Commun. Theor. Phys. **69** (2018) 14–22 Vol. 69, No. 1, January 1, 2018

A Direct Algorithm Maple Package of One-Dimensional Optimal System for Group Invariant Solutions*

Lin Zhang (张琳),[1] Zhong Han (韩众),[1] and Yong Chen (陈勇)[1,2,†]

[1]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China

[2]Department of Physics, Zhejiang Normal University, Jinhua 321004, China

(Received October 25, 2017; revised manuscript received November 30, 2017)

Abstract *To construct the one-dimensional optimal system of finite dimensional Lie algebra automatically, we develop a new Maple package One Optimal System. Meanwhile, we propose a new method to calculate the adjoint transformation matrix and find all the invariants of Lie algebra in spite of Killing form checking possible constraints of each classification. Besides, a new conception called invariance set is raised. Moreover, this Maple package is proved to be more efficiency and precise than before by applying it to some classic examples.*

PACS numbers: 02.70.Wz, 02.20.Sv, 05.45.Yv, 02.30.Jr **DOI:** 10.1088/0253-6102/69/1/14
Key words: symbolic computation, mathematics mechanization, direct algorithm, optimal system

1 Introduction

Symmetry group theory has a priority to gain much concern in finding exact solutions of differential equations.[1−5] The theory is commonly applied to construct explicit solutions for integrable and non-integrable nonlinear equations. Based on the theory, an original nonlinear system of any given subgroup can be reduced to a system with fewer independent variables. The reduction of the variables is established on group invariant solutions. Since infinite subgroups always exist, which means the impossibility of finding all the group invariant solutions[6−9] to the given system, the classification[10−20] of symmetry subgroup is essentially important in solving the problem. Then to find those complete but inequivalent group-invariant solutions is expected. From now on, many systematic and effective method has formulated, and the concept of "optimal system" for group-invariant solutions is proposed to solve the problem.

People always find an optimal system of subalgebras instead of finding subgroups, which are equivalent. Ovsiannikov[21] initially uses the adjoint representation[22] of a Lie group on Lie algebra to classify group-invariant solutions and the method is developed by Olver, Winternitz, Zassenhaus, and Patera[23−24] afterwards. In particular, Olver[22] gave an ingenious method, which is more simple to calculate. For the one-dimensional optimal system, he successfully applied the method in the Korteweg-de Vries (KdV) equation and the heat equation. Later on, many new methods have appeared and the mechanization has been realized[25−31] through computer. The mechanized realization simplifies complicated manual computation efficiently. So to many important partial differential equations, this has always been paid substantial concentration.

The objective of this paper is to give a new *Maple* package for constructing the one-dimensional optimal system,[32] which can be more precise and common used. Here we propose a new symbolic computation algorithm and develop a *Maple* package named *One Optimal System*. For a given Lie algebra, it can find different cases of classification and the corresponding one-dimensional optimal system automatically. Compared to existing software packages, the improvement of the new software package is that it gets more precise result, which is equal to Olver's result. What is more, it can be more common used in many important partial differential equations. So we have made a meaningful improvement in constructing the one-dimensional optimal system of finite dimensional Lie algebra.

This paper is arranged as follows. In Sec. 2, we present some necessary definitions and formulas refereed in the paper. Section 3 includes an illustration of essential algorithm details and description of some vital functions and variables. In Sec. 4, we verify the validity and efficiency of the software package by applying it to some well-known partial differential equations. In Sec. 5, we generalize the advantages and innovations of our work and propose future work expectation.

2 Conceptions and Formulas

2.1 *Optimal System*

Suppose G is a Lie group. We define H as an optimal system of s-parameter subgroups, which are inequivalent

*Supported by the Global Change Research Program of China under Grant No. 2015CB95390, National Natural Science Foundation of China under Grant Nos. 11675054 and 11435005, and Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things under Grant No. ZF1213

†Corresponding author, E-mail: ychen@sei.ecnu.edu.cn

to each other. It is a system of differential equations in p ($p > s$) independent variables based on a family of group-invariant solutions. For any elements $g \in G$ not in H can be transformed to the ones included in H. In general, the subgroups in the optimal system can represent the whole elements from the infinite subgroups of a given equation. Commonly, we use symmetry subalgebra to form an optimal system instead of subgroups.

