:=0 $
diffeqs:={u1_t+u2_x+u3_y,
          u2_t+p1_x+p1_u1*u1_x+p1_u2*u2_x+p1_u3*u3_x
              +p2_y+p2_u1*u1_y+p2_u2*u2_y+p2_u3*u3_y-p5,
          u3_t+p3_x+p3_u1*u1_x+p3_u2*u2_x+p3_u3*u3_x
              +p4_y+p4_u1*u1_y+p4_u2*u2_y+p4_u3*u3_y-p6} $
leadders:={u1_t,u2_t,u3_t} $
relie(4) $
```

As a result we have the infinitesimals:

```
symmetries ->
{
 {
  {
   df(f_3,y)+df(f_6,t)+df(f_7,x)
  },
  {
   eta_u3=-df(f_1,t)*u1-df(f_1,x)*u2+df(f_10,t)*u3+df(f_2,x)*u3+f_3-k_1*u3,
```

```
   eta_u2=df(f_1,y)*u2+df(f_10,t)*u2-df(f_2,t)*u1-df(f_2,y)*u3+f_7-k_1*u2,
   eta_u1=df(f_1,y)*u1+df(f_2,x)*u1+f_6-k_1*u1,
   xi_y=-f_1,
   xi_x=-f_2,
   xi_t=-f_10,
   mu_p6=-2*df(f_1,t,x)*u2-2*df(f_1,t,y)*u3-df(f_1,t,2)*u1-df(f_1,x,y)*p2
         -df(f_1,x,y)*p3-df(f_1,x,2)*p1-df(f_1,x)*p5-df(f_1,y,2)*p4
         +df(f_10,t,2)*u3+2*df(f_10,t)*p6+df(f_2,x)*p6+df(f_3,t)
         +df(f_4,x)-df(f_5,y)-k_1*p6,
   mu_p5=df(f_1,y)*p5+df(f_10,t,2)*u2+2*df(f_10,t)*p5-2*df(f_2,t,x)*u2
         -2*df(f_2,t,y)*u3-df(f_2,t,2)*u1-df(f_2,x,y)*p2
         -df(f_2,x,y)*p3-df(f_2,x,2)*p1-df(f_2,y,2)*p4-df(f_2,y)*p6
         +df(f_7,t)+df(f_8,y)+df(f_9,x)-k_1*p5,
   mu_p4=-2*df(f_1,t)*u3-df(f_1,x)*p2-df(f_1,x)*p3-df(f_1,y)*p4
         +2*df(f_10,t)*p4+df(f_2,x)*p4-f_5-k_1*p4,
   mu_p3=-df(f_1,t)*u2-df(f_1,x)*p1+2*df(f_10,t)*p3-df(f_2,t)*u3
         -df(f_2,y)*p4+f_4-k_1*p3,
   mu_p2=-df(f_1,t)*u2-df(f_1,x)*p1+2*df(f_10,t)*p2-df(f_2,t)*u3
         -df(f_2,y)*p4+f_8-k_1*p2,
   mu_p1=df(f_1,y)*p1+2*df(f_10,t)*p1-2*df(f_2,t)*u2-df(f_2,x)*p1
         -df(f_2,y)*p2-df(f_2,y)*p3+f_9-k_1*p1
  },
  {
    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`, *...,* `f_10` *are functions of* t, x, y*, along with the constraint*

$$\frac{\partial f_6}{\partial t} + \frac{\partial f_3}{\partial x} + \frac{\partial f_7}{\partial y} = 0. \tag{12}$$

Example 16 *The equivalence transformations of the equation*

$$\frac{\partial^2 u}{\partial t^2} - f\left(x, \frac{\partial u}{\partial x}\right)\frac{\partial^2 u}{\partial x^2} - g\left(x, \frac{\partial u}{\partial x}\right) = 0 \tag{13}$$

are immediately found in ReLie *with the following code:*

```
load_package relie $
off nat $
jetorder:=2 $
xvar:={t,x} $
uvar:={u} $
zorder:=1 $
arborder:=1 $
arbelem:={f,g} $
diffeqs:={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 `diffeqs` *the auxiliary conditions; the variable* `zorder` *has been set to 1 since the functions f and g depend on first order derivatives; moreover, the variable* `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:

```
symmetries ->
{
 {
  {},
  {eta_u=(2*f_1-2*k_1*u-k_6*t**2-2*k_7*t-2*k_8)/2,
   xi_x= -k_4*x-k_5,
   xi_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},
  {}
 }
}
```

where the function `f_1` *is arbitrary and depends on x.*

Remark 4 *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 [21], then we have to add the statement*

