Computer Physics Communications 66 (1991) 319–340
North-Holland

The computer calculation of Lie point symmetries of large systems of differential equations

B. Champagne [a,1], W. Hereman [b,2] and P. Winternitz [a,3]

[a] *Centre de Recherches Mathématiques, Université de Montréal, C.P. 6128, Succursale A, Montréal, Québec, Canada H3C 3J7*
[b] *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401, USA*

Received 13 September 1990; in revised form 28 January 1991

A MACSYMA program is presented that greatly helps in the calculation of Lie symmetry groups of large systems of differential equations.

The program calculates the determining equations for systems of m differential equations of order k, with p independent and q dependent variables, where m, k, p and q are arbitrary positive integers.

The program automatically produces a list of determining equations for the coefficients of the vector field. This list has been parsed so that it is free of duplicate equations and trivial differential redundancies. Numerical factors and non-zero parameters occurring as factors are also removed. From the solution of these determining equations one can construct the Lie symmetry group.

An example shows the use of the program in batch mode. It also illustrates a feedback mechanism, that not only allows the treatment of a large number of complicated partial differential equations but also aids in solving the determining equations step by step.

PROGRAM SUMMARY

Title of program: SYMMGRP.MAX

Catalogue number: ACBI

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

Licensing provisions: none

Computer for which the program is designed and others on which it has been tested:
Computers: (i) VAX 11/750; (ii) VAX 11/780; (iii) VAX 8600;
Installations: (i) Centre de Calcul, Université de Montréal, Montréal, Québec, Canada H3C 3J7; (ii) Center for the Mathematical Sciences, University of Wisconsin-Madison, Madison, WI 53705, USA; (iii) Computer Center, Colorado School of Mines, Golden, CO 80401, USA

Operating system under which the progam has been tested: VMS 3.7 or higher

Programming language used: REX MACSYMA 412, compatible with MACSYMA 305 and 309

Memory required to execute with typical data: problem dependent, typical working set size about 0.5 Mbytes

No. of bits in a word: 32

No. of processors used: 1

Has the code been vectorised? no

No. of lines in distributed programs including documentation file, test data, etc.: 5155

Keywords: symmetry group, differential equations, Lie algebras, symbolic computation, MACSYMA

[1] Present address: INRS-Télécommunications, Université du Québec, 3 Place du Commerce, Verdun, Québec, Canada H3E 1H6.

[2] This work was partly supported by AFOSR under Grant No. 85-NM-0263.

[3] Ce rapport a été publié grâce à une subvention du fonds FCAR pour l'aide et le soutien à la recherche.

Nature of physical problem
The symmetry group of a given system of differential equations modeling a physical phenomenon may be used to achieve several goals. These include the classification of the solutions of the system, the generation of new solutions from known ones, the simplification of the system by the method of symmetry reduction, to name a few. In the case where particular symmetries must be present, the symmetry group can be used to determine the validity of the modeling differential equations.

Method of solution
The construction of the symmetry groups of differential equations is based on an adaptation of the notation, the terminology and the method described in ref. [1]. This procedure is translated into a MACSYMA [2,3] program that performs the most elaborate part of the job, namely the construction of a complete list of determining equations which is free of redundant factors, repetitions and trivial differential consequences.

Restrictions on the complexity of the problem
For complicated systems of differential equations involving derivatives of high order, time limits and available computer memory may cause restrictions. Further limitations are discussed in section 3.5 of the Long Write-Up.

Typical running time
Given a system of m differential equations of order k with q unknowns and p independent variables, running time is an increasing function of m, k, q and p. Typical running times (CPU) for an example are given in section 4.

Unusual features of the program
The flexibility of this program and the possibility of using it in a partly interactive mode, allow one to find the symmetry groups of essentially arbitrarily large systems of equations. This is the main justification for presenting a new symbolic manipulation program in a field where several programs already exist. Furthermore, this program has been in use (at the Université de Montréal and elsewhere) for over five years, it has been tested on hundreds of systems of equations and has thus been comprehensively debugged.

References
[1] P.J. Olver, Applications of Lie Groups to Differential Equations (Springer, New York, 1988).
[2] MACSYMA Reference Manual, Version 13, Computer Aided Mathematics Group (Symbolics, Burlington, MA, 1989).
[3] MACSYMA User's Guide, Computer Aided Mathematics Group (Symbolics, Burlington, MA, 1988).

LONG WRITE-UP

1. Introduction

1.1. The problem

For the purpose of this article the "symmetry group of a system of differential equations" is the largest local Lie group of local point transformations, acting on the independent and dependent variables of the equations and leaving the solution set of the system invariant. The symmetry group thus transforms solutions of the system amongst each other. A large body of old and new literature exists on this topic; here we just refer to some recent books and reviews [1,4–19]. We also recommend the special issue of Acta Applicandae Mathematicae on "Symmetries of Partial Differential Equations" [20]. A major obstacle in the application of Lie group theory to solving differential equations is that usually a large number of tedious calculations is involved. The purpose of this article is to present and make available a MACSYMA program for the computer assisted calculation of symmetry groups.

The setting is an entirely general one and the method is well known and described e.g. in ref. [1].

We consider a system of m differential equations

$$\Delta^i(x, u^{(k)}) = 0, \quad i = 1, 2, \ldots, m, \tag{1}$$

of order k, with p independent and q dependent real variables, denoted by

$$x = (x_1, x_2, \ldots, x_p) \in \mathscr{R}^p, \tag{2}$$

$$u = (u^1, u^2, \ldots, u^q) \in \mathscr{R}^q. \tag{3}$$

We stress that m, k, p and q are *arbitrary* positive integers. The group transformations have the form

$$\tilde{x} = \Lambda_g(x, u), \qquad \tilde{u} = \Omega_g(x, u), \tag{4}$$

where the functions Λ_g and Ω_g are to be determined. Note that the subscript g refers to the group parameters. The approach is an infinitesimal one; instead of looking for a Lie group G, we look for its Lie algebra $\mathscr{L}$, realized by vector fields of the form

$$\alpha = \sum_{i=1}^{p} \eta^i(x, u)\frac{\partial}{\partial x_i} + \sum_{l=1}^{q} \varphi_l(x, u)\frac{\partial}{\partial u^l}. \tag{5}$$

The procedure [1] for finding the coefficients $\eta^i(x, u)$ and $\varphi_l(x, u)$ is described below. In essence, the computer constructs the kth prolongation $\mathrm{pr}^{(k)}\alpha$ of the vector field α, applies it to the system of equations (1) and requests that the resulting expression vanishes on the solution set of (1),

$$\mathrm{pr}^{(k)}\alpha\Delta^i|_{\Delta^j=0}, \quad i, j = 1, \ldots, m. \tag{6}$$

The result of implementing (6) is a system of linear homogeneous PDEs for η^i and φ_l, in which x and u are independent variables. These are the so-called *determining equations* for the symmetries of the system.

The procedure thus consists of two major steps: *deriving* the determining equations and *solving* them.

1.2. Review of symbolic programs

Several computer packages [15,21–47] exist for this purpose, and some other programs were written for specific examples [48].

The well-documented REDUCE program developed by F. Schwarz [15,21–25], is definitely going the furthest in solving the determining equations with minimal intervention by the user. This program, called SPDE, is distributed with version 3.3 of REDUCE for various types of computers, ranging from PCs to CRAYs. Schwarz also rewrote SPDE [15,25] for use with SCRATCHPAD II, a symbolic manipulation program developed by IBM.

Based on Cartan's exterior calculus, Edelen [26] and Gragert and Kersten [27] did some pioneering work in using REDUCE to calculate the classical Lie symmetries of differential equations. Kersten [28,29] later developed a REDUCE software package for the calculation of the Lie algebra of infinitesimal symmetries (and corresponding Lie–Bäcklund transformations) of an exterior differential system. Eliseev et al. [30] wrote a REDUCE program to generate (but not solve) the system of determining equations for point and contact symmetries. Fedorova and Kornyak [31] generalized the algorithm to include the case of Lie–Bäcklund symmetries.

Apart from packages in REDUCE, we should mention the FORMAC programs by Fedorova and Kornyak [32] and Fushchich and Kornyak [33] that create the system of determining equations for the Lie–Bäcklund symmetries and solve these equations as far as possible. The FORMAC package CRACKSTAR developed by Wolf [34] also allows investigation of Lie symmetries of PDEs, besides dealing with dynamical symmetries of ODEs and the like.

The program LIE by Head [35] is based on version 4.12 of muMATH, running on IBM compatible PCs. Head's program calculates and solves the determining equations automatically. Interventions by the user are sometimes needed and therefore are made possible.

The SYMCON package written by Vafeades [36] also uses muMATH to calculate the determining equations (without solving them). Furthermore, the program verifies whether the symmetry group is of variational or divergence type and computes the conservation laws associated with the symmetries.

