GeM software package for computation of symmetries and conservation laws of differential equations

Alexei F. Cheviakov [a]

[a] *Department of Mathematics, University of British Columbia, Vancouver, V6T 1Z2 Canada*

Abstract

We present a recently developed Maple-based "GeM" software package for automated symmetry and conservation law analysis of systems of partial and ordinary differential equations (DE). The package contains a collection of powerful easy-to-use routines for mathematicians and applied researchers. A standard program that employs "GeM" routines for symmetry, adjoint symmetry or conservation law analysis of any given DE system occupies several lines of Maple code, and produces output in the canonical form. Classification of symmetries and conservation laws with respect to constitutive functions and parameters present in the given DE system is implemented. The "GeM" package is being successfully used in ongoing research. Run examples include classical and new results.

Key words:
Symbolic computation; Symmetries; Conservation laws; Differential equations; PDE; Classification
`CPC Program Library Classification(s)`

PROGRAM SUMMARY

Title of program: GeM [1]

Catalogue number: (supplied by Elsevier)

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Licensing provisions: none

[1] Electronic mail: alexch@math.ubc.ca

Computer for which the program is designed and others on which it has been tested: *Computers:* PC-compatible running Maple on MS Windows or Linux; SUN systems running Maple for Unix on OS Solaris. *Installations:*

Operating systems under which the program has been tested: Windows 2000, Windows XP, Linux, Solaris

Programming language used: Maple 9.5

Memory required to execute with typical data: below 100 Megabytes

No. of bits in a word:

No. of processors used:

Has the code been vectorised?:

Peripherals used:

No. of lines in distributed program, including test data, etc.: Maple package: 1600 lines. Examples: 10 – 30 lines each.

CPC Program Library subprograms used:

Keywords: Symbolic computation; Lie Symmetry; Lie-Bäcklund symmetry; Higher symmetry; Approximate symmetry; Adjoint symmetry; Conservation law; Classification.

Nature of physical problem
Any physical model containing linear or nonlinear partial or ordinary differential equations.

Method of solution
Symbolic computation of Lie, higher and approximate symmetries by Lie's algorithm. Symbolic computation of conservation laws and adjoint symmetries by using multipliers and Euler operator properties.

High performance is achieved by using an efficient representation of the system under consideration and resulting symmetry / conservation law determining equations: all dependent variables and derivatives are represented as symbols rather than functions or expressions.

Restrictions on the complexity of the problem
The `GeM` module routines are normally able to handle ODE/PDE systems of high orders (up to order seven and possibly higher), depending on the nature of the problem. Classification of symmetries/conservation laws with respect to one or more arbitrary constitutive function(s) of one or two arguments is normally accomplished successfully.

Typical running time
1 – 20 seconds for problems that do not involve classification; 5 – 1000 seconds for

problems that involve classification, depending on complexity.

Unusual features of the program

References

[1] The updated versions of the `GeM` package and the documentation are available at `http://www.math.ubc.ca/~alexch/gem/`.

LONG WRITE-UP

1 Introduction

Symmetry, adjoint symmetry and conservation law analysis are practically important means of analyzing nonlinear systems of ordinary and partial differential equations [1,2].

A symmetry of a system of DEs is any transformation of its solution manifold into itself. One-parameter and multi-parameter Lie groups of point symmetries are found using the general Lie algorithm, which is equally applicable to algebraic, ordinary and partial differential equations. Contact and higher symmetries are computed in a similar algorithmic manner Closely related is the potential symmetry method [2] for discovering nonlocal symmetries.

Symmetries of ordinary differential equations (ODE) are used for reduction of order and complete integration, as well as for the construction of invariant solutions. Symmetries of partial differential equations (PDE) yield reductions of order and/or number of variables. Invariant (in particular, self-similar) solutions that arise from reduced systems often have transparent physical meaning. Many appropriate examples are found in [3]. Infinite-dimensional symmetry groups are used for the construction of families of new exact solutions from known ones (e.g. [4–6].) For a nonlinear DE system, from its admitted symmetry group, one can determine whether or not it can be mapped into a linear system by an invertible transformation, and find the explicit form of that transformation [2].

Application of Lie's symmetry method for nontrivial systems often requires extensive algebraic manipulation and the solution of large overdetermined systems of linear PDEs. For many contemporary DE models, especially those that do possess non-trivial symmetry structure, such analysis presents a significant computational challenge. In Section 2.1, we overview the algorithm for symmetry group analysis and related challenges.

An important counterpart to knowledge of the symmetry structure of a PDE system is information about its conservation laws. Conservation laws describe essential physical properties of the modeled process. They are used for the development of appropriate numerical methods and for analysis, in particular, existence, uniqueness and stability analysis (e.g.[7–9].)

Several algorithms for finding conservation laws of PDEs and PDE systems exist [10]. Such algorithms often involve finding a set of multipliers, such that a linear combination of equations of the given system is a divergence expression. After finding multipliers, fluxes are algorithmically reconstructed (Section 2.2.) Sets of symmetries and conservation laws for a given PDE system are closely related (e.g. [11]).

Finding conservation laws and finding symmetries involves similar computational challenges, that consist in reduction of redundant systems of determining equations. Several methods for reduction of large overdetermined systems of partial differential equations have been developed (for a review, see [12].) Many of these methods have been implemented in symbolic software packages. In particular, the `Rif` package [13] is a part of `Waterloo Maple`, and is used in the `GeM` package described in this paper.

A `REDUCE`-based set of programs by Wolf [14] includes routines for point, contact and higher symmetry, and conservation law / adjoint symmetry analysis of DE systems.

In this work, we introduce a recently developed package `GeM` for `Maple` (Section 3.) It contains practically useful routines to perform local (Lie, contact, higher, adjoint and approximate) symmetry analysis and search for conservation laws of ODEs and PDEs without human intervention. The use of package routines does not require special knowledge or continuous input. The package is compatible with modern `Maple` versions (9.5 and above) and thus available for all popular platforms (Linux, Windows, UNIX.) Package routines automatically generate determining equations in a form suitable for effective automatic reduction and solution. For DE systems involving constitutive functions or parameters, symmetry / conservation law classification is automatically performed. The module routines handle DE systems of high order and with many equations and many dependent and/or independent variables. Programs that use `GeM` routines, even ones involving advanced classification, in most cases run effectively on a desktop computer.

Several run examples are presented in the end of the paper; more are available on the program website [15].

