REDUCE-Procedures for the Study of Adjoint Symmetries of Second-Order Differential Equations

W. SARLET and J. VANDEN BONNE

Instituut voor Theoretische Mechanica, Universiteit Gent,
Krijgslaan 281 – B-9000 Gent (Belgium)

Two REDUCE programs are presented which should be of assistance in computing and studying so-called adjoint symmetries of second-order ordinary differential equations. The first program essentially serves to construct the determining equations for adjoint symmetries, whose leading coefficients are allowed to be polynomial functions of the velocities. The second program runs various tests concerning the possible construction of a first integral or a Lagrangian for the given system.

1. Introduction

The concept and importance of symmetries of differential equations is well known. Their actual construction in applications involves many steps which are fairly straightforward, in principle, but can be quite tedious as well. Not surprisingly therefore, many people in recent years have developed computer algebra packages which have become indispensible for studying complicated systems. For a recent contribution to this library of programs and a review of what is currently available, see Champagne et al (1990).

The knowledge of a symmetry of a differential equation generally helps to reduce its order, for example by providing a corresponding first integral. This is in particular true for the subclass of symmetries of a Lagrangian system which respect the variational character of the system (Noether's theorem).

The notion of an adjoint symmetry is probably much less familiar. Yet, recent investigations have indicated that adjoint symmetries sometimes constitute the more fundamental concept for understanding certain aspects of the theory. For example, restricting ourselves from now on to second-order ordinary differential equations, there is a bijective correspondence between equivalence classes of first integrals and classes of adjoint symmetries. When the system happens to be Lagrangian, there is a natural map between adjoint symmetries and symmetries and we are just looking at Noether's theorem. But the point is that the property remains valid for non-Lagrangian systems, provided we stick to the adjoint symmetries.

The purpose of the present note is to report on REDUCE-procedures which have been developed (at the algebraic level) to assist in the computation of adjoint symmetries. Such a computation, as in the case of symmetries, essentially involves two steps. The first step is a matter of setting up the determining equations. The second and more difficult one concerns solving these equations and requires programming at the symbolic level. It would, however, be a waste of energy to do this for our situation because it would simply duplicate work that has already been done. Indeed, determining equations for adjoint symmetries are of the same nature – overdetermined systems of linear partial differential equations – as those for symmetries. So what one wants is to access the solve-routines of the existing packages referred to above. Our first program accordingly only takes care of step 1, but in doing so allows more flexibility than what is usually offered. To be precise, in most applications, the right-hand sides of the differential equations depend polynomially on the velocity variables and all packages then set out to find the Lie point symmetries, i.e. the symmetries whose leading coefficients do not depend on velocities. We allow the forces to be rational functions of the velocities and let the user make an ansatz about the structure of the leading coefficients of adjoint symmetries which in principle could also be chosen to depend rationally on the v's. Introducing denominators in these leading coefficients, however, quite drastically changes the nature of the problem. In passing, although adjoint symmetries are our main objective, the program likewise offers the option to compute determining equations for symmetries (always, of course, in the context of second-order ordinary differential equations).

The second program is one for running various tests. In connection with the preceding remark, it can be used to test whether a given vector field is a dynamical symmetry of a second-order system, but the main emphasis is on adjoint symmetries. If a candidate for being an adjoint symmetry successfully passes the test, there are two further options. With the first one, it is possible to check whether the adjoint symmetry corresponds to a first integral of the system and if this is the case, the program actually computes the first integral. The second subtest, which covers a larger class of situations, verifies whether the adjoint symmetry determines a Lagrangian for the given system and, if the answer is favourable, again tries to compute it.

Before entering into a more detailed description of these programs and the way to use them, we briefly summarize the theoretical background for the various aspects

referred to above. The next section lists the abstract mathematical formulation of all results in question. The reader who is not familiar with the differential geometric content can largely skip over it and pass to the relevant coordinate formulae in section 3.

2. Adjoint symmetries – theoretical background

A system of (generally time-dependent) second-order ordinary differential equations in normal form is geometrically described by a vector field Γ on the manifold $\mathbb{R}\times TM$, which is canonically equipped with a type (1,1) tensor field S, commonly referred to as the vertical endomorphism. The following definitions and results are taken from Sarlet et al (1990), where more references to the relevant literature can be found.