2.2 *Adjoint Representation*

Let $\mathcal{G}$ be an n-dimensional Lie algebra and v_i ($i = 1 \dots n$) be n generators in the vector fields of $\mathcal{G}$. The adjoint representation is defined as

$$\mathrm{Ad}_{\exp(\varepsilon_1 v_1)}(v_2) = v_2 - \varepsilon[v_1, v_2] + \frac{1}{2!}\varepsilon^2[v_1, [v_1, v_2]] - \cdots , \quad (1)$$

and if we suppose $v = \sum_{i=1}^{n} a_i v_i$, we can apply the adjoint representation of v_1 to v as follows

$$\begin{aligned} &\mathrm{Ad}_{\exp(\varepsilon_1 v_1)}(a_1 v_1 + a_2 v_2 + \cdots + a_n v_n) \\ &\quad = a_1 \mathrm{Ad}_{\exp(\varepsilon_1 v_1)} v_1 + a_2 \mathrm{Ad}_{\exp(\varepsilon_1 v_1)} v_2 + \cdots \\ &\qquad + a_n \mathrm{Ad}_{\exp(\varepsilon_1 v_1)} v_n \\ &\quad = R_1 v_1 + R_2 v_2 + \cdots + R_n v_n \,, \end{aligned} \quad (2)$$

with $R_i \equiv R_i(a_1, a_2, \dots, a_n, \varepsilon_1)$, $i = 1 \cdots n$. Furthermore, we can transform formula (2) into the following matrix form:

$$\begin{aligned} v = (a_1, a_2, \dots, a_n) &\longrightarrow (R_1, R_2, \dots, R_n) \\ &= (a_1, a_2, \dots a_n) A_1 \,. \end{aligned} \quad (3)$$

In the same manner, the corresponding matrix A_2, $A_3, \dots, A_n$ can be received by applying the adjoint action of $v_2, v_3, \dots, v_n$ to v. Then we can construct the adjoint transformation matrix A in the following form

$$A \equiv A(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n) = A_1 A_2 \cdots A_n \,. \quad (4)$$

The product of $A_1, A_2, \dots, A_n$ do not have to be multiplied in an exact order in spite of receiving different A. On the other hand, formula (2) means to apply the product of adjoint action $\mathrm{Ad}_{\exp(\varepsilon_n v_n)} \cdots \mathrm{Ad}_{\exp(\varepsilon_2 v_2)} \mathrm{Ad}_{\exp(\varepsilon_1 v_1)}$ to v. So we have

$$\begin{aligned} v = (a_1, a_2, \dots, a_n) &\longrightarrow (\tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_n) \\ &= (a_1, a_2, \dots, a_n) A \,. \end{aligned} \quad (5)$$

Thus, generator $\sum_{i=1}^{n} a_i v_i$ can be transformed to $\sum_{i=1}^{n} \tilde{a}_i v_i$ through the adjoint transformation matrix A. That is to say $\sum_{i=1}^{n} a_i v_i$ is equivalent to $\sum_{i=1}^{n} \tilde{a}_i v_i$ under the adjoint action.

2.3 *Invariant*

Suppose a real function ϕ. For all $v \in \mathcal{G}$ and all $g \in G$, it satisfies

$$\phi(\mathrm{Ad}_g(v)) = \phi(v) \,. \quad (6)$$

Then ϕ can be called an invariant. For any invariant, two vectors v and v' are equivalent under the adjoint action. As Olver[21] said, it is very important to find the invariant that decides how far we can simplify a Lie algebra. We expect to find all the invariants of the given symmetry algebra $\mathcal{G}$ in spite of Killing form.

2.4 *Classification Rules*

We classify all the generators by assigning different values to the invariants. The rules of classification are as follows:

(i) Check every invariant's degree. If we find the degree of an invariant is even, we can make it either 1 or 0. On the other hand, if the degree is odd, we can make it 1, -1 or 0.

(ii) For every invariant, it must be assigned either 1 or -1 (if degree is odd) firstly. And in each case, we must and just make only one invariant (marked as a flag invariant) either 1 or -1. Then the remaining invariants are assigned c at the same time. After that, we can make the flag invariant 0 with any other invariant assigned either 1 or -1. And the remaining invariants are assigned c at the same time.

(iii) Once an invariant is assigned 0, it cannot be assigned c any more. When all the invariants have been assigned either 1 or -1. The last case must be all the invariants are assigned 0.

