Short guide to `YaLie`

Yet another Lie *Mathematica* package for Lie symmetries

José Manuel Díaz
Departamento de Matemáticas
Universidad de Cádiz
España
`josemanuel.diaz@uca.es`

Contents

1 Introduction

Nonlinear phenomena play a crucial role in most areas of sciences. Essentially all the fundamental equations of physics are nonlinear and, in general, very difficult to solve explicitly. However, in many situations, it is necessary to obtain exact analytical solutions.

Lie group techniques provide a method to obtain solutions of partial differential equations (PDEs) by exploiting symmetries of the equation. This method provides some special solutions in terms of solutions of ordinary differential equations.

The classical method for finding symmetry reductions of PDEs describing nonlinear phenomena is the method of infinitesimal transformations, which provides tools for obtaining, at least, particular solutions of the equations. For a PDEs system, it must be constructed the associated determining equations; they constitute an overdetermined linear system for the infinitesimal transformation. Although this method is entirely algorithmic, it often involves a large amount of tedious algebraic manipulations, which can become virtually unmanageable; so symbolic programs in MACSYMA, REDUCE, MAPLE and *Mathematica* [1] have been developed to facilitate the calculations.

Here, we present another *Mathematica* package to determine Lie symmetries. In Section 2, we will see what determining equations are. Next, in sections 3-8, we will explain how to use the Yet Another Lie *Mathematica* Package `YaLie`, through working examples. In section 9 we show the commands of this package and some technical notes. Finally, in section 10, we explain the advantages of YaLie with respect to another Lie packages.

2 Determining equations for Lie symmetries

We consider the general case given by a system of partial differential equations

$$\Sigma = \{\psi_1 = 0,\ \psi_2 = 0,\ \ldots,\ \psi_r = 0\}$$

in n independent variables

$$x = (x_1,\, x_2,\, \ldots, x_n),$$

m dependent variables

$$u = (u_1,\, u_2,\, \ldots, u_m)$$

and the derivative terms $\{u_k^{\mathbf{a}},\ \mathbf{a} \in \mathbb{N}^n\}$ with

$$u_k^{\mathbf{a}} = \frac{\partial^{|\mathbf{a}|} u_k}{\partial x_1^{a_1} \ldots \partial x_n^{a_n}}, \quad |\mathbf{a}| = a_1 + \cdots + a_n \leq N$$

To find the classical Lie point symmetries of the system Σ, we consider a one-parameter Lie group of transformations

$$\begin{aligned} x^* &= \Psi(x, u; \varepsilon) \\ u^* &= \Phi(x, u; \varepsilon) \end{aligned}$$

under which Σ must be invariant. The group action is infinitesimally given by

$$\begin{aligned} x_i^* &= x_i + \varepsilon \xi_i(x, u) + \mathcal{O}(\varepsilon^2), \quad i = 1, 2, \ldots, n \\ u_j^* &= u_j + \varepsilon \phi_j(x, u) + \mathcal{O}(\varepsilon^2), \quad j = 1, 2, \ldots, m. \end{aligned}$$

In order to determinate the infinitesimals $\xi_i(x,u)$ and $\phi_j(x,u)$, we require that the previous transformation leaves invariant the space of solutions of Σ:

$$\mathcal{S}_\Sigma = \{u(x) \ : \ \psi_1 = 0,\ \psi_2 = 0,\ \ldots,\ \psi_r = 0\}.$$

This condition is equivalent [8] to

$$\mathrm{pr}^N \mathbf{v}(\psi_i)\big|_\Sigma = 0, \quad i = 1, 2, \ldots, r, \tag{1}$$

where

$$\mathbf{v} \equiv \sum_{i=1}^{n} \xi_i(x,u)\partial_{x_i} + \sum_{j=1}^{m} \phi_j(x,u)\partial_{u_k},$$

$$\mathrm{pr}^N \mathbf{v} \equiv \sum_{i=1}^{n} \xi_i\partial_{x_i} + \sum_{k=1}^{m} \phi_k\partial_{u_k} + \sum_{k=1}^{m}\sum_{|\mathbf{a}|\geq 1} \phi_k^{[\mathbf{a}]}\partial_{u_k^\alpha} \tag{2}$$

and $\mathrm{pr}^N \mathbf{v}$ denotes the N-th prolongation of the vector field $\mathbf{v}$; here $\phi_k^{[\mathbf{a}]}$ is recursively given by

$$\phi_k^{[\mathbf{a}+i]} = D_{x_i}\phi_k^{[\mathbf{a}]} - \sum_{j=1}^{n} D_{x_i}\xi_j\ u_k^{\mathbf{a}+j}.$$

This yields an overdetermined linear system of determining equations for the infinitesimals $\xi(x,u)$ and $\phi(x,u)$, which can be obtained by setting null the coefficients of derivative terms $u_k^{\mathbf{a}}$ in (1).

In practice, the algorithm for finding the infinitesimals $\xi(x,u)$ and $\phi(x,u)$ follows the steps:

1. Determine the N-th prolongation (2).
2. Apply $\mathrm{pr}^N \mathbf{v}$ to the system Σ.
3. Consider the original system Σ as a system of algebraic equations and solve it for unambiguously derivatives (preferably linearly).
4. Substitute the solutions in $\mathrm{pr}^N \mathbf{v}(\Sigma)$.
5. Set the coefficient of the remaining derivatives $u_k^{\mathbf{a}}$ equal to zero. This provides the determining equations.
6. Solve the determining equations to obtain $\xi(x,u)$ and $\phi(x,u)$.

The above steps 1-5 are algorithmic and they are best performed with a computer. Several routines to perform this task have been built in REDUCE, MACSYMA, Mathematica and other languages. Nevertheless, our `YaLie` package present several advantages over these packages, as we will see in Section 10.

The last step is less algorithmic. (The author is actually working in a package, `FormaEstándar`, that builds an equivalent system, which is more adequate for this purposes).

The system obtained in 5. is usually overdetermined and the following possibilities can happen:

1. The unique solution is $\xi_i = 0$ and $\phi_k = 0$.
2. The general solution depends on a finite number ℓ of constants. In this case, we obtain an ℓ-dimensional symmetry group.
3. The general solution depends on arbitrary functions. In this case we obtain an infinite-dimensional Lie group.

4. The general solution depends on a finite number ℓ of constants and on arbitrary functions. We then obtain an ℓ-dimensional symmetry group and an infinite-dimensional Lie group.

3 The traditional one dimensional heat equation

As a first example, we will not violate a longstanding tradition and we consider the heat equation

$$u_t - u_{xx} = 0$$

whose symmetries has been widely studied [2, 8, 1]

We start a *Mathematica* session, prepare directories, and read in the `YaLie` package:

```
$Path = Append[$Path, "d:\josé manuel\Lie\Software\JM\YaLie"];
SetDirectory["d:\josé manuel\Lie\Ecuaciones"]
Get["YaLie.m"]
```

Now we directly prepare the problem (or read from a file):

```
Calor1D = {D[u[x,t], t] - D[u[x,t], x,x] == 0};

VD = {u};            (* Dependent variable *)
VIs = {x,t};         (* Independent variables *)
VP = {D[u[x,t], t]}; (* Main variable *)
```

In order to find the determining equations we use `YaLie[]`:

```
EDsCalor1D = YaLie[Calor1D, VD, VIs, VP];
```

This gives us 7 determining equations:

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

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

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

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

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

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

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

The infinitesimals ξ_i and ϕ_j are then obtained from the above system of partial differential equations. Let us observe that ξ_i and ϕ_j are denoted by `Xi[i]` and `Phi[j]` respectively.

