

# Computing ODE Symmetries as Abnormal Variational Symmetries

Paulo D. F. Gouveia  
pgouveia@ipb.pt

Delfim F. M. Torres  
delfim@ua.pt

Bragança Polytechnic Institute  
5301-854 Bragança, Portugal

University of Aveiro  
3810-193 Aveiro, Portugal

## Abstract

We give a new computational method to obtain symmetries of ordinary differential equations. The proposed approach appears as an extension of a recent algorithm to compute variational symmetries of optimal control problems [Comput. Methods Appl. Math. **5** (2005), no. 4, pp. 387-409], and is based on the resolution of a first order linear PDE that arises as a necessary and sufficient condition of invariance for abnormal optimal control problems. A computer algebra procedure is developed, which permits to obtain ODE symmetries by the proposed method. Examples are given, and results compared with those obtained by previous available methods.

**Mathematics Subject Classification 2000:** 34-04; 49-04; 34C14; 49K15.

**Keywords.** Symmetries, variational symmetries, dynamic symmetries, ODEs, computer algebra systems, optimal control, abnormality.

## 1 Introduction

Sophus Lie was the first to introduce the use of symmetries into the study of differential equations, Emmy Noether the first to recognize the important role of symmetries in the calculus of variations. Currently, all the computer algebra systems that address differential equations provide several tools to help the user with the analysis of Lie symmetries. Recently, the authors developed a computer algebra package for the automatic computation of Noether variational symmetries in the calculus of variations and optimal control [5], now available as part of the Maple Application Center at [http://www.maplesoft.com/applications/app\\_center\\_view.aspx?AID=19](http://www.maplesoft.com/applications/app_center_view.aspx?AID=19)

The omnipresent tools for Lie symmetries provide a great help for the search of solutions of ODEs, their classification, order reduction, proof of integrability, or in the construction of first integrals. From the mathematical point of view, a ODE symmetry is described by a group of transformations that keep the ordinary differential equation invariant. Depending on the type of transformations

one is considering, different symmetries are obtained. An important class of symmetries is obtained considering a one-parameter family of transformations, which form a local Lie group. Those transformations are often represented by a set of functions known as the infinitesimal generators. From the practical point of view, the determination of the infinitesimal generators that define a symmetry for a given ODE is, in general, a complex task [6, 11]. To address the problem, we follow a different approach.

We propose a new method for computing symmetries of ODEs by using a Noetherian perspective. Making use of our previous algorithm [5], that has shown up good results for the computation of Noether variational symmetries of problems of the calculus of variations and optimal control, we look to an ODE as being the control system of an optimal control problem. Then, we obtain symmetries for the ODE by computing the abnormal variational symmetries of the associated optimal control problem.

This paper is organized as follows. In §2, the necessary concepts associated with variational symmetries in optimal control are reviewed. The new method for computing symmetries of ODEs is explained in §3. The method is illustrated in §4, where we compute symmetries for three distinct ODEs and compare the results with the ones obtained by the standard procedures available in `Maple`. We end the paper with some conclusions and comments §5. The definitions of the new `Maple` procedure that implements our method are given in Appendix.

## 2 Symmetries in optimal control

Without loss of generality, we consider the optimal control problem in Lagrange form: to minimize an integral functional

$$I[\mathbf{x}(\cdot), \mathbf{u}(\cdot)] = \int_a^b L(t, \mathbf{x}(t), \mathbf{u}(t)) dt \quad (1)$$

subject to a control system described by a system of ordinary differential equations of the form

$$\dot{\mathbf{x}}(t) = \boldsymbol{\varphi}(t, \mathbf{x}(t), \mathbf{u}(t)), \quad (2)$$

together with appropriate boundary conditions. The Lagrangian  $L : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$  and the velocity vector  $\boldsymbol{\varphi} : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$  are assumed to be continuously differentiable functions with respect to all their arguments. The controls  $\mathbf{u} : [a, b] \rightarrow \mathbb{R}^m$  are piecewise continuous functions; the state variables  $\mathbf{x} : [a, b] \rightarrow \mathbb{R}^n$  continuously differentiable functions.

The celebrated Pontryagin Maximum Principle [10] (PMP for short) gives a first-order necessary optimality condition. The PMP can be proved from a general Lagrange multiplier theorem. One introduces the Hamiltonian function

$$H(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) = \psi_0 L(t, \mathbf{x}, \mathbf{u}) + \boldsymbol{\psi}^T \cdot \boldsymbol{\varphi}(t, \mathbf{x}, \mathbf{u}), \quad (3)$$

where  $(\psi_0, \boldsymbol{\psi}(\cdot))$  are the “Lagrange multipliers”, with  $\psi_0 \leq 0$  a constant and  $\boldsymbol{\psi}(\cdot)$  a  $n$ -vectorial piecewise  $C^1$ -smooth function, and the multiplier theorem

asserts that the optimal control problem is equivalent to the maximization of the augmented functional

$$J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), \psi_0, \boldsymbol{\psi}(\cdot)] = \int_a^b (H(t, \mathbf{x}(t), \mathbf{u}(t), \psi_0, \boldsymbol{\psi}(t)) - \boldsymbol{\psi}(t)^\top \cdot \dot{\mathbf{x}}(t)) \, dt. \quad (4)$$

**Definition 1.** A quadruple  $(\mathbf{x}(\cdot), \mathbf{u}(\cdot), \psi_0, \boldsymbol{\psi}(\cdot))$  satisfying the Pontryagin Maximum Principle is said to be a (Pontryagin) extremal. An extremal is said to be normal when  $\psi_0 \neq 0$ , abnormal when  $\psi_0 = 0$ .

