Description:
GeM symbolic software package (version 0.31+)
for general symmetry and conservation law analysis
of differential equations

© Alexei F. Cheviakov,
Department of Mathematics and Statistics, University of Saskatchewan, Saskatoon, S7N 5E6 Canada

December 22, 2009

Abstract

This document gives a brief introduction to algorithms and notation for symmetry and conservation law analysis of differential equations (DE), and describes corresponding routines of the `Maple` (v.12+)– based package `GeM` (v.0.31).

The examples are not reviewed in detail, but are available as Maple programs on the same webpage.

1 Symmetry and conservation law analysis of DEs

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 and ordinary and partial differential equations. Contact and higher-order symmetries are computed in a similar algorithmic manner. Closely related are methods for discovering nonlocal symmetries of differential equations. The framework further extends on symmetries of integro-differential equations, symmetries of difference equations, and various kinds of approximate symmetries of differential equations.

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 analysis, in particular, existence, uniqueness and stability analysis, study of analytical (including geometrical) properties of solutions of PDE systems, and construction of effective numerical schemes. Conservation law structure contains information about the possibility of linearization of the given PDE system by an invertible mapping and allows to construct the explicit mapping. Moreover, conservation laws of PDE systems yield nonlocally-related PDE systems, which in turn may yield new analytical results for the given system, in particular, new nonlocal symmetries, nonlocal conservation laws, or linearization.

For applications, examples, and further details on symmetries and conservation laws, the reader is referred to [1–4] and references therein. In particular, the notation adopted below follows that of the book [3].

In general, we consider a system of N differential equations (DEs) of order k with n independent variables $x = (x^1, \ldots, x^n)$ and m dependent variables $u(x) = (u^1(x), \ldots, u^m(x))$, given by

$$R^\sigma(x, u, \partial u, \ldots, \partial^k u) = 0, \quad \sigma = 1, \ldots, N. \tag{1.1}$$

The notation

$$\partial u \equiv \partial^1 u = \Big(u^1_1(x), \ldots, u^1_n(x), \ldots, u^m_1(x), \ldots, u^m_n(x) \Big)$$

denotes the set of all first-order partial derivatives;

$$\begin{aligned} \partial^p u &= \Big\{ u^\mu_{i_1 \ldots i_p} \mid \mu = 1, \ldots, m; \quad i_1, \ldots, i_p = 1, \ldots, n \Big\} \\ &= \left\{ \frac{\partial^p u^\mu(x)}{\partial x^{i_1} \ldots \partial x^{i_p}} \mid \mu = 1, \ldots, m; i_1, \ldots, i_p = 1, \ldots, n \right\} \end{aligned}$$

denote higher-order derivatives.

1.1 Point symmetries

Consider a one-parameter Lie group of point transformations

$$(x^*)^i = f^i(x, u; \varepsilon), \tag{1.2a}$$
$$(u^*)^\mu = g^\mu(x, u; \varepsilon), \tag{1.2b}$$

with the corresponding infinitesimal generator

$$\mathrm{X} = \xi^i(x, u) \frac{\partial}{\partial x^i} + \eta^\mu(x, u) \frac{\partial}{\partial u^\mu}. \tag{1.3}$$

The kth extension (prolongation) of (1.3) is given by

$$\begin{aligned} \mathrm{X}^{(k)} = \quad & \xi^i(x, u) \frac{\partial}{\partial x^i} + \eta^\mu(x, u) \frac{\partial}{\partial u^\mu} + \eta^{(1)\,\mu}_i(x, u, \partial u) \frac{\partial}{\partial u^\mu_i} \\ & + \ldots + \eta^{(k)\,\mu}_{i_1 \ldots i_k}(x, u, \partial u, \ldots, \partial^k u) \frac{\partial}{\partial u^\mu_{i_1 \ldots i_k}}, \end{aligned} \tag{1.4}$$