Unfortunately, these programs are confined to the 256 K memory accessible by muMATH and can therefore presently not handle very large systems of equations. This limitation motivated Vafaedes to rewrite his SYMCON program in MACSYMA syntax [37]. Although this program is similar in mission to

ours, Vafeades' program requires quite a bit more interaction by the user. Geoff Prince and James Sherring from LaTrobe University (Melbourne, Australia) are working at a "translation" of the source code of LIE into REDUCE.

The calculation of the Lie group by computer was also proposed by Popov, who used the program SOPHUS for the calculation of conservation laws of evolution equations [38].

The package DELiA by Bocharov [39] also runs on PC and claims to perform various tasks based on Lie's approach, such as the computation of point symmetries, conserved currents and conservation laws; simplification and partial integration of overdetermined systems of differential equations, etc. The marketing material that comes with the demonstration disk for DELiA does not specify any underlying symbolic manipulation package. We believe that the program is written in PASCAL. In ref. [40] Bocharov and Bronstein present SCoLAr, a package based on standard PASCAL, for finding infinitesimal symmetries and conservation laws of arbitrary systems of differential equations.

To the best of our knowledge, no package is available yet for the calculation of Lie symmetries with MAPLE and MATHEMATICA.

For completeness, we mention the pioneering work by C. Wulfman and his master students Davison and Nagao [41,42]. Already in the early seventies, Davison [41] developed computer algorithms in SNOBOL, a now obsolete computer language, that could handle symbolic manipulations with differential operators. In 1980, Nagao [42] wrote the computer program DETERMININGEQS (in PASCAL) that could approximate Lie generators for dynamical systems.

Last but not least, we discuss the programs written in MACSYMA, the symbolic package our symmetry program is based upon. Just as REDUCE, MACSYMA is currently available for various types of computers, ranging from PCs to various work stations and main-frame computers (such as VAX) and it is used all over the world.

Apart from an earlier version of our program [47] and the work done by Rosencrans [48], their are only three other MACSYMA-based symmetry programs. The MACSYMA version of SYMCON by Vafeades [37] was discussed above. Schwarzmeier and Rosenau [43,44] made a program that calculates the determining equations in their simplest form, but does not solve them automatically.

The program SYM_DE by Steinberg [45,46] was recently added to the out-of-core library of MACSYMA. The program solves some (or all) of the determining equations automatically and, if needed, the user can (interactively) add extra information. Currently, Steinberg is working at the extension of his program so that it would include the calculation of generalized (i.e. derivative dependent) symmetries.

1.3. The program SYMMGRP.MAX

The present program, called SYMMGRP.MAX is a modification of a package [47] that has been extensively used over the last five years at the University of Montréal and elsewhere. It has been tested on hundreds of systems of equations and has thus been solidly debugged. The flexibility of this program and the possibility of using it in a partly interactive mode, allow to find the symmetry group of in principal arbitrarily large and complicated systems of equations on relatively small computers. There are the main justifications for presenting yet another new symbolic program in a field where several programs already exist.

The amount of interaction by the user will depend on the complexity of the system of differential equations and on the capacity of the computer used. Our experience is that for systems of equations the most time consuming part of the calculation (when done by hand) is the derivation of the determining equations and the elimination of redundant equations from the system.

The actual solving of the determining equations can usually be done by inspection, using elementary results from the theory of linear PDEs. Solving them on a computer may be time consuming, since the simplest approach varies greatly from case to case. Furthermore, a computer program may accidentally not catch the most general result and therefore may return an incomplete symmetry group. The authors are

very aware of this problem which occurred in testing some of the other existing programs! Fortunately, as soon as the new programs by Schwarz [49] and Reid [50–52], both for the determination of the size of a symmetry group, become available, this problem will be easily detectable.

Let us briefly digress on this topic. Indeed, Schwarz and Reid independently developed algorithms to determine the size of a symmetry group of Lie (point) symmetries. Schwarz's program in REDUCE [49] calculates the number of parameters if the group is finite and the number of unspecified functions and its arguments if the group is infinite.

Recently, Reid [50–52] took up the same task. His program SYMCAL [51], written originally in MACSYMA and currently being converted into MAPLE, computes the dimension and the structure constants of the Lie symmetry algebra of any system of PDEs. An extension of the algorithm [52] also allows to classify differential equations (with variable coefficients) according to the structure of their symmetry groups. Furthermore, the approach advocated by Reid applies to the determination of symmetries of Lie, contact, and Lie–Bäcklund type as well as potential symmetries.

In the interest of versatility and simplicity, our present MACSYMA program concentrates on deriving the determining equations. It does not solve them neither does it calculate the size of the symmetry group. Nevertheless, we believe that our program has some distinguished features and advantages:

(1) The equations (6) can be treated simultaneously, i.e. $\mathrm{pr}^{(k)}\alpha$ can be applied to all m equations in the system. The output is then a system of determining equations that is partly solved. This means that the program takes all "first-order equations with one term" and their differential consequences and uses them to simplify the remaining determining system. This greatly decreases the number of equations to be solved manually.

(2) If the computer can be expected to run out of space when applying the prolongation to the system (1), it is possible to apply $\mathrm{pr}^{(k)}\alpha$ to a subset of equations, for instance just to one equation, say $\Delta^1 = 0$. When implementing the requirement $\mathrm{pr}^{(k)}\alpha\Delta^1 \mid_{\Delta^j=0} = 0$, $(j = 1, \ldots, m)$, of course the program takes into account the entire system (1), not just the equation $\Delta^1 = 0$ itself.

(3) If the individual equations in the system (1) are so complex that the computer still runs out of space, it is possible to derive only a subset of the determining equations, e.g. those that occur as coefficients of the highest derivatives in (6). These are usually single term equations.

(4) A feedback mechanism has been incorporated. Once some of the determining equations have been solved, the information obtained about the coefficients η^i and φ_j can be submitted to the computer, which will present a new and simplified system of determining equations. The new information usually includes that the η's and φ's are independent of some variables or depend linearly on some of the other variables. This greatly simplifies further calculations and, after several runs, makes it possible to apply $\mathrm{pr}^{(k)}\alpha$ to the entire system (1), even for very complicated ones. The feedback mechanism can be used all the way to the end. At the last stage, when the completely determined forms of the η's and φ's are submitted, the program will print out that the number of determining equations is zero, i.e. the solution is verified. Hence, the program also allows to verify any solution previously calculated by hand or by means of other programs.

To summarize, whenever $\mathrm{pr}^{(k)}\alpha$ can be applied successfully to the system (1), or a subset thereof, it produces a complete list of determining equations. This list is free of trivial factors, duplication and differential redundancies. If, however, due to memory, time or space limitations, a complete list of determining equations cannot be obtained, it is still possible to derive a subset of the determining equations. In this case, heuristics are used to extract relevant information from that subset. This information is then fed into the program which resumes its calculations.

In this respect, the philosophy of the approach implemented in the present program is to follow the path that would be taken in a manual calculation. That is, obtain in as simple a manner as possible all one term equations, solve them and feed the information back to the computer. This can be done by first

treating just one equation of the given system and usually by extracting only the coefficients of the highest derivatives. All the necessary substitutions and simplifications leading to the new determining system, which are error-prone if done by hand, are carried out automatically and correctly by this program.

2. Procedures for computing the determining equations

We closely follow the notations, the terminology and the method for symmetry analysis used in ref. [1] which are well adapted to computer programming. Recall that the independent and dependent variables for the system (1) are denoted by (2) and (3), respectively. The partial derivatives of u^l are represented using a multi-index notation: for $J=(j_1, j_2, \ldots, j_p) \in \mathbb{N}^p$, we put

$$u^l_J \equiv \frac{\partial^{|J|} u^l}{\partial x_1^{j_1} \partial x_2^{j_2} \cdots \partial_{x_p}^{j_p}}, \tag{7}$$

where $|J| = j_1 + j_2 + \cdots + j_p$. Finally, $u^{(k)}$ will denote a vector whose components are all the partial derivatives of order 0 up to k of all the u^l.