2 Symmetry, adjoint symmetry and conservation law analysis of systems of differential equations

2.1 The Lie procedure for finding symmetries

Consider a general system of l differential equations of order p

$$\begin{aligned}
&G(x, u, \underset{1}{u}, ..., \underset{p}{u}) = 0,\\
&G = (G^1, \dots, G^l),\ x = (x^1, \dots, x^n) \in \mathcal{X},\quad u = (u^1, \dots, u^m) \in \mathcal{U},\\
&\underset{k}{u} = \left(\frac{\partial^k u^j}{\partial x^{i_1} ... \partial x^{i_k}} |\ \ j = 1, \dots, m \right) \in \mathcal{U}_k,\quad k = 1, \dots, p,
\end{aligned} \tag{2.1}$$

with m dependent and n independent variables. The set of all solutions of (2.1) is a manifold Ω in $(m+n)$ - dimensional space $\mathcal{X} \times \mathcal{U}$, which corresponds to a manifold Ω_p in the prolonged (jet) space $\mathcal{X} \times \mathcal{U} \times \mathcal{U}_1 \times \ldots \times \mathcal{U}_p$ of dependent and independent variables together with partial derivatives [1].

Three types of local symmetries of DE systems are distinguished: point symmetries, contact symmetries and higher symmetries.

1. Lie Point symmetries. The Lie method yields Lie groups of point transformations (symmetries)

$$\begin{aligned}
&(x')^i = f^i(x, u, \varepsilon),\quad i = 1, \dots, n,\\
&(u')^j = g^j(x, u, \varepsilon),\quad j = 1, \dots, m,
\end{aligned} \tag{2.2}$$

or, in the local form,

$$\begin{aligned}
&(x')^i = x^i + \epsilon \xi^i(x, u) + O(\epsilon^2),\quad i = 1, \dots, n,\\
&(u')^j = u^j + \epsilon \eta^j(x, u) + O(\epsilon^2),\quad j = 1, \dots, m,
\end{aligned} \tag{2.3}$$

that map solutions of (2.1) into solutions. Here ε is a group parameter, or a vector of group parameters (in the case of multi-parameter groups.)

One considers the Lie algebra of infinitesimal operators (or symmetry generators)

$$X = \xi^i(x, u) \frac{\partial}{\partial x^i} + \eta^k(x, u) \frac{\partial}{\partial u^k}. \tag{2.4}$$

(Here and below, summation with respect to repeating indices is assumed.) Each symmetry generator defines a vector field tangent to the solution manifold Ω of the system (2.1). To each symmetry generator (2.4) there corresponds a unique prolonged vector field

$$v = \xi^i \frac{\partial}{\partial x^i} + \eta^k \frac{\partial}{\partial u^k} + \eta_i^{(1)\,k} \frac{\partial}{\partial u_i^k} + ... + \eta_{i_1,...,i_p}^{(p)\,k} \frac{\partial}{\partial u_{i_1,...,i_p}^k}. \tag{2.5}$$

tangent to the solution manifold Ω_p in the jet space. Here $\eta_{i_1,...,i_w}^{(w)\ k}$ are the coordinates of the prolonged tangent vector field corresponding to the derivatives of order w: $u_{i_1,...,i_w}^k$; $w = 1, ..., p$. The prolonged coordinates $\eta_{i_1,...,i_w}^{(w)\ k}$ are fully determined from (ξ^i, η^k) by differential relations

$$\eta_{i_1,...,i_w}^{(w)\,k} = D_{i_w} \eta_{i_1,...,i_{w-1}}^{(w-1)\,k} - (D_{i_w} \xi^l) u_{i_1,...,i_{w-1},l}^k. \tag{2.6}$$

Here it is assumed that $\eta^{(0)\,k} \equiv \eta^k$. $D_i = \dfrac{\partial}{\partial x^i} + u_i^k \dfrac{\partial}{\partial u^k} + u_{ij}^k \dfrac{\partial}{\partial u_i^k} + ...$ is a full derivative with respect to x_i.

Once a symmetry generator (2.4) is known, the global Lie transformation group (2.2) is reconstructed explicitly by solving the initial value problem

$$\frac{\partial f^i}{\partial \varepsilon} = \xi^i(f, g), \quad \frac{\partial g^k}{\partial \varepsilon} = \eta^k(f, g), \qquad f^i|_{\varepsilon=0} = x^i, \quad g^k|_{\varepsilon=0} = u^k. \tag{2.7}$$

To find all Lie group generators (2.4) admissible by the original system (2.1), one needs to solve linear determining equations

$$vG(x, u, \underset{1}{u}, ..., \underset{p}{u})|_{G(x,u,\underset{1}{u},...,\underset{p}{u})=0} = 0 \tag{2.8}$$

for $m + n$ unknown functions ξ^i, η^k that depend on $m + n$ variables (x, u). The determining equations (2.8) state the invariance of the solution manifold under the action of the vector field v.

The overdetermined system for ξ^i, η^k is obtained from (2.8) using the fact that ξ^i, η^k do not depend on derivatives u_i^k. Thus the system (2.8) splits into an $N \leq l(mn + 1)$ linear partial differential equations. Such overdetermined linear PDE systems can be rather large (containing hundreds or thousands of equations).

2. Contact and higher symmetries. When tangent vector field components ξ^i, η^k of the symmetry generator (2.5) essentially depend on derivatives,

$$\begin{aligned}(x')^i &= x^i + \epsilon\xi^i(x, u, \underset{1}{u}, ..., \underset{q}{u}) + O(\epsilon^2), \quad i = 1, \ldots, n,\\ (u')^j &= u^j + \epsilon\eta^j(x, u, \underset{1}{u}, ..., \underset{q}{u}) + O(\epsilon^2), \quad j = 1, \ldots, m,\end{aligned} \tag{2.9}$$

corresponding symmetries are called *higher symmetries* [1]. Unlike Lie symmetries, they act on the solution manifold Ω_∞ in the infinite-dimensional space $\mathcal{X} \times \mathcal{U} \times \ldots \times \mathcal{U}_p \times \ldots$ of all variables and derivatives. Due to this fact, higher symmetries, in general, can not be integrated to yield a global group representation similar to (2.2), and thus can not be used to explicitly map solutions of differential equations into new solutions.