2.5 *Invariance Set*

Suppose a nonzero vector $\sum_{i=1}^{n} a_i v_i$, which is an ordinary generator of Lie algebra $\mathcal{G}$. $a_m, a_{m+1}, \dots, a_{m+n}$, n arbitrary coefficients of v. We assume that all of $a_m, a_{m+1}, \dots, a_{m+n}$ are not 0 with the rest of coefficients of v are 0. Then v can be transformed to an equivalent generator v' with none of $a'_m, a'_{m+1}, \dots, a'_{m+n}$ are 0. The transformation is based on formula (5). If v satisfies above transformation, $\{a_m = 0, a_{m+1} = 0, \dots, a_{m+n} = 0\}$ is an invariance set. We should notice that not all the Lie algebra have an invariance set. The existence of invariance set is not relative to the generator v you choose, but depends on the adjoint transformation matrix A.

3 Algorithms

We developed an automated software package *One Optimal System* on *Maple* versions 16 and above. The package is initialized by the command with *One Optimal System*. The software package is consisted of Lie_Bracket, commutator, lie_bracket, commutator_table, lie_bracket_cal, Linear Co, e_index, e0,eps, data_data, fun_set, n_deta, deta_pd, ep_solve, judge_s, special_e, *aaa*_solve, get_rid, get_solution, beauty_m, check_30, exper_30, check_fun, trans_ornot, excep_vv, deno_ax, beauty_c, and *One Optimal* functions. The main function is *One Optimal* := proc(vs., consts). We will illustrate the whole software flow and introduce algorithm of fundamental steps with some important functions and parameters description.

3.1 *Overview of One Optimal System*

Given the m-dimensional Lie algebra $\mathcal{G}$. Firstly, we can get the commutator table by formula (7) for every pairs of generators:

$$[v_i, v_j] = v_i v_j - v_j v_i \,. \tag{7}$$

Assume $v = \sum_{i=1}^{n} a_i v_j$ and $w = \sum_{i=1}^{n} b_j v_j$. By applying the adjoint representation of w to v, we have

$$\begin{aligned} \mathrm{Ad}_{\exp(ew)}(v) &= v - \varepsilon[w, v] + \frac{1}{2!}\varepsilon^2[w, [w, v]] - \cdots \\ &= (a_1 v_1 + \cdots + a_n v_n) - \varepsilon[b_1 v_1 + \cdots \\ &\quad + b_n v_n, a_1 v_1 + \cdots + a_n v_n] + o(\varepsilon^2) \\ &= (a_1 v_1 + \cdots + a_n v_n) \\ &\quad - \varepsilon(\Theta_1 v_1 + \cdots + \Theta_n v_n) + o(\varepsilon^2)\,, \end{aligned} \tag{8}$$

where $\Theta_i \equiv \Theta_i(a_1, \ldots, a_n, b_1, \ldots, b_n)$ is gained from the commutator table.

According to the definition of invariants (6), we get

$$\begin{aligned} &\phi(a_1, a_2, \ldots, a_n) = \\ &\phi(a_1 - \varepsilon\Theta_1 + o(\varepsilon), a_2 - \varepsilon\Theta_2 + o(\varepsilon) \ldots, a_n - \varepsilon\Theta_n + o(\varepsilon)) = 0. \end{aligned} \tag{9}$$

Take the derivative of Eq. (9) with respect to ε and set $\varepsilon = 0$

$$\Theta_1 \frac{\partial \phi}{\partial a_1} + \cdots + \Theta_n \frac{\partial \phi}{\partial a_n} = 0\,. \tag{10}$$

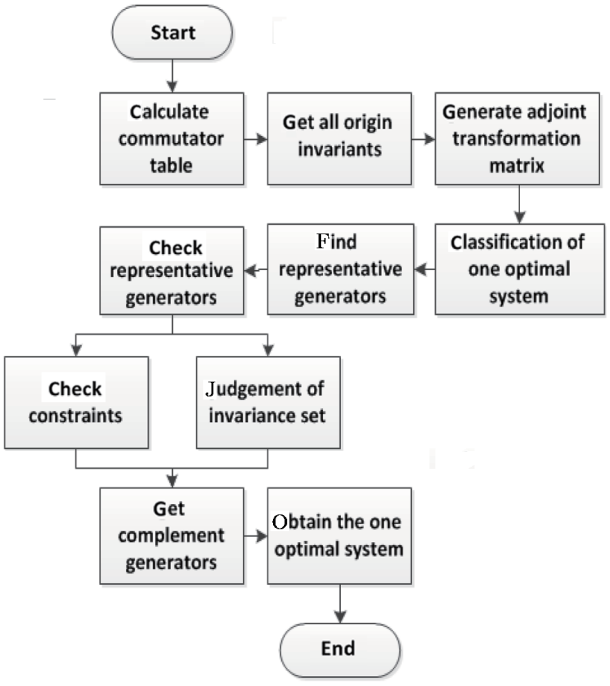

Fig. 1 Software flow diagram of the *One Optimal System*.