The general solution of these equations determines the infinitesimals of the symmetry group for the heat equation. From the first four equations, we know that

$$\begin{aligned}\xi_1 &= \xi_1(x,t)\\ \xi_2 &= \xi_2(t)\\ \phi_1 &= f(x,t)\,u + g(x,t).\end{aligned}$$

Next, we introduce this information:

```
LXi = {Xi[1][x,t], Xi[2][t]}

LPhi = {f[x,t] u + g[x,t]}
```

and we run `YaLie[]` again, in order to obtain a simplified system of equations

```
EDsCalor1D = YaLie[Calor1D, VD, VIs, VP, LXi, LPhi];
```

This gives us the following system:

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

        (0,1)            (1,0)                (2,0)
-(Xi[1])     [x, t] - 2 f     [x, t] + (Xi[1])     [x, t] == 0

                          (1,0)
-(Xi[2])'[t] + 2 (Xi[1])     [x, t] == 0
```

As we will see in the next section, a straightforward integration yields

$$\begin{aligned}\xi_1(x,t) &= K_1 + K_2\,x + 2\,K_3\,t + 4\,K_4\,x\,t\\ \xi_2(t) &= K_5 + 2\,K_2\,t + 4\,K_4\,t^2\\ \phi_1(x,t,u) &= \big(K_6 - K_3\,x - K_4(x^2+2t)\big)\,u + g(x,t),\end{aligned}$$

where $g(x,t)$ is a particular solution of the heat equation.

In order to check it, we introduce

```
LXi = {K[1] + K[2] x + 2 K[3] t + 4 K[4] x t, K[5] + 2 K[2] t + 4 K[4] t^2}

LPhi = {(K[6]-K[3] x - K[4](x^2+2t))u + g[x,t]}


EDs = YaLie[Calor1D, VD, VIs, VP, LXi, LPhi];
```

to obtain

```
                             (0,1)          (2,0)
{True, True, True, True, g      [x, t] - g      [x, t] == 0, True, True}
```

Once we have determined the infinitesimals, we could calculate the corresponding Lie algebra and we could study it. Although this task cannot be performed with `YaLie`, a new package `LieAlgebra` to help in that study will submitted as soon as possible.

4 Working step by step

When the former problem is too complicate sometimes it is convenient to work step by step. Again, `YaLie` provides the adequate functions to perform this task.

Let us go back to the heat equation:

$$\psi = u_t - u_{xx} = 0$$

Now, we will work step by step to determine the infinitesimals ξ_i and ϕ_j for this equation.

First, we need to find the second prolongation, acting on ψ, $\mathrm{pr}^2\,\mathbf{v}(\psi)$ where

$$\mathbf{v} = \xi^{(1)}(x,t,u)\partial_x + \xi^{(2)}(x,t,u)\partial_t + \phi^{(1)}(x,t,u)\partial_u.$$

therefore we start defining the `Difusión` equation and next we invoke the `Prolongación` function.

```
Difusión = D[u[x,t], t] - D[u[x,t], {x,2}];

prL = Prolongación[Difusión, {u}, {x,t}, 2];
```

In what follows, sometimes we will use the character `;` to avoid displaying the output, when this is too complex.

Now, we solve (perhaps it is trivial) $\psi = 0$ for the variable u_t:

```
Sust =  First[Solve[Difusión==0, D[u[x,t], t]]]

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

Then, we determine

$$\mathrm{pr}^2\,\mathbf{v}(\psi)\big|_{\psi=0}$$

```
prL = prL /. Sust;
```

In this point, we are ready to get the determining equations with the `EDeterminantes` function:

```
EDs = EDeterminantes[prL, {u}, {x,t}];

         (0,0,1)
2 (Xi[2])       [x, t, u]

       (0,0,2)
(Xi[1])       [x, t, u]

       (0,0,2)
(Xi[2])       [x, t, u]

         (1,0,0)

2 (Xi[2])       [x, t, u]

         (0,0,2)                     (1,0,1)
-(Phi[1])       [x, t, u] + 2 (Xi[1])       [x, t, u]

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

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

        (0,1,0)                      (1,0,1)                  (2,0,0)
-(Xi[1])       [x, t, u] - 2 (Phi[1])       [x, t, u] + (Xi[1])       [x, t, u]

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

We have obtained 9 determining equations, which can be simplified to 7 if we perform a simple (in this case) but sometimes meaningful reduction:

```
EDs = Simplifica[EDs];


       (0,0,1)
(Xi[1])       [x, t, u]

       (0,0,1)
(Xi[2])       [x, t, u]

        (0,0,2)
(Phi[1])       [x, t, u]

       (1,0,0)
(Xi[2])       [x, t, u]

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

        (0,1,0)                      (1,0,1)                  (2,0,0)
-(Xi[1])       [x, t, u] - 2 (Phi[1])       [x, t, u] + (Xi[1])       [x, t, u]

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

The next step consists in finding the general solution of the above system. Let us go step by step. From the first four equations, we see that

$$\begin{aligned}
\xi^{(1)} &= \xi^{(1)}(x,t) \\
\xi^{(2)} &= \xi^{(2)}(t) \\
\phi^{(1)} &= f(x,t)\,u + g(x,t)
\end{aligned}$$

and, with this information, we set `LXi` and `LPhi` as the infinitesimals. We introduce them into the determining equations

```
LXi = {Xi[1][x,t], Xi[2][t]};

LPhi = {f[x,t] u + g[x,t]};


EDsI = SInfinitesimales[EDs, LXi, LPhi]


   (0,1)           (0,1)             (2,0)           (2,0)
u f     [x, t] + g      [x, t] - u f      [x, t] - g      [x, t]

        (0,1)             (1,0)                (2,0)
-(Xi[1])     [x, t] - 2 f     [x, t] + (Xi[1])     [x, t]

                                (1,0)
-(Xi[2])'[t] + 2 (Xi[1])     [x, t]
```

Here, from the third equation, we can deduce that

$$\xi^{(1)}(x,t) = (1/2)\frac{\partial \xi^{(2)}(t)}{\partial t}\,x + h(t),$$

thus, we update `LXi` and we substitute again

```
LXi = {1/2 D[Xi[2][t], t] x + h[t], Xi[2][t]};

LPhi = {f[x,t] u + g[x,t]};


EDsI = SInfinitesimales[EDs, LXi, LPhi]


   (0,1)           (0,1)             (2,0)           (2,0)
u f     [x, t] + g      [x, t] - u f      [x, t] - g      [x, t]

         x (Xi[2])''[t]      (1,0)
-h'[t] - -------------- - 2 f     [x, t]
               2
```

Let us observe that, in the `SInfinitesimales` function, we have use `EDs`, instead of `EDsI`. The reason is that the algorithm substitutes the formal infinitesimals `Xi` and `Phi` and not other functions.

Again, from the last equations, we see that

$$f(x,t) = -(1/8)\frac{\partial^2 \xi^{(2)}(t)}{\partial t^2}\,x^2 - (1/2)\,h'(t)\,x + p(t)$$

and so

```
LXi = {1/2 D[Xi[2][t], t] x + h[t], Xi[2][t]};

LPhi = {(-1/8 D[Xi[2][t], {t,2}] x^2 - 1/2 h'[t] x + p[t])u + g[x,t]};


EDsI = SInfinitesimales[EDs, LXi, LPhi]

                                                      2         (3)
          u x h''[t]   u (Xi[2])''[t]   u x  (Xi[2])   [t]    (0,1)          (2,0)
u p'[t] - ---------- + -------------- - ------------------ + g      [x, t] - g     [x, t]
              2              4                  8
```

Finally, by imposing that the above equation is identically zero, we obtain

$$
\begin{aligned}
\partial_{ttt}\xi^{(2)}(t) &= 0 \\
\partial_{tt}\xi^{(2)}(t) + 4\,p'(t) &= 0 \\
h''(t) &= 0
\end{aligned}
$$

and, hence, it is straightforward to deduce that

$$
\begin{aligned}
\xi^{(1)} &= K_2\,x\,t + (1/2)\,K_1\,x + K_3\,t + K_4 \\
\xi^{(2)} &= K_2\,t^2 + K_1\,t + K_0 \\
\phi^{(1)} &= -\left[(1/4)K_2\,x^2 + (1/2)K_3\,x + (1/2)K_2\,t + K_5\right]\,u + g(x,t)
\end{aligned}
$$

where $g(x,t)$ is an arbitrary solution of the original equation.