Contact symmetries are symmetries of the form (2.9) that preserve the contact condition $du' = u'_j dx'_j$. Similarly to Lie point symmetries, contact symmetries act in the finite-dimensional space, and their global group representation can be reconstructed from the local representation. In the case of one equation, $m = 1$, tangent vector field components ξ^i, η of contact symmetry generators essentially depend on first derivatives. For $m > 1$, contact symmetries are independent of derivatives.

Contact and higher symmetries are computed algorithmically – corresponding determining equations are obtained from the requirement (2.8) that the given system of equations is invariant under the action of the prolonged vector field (2.5). The overdetermined system of linear PDEs for ξ^i, η^k is found by equating to zero all coefficients at derivatives of order higher than q, in the determining equations (2.8).

3. Approximate symmetries. In literature there exist several definitions of approximate symmetries, used by different authors.

In particular, Fushchich and Shtelen [16] define approximate symmetries as follows. For a given system containing a small parameter ε, write the solution as $u = u_1 + \varepsilon u_2$, substitute in the given system , and set to zero Taylor coefficients of each equation at powers $\varepsilon^0, \varepsilon^1$. Thus instead of the system G of l equations for m unknown functions, a new system G' of $2l$ equations for $2m$ unknown functions is obtained (it does not contain the small parameter ε). Point symmetries of the new system G' are called by Fushchich and Shtelen *approximate symmetries of the original system G.*

A similar but different definition of approximate symmetries was given by Baikov *et al* [17]. For a given system containing a small parameter ε, one seeks *generators of approximate symmetries* in the form (cf. (2.4))

$$X_\varepsilon \simeq X_1 + \varepsilon X_2, \tag{2.10}$$

where X_1 and X_2 do not involve ε. Components of X_1 and X_2 are found from an "approximate version" of determining equations (2.8):

$$v_\varepsilon G(x, u, \underset{1}{u}, ..., \underset{p}{u})|_{G(x,u,\underset{1}{u},...,\underset{p}{u})=O(\varepsilon^2)} = O(\varepsilon^2), \tag{2.11}$$

where $v_\varepsilon \simeq v_1 + \varepsilon v_2$ is a prolonged version of (2.10).

Other definitions of approximate symmetries also exist but are less related to goals and methods of Lie symmetry approach.

2.2 Methods for finding conservation laws of PDE systems

Consider the problem of finding conservation laws in the form

$$D_i \Phi^i[u] = 0 \tag{2.12}$$

that follow from the system (2.1). The conserved densities (fluxes) $\Phi^i[u]$ may depend on x, u and derivatives up to an arbitrary order.

A conservation law (2.12) is called *trivial* if its fluxes are

$$\Phi^i = M^i + H^i,$$

where M^i and H^i are sets of smooth functions such that M^i vanish on the solutions of the system (2.1), and $\mathrm{div}(H^1, ..., H^n) \equiv 0$. Two conservation laws $D_i\Phi^i[u] = 0$ and $D_i\Psi^i[u] = 0$ are *equivalent* if $D_i(\Phi^i[u] - \Psi^i[u]) = 0$ is a trivial conservation law. The more general 'triviality' idea is the notion of liner dependence of conservation laws. Conservation laws are *linearly dependent* if there exists a linear combination of them which is a trivial conservation law.

In practice, it is important to obtain nontrivial linearly independent conservation laws of a given system of differential equations.

Most of contemporary practically useful methods for finding conservation laws involve considering a linear combination of equations of the given system (2.1) with a set of *multipliers* (or factors, or characteristic functions) $\{\Lambda^p[U]\}$. Multipliers may depend on independent and dependent variables and derivatives. A linear combination yields a conservation law (2.12) if and only if

$$\Lambda^p[U]G_p(x, U, \underset{1}{U}, ..., \underset{p}{U}) = D_i\Phi^i[U] \tag{2.13}$$

for some set of fluxes $\{\Phi^i[U]\}$ (here U denotes a vector of arbitrary functions.) Then the conservation law $D_i\Phi^i[u] = 0$ holds on the solutions $U = u$ of the system (2.1).

We note that *each* conservation law (2.12) of the system (2.1) may be represented by a linear combination (2.13), if the given system is not degenerate (namely, is of Cauchy-Kovalevskaya type, see [1].)

The determining equations that yield sets of multipliers $\{\Lambda^p[U]\}$ of the conservation laws for the given system (2.1) are found from the known observation that an expression is a divergence expression if and only if it is annihilated by Euler operators with respect to all dependent variables in the system. The following theorem holds (see [1,2]):

Theorem 1 *A set of multipliers $\{\Lambda^\sigma[U]\}$ yields a conservation law of the given system of differential equations (2.1) if and only if the equations*

$$E_{U^k}(\Lambda^i[U]G_i(x, U, \underset{1}{U}, ..., \underset{p}{U})) = 0, \quad k = 1, \ldots, m, \tag{2.14}$$

where E_{U^k} is the Euler operator with respect to U^k:

$$E_{U^k} = \frac{\partial}{\partial U^k} - D_i\frac{\partial}{\partial U^k_i} + \cdots + (-1)^j D_{i_1}\cdots D_{i_j}\frac{\partial}{\partial U^k_{i_1\cdots i_j}} + \cdots \quad , \tag{2.15}$$

hold for arbitrary functions $U = (U^1(x), \ldots, U^m(x))$.

Equations (2.14) are linear determining equations for the multipliers for conservation laws admitted by the given system (2.1). In practice, to perform a computation, one chooses the maximal order of derivatives $q \geq 0$ in the dependence of multipliers $\Lambda^i[U] = \Lambda^i(x, U, \underset{1}{U}, ..., \underset{q}{U})$. Then, by setting to zero the coefficients of all derivatives of U higher than q, the determining system (2.14) is split into an overdetermined system of simpler linear PDEs.

After finding a set of multipliers $\{\Lambda^\sigma[U]\}$ that yield a conservation law (2.13), one can obtain, by an algorithmic procedure, the corresponding set of fluxes $\{\Phi^i[u]\}$ for each set of multipliers. This is achieved either by directly solving the system (2.13) for the unknown fluxes $\Phi^i[U]$, or using integral formulas arising from homotopy operators ([18–20].)

The described method does not require the PDE system under consideration to have a variational formulation, and does not use any version of Noether's theorem when the considered system is variational. The method is algorithmic and therefore practical. It yields all conservation laws with fluxes depending on a prescribed set of dependent and / or independent variables and derivatives.