By extracting the coefficients of b_i, $i = 1, \ldots, n$, we obtain $N(N \leq n)$ linear differential equations about $\phi(a_1, \ldots, a_n)$. Solve the equations, the invariant ϕ is obtained. According to the commutator table, we can get the adjoint transformation matrix by using formula (1). Then we classified different cases through the classification rules. For each case, we must choose a representative generator and check the chosen generator in case of an incomplete representation. What is more, we should also check the existence of invariance set for more representative generators. Finally, we get all the representative generators to compose the one-dimensional optimal system.

3.2 *Generation of Adjoint Transformation Matrix*

We can get the adjoint representation table by using formula (2), then the adjoint transformation matrix A is obtained by calculating Eqs. (3) and (4). However, it remains a problem to do in this way. In formula (1), the degree of adjoint representation is infinite. So we will have difficulty in constructing an infinite series expansion through symbolic computation. In this situation, constructing a finite estimated series expansion is commonly to be done, but it definitely makes the result imprecise.

Here we proposed a new algorithm to solve the problem. Without calculating formula (2) directly, we extract every element of the commutator table to construct A_i. Take A_1 for example, it mainly depends on line 1 of the commutator table (line i is for A_i). Suppose a vector $V = [a_1 v_{m+1}, \ldots, a_n v_{m+n}]$ in line 1 and a matrix A_{11}. Then in A_{11}, we can get $A_{11}[i][j] = a_n$, where $i = n$, $j = m + n$. After making exponential calculation of A_{11}, we will obtain A_1. In software package, function $e_$index: = proc (ct0::Matrix) realizes this process.

Here is an example for KdV equation. Table 1 is the commutator table and Eq. (11) presents A_i, $i = 1, \ldots, 4$.

Table 1 The commutator table of KdV equation.

	v_1	v_2	v_3	v_4
v_1	0	0	0	v_1
v_2	0	0	v_1	$3v_2$
v_3	0	$-v_1$	0	$-2v_3$
v_4	$-v_1$	$-3v_2$	$2v_3$	v_1

Compared with the old method, the new method is more sufficient and it is much easier for us to realize the calculating process through symbolic computation. Above all, the result is more accuracy than before.

$$\boldsymbol{A}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 \end{pmatrix}, \quad \boldsymbol{A}_4 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{11}$$