Using these notations the procedure for obtaining the determining equations involves the following five steps:

(1) Construct the kth prolongation of the vector field α in eq. (5) by means of the formula

$$\mathrm{pr}^{(k)}\alpha = \alpha + \sum_{l=1}^{q} \sum_{J} \psi_l^J(x, u^{(k)}) \frac{\partial}{\partial u^l_J}, \quad 1 \le |J| \le k, \tag{8}$$

where the coefficients ψ_l^J are defined as follows. The coefficients of the first prolongation are

$$\psi_l^J = \mathrm{D}_i \varphi_l(x, u) - \sum_{j=1}^{p} u^l_{J_j} \mathrm{D}_i \eta^j(x, u), \tag{9}$$

where J_i is a p-tuple with 1 on the ith position and zeros elsewhere, and D_i is the total derivative operator

$$\mathrm{D}_i = \frac{\partial}{\partial x_i} + \sum_{l=1}^{q} \sum_{J} u^l_{J+J_i} \frac{\partial}{\partial u^l_J}, \quad 0 \le |J| \le k. \tag{10}$$

The higher-order prolongations are defined recursively as

$$\psi_l^{J+J_i} = \mathrm{D}_i \psi_l^J - \sum_{j=1}^{p} u^l_{J+J_j} \mathrm{D}_i \eta^j(x, u), \quad |J| \ge 1. \tag{11}$$

(2) Apply the prolonged operator $\mathrm{pr}^{(k)}\alpha$ to each equation $\Delta^i(x, u^{(k)})$ and require that

$$\mathrm{pr}^{(k)}\alpha \Delta^i |_{\Delta^j = 0} = 0, \quad i, j = 1, \ldots, m. \tag{12}$$

The meaning of condition (12) is that $\mathrm{pr}^{(k)}\alpha$ vanishes on the solution set of the system (1). Precisely this condition assures that α is an infinitesimal symmetry generator of the transformation (4), i.e. that $u(x)$ is a solution of (1) whenever $\tilde{u}(\tilde{x})$ is one.

(3) Choose m components of the vector $u^{(k)}$, say $v^1, \ldots, v^m$, such that:

(a) Each v^i is equal to a derivative of a u^l $(l = 1, \ldots, q)$ with respect to at least one variable x_i $(i = 1, \ldots, p)$.

(b) None of the v^i is the derivative of another one in the set.

(c) The system (1) can be solved algebraically for the v^i in terms of the remaining components of $u^{(k)}$, which we denoted by w. Hence,

$$v^i = S^i(x, w), \quad i = 1, \ldots, m. \tag{13}$$

(d) The derivatives of v^i,

$$v^i_J = \mathrm{D}_J S^i(x, w), \tag{14}$$

where $\mathrm{D}_J \equiv \mathrm{D}_1^{j_1}\mathrm{D}_2^{j_2} \cdots \mathrm{D}_p^{j_p}$, can all be expressed in terms of the components of w and their derivatives, without ever reintroducing the v^i or their derivatives.

While the above procedure sounds complicated, for all specific systems that have been considered the choice of the appropriate v^i has been quite simple. For instance, for a system of evolution equations

$$u^i_t(x_1, \ldots, x_{p-1}, t) = F^i\big(x_1, \ldots, x_{p-1}, t, u^{(k)}\big), \quad i = 1, \ldots, m, \tag{15}$$

where $u^{(k)}$ involves derivatives with respect to the variables x_i but not t, the appropriate choice is clearly $v^i = u^i_t$.

(4) Use (13) to eliminate all v^i and their derivatives from the expression (12), so that all the remaining variables are now independent of each other.

(5) Obtain the determining equations for $\eta^i(x, u)$ and $\varphi_i(x, u)$ by equating to zero the coefficients of all functionally independent expressions in the remaining derivatives u^i_J.

The described procedure is well defined as long as the variables v^i in eq. (13) exist. Furthermore, the length and complexity of the calculations increase rapidly as p, q, m and especially k increase.

3. Description of the program

We now present the MACSYMA program SYMMGRP.MAX that realizes the procedure in section 2 and that provides a set of determining equations for the Lie symmetries of an arbitrary system of differential equations.

For this program we used MACSYMA release 412.61, which is usually implemented on VAX computers operating under VMS. Note however that the program contains nothing beyond standard MACSYMA statements and it is therefore compatible with earlier versions of MACSYMA, e.g. REX MACSYMA 305 and MACSYMA 309.6 (running under UNIX). The user is supposed to have minimal experience with MACSYMA. Information about the syntax of MACSYMA and many examples of its use may be found in refs. [2,3].

The program SYMMGRP.MAX consists essentially of function definitions. In fact, an appropriate function is defined for each major task in the process. As we will see later, these functions may be used individually provided that care is taken with respect to their arguments.

The primary function is called SYMMETRY and it may be considered as the main program. Once called, it reads the data, sets up the environment for the calculation and then goes through the major steps in the calculation by sequentially calling the other functions.

In addition to function definitions, there is a set of statements at the top of the program that serves many purposes throughout the execution. This set contains replacement RULE definitions, MATCHDECLARE statements and PATTERN MATCHING definitions.

3.1. Description of the principal functions

PROVF(F): Applies the kth prolongation of α in eq. (5) to a function $F(x, u^{(k)})$ and outputs the result.

TOTDF(I, F): Applies the operator of total derivative D_i defined in (10) to a function $F(x, u^{(k)})$ and outputs the result.

FPSI(L, J): Calculates the coefficients ψ_l^J in eq. (8) through the use of the recursive formulae (9) and (11). Actually, these coefficients are stored in an array PSI[L,J] defined by PSI[L,J] := FPSI(L, J), so that one may call PSI instead of FPSI. The reason for this is that we can clear the array PSI without clearing the function definition used to calculate the ψ_l^J. Note that according to the notation of section 2, J must be a MACSYMA list of p integers.

EXTSUBST(EXPR): Applies to an expression EXPR depending on x and $u^{(k)}$ the substitutions (13) and (14) until all the v^i and their partial derivatives have been eliminated from EXPR. It returns the new expression so obtained. The function EXTSUBST may be used separately but the information concerning the basic substitution (13) must be given. This is done by rewriting (13) in the form

$$u^{l_i}_{j_i} = S^i(x, w), \quad i = 1, \ldots, m, \tag{16}$$

and according to this, defining the following arrays before using EXTSUBST:

$$\left.\begin{aligned} &\mathrm{INDJ}[i] : J^i \\ &\mathrm{INDL}[i] : l^i \\ &\mathrm{SUB}[i] : S^i(x, w) \end{aligned}\right\} i = 1, \ldots, m. \tag{17}$$

SEARCHOEFF(EXPR): Given an expression EXPR which is a polynomial in the derivatives of u_J^I (exponents of u_J^I need not be integers but instead may be any real numbers), this function finds all the coefficients of the various partial derivatives of u and puts these coefficients in one of two lists according to their length. More explicitly, at the end of the procedure, LODE[1] will be a list containing all the coefficients which are monomials, and LODE[2] will be a list containing all the coefficients which are polynomials (containing "+" as the main operator). Note that in the present context, these coefficients will be precisely the determining equations.

SIMPEQN(): Given the two lists of determining equations in η^i and φ_I, LODE[1] and LODE[2], this function produces a unique list of determining equations called LODE, equivalent to the union of the first two lists but free of repetition and differential redundancy. More precisely,

1. equations of LODE will be ordered by increasing length relative to the operator "+" (beginning with the monomials and ending with the longest polynomials);
2. monomial equations of LODE will be ordered by increasing order of differentiation with respect to x_i and u^I;
3. the list of equations LODE will be free of repetition and trivial differential redundancy;
4. any common factors, such as x_i, u^I, their products and powers thereof, occurring in the equations of LODE will be factored out and canceled. The ELIMINATOR will also cancel all non-zero parameters (their products and powers) given explicitly in the data file (see further). There will be a message for these cancellations provided the parameter warnings is set to true. As a precaution, at the end of the simplifications the user is provided with a list of all the (non-trivial) factors that have been canceled during the execution of SIMPEQN. Special warning messages are given if division by parameters occurs. Note that the program will not clear sums or differences of parameters, derivatives of functions, and the like. This enables the user to determine how the symmetry group is affected by various choices of free parameters and functions.

PRINTEQN(LIST): This function prints the elements of a LIST in an equation like form (see Test Run Output) at the end of this paper.

SYMMETRY(IND1, IND2, IND3): This function stands for the main program. A call to it initiates the computation while the three arguments enable the user to partially control the execution. These

Table 1
Description of the arguments of SYMMETRY

Parameter	Value	Effect on the execution
IND1	0	The program is used in interactive mode
	1	The program is used in batch mode
IND2	0	The array PSI is cleared before the computation starts
	1	PSI is not cleared, only new PSI will be computed
IND3	0	No trace of the calculations will be given
	1	A trace of the calculations will be given

arguments may take the values 0 and 1 and according to their values, different actions are taken by the program as shown in table 1.

3.2. List of principal identifiers

The correspondence between the identifiers used in the program and the mathematical symbols introduced in section 1 and 2 appears in table 2.