Methods involving multipliers yield all conservation laws that involve derivatives up to a certain (prescribed) order. However, in some cases, conservation law analysis through multipliers may lead to finding *the complete set* of conservation laws. A relevant example is the case of evolution equations of even order, for which the maximal order of conservation law is finite. Alternatively, if a given system possesses a countable number of conservation laws, it may be possible to construct them recursively.

The use of adjoint linearization to find multipliers for conservation laws is demonstrated in the following subsection.

2.3 Symmetries, adjoint symmetries and conservation laws of PDE systems

A symmetry generator (2.4) can be equivalently represented in the *evolutionary form* (see e.g. [2]):

$$X_{ev} = \zeta^k[u]\frac{\partial}{\partial u^k}, \qquad \zeta^k[u] = \eta^k(x,u) - u_i^k\xi^i(x,u). \tag{2.16}$$

It defines a local coordinate transformation

$$\begin{aligned} &(x')^i = x^i, \quad (i = 1,\dots,n),\\ &(u')^k = u^k + \varepsilon\zeta^k[u] + O(\varepsilon^2) \quad (k = 1,\dots,m). \end{aligned} \tag{2.17}$$

From this fact it follows that if L is the linearizing operator (Fréchet derivative) for the given system (2.1), then the determining equations (2.8) are equivalent to

$$\mathrm{L}\zeta[u] = 0. \tag{2.18}$$

In other words, the operator X_{ev} (2.16) defines a symmetry if and only if its components solve the linearized version of the given system (2.1). (The linearization is understood in the Fréchet derivative sense, not as linearization near a particular solution.)

The given system (2.1), as written, has a *variational principle* (i.e. corresponds to an extremum of some action functional) if and only if L is a self-adjoint operator, i.e., $\mathrm{L}^* = \mathrm{L}$ where L^* is the *adjoint* of L. A set of functions $\kappa[u]$ defines *an adjoint symmetry* if these functions satisfy the adjoint linearized equation

$$\mathrm{L}^*\kappa[u] = 0. \tag{2.19}$$

One can show [1] that the set of multipliers $\{\Lambda^p[U]\}$ for the conservation law (2.13) satisfies the adjoint equations

$$\mathrm{L}^*\Lambda[u] = 0, \tag{2.20}$$

hence a conservation law always defines an adjoint symmetry. However the converse is not true, since the determining equations (2.14) for conservation law multipliers are more restrictive than the determining equations for adjoint symmetries (2.19). In particular, the set of determining equations for conservation law multipliers consists of the system of adjoint symmetry determining equations augmented by the extra determining equations [19].

3 "GeM" symbolic package for general Lie symmetry computation

3.1 Description

A `Waterloo Maple` - based package `GeM` has been recently developed by the author. The package routines are capable of finding local (Lie, contact and higher) symmetries, approximate symmetries in the sense of Fushchich [16], adjoint symmetries and conservation laws of any ODE/PDE system without significant limitations on DE order and number of variables, and without human intervention.

The routines of the module allow the analysis of ODE/PDE systems containing arbitrary constitutive functions and/or parameters. Automatic *symmetry and conservation law classification* with respect to constitutive functions is implemented. Classification problems naturally arise in the analysis of classes of DE systems from applications. Classification leads to isolation of particular forms of constitutive functions and/or parameter values for which the given DE system possesses an extended symmetry or conservation law structure. In particular, the discovery of infinite-dimensional sets of symmetries or conservation laws for particular forms of constitutive functions and/or parameter values may lead to linearization. For review and examples, see [2].

The `GeM` package employs an efficient representation of the system under consideration and resulting symmetry / conservation law determining equations: all dependent variables and derivatives are treated as `Maple` *symbols*, rather than functions or expressions. This significantly speeds up the computation of quantities corresponding to derivatives and the production of the overdetermined system of determining equations for the unknown tangent vector field coordinates / multipliers. The latter is reduced by `Rif` [13] package routines.

Reduction of overdetermined systems of symmetry / conservation law determining equations is usually the most resource-demanding task, in particular, in problems that involve classification. Reduced systems of determining equations are normally much simpler and are integrated automatically or even by hand.

In symmetry analysis, after the determining equations are solved, a `GeM` routine outputs all symmetry generators (2.4), thus completing the symmetry analysis. In conservation law analysis, after solving the determining equations and finding conservation law multipliers, a `GeM` routine computes fluxes and outputs all conservation laws in a canonical form (2.12).

Below we outline `Maple` program sequences for symmetry, adjoint symmetry, approximate symmetry and conservation law analysis using `GeM` routines. Run examples are presented in the end of the paper.

3.2 "GeM" routine sequence for symmetry analysis

In this section contains an outline the essential structure of a `Maple` program for local (Lie, contact and higher-order) symmetry analysis of PDE systems, and a description of related `GeM` package routines.

A. Initialization

```
with(GeM):

with(linalg): with(DEtools): with(PDEtools):
```

B. Declaration of variables and equations

The following command is used to declare dependent and independent variables, constitutive functions (optional), free constants (optional), and the given system of equations (ODEs/PDEs):

```
gem_init_defs([IV],[DV],[AF],[AC],
                    hi_der, [eqs],[solve_for] <,options>);
```

Here `[IV]` is a vector of independent variables, `[DV]` a vector of dependent variables, and `[AF]` and `[AC]` vectors of arbitrary (constitutive) functions and constants present in the system. `hi_der` is the integer (≥ 0) order of highest derivative of a dependent variable on which arbitrary functions listed in `[AF]` depend. `[eqs]` is a vector of differential equations in the form $E = 0$ or $A = B$; `[solve_for]` a vector of quantities (dependent variables or derivatives, usually highest derivatives) the equations are solved for. The differential equations of the system under consideration must be solvable for quantities specified in `[solve_for]`, otherwise an error is returned. The simpler the expressions for these quantities are, the more elegant equations on further steps will look. The sequence of quantities in `[solve_for]` does not have to match the sequence of equations in `[eqs]`.

Within this routine, all derivatives are defined as `Maple` symbols: $\partial B_1/\partial x \equiv$ `B1x`, etc. For example, the `Maple` expression for the equation involving a divergence of a 3-vector $\mathbf{B}(x, y, z) = (B1(x, y, z), B2(x, y, z), B3(x, y, z))$:

$$\frac{\partial}{\partial x}B1(x,y,z) + \frac{\partial}{\partial y}B2(x,y,z) + \frac{\partial}{\partial z}B3(x,y,z) = 0$$

is represented as

```
B1x + B2y + B3z = 0,
```

and dependent variables and partial derivatives become `Maple` symbols: `B1, B2, B3, B1x, B2y, B3z, ...` After this conversion, the unknown quantities of the problem (symmetry generator components) are not composite functions but functions of new “independent variables”, which now include dependent and independent variables of the system under consideration and all necessary partial derivatives.