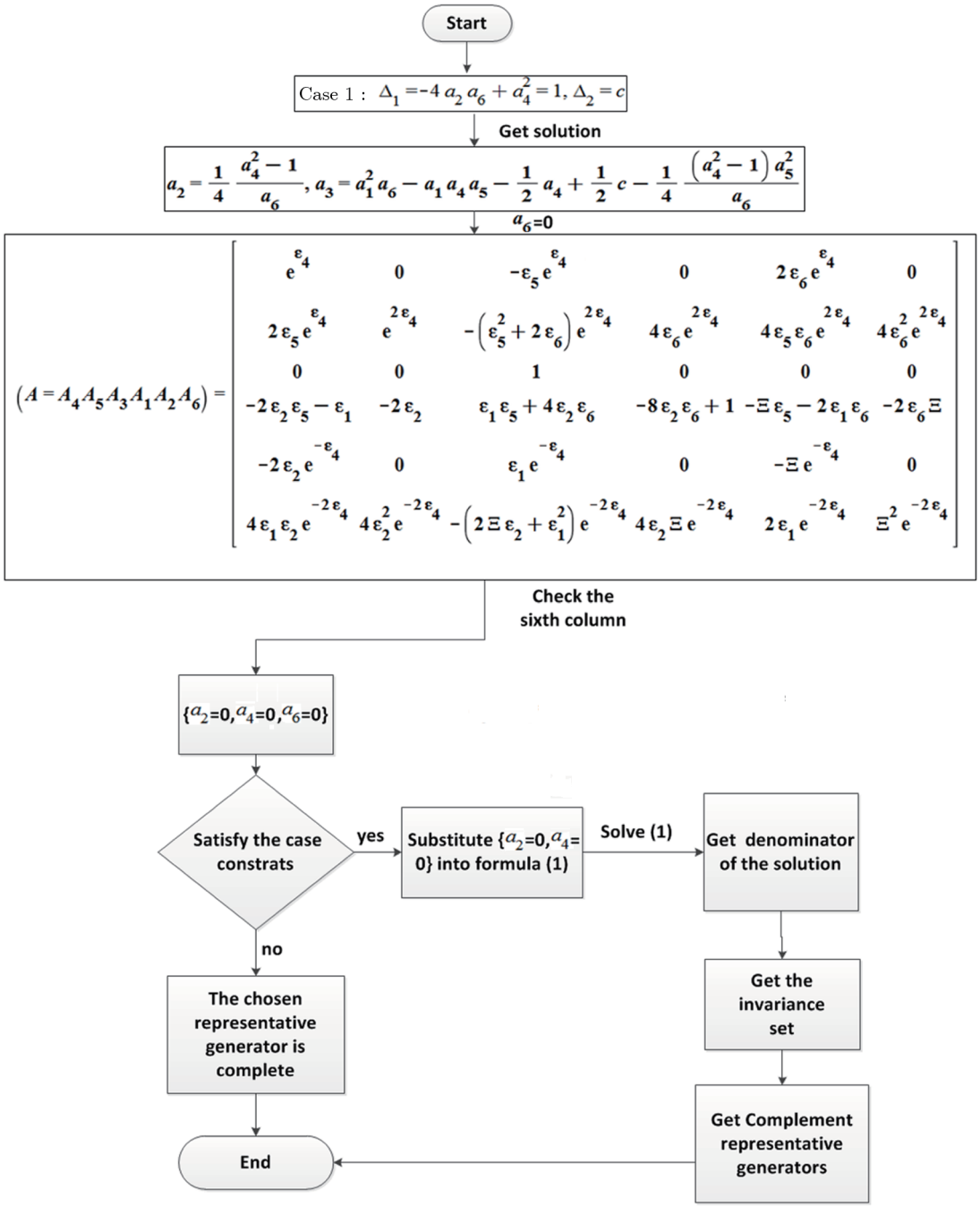

Fig. 2 Software flow diagram of situation 2.

Functions implementing this part as well as some important parameters are in Table 2.

Table 2 Functions and parameters of the *One Optimal System*.

Function && parameters	Description
$e_$index: = proc(ct0::Matrix)	Generate adjoint transformation matrix A
A	Adjoint transformation matrix, Global parameter
$ct0$	Commutater table
$aL[i]$	Array for $A_i, i = 1, \ldots, n$, the product of which is adjoint transformation matrix A

3.3 *Classification of One Optimal System*

After we get all the invariants, we can classify each case of one-dimensional optimal system according to classification

rules (details in Table 3).

Table 3 Different cases of classification for an optimal system.

Case	Invariant_1	Invariant_2	Invariant_3	Invariant_4	Current set	Check degree
1	1	c	\	\	Invariant_1 Invariant_2	√
2	−1	c	\	\	Invariant_1 Invariant_2	√
3	0	1	c	c	Invariant_1 Invariant_2	√
4	0	−1	\	\	Invariant_1 Invariant_2	×
5	0	0	1	\	Invariant_1 Invariant_2 Invariant_3	√
6	0	0	-1	\	Invariant_1 Invariant_2 Invariant_3	×
7	0	0	0	\	Invariant_1 Invariant_2 Invariant_3	√

Remark

(i) We assume invariant_1 and invariant_2 are origin invariants and initialize a current invariant set to put them in. For every cases, we assign exact value to the invariants in the current invariants set according to classification rules.

(ii) When there is no invariant assigned c , we substitute the case constraints into the differential equation to find if new invariants (assuming invariant_3 and invariant_4 represents new invariants in each case) exists. If all the invariants in the current set are not assigned 0, we should make all new invariants c. On the other hand, if all the invariants in the current set are assigned 0, we can add the new invariants into the current invariants set and assign exact value to them according to classification rules.

(iii) What we should notice is the new invariants are not constant if they are not added to the current set. The new invariants' expression depend on the differential equation, which are substituted different case constraints.

(iv) The last column of Table 3 are used to check the degree of invariants.

Functions implementing this part as well as some important parameters are described in Table 4.