5 The three dimensional Laplace equation

We consider here another example: the three dimensional Laplace equation

$$u_{xx} + u_{yy} + u_{zz} = 0$$

We start by defining the Laplace 3D equation and its associated variables:

```
Laplace3D = {D[u[x,y,z], {x,2}] + D[u[x,y,z], {y,2}] + D[u[x,y,z], {z,2}]==0};

VD = {u};

VIs = {x,y,z};

VP = {D[u[x,y,z], {z,2}]};
```

and we invoke the `YaLie` function:

```
EDs = YaLie[Laplace3D, VD, VIs, VP];


        (0,0,0,1)                              (0,0,0,1)
{(Xi[1])         [x, y, z, u] == 0, (Xi[2])         [x, y, z, u] == 0,

        (0,0,0,1)                               (0,0,0,2)
 (Xi[3])         [x, y, z, u] == 0, (Phi[1])         [x, y, z, u] == 0,
```

```
          (0,0,1,0)                          (0,1,0,0)
 2 (Xi[3])         [x, y, z, u] - 2 (Xi[2])         [x, y, z, u] == 0,

           (0,0,1,0)                          (0,1,0,0)
 -2 (Xi[2])         [x, y, z, u] - 2 (Xi[3])         [x, y, z, u] == 0,

          (0,0,1,0)                          (1,0,0,0)
 2 (Xi[3])         [x, y, z, u] - 2 (Xi[1])         [x, y, z, u] == 0,

           (0,1,0,0)                          (1,0,0,0)
 -2 (Xi[1])         [x, y, z, u] - 2 (Xi[2])         [x, y, z, u] == 0,

           (0,0,1,0)                          (1,0,0,0)
 -2 (Xi[1])         [x, y, z, u] - 2 (Xi[3])         [x, y, z, u] == 0,

         (0,0,2,0)                        (0,2,0,0)                          (2,0,0,0)
 (Phi[1])         [x, y, z, u] + (Phi[1])         [x, y, z, u] + (Phi[1])         [x, y, z, u] == 0,

         (0,0,2,0)                       (0,2,0,0)
 -(Xi[1])         [x, y, z, u] - (Xi[1])         [x, y, z, u] +
             (1,0,0,1)                        (2,0,0,0)
   2 (Phi[1])         [x, y, z, u] - (Xi[1])         [x, y, z, u] == 0,

         (0,0,2,0)                          (0,1,0,1)
 -(Xi[2])         [x, y, z, u] + 2 (Phi[1])         [x, y, z, u] -
          (0,2,0,0)                       (2,0,0,0)
   (Xi[2])         [x, y, z, u] - (Xi[2])         [x, y, z, u] == 0,

           (0,0,1,1)                        (0,0,2,0)
 2 (Phi[1])         [x, y, z, u] - (Xi[3])         [x, y, z, u] -
           (0,2,0,0)                        (2,0,0,0)
   (Xi[3])         [x, y, z, u] - (Xi[3])         [x, y, z, u] == 0}
```

Now, one can try to find the general solution of this system and consequently the symmetry algebra. This can be achieved in this case, but in the general case this could be virtually impossible. The author is actually working in the package `FormaEstándar` that use the standard form algorithm by Reid [9]. This package builds an equivalent system which is more adequate for our purposes. For example, we obtain the determining equations of the symmetry group of the Laplace equation:

```
         (0,0,0,1)                            (0,0,0,1)
{{(Xi[1])         [x, y, z, u] == 0, (Xi[2])         [x, y, z, u] == 0,

          (0,0,0,1)                            (0,0,0,2)
   (Xi[3])         [x, y, z, u] == 0, (Xi[4])         [x, y, z, u] == 0,

          (0,1,0,0)                        (0,0,1,0)
   (Xi[2])         [x, y, z, u] == (Xi[3])         [x, y, z, u],

          (0,1,0,0)                        (0,0,1,0)
   (Xi[3])         [x, y, z, u] == (Xi[2])         [x, y, z, u],

          (1,0,0,0)                        (0,0,1,0)
   (Xi[1])         [x, y, z, u] == (Xi[3])         [x, y, z, u],

          (1,0,0,0)                        (0,1,0,0)
   (Xi[2])         [x, y, z, u] == (Xi[1])         [x, y, z, u],

          (1,0,0,0)                         (0,0,1,0)
   (Xi[3])         [x, y, z, u] == -(Xi[1])         [x, y, z, u],
```

```
       (2,0,0,0)                          (0,0,2,0)                         (0,2,0,0)
(Xi[4])         [x, y, z, u] == -(Xi[4])         [x, y, z, u] + (Xi[4])         [x, y, z, u],

       (0,0,2,0)                            (0,0,1,1)
(Xi[3])         [x, y, z, u] == -2 (Xi[4])         [x, y, z, u],

                                       (0,0,2,0)
       (1,0,0,1)                 (Xi[1])         [x, y, z, u]
(Xi[4])         [x, y, z, u] == ----------------------------,
                                             2

                                         (0,0,2,0)
       (0,1,0,1)                  -(Xi[2])         [x, y, z, u]
(Xi[4])         [x, y, z, u] ==  -----------------------------
                                               2

        (0,1,1,0)
(Xi[1])         [x, y, z, u] == 0,


       (0,2,0,0)                          (0,0,2,0)
(Xi[1])         [x, y, z, u] == -(Xi[1])         [x, y, z, u],

       (0,0,3,0)                            (0,0,2,1)
(Xi[1])         [x, y, z, u] == 0, (Xi[4])         [x, y, z, u] == 0,

       (0,0,3,0)
(Xi[2])         [x, y, z, u] == 0}, {}}
```

The above system, in which `Xi[4]` represents `Phi[1]`, is considerably more easy to solve than the original system and we can obtain [5] the infinitesimals

$$
\begin{aligned}
\xi^{(1)}(x,y,z) &= a_1 + 2\,b_1\,x + b_2\,y + b_3\,z + \\
&\quad c_1\,\left(x^2 - y^2 - z^2\right) + 2\,c_2\,x\,y + 2\,c_3\,x\,z, \\
\xi^{(2)}(x,y,z) &= a_2 - b_2\,x + 2\,b_1\,y + b_4\,z + \\
&\quad 2\,c_1\,x\,y + c_2\,\left(-x^2 + y^2 - z^2\right) + 2\,c_3\,y\,z, \\
\xi^{(2)}(x,y,z) &= a_3 - b_3\,x - b_4\,y + 2\,b_1\,z + \\
&\quad 2\,c_1\,x\,z + 2\,c_2\,y\,z + c_3\,\left(-x^2 - y^2 + z^2\right). \\
\phi^{(1)}(x,y,z,u) &= (d - c_1\,x - c_2\,y - \,c_3 z)\,u + \rho(x,y,z)
\end{aligned}
$$

where $\rho(x,y,z)$ is any particular solution of the original equation.