3.3. Input data

Every data file must have the following information:

1. The number of independent variables: p (positive integer).
2. The number of dependent variables: q (positive integer).
3. The number of equations in the complete system: m (positive integer).
4. The list of non-zero parameters (occurring in the given equations) that may be factored out and subsequently canceled; parameters: [a1, s1, aa,..., w2]. If there are no such parameters then parameters: [ ]. A discussion of names for parameters that may not be used is given in section 3.5.
5. The number of equations the program has to treat, controlled by sublisteqs. For example, sublisteqs: [e1] for the first equation; sublisteqs: [ei,..., em] for the ith through the mth equations only; sublisteqs: [all] for all the equations in the system.

Table 2
List of the principal identifiers

Text	Program	Meaning
p	P	number of independent variables
q	Q	number of dependent variables
m	M	number of equations in the system
x_i	X[I]	independent variables ($I \in \{1, 2, \ldots, P\}$)
u^l	U[L]	dependent variables ($L \in \{1, 2, \ldots, Q\}$)
J_i	J[I]	canonical vectors, e.g., J[2] = [0,1,0,...,0]
$(j_1, j_2, \ldots, j_p)$	[J[1],J[2],...,J[p]]	list of canonical vectors
u^l_J	U[L,J]	derivatives of u^l
η^l	ETA[I]	coefficient of $\partial/\partial x_i$ in the vector field
φ_l	PHI[L]	coefficient of $\partial/\partial u^l$ in the vector field
ψ^J_l	PSI[L,J]	derivatives of ψ_l
Δ^l	EI	equations in the system, E1, E2, etc.
v^l	VI	variables for the substitution, V1, V2, etc.

6. The desired subset of the determining equations is controlled by setting highest_derivatives: s, where s stands for a positive integer indicating a count-down for the orders of derivatives in the prolongation. For example, 1 refers to information from the highest derivatives only, 2 for highest and second highest derivatives, etc., *all* will result in all the determining equations since all the terms in the prolongation were considered.
7. The flag warnings: true or false, controls the printout of messages about division by parameters and other simplifications.
8. The flag info_given: false or true, is used in connection with the feedback mechanism for solving the determining equations. Information about the coefficients η^i and the φ_l in the vector field α must be entered in a specific way as discussed and exemplified in section 4.
9. The real equations Δ^i in (1) must be given as ei: ..., with $i = 1, \ldots, m$. You only have to put in the left hand side of the equation, leaving out " = 0". For complex equations see section 3.5.
10. The variables v^i in eq. (13), chosen for the substitution are given as vi: ..., with $i = 1, \ldots, m$.

Whether the program is used in batch mode or interactively, the data must be entered in an appropriate way:

(1) Batch mode

This mode is definitely recommended for equations of fairly high order or for systems of equations (see section 4). A simple batch file contains the MACSYMA commands necessary to run the program and to read the data (most often given in a separate file). The data are put into a file with the use of MACSYMA assignment statements. This file is read before the function SYMMETRY is called (with IND1 set to 1).

(2) Interactive mode

This mode is invoked by calling SYMMETRY with IND1 set to zero. It is only useful for single equations or fairly simple systems, e.g. for equations of rather low order of derivation. All parameters can be controlled interactively. The program will prompt for the ten items listed above under "input data".

If info_given is set to true, the program will prompt the user to put in the explicit forms of all the η's and the φ's. If for some of these functions no explicit information is available yet, then one simply submits their names, e.g. eta1, phi1, etc. The array notation is no longer allowed when info is given, one has to use the "concatenated" notation! Information about dependencies must be given with a DEPENDS statement before calling the function SYMMETRY, otherwise dependencies have to be stated explicitly as arguments of the functions.

3.4. Output data

At the end of the computation the determining equations are not automatically printed but instead they are stored in a list of determining equations called LODE. A printout of the determining equations can be obtained with the command PRINTEQN(LODE).

For convenience, the function PRINTEQN is provided separately under the name PRINTEQN.MAX. If one wishes to use any other of the special functions described in section 3.2 separately, then SYMMETRY must be called first with input data consisting only of the p, q and with 0 assigned to m.

Table 3 summarizes the information available at the end of the computation and how to access it.

3.5. Type of systems

As indicated previously, the program can be applied to systems of m differential equations of order k, with p independent and q dependent variables, where m, k, p and q are arbitrary positive integers. Besides limitations due to allowable computer memory and CPU time, the main restriction imposed on the

Table 3
Information available at the end of the computation

Identifier	Type	Printout	Contents
LODE	list	printeqn (lode);	list of the determining equations
PSI	array	listarray (psi);	ψ_i^l evaluated during the computation
ALGSOLS	list	printeqn (algsols);	algebraic solution of the equations $\Delta^l = 0$ for the variables v^l
DIFSUB	array	listarray (difsub);	substitutions used to eliminate the v^l and their derivatives
ETA	array	listarray (eta);	components of ETA
PHI	array	listarray (phi);	components of PHI

original system (1) is related to the substitution procedure described in step (3) of section 2. Indeed, it must be possible for MACSYMA to solve the system algebraically for the v^l, as in eq. (13). For instance, this might prevent the application of the program to a system involving five-fold nested radicals of the u_j^l. Note, however, that in many applications, difficulties of this type can be avoided by a judicious choice of the v^l. It is important to emphasize that the individual equations composing the system need not be polynomial in the x_i, u^l and u_j^l.

If only multiplicative and additive combinations of powers (not necessarily integer ones) of the u_j^l appear in eq. (12) after the substitution procedure, then the determining equations can be used directly. If irrational or transcendental functions of the u_j^l appear in (12), because they were present in the original differential equations or they were introduced during the computations, then the determining equations in the output will also contain some of the u_j^l in a irrational or transcendental way. In that case, the user has the task of obtaining the final list of independent determining equations from the output.

The system of differential equations for which one wants to compute the determinining equations may contain arbitrary parameters and even arbitrary functions of the variables x_i and u^l. However, names for the parameters or functions should not be in conflict with any other identifiers used in the program: in particular, I, J, L, M, P, Q and E1, E2,..., V1, V2,... may *not* be used. Note that MACSYMA is not sensitive to cases, e.g. v1 is the same V1, etc. For arbitrary functions, the dependencies must be declared with the help of a DEPENDS statement before calling SYMMETRY.

The program assumes that the u^l are functions from $\mathbb{R}^p \mapsto \mathbb{R}$. If a system of differential equations contains complex valued dependent variables, i.e. functions from $\mathbb{R}^p \mapsto \mathbb{C}$, the decomposition into real and imaginary parts must be made before using the program (see section 4).

In entering the orginal system of differential equations, any use of C1, C2,..., D1, D2,..., must be avoided since it may confuse MACSYMA which uses these symbols to denote command and display lines, respectively. Furthermore, Greek letters should be avoided (for instance 'beta' refers to the Beta function and 'gamma' to the Gamma function). Before using any special character, quickly check the index of ref. [2,3] to verify if the name or symbol does not interfere with a function or command name in MACSYMA.

When the feedback mechanism is used, avoid any confusion between symbols and parameters occurring in the original equations and those used in the explicit forms for the η's and φ's.

Note that MACSYMA starts the label for an array at 0. For instance, the first component of the array ETA is ETA[0]. Precautionary, we have assigned the value 0 to the first components of such arrays.

4. Example: the Karpman equations

This example shows how the program SYMMGRP.MAX can be used in batch mode. It also illustrates the feedback mechanism for solving the determining equations.

The Karpman equations [53] describe the effect of modulation instability of a high-frequency (whistler) wave due to its resonant parametric interaction with a low-frequency wave in a plasma. The normalized complex amplitude ψ of the whistler wave and the particle density ν of the plasma are given by the non-linear system,

$$\begin{aligned} &\mathrm{i}(\psi_t + w_1\psi_z) + \tfrac{1}{2}\left[s_1(\psi_{xx} + \psi_{yy}) + s_2\psi_{zz}\right] - a_1\nu\psi = 0, \\ &\nu_{tt} - (w_2)^2(\nu_{xx} + \nu_{yy} + \nu_{zz}) - a_2\left(|\psi|_{xx}^2 + |\psi|_{yy}^2 + |\psi|_{zz}^2\right) = 0, \end{aligned} \tag{18}$$

where a_1, a_2, s_1, s_2, w_1 and w_2 are just constants.

Since the program cannot handle the complex variable ψ, we split it in its real and imaginary parts by putting

$$\psi = \rho(x, y, z, t)\exp(\mathrm{i}\phi(x, y, z, t)). \tag{19}$$

Thus,

$$\begin{aligned} &\rho_t + w_1\rho_z + \tfrac{1}{2}\left[s_1(2\rho_x\phi_x + 2\rho_y\phi_y + \rho\phi_{xx} + \rho\phi_{yy}) + s_2(2\rho_z\phi_z + \rho\phi_{zz})\right] = 0, \\ &\phi_t + w_1\phi_z - \tfrac{1}{2}\left[s_1\left(\frac{\rho_{xx}}{\rho} + \frac{\rho_{yy}}{\rho} - \phi_x^2 - \phi_y^2\right) + s_2\left(\frac{\rho_{zz}}{\rho} - \phi_z^2\right)\right] + a_1\nu = 0, \\ &\nu_{tt} - (w_2)^2(\nu_{xx} + \nu_{yy} + \nu_{zz}) - 2a_2\rho(\rho_{xx} + \rho_{yy} + \rho_{zz}) - 2a_2\left(\rho_x^2 + \rho_y^2 + \rho_z^2\right) = 0. \end{aligned} \tag{20}$$

For the system (20) the MACSYMA calculations are fairly lengthy and involved. On some computers, in particular on PCs, it may not be possible to run all the equations at once.