When an *approximate symmetry* is being computed, the following *options* must be used: `symm_type=ApproxFS`, `small_param=epsilon`, where `epsilon` (or any other identifier) is a small parameter present in the given system of equations. The small parameter must be also included in the declaration of free constants: the list `[AC]`. According to the algorithm of Fushchich-Shtelen type approximate symmetry analysis, each dependent variable (e.g. `U`) is split in a combination (e.g. `U=U1+epsilon*U2`), and each equation is split into two (powers zero and one of the small parameter.) For each user-listed dependent variable (e.g. `U`), `GeM` declares two corresponding variables (e.g. `U1, U2`), which should be used in further analysis. For an example, see [15].

C. Generation of symmetry determining equations

The overdetermined (split) system of determining equations is obtained by the command

```
overdet_sys := gem_get_split_sys([symmetry_depends_on],

                max_der_order <,options>);
```

Here `[symmetry_depends_on]` is a list that specifies the dependence of tangent vector field components ξ^i, η^k. It may include independent variables, dependent variables, and derivatives of dependent variables. On this stage, one can introduce an ansatz, for example, assume independence of ξ^i, η^k of independent variables.) Ansatzes can be useful for complicated PDE systems for which complete symmetry analysis is not affordable.

The parameter `max_der_order` specifies maximal order of derivative present in `[symmetry_depends_on]`.

An optional parameter `in_ev_form=true` forces the computation of symmetries in evolutionary form. This is normally done in computing contact and higher symmetries (i.e. when `max_der_order>0`.)

D. Simplification of the overdetermined system. Classification

Overdetermined systems are often large and contain many redundant equations. In `Maple`, overdetermined PDE system simplification is effectively done using `Rif` package routines [13].

First, the names of unknown `GeM`-defined tangent vector field coordinates are stored in a program variable:

```
twf_coords:=gem_tvf_coords_get_();
```

Then the `Rif` package routine is called:

```
    simplified_sys:=rifsimp(overdet_sys, twf_coords):
```

After the execution of the above command, the reduced system of determining equations is found in the variable `simplified_sys[Solved]`.

Remark. If the given PDE system contains constitutive functions and/or parameters, it is necessary to perform a classification of all cases of constitutive functions and parameter values that give different results. This is achieved by running the `rifsimp` command with an option `casesplit`:

```
    split_sys:=rifsimp(overdet_sys,  twf_coords, casesplit);
```

The result is a Maple table of cases ("pivots") and equations on tangent vector field components for each case. A tree of cases is visualized, with pivots explicitly written, by the command

```
    caseplot(split_sys, pivots);
```

E. Solution of determining equations

The possibility of complete automatic integration of the reduced system of determining equations depends on the actual complexity of the symmetry structure of the DE system under consideration.

First we suggest to check the dimension of solution space:

```
    maxdimsystems(overdet_sys, twf_coords, output=rif)[dimension];
```

If the dimension is finite, the `Maple` built-in integrator is used for automated integration:

```
    symmetry_sol:=pdsolve(simplified_sys[Solved], twf_coords):
```

For systems with constitutive functions or arbitrary constants, one applies the same routine to one of the cases in the case tree:

```
    symmetry_sol:=pdsolve(split_sys[i][Solved], twf_coords):
```

where `i` is an integer between 1 and the number of cases in the case tree.

We note that though the `Maple` `pdsolve` routine is one of the best of its kind, it sometimes returns incomplete or unnecessarily complicated results, or no result, even when it is not difficult to obtain it by hand. However we enthusiastically recommend to use it, in conjunction with a completeness test. (The completeness test consists in checking the agreement of the number arbitrary constants in the solution returned by `pdsolve` and the number returned by the above `maxdimsystems` command.)

3.3 "GeM" routines for conservation law / adjoint symmetry analysis

A, B: Initialization, declaration of variables, functions and equations

Same as for symmetry analysis.

C. Generation of an overdetermined PDE system for conservation law multipliers / adjoint symmetry components

The following routine generates an overdetermined system of determining equations for conservation law multipliers / adjoint symmetry components:

```
    CLde:=gem_CL_gen_det_eqs([vars], hi_der <,options>);
```

The routine proceeds by computing determining equations (2.14) using Euler operators, and setting to zero the coefficients at all derivatives on which the multipliers do not depend.

Here `[vars]` is a vector that can include independent variables, dependent variables, and derivatives of dependent variables. It specifies the dependence of conservation law multipliers or adjoint symmetry components. `hi_der` is the integer (≥ 0) order of the highest derivative present in `[vars]`.

The only possible option is `on_solution_space=true`, which forces the *adjoint symmetry* computation. In this case, determining equations (2.14) are computed on the solution space of the given DE system. The default value of the option is `on_solution_space=false`, which requires the determining equations to be considered off solution space, and hence *multipliers of conservation laws* will be found (see Section 2.3.)

The returned set variable `CLde` contains the corresponding overdetermined system of determining equations.

D. Solution of determining equations. Canonical form

First, multipliers are stored in a program variable, using a command

```
Multipliers:= gem_CL_multipliers_get_();
```

Then the overdetermined system of determining equations is simplified and reduced to eliminate redundancy:

```
simplified_system:=rifsimp(CLde, Multipliers);
```
(3.1)

If the given PDE system contains constitutive functions, then instead of (3.1), one performs a classification, as follows:

```
split_system:=rifsimp(CLde, Multipliers, casesplit);
```

After classification, the resulting tree of cases ("pivots") can be plotted with the command

```
caseplot(split_system, pivots);
```

Second, the reduced system of determining equations `simplified_system[Solved]` (or, in the case of classification, the reduced system of determining equations for each case, `split_system[i][Solved]`) is solved. To use the built-in `Maple` solver `pdsolve`, one needs first to check the dimension of solution space by an appropriate command:

```
maxdimsystems(simplified_system[Solved],Multipliers,output=rif)[dimension];
```

or for one of the cases in the classification:

```
maxdimsystems(split_system[i][Solved],Multipliers,output=rif)[dimension];
```

where `i` is an integer between 1 and the number of cases.