6 A more time consuming example: Maxwell-Bloch system

As a less trivial example, we consider the system of Maxwell-Bloch equations

$$
\begin{aligned}
F_t + -\frac{1}{4\alpha} i(F_{xx} + F_{yy}) + \sigma(F - P) &= 0 \\
P_t + (1 + i\,\delta)P - F\,r &= 0 \\
r_t + \gamma(r - \rho + 1/2(\overline{F}P + F\overline{P})) &= 0
\end{aligned}
$$

where $F = f(x,y,t) + ig(x,y,t)$, $P = p(x,y,t) + iq(x,y,t)$ and $r = r(x,y,t)$.

First, we obtain the following system of partial differential equations by separating the real and imaginary parts of the dependent variables:

$$
\begin{aligned}
f_t + \frac{1}{4\alpha}(g_{xx} + g_{yy}) + \sigma(f - p) &= 0 \\
g_t - \frac{1}{4\alpha}(f_{xx} + f_{yy}) + \sigma(g - q) &= 0 \\
p_t + p - \delta q - f\, r &= 0 \\
q_t + q + \delta p - g\, r &= 0 \\
r_t + \gamma(r - \rho + f\, p + g\, q) &= 0
\end{aligned}
$$

Then, by using the command

```
Timing[Length[YaLie[MaxwellBloch, VDs, VIs, VPs]]]
```

we obtain 58 determining equations, in 23.23 seconds (in my computer).

```
{23.23 Second, 58}
```

In some cases, the original system can be very complicated and time consuming. In such cases, we can use the option `Verborrea` for trace the work of `YaLie[]`. This option can be used as

```
YaLie[MaxwellBloch, VDs, VIs, VPs, Verborrea->True]
```

or, in a more convenient way, we put

```
If[TimeConstrained[Ed = YaLie[MaxwellBloch, VDs, VIs, VPs],50]===$Aborted,
       Ed = YaLie[MaxwellBloch, VDs, VIs, VPs, Verborrea->True]];
```

```
:: Ecuación (1) (0.05)
     Prolongación (1.21)
     Ecuaciones determinantes (5.33)
:: Ecuación (2) (5.38)
     Prolongación (6.48)
     Ecuaciones determinantes (10.6)
:: Ecuación (3) (10.6)
     Prolongación (10.82)
     Ecuaciones determinantes (11.04)
:: Ecuación (4) (11.09)
     Prolongación (11.31)
     Ecuaciones determinantes (11.53)
:: Ecuación (5) (11.59)
     Prolongación (11.81)
     Ecuaciones determinantes (12.08)
:: Simplificación (23.62)
```

7 A nonlinear diffusion-absorption equation

We will work now on a complete example. The model equation to be considered here is

$$u_t = u_{xx} + u_{yy} + f(u),$$

which is a diffusion-absorption equation. The term $f(u)$ on the right side is a nonlinear reaction rate which represents volumetric absorption. We are interested in the special case

$$f(u) = c\,u\,(b + \log(a\,u))$$

The nonlinear diffusion-absorption equation

$$u_t = u_{xx} + u_{yy} + c\,u\,(b + \log(a\,u))$$

was considered by Gandarias and del Águila in [6]. I thanks them for letting me mimic their work.

We start by preparing a new *Mathematica* session and reading in the equation file

```
$Path = Append[$Path, "d:\josé manuel\Lie\Software\JM\YaLie"];

SetDirectory["d:\josé manuel\Lie\Ecuaciones"];

Get["YaLie.m"];
Get["Difusión-Absorción\DifusiónAbsorción.m"];
```

We use `YaLie[]` and we obtain 12 determining equations:

```
EDs = YaLie[DALog, VDs, VIs, VPs]; EDs // ColumnForm

       (0,0,0,1)
(Xi[1])         [x, y, t, u] == 0

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

       (0,0,0,1)
(Xi[3])         [x, y, t, u] == 0

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

       (0,1,0,0)
(Xi[3])         [x, y, t, u] == 0

       (1,0,0,0)
(Xi[3])         [x, y, t, u] == 0

         (0,1,0,0)                        (1,0,0,0)
2 (Xi[1])         [x, y, t, u] + 2 (Xi[2])         [x, y, t, u] == 0
        (0,0,1,0)                      (0,2,0,0)
-(Xi[1])         [x, y, t, u] + (Xi[1])         [x, y, t, u]
            (1,0,0,1)                      (2,0,0,0)
- 2 (Phi[1])         [x, y, t, u] + (Xi[1])         [x, y, t, u] == 0
```

```
       (0,0,1,0)                         (0,1,0,1)
-(Xi[2])         [x, y, t, u] - 2 (Phi[1])         [x, y, t, u]
       (0,2,0,0)                      (2,0,0,0)
+(Xi[2])         [x, y, t, u] + (Xi[2])         [x, y, t, u] == 0

       (0,0,1,0)                        (0,1,0,0)
-(Xi[3])         [x, y, t, u] + 2 (Xi[2])         [x, y, t, u] == 0

       (0,0,1,0)                        (1,0,0,0)
-(Xi[3])         [x, y, t, u] + 2 (Xi[1])         [x, y, t, u] == 0
                                                                    (0,0,0,1)
(-c-c(b + Log[a u])) Phi[1][x, y, t, u] + c u (b + Log[a u]) (Phi[1])         [x, y, t, u] +
          (0,0,1,0)                                        (0,0,1,0)
   (Phi[1])         [x, y, t, u] - c u (b + Log[a u]) (Xi[3])         [x, y,t, u] -
          (0,2,0,0)                              (2,0,0,0)
   (Phi[1])         [x, y, t, u] - (Phi[1])         [x, y, t, u] == 0
```

Sometimes I use `DFormat` package by Pekka Janhunen [7] in order to obtain a better appearence:

```
Get["..\Software\Janhunen\DFormat.m"];
DFormat[NoVars]
```

Indeed, using `DFormat[NoVars]` we see

```
                                          2
 dXi[1]       dXi[2]       dXi[3]       d Phi[1]       dXi[3]       dXi[3]
{------ == 0, ------ == 0, ------ == 0, -------- == 0, ------ == 0, ------ == 0,
   du           du           du             2            dy           dx
                                          du


                                         2           2           2
   dXi[1]     dXi[2]         dXi[1]    d Xi[1]     d Phi[1]   d Xi[1]
 2 ------ + 2 ------ == 0, -(------) + ------- - 2 -------- + ------- == 0,
     dy         dx             dt          2         dxdu         2
                                         dy                     dx

                2          2          2
   dXi[2]      d Phi[1]   d Xi[2]   d Xi[2]          dXi[3]      dXi[2]
 -(------) - 2 -------- + ------- + ------- == 0, -(------) + 2 ------ == 0,
     dt          dydu         2         2              dt          dy
                            dy        dx

   dXi[3]      dXi[1]
 -(------) + 2 ------ == 0,
     dt          dx

                                                                  dPhi[1]    dPhi[1]
 (-c - c (b + Log[a u])) Phi[1][x, y, t, u] +  c u (b + Log[a u]) ------- + ------- -
                                                                     du         dt
                               2          2
                       dXi[3]   d Phi[1]   d Phi[1]
   c u (b + Log[a u]) ------ - -------- - -------- == 0}
                          dt         2           2
                                   dy          dx
```

It is also possible to use the command `DFormat[True]` included in `YaLie` package:

```
DFormat[True]; EDs

{Xi[1]  == 0, Xi[2]  == 0, Xi[3]  == 0, Phi[1]   == 0, Xi[3]  == 0, Xi[3]  == 0,
      u           u           u             uu            y             x

 2 Xi[1]  + 2 Xi[2]  == 0, -Xi[1]  + Xi[1]   - 2 Phi[1]   + Xi[1]   == 0,
       y          x              t        yy            xu        xx

 -Xi[2]  - 2 Phi[1]   + Xi[2]   + Xi[2]   == 0, -Xi[3]  + 2 Xi[2]  == 0,
       t           yu        yy        xx             t          y

 -Xi[3]  + 2 Xi[1]  == 0,
       t          x

  (-c - c (b + Log[a u])) Phi[1][x, y, t, u] +  c u (b + Log[a u]) Phi[1]  + Phi[1]  -
                                                                       u         t

   c u (b + Log[a u]) Xi[3]  - Phi[1]   - Phi[1]   == 0}
                           t         yy         xx
```

Nevertheless, it could be better to use the template `yalie.mx` to generate a tex archive

```
LieTeX["Difusión-Absorción\DifusiónAbsorción.tex", EDs]
```

This command produce the TEX archive `DifusiónAbsorción.tex` that you can compile and see the results.