where the prolonged components $\eta^{(1)\,\mu}_i$, …, $\eta^{(k)\,\mu}_{i_1 \ldots i_k}$ are defined in terms of $\{\xi^i(x, u), \eta^\mu(x, u)\}$ by

$$\eta^{(1)\,\mu}_i = \mathrm{D}_i \eta^\mu - (\mathrm{D}_i \xi^j) u^\mu_j, \tag{1.5}$$

and

$$\eta^{(k)\,\mu}_{i_1 \ldots i_k} = \mathrm{D}_{i_k} \eta^{(k-1)\,\mu}_{i_1 \ldots i_{k-1}} - (\mathrm{D}_{i_k} \xi^j) u^\mu_{i_1 \ldots i_{k-1} j}, \tag{1.6}$$

for $\mu = 1, \ldots, m$, and $i, i_j = 1, \ldots, n$ for $j = 1, \ldots, k$, and

$$\mathrm{D}_i = \frac{\partial}{\partial x^i} + u^\mu_i \frac{\partial}{\partial u^\mu} + u^\mu_{i i_1} \frac{\partial}{\partial u^\mu_{i_1}} + u^\mu_{i i_1 i_2} \frac{\partial}{\partial u^\mu_{i_1 i_2}} + \cdots, \tag{1.7}$$

$i = 1, \ldots, n$.

A one-parameter Lie group of point transformations (1.2) *leaves the PDE system* $\mathbf{R}\{x\,; u\}$ (1.1) *invariant* (and hence is *a point symmetry*) if and only if its kth extension (1.4) leaves invariant the

solution manifold of $\mathbf{R}\{x\,;u\}$ in $(x,u,\partial u,\ldots,\partial^k u)$-space. This is the case if and only if for each $\alpha=1,\ldots,N$,

$$\mathrm{X}^{(k)}R^\alpha(x,u,\partial u,\ldots,\partial^k u)=0, \tag{1.8}$$

when

$$R^\sigma(x,u,\partial u,\ldots,\partial^k u)=0,\quad \sigma=1,\ldots,N. \tag{1.9}$$

In order to find point symmetries admitted by a given PDE system (1.1), one needs to determine the tangent vector field coordinates ξ^i,η^μ of the symmetry generator (1.3).

The PDE system for ξ^i,η^μ is obtained from determining equations (1.8) by first substituting the conditions (1.9) (to restrict to solution manifold), and then using the fact that ξ^i,η^μ do not depend on derivatives of u^μ. Thus one obtains an overdetermined system of linear PDEs for the unknown for ξ^i,η^μ. Such systems can be rather large. For example, for the system of adiabatic compressible Plasma Equilibrium equations, one obtains 188 linear PDEs for 13 unknown functions [9].

The symmetry generator (1.3) can be equivalently rewritten in the *evolutionary (characteristic) form*

$$\hat{\mathrm{X}}=\left[\eta^\mu(x,u)-u^\mu_i\xi^i(x,u)\right]\frac{\partial}{\partial u^\mu}. \tag{1.10}$$

1.2 Contact and higher-order symmetries

When tangent vector field components ξ^i,η^μ of the symmetry generator (1.3) essentially depend on derivatives,

$$\begin{aligned}&(x^*)^i=x^i+\varepsilon\xi^i(x,u,\partial u,...,\partial^q u)+O(\varepsilon^2),\quad i=1,\ldots,n,\\&(u^*)^\mu=u^\mu+\varepsilon\eta^\mu(x,u,\partial u,...,\partial^q u)+O(\varepsilon^2),\quad \mu=1,\ldots,m,\end{aligned} \tag{1.11}$$

corresponding symmetries are called *higher-order symmetries*. Unlike Lie symmetries, they act on the solution manifold in the infinite-dimensional space of all variables and derivatives. Due to this fact, higher-order symmetries, in general, cannot be integrated to yield a global group representation similar to (1.2), and thus cannot be used to explicitly map solutions of differential equations into new solutions.