If the returned dimension is finite, it is convenient to use `pdsolve`:

```
solved_mult:=pdsolve(simplified_system[Solved], Multipliers);
```
(3.2)

or for the cases in the classification:

```
solved_mult:=pdsolve(split_system[i][Solved], Multipliers);
```
(3.3)

The dimension of solution space returned by `maxdimsystems` is to be compared to the number of degrees of freedom in the solution obtained through `pdsolve`.

Remark. If the dimension of solution space returned by `maxdimsystems` is *infinite*, as well as in some other (rare) cases, the `pdsolve` routine returns only particular solutions. Such situations are to be considered individually. In particular, the presence of infinite-dimensional conservation laws or adjoint symmetries can imply that the given PDE system is *linearizable by an invertible mapping* [2].

If *adjoint symmetry analysis* is being undertaken, the procedure stops here, since adjoint symmetry components $\{\Lambda^p\}$ are found. In the case of *conservation law analysis*, after finding the multipliers $\{\Lambda^p\}$, the canonical form of the conservation law $D_i\Phi^i[u] = 0$ (2.12) is yet to be determined. The fluxes Φ^i are found through the conversion of an expression $\Lambda^p[u]G_p = 0$ to the form $D_i\Phi^i[u] = 0$, which in many cases is a simple computational exercise. Fluxes can be automatically found using the following routine provided in `GeM` :

```
gem_GetFluxes([solved_multipliers_set],max_der_order);    (3.4)
```

As input, this routine takes the set of multipliers `solved_mult` found in (3.2) or one of the cases (3.3), which are to be supplied in the place of the parameter `[solved_multipliers_set]`. The second parameter `max_der_order` specifies the highest order of derivative present in `[solved_multipliers_set]`.

This routine attempts to solve the condition $\Lambda^p[u]G_p = D_i\Phi^i[u]$ to find and output the general expression for fluxes. If multipliers `[solved_multipliers_set]` are linear combinations involving constants of integration, the fluxes are split accordingly, which results in a set of linearly independent nontrivial conservation laws.

4 Summary and further remarks

The `GeM` package for `Maple` presented in this paper contains a set of automated routines tools for local (Lie, contact and higher-order) and approximate symmetry analysis and classification, conservation law analysis and classification (through the determination of conservation law multipliers), and adjoint symmetry analysis classification, for systems of partial and ordinary differential equations. The package routines efficiently produce sets of determining equations that are optimized for further reduction and solution with `Waterloo Maple` internal routines. The integration results are presented as symmetry generators or conservation laws in the canonical form.

GeM is effectively used in the research of the author and his collaborators. Maple programs that use GeM are readily created for most PDE and ODE systems that arise in theoretical research and applications. Such programs are short, time-efficient, and demonstrate reliable behaviour in the analysis of DEs or DE systems involving various numbers of dependent / independent variables, and nonlinear DEs of higher order.

Further development of the package will proceed, in particular, in the following directions:

(1) Increasing work efficiency and speed in problems that involve more than two independent variables and (explicitly or implicitly) derivatives of high orders, 8 and above;
(2) Increasing the effectiveness and generality of integration of expressions of the type $M = \Lambda^p[U]G_p = 0$ to obtain the canonical form of the corresponding conservation law $D_i\Phi^i[U] = 0$, by using alternative methods (including homotopy operators);
(3) Implementation of automatic introduction of potential variables and generation of *potential systems* (nonlocally related to the given system). Implementation of automatic generation of *subsystems*, nonlocally related to the given system, by eliminating dependent variables. Creation of routines for automatic nonlocal symmetry / nonlocal conservation law analysis through the consideration of potential systems and subsystems nonlocally related to the given system [21].

Current version with various examples is available on the website [15]. New versions with additional features will be published on the same website.

Acknowledgements

The author thanks George Bluman, Stephen Anco, Nataliya Ivanova and the referee of this paper for useful discussion and suggestions, and the National Sciences and Engineering Council of Canada and the Killam Foundation for support.

References

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

[2] Bluman G.W., Kumei S. *Symmetries and Differential Equations.* Springer-Verlag (1989).

[3] *CRC handbook of Lie group analysis of differential equations. (Editor: N.H. Ibragimov).* Boca Raton, Fl.: CRC Press (1993).

[4] Bogoyavlenskij O. I., “Infinite symmetries of the ideal MHD equilibrium equations.” *Phys. Lett. A.* **291** (4-5), 256-264 (2001).

[5] Cheviakov A.F. and Bogoyavlenskij O.I., “Exact anisotropic MHD equilibria.” *J. Phys. A* **37**, 7593 (2004).

[6] Cheviakov A. F., “Construction of Exact Plasma Equilibrium Solutions with Different Geometries.” *Phys. Rev. Lett.* **94**, 165001 (2005).

[7] Lax P.D., “Integrals of nonlinear equations of evolution and solitary waves.” *Comm. Pure and Appl. Math.* **21**, 467-490 (1968).

[8] Benjamin T.B., “The stability of solitary waves.” *Proc. Roy. Soc. Lond.* **A** 328, 153-183 (1972).

[9] Knops R.J. and Stuart C.A., “Quasiconvexity and uniqueness of equilibrium solutions in nonlinear elasticity.” *Arch. Rat. Mech. Anal.* **86**, 234-249 (1984).

[10] Wolf T., “A comparison of four approaches to the calculation of conservation laws.” *EJAM* **13** (2) 129-152, (2002).

[11] Bluman G., “Connections Between Symmetries and Conservation Laws.” *SIGMA* **1** , Paper 011, 16 pages (2005).

[12] Hereman W., “Review of symbolic software for Lie symmetry analysis”, *Math. Comp. Mod.* **25**, 115-132 (1997).

[13] See `Waterloo Maple` help system, sections on `rifsimp`,`maxdimsys`, and related commands.

[14] Wolf T., “Crack, LiePDE, ApplySym and ConLaw, section 4.3.5 and computer program on CD-ROM” in: *Grabmeier, J., Kaltofen, E. and Weispfenning, V. (Eds.): Computer Algebra Handbook*, Springer, pp. 465-468 (2002).

[15] The `GeM` package and documantation is available at `http://www.math.ubc.ca/~alexch/gem/`.

[16] Fushchich W.I., Shtelen W.M., “On approximate symmetry and approximate solutions of the non-linear wave equation with a small parameter.” *J. Phys. A* **22**, L887-L890 (1989).