Let  $\mathbf{h}^s : [a, b] \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$  be a one-parameter group of  $\mathbb{C}^1$  transformations of the form

$$\begin{aligned} \mathbf{h}^s(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) = \\ (h_t^s(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}), \mathbf{h}_\mathbf{x}^s(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}), \mathbf{h}_\mathbf{u}^s(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}), \mathbf{h}_\boldsymbol{\psi}^s(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi})). \end{aligned} \quad (5)$$

Without loss of generality, we assume that the identity transformation of the group (5) is obtained when the parameter  $s$  is zero:

$$\begin{aligned} h_t^0(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) &= t, \quad \mathbf{h}_\mathbf{x}^0(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) = \mathbf{x}, \\ \mathbf{h}_\mathbf{u}^0(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) &= \mathbf{u}, \quad \mathbf{h}_\boldsymbol{\psi}^0(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) = \boldsymbol{\psi}. \end{aligned}$$

Associated with a one-parameter group of transformations (5), we introduce its infinitesimal generators:

$$\begin{aligned} T(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) &= \left. \frac{\partial}{\partial s} h_t^s \right|_{s=0}, \quad \mathbf{X}(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) = \left. \frac{\partial}{\partial s} \mathbf{h}_\mathbf{x}^s \right|_{s=0}, \\ \mathbf{U}(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) &= \left. \frac{\partial}{\partial s} \mathbf{h}_\mathbf{u}^s \right|_{s=0}, \quad \boldsymbol{\Psi}(t, \mathbf{x}, \mathbf{u}, \psi_0, \boldsymbol{\psi}) = \left. \frac{\partial}{\partial s} \mathbf{h}_\boldsymbol{\psi}^s \right|_{s=0}. \end{aligned} \quad (6)$$

We can define variational invariance using the augmented functional (4) and the one-parameter group of transformations (5) or an equivalent condition in terms of the generators (6):

**Definition 2** ([3, 13]). An optimal control problem (1)-(2) is said to be invariant under (6) or, equivalently, (6) is said to be a symmetry of the problem (1)-(2) if

$$\frac{\partial H}{\partial t} T + \frac{\partial H}{\partial \mathbf{x}} \cdot \mathbf{X} + \frac{\partial H}{\partial \mathbf{u}} \cdot \mathbf{U} + \frac{\partial H}{\partial \boldsymbol{\psi}} \cdot \boldsymbol{\Psi} - \boldsymbol{\Psi}^\top \cdot \dot{\mathbf{x}} - \boldsymbol{\psi}^\top \cdot \frac{d\mathbf{X}}{dt} + H \frac{dT}{dt} = 0, \quad (7)$$

with  $H$  the Hamiltonian (3).

A computational algorithm to obtain the infinitesimal generators  $T$ ,  $\mathbf{X}$ ,  $\mathbf{U}$ , and  $\boldsymbol{\Psi}$  that form a variational symmetry (7) for a given optimal control problem (1)-(2) was developed in [5]. Here we remark that the abnormal variational symmetries (i.e. the ones associated with  $\psi_0 = 0$ ) obtained by the method introduced in [5] provide symmetries for ordinary differential equations.

### 3 Computing ODE symmetries from an optimal control perspective

To ODE symmetries (see e.g. [1, 15]) we associate a smaller group of infinitesimal transformations than that used in optimal control: we are only interested in transformations of the independent variable  $t$  and transformations of the dependent variables  $\mathbf{x}$ . Let  $\mathbf{g}^s : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R} \times \mathbb{R}^n$  be a one-parameter group of  $\mathbb{C}^1$  transformations of the form

$$\mathbf{g}^s(t, \mathbf{x}) = (g_t^s(t, \mathbf{x}), \mathbf{g}_\mathbf{x}^s(t, \mathbf{x})),$$

with  $\mathbf{g}^0(t, \mathbf{x}) = (t, \mathbf{x})$ . Let us denote the respective *infinitesimal generators* by

$$\xi(t, \mathbf{x}) = \left. \frac{\partial}{\partial s} g_t^s(t, \mathbf{x}) \right|_{s=0}, \quad \eta(t, \mathbf{x}) = \left. \frac{\partial}{\partial s} \mathbf{g}_\mathbf{x}^s(t, \mathbf{x}) \right|_{s=0}. \quad (8)$$

Our method begins by identifying a richer set of variational symmetries in the form (6), from which we then obtain (8). In this section, we explain in detail how to arrive to the set of generators (8) that keep an ODE invariant.