(Remark: In order to work properly, it is important to observe that the path to `yalie.mx` should be included in the `$Path` variable.)

The TEX archive `DifusiónAbsorción.tex` include something like

```
\item Ecuaciones determinantes
\begin{eqnarray}
  \xi\RawWedge {(1)}\_{u} \&=\& 0  \\   \xi\RawWedge {(2)}\_{u} \&=\& 0  \\
 \xi\RawWedge {(3)}\_{u} \&=\& 0  \\   \phi\RawWedge {(1)}\_{uu} \&=\& 0
  \\   \xi\RawWedge {(3)}\_{y} \&=\& 0  \\   \xi\RawWedge {(3)}\_{x} \&=\& 0
  \\   2\,\xi\RawWedge {(1)}\_{y} + 2\,\xi\RawWedge {(2)}\_{x} \&=\& 0  \\
 -\xi\RawWedge {(1)}\_{t} + \xi\RawWedge {(1)}\_{yy} -
  2\,\phi\RawWedge {(1)}\_{xu} + \xi\RawWedge {(1)}\_{xx} \&=\& 0  \\
 -\xi\RawWedge {(2)}\_{t} - 2\,\phi\RawWedge {(1)}\_{yu} +
  \xi\RawWedge {(2)}\_{yy} + \xi\RawWedge {(2)}\_{xx} \&=\& 0  \\
 -\xi\RawWedge {(3)}\_{t} + 2\,\xi\RawWedge {(2)}\_{y} \&=\& 0  \\
 -\xi\RawWedge {(3)}\_{t} + 2\,\xi\RawWedge {(1)}\_{x} \&=\& 0  \\
 \left( -c - c\,\left( b + \log (a\,u) \right)  \right) \,
   \phi\RawWedge {(1)} + c\,u\,\left( b + \log (a\,u) \right) \,
   \phi\RawWedge {(1)}\_{u} + \phi\RawWedge {(1)}\_{t} -
  c\,u\,\left( b + \log (a\,u) \right) \,\xi\RawWedge {(3)}\_{t} -
  \phi\RawWedge {(1)}\_{yy} - \phi\RawWedge {(1)}\_{xx} \&=\& 0
\end{eqnarray}
```

After the compilation, the `.dvi` file is shown as

$$
\begin{aligned}
\xi^{(1)}_u &= 0 && (3)\\
\xi^{(2)}_u &= 0 && (4)\\
\xi^{(3)}_u &= 0 && (5)\\
\phi^{(1)}_{uu} &= 0 && (6)\\
\xi^{(3)}_y &= 0 && (7)\\
\xi^{(3)}_x &= 0 && (8)\\
2\,\xi^{(1)}_y + 2\,\xi^{(2)}_x &= 0 && (9)\\
-\xi^{(1)}_t + \xi^{(1)}_{yy} - 2\,\phi^{(1)}_{xu} + \xi^{(1)}_{xx} &= 0 && (10)\\
-\xi^{(2)}_t - 2\,\phi^{(1)}_{yu} + \xi^{(2)}_{yy} + \xi^{(2)}_{xx} &= 0 && (11)\\
-\xi^{(3)}_t + 2\,\xi^{(2)}_y &= 0 && (12)\\
-\xi^{(3)}_t + 2\,\xi^{(1)}_x &= 0 && (13)\\
(-c - c\,(b + \log(a\,u)))\,\phi^{(1)} + c\,u\,(b + \log(a\,u))\,\phi^{(1)}_u & && \\
+\phi^{(1)}_t - c\,u\,(b + \log(a\,u))\,\xi^{(3)}_t - \phi^{(1)}_{yy} - \phi^{(1)}_{xx} &= 0 && (14)
\end{aligned}
$$

From (3-8) we see that: $\xi^{(1)}$ and $\xi^{(2)}$ do not depends on u, $\xi^{(3)}$ depend only on t and $\phi^{(1)}$ is linear in u. So, we define

```
LXi = {Xi[1][x,y,t], Xi[2][x,y,t], Xi[3][t]}

LPhi = {Phi[1][x,y,t] u + Phi[2][x,y,t]}
```

and, in order to obtain a simplified system of equations, we run `YaLie[]` again

```
EDs = Lie[DA, VDs, VIs, VPs, LXi, LPhi];
```

The output is

$$
\begin{aligned}
2\,\xi^{(1)}_y + 2\,\xi^{(2)}_x &= 0 && (15)\\
-\xi^{(1)}_t + \xi^{(1)}_{yy} - 2\,\phi^{(1)}_x + \xi^{(1)}_{xx} &= 0 && (16)\\
-\xi^{(2)}_t - 2\,\phi^{(1)}_y + \xi^{(2)}_{yy} + \xi^{(2)}_{xx} &= 0 && (17)\\
-\xi^{(3)}_t + 2\,\xi^{(2)}_y &= 0 && (18)\\
-\xi^{(3)}_t + 2\,\xi^{(1)}_x &= 0 && (19)\\
\phi^{(2)}_t - c\,u\,\phi^{(1)} - c\,\phi^{(2)} - b\,c\,\phi^{(2)} & && \\
-c\,\log(a\,u)\,\phi^{(2)} - b\,c\,u\,\xi^{(3)}_t - c\,u\,\log(a\,u)\,\xi^{(3)}_t & && \\
+u\,\phi^{(1)}_t - u\,\phi^{(1)}_{yy} - \phi^{(2)}_{yy} - u\,\phi^{(1)}_{xx} - \phi^{(2)}_{xx} &= 0 && (20)
\end{aligned}
$$

From (18-19) we obtain,

$$
\begin{aligned}
\xi^{(1)} &= (1/2)\,x\,\xi^{(3)}_t + f(y,t)\\
\xi^{(2)} &= (1/2)\,y\,\xi^{(3)}_t + g(y,t)
\end{aligned}
$$

and using (15) we find

$$
\begin{aligned}
f(y,t) &= p(t)\,y + q(t) \\
g(x,t) &= -p(t)\,x + r(t)
\end{aligned}
$$

Then, we put

```
LXi = {1/2 D[Xi[3][t],t] x + p[t] y + q[t],
       1/2 D[Xi[3][t],t] y - p[t] x + r[t], Xi[3][t]}

LPhi = {Phi[1][x,y,t] u + Phi[2][x,y,t]}
```