Contact symmetries are symmetries of the form (1.11) 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, $N=1$, tangent vector field components of contact symmetry generators essentially depend on first derivatives. For $N>1$ equations, contact symmetries are independent of derivatives and are equivalent to point symmetries.

Contact and higher-order symmetries are computed the same way as point symmetries. In particular, determining equations are still given by (1.8) with (1.9). The overdetermined system of linear PDEs for ξ^i,η^μ is found by equating to zero, in the determining equations, all coefficients at derivatives of order higher than q.

Note that to restrict to the solution space of the given system, one now has to substitute into (1.8) not only the equations (1.9) but also all their differential consequences.

Without loss of generality, the symmetry generators for higher-order symmetries are usually sought in the evolutionary (characteristic) form

$$\hat{\mathrm{X}}=\zeta^\mu(x,u,\partial u,...,\partial^q)\frac{\partial}{\partial u^\mu}, \tag{1.12}$$

and thus $\xi^i(x,u,\partial u,...,\partial^q u)=0$ is assumed.

1.3 Conservation laws

Consider a problem of finding divergence-type *conservation laws* in the form

$$D_i\Phi^i(x,u,\partial u,...,\partial^q u)=0 \tag{1.13}$$

that hold for the system (1.1). The conserved densities (fluxes) Φ^i may depend on x,u and derivatives up to an arbitrary order.

A conservation law (1.13) 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 (1.1), and $\mathrm{div}(H^1,...,H^n)\equiv 0$. Two conservation laws $D_i\Phi^i=0$ and $D_i\Psi^i=0$ are *equivalent* if $D_i(\Phi^i-\Psi^i)=0$ 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, one seeks nontrivial linearly independent local conservation laws of a given system of differential equations.

The direct method of construction of conservation laws.

In general, for a given PDE system (1.1), nontrivial local conservation laws arise from linear combinations of the equations of the PDE system (1.1) with *multipliers* that yield nontrivial divergence expressions. In seeking such expressions, the dependent variables (and their derivatives) that arise in the PDE system (1.1), or appear in the multipliers, are replaced by arbitrary functions (and their derivatives). By their construction, such divergence expressions vanish on all solutions of the PDE system (1.1).

A set of multipliers $\{\Lambda_\sigma[U]\}_{\sigma=1}^N=\{\Lambda_\sigma(x,U,\partial U,\dots,\partial^l U)\}_{\sigma=1}^N$ yields a divergence expression for the PDE system $\mathbf{R}\{x\,;u\}$ (1.1) if the identity

$$\Lambda_\sigma[U]R^\sigma[U]\equiv \mathrm{D}_i\Phi^i[U] \tag{1.14}$$

holds for *arbitrary* functions $U(x)$. Then on the solutions $U(x)=u(x)$ of the PDE system (1.1), if $\Lambda_\sigma[U]$ is non-singular, one has a local conservation law

$$\Lambda_\sigma[u]R^\sigma[u]=\mathrm{D}_i\Phi^i[u]=0. \tag{1.15}$$

A multiplier $\Lambda_\sigma[U]$ is *singular* if it is a singular function when evaluated on solutions $U(x)=u(x)$ of the given PDE system (1.1). [In practice, one is only interested in non-singular sets of multipliers, since considering singular multipliers can lead to arbitrary divergence expressions that are not conservation laws of the given system. For example, $\Lambda_\sigma[U]=\mathrm{D}_i\Phi^i[U]/R^\sigma[U]$ yields $\Lambda_\sigma[U]R^\sigma[U]\equiv \mathrm{D}_i(N\Phi^i[U])$, in terms of arbitrary functions $\Phi^1[U],\dots,\Phi^n[U]$.]