We are interested in determining symmetries for systems of ODEs in the canonical form

$$\begin{cases} y_1^{(r_1)} &= \phi_1(t, y_1, \dot{y}_1 \dots y_1^{(r_1-1)}, \dots, y_n, \dot{y}_n \dots y_n^{(r_n-1)}), \\ &\vdots \\ y_n^{(r_n)} &= \phi_n(t, y_1, \dot{y}_1 \dots y_1^{(r_1-1)}, \dots, y_n, \dot{y}_n \dots y_n^{(r_n-1)}), \end{cases} \quad (9)$$

where functions  $\phi_k : \mathbb{R} \times \mathbb{R}^{\sum_{i=1}^n r_i} \rightarrow \mathbb{R}$ ,  $k = 1, \dots, n$ , are continuously differentiable with respect to all their arguments. To write the system (9) of differential equations as a control system (2), we begin by converting it as a system of equations of first order. For that we introduce a new set of variables, represented by the vector  $\mathbf{x}$ :

$$\begin{aligned} \mathbf{x} &= [x_1, \dots, x_r]^T \\ &= \left[ y_1, \dot{y}_1, \dots, y_1^{(r_1-1)}, \dots, y_n, \dot{y}_n, \dots, y_n^{(r_n-1)} \right]^T, \end{aligned} \quad (10)$$

where  $r = \sum_{i=1}^n r_i$ . With this notation, we get the control system

$$\begin{cases} \dot{x}_1 &= x_2, \\ &\vdots \\ \dot{x}_{r_1-1} &= x_{r_1}, \\ \dot{x}_{r_1} &= \phi_1(t, \mathbf{x}), \end{cases} \quad \dots \quad \begin{cases} \dot{x}_{r_1+\dots+r_{n-1}+1} &= x_{r_1+\dots+r_{n-1}+2}, \\ &\vdots \\ \dot{x}_{r_1+\dots+r_{n-1}} &= x_{r_1+\dots+r_n}, \\ \dot{x}_{r_1+\dots+r_n} &= \phi_n(t, \mathbf{x}), \end{cases} \quad (11)$$

with  $r$  state variables but no control variables (i.e., (11) is a particular case of (2) where  $\varphi$  does not depend on  $\mathbf{u}$ ).

To use the formalism of optimal control and the notion of variational symmetry [5], one thing is missing: the existence of an integral functional (1) to be minimized, and whose Lagrangian  $L$  enters into the definition of the Hamiltonian (3), thus being necessary for computing symmetries by (7). However, if we restrict ourselves to the abnormal case, where the cost functional has no role, i.e. if we fix  $\psi_0 = 0$ , then the Hamiltonian  $H$  does not depend on the Lagrangian  $L$  and we can look to our system (11) as an optimal control problem. If we only consider the abnormal case, the control system (11) is everything one needs to write (7) and find symmetries.

We are now in conditions to use our **Maple** optimal control package [5] and its procedure **Symmetry** to obtain symmetries for systems of ODEs (9). We only need to rewrite (9) as in (11) and then call function **Symmetry** of [5] for the abnormal case. After using this technique with several concrete examples, and to be able to compare the obtained results with the ones from standard techniques, we concluded that a great manual effort is necessary at each particular problem in converting the initial system into the canonical form, then to (11), and finally recovering the initial notation to compare the results with those obtained by the tools already available in the Computer Algebra System **Maple**. To do all the process in an entirely automatic way, and also to optimize the algorithm, we define here a new **Maple** function **odeSymm** (see Appendix) whose purpose is to compute symmetries for systems (9) of ODEs.

## The algorithm

We consider the abnormal Hamiltonian

$$H(t, \mathbf{x}, \boldsymbol{\psi}) = \boldsymbol{\psi}^T \cdot \boldsymbol{\varphi}(t, \mathbf{x}),$$

where the velocity vector is given by

$$\begin{aligned} \boldsymbol{\varphi}(t, \mathbf{x}) = [x_2, \dots, x_{r_1}, \phi_1(t, \mathbf{x}), x_{r_1+2}, \dots, x_{r_1+r_2}, \phi_2(t, \mathbf{x}), \dots \\ \dots, x_{r_1+\dots+r_{n-1}+2}, \dots, x_r, \phi_n(t, \mathbf{x})]^T. \end{aligned} \quad (12)$$

Condition (7) simplifies to

$$\boldsymbol{\psi}^T \cdot \left( \frac{\partial \boldsymbol{\varphi}}{\partial t} T + \frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{x}} \cdot \mathbf{X} - \frac{d\mathbf{X}}{dt} + \boldsymbol{\varphi} \frac{dT}{dt} \right) + \boldsymbol{\Psi}^T \cdot (\boldsymbol{\varphi} - \dot{\mathbf{x}}) = 0. \quad (13)$$

Thus, given a system of ODEs, we determine the infinitesimal generators  $\xi$  and  $\boldsymbol{\eta}$  (8), which define a symmetry of the given ODEs, in the following way:

1. First we rewrite the given system of ODEs in the form (9);<sup>1</sup>
2. We represent the dependent variables  $y_i$  and their derivatives  $y_i^{(j)}$ ,  $i = 1, \dots, n$ ,  $j = 1, \dots, r_i - 1$ , present in functions  $\phi_k$  of the canonical system (9), by a new set of variables  $\mathbf{x}$ , in accordance with (10);

---

<sup>1</sup>We only deal with differential equations that can be written in the canonical form (9).