[17] Baikov V.A., Gazizov R.K. and Ibragimov N. H. *J. Sov. Math.* **55**, 14501455 (1991).

[18] Anco S. and Bluman G., “Direct construction method for conservation laws of partial differential equations. Part I: Examples of conservation law classfications.” *European J. Appl. Math.,* , V.13, 545-566 (2002).

[19] Anco S. and Bluman G., “Direct construction method for conservation laws of partial differential equations. Part II: General treatment.” *European J. Appl. Math.* **13**, 567-585 (2002).

[20] Hereman W., “Symbolic Computation of Conservation Laws of Nonlinear Partial Differential Equations in Multi-dimensions.” Preprint (2006).

[21] Bluman G. and Cheviakov A. F. “Framework for potential systems and nonlocal symmetries: Algorithmic approach”. *Journal of Mathematical Physics* **46**, 123506 (2005).

TEST RUN OUTPUT

This section contains code and output for several examples:

(a) Higher symmetry analysis of Burgers' equation (Example 1);
(b) Classification of local conservation laws of Nonlinear Telegraph equations, with respect to two constitutive functions (Example 2);
(c) A comparison of point symmetries and adjoint symmetries of a nonlinear equation (Example 3).

Example A. Higher symmetries of Burgers' equation

As en example, we present Maple code and output of a program that computes higher-order symmetries of Burgers equation. Following [2] (chapter 5.2), we seek symmetries of Burgers' equation

$$u_t + uu_x = u_{xx} \tag{4.1}$$

that depend on $(x, t, u_x, u_{xx}, u_{xxx})$, in the evolutionary form (2.16).

The Maple program has the form:

```
restart;
with(GeM): with(linalg): with(DEtools): with(PDEtools):
gem_init_defs([x,t], [U(x,t)],[ ],[ ],0,
[diff(U(x,t),t)+U(x,t)*diff(U(x,t),x) =diff(U(x,t),x,x)],[diff(U(x,t),t)]);
overdet_sys:=gem_get_split_sys(
[x,t,U(x,t),diff(U(x,t),x),diff(U(x,t),x,x),
diff(U(x,t),x,x,x)],3,in_ev_form=true);
twf_coords:=gem_tvf_coords_get_();
simplified_system:=rifsimp(overdet_sys, twf_coords):
#Verify dimension of solution space:
maxdimsystems(overdet_sys, twf_coords, output=rif)[dimension];
#Solve for symmetry generator:
SymmetrySolution:=pdsolve(simplified_system[Solved], twf_coords):
#Print 9 independent symmetry generators:
gem_GenAllX(SymmetrySolution,9);
```

The computation of the nine-dimensional Lie algebra of symmetry generators takes several seconds. The output symmetry generators are as follows (cf. [2]):

$$
\begin{aligned}
X_1 = {} & \Big(\frac{4}{3}u_{xxx}t^3 + (-2ut^3 + 2t^2x)u_{xx} - 2u_x^2t^3 + (-2xut^2 + u^2t^3 + x^2t + 4t^2)u_x \\
& -2t - u^2t^2 + 2xtu - x^2\Big)\,\frac{\partial}{\partial u}; \\
X_2 \quad & = \Big(2u_{xxx}t^2 + (-3t^2u + 2tx)u_{xx} - 3u_x^2t^2 \\
& + \left(\frac{3}{2}u^2t^2 - 2xut + 3t + \frac{1}{2}x^2\right)u_x + xu - u^2t\Big)\,\frac{\partial}{\partial u}; \\
X_3 \quad & = \Big(4u_{xxx}t + (-6ut + 2x)u_{xx} - 6ux^2t + (-2ux + 3u^2t)u_x - u^2\Big)\,\frac{\partial}{\partial u}; \\
X_4 \quad & = \Big(xu_xt - uu_xt^2 - x + u_{xx}t^2 + ut\Big)\,\frac{\partial}{\partial u}; \\
X_5 \quad & = (u - 2uu_xt + 2u_{xx}t + xu_x)\,\frac{\partial}{\partial u}; \\
X_6 \quad & = (-uu_x + u_{xx})\,\frac{\partial}{\partial u}; \\
X_7 \quad & = \left(-\frac{2}{3}u_{xxx} + u_x^2 + uu_{xx} - \frac{1}{2}u_xu^2\right)\frac{\partial}{\partial u}; \\
X_8 \quad & = (1 - u_xt)\frac{\partial}{\partial u}; \\
X_9 \quad & = u_x\frac{\partial}{\partial u}.
\end{aligned}
$$

Example B. Classification of local conservation laws of Nonlinear Telegraph equations

We consider Nonlinear Telegraph (NLT) equations

$$u_{tt} - (F(u)u_x)_x - (G(u))_x = 0 \tag{4.2}$$

for general $F(u), G(u)$, and with power nonlinearities $F(u) = u^\alpha$, $G(u) = u^\beta$ $(\alpha, \beta \neq 0)$.

We perform a classification of conservation laws of these equations for multipliers of the form $\Lambda = \Lambda(x, t, U)$.

Fig. 1. Tree of cases of constitutive functions $F(u)$, $G(u)$ in the classification of conservation laws of the NLT equations (4.2)

The `Maple` code for conservation law classification is

```
restart;
with(GeM): with(linalg): with(DEtools): with(PDEtools):
gem_init_defs([x,t], [U(x,t)],[F(U(x,t)),G(U(x,t))],[ ],0,
[diff(U(x,t),t,t)=diff((F(U(x,t))*diff(U(x,t),x) + G(U(x,t))),x)],
[diff(U(x,t),t,t)]);
#Generate and print conservation law determining equations.
CLde:=gem_CL_gen_det_eqs([t,x,U(x,t)],0);
Multipliers:=gem_CL_multipliers_get_();
#Simplify and split cases
split_system:=rifsimp(CLde union {F(U)<>0},Multipliers,casesplit);
caseplot(split_system,pivots);
```

The output of the last command, showing the tree of cases in the conservation law classification, is shown on Fig. 1.

We now explicitly find conservation laws for all cases, restricting ourselves to power nonlinearities $F(u) = u^{\alpha}$, $G(u) = u^{\beta}$ $(\alpha, \beta \neq 0)$.