With this information, we run again `YaLie` and `YaLieTeX` to obtain

$$
\begin{aligned}
-\left(y\,p_t\right) - q_t - \frac{x\,\xi^{(3)}_{tt}}{2} - 2\,\phi^{(1)}_x &= 0 && (21) \\
x\,p_t - r_t - \frac{y\,\xi^{(3)}_{tt}}{2} - 2\,\phi^{(1)}_y &= 0 && (22) \\
\phi^{(2)}_t - c\,u\,\phi^{(1)} - c\,\phi^{(2)} - b\,c\,\phi^{(2)} & \\
-c\,\log(a\,u)\;\phi^{(2)} - b\,c\,u\,\xi^{(3)}_t - c\,u\,\log(a\,u)\,\xi^{(3)}_t & \\
+u\,\phi^{(1)}_t - u\,\phi^{(1)}_{yy} - \phi^{(2)}_{yy} - u\,\phi^{(1)}_{xx} - \phi^{(2)}_{xx} &= 0 && (23)
\end{aligned}
$$

Hence, from (22-23)

$$p(t) = K_1,$$

$$\phi^{(1)}(x,y,t) = -(1/8)\,\xi^{(3)}_{tt}x^2 - (1/2)q_t\,x - (1/8)\,\xi^{(3)}_{tt}y^2 - (1/2)\,r_t\,y + \phi(t)$$

and, as a consequence,

$$
\begin{aligned}
-c\,u\,\phi(t) - c\,\phi^{(2)} - b\,c\,\phi^{(2)} - c\,\log(a\,u)\;\phi^{(2)} + \frac{c\,u\,x\,q_t}{2} + \frac{c\,u\,y\,r_t}{2} + \\
u\,\phi_t - b\,c\,u\,\xi^{(3)}_t - c\,u\,\log(a\,u)\;{\xi^{(3)}}_t - \frac{u\,x\,q_{tt}}{2} - \frac{u\,y\,r_{tt}}{2} + \frac{u\,\xi^{(3)}_{tt}}{2} + \\
\frac{c\,u\,x^2\,\xi^{(3)}_{tt}}{8} + \frac{c\,u\,y^2\,\xi^{(3)}_{tt}}{8} - \frac{u\,x^2\,\xi^{(3)}_{ttt}}{8} - \frac{u\,y^2\,\xi^{(3)}_{ttt}}{8} + \phi^{(2)}_t - \phi^{(2)}_{yy} - \phi^{(2)}_{xx} &= 0
\end{aligned}
$$

Since the former expression must be identically zero, we obtain

$$
\begin{aligned}
\xi^{(1)} &= K_1\,y + \frac{1}{c}K_2\exp(c\,t) + K_3 \\
\xi^{(2)} &= -K_1\,x + \frac{1}{c}K_4\exp(c\,t) + K_5 \\
\xi^{(3)} &= K_6 \\
\phi &= u\left(K_7\exp(c\,t) - \frac{1}{2}K_2 x\,\exp(c\,t) - \frac{1}{2}K_4\,y\,\exp(c\,t)\right)
\end{aligned}
$$

8 Application to nonclassical symmetries: Boussinesq equation

It is well known that there exist PDEs which posses symmetry reductions that cannot be obtained by using the classical Lie group method.

Recently, the nonclassical method [4] have been used to generate some new symmetries of significant PDEs. This nonclassical method involves to add, to the classical method, the condition

$$\Gamma = \{\zeta_1 = 0,\ \zeta_2 = 0,\ \ldots,\ \zeta_m = 0\}$$

where

$$\zeta_j = \sum_{i=1}^{n} \xi_i(x,u)\partial_{x_i} u_j - \phi_j$$

This means that one only requires the subset of $\mathcal{S}_\Sigma$ given by

$$\mathcal{S}_{\Sigma\cup\Gamma} = \{u(x)\ :\ \psi_1 = 0,\ \ldots,\ \psi_r = 0,\ \zeta_1 = 0,\ \ldots,\ \zeta_s = 0\}.$$

to be invariant under the infinitesimal transformation. Therefore the system of determining equations will be more general than those for the classical method, since derivatives $\partial_{x_1} u_k$ and $\partial_{x_2} u_k$, for example, are no longer independent.

The usual approach consists on determining the N-prolongation

$$\mathrm{pr}^N \mathbf{v}(\psi_i), \quad (i = 1, \ldots, r)$$

and on reducing it with respect to the augmented system $\Sigma\cup\Gamma$. However, in practice, several difficulties can appear. The following procedure, due to Clarkson [3], avoids many of the difficulties commonly encountered.

Consider, for example, the Boussinesq equation

$$u_{tt} + u\,u_{xx} + u_x^2 + u_{xxxx} = 0,$$

which arises in several physical applications. We start by calculating the fourth prolongation

```
Bq = D[u[x,t],{t,2}] + u[x,t] D[u[x,t],{x,2}] + D[u[x,t],x]^2 + D[u[x,t],{x,4}];

prL = Prolongación[Bq, {u}, {x,t}, 4];
```

Now, we must consider the invariant surface condition

$$\xi_1\, u_x + \xi_2\, u_t - \phi_1 = 0.$$

We can distinguish two cases:

1. $\xi_2 \neq 0$, and
2. $\xi_2 = 0$ and $\xi_1 \neq 0$

In the first case we set, without loss of generality, $\xi_2 = 1$, we substitute the adequate infinitesimals and we find out the derivatives which appear:

```
pr[1] = SInfinitesimales[prL, {Xi[1][x,t,u], 1}, {Phi[1][x,t,u]}];

Union[Cases[pr[1], Derivative[m___][u][x___], -1]]


    (0,1)         (0,2)         (1,0)         (1,1)
  {u      [x, t], u      [x, t], u      [x, t], u      [x, t],

    (2,0)         (3,0)         (4,0)
   u      [x, t], u      [x, t], u      [x, t]}
```

The prolongation must now be reduced with respect to the augmented system. Here we must be very careful. We must decide, for example, if the term u_{tt} would be elimnated by using the original equation or by deriving, with respect t, the invariant surface condition

$$u_t = \phi_1 - \xi_1 \, u_x.$$

In order to avoid this dilemma, we choose u_{xxxx} as the term to be eliminated by using the original equation as it does not contain derivative in t and therefore it is independent of u_t.

```
Sust[1] = Solve[Bq==0, D[u[x,t],{x,4}]][[1]]

pr[1] = pr[1] /. Sust[1];
```

With the aid of commands `DReduce` we can reduce, without difficulties, with respect to the surface invariant condition

```
pr[1] = DReduce[pr[1],
                D[u[x,t],t]== Phi[1][x,t,u[x,t]]-Xi[1][x,t,u[x,t]] D[u[x,t],x]];
```

Finally, with

```
EDs[1] = EDeterminantes[pr[1], {u}, {x,t}];
EDs[1] = Simplifica[EDs[1]];
```

we obtain the nonlinear system of determining equations

$$\begin{aligned}
\xi^{(1)}_u &= 0 \\
\phi^{(1)}_{uu} &= 0 \\
4\,\phi^{(1)}_{xu} - 6\,\xi^{(1)}_{xx} &= 0 \\
\phi^{(1)} + 2\,\xi^{(1)}\,\xi^{(1)}_t + 2\,u\,\xi^{(1)}_x + 4\,{\xi^{(1)}}^2\,\xi^{(1)}_x + 6\,\phi^{(1)}_{xxu} - 4\,\xi^{(1)}_{xxx} &= 0 \\
\phi^{(1)}_u + 2\,\xi^{(1)}_x &= 0 \\
\phi^{(1)}_{tt} - 2\,\xi^{(1)}_t\,\phi^{(1)}_x + 4\,\phi^{(1)}_t\,\xi^{(1)}_x - 4\,\xi^{(1)}\,\phi^{(1)}_x\,\xi^{(1)}_x + \qquad & \\
\phi^{(1)}\,\left(2\,\phi^{(1)}_{tu} + 4\,\phi^{(1)}_u\,\xi^{(1)}_x\right) + u\,\phi^{(1)}_{xx} + \phi^{(1)}_{xxxx} &= 0 \\
-2\,\phi^{(1)}_u\,\xi^{(1)}_t - \xi^{(1)}_{tt} + 2\,\phi^{(1)}_x - 2\,\xi^{(1)}_t\,\xi^{(1)}_x + 2\,u\,\phi^{(1)}_{xu} + \qquad & \\
\xi^{(1)}\,\left(-2\,\phi^{(1)}_{tu} - 8\,\phi^{(1)}_u\,\xi^{(1)}_x + 4\,{\xi^{(1)}_x}^2\right) - u\,\xi^{(1)}_{xx} + 4\,\phi^{(1)}_{xxxu} - \xi^{(1)}_{xxxx} &= 0
\end{aligned}$$