Consider the subset $\mathcal{X}^\star_\Gamma$ of 1-forms on $\mathbb{R}\times TM$, characterized by

$$\mathcal{X}^\star_\Gamma = \{\phi \in \mathcal{X}^\star(\mathbb{R}\times TM) \mid \mathcal{L}_\Gamma(S(\phi)) = \phi - \langle\Gamma,\phi\rangle dt\}\,,$$

and the associated projection operator

$$\pi_\Gamma : \mathcal{X}^\star(\mathbb{R}\times TM) \longrightarrow \mathcal{X}^\star_\Gamma \quad , \quad \phi \longmapsto \pi_\Gamma(\phi) = \mathcal{L}_\Gamma(S(\phi)) + \langle\Gamma,\phi\rangle dt.$$

Two elements $\alpha_1, \alpha_2 \in \mathcal{X}^\star_\Gamma$ are said to be equivalent if $\alpha_1 \wedge dt = \alpha_2 \wedge dt$.

An adjoint symmetry of a second-order equation field Γ is a 1-form $\alpha \in \mathcal{X}^\star_\Gamma$, whose Lie derivative with respect to Γ again belongs to $\mathcal{X}^\star_\Gamma$.

There exists a bijection between the set of equivalence classes of adjoint symmetries and the set of so-called Γ-basic forms, the latter being 1-forms β on $\mathbb{R}\times TM$ with the properties : $i_\Gamma\beta = 0$ and $i_\Gamma\, d\beta = 0$. This bijection is actually determined by the type (1,1) tensor field $\mathcal{L}_\Gamma S$.

Proposition 1. *Let α be an adjoint symmetry of Γ such that the associated Γ-basic form $\beta = \mathcal{L}_\Gamma S(\alpha)$ is dF for some function F. Then F is a first integral of Γ. Conversely, every first integral F can be obtained in this scheme.*

This simple statement is the one which, as mentioned in the introduction, reduces to Noether's theorem for the case of a Lagrangian system.

Proposition 2. *Let α be an adjoint symmetry of Γ such that $\alpha = \pi_\Gamma(dF)$ for some function F. Then $L = \Gamma(F)$ is a Lagrangian for the given system.*

Although it may look rather different, this result is actually a generalization of the preceding one. Obviously indeed, it might happen that $\Gamma(F) = 0$. More generally, if α matches the requirement for some function F on $\mathbb{R}\times TM$, then $\pi_\Gamma(d(F-\mathtt{g}))$ produces an equivalent adjoint symmetry for any function $\mathtt{g}$ on $\mathbb{R}\times M$ and so it is possible that subtracting such a function $\mathtt{g}$ in fact brings us back to the case of a first integral. We will see that the test-program takes care of this possibility.

For completeness, let us recall that the concept of symmetries we are talking about in this context refers to what are often called *dynamical symmetries* of Γ, i.e. vector fields X on $\mathbb{R} \times TM$ for which $[X, \Gamma] = h\Gamma$ for some function h. Such dynamical symmetries must also be collected into equivalence classes, X_1 and X_2 being equivalent if their difference is a multiple of Γ.

As a final remark, we should mention that there is a version of this theory in which all geometrical objects live on a tangent bundle TM, which means that they do not explicitly depend on time (see Sarlet et al (1987)).

3. Coordinate expressions

Denoting coordinates on $\mathbb{R} \times TM$ by (t, q^i, v^i) (where i runs from 1 to $n = \texttt{DIM}$) and thinking of the system of second-order equations $\ddot{q}^i = \Lambda^i(t, q, \dot{q})$, the corresponding vector field Γ reads

$$\Gamma = \frac{\partial}{\partial t} + v^i \frac{\partial}{\partial q^i} + \Lambda^i(t, q, v) \frac{\partial}{\partial v^i}. \tag{1}$$

A general 1-form $\alpha \in \mathcal{X}^\star_\Gamma$ is of the form

$$\alpha = \alpha_i \, dv^i + \Gamma(\alpha_i) \, dq^i + \tau \, dt, \tag{2}$$

where in view of the adopted equivalence relation we can, without loss of generality, take the function τ to be zero. Our 1-form α is going to be an adjoint symmetry if the leading coefficients α_i are solutions of the linear second-order PDE's

$$\Gamma^2(\alpha_i) + \Gamma\left(\alpha_j \frac{\partial \Lambda^j}{\partial v^i}\right) - \alpha_j \frac{\partial \Lambda^j}{\partial q^i} = 0. \tag{3}$$