Table 4 Functions and parameters of the *One Optimal System*.

Function && Parameters	Description
fun_set : = proc(data0)	Give a classification set of all cases
n_deta : = proc(fun_now)	Find new invariants
aaa_solve : = proc(fun)	Solve case constraints
fun_s	A classification set of all cases
fun_now	Represents differents cases
fun_n	A set that current invariants are assigned 0
fun_h	New invariants set
data0	Origin invariants set
data	Current invariants set
flag	Shows the existance of any new invariants. It will be assigned 0 if there is no new invariant
deta_new	New invariants
as	Solution of a case constraint
n	Dimension of Lie algebra, Global parameter
Eq0	N linear differential equation about $\phi(a_1, \ldots, a_n)$, Global parameter

3.4 *Find the Representative Generator*

(i) Find a representative generator

For each case, we must choose a representative generator and it is better to be as simple as possible. For this intention, we solve case constraint firstly. Then we assume each a_i showing in the numerator of the solution to be 0, except the one showing up in the denominator as well. And the one in the denominator should be assigned either 1 or −1, determined by another constraint (expressions like $\ln(a_i), \sqrt{a_i}, \sqrt{-a_i}, \ldots$) in the solution. This method can make sure of choosing more simple representative generators.

(ii) Check the representative generator

By solving Eq. (5), we can determine if the chosen representative generator is appropriate for this case. If there is a solution exists, we should also judge if there are some constraints as follows in the solution.

(a) Exists expressions like $\ln(a_i), \sqrt{a_i}, \sqrt{-a_i}, \ldots$

(b) Exists denominator a_i.

If so, this will make the chosen representative generator not complete. That means it can not represent the whole generators of this case. Then we should find another representative generator to complement the chosen one.

Situation 1 For the first situation, we give two examples in the following Table 5.

Table 5 Two examples of the first situation.

Case 1	
Case constraints	$\Delta_1 = -\frac{1}{4}a_4^2 + a_2a_6 = 1, \quad \Delta_2 = -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = c$
Solution_1	$a_2 = \frac{a_4^2+4}{4a_6}, \quad a_3 = -\frac{8a_4a_6+a_4^2a_5^2+4a_5^2+4a_6^2a_1^2-4a_6a_4a_1a_5+2ca_6}{16a_6}$
Generator_1	$v_2 - \frac{1}{8}cv_3 + v_6(v_2 - cv_3 + v_6)$
Solution_2	$\varepsilon_1 = -\frac{1}{8}\frac{-a_4^2a_5+2a_4a_1a_6-4a_5}{\sqrt{a_6}}, \quad \varepsilon_2 = \frac{1}{4}a_4, \quad \varepsilon_4 = \frac{1}{2}\ln(a_6), \quad \varepsilon_5 = -\frac{1}{4}\frac{2a_1a_6-a_4a_5}{\sqrt{a_6}}, \quad \varepsilon_6 = 0$
Check completion	Not completed($a_6 > 0$)
Generator_2	$-v_2 + \frac{1}{8}cv_3 - v_6$
Case 2	
Case constraints	$\Delta_1 = -\frac{1}{4}a_4^2 + a_2a_6 = 0, \quad \Delta_2 = -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = 1$
Solution_1	$a_1 = \frac{1}{2}\frac{a_4a_5+\sqrt{-2a_6}}{a_6}, \quad a_2 = \frac{1}{4}\frac{a_4^2}{a_6}$
Generator_1	$-\frac{1}{2}\sqrt{2}v_1 - v_6$
Solution_2	$\varepsilon_1 = \frac{1}{8}\frac{\sqrt{2}(2a_5\sqrt{2}\sqrt{-a_6}+4a_6\varepsilon_6+2a_4^2a_6+4a_4a_3a_6+2a_4a_6\varepsilon_6^2+a_4a_5^2)}{a_6}, \quad \varepsilon_2 = -\frac{1}{4}a_4, \quad \varepsilon_4 = \frac{1}{2}\ln(-a_6),$ $\varepsilon_5 = -\frac{1}{4}\frac{\sqrt{2}(2a_4a_6+4a_3a_6+2a_6\varepsilon_6^2+a_5^2)}{a_6}$
Check completion	Completed
Generator_2	\

Remark

(i) If the same constraint also exists in the case constraints, we do not need to find another generator as the constraints exist originally in this case.