3. We define the velocity vector  $\varphi$  (12);
4. We substitute  $\varphi$  and its partial derivatives into (13);
5. From equation (13), we determine the variational generators  $T(t, \mathbf{x}, \psi)$ ,  $\mathbf{X}(t, \mathbf{x}, \psi)$  and  $\Psi(t, \mathbf{x}, \psi)$ ;
6. In the results obtained, we go back to the initial notation of variables, by means of the inverse relations of (10);
7. From the obtained set of variational generators  $T$ ,  $\mathbf{X}$  and  $\Psi$ , we extract the subset of generators  $\xi$  and  $\eta$ ,

$$\xi \equiv T, \quad \eta_i \equiv X_{1+\sum_{k=1}^{i-1} r_k}, \quad i = 1, \dots, n,$$

which represent symmetries for the given system of ODEs.

The infinitesimal generators obtained in step 5 are functions of the auxiliary variables  $\mathbf{x}$ . Since in step 6 the variables  $\mathbf{x}$  resume to its initial meaning, we conclude that our method is able to give dynamic symmetries (cf. Example 2). Indeed, the generators may involve derivatives of the dependent variables:

$$(\xi, \eta) \equiv (\xi(t, \mathbf{y}, \dot{\mathbf{y}}, \dots), \eta(t, \mathbf{y}, \dot{\mathbf{y}}, \dots))$$

with  $(t, \mathbf{y}, \dot{\mathbf{y}}, \dots) = (t, y_1, \dot{y}_1 \dots y_1^{(r_1-1)}, \dots, y_n, \dot{y}_n \dots y_n^{(r_n-1)})$ .

We now address the non-trivial part of our seven-step algorithm, which resides precisely in step 5: the determination of the associated variational generators. We use the following strategy. Expanding the total derivatives

$$\begin{aligned} \frac{dT}{dt} &= \frac{\partial T}{\partial t} + \frac{\partial T}{\partial \mathbf{x}} \cdot \dot{\mathbf{x}} + \frac{\partial T}{\partial \psi} \cdot \dot{\psi}, \\ \frac{d\mathbf{X}}{dt} &= \frac{\partial \mathbf{X}}{\partial t} + \frac{\partial \mathbf{X}}{\partial \mathbf{x}} \cdot \dot{\mathbf{x}} + \frac{\partial \mathbf{X}}{\partial \psi} \cdot \dot{\psi}, \end{aligned}$$

we write equation (13) as a polynomial

$$A(t, \mathbf{x}, \psi) + B(t, \mathbf{x}, \psi) \cdot \dot{\mathbf{x}} + C(t, \mathbf{x}, \psi) \cdot \dot{\psi} = 0 \quad (14)$$

in the  $2r$  derivatives  $\dot{\mathbf{x}}$  and  $\dot{\psi}$ :

$$\begin{aligned} &\psi^T \cdot \left( \frac{\partial \varphi}{\partial t} T + \frac{\partial \varphi}{\partial \mathbf{x}} \cdot \mathbf{X} + \varphi \frac{\partial T}{\partial t} - \frac{\partial \mathbf{X}}{\partial t} \right) + \Psi^T \cdot \varphi \\ &+ \left( -\Psi^T + \psi^T \cdot \varphi \cdot \frac{\partial T}{\partial \mathbf{x}} - \psi^T \cdot \frac{\partial \mathbf{X}}{\partial \mathbf{x}} \right) \cdot \dot{\mathbf{x}} \\ &+ \left( \psi^T \cdot \varphi \cdot \frac{\partial T}{\partial \psi} - \psi^T \cdot \frac{\partial \mathbf{X}}{\partial \psi} \right) \cdot \dot{\psi} = 0. \end{aligned} \quad (15)$$

The terms in (15), which involve derivatives with respect to vectors, are expanded in row-vectors or in matrices, depending, respectively, if the function is

a scalar function or a vectorial one. Equation (15) is a differential equation in the  $2r + 1$  unknown functions  $T, X_1, \dots, X_r$  and  $\Psi_1, \dots, \Psi_r$ . This equation must hold for all  $\dot{x}_1, \dots, \dot{x}_r, \dot{\psi}_1, \dots, \dot{\psi}_r$ , and therefore the coefficients  $A, B$ , and  $C$  of polynomial (14) must vanish, i.e.,

$$\begin{cases} \psi^T \cdot \left( \frac{\partial \varphi}{\partial t} T + \frac{\partial \varphi}{\partial \mathbf{x}} \cdot \mathbf{X} + \varphi \frac{\partial T}{\partial t} - \frac{\partial \mathbf{X}}{\partial t} \right) + \Psi^T \cdot \varphi = 0, \\ -\Psi^T + \psi^T \cdot \varphi \cdot \frac{\partial T}{\partial \mathbf{x}} - \psi^T \cdot \frac{\partial \mathbf{X}}{\partial \mathbf{x}} = \mathbf{0}, \\ \psi^T \cdot \varphi \cdot \frac{\partial T}{\partial \psi} - \psi^T \cdot \frac{\partial \mathbf{X}}{\partial \psi} = \mathbf{0}. \end{cases} \quad (16)$$

Although a system of partial differential equations, solving (16) is possible using the **Maple** command `pdsolve` because the system is of the first order, homogeneous, and linear with respect to the unknown functions and their derivatives. We also remark that since system (16) is homogeneous, we always have, as trivial solution,  $(T, \mathbf{X}, \Psi) = \mathbf{0}$ .