In the second case, the invariant surface condition is given by

$$u_x = \phi_1$$

and we can proceed similarly (here we use u_{tt} as the derivative to be eliminated from the Boussinesq equation).

```
pr[2] = SInfinitesimales[prL, {1, 0}, {Phi[1][x,t,u]}];

Sust[2] = Solve[Bq==0, D[u[x,t],{t,2}]][[1]]

pr[2] = pr[2] /. Sust[2];

pr[2] = DReduce[pr[2], D[u[x,t],x]== Phi[1][x,t,u[x,t]]];

EDs[2]= Simplifica[EDeterminantes[pr[2], {u}, {x,t}]]
```

Thus, we obtain

$$
\begin{aligned}
\phi^{(1)}_{uu} &= 0 \\
\phi^{(1)}_{tu} &= 0 \\
{\phi^{(1)}}^2\,\phi^{(1)}_{u} + \phi^{(1)}_{tt} + \phi^{(1)}\,\left(\phi^{(1)}\,\phi^{(1)}_{u} + \phi^{(1)}_{x}\right) + 4\,\phi^{(1)}_{u}\,\phi^{(1)}_{x}\,\phi^{(1)}_{xu} + & \\
u\,\phi^{(1)}_{xx} + 4\,\phi^{(1)}_{xu}\,\phi^{(1)}_{xx} + 6\,\phi^{(1)}_{x}\,\phi^{(1)}_{xxu} + \phi^{(1)}_{xxxx} + & \\
\phi^{(1)}\,\left(2\,\phi^{(1)}_{x} + 4\,{\phi^{(1)}_{u}}^2\,\phi^{(1)}_{xu} + 6\,\phi^{(1)}_{u}\,\phi^{(1)}_{xxu} + 4\,\phi^{(1)}_{xxxu}\right) + & \\
\phi^{(1)}\,\phi^{(1)}_{xu}\,\left(2\,u + 8\,\phi^{(1)}_{xu}\right) &= 0
\end{aligned}
$$

9 The commands and some technical notes

- `YaLie[Eqs, VDs, VIs, VPs, Opciones]`
  `YaLie[Eqs, VDs, VIs, VPs, LXi, LPhi, Opciones]`

  `YaLie` is the main procedure. It drives the process and performes the following steps:

  1. Determine the prolongation operator $\mathrm{pr}^N\,\mathbf{v}$ with N being the order of the system. Here, for the sake of efficiency, we do not use recursion.
  2. Apply $\mathrm{pr}^N\,\mathbf{v}$ to each equation $\psi_i = 0$
  3. Consider the original system Σ as a system of algebraic equations, where derivatives become the new variables and we solve it for unambiguously derivatives in the list `VPs` with the *Mathematica* function `Solve[]`. If `Solve[]` do not works, `YaLie[]` return `{}` and the warning message `LieError::Solve`
  4. Substitute the calculated solution into $\mathrm{pr}^N\,\mathbf{v}(\psi_i)$ to obtain
     $$\mathrm{pr}^N\,\mathbf{v}(\psi_i)\big|_{\Sigma}$$
  5. Obtain the determining equations for the formal infinitesimals ξ_i and ϕ_j by equating to zero the coefficients of all functionally independent expressions in the remaining derivatives.
  6. Simplify (in the possible) the equations previously obtained.
  7. If the infinitesimals `LXi` and `LPhi` are (perhaps partially) given, they are used to substitute in the equations.

It permits an iterative procedure, partially solving the determining equations and feeding back the obtained information into the program.

At the beginning, `YaLie[]` performs a simple check of its arguments and return `{}`, with a warning message if any of the following occurs:

- `Eqs` is not a list of equations (`lhs == rhs`).
- Some variables `Xi` or `Phi` are assigned.
- `VDs` or `VIs` is not a list of symbols.
- $\mathrm{pr}^N\,\mathbf{v}(\psi_i)\big|_\Sigma$ is not polynomial in the derivatives $u_k^{\mathbf{a}}$.

The unique option to `YaLie` is `Verborrea`, that can take the value `True` or `False` (default), in order to print some information about the process.

In the simplification process, `YaLie` uses the global variable `Pivotes` to save the canceled factors, assuming that such factors are not identically zero. Therefore, it is always advisable to check such variable.

- `Prolongación[Eq, VDs, VIs, k]`

  Determines the k-th prolongation operator $\mathrm{pr}^k\,\mathbf{v}$ and applies $\mathrm{pr}^k\,\mathbf{v}$ to the expression `Eq` in the variables `VDs` and `VIs`.

  The command `Prolongación[]` returns an expression in formal infinitesimals `Xi[k]` and `Phi[k]`.

  `Prolongación[]` performs a simple check of its arguments and return `{}`, with a warning message if some of the following occurs:

  - Some variables `Xi` or `Phi` are assigned.
  - `VDs` or `VIs` is not a list of symbols.

- `EDeterminantes[Prlg, VDs, VIs]`

  Obtains the determining equations for the formal infinitesimals ξ_i and ϕ_j, by equating to zero the coefficients of all the functionally independent derivatives $u_k^{\mathbf{a}}$.

  `Prlg` can be any expression, however it is more natural to see `Prlg` as the expression returned by `Prolongación[]`, with the adequate substitutions.

  Firstly, `EDeterminantes[]` performs a simple check of its arguments and return `{}`, with a warning message if some of the following occurs:

  - Some variables `Xi` or `Phi` are assigned.
  - `VDs` or `VIs` is not a list of symbols.
  - `Prlg` does not have a polynomial dependence in the derivatives $u_k^{\mathbf{a}}$.

- `Simplifica[EDs]`

  Performs a first (and simple) reduction of the determining equations `EDs`.

The algorithm assumes that `EDs` are expressions in the variables `Xi[k]`, `Phi[k]` and its derivatives. It is more natural to see `EDs` as the expression returned by `EDeterminantes[]`.

If any variable `Xi` o `Phi` is assigned, the routine returns `{}`, with a warning message.

`Simplifica` uses the global variable `Pivotes` to save the canceled factors. In the reduction process, `Simplifica` assumes that such factors are not identically zero.

- `SInfinitesimales[EDs, LXi, LPhi]`

  Substitutes the infinitesimals `LXi` and `LPhi` into the expressions `EDs`.

  This permits an iterative procedure, solving partially the determining equations and feeding back the obtained information into the program.

  Although `EDs` can be any expression in `Xi[]`, `Phi[]` and its derivatives, it is more natural to see `EDs` as the expression returned by `EDeterminantes[]` or `Simplifica[]`.

  At the beginning, `SInfinitesimales[]` performs a simple check of its arguments and returns `{}`, with a warning message if some of the following occurs:

  – `EDs` are not expressions in the variables `Xi[]`, `Phi[]` and its derivatives.

  – Some variables `Xi` or `Phi` are assigned.

- `YaLieTeX["Archivo", EDs]`
  `YaLieTeX["Archivo", EDs, LXi, LPhi]`

  Writes the obtained determining equations in the TEXfile `Archivo`.