The tensor field S acts as a linear map on 1-forms, according to the following rule :

$$S(\alpha) = \left(\frac{\partial}{\partial v^k} \lrcorner \alpha\right)(dq^k - v^k \, dt). \tag{4}$$

The computation of the Γ-basic form $\beta = \mathcal{L}_\Gamma S(\alpha)$, corresponding to an adjoint symmetry α, is then easy to implement in the following way :

$$\beta = \Gamma \lrcorner d(S(\alpha)) + d(\Gamma \lrcorner S(\alpha)) - S(\mathcal{L}_\Gamma \alpha).$$

If we write

$$\beta = b_i(x) \, dx^i,$$

where $2n + 1$ collective coordinates x^i have been introduced in the order (v^j, q^j, t), the situation of proposition 1 is the case where

$$d\beta = 0, \quad \text{or} \quad \partial b_i / \partial x^j - \partial b_j / \partial x^i = 0. \tag{5}$$

Assuming that the coefficients b_i are well-defined in a star-shaped region around the origin, the corresponding first integral can be computed as follows :

$$F = x^i \int_0^1 b_i(sx)\, ds. \tag{6}$$

Testing the more general situation covered by proposition 2 amounts to verifying the conditions

$$\partial\alpha_i/\partial v^j - \partial\alpha_j/\partial v^i = 0. \tag{7}$$

If they hold true and we compute the function

$$F = v^i \int_0^1 \alpha_i(sv, q, t)\, ds, \tag{8}$$

we know that $L = \Gamma(F)$ is potentially a Lagrangian for the given system. The subcase where actually the more restrictive requirements (5) are satisfied corresponds to the case that $\Gamma(F)$ is the total time derivative of a function $g(q,t)$. For this to be true, it is necessary and sufficient that

$$\frac{d}{dt}\left(\frac{\partial\Gamma(F)}{\partial v^i}\right) - \frac{\partial\Gamma(F)}{\partial q^i} \equiv 0. \tag{9}$$

Finally, concerning the study of symmetries, a dynamical symmetry of Γ is a vector field of the form

$$X = \mu^i \frac{\partial}{\partial q^i} + \Gamma(\mu^i)\frac{\partial}{\partial v^i} + \tau\,\Gamma, \tag{10}$$

where the functions μ^i satisfy the second-order PDE's

$$\Gamma^2(\mu^i) - \Gamma(\mu^j)\frac{\partial\Lambda^i}{\partial v^j} - \mu^j\frac{\partial\Lambda^i}{\partial q^j} = 0 \tag{11}$$

and the function τ, as before, is irrelevant in view of the adopted equivalence relation. Note in passing that the terminology *adjoint symmetry* originates from the fact that the equations (3) are precisely the adjoint equations of (11).

4. The programs DETSYS

As mentioned before, one sometimes wants to keep all calculations concerning autonomous differential equations within the time-independent framework. We have accordingly developed separate programs, called AUTODETSYS and TIMEDETSYS, which of course show only minor differences. In this section, we briefly explain the different procedures and operators constituting the program TIMEDETSYS and comment on the way to use the program.

For a start, TIMEDETSYS runs under REDUCE 3.3 (Hearn (1987)) and makes use of the package EXCALC developed by E. Schruefer (1986). It is assumed that EXCALC has been loaded before calling TIMEDETSYS. The structure of the program is as follows :

```
PROCEDURE DATA $
OPERATOR HP $
PROCEDURE CO(P,M) $
PROCEDURE SYSTEM $
PROCEDURE CLEARDETSYS $
WRITE ‘‘Type ‘DATA’.’’ $
```

Having given the command `IN TIMEDETSYS $`, it is clear that you will be invited to call the procedure DATA, which will set up the problem. DATA establishes the duality rules between coordinate vector fields and 1-forms, initiates certain counters and essentially asks you to enter the dimension `DIM` of the system (i.e. the number of second-order ODE's) and the right-hand sides `LMBD(I)` of the equations (which must be rational functions of the velocities). Actually, the screen will first ask whether these data are already present. If the answer is yes, the assumption is that they have been read in from an input file. This can be done at any stage before calling `DATA` and such an input file will typically contain the following lines (the example below relates to the so-called Emden equation) :

```
PFORM Q(J)=0, V(J)=0, LMBD(J)=0 $
DIM := 1 $
LMBD(1) := -2*V(1)/Q(0)-Q(1)**5 $
END $
```