When dealing with ODEs with several dependent variables and high-order derivatives, the number of calculations to be done is big enough, and the help of the computer is more than welcome. We use the computer algebra system **Maple** 10 to define a new procedure `odeSymm` that does all the cumbersome computations for us – all the steps 1 to 7 of our algorithm.

Our procedure `odeSymm` receives, as input, a system of ODEs, and returns, as output, a family of symmetries  $(\xi, \eta)$  – see definition of procedure `odeSymm` in Appendix. To optimize the resolution of (16), we give the possibility to pass several optional parameters to `odeSymm`. These optional parameters are described in the Appendix and illustrated with concrete examples in §4. Here we just mention that, by default, we use the method of separation of variables (see [8, 15]) to solve (16). More precisely, we follow [2]: the generators are replaced by the sum of unknown functions, one for each variable. For example,  $T(t, x_1, x_2) = T_1(t) + T_2(x_1) + T_3(x_2)$ . Through the optional parameters, one can use the default solving process of the **Maple** solver `pdsolve` or other specific methods (cf. Example 1).

## 4 Illustrative examples

To show the functionality and the usefulness of our new procedure `odeSymm`, we consider three concrete problems found in the literature. All the examples were carried out with **Maple** version 10 on a 1.4GHz 512MB RAM Pentium Centrino. The running time of procedure `odeSymm` is indicated, for each example, in seconds.

**Example 1** (Kamke’s ODE 120). *We begin with a first order ODE found in Kamke’s book [7]:*

```
> ode:= t*diff(y(t),t)-y(t)*(t*ln(t^2/y(t))+2)=0;
```

$$\text{ode} := t \frac{d}{dt} y(t) - y(t) \left( t \ln \left( \frac{t^2}{y(t)} \right) + 2 \right) = 0$$

To obtain symmetries of the equation we use our **Maple** procedure `odeSymm` with the additional parameter `hint=noint`. This means that we will use the default method of resolution of PDEs of the **Maple** solver `pdsolve`. If the optional parameter `hint` is not used (see Examples 2 and 3 below), our procedure `odeSymm` uses the method of separation of variables. We obtain the following infinitesimal generators (0.72 sec):

```
> gerad:= odeSymm(ode, y(t), split, hint=nohint);
```

$$\text{gerad} := \left[ \xi = -\frac{1}{2}, \eta = -\frac{y}{t} \right], \left[ \xi = 0, \eta = -\frac{y}{e^t} \right]$$

One can test the validity of the obtained symmetries with the `symtest` command of the **DEtools** **Maple** package:

```
> map(DEtools[symtest], [gerad], ode, y(t));
```

[0, 0]

The `symtest` confirm that the infinitesimal generators leave the given ODE invariant, i.e., the generators obtained by our method give indeed a symmetry to Kamke's ODE 120. It is interesting to remark that, without the knowledge of the computed symmetries, the ODE **Maple** solver `dsolve` is not able to integrate the ODE:

```
> dsolve(ode, y(t), class);
```

However, when one gives to the **Maple** solver the infinitesimal generators found by our method, the ODE is correctly solved:

```
> dsolve(ode, y(t), HINT=[gerad]);
```

$$y(t) = t^2 e^{(C_1 - 1)e^{-t}}$$

It is also interesting to note that our method is able to find one symmetry that is different from the ones obtained using the standard methods of the literature. The **Maple** system provides nine different algorithms to compute symmetries of ODEs through the command `symgen` of the **DEtools** package. All the available schemes for determining the infinitesimal generators – option `way=all` – are not able to identify our pair of infinitesimals  $[\xi = 0, \eta = -\frac{y}{e^t}]$ :

```
> DEtools[symgen](ode, y(t), way=all);
```

$$\left[ \xi = 1, \eta = 2 \frac{y}{t} \right], \left[ \xi = 0, \eta = y \ln \left( \frac{t^2}{y} \right) \right]$$

**Example 2** (Damped Harmonic Oscillator). We consider a harmonic oscillator with restoring force  $-kx$ , emerged in a liquid in such a way that the motion of the mass  $m$  is damped by a force proportional to its velocity. Using Newton's second law one obtains, as the equation of motion, the following second order differential equation [9, pp. 432–434]:

```
> EL:= m*diff(x(t),t,t)+a*diff(x(t),t)+k*x(t)=0;
```



$$EL := m \frac{d^2}{dt^2} x(t) + a \frac{d}{dt} x(t) + kx(t) = 0$$

The symmetries for this equation are easily obtained with our **Maple** procedure `odeSymm` (1.21 sec)

```
> gerad:= odeSymm(EL, x(t), split);
```

$$\text{gerad} := [\xi = 0, \eta = x], \left[ \xi = 0, \eta = -\frac{mx'}{k} \right], [\xi = 1, \eta = 0], \\ \left[ \xi = 0, \eta = e^{-\frac{at}{2m}} e^{\frac{t\sqrt{a^2-4km}}{2m}} \right], \left[ \xi = 0, \eta = e^{-\frac{at}{2m}} e^{-\frac{t\sqrt{a^2-4km}}{2m}} \right]$$

One can confirm that these infinitesimals represent valid symmetries for the differential equation:

```
> map(DEtools[symtest], [gerad], EL, x(t));
```

$$[0, 0, 0, 0, 0]$$

Note that the output of our `odeSymm` procedure includes a dynamical symmetry: the derivative of the dependent variable is present in the second pair of obtained infinitesimal generators.