  In order to get the `YaLieTeX` functions working properly, the variable `$Path` should include the folder that contains the file `yalie.mx`.

  If the variables `LXi` and `LPhi` are not given in the command, the function `YaLieTeX` would use the formal infinitesimals.

  The command `DFormat` has a decisive influence in the way the TEXfile is written. In general, it is preferable to use the command `DFormat[True]` before using `YaLieTeX`.

  In the *manual* mode, it may occurs that some variables are not assigned and, therefore, will not be written in the file.

- `DFormat[True | False]`

  Switch on-off the short format for derivatives.

  `DFormat` is an extension of the `DFormat` package by Pekka Janhunen [7].

  Remark: If you read in the same session the `YaLie` package and the `DFormat` package, you would get the warning `DFormat::shdw:`. In this case, do not worry: it works fine.

- `DReduce[F, D[u, x]==expr]`

  Reduces `F` with respect to the condition `D[u,x]==expr`.

10 Why using YaLie?. A benchmark

In mathsource we find the package `Lie` by Gerd Baumann, so why a new Lie pacakge?. Well, all I can say is this: the Gerd Baumann Lie package is not really a Mathematica Package. It likes as a traduction of other programs which deal with Lie symmetries, and forthcoming it is slow. In order to see this I builded a time table with the following cases using a Pentium II (333 MgH.) and a Athlon (850 MgH.) both with 256 Mg. RAM

- Korteweg de Vries family.

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

  with $u = u(x,t)$.

- Kadomtsev-Petviashvili.

$$\text{KP} \equiv \partial_x \left(u_t + \frac{3}{2} u u_x + \frac{1}{4} u_{xx} \right) + \frac{3}{4} u_{yy} = 0$$

  with $u = u(x,y,t)$.

- Schrödinger.

$$\text{Sch} \equiv i\psi_t + \psi_{xx} + 2|\psi|^2\psi = 0$$

  with $\psi = u(x,t) + i\,v(x,t)$.

- Madelung (Mdg).

$$\begin{aligned} \psi^1_t + \partial_x(\psi^1 \phi^1_x) &= 0 \\ \psi^2_t + \partial_x(\psi^2 \phi^2_x) &= 0 \\ \phi^1_t + \frac{1}{2}\left(\phi^1_x\right)^2 + \Gamma_1 \psi_1 + \frac{1}{8}\left(\frac{\psi^1_x}{\psi^1}\right)^2 - \frac{1}{4}\left(\frac{\psi^1_{xx}}{\psi_1}\right) + \psi^2 &= 0 \\ \phi^2_t + \frac{1}{2}\left(\phi^2_x\right)^2 + \Gamma_2 \psi_2 + \frac{1}{8}\left(\frac{\psi^2_x}{\psi^2}\right)^2 - \frac{1}{4}\left(\frac{\psi^2_{xx}}{\psi^2}\right) + \psi^1 &= 0 \end{aligned}$$

  with $\psi^i = \psi^i(x,t)$ y $\phi^i = \phi^i(x,t)$, $(i = 1,2)$.

- Maxwell-Bloch (MB).

$$\begin{aligned} F_t - \frac{1}{4\alpha} i\Delta F + \sigma(F - P) &= 0 \\ P_t + (1 + i\,\delta)P - F\,D &= 0 \\ D_t + \gamma\left(D - r + \frac{1}{2}\left(\overline{F}P + F\overline{P}\right)\right) &= 0 \end{aligned}$$

  with $F = f(x,\, y,\, t) + i\, g(x,\, y,\, t),\ P = p(x,\, y,\, t) + i\, q(x,\, y,\, t)$ and $D = D(x,y,t)$.

- A difficult MHD.

$$\begin{aligned}
\varrho_t + \operatorname{div}(\varrho\,\mathbf{v}) &= 0\\
\operatorname{div}\mathbf{H} &= 0\\
\varrho(\mathbf{v}_t + \nabla\mathbf{v}\cdot\mathbf{v}) + \nabla P - R_h \operatorname{rot}\mathbf{H}\times\mathbf{H} - \chi\Delta\mathbf{v} &\\
-(\zeta+\chi/3)\nabla(\operatorname{div}\mathbf{v}) &= 0\\
\mathbf{H}_t - \operatorname{rot}(\mathbf{v}\times\mathbf{H}) - \frac{1}{R_s}\operatorname{rot}(v_h \operatorname{rot}\mathbf{H}) &= 0\\
P_t + \mathbf{v}\cdot\nabla P + \gamma P \operatorname{div}\mathbf{v} - (\gamma-1)\frac{R_h}{R_s} v_h \operatorname{rot}\mathbf{H}\cdot\operatorname{rot}\mathbf{H} &\\
-(\gamma-1)\left[\sum_{k=1}^{3}\sigma^{(k)}\nabla v_k + \operatorname{div}(\kappa\nabla(P/\varrho))\right] &= 0
\end{aligned}$$

with $\mathbf{v} = (v_1(x,y,z,t), v_2(x,y,z,t), v_3(x,y,z,t))$, $P = P(x,y,z,t)$, $\varrho = \varrho(x,y,z,t)$, $\mathbf{H} = (H_1(x,y,z,t), H_2(x,y,z,t), H_3(x,y,z,t))$ and $\sigma^k = (\sigma_{k1}, \sigma_{k2}, \sigma_{k3})$.

Time table in seconds

	PII (333 MgH)		Athlon (850 MgH)	
Equation	`YaLie`	`Lie`	`YaLie`	`Lie`
KdV	0.22	0.99	0.06	0.27
mKdV	0.28	0.88	0.11	0.22
cKdV	0.27	1.10	0.06	0.39
lKdV	0.28	0.77	0.11	0.27
Sch	0.55	2.75	0.17	0.72
KP	0.87	3.84	0.22	1.10
Mdg	4.18	43.50	1.15	15.27
MB	10.49	795.87	2.97	234.64
MHD	4984.15	*	1580.75	*

* `No more memory available.`
`Mathematica kernel has shut down.`
(after 4h)

You can mimic the precedent table using the notebook `benchmark.nb` in your computer.

References

[1] G. Baumann. Lie symmetries of differential equations. A mathematica program to determine lie-symmetries. In `www.mathsource.com`, `/Applications/Mathematics/0202-622`.

[2] G.W. Bluman and S. Kumei. *Symmetries and differential equations.* Springer-Verlag, 1989.

[3] P.A. Clarkson and E. Mansfield. Algorithms for the nonclassical method of symmetry reductions. *SIAM Journal on Applied Mathematics* 54, no. 6, 1693–1719, 1994

[4] P.A. Clarkson. Nonclassical symmetry reductions of the Boussinesq equation. *Chaos, Solitons and Fractals*, 5(12):2261–2301, 1995.

[5] J.M. Díaz. *Soluciones de similaridad de la ecuación de Laplace.* Boletín ROA 6/97. Real Instituto y Observatorio de la Armada, 1997.

[6] M.L. Gandarias and L. del Águila. Symmetry reductions for a nonlinear diffusion-absorption equation in two spatial dimensions. *Europhys. Lett.*, 42 (6):589–594, 1998.

[7] P. Janhunen. Dformat. Fancy formatting of partial derivatives. In `www.mathsource.com`, `/Enhacements/Algebraic/0202-161.`

[8] P.J. Olver. *Applications of Lie groups to differential equations.* Springer-Verlag, 1986.

[9] G.J. Reid. Algorithms for reducing a system of PDEs to standard form, determining the dimension of its solution space and calculating its taylor series solution. *Euro. Jnl. of Applied Mathematics*, 2:293–318, 1991.