The *Euler operator* with respect to U^j is the operator defined by

$$\mathrm{E}_{U^j}=\frac{\partial}{\partial U^j}-\mathrm{D}_i\frac{\partial}{\partial U^j_i}+\dots+(-1)^s\mathrm{D}_{i_1}\dots\mathrm{D}_{i_s}\frac{\partial}{\partial U^j_{i_1\dots i_s}}+\dots \tag{1.16}$$

for each $j=1,\dots,m$.

It is well known that the Euler operators (1.16) annihilate *any* divergence expression $\mathrm{D}_i\Phi^i[U]$. In particular, the following identities hold for arbitrary $U(x)$:

$$\mathrm{E}_{U^j}(\mathrm{D}_i\Phi^i(x,U,\partial U,\dots,\partial^r U))\equiv 0,\quad j=1,\dots,m. \tag{1.17}$$

The converse also holds.

It follows that in order to find non-singular local conservation law multipliers $\{\Lambda_\sigma(x, U, \partial U, \ldots, \partial^l U)\}_{\sigma=1}^N$ for the PDE system $\mathbf{R}\{x\,;u\}$ (1.1), one needs to solve *multiplier determining equations*

$$\begin{aligned} &\mathrm{E}_{U^j}(\Lambda_\sigma(x, U, \partial U, \ldots, \partial^l U)R^\sigma(x, U, \partial U, \ldots, \partial^k U)) \equiv 0, \\ &j = 1, \ldots, m, \end{aligned} \tag{1.18}$$

holding for *arbitrary functions* $U(x)$.

The set of equations (1.18) yields the set of *linear* determining equations to find *all* sets of local conservation law multipliers of the PDE system $\mathbf{R}\{x\,;u\}$ (1.1) by considering multipliers of all orders $l = 1, 2, \ldots$. Since equations (1.18) hold for arbitrary $U(x)$, it follows that one can treat each U^μ and each of its derivatives U^μ_i, U^μ_{ij}, etc. as independent variables along with x^i. Consequently the linear PDE system (1.18) splits into an over-determined linear system of determining equations for unknown local multipliers Λ_σ.

Computation of fluxes.

After finding a set of multipliers $\{\Lambda^\sigma\}$ that yield a conservation law (1.13), one can obtain the corresponding set of fluxes $\{\Phi^i\}$ using one of the available methods. For details, see [3, 5]. `GeM` package offers four methods of flux computation.

1. Direct method. Here one simply writes down PDEs (1.15) and solves them for the set of fluxes $\{\Phi^i\}$. This method is most straightforward for conservation laws arising for simple PDE systems from simple forms of multipliers. The corresponding `GeM` routine uses `Maple` `pdsolve` to solve (1.15) for fluxes.

2. The first homotopy (integral) formula. This formula is due to Bluman and Anco. It is described in detail in [5]. The formula involves integrations, and also an arbitrary function $\widetilde{U}(x)$, chosen manually so that the integral converges. Different choices of $\widetilde{U}$ yield fluxes of equivalent conservation laws. One normally chooses $\widetilde{U} = 0$ (provided that the integral converges). This formula is rather powerful and is a preferred formula for complicated PDE systems and conservation law multipliers which *do not* involve arbitrary (constitutive) functions.

3. The second homotopy (integral) formula. This formula is due to Hereman et al. It is in many ways similar to the first homotopy formula; it does not involve an arbitrary function; it sometimes produces simpler fluxes than the first homotopy formula. The only restriction in applying the second homotopy formula is that the expression

$$f(x, U, ...) = \mathrm{D}_1\Phi^1(x, U, ...) + \ldots + \mathrm{D}_n\Phi^n(x, U, ...),$$

has to satisfy the condition $f[0] = 0$, which is *not* the case for some conservation laws.