Even on main frame computers it takes a long time. For example, on a VAX 8600, the determining equations in simplified form were obtained in about 4 hours of CPU time. On a VAX 8650 with a central memory of 96 megabyte, this same calculation took 3 hours of CPU time. The number of determinining equations before simplification was 2321. After automatic simplifications only 69 determining equations were left. The peak working size being limited to about 16400 pages, MACSYMA 412.61 needed 100 garbage collections due to dynamic 0 and 1 space overflow.

For users of less sophisticated computers or when working with still larger systems of equations, the strategy is to break up the problem in smaller pieces. The idea is to obtain information about independencies as soon as possible and to submit that information with a subsequent run. This is done with a judicious setting of the parameters and with the feedback mechanism. We illustrate this in all details for the system (20).

For this example the number of independent variables is $p = 4$, the number of dependent variables is $q = 3$ and there are clearly $m = 3$ equations. The correspondences are as follows:

$$\begin{aligned} &x \mapsto \mathrm{x}[1], \quad \rho \mapsto \mathrm{u}[1], \\ &y \mapsto \mathrm{x}[2], \quad \phi \mapsto \mathrm{u}[2], \\ &z \mapsto \mathrm{x}[3], \quad \nu \mapsto \mathrm{u}[3], \\ &t \mapsto \mathrm{x}[4]. \end{aligned} \tag{21}$$

This permits to write eqs. (20) in a standard form accepted by the program (see below under "e1" through "e3"). Finally, one selects the variables needed for the substitution (elimination) process: these will be ρ_t, ϕ_t and ν_{tt}. In the notation of the program, these variables are called v1, v2 and v3.

The "translation" of eqs. (20) is given in the data file KARPMANRUN1.DAT below. For example, ρ_t becomes u[1, [0,0,0,1]], etc.

The batch file containing the MACSYMA commands to run SYMMGRP.MAX is called KARPMANRUN1.COM. It contains:

```
batchload ("symmgrp.max");
writefile ("karpmanrun1.412");
batch ("karpmanrun1.dat");
symmetry (1, 0, 0);
printeqn (lode);
save ("lodekarpmanrun1.lsp", lode);
closefile();
quite();
```

For every new run this batch file has to be slightly updated by changing "run1" into "run2", etc. As for its contents, apart from saving a transcript of the session in KARPMANRUN1.412, we also save the list of determining equations (LODE) as the LISP file LODEKARPMANRUN1.LSP, in case one wants to use these equations in a separate MACSYMA session. In turn, the above command file batches the file KARPMANRUN1.DAT with the data for the first run:

```
p: 4$
q: 3$
m: 3$
parameters: [a1, a2, s1, s2, w1, w2] $
warnings: true $
sublisteqs: [e1] $
info_given: false $
highest_derivatives: 1 $
e1: u[1, [0, 0, 0, 1]] + w1*u[1, [0, 0, 1, 0]] + (1/2)*(s1*(2*u[1, [1, 0, 0, 0]]
    *u[2, [1, 0, 0, 0]] + 2*u[1, [0, 1, 0, 0]]*u[2, [0, 1, 0, 0]] + u[1]*u[2, [2, 0, 0, 0]]
    + u[1]*u[2, [0, 2, 0, 0]]) + s2*(2*u[1, [0, 0, 1, 0]]*u[2, [0, 0, 1, 0]]
    + u[1]*u[2, [0, 0, 2, 0]]));
e2: u[2, [0, 0, 0, 1]] + w1*u[2, [0, 0, 1, 0]] - (1/2)*(s1*(u[1, [2, 0, 0, 0]]/u[1]
    + u[1, [0, 2, 0, 0]]/u[1] - u[2, [1, 0, 0, 0]]^2 - u[2, [0, 1, 0, 0]]^2)
    + s2*(u[1, [0, 0, 2, 0]]/u[1] - u[2, [0, 0, 1, 0]]^2)) + a1*u[3];
e3: u[3, [0, 0, 0, 2]] - v2^2*(u[3, [2, 0, 0, 0]] + u[3, [0, 2, 0, 0]] + u[3, [0, 0, 2, 0]])
    - 2*a2*u[1]*(u[1, [2, 0, 0, 0]] + u[1, [0, 2, 0, 0]] + u[1, [0, 0, 2, 0]])
    - 2*a2*(u[1, [1, 0, 0, 0]]^2 + u[1, [0, 1, 0, 0]]^2 + u[1, [0, 0, 1, 0]]^2);
v1: u[1, [0, 0, 0, 1]];
v2: u[2, [0, 0, 0, 1]];
v3: u[3, [0, 0, 0, 2]];
```

All the parameters a1, a2,..., w2 are supposed to be non-zero and may be canceled (as factors) in simplifications. Since we selected only the first equation of the system (however substituting from the entire system!) and since we extract only the simple determining equations (from the coefficients of the highest derivatives in the prolongation), this run takes only 20 minutes of CPU time on a VAX 8600.

The simplifications implemented in the program reduce the number of determining equations from 20 to 6 single term equations. They are saved in KARPMANRUN1.412 and these equations reveal that eta4 only depends on x[4] as listed in table 4. "No" stands for "not dependent on" and the subscript 1 refers to the information obtained from the *first* run, etc.

For the second run, we provide the program with this information and we ask for the determining equations coming from the second prolongation applied to e2 and e3 only. We modify a few lines in the

Table 4
Dependencies for the Karpman case

	x[1]	x[2]	x[3]	x[4]	u[1]	u[2]	u[3]
eta1	no_4		no_4	no_4	no_3	no_3	no_2
eta2		no_4	no_4	no_4	no_3	no_3	no_2
eta3	no_4	no_4	no_4	no_4	no_3	no_3	no_2
eta4	no_1	no_1	no_1	no_4	no_1	no_1	no_1
phi1						no_3	no_2
phi2					no_3	$linear_3$	no_3
phi3					no_4	no_4	$linear_4$

data file:

```
sublisteqs: [e2, e3] $
info_given: true $
depends([eta1, eta2, eta3, phi1, phi2, phi3], [x[1], x[2], x[3], x[4], u[1], u[2], u[3]]);
depends(eta4, x[4]);
```

All the remaining lines are the same and we save the updated file as KARPMANRUN2.DAT. We run the program again with a batch file similar to the one used for the first run. After 1 h 15 min of CPU time, four determining equations are obtained, they give new information about the dependencies (see table 4).

For the third run, in KARPMANRUN3.DAT we update the information about the dependencies (by changing dependency declarations as shown before) and we ask the program for all the determining equations coming from the first equation:

```
sublisteqs: [e1]$
highest_derivatives: all$
```

In 6 min of CPU time on the VAX 8600, the program extracted 130 determining equations and automatically simplified them to 26 equations. The information from the first 7 (single term) equations is added to table 4. At this point we want to solve some of the 19 remaining equations, to prepare the data for the next run. The determining equations are all linear and homogeneous. So they usually do not require any solution techniques beyond a straightforward separation of variables, occasionally a simple integration, at worst an application of the method of the characteristics. We first look for more information on dependencies. Since phi1 and phi2 are independent of u[3] the (longest) equation, i.e.,

$$2\,\mathrm{w1}\frac{\partial \mathrm{phi1}}{\partial \mathrm{x[3]}} + \mathrm{s2}\,\mathrm{u[1]}\frac{\partial^2 \mathrm{phi2}}{\partial \mathrm{x[3]}^2} + \mathrm{s1}\,\mathrm{u[1]}\frac{\partial^2 \mathrm{phi2}}{\partial \mathrm{x[2]}^2} + \mathrm{s1}\,\mathrm{u[1]}\frac{\partial^2 \mathrm{phi2}}{\partial \mathrm{x[1]}^2} + 2\frac{\partial \mathrm{phi1}}{\partial \mathrm{x[4]}} - 2\,\mathrm{a1}\,\mathrm{u[3]}\frac{\partial \mathrm{phi1}}{\partial \mathrm{u[2]}} = 0, \tag{22}$$

implies that phi1 is also independent of u[2]. With

$$\mathrm{u[1]}^2\frac{\partial \mathrm{phi2}}{\partial \mathrm{u[1]}} + \frac{\partial \mathrm{phi1}}{\partial \mathrm{u[2]}} = 0, \tag{23}$$

we have that phi2 is independent of u[1].Next,

$$\mathrm{u[1]}\frac{\partial^2 \mathrm{phi2}}{\partial \mathrm{u[2]}^2} + \frac{\partial \mathrm{phi1}}{\partial \mathrm{u[2]}} = 0, \tag{24}$$

gives that phi2 is linear in u[2]. We add these three conclusions to table 4.