The most general Case 1 holds for all functions $F(u), G(u)$, in particular, for arbitrary powers $F(u) = u^{\alpha}$, $G(u) = u^{\beta}$ $(\alpha, \beta \neq 0)$. We compute and output multipliers and fluxes of conservation laws with the following commands:

```
mm:=1;
maxdimsystems(split_system[mm][Solved],Multipliers);
lambdas_solved:=pdsolve(split_system[mm][Solved]);
gem_GetFluxes(lambdas_solved,0);
```

The output multipliers and fluxes of conservation laws are given in Table 1.

For the power nonlinearities, two possibilities exist in which Case 2 on Fig. 1 is realized: (2a) $F(u) = (\alpha+1)u^\alpha, G(u) = u^{\alpha+1}$ and (2b) $F(u) = (\alpha+1)u^\alpha, G(u) = u$ $(\alpha \neq -1)$. We compute and output multipliers and fluxes of conservation laws with the following sets of commands.

For the case (2a):

```
mm:=2;
maxdimsystems(split_system[mm][Solved],Multipliers);
F(U):=(alpha+1)*U^alpha:G(U):=U^(alpha+1);
lambdas_solved:=pdsolve(split_system[mm][Solved]);
gem_GetFluxes(lambdas_solved[1],0);
```

For the case (2b):

```
mm:=2;
maxdimsystems(split_system[mm][Solved],Multipliers);
F(U):=(alpha+1)*U^alpha:G(U):=U;
lambdas_solved:=pdsolve(split_system[mm][Solved]);
gem_GetFluxes(lambdas_solved[1],0);
```

For each of the cases (2a) and (2b), two new conservation laws are found. The output multipliers and fluxes are given in Table 1.

In Case 3, $F(u) = \text{const}, G(u) = u^{\alpha+1}$, no new conservation laws compared to Case 1 are found. Case 4 is linear and hence is not considered.

Table 1: Classification of Local Conservation Laws $D_tT + D_xX = 0$ of NLT equations (4.2)

$F(u)$	$G(u)$	Multiplier	T	X
Arbitrary	Arbitrary	$\Lambda = 1$ $\Lambda = t$	u_t $tu_t - u$	$F(u)u_x + G(u)$ $t(F(u)u_x + G(u))$
$(\alpha+1)u^\alpha$	$u^{\alpha+1}$	$\Lambda = e^x$ $\Lambda = te^x$	$e^x u_t$ $e^x(tu_t - u)$	$e^x(u^{\alpha+1})_x$ $te^x(u^{\alpha+1})_x$
$(\alpha+1)u^\alpha$	u	$\Lambda = x - \frac{t^2}{2}$ $(\Lambda = xt - \frac{t^3}{6}$	$\left(x - \frac{t^2}{2}\right) u_t + ut$ $\left(tx - \frac{t^3}{6}\right) u_t - u\left(x - \frac{t^2}{2}\right)$	$\left(x - \frac{t^2}{2}\right)((u^{\alpha+1})_x + u) - u^{\alpha+1}$ $\left(tx - \frac{t^3}{6}\right)((u^{\alpha+1})_x + u) - tu^{\alpha+1}$

The full `Maple` program for the above example is available in [15].

Example C. A comparison of point symmetries and adjoint symmetries

In this example, we follow [11] to compare symmetries and adjoint symmetries (in evolutionary form) of a nonlinear PDE equation

$$u_{tt} + H'(u_x)u_{xx} + H(u_x) = 0, \tag{4.3}$$

for the case $H(y) = y^4$.

The following program computes symmetries of (4.3) in evolutionary form (note the option in the call to `gem_get_split_sys`):

```
restart;
with(GeM): with(linalg): with(DEtools): with(PDEtools):
gem_init_defs([t,x], [U(x,t)],[],[],0,
[diff(U(x,t),t,t)+diff(HH(diff(U(x,t),x)),x)+HH(diff(U(x,t),x))=0],
[diff(U(x,t),t,t)]);
#Generate symmetry determining equations in evolutionary form:
overdet_sys:=gem_get_split_sys([x,t, U(x,t),diff(U(x,t),x),diff(U(x,t),t)],
   1,in_ev_form=true);
twf_coords:=gem_tvf_coords_get_();
#Simplify:
simplified_system:=rifsimp(overdet_sys, twf_coords):
#Dimension of solution space:
maxdimsystems(overdet_sys, twf_coords, output=rif)[dimension];
#Symmetry tangent vector field coordinates:
SymmetrySolution:=expand(pdsolve(simplified_system[Solved], twf_coords)):
#Output:
gem_GenAllX(SymmetrySolution,5);
```

The output symmetry generator is

$$X = \left(C_1 + C_2 t + C_3 u_x + C_4 u_t + C_5 \left(t u_t + \frac{2}{3} u\right)\right) \frac{\partial}{\partial u}.$$

We now compute adjoint symmetries of (4.3):

```
restart;
with(GeM): with(linalg): with(DEtools): with(PDEtools):
gem_init_defs([t,x], [U(x,t)],[],[],0,
[diff(U(x,t),t,t)+diff(HH(diff(U(x,t),x)),x)+HH(diff(U(x,t),x))=0],
[diff(U(x,t),t,t)]);
#Generate adjoint symmetry determining equations:
ASde:=gem_CL_gen_det_eqs([x,t, U(x,t),diff(U(x,t),x),diff(U(x,t),t)],
    1,  on_solution_space=true);
AdjSymmComp:= gem_CL_multipliers_get_();
#Simplify:
simplified_system:=rifsimp(ASde, AdjSymmComp);
#Dimension of solution space:
maxdimsystems(simplified_system[Solved],AdjSymmComp,
    output=rif)[dimension];
#Adjoint symmetry vector field coordinates:
solved_adjsym:=pdsolve(simplified_system[Solved], AdjSymmComp);
```

The resulting adjoint symmetry generator is

$$X^* = e^x \left(C_1 + C_2 t + C_3 u_t + C_4 \left(t u_t + \frac{2}{3} u \right) \right) \frac{\partial}{\partial u},$$

which does not coincide with X. This shows that the linearization of the equation (4.3) as it stands is not self-adjoint (which can also be checked directly), and hence the equation (4.3) does not follow from a variational principle.

However, for the equation

$$e^x \left(u_{tt} + H'(u_x) u_{xx} + H(u_x) \right) = 0, \tag{4.4}$$

its linearization and adjoint linear system coincide. Analogous computation can be used to check that indeed point symmetry and adjoint symmetry generators also coincide. Therefore (4.4) does follow from a variational principle. This example demonstrates that the property of existence of a variational formulation for a nonlinear PDE is not "invariant" in a reasonable sense, and therefore is artificial.