```
generalequiv:=1 $
```

4.10 Inverse Lie problem

Here we show with a simple example how ReLie can be used for investigating the Lie remarkability of differential equations [20, 19, 18, 9]. 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 [20]:

Example 17
```
load_package relie $
off nat $
jetorder:=2 $
xvar:={x,y} $
uvar:={u} $
relieinit() $
```

```
generatealgebra(2) $
```

After setting the necessary objects (`jetorder`, `xvar` *and* `uvar`*), and calling* `relieinit()`*, we generate the vector fields spanning the affine Lie algebra in* $\mathbb{R}^3$*. Then, calling* `testrank()` *(this function uses the list* `splitsymmetries` *returned by the function* `generatealgebra()`*), we ascertain that the rank of the second order distribution is maximal (equal to the dimension of second order jet space); the call* `inverselie(8)` *computes all minors of order 8 of such a distribution; if we solve* `allminors` *with respect to* `u_yy` *we get*

```
{{u_yy=u_xy^2/u_xx}};
```

i.e., the Monge-Ampère equation. As a last check, substituting in `distribution` *this expression for* `u_tt`*, the rank of the matrix* `distribution` *reduces to 7, which is dimension of the manifold characterized by Monge-Ampère equation in the second order jet space.*

The log of the REDUCE *session is as follows.*

```
*****************************************************************

ReLie, version 3.0

A Reduce program for Lie group analysis of differential equations

(c) Francesco Oliveri (foliveri@unime.it) - 2020

Last update August 9, 2020

http://mat521.unime.it/oliveri/

*****************************************************************


off nat $

jetorder:=2 $

xvar:={x,y} $

uvar:={u} $

relieinit() $
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).$
```

```
generatealgebra(2) $
The lists 'generators' and 'splitsymmetries' have been computed.$

{{1,0,0},
{0,1,0},
{0,0,1},
{x,0,0},
{y,0,0},
{u,0,0},
{0,x,0},
{0,y,0},
{0,u,0},
{0,0,x},
{0,0,y},
{0,0,u}}$

% we have a basis of the Affine Lie algebra in R^3

testrank();
'prolongation' has been computed.$

the matrix 'distribution' has been computed.$

'rankdistr' has been computed.$


rankdistr;
8$

% the 2nd order distribution has maximal rank

% let us compute all minors of order 8 of the matrix distribution

inverselie(8) $
'prolongation' has been computed.$

the matrix 'distribution' has been computed.$

The list 'allminors' has been computed.$


length(allminors);
489$


sol:=solve(allminors,u_yy);
sol := {u_yy=u_xy**2/u_xx}$


distribution:=sub(sol,distribution) $
```

```
rank(distribution);
7$


quit;
```

5 Interactive use of ReLie

In most cases ReLie is able to complete its assignments. However, there are situations where the function `reliesolve()` is not able to solve the determining equations, and to proceed the user has to make some special assumptions to force the computation. Also, there are cases where it is convenient (for instance in a teaching context) to interact with Reduce and get the solution interactively.

Example 18 (Non-classical symmetries of Burgers equation) *The following code computes the nonlinear determining equations for the non-classical symmetries of Burgers equation,* i.e. *the symmetries of the solutions of the system*

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} = 0, \qquad \frac{\partial u}{\partial t} + \xi(t,x,u)\frac{\partial u}{\partial x} - \eta(t,x,u) = 0.$$