Note here that for the program, the time variable t is called $Q(0)$. The procedure `DATA` finally urges you to type the commands

```
FDOMAIN F = F(Q(0),...,Q(<dim>)),
        H = H(Q(0),...,Q(<dim>)) $
```

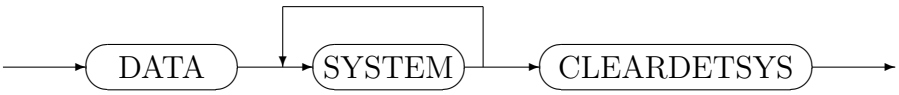
and then to call the procedure SYSTEM. Here, F and H are going to be the unknown functions in the ansatz which will be made later on about the leading coefficients α_i of an adjoint symmetry (or μ^i for a dynamical symmetry). Recall that the α_i or μ^i can, in principle, also be chosen to depend rationally on the velocities; the F's then refer to the numerator and the H's to the denominator. Both are double-indexed variables : the upper index relates to the α_i or μ^i in question, while the lower index is augmented by 1 for each newly encountered independent coefficient in the polynomial expressions. It is a pity that the above `FDOMAIN`-declarations cannot be taken care of automatically as soon as the program knows `DIM`.

The procedure SYSTEM defines the vector field (1) (called G in the program) and first offers the choice between S for symmetries and A for adjoint symmetries. Next, having entered upon request the desired degree (in the velocities) of the numerator and denominator of the leading coefficients, SYSTEM calls the operator HP to create the necessary polynomials. HP takes two arguments and is a recursive operator : $HP(L, D)$ sets up a polynomial of degree D in $L-1$ velocity variables, the

unknown coefficients being called F's or H's, depending on the value of a constant ND, which is 0 for a numerator and 1 for a denominator. Actually, the desired result is obtained by creating recursively a homogeneous polynomial of degree D in L variables, with an auxiliary variable $V(0)$ which is set equal to 1. The resulting format for the leading coefficients α_i or μ^i is displayed on the screen. The next step taken by SYSTEM is the calculation of the left-hand sides of equations (3) or (11). With the switches `GCD` and `EZGCD` on, the numerator of these expressions is calculated and the recursive procedure CO(P,M) is called, which singles out the coefficients of all independent monomials of a polynomial P in M variables $V(I)$. The result is the desired set of determining equations for the unknown F's and H's and is of course sent to the screen. At this stage, one can type SYSTEM again or call instead the procedure CLEARDETSYS. The first option is appropriate if one wants to compute other determining equations (with a different ansatz for the leading coefficients) for a symmetry or adjoint symmetry of the same second-order system.

The procedure CLEARDETSYS does all the cleaning up and allows to type DATA again for starting investigations on another system.

At all stages where an input or selection is expected from the user, the different actions which are acceptable are listed on the screen. Leaving internal options of a procedure apart, the structure of the program is summarized by the simple scheme below:



If one obtains a set of determining equations which does not look too discouraging from the outset, one will of course want to solve them and, as indicated before, the idea is to use existing packages for this purpose. A variety of such packages exists but not all of them may give access to the solve-routines of the program with the same ease. It should be emphasized at this point that all such programs, for obvious reasons, work under the assumption that the PDE's to be solved are linear first-order ones. This feature will remain verified as long as the leading coefficients of the problem at hand are taken to be polynomials of any order in the velocities. When one allows for denominators, however, the ensuing PDE's are going to be nonlinear. One should therefore only venture such a possibility when there is a special reason for it and then quite likely an additional assumption will have to be that the coefficients are polynomial functions of the $Q(I)$. Existing packages for solving the determining equations range from fully automatic ones to very interactive-minded ones and it is perhaps a matter of personal taste to find what suits you best. As an example of the first category of programs, we quote the package LIE, developed by A. Head (1990) and which comes with an interface for entering directly at the level of the

solve-routine. LIE is a muMATH package and, although quite remarkable in its achievements, is of course fairly limited by its environment. A REDUCE-version of LIE is announced by G.E. Prince (private communication). An equally automatic, REDUCE-based package for symmetries is of course F. Schwarz's SPDE (available with the 3.3 version), but we are unaware of a possibility for accessing directly his SIMPSYS-procedure. Among the programs which advocate a more interactive use for solving the kind of linear, overdetermined PDE's arising in the study of symmetries, we mention the one by Kersten (1987), which is accessible in REDUCE.