4. The scaling formula. This formula is due to Anco. If the given PDE system that has a scaling symmetry

$$\mathrm{X}_s[u] = p^{(i)}x^i\frac{\partial}{\partial x^i} + q^{(\rho)}u^\rho\frac{\partial}{\partial u^\rho}, \tag{1.19}$$

Suppose a given conservation law is scaling-invariant, and moreover, homogeneous under the scaling symmetry (1.19), i.e.,

$$\mathrm{X}^{(l)}_s[U]\mathrm{D}_i\Phi^i[U] = P\mathrm{D}_i\Phi^i[U] \tag{1.20}$$

(which is often the case for physical systems).

When a given PDE system is scaling-invariant and the conservation law is scaling-homogeneous, then the scaling symmetry method is the method of choice, since it involves simplest computations (no integration is required!). Moreover, this formula yields a result even when given PDEs and/or multipliers involve arbitrary functions. In this situation, the scaling symmetry method is the only systematic method available.

If the given system admits several scaling symmetries (1.19), one should use one that involves only (or mostly) components for the dependent variables, since the form of fluxes will be the simplest in that case.

The Table 1 is borrowed from [5].

Table 1: Comparison of Four Methods of Flux Computation.

Method	Applicability	Computational complexity
Direct	Simpler multipliers/PDE systems, which may involve arbitrary functions.	Solution of an overdetermined linear PDE system for fluxes.
Homotopy 1	Complicated multipliers/PDEs, not involving arbitrary functions.	One-dimensional integration.
Homotopy 2	Complicated multipliers/PDEs, not involving arbitrary functions. The divergence expression must vanish for $U = 0$. For some conservation laws, this method can yield divergent integrals.	One-dimensional integration.
Scaling	Complicated multipliers/PDEs, may involve arbitrary functions. Scaling-homogeneous PDEs and multipliers. Nontrivial conservation laws.	Repeated differentiation.

2 "GeM" symbolic package: Description

A `Maple` - based package `GeM` has been recently developed by the author. The package routines are capable of finding local (Lie, contact and higher-order) symmetries, 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 or parameters. Such symmetry/conservation law classification problems naturally arise in 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.

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: $\partial B_1/\partial x \equiv$ `B1x`, etc. For example, the resulting `Maple` expression for the divergence of a 3-vector $\mathbf{B}(x, y, z) = (B1(x, y, z), B2(x, y, z), B3(x, y, z))$ becomes

$$\frac{\partial}{\partial x} B1(x, y, z) + \frac{\partial}{\partial y} B2(x, y, z) + \frac{\partial}{\partial z} B3(x, y, z) = \texttt{B1x + B2y + B3z = 0}. \tag{2.1}$$

This significantly speeds up the computation involving establishing, splitting and solution of symmetry and conservation law determining equations. The final overdetermined linear PDE systems are reduced using `Maple rifsimp` [6] routine.

The 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 (`Maple pdsolve`, `dsolve`) or even by hand.

In symmetry analysis, after the determining equations are solved, a `GeM` routine is called that outputs all symmetry generators, thus completing the symmetry analysis.

In conservation law analysis, after the determining equations are solved and conservation law multipliers are found, another `GeM` routine is used to compute fluxes and output them in the canonical form.

Below, `Maple` program sequences for symmetry and conservation law analysis using `GeM` are outlined.

The program sequence for local symmetry or conservation law analysis is similar for most symbolic packages and typically includes the following steps.

1. Declaration of variables and the given PDE system.
2. Construction of a set of symmetry or conservation law determining equations.
3. If necessary (e.g., for finding symmetries or solutions of the adjoint linearized system), in the set of determining equations, the dependent variables arising from the given PDE system are restricted to solutions of the given PDE system.
4. Simplification (e.g., elimination of redundancies, partial solution) of the over-determined set of determining equations.
5. Solution of the simplified set of determining equations. Output of the point symmetries or conservation law multipliers.
6. For conservation laws: generation of fluxes.

If the given PDE system contains constitutive function(s) and/or constant parameter(s), a classification and case splitting is performed at Step 4, and Steps 5 and 6 are performed separately for each case that arises.