**Example 3** (Kepler's problem). We now consider the Kepler's problem: a problem of the calculus of variations – see [14, p. 217]. In this case, the Lagrangian depends on two dependent variables  $q_1$  and  $q_2$ :

$$L(t, \mathbf{q}, \dot{\mathbf{q}}) = \frac{m}{2} (\dot{q}_1^2 + \dot{q}_2^2) + \frac{K}{\sqrt{q_1^2 + q_2^2}}.$$

We will use the proposed method to determine symmetries for the corresponding Euler-Lagrange differential equation. The Euler-Lagrange equation is trivially obtained using our package of the calculus of variations [4, Example 5.2]:

```
> L:= m/2*(v[1]^2+v[2]^2)+K/sqrt(q[1]^2+q[2]^2);
```

$$L := \frac{1}{2} m (v_1^2 + v_2^2) + \frac{K}{\sqrt{q_1^2 + q_2^2}}$$

```
> EL:= CLaws[CV][EulerLagrange](L, t, [q[1],q[2]], [v[1],v[2]]);
```

$$EL := \left\{ -m \frac{d^2}{dt^2} q_1(t) - \frac{K q_1(t)}{(q_1(t)^2 + q_2(t)^2)^{3/2}} = 0, \right. \\ \left. -m \frac{d^2}{dt^2} q_2(t) - \frac{K q_2(t)}{(q_1(t)^2 + q_2(t)^2)^{3/2}} = 0 \right\}$$

In this case, the Euler-Lagrange equation is a system of two second order ODEs. Our `odeSymm` procedure is able to determine symmetries for systems of differential equations as well (13.32 sec):

```
> odeSymm(EL, [q[1](t),q[2](t)], split);
```

$$[\xi = 0, \eta_1 = -q_2, \eta_2 = q_1], \left[ \xi = \frac{3}{2}t, \eta_1 = q_1, \eta_2 = q_2 \right], [\xi = 1, \eta_1 = 0, \eta_2 = 0]$$

*It is worth to mention that this example can not be handled by the algorithms available in **Maple**. Indeed, the **Maple** command **symgen** that looks for a symmetry generator for a given ODE is not able to address more than one dependent variable.*

## 5 Conclusions

We have used the CAS **Maple** to define a new computational procedure that determines, in an automatic way, symmetries of ODEs. The automatic calculation of symmetries is a subject much studied under the theory of differential equations, with many results and applications in many different areas. Our main novelty is the presentation of a new algorithm, alternative to existing ones, which looks to symmetries of ODEs as particular cases of Noether-variational symmetries. As explained in §3, our algorithm involves the resolution of a first order, homogeneous, and linear PDE, which is the abnormal case of the necessary and sufficient condition of invariance for problems of optimal control studied with Noether's theorem [5, 12]. Interesting points of the proposed method are: (i) it is based on a new approach to the subject – in particular, it is different from all the nine alternative algorithms available in **Maple**; (ii) allows us to obtain dynamic symmetries for ODEs of any order; (iii) allows to determine symmetries for systems of ODEs, when the analog **simgen Maple** command of the **DEtools** package can only obtain solutions for a single ODE.

## Acknowledgements

The authors acknowledge the support from the *Portuguese Foundation for Science and Technology* (FCT), through the R&D unit *Centre for Research on Optimization and Control* (CEOC) of the University of Aveiro, cofinanced by the European Community Fund FEDER/POCI 2010. PG is also grateful to the support given by the program PRODEP III/5.3/2003.

## References

- [1] E. S. Cheb-Terrab, L. G. S. Duarte and L. A. C. P. da Mota, Computer algebra solving of second order ODEs using symmetry methods, *Comput. Phys. Comm.* **108** (1998), no. 1, 90–114.
- [2] E. S. Cheb-Terrab and K. von Blow, A computational approach for the analytical solving of partial differential equations, *Comput. Phys. Comm.* **90** (1995), no. 1, 102–116.
- [3] Đ. S. Đukić, Noether's theorem for optimum control systems, *Internat. J. Control* (1) **18** (1973), 667–672.