At this point, we would like to indicate a problem area. Clearly, one would like to supplement our TIMEDETSYS with an extra procedure which writes the determining equations to the disk, for further handling by another computer algebra package. We have not done so, because every potential user might have a different preference in this respect, but whatever this preference is, there will be a lot of ordinary text editing required, because the file created by the `OUT` command in REDUCE always looks rather akward.

5. The TEST-programs

As with the DETSYS-programs, there is an AUTOTEST and a TIMETEST and it will be sufficient to explain how the latter works for time-dependent systems. The program has the following structure :

```
PROCEDURE DATATEST $
PROCEDURE TESTSYM $
PROCEDURE TESTADSYM $
OPERATOR S $
PROCEDURE TESTFINT $
PROCEDURE TESTLAGR $
PROCEDURE CLEARTEST $
WRITE ``Type `DATATEST','' $
```

The second-order system under investigation is set up during execution of DATATEST (interactively or via an input file) and one subsequently types TESTSYM or TESTADSYM. With TESTSYM one is asked to enter the leading coefficients μ^j of a potential dynamical symmetry. These again may be read in before typing TESTSYM from a second input file or could already have been incorporated in the first one. The program tells you whether or not the conditions (11) are satisfied and leaves the option of calling TESTSYM (for testing another symmetry of the same system), TESTADSYM or CLEARTEST.

The beginning of TESTADSYM runs along the same lines : one enters the leading coefficients α_i of a candidate for an adjoint symmetry. Assuming we are dealing, for example, with the Emden equation referred to before, an input file (to be entered before calling TESTADSYM) could contain the following commands :

```
PFORM ALPHA(J) = 0 $
ALPHA(1) := 2*V(1)*Q(0)**3 + Q(1)*Q(0)**2 $
END $
```

One now of course replies with Y to the question whether the leading coefficients are already present and the program will check the conditions (3). Assuming, as is the case with the above example, the test is positive, you are invited to proceed with TESTFINT or TESTLAGR (TESTSYM, TESTADSYM or CLEARTEST are the alternative options).

The procedure TESTFINT will first compute the 1-form β, making use of the operator S, defined as in (4). If the test on the closure of β fails, you can ask to see the non-vanishing expressions in (5). The idea here is that the quantities involved might contain a number of parameters so that it could be interesting to see if the requirements for a first integral can be met for special values of these parameters. With the example at hand, the test on conditions (5) would be positive again and the program would want to find the corresponding first integral F with the aid of the formula (6). Such an integration, however, will not always be successful. For this reason, you are first asked whether you wish to see the determining equations for F, i.e. the equations $\beta_i = \partial F/\partial x^i$. You normally will first reply with N and let the machine have a go. If it fails, you simply type TESTFINT again, reply with Y this time and do the integration by hand. One of the possible causes for failure is the appearance of singularities at the origin in the integrand of (6) (think, for example, of central force problems). A rudimentary test in that respect is incorporated into the procedure and if singularities are detected, you will rightaway get the determining equations for F in return. One might envisage programming the computation of a line integral along a different path in such a case, but REDUCE sometimes appears to have great trouble with the integration of operators (such as our indexed variables) and the process of suppressing temporarily the operator form of such variables seems a bit too cumbersome for the problem at hand.

In our example for the Emden equation, no such problems will occur and the machine will happily tell you that it found the following first integral :