```
load_package relie $
off nat $
jetorder:=2 $
xvar:={t,x} $
uvar:={u} $
nonclassical:=1 $
qcond:={1} $
diffeqs:={u_t+u*u_x-u_xx} $
leadders:={u_xx} $
relie(3) $
```

The function `reliesolve()` is not able to solve the nonlinear determining equations; thus, we need to work interactively to find the nonclassical solutions. Then the user may start an interactive session to analyze `deteqs`.

References

[1] G. Baumann. *Symmetry analysis of differential equations with Mathematica.* Springer, New York, 2000.

[2] G. W. Bluman and S. C. Anco. *Symmetry and integration methods for differential equations.* Springer, New York, 2002.

[3] G. W. Bluman, A. F. Cheviakov, and S. C. Anco. *Applications of symmetry methods to partial differential equations.* Springer, New York, 2009.

[4] J. Butcher, J. Carminati, and K. T. Vu. A comparative study of some computer algebra packages which determine the Lie point symmetries of differential equations. *Computer Physics Communications*, 155:92–114, 2003.

[5] J. Carminati and K. T. Vu. Symbolic computation and differential equations: Lie symmetries. *Journal of Symbolic Computation*, 29:95–116, 2000.

[6] A. F. Cheviakov. GeM software package for computation of symmetries and conservation laws of differential equations. *Computer Physics Communications*, 176:48–61, 2007.

[7] A. F. Cheviakov. Symbolic computation of local symmetries of nonlinear and linear partial and ordinary differential equations. *Mathematics in Computer Science*, 4:203–222, 2010.

[8] R. Di Salvo, M. Gorgone, and F. Oliveri. A consistent approach to approximate Lie symmetries of differential equations. *Nonlinear Dynamics*, 91:371–386, 2018.

[9] M. Gorgone and F. Oliveri. Lie remarkable partial differential equations characterized by Lie algebras of point symmetries. *Journal of Geometry and Physics*, 144:314–323, 2019.

[10] A. K. Head. LIE: A PC program for lie analysis of differential equations. *Computer Physics Communications*, 77:241–248, 1993.

[11] A. C. Hearn. *Reduce User's manual*. Rand, Santa Monica, 1993.

[12] W. Hereman. Review of symbolic software for the computation of lie symmetries of differential equations. *Euromath Bulletin*, 1:45–82, 1994.

[13] W. Hereman. Review of symbolic software for Lie symmetry analysis. *Mathematical and Computer Modelling*, 25:115–132, 1997.

[14] N. H. Ibragimov (ed.). *CRC Handbook of Lie group analysis of differential equations, vol. 1. Symmetries, Exact Solutions, and Conservation Laws*. CRC Press, Boca Raton, 1994.

[15] N. H. Ibragimov (ed.). *CRC Handbook of Lie group analysis of differential equations, vol. 2. Applications in engineering and physical sciences*. CRC Press, Boca Raton, 1995.

[16] N. H. Ibragimov (ed.). *CRC Handbook of Lie group analysis of differential equations, vol. 3. New trends in theoretical developments and computational methods*. CRC Press, Boca Raton, 1996.

[17] G. F. Jefferson and J. Carminati. ASP: Automated symbolic computation of approximate symmetries of differential equations. *Computer Physics Communications*, 184:1045–1063, 2013.

[18] G. Manno, F. Oliveri, G. Saccomandi, and R. Vitolo. Ordinary differential equations described by their Lie symmetry algebra. *Journal of Geometry and Physics*, 85:2–15, 2014.

[19] G. Manno, F. Oliveri, and R. Vitolo. Differential equations uniquely determined by algebras of point symmetries. *Theoretical and Mathematical Physics*, 151:843–850, 2007.

[20] G. Manno, F. Oliveri, and R. Vitolo. On differential equations characterized by their Lie point symmetries. *Journal of Mathematical Analysis and Applications*, 332:767–786, 2007.

[21] S. V. Meleshko. Generalization of the equivalence transformations. *Nonlinear Mathematical Physics*, 3:170–174, 1996.

[22] F. Oliveri. Lie symmetries of differential equations: classical results and recent contributions. *Symmetry*, 2:658–706, 2010.

[23] F. Oliveri. ReLie: a Reduce program for Lie group analysis of differential equations. Submitted, 2021.

[24] F. Oliveri and M. P. Speciale. Equivalence transformations of quasilinear first order systems and reduction to autonomous and homogeneous form. *Acta Applicandae Mathematicae*, 122:447–460, 2012.

[25] F. Oliveri and M. P. Speciale. Reduction of balance laws to conservation laws by means of equivalence transformations. *Journal of Mathematical Physics*, 54:041506, 2013.

[26] P. J. Olver. *Applications of Lie groups to differential equations*. Springer, New York, 1986.

[27] L. V. Ovsiannikov. *Group analysis of differential equations*. Academic Press, New York, 1982.

[28] T. M. Rocha Filho and A. Figueiredo. [SADE] a Maple package for the symmetry analysis of differential equations. *Computer Physics Communications*, 182:467–476, 2010.

[29] F. Schwarz. Automatically determining symmetries of partial differential equations. *Computing*, 34:91–106, 1985. Addendum. Computing, **36**, 279–280 (1986).

[30] F. Schwarz. Symmetries of differential equations from Sophus Lie to computer algebra. *SIAM Review*, 30:450–481, 1988.

[31] J. Sherring and G. Prince. DIMSYM–Symmetry determination and linear differential equations package. Technical report, Department of Mathematics, LaTrobe University, Bundoora, Australia, 1996.

[32] W. H. Steeb. *Continuous symmetries, Lie algebras, differential equations and computer algebra*. World Scientific Publishing Co. Pte. Ltd., Singapore, 2nd edition, 2007.

[33] T. Wolf. Investigating differential equations with CRACK, LiePDE, Applysym and ConLaw. In J. Grabmeier, E. Kaltofen, and V. Weispfenning, editors, *Handbook of Computer Algebra, Foundations, Applications, Systems*, pages 465–468. Springer, New York, 2002.