(ii) Solution_1 is for the case constraint and solution_2 is for Eq. (5), generator_1 is the origin generator and generator_2 is the one to make complement.

Situation 2

For this situation, it is related to the invariance set and we have a check method in Fig. 2. (Remark: $\Delta_1 = -(1/4)a_4^2 + a_2a_6 = 1$, $\Delta_2 = -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = c$).

Table 6 Functions and parameters of the *One Optimal System*.

Function && Parameters	Description
Ep_solve := proc(fun)	Give all the representative generators of one optimal system
Special_e := proc(aaa, spe_2)	Find the representative generator of each case
Excep_vv := proc(aaa,A,fun)	Find whether a invariance set exists like situation 1
Judge_s := proc(solution,aaa,L,fun)	Check situation 2 and give the complement gnerator
Trans_ornot := proc(g_f,A,aaa)	Check all of the candidate representative generators to see whether they
	Can be equal or not. If so, vanish the equal one from generator_f set
aaa_solve := proc(fun)	Solve the case constraints
Get_solution := proc(fa)	Solve Eq. (1)
Beauty_c := proc(g_f)	Merge some comparable genereators by using c
Gen	Represents every new representative generator, Global parameter
Generator_f	Set of representative generator from one case, Global parameter
Generator_final	Set of all final representative generators, Global parameter
Fun	Classification set of all cases
Solution	Solution of Eq. (1)
fa	Final form of Eq. (1)
aaa	Solution of case constraints
a_set	The invariance set

If we get the invariance set, we can classify two cases as follows:

(i) Make all the variants 0 in the invariance set.

(ii) Make not all the variants 0 in the invariance set.

Then we can get two new representative generators. Through the two steps above, we can find the representative generator in every cases, but there is still a special situation we should also take a consideration. Take KdV equation for example. When we find the representative generators under the case constraints $\{a_4 = 0, a_2^2 a_3^3 = 0\}$, we can list v_1, v_2, v_3 as candidate representative generators corresponding to $\{a_4 = 0, a_2 = 0, a_3 = 0\}$, $\{a_4 = 0, a_3 = 0, a_2 \neq 0\}$, $\{a_4 = 0, a_2 = 0, a_3 \neq 0\}$ respectively. From the adjoint matrix of KdV, we can see from the adjoint transformation matrix (14) when the generator satisfies $\{a_4 = 0, a_2 = 0, a_3 = 0\}$, the second and third column of the matrix are 0 constantly by computing Eq. (5). So for this situation, we should check all of the candidate representative generators to see whether they are equal or not. Finally, we should list all the unequal ones to complement,

$$\begin{pmatrix} \mathrm{e}^{\varepsilon_4} & 0 & 0 & 0 \\ \varepsilon_3\,\mathrm{e}^{\varepsilon_4} & \mathrm{e}^{3\varepsilon_4} & 0 & 0 \\ -\varepsilon_2\,\mathrm{e}^{\varepsilon_4} & 0 & \mathrm{e}^{-2\varepsilon_4} & 0 \\ (-\varepsilon_1 - 3\varepsilon_2\varepsilon_3)\,\mathrm{e}^{\varepsilon_4} & -3\varepsilon_2\,\mathrm{e}^{3\varepsilon_4} & 2\varepsilon_3\,\mathrm{e}^{-2\varepsilon_4} & 1 \end{pmatrix}. \quad (12)$$

Functions implementing this part as well as some important parameters are described as in Table 6.

4 Examples

According to the previous software package *One Optimal*,[33] which generates one-dimensional optimal system of finite dimensional Lie Algebra, the result has to adjust the coefficients containing ε, which is uncertainty to give a complete classification. This uncertainty makes problems that some of the representative generators of the one-dimensional optimal system are absent or belong to the same case. The new software package *One Optimal System* can overcome this problem by giving a direct classification, whose answer is equal to Olver'a result. In this chapter, we apply it to many important equations to check its effectiveness and correctness.