From the three remaining equations (with only two terms) we learn that

$$\frac{\partial \text{eta3}}{\partial x[1]} = -\left(\frac{s2}{s1}\right)\frac{\partial \text{eta1}}{\partial x[3]}, \tag{25}$$

$$\frac{\partial \text{eta3}}{\partial x[2]} = -\left(\frac{s2}{s1}\right)\frac{\partial \text{eta2}}{\partial x[3]}, \tag{26}$$

$$\frac{\partial \text{eta2}}{\partial x[1]} = -\frac{\partial \text{eta1}}{\partial x[2]}. \tag{27}$$

Comparison of three equations with 4 terms each, such as

$$\frac{\partial \text{phi2}}{\partial u[2]} + u[1]\frac{\partial^2 \text{phi2}}{\partial u[1]\,\partial u[2]} + \frac{\partial \text{eta4}}{\partial x[4]} - 2\frac{\partial \text{eta3}}{\partial x[3]} = 0, \tag{28}$$

leads to

$$\frac{\partial \text{eta3}}{\partial x[3]} = \frac{\partial \text{eta2}}{\partial x[2]} = \frac{\partial \text{eta1}}{\partial x[1]}. \tag{29}$$

Upon integration of eq. (28) we get

$$\text{phi2} = \left(2\frac{\partial \text{eta1}}{\partial x[1]} - \frac{\partial \text{eta4}}{\partial x[4]}\right)u[2] + f2(x[1], x[2], x[3], x[4]), \tag{30}$$

where f2 will be determined later. Substitution of eq. (30) into

$$u[1]\frac{\partial \text{phi2}}{\partial u[2]} - u[1]\frac{\partial \text{phi1}}{\partial u[1]} + \text{phi1} + u[1]\frac{\partial \text{eta4}}{\partial x[4]} - 2u[1]\frac{\partial \text{eta1}}{\partial x[1]} = 0 \tag{31}$$

and integration yields

$$\text{phi1} = f1(x[1], x[2], x[3], x[4])u[1], \tag{32}$$

where f1 will be determined later.

To save time we shall not solve the rest of the equations for eta1, eta2, eta3 and phi3 but rather submit the above information and carry out the next run.

Hence, the data file KARPMANRUM4.DAT contains the information from table 4 and also the lines

```
sublisteqs: [all] $
depends([f1, f2], [x[1], x[2], x[3], x[4]]);
phi1: f1*u[1];
phi2: (2*diff(eta1, x[1]) − diff(eta4, x[4]))*u[2] + f2;
gradef(eta3, x[1], −(s2/s1)*diff(eta1, x[3]));
gradef(eta3, x[2], −(s2/s1)*diff(eta2, x[3]));
gradef(eta2, x[1], −diff(eta1, x[2]));
gradef(eta3, x[3], diff(eta1, x[1]));
gradef(eta2, x[2], diff(eta1, x[1]));
```

After 28 min of CPU time, 30 simple determining equations are obtained (see the Test Run Output). The simplifications described in the outline of the program actually reduced 249 determining equations to 30 this time. Since $s1 \neq s2$, 10 of these equations together with the conditions (25)–(27) and (29) lead to the information listed in table 4.

A quick inspection of the remaining equations in the Test Run Output allows to conclude that f1 = k6 is constant. Hence, with eq. (32) we get

$$\text{phi1} = \text{k6 u[1]}. \tag{33}$$

Similarly, eq. (30) simplifies into

$$\text{phi2} = \text{f2(x[4])}. \tag{34}$$

Further, we obtain phi3 up to an unknown function f4,

$$\text{phi3} = 2\ \text{k6 u[3]} + \text{f4(x[1], x[2], x[3], x[4])}. \tag{35}$$

We also find that eta1 is linear in x[2], i.e.

$$\text{eta1} = \text{k1 x[2]} + \text{k2}, \tag{36}$$

where k1 and k2 are constants. The eqs. (25)–(27) and (29) then determine

$$\text{eta2} = -\text{k1 x[1]} + \text{k3}, \tag{37}$$

$$\text{eta3} = \text{k4}, \tag{38}$$

$$\text{eta4} = \text{k5}. \tag{39}$$

We again modify the previous data file, to account for the info in table 4, the forms of the eta's and phi's and the dependencies of f2 and f4:

```
depends(f2, x[4]);
depends(f4, [x[1], x[2], x[3], x[4]]);
eta1: k1*x[2] + k2;
eta2: -k1*x[1] + k3;
eta3: k4;
eta4: k5;
phi1: k6*u[1];
phi2: f2;
phi3: 2*k6*u[3] + f[4];
```

and with this file KARPMANRUN5.DAT we start the last run. Only 2 determining equations are left in KARPMANRUN5.412:

$$2\ \text{u[3]}\ \text{a1}\ \text{k6} + \text{a1}\ \text{f4} + \frac{\partial \text{f2}}{\partial \text{x[4]}} = 0, \tag{40}$$

$$\text{w2}^2 \frac{\partial^2 \text{f4}}{\partial \text{x[3]}^2} + \text{w2}^2 \frac{\partial^2 \text{f4}}{\partial \text{x[2]}^2} + \text{w2}^2 \frac{\partial^2 \text{f4}}{\partial \text{x[1]}^2} - \frac{\partial^2 \text{f4}}{\partial \text{x[4]}^2} = 0. \tag{41}$$

The first one requires that k6 = 0, hence, phi1 = 0, and also

$$\frac{\partial \text{f2}}{\partial \text{x[4]}} = -\text{a1}\ \text{f4}. \tag{42}$$

Since f2 depends only on x[4], f4 must be independent of x[1], x[2] and x[3]. As a consequence of eq. (41), f4 is linear in x[4] and with eq. (42) the final solution is known,

$$\text{f2} = \text{a1 k7 x[4]}^2 + \text{a1 k8 x[4]} + \text{k9}, \tag{43}$$

$$\text{f4} = -2\ \text{k7 x[4]} - \text{k8}, \tag{44}$$

where k7, k8 and k9 are free constants. These functions determine the final form of phi2 and phi3 in eqs. (34) and (35).

One could submit these data for verification. We have done so and no determining equations were left, as expected. This final test of the solution took only 1 min 30 s of CPU time.

Let us summarize. The general solution of the determining equations leads to a Lie algebra with 8 infinitesimal generators. In terms of the original independent variables x, y, z, t and the dependent variables ρ, ϕ and ν, the vector field reads:

$$\alpha = \eta^x \frac{\partial}{\partial x} + \eta^y \frac{\partial}{\partial y} + \eta^z \frac{\partial}{\partial z} + \eta^t \frac{\partial}{\partial t} + \varphi^\rho \frac{\partial}{\partial \rho} + \varphi^\phi \frac{\partial}{\partial \phi} + \varphi^\nu \frac{\partial}{\partial \nu}, \tag{45}$$

where

$$\begin{aligned} &\eta^x = k_1 y + k_2, && \varphi^\rho = 0, \\ &\eta^y = -k_1 x + k_3, && \varphi^\phi = k_7 a_1 t^2 + k_8 a_1 t + k_9, \\ &\eta^z = k_4, && \varphi^\nu = -2k_7 t - k_8, \\ &\eta^t = k_5. \end{aligned} \tag{46}$$

Here k_1 through k_9 are independent arbitrary constants ($k_6 = 0$ making $\varphi^\rho = 0$). Recall that a_1 is a parameter in the Karpman equations (18). The 8 infinitesimal generators for these equations are

$$\begin{aligned} &P_1 = \partial_x, && L_3 = y\,\partial_x - x\,\partial_y, \\ &P_2 = \partial_y, && R_1 = \partial_\phi, \\ &P_3 = \partial_z, && R_2 = a_1 t\,\partial_\phi - \partial_\nu, \\ &P_4 = \partial_t, && R_3 = a_1 t^2\,\partial_\phi - 2t\,\partial_\nu. \end{aligned} \tag{47}$$

5. Conclusion

We presented a MACSYMA program that can assist in the calculation of the symmetry group of a system of differential equations. Among various features of this program, let us emphasize a few.