Run examples are given in separate `Maple` worksheets on the web page [8].

3 Program sequence for symmetry analysis

1. Clear the variables and read in the `GeM` module as a `Maple` text input file.

```
restart:

read(d:/gem31.txt):
```

Here `d:/` denotes the full path to the file.

2. Declare variables and arbitrary functions/constants.

```
gem_decl_vars(indeps=[...], deps=[...], freefunc=[...], freeconst=[...]);
```

where [...] denotes a `Maple` list of objects.

- `indeps`: a list of independent variables. For DEs, a list of one entry.
- `deps`: a list of dependent variables.

- `freefunc`: a list of arbitrary constitutive functions present in the given DE system (may depend on independent variables, dependent variables, and derivatives of dependent variables).
- `freeconst`: a list of arbitrary constants present in the given DE system.

3. Declare the given PDE system.

```
gem_decl_eqs([...], solve_for=[...]);
```

- First list `[...]`: A set of equations (ODE or PDE) involving ONLY the above-defined independent and dependent variables, arbitrary constitutive functions, arbitrary constants.
- For symmetry analysis, it is necessary that a given PDE system can be written in a solved form with respect to a set of leading derivatives specified in the `solve_for` parameter.

4. Generate symmetry determining equations

```
det_eqs:=gem_symm_det_eqs([...]);
```

- The name in the left-hand side can be arbitrary; the corresponding `Maple` variable will contain the determining equations.
- The list `[...]` contains the dependence of symmetry components. Can include independent and dependent variables (for point symmetries), and also derivatives of dependent variables (for higher-order symmetries).
- If symmetries are sought in the evolutionary (characteristic) form (1.12), one specifies an additional parameter

  ```
  in_evolutionary_form=true
  ```

- In case of higher-order symmetries, symmetries are normally sought in the evolutionary form.
- If one does not wish the determining equations to be split with respect to (higher) derivatives which do not participate in tangent vector field coordinates, one specifies an additional parameter

  ```
  return_unsplit=true
  ```

- If one wishes the determining equations to be split with respect to some *particular* variables and/or derivatives, one specifies an additional list-type parameter

  ```
  split_wrt=[...]
  ```

  (this is an option needed only in very special situations!)

5. Request names of symmetry components and place them in some user variable.

```
sym_components:=gem_symm_components();
```

6. Simplify and reduce the overdetermined system of symmetry determining equations.

```
simplified_eqs:=DEtools[rifsimp](det_eqs, sym_components,
 mindim=1);
```

[In particular, the option `mindim=1` forces the output of the number of linearly independent solutions of equations `simplified_eqs`, i.e., the number of symmetries of the given PDE system.]

7. Solve the determining equations.

```
symm_sol:=pdsolve(simplified_eqs[Solved]);
```

8. Print all symmetries.

```
gem_output_symm(symm_sol);
```

Remark 1. The given DE system is assumed to be non-linear. Linear DE systems involve trivial infinite symmetries which meed to be excluded. `GeM` routines for symmetry analysis of linear DE systems will be added soon.

Remark 2. If a given system involves arbitrary constitutive functions/constant parameters, and symmetries need to be classified, this can be done on a case-by-case basis. See specific examples on the web site: [8].

4 Program sequence for conservation law analysis

1. Clear the variables and read in the `GeM` module as a `Maple` text input file. (Same as in the symmetry analysis sequence.)

2. Declare variables and arbitrary functions/constants. (Same as in the symmetry analysis sequence.)

3. Declare the given PDE system. (Same as in the symmetry analysis sequence.)

- For obtaining conservation law multipliers, it is NOT necessary that a given PDE system can be written in a solved form with respect to a set of leading derivatives. The parameter may therefore be omitted.
- If fluxes of conservation laws are to be computed later, specification of the `solve_for` parameter is necessary, since the program verifies whether or not, indeed, the divergence expression $\mathrm{D}_i\Phi^i[U]$ vanishes on the solutions $U(x) = u(x)$.