- [4] P. D. F. Gouveia and D. F. M. Torres, Algebraic computation in the calculus of variations: determining symmetries and conservation laws (in Portuguese), *TEMA Tend. Mat. Apl. Comput.* **6** (2005), no. 1, 81–90.
- [5] P. D. F. Gouveia and D. F. M. Torres, Automatic computation of conservation laws in the calculus of variations and optimal control, *Comput. Methods Appl. Math.* **5** (2005), no. 4, 387–409.
- [6] W. Hereman, Review of symbolic software for the computation of Lie symmetries of differential equations, *Euromath Bull.* **1** (1994), no. 2, 45–82.
- [7] E. Kamke, *Differentialgleichungen. Lösungsmethoden und Lösungen. Teil I: Gewöhnliche Differentialgleichungen. 6. Aufl.; Teil II: Partielle Differentialgleichungen erster Ordnung für eine gesuchte Funktion. 4. Aufl.*, Geest & Portig, Leipzig, 1959.
- [8] P. K. Kythe, P. Puri and M. R. Schäferkotter, *Partial differential equations and Mathematica*, CRC, Boca Raton, FL, 1997.
- [9] J. D. Logan, *Applied mathematics*, Wiley, New York, 1987.
- [10] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze and E. F. Mishchenko, *The mathematical theory of optimal processes*, Translated from the Russian by K. N. Trirogoff; edited by L. W. Neustadt, Interscience Publishers John Wiley & Sons, Inc. New York, 1962.
- [11] A. Samokhin, Full symmetry algebra for ODEs and control systems, *Acta Appl. Math.* **72** (2002), no. 1-2, 87–99.
- [12] D. F. M. Torres, Conservation laws in optimal control, in *Dynamics, bifurcations, and control (Kloster Irsee, 2001)*, 287–296, Lecture Notes in Control and Inform. Sci., 273, Springer, Berlin, 2002.
- [13] D. F. M. Torres, Weak conservation laws for minimizers which are not Pontryagin extremals, Proc. of the 2005 International Conference “Physics and Control” (PhysCon 2005), August 24-26, 2005, Saint Petersburg, Russia. Edited by A.L. Fradkov and A.N. Churilov, 2005 IEEE, pp. 134–138.
- [14] B. van Brunt, *The calculus of variations*, Springer, New York, 2004.
- [15] D. Zwillinger, *Handbook of differential equations*, Academic Press, Boston, MA, 1989.

## Appendix: the new Maple procedure odeSymm

The procedure *odeSymm*, introduced in this paper, has been implemented for the computer algebra system **Maple** (version 10). The complete **Maple** definitions can be freely obtained from <http://www.ipb.pt/~pgouveia/odeSymm.htm> together with an online help database for the **Maple** system.

**odeSymm** computes the infinitesimal generators which define the symmetries of the ODE, or system of ODEs, specified in the input. As explained in section 3, this procedure involves the resolution of a system of partial differential equations. We have used the **Maple** solver *pdsolve*, using, as preferential method, the separation of the variables by sum.

Output:

- one or more lists of symmetry generators for a given ODE, or system of ODEs ( $[\xi = ?, \eta_1 = ?, \eta_2 = ?, \dots, \eta_n = ?], \dots$ ).

Syntax:

- `odeSymm(ode, x(t), opts)`

Input:

- ode - ordinary differential equation, or a set or list of ODEs;
- x(t) - any indeterminate function of one variable, or a list of them, representing the unknowns of the ODE problem;
- opts - (optional) specify options for the **odeSymm** command, where *opts* is one or more of the following:
  - allconst** - When this argument is given, the output presents all the constants given by the **Maple** command *pdsolve*. By default, that is, without option **allconst**, we eliminate redundant constants; this is done by our **Maple** procedure *reduzConst*, which is a technical routine, and thus not provided here. Essentially, the procedure transforms in one constant each sum of constants. The interested reader can find the **Maple** file with its definition at <http://www.ipb.pt/~pgouveia/odeSymm.htm>.
  - mindep** - When one wants to restrict to the minimum the dependencies of the infinitesimal generators:  $\xi(t)$  and  $\eta(\mathbf{x})$ . By default, that is, in the absence of options **mindep** and **alldep**, the following dependencies are considered:  $\xi(t)$  and  $\eta(t, \mathbf{x})$ ;
  - alldep** - All possible dependencies for the infinitesimal generators:  $\xi(t, \mathbf{x}, \psi)$  and  $\eta(t, \mathbf{x}, \psi)$ ;
  - split** - When this argument is given, the procedure invoke the **split** command to divide the resultant set of infinitesimal generators into uncoupled subsets, by fixing the values for all the constants given by the **Maple** command *pdsolve*. The procedure **split** is a technical routine. The interested reader can find the **Maple** file with its definition at <http://www.ipb.pt/~pgouveia/odeSymm.htm>.
  - showdep** - Shows, in the obtained solution, all the dependencies of the generators; otherwise only the name of the generators is shown;

**showt** - Shows, in the obtained solution, the dependence on the time variable (independent variable); otherwise, the time variable is omitted as a function parameter;

**showgen** - Shows, in the obtained solution, besides the infinitesimal generators  $\xi$  and  $\eta$ , the augmented set of variational generators,  $T$ ,  $\mathbf{X}$  and  $\Psi$ ;