(1) The program is applicable to a system of m equations of order k, with q unknowns and p independent variables, where all these labels are arbitrary positive integers.
(2) The output is a system of determining equations that is free of repetition and partially solved in the sense that higher-order equations which are differential consequences of lower-order ones are automatically eliminated.
(3) The parameters "sublisteqs" and "highest-derivatives" allow partial information to be extracted very rapidly. These parameters help prevent MACSYMA running out of space (and/or time) when very large systems are submitted.
(4) Warnings remind the user about division by parameters that were listed as different from zero.
(5) The feedback mechanism allows the determining equations to be solved step by step on the computer, hence avoiding human error in the algebraic simplifications.
(6) The program can be used to test solutions of the determining equations and hence detect errors in the literature on the subject.
(7) The program can be used interactively and in batch mode and the amount of messages that are printed out can be adequately controlled.
(8) The program needs very little data and is straightforward to use provided the user has access to MACSYMA.

Application of this program to determine the symmetry group of the Karpman equations has been straightforward and has lead to new results. The development of a MACSYMA program that solves the determining equations in part is planned for the future. Upon modification of the algorithm, the program can be extended to the computation of more general Lie–Bäcklund transformation groups [7].

Acknowledgements

This work was partially supported by research grants from the Natural Science and Engineering Research Council of Canada and the "FCAR du Gouvernement du Québec". B.C. acknowledges a fellowship awarded by NSERC of Canada. W.H. is grateful for support from the Air Force Office of Scientific Research under Grant No. 85-NM-0263. He also thanks the Centre de Recherches Mathématiques for their hospitality and support. The authors thank D. Rand for help with MACSYMA, and A.K. Head, D. Holm, M. Homer, A. Mikhailov, P. Olver, G. Prince, W. Sarlet and S. Steinberg for bringing various software packages to their attention. M. Grundland and D. Levi are gratefully acknowledged for using the program extensively and suggesting improvements.

Notes added in proof

One new parameter must be added to the input data as described in subsection 3.3 of this paper. The parameter subst_deriv_of_vi: true, controls the substitution of the partial derivatives of the v^i in eq. (12). These derivatives are given by (14).

In some cases it is not possible to select the v^i in such a way that the differential consequences would not reintroduce lower order derivatives of the v^i, hence causing a loop! Therefore, we have made the substitution of the partial derivatives of the v^i optional. If only the v^i should be replaced and not their derivatives, one puts subst_deriv_of_vi: false.

The resulting system of determining equations is "equivalent" with the one obtained using the substitution of all the partial derivatives of the v^i. In the later case the system of determining equations may be somewhat simpler, but the extra substitutions consume time.

The authors became aware of yet another REDUCE program for the calculation of Lie symmetries (including Lie–Bäcklund symmetries) developed by Clara Maria Nucci at the School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia.

References

[1] P.J. Olver, Applications of Lie Groups to Differential Equations (Springer, New York, 1988).
[2] MACSYMA Reference Manual, Version 13, Computer Aided Mathematics Group (Symbolics, Burlington, MA, 1989).
[3] MACSYMA User's Guide, Computer Aided Mathematics Group (Symbolics, Burlington, MA, 1988).
[4] W.F. Ames, Nonlinear Partial Differential Equations in Engineering (Academic, New York, 1972).
[5] G.W. Bluman and J.D. Cole, Similarity Methods for Differential Equations (Springer, New York, 1974).
[6] W. Miller, Symmetry and Separation of Variables (Addison-Wesley, Reading, MA, 1977).
[7] R.L. Anderson and N.H. Ibragimov, Lie–Bäcklund Transformations in Applications, Studies in Applied Mathematics vol. 1 (SIAM, Philadelphia, 1979).
[8] L.V. Ovsiannikov, Group Analysis of Differential Equations (Academic, New York, 1982).
[9] P. Winternitz, in: Nonlinear Phenomena, K.B. Wolf, ed., Lecture Notes in Physics vol. 189 (Springer, New York, 1983) p. 263.
[10] N.H. Ibragimov, Transformation Groups Applied to Mathematical Physics (Reidel, Boston, 1985).
[11] E.G. Kalnins, Separation of Variables for Riemannian Spaces of Constant Curvature (Longman, Essex, 1986).
[12] D.H. Sattinger and O.L. Weaver, Lie Groups and Algebras with Applications to Physics, Geometry, and Mechanics (Springer, New York, 1986).
[13] V.I. Fushchich and A.G. Nikitin, Symmetries of Maxwell Equations (Reidel, Dordrecht, 1987).

[14] D. Levi and P. Winternitz, eds., Symmetries and Nonlinear Phenomena (World Scientific, Singapore, 1988).
[15] F. Schwarz, SIAM Rev. 30 (1988) 450.
[16] G.W. Bluman and S. Kumei, Symmetries and Differential Equations (Springer, New York, 1989).
[17] C. Rogers and W.F. Ames, Nonlinear Boundary Value Problems in Science and Engineering (Academic, New York, 1989).
[18] H. Stephani, Differential Equations: Their Solution using Symmetries (Cambridge Univ. Press, Cambridge, 1989).
[19] P. Winternitz, in: Partially Integrable Nonlinear Evolution Equations and their Physical Applications, R. Conte and N. Boccara, eds. (Kluwer, Dordrecht, 1990) p. 515.
[20] A.M. Vinogradov, ed., Symmetries of partial differential equations, Part I, Acta Appl. Math. 15 (1 & 2) (1989); Part II, Acta Appl. Math. 16 (1) (1989); Part III, Acta Appl. Math. 16 (2) (1989).
[21] F. Schwarz, Comput. Phys. Commun. 27 (1982) 179.
[22] F. Schwarz, Computing 34 (1985) 91; Addendum, Computing 36 (1986) 279.
[23] F. Schwarz, Comput. Phys. Commun. 39 (1986) 285.
[24] F. Schwarz, The Package SPDE for Determining Symmetries of Partial Differential Equations, User's Manual, distributed with REDUCE 3.3 (Rand Corp., Santa Monica, CA, 1987).
[25] F. Schwarz, in: Trends in Computer Algebra, R. JanBen, ed., Lecture Notes in Computing Science vol. 296 (Springer, New York, 1988) p. 167.
[26] D.G.B. Edelen, Isovector Methods for Equations of Balance (Sijthoff & Noordhoff, Alphen aan den Rijn, 1981).
[27] P. Gragert, P.H.M. Kersten and A. Martini, Acta Appl. Math. 1 (1983) 43.
[28] P.H.M. Kersten, Infinitesimal symmetries: a computational approach, CWI Tract 34, Center for Mathematics and Computer Science, Amsterdam (1987).
[29] P.H.M. Kersten, Acta Appl. Math. 16 (1989) 207.
[30] V.P. Eliseev, R.N. Fedorova and V.V. Kornyak, Comput. Phys. Commun. 36 (1985) 383.
[31] R.N. Fedorova and V.V. Kornyak, A REDUCE program for computing determining equations of Lie–Bäcklund symmetries of differential equations, Report R11-87-19, JINR, Dubna (1987).
[32] R.N. Fedorova and V.V. Kornyak, Comput. Phys. Commun. 39 (1986) 93.
[33] W.I. Fushchich and V.V. Kornyak, J. Symb. Comput. 7 (1989) 611.
[34] T. Wolf, Analytic solutions of differential equations with computer algebra systems, Preprint 87/5, Universität Jena (1987).
[35] A.K. Head, LIE: A muMATH Program for the calculation of the LIE algebra of differential equations (CSIRO Division of Material Sciences, Clayton, 1990).
[36] P. Vafeades, in: Proc. ISMM Int. Symp. Computer Applications in Design, Simulation and Analysis, E.K. Park, ed. (New Orleans, Louisiana, 1990) p. 310.
[37] P. Vafeades, SYMCON: a MACSYMA package for the determination of symmetries and conservation laws of PDEs, J. Symb. Comput., submitted.
[38] M.D. Popov, Izv. Akad. Nauk B. SSR Ser. Fiz. Mat. 2 (1985) 33.
[39] A.V. Bocharov, DEliA: a system of exact analysis of differential equations using S. Lie approach, Report OWIMEX, Program Systems Institute, Pereslavl-Zalessky, USSR (1989).
[40] A.V. Bocharov and M.L. Bronstein, Acta Appl. Math. 16 (1989) 143.
[41] D.K. Davison, MANDO: a computer program for symbolic manipulation of differential operators generating continuous transformations, M.Sc. Thesis, University of the Pacific, Stockton, CA (1973).
[42] G.G. Nagao, DETERMININGEQS: a computer program for approximating Lie generators admitted by dynamical systems, M.Sc. Thesis, University of the Pacific, Stockton, CA (1980).
[43] P. Rosenau and J.L. Schwarzmeier, Similarity solutions of systems of partial differential equations using MACSYMA, Report COO-3077-160 MF-94, Courant Institute of Mathematical Sciences, New York University (1979).
[44] J.L. Schwarzmeier and P. Rosenau, Using MACSYMA to calculate similarity transformations of partial differential equations, Report LAUR 88-4157, Los Alamos National Laboratory (1988).
[45] S. Steinberg, in: Proc. 1979 MACSYMA User's Conf., V.E. Lewis, ed. (MIT Press, Boston, 1979) p. 408.
[46] S. Steinberg, MACSYMA Newsletter 7 (1990) 3.
[47] B. Champagne and P. Winternitz, A MACSYMA program for calculating the symmetry group of a system of differential equations, Report CRM-1278, Centre de Recherches Mathématiques, Montréal (1985).
[48] S.I. Rosencrans, Comput. Phys. Commun. 38 (1985) 347.
[49] F. Schwarz, An algorithm for determining the size of symmetry groups, private communication (1989).
[50] G.J. Reid, in: Lie Theory, Differential Equations and Representation Theory, Proc. Annual Seminar of the Canadian Math. Soc., V. Hussin, ed. (Les Publications de Centre de Recherches Mathématiques, Montréal, 1990) p. 363.
[51] G.J. Reid, Finding abstract Lie symmetry algebras of differential equations without integrating determining equations, Technical Report 90-4, Inst. of Applied Mathematics, The University of British Columbia, Vancouver (1990); Europ. J. Appl. Math, in press.
[52] G.J. Reid, J. Phys. A. 23 (1990) L853.
[53] V.I. Karpman, Phys. Lett. A 136 (1989) 216.