4. Generate multiplier determining equations

```
det_eqs:=gem_conslaw_det_eqs([...]);
```

- The name in the left-hand side can be arbitrary; the corresponding `Maple` variable will contain the determining equations.
- The list `[...]` contains the dependence of multipliers. Can include independent and dependent variables and derivatives of dependent variables up to any order.

5. Request names of unknown multipliers and place them in some user variable.

```
CL_multipliers:=gem_conslaw_multipliers();
```

6. Simplify and reduce the overdetermined system of multiplier determining equations.

```
simplified_eqs:=DEtools[rifsimp](det_eqs,
  CL_multipliers, mindim=1);
```

Again, `mindim=1` forces the output of the number of linearly independent solutions of equations `simplified_eqs`, i.e., the number of independent local conservation laws of the given PDE system within the specified multiplier ansatz.]

7. Solve the determining equations.

```
multipliers_sol:=pdsolve(simplified_eqs[Solved]);
```

8. Compute the corresponding fluxes. If `multipliers_sol` involves arbitrary constants, print fluxes of independent conservation laws separately.

```
gem_get_CL_fluxes(multipliers_sol);
```

- To employ the first homotopy formula, use an additional parameter `method="Homotopy1"`. In this case, one may also specify an arbitrary function $\widetilde{U}(x)$ (by default, it is zero). The arbitrary function is a vector function having the same dimension as the vector of dependent variables. For example, if the given PDE system has independent variables (x, y, z) and dependent variables $A(x, y, z), B(x, y, z)$, the $A-$ and $B-$ components of the arbitrary function are specified as the following additional set-type parameter:

  ```
  arb_func_Homotopy1={A=x+y, B=z+1}
  ```

  In particular, the corresponding right-hand sides can be any expressions involving independent variables.

- To employ the second homotopy formula, use an additional parameter `method="Homotopy2"`.

- To employ the scaling formula, use an additional parameter `method="Scaling"`. In this case, one needs to specify the scaling symmetry to be used. For example, suppose the given PDE system has independent variables (x, y, z) and dependent variables $A(x, y, z), B(x, y, z)$, and the scaling symmetry is given by

  $$\mathrm{X} = x\frac{\partial}{\partial x} + 2y\frac{\partial}{\partial x} + z\frac{\partial}{\partial z} + 2A\frac{\partial}{\partial A} - 3B\frac{\partial}{\partial B},$$

  Then the necessary parameter is:

  ```
  symmetry={xi_x=x, xi_y=2*y, xi_z=z, eta_A=2*A, eta_B=-3*B}
  ```

5 Further remarks on execution of GeM routines

1. **DE systems involving transcendental functions.** If given equations are not polynomial in independent and dependent variables and derivatives of dependent variables, but involves transcendental functions, say, sines, powers, exponents, etc., then `Maple rifsimp` routine will not work. One should either omit the `rifsimp` step and directly proceed to solution, or denote the transcendental function as an "arbitrary" function $F(...)$, and later, in the `rifsimp` step, append determining equations with conditions on $F(...)$ that determine it uniquely. The corresponding example is given on the web site.

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] Bluman G.W., Cheviakov A.F., and Anco S.C., *Applications of Symmetry Methods to Partial Differential Equations.* Springer: Applied Mathematical Sciences , Vol. 168 (2010), ISBN 978-0-387-98612-8.

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

[5] Cheviakov A. F., Computation of fluxes of conservation laws. J.Eng.Math. DOI `10.1007/s10665-009-9307-x` (2009).

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

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

[8] The `GeM` package, documantation, and examples are available at `http://www.math.usask.ca/~cheviakov/gem/`.

[9] Cheviakov A. F., "Bogoyavlenskij symmetries of ideal MHD equilibria as Lie point transformations." *Phys. Lett. A.* **321/1**, 34-49 (2004).