**hint=<value>** - Indicate a method of solution of the PDE system (16), where **<value>** is one of ``+``, ``*``, or any other expression allowed by the command `pdsolve`, being also possible to use **hint=nohint** for the case one wants to use the standard method of resolution of Maple; by default, the system is solved by separating the variables by sum (**hint=`+`**).

```
odeSymm := proc(ODEs::{`=`, list(`=`), set(`=`)},
                depvars::{function,list(function)})
local n, tt, xx, pp, k, vX, vPSI, syseqd, sol, lstGerad, valGerad, phi,
      vphi, lpsi, vpsi, Hi, t, Sr, x0, r, aux, mapx, sys, xieta, sol2;
unprotect(Psi); unassign('T'); unassign('X'); unassign('Psi');
unassign('psi');
Hi:=subs(select(type,[args[3..-1]],`=`,hint);
if Hi='hint' then Hi:='+'; fi;
n:=nops(depvars);
if n=1 then x0:=[depvars] else x0:=depvars fi;
t:=op(1,x0[1]);
r:=[]:
for aux in x0 do
  for k from 1 by 1 while evalb(subs(diff(aux,t$k)=_zzz,ODEs)<>ODEs) do
    od;
    r:=[r[], k-1]:
  od:
  Sr:=sum(r['i'], 'i' =1..n);
  mapx:=[seq(x0[i]=_x[1+(sum(r['k'], 'k'=1 ..i-1))], i=1..n)];
  mapx:=[mapx[], seq(seq(diff(x0[i],t$j)=_x[j+1+sum(r['k'], 'k'=1..i-1)],
                                                                    j=1..r[i]-1), i=1..n)];
  mapx:=[mapx[], seq(diff(x0[i],t$r[i])=_xx[i], i=1..n)];
  mapx:=[seq(mapx[nops(mapx)+1-i], i=1..nops(mapx))];
  sys:=subs(mapx,ODEs);
  if n=1 then solve(sys, {_xx[1]})
    else solve({sys[]}, {seq(_xx[i], i=1..n)}) fi;
  phi:=subs(% , [seq(_xx[i], i=1..n)]);
  vphi:=Vector([seq([seq(_x[j], j=2+sum(r['k'], 'k'=1..i-1)..sum(r['k'],
                                                                    'k'=1..i)), phi[i]][], i=1..n)]);

  x0:= [seq(_x[i], i = 1 .. Sr)];
  if Sr>1 then lpsi:=[seq(psi[i], i=1..Sr)] else lpsi:=[psi] fi;
  vpsi:=Vector[row](lpsi);
  if member('alldep',[args[3..-1]]) then
    tt:=t,op(x0),op(lpsi); xx:=tt; pp:=tt;
  elif member('mindep',[args[3..-1]]) then
    tt:=t; xx:=op(x0); pp:=op(lpsi);
```

```

else tt:=t; xx:=t,op(x0); pp:=t,op(lpsi); fi;
if Sr>1 then vX:=Vector([seq(X[i](xx), i=1..Sr)]);
    else vX:=Vector([X(xx)]); fi;
if Sr>1 then vPSI:=Vector[row]([seq(PSI[i](pp), i=1..Sr)]);
    else vPSI:=Vector[row]([PSI(pp)]); fi;
syseqd:={ vpsi.( map(diff,vphi,t)*T(tt)
    +Matrix([seq(map(diff,vphi,i),i=x0)]) .vX
    +vphi*diff(T(tt),t)-map(diff,vX,t) )+vPSI.vphi,
    convert(-vPSI+(vpsi.vphi)*Vector[row]([seq(diff(T(tt),i),i=x0)])
    -vpsi.Matrix([seq(map(diff,vX,i),i=x0)]), 'list')[],
    convert((vpsi.vphi)*Vector[row]([seq(diff(T(tt),i),i=lpsi)])
    -vpsi.Matrix([seq(map(diff,vX,i),i=lpsi)]), 'list')[] } minus {0};
lstGerad:=[T(tt), convert(vX,'list')[], convert(vPSI,'list')[]];
if Hi='nohint' then sol:=pdsolve(syseqd, lstGerad);
else sol:=pdsolve(syseqd, lstGerad, HINT=Hi); fi;
if not member('allconst',[args[3..-1]]) then sol:=reduzConst(sol); fi;
valGerad:=subs(sol,lstGerad);
sol:=[(lstGerad[i]=valGerad[i])$i=1..nops(lstGerad)];
sol:=collect(expand(simplify(sol)), [t,op(x0),op(lpsi)]);
if not member('showdep',[args[3..-1]]) then
    xieta:=[xi,seq(eta[i],i=1..n)];
    sol:=subs(map(i->i=op(0,i),lstGerad),sol);
else xieta:=[xi(tt),seq(eta[i](xx),i=1..n)]; fi;
if n=1 then xieta:=subs(eta[1]=eta,xieta) fi;
sol:=subs('PSI'='Psi', sol);
sol:=subs(map(i->rhs(i)=lhs(i),mapx),sol);
xieta:=subs(map(i->rhs(i)=lhs(i),mapx),xieta);
sol2:=[xieta[1]=rhs(sol[1]),
    seq(xieta[i+1]=rhs(sol[2+sum(r['k'], 'k'=1 ..i-1)]), i=1..n)];
if member('split',[args[3..-1]]) then sol2:=[split(sol2)]
else sol2:=[sol2] fi;
if member('showgen',[args[3..-1]]) then sol:=[sol,sol2[]];
else sol:=sol2 fi;
if n=1 then x0:=op(0,depx) else x0:=map(i->op(0,i),depx)[] fi;
sol:=subs({map(i->i(t)=i,[x0,op(lpsi)][]), sol);
if member('showt',[args[3..-1]]) then
    sol:=subs({map(i->i(t)=i(t),[x0,op(lpsi)][]),sol) fi;
return sol[];
end proc:

```