TEST RUN OUTPUT

```
You are using the 3 equations of the system.

*** Number of determining equations before simplifications: 249 . ***

                                    2      2
WARNING ! We eliminated the factor:  U  S1 S2
                                     1

                                  2 dPHI3      2
which was the coefficient of  - U  ----- S1 S2
                                  1  dU
                                       2

                                     2                          3          2
List of factors that are cancelled: [U  A2 S1, U , U  S2, U  A2 S1, U  A2,
                                      1         1   1        1          1

             3      2      2   2   2           2      3     2   2      2
            U  A2, U  S1 W2 , U  W2 , S1, S2, U  S1, U  S1 S2 , U  S1 S2 ]
             1      1          1               1      1          1

*** Number of determining equations after simplifications: 30 . ***

*** These determining equations are stored in LODE. ***

(C32) PRINTEQN(LODE);

                dPHI3
Equation 1 : ----- = 0
                dU
                  2

                dPHI3
Equation 2 : ----- = 0
                dU
                  1

                 2
                d PHI3
Equation 3 : ------ = 0
                   2
                 dU
                   3

                dETA3
Equation 4 : ----- = 0
                dX
                  4

                dETA2
Equation 5 : ----- = 0
                dX
                  4

                dETA1
Equation 6 : ----- = 0
                dX
                  4


Equation 7 :

          2
  dF1   d ETA2
2 --- + ------- = 0
  dX     dX  dX
    3      2   3

Equation 8 :

dETA4      dETA1
----- - 2 ----- = 0
 dX         dX
   4          1
```

Equation 9 :

$$\frac{d^2 ETA2}{dX_3^2} S2 - 2\frac{dF1}{dX_2} S1 = 0$$

Equation 10 :

$$\frac{dETA2}{dX_3} (S2 - S1) = 0$$

Equation 11 :

$$\frac{dETA1}{dX_3} (S2 - S1) = 0$$

Equation 12 :

$$\frac{dETA4}{dX_4} - \frac{dETA1}{dX_1} = 0$$

Equation 13 :

$$2\frac{d^2 PHI3}{dU\, dX_3\, dX_4} - \frac{d^2 ETA4}{dX_4^2} = 0$$

Equation 14 :

$$2\frac{d^2 PHI3}{dX_2\, dU\, dX_3} - \frac{d^2 ETA2}{dX_3^2} = 0$$

Equation 15 :

$$4\frac{dF1}{dX_2} - \frac{d^2 ETA2}{dX_3^2} = 0$$

Equation 16 :

$$2\frac{dF1}{dX_3} + \frac{d^2 ETA2}{dX_2\, dX_3} + 4\frac{d^2 ETA1}{dX_1\, dX_3} = 0$$

Equation 17 :

$$\frac{d^2 ETA2}{dX_3^2} S2 - 2\frac{dF1}{dX_2} S1 - 4\frac{d^2 ETA1}{dX_1\, dX_2} S1 = 0$$

Equation 18 :

$$\frac{dETA2}{dX_3} W1 - \frac{dF2}{dX_2} S1 - 2U\frac{d^2 ETA1}{dX_1\, dX_2} S1 = 0$$

Equation 19 :

$$\frac{dETA1}{dX_3} W1 - \frac{dF2}{dX_1} S1 - 2U\frac{d^2 ETA1}{dX_2^2} S1 = 0$$

Equation 20 :

$$\frac{d^2 ETA1}{dX_3^2} S2 - 2\frac{dF1}{dX_1} S1 + \frac{d^2 ETA1}{dX_2^2} S1 - 3\frac{d^2 ETA1}{dX_2^2} S1 = 0$$

Equation 21 :

$$\frac{dETA4}{dX_4} W1 - \frac{dETA1}{dX_1} W1 + \frac{dF2}{dX_3} S2 + 2U\frac{d^2 ETA1}{2\, dX_1\, dX_3} S2 = 0$$

Equation 22 :

$$\frac{d^2 ETA1}{dX_3^2} S2 - 2\frac{dF1}{dX_2} S1 + \frac{d^2 ETA1}{dX_2^2} S1 + \frac{d^2 ETA1}{dX_2^2} S1 = 0$$

Equation 23 :

$$\frac{d^2 ETA2}{dX_2\, dX_3} S2 + \frac{d^2 ETA1}{dX_2\, dX_3} S2 + 2\frac{d^2 PHI3}{dU\, dX_3} S1 - \frac{d^2 ETA1}{dX_1\, dX_3} S1 = 0$$

Equation 24 :

$$2\frac{d^2 PHI3}{dX_1\, dU} - \frac{d^2 ETA1}{dX_3^2} - \frac{d^2 ETA1}{dX_2^2} - \frac{d^2 ETA1}{dX_1^2} = 0$$

Equation 25 :

$$\frac{dPHI3}{dU} - 2\,F1 - 2\frac{dETA4}{dX_4} + 2\frac{dETA1}{dX_1} = 0$$

Equation 26 :

$$4\frac{dF1}{dX_1} - \frac{d^2 ETA1}{dX_3^2} - \frac{d^2 ETA1}{dX_2^2} - \frac{d^2 ETA1}{dX_1^2} = 0$$

Equation 27 :

$$\frac{d^2 ETA2}{dX_2\, dX_3} S2 + \frac{d^2 ETA1}{dX_2\, dX_3} S2 + 4\frac{dF1}{dX_3} S1 - \frac{d^2 ETA1}{dX_1\, dX_3} S1 = 0$$

Equation 28 :

$$\frac{d^2\,PHI3}{dX_3^2}\,W2 + \frac{d^2\,PHI3}{dX_2^2}\,W2 + \frac{d^2\,PHI3}{dX_1^2}\,W2 - \frac{d^2\,PHI3}{dX_4^2} + 2\,U_1\,A2\,\frac{d^2\,F1}{dX_3^2} + 2\,U_1\,A2\,\frac{d^2\,F1}{dX_2^2} + 2\,U_1\,A2\,\frac{d^2\,F1}{dX_1^2} = 0$$

Equation 29 :

$$2\,\frac{dF1}{dX_3}\,W1 + \frac{d^2\,F2}{dX_3^2}\,S2 + 2\,U_2\,\frac{d^3\,ETA1}{dX_1\,dX_3^2}\,S2 + \frac{d^2\,F2}{dX_2^2}\,S1 + \frac{d^2\,F2}{dX_1^2}\,S1 + 2\,U_2\,\frac{d^3\,ETA1}{dX_1^3}\,S1 + 2\,U_2\,\frac{d^3\,ETA1}{dX_1\,dX_2^2}\,S1 + 2\,\frac{dF1}{dX_4} = 0$$

Equation 30 :

$$2\,\frac{dF2}{dX_3}\,W1 + 4\,U_2\,\frac{d^2\,ETA1}{dX_1\,dX_3}\,W1 - \frac{d^2\,F1}{dX_3^2}\,S2 - \frac{d^2\,F1}{dX_2^2}\,S1 - \frac{d^2\,F1}{dX_1^2}\,S1 + 2\,A1\,PHI3 + 2\,\frac{dF2}{dX_4} - 2\,U_2\,\frac{d^2\,ETA4}{dX_2^2} + 4\,U_3\,A1\,\frac{dETA4}{dX_4} - 4\,U_3\,A1\,\frac{dETA1}{dX_1} = 0$$

(D32) DONE