$$\texttt{fint} := \frac{(q^0)^2*(q^0*(q^1)^6+3*q^0*(v^1)^2+3*q^1*v^1)}{3} \ .$$

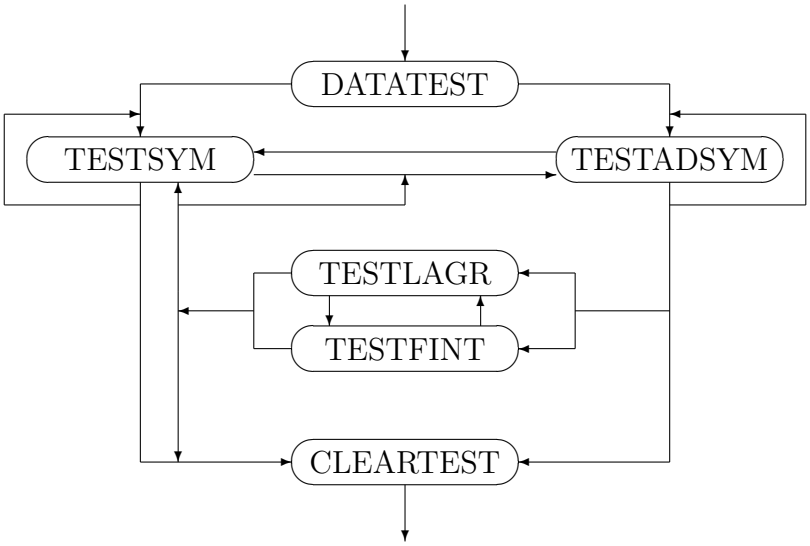
You can now again select TESTLAGR, TESTADSYM, TESTSYM or CLEARTEST.

The procedure TESTLAGR is fairly similar. It will first check the requirements (7) and in the case of a negative test offer the possibility for seeing the non-vanishing conditions. In the event of a positive test, you will again be asked first whether you want to get the determining equations for the function F giving rise to a Lagrangian (i.e. the equations $\alpha_i = \partial F/\partial v^i$). When you say N (as we would always try), an attempt is made to compute F with the formula (8). We know what to do when the integration fails, but in the successful case, the procedure will immediately check the

condition (9) and invite you to call upon TESTFINT if (9) happens to be verified. Needless to say, this is exactly what will happen when TESTLAGR is run on the above adjoint symmetry for the Emden equation. In any event, at the end of this procedure, you can select TESTFINT, TESTADSYM, TESTSYM or CLEARTEST.

Procedure CLEARTEST will do what is necessary to start, if desired, with DATATEST again and investigate a different system.

The formal scheme which summarizes the different ways one can go from one procedure to another this time looks as follows:



For a more complete illustration of the test-program at work, we consider an example which was discussed in Sarlet (1991), in the context of generalized Hénon-Heiles type systems and integrability. In fact we will take one of the cases which were mentioned there merely in the concluding remarks, so that the first integrals which will emerge have not been published before. To keep the presentation within reasonable length, we will abbreviate or omit messages appearing on the screen where possible. The different steps which are being executed can be summarized as follows. After entering the differential equations

$$\begin{aligned} \ddot{q}_1 &= -(3m/5)q_1^2, \\ \ddot{q}_2 &= -2mq_1q_2, \end{aligned} \tag{12}$$

we find a first adjoint symmetry to pass the test. It further satisfies the requirement for producing a "Lagrangian", which turns out, however, to be a total time-derivative and thus, more interestingly, must correspond to a first integral, computed in step 5. Subsequently, a second candidate for an adjoint symmetry is entered, which again gives rise to a first integral for the given system.

```
1:  IN AUTOTEST $
Type DATATEST
2:  DATATEST $
Are DIM and accelerations LMBD(J) already present?  (Y/N)
N $
Enter dimension ≫ 2 $
Enter accelerations
lmbd[1] := -3*(M/5)*Q(1)**2 $
lmbd[2] := -2*M*Q(1)*Q(2) $
Type TESTSYM or TESTADSYM
3:  TESTADSYM $
Are leading coefficients ALPHA(J) present?  (Y/N)
N $
Enter coefficients of possible adjoint symmetry
alpha[1] := -(8/5)*M*Q(2)**3*V(1) + (12/5)*M*Q(1)*Q(2)**2*V(2)
          + V(2)**3 $
alpha[2] := (12/5)*M*Q(1)*Q(2)**2*V(1) + (6/5)*M*Q(1)**2*Q(2)*V(2)
          + 3*V(1)*V(2)**2 $
Yes, you have an adjoint symmetry!
Type TESTFINT, TESTLAGR, TESTADSYM, TESTSYM or CLEARTEST
4:  TESTLAGR $
Positive!  Do you want the determining system?  (Y/N)
N §
TOTAL TIME-DERIVATIVE:
ttd := -  96*(q[1])^2*(q[2])^2*m^2*(q[1]*v^2+q[2]*v[1])
          ------------------------------------------- .
                                25
TESTFINT should give a first integral!
Type TESTFINT, TESTADSYM, TESTSYM or CLEARTEST
5:  TESTFINT $
Positive!  Do you want the determining system?  (Y/N)
N $
FIRST INTEGRAL:
fint := (32*(q[1])^3*(q[2])^3*m^2 + 15*(q[1])^2*q[2]*(v^2)^2*m
        + 60*q[1]*(q[2])^2*v[1]*v^2*m - 20*(q[2])^3*(v[1])^2*m
        + 25*v[1]*(v^2)^3)/25
Type TESTLAGR, TESTADSYM, TESTSYM or CLEARTEST
6:  TESTADSYM $
Are leading coefficients ALPHA(J) present?  (Y/N)
N $
Enter coefficients of possible adjoint symmetry
alpha[1] := 20*M*Q(1)**2*Q(2)**2*V(1) - 14*M*Q(1)**3*Q(2)*V(2)
          + 30*Q(2)*V(1)**2*V(2) - 20*Q(1)*V(1)*V(2)**2 $
alpha[2] := -14*M*Q(1)**3*Q(2)*V(1) + M*Q(1)**4*V(2)
          + 10*Q(2)*V(1)**3 - 20*Q(1)*V(1)**2*V(2) $
```

```
Yes hou have an adjoint symmetry!
Type TESTFINT, TESTLAGR, TESTADSYM, TESTSYM or CLEARTEST
7:  TESTFINT $
Positive!  Do you want the determining system?  (Y/N)
N $
FIRST INTEGRAL:
fint := -(32*(q^1)^5*(q^2)^2*m^2 - 5*(q^1)^4*(v^2)^2*m
         + 140*(q^1)^3*q^2*v^1*v^2*m - 100*(q^1)^2*(q^2)^2*(v^1)^2*m
         + 100*q^1*(v^1)^2*(v^2)^2 - 100*q^2*(v^1)^3*v^2)/10
Type TESTLAGR, TESTADSYM, TESTSYM or CLEARTEST
8:  CLEARTEST $
Type DATATEST for running tests on another system
```

A few words are in order now, concerning the investigations which precede the above tests. The adjoint symmetries we tested are of degree 3 in the velocities. It is impossible to exhibit here how AUTODETSYS produces the determining equations, so we will limit ourselves to some indications about the time it takes. On Acorn Archimedes 440 (ARM not upgraded, 4 Mbyte RAM), obtaining the 42 determining equations for degree 3 takes not more than 1 minute. Always in the context of equations (12), the 72 determining equations for adjoint symmetries of degree 5 will be displayed in about 4 minutes. If one would fancy degree 15, which produces 272 PDE's, one can easily get some correspondence in the out tray, as Archimedes will be happy for about 75 minutes. One should never attempt denominators of degree greater than 1. For example, with the numerator of degree 2 and the denominator of degree 1, although the number of determining equations here is "limited" to 64, you will only get to see them after almost 2 hours. Their non-linearity, as indicated before, will not have an inspiring effect on the activities of the rest of the day.

The source file for the programs DETSYS and TEST is available upon simple request. It is our intention to let these programs keep track of future developments in the theory about adjoint symmetries.

Acknowledgements

This work was supported by a research grant of the Belgian National Fund for Scientific Research. We further are indebted to the CAGe group for facilitating the use of computer algebra.

References

Champagne B., Hereman W. and Winternitz P. (1990), *The computer calculation of Lie point symmetries of large systems of differential equations*, preprint.

Head A.K. (1990), *LIE : a muMATH program for the calculation of the LIE algebra of differential equations — User's manual* (CSIRO Division of Material Sciences, Clayton, Australia).

Hearn A.C. (1987), *REDUCE User's Manual, Version 3.3.* The Rand Corporation, Santa Monica.

Kersten P.H.M. (1987), *Infinitesimal symmetries : a computational approach*, CWI Tract **34** (Amsterdam).

Sarlet W., Cantrijn F. and Crampin M. (1987), J. Phys. A : Math. Gen. **20**, 1365–76.

Sarlet W., Prince G.E. and Crampin M. (1990), J. Phys. A : Math. Gen. **23**, 1335–47.

Sarlet W. (1991), J. Phys. A: Math. Gen. **24**, 5245–51.

Schruefer E. (1986), *EXCALC — User's Manual* (distributed with REDUCE 3.3).