4.1 *KdV (Korteweg-de Vries) Equation*

$$u_t + 6uu_x + u_{xxx} = 0\,, \quad (13)$$

which is generated by the vector fields

$$v_1 = \frac{\partial}{\partial x}\,, \quad v_2 = \frac{\partial}{\partial t}\,, \quad v_3 = t\frac{\partial}{\partial x} + \frac{\partial}{\partial u}\,,$$
$$v_4 = x\frac{\partial}{\partial x} + 3t\frac{\partial}{\partial t} - 2u\frac{\partial}{\partial u}\,. \quad (14)$$

—Inputs:

> with (One Optimal System):

> alias($\phi = \phi(x, t, u)$):

> kdv : =[diff(ϕ, x), diff(ϕ, t), t*diff(ϕ, x)+diff(ϕ, u), x*diff(ϕ, x)+3*t*diff(ϕ, t) -2*u*diff(ϕ, u)]:

> One Optimal (kdv, {});

—Outputs:

The commutator table is as Table 7

Table 7 The commutator table of KdV equation.

	v_1	v_2	v_3	v_4
v_1	0	0	0	v_1
v_2	0	0	v_1	$3v_2$
v_3	0	$-v_1$	0	$-2v_3$
v_4	$-v_1$	$-3v_2$	$2v_3$	v_1

The origin invariants are: $[a_4]$.

The cases discussed: $[[a_4 = 1], [a_4 = 0, a_3 a_2^{2/3} = 1], [a_4 = 0, a_3 a_2^{2/3} = 0]]$.

The Optimal System is: $\{v_1, v_2, v_3, v_4, -v_2 + v_3, v_2 + v_3\}$.

According to Olver,[22] the one-dimensional optimal system of KdV Equation is

$$\text{(a)}\ \ v_4 \quad (\mathrm{b}_1)\ \ v_3 + v_2 \quad (\mathrm{b}_2)\ \ v_3 - v_2$$
$$(\mathrm{b}_3)\ \ v_3 \quad \text{(c)}\ \ v_2 \quad (d)v_1\,. \quad (15)$$

The result ($\{v_i | i = 1, \ldots, 4\}$ represents the same vector in Table 7 is just the same as our answer.

4.2 *Heat Equation*

$$u_t = u_{xx}\,, \quad (16)$$

which is generated by the vector fields

$$v_1 = \frac{\partial}{\partial x}\,, \quad v_2 = \frac{\partial}{\partial t}\,, \quad v_3 = u\frac{\partial}{\partial u}\,,$$
$$v_4 = x\frac{\partial}{\partial x} + 2t\frac{\partial}{\partial t}\,, \quad v_5 = 2t\frac{\partial}{\partial x} - xu\frac{\partial}{\partial u}\,,$$
$$v_6 = 4tx\frac{\partial}{\partial x} + 4t^2\frac{\partial}{\partial t} - (x^2 + 2t)u\frac{\partial}{\partial u}\,. \quad (17)$$

—Inputs:

> with (*One Optimal System*):

> alias ($\phi = \phi(x, t, u)$) :

> heat : =[diff(ϕ, x), diff(ϕ, t), u*diff(ϕ, u), x*diff(ϕ, x) +2 * t* diff(ϕ, t), 2 * t*diff(ϕ, x)–x * u*diff(ϕ, u), *t * x*diff(ϕ, x) +4 * t^2*diff(ϕ, t)–(x^2 + 2 * t) * u*diff(ϕ, u)]:

> One Optimal (heat, {}).

—Outputs:

The commutator table is as Table 8

Table 8 The commutator table of heat equation.

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	0	0	v_1	$-v_3$	$2v_5$
v_2	0	0	0	$2v_2$	$2v_1$	$-2v_3 + 4v_4$
v_3	0	0	0	0	0	0
v_4	$-v_1$	$-2v_2$	0	0	v_5	$2v_6$
v_5	v_3	$-2v_1$	0	$-v_5$	0	0
v_6	$-2v_5$	$2v_3 - 4v_4$	0	$-2v_6$	0	0

The origin invariants are: $[-\frac{1}{4}a_4^2 + a_2a_6, -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1]$. The cases discussed: $[[-\frac{1}{4}a_4^2 + a_2a_6 = 1, -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = c], [-\frac{1}{4}a_4^2 + a_2a_6 = -1, -4a_2a_6a_4 + a_4^3 -$

$8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = c]$, $[-\frac{1}{4}a_4^2 + a_2a_6 = 0, -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = 1]$, $[-\frac{1}{4}a_4^2 + a_2a_6 = 0, -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = 0, \frac{2a_6a_4+4a_3a_6+a_5^2}{a_6} = 1]$, $[-\frac{1}{4}a_4^2 + a_2a_6 = 0, -4a_2a_6a_4 + a_4^3 - 8a_3a_2a_6 - 2a_2a_5^2 - 2a_1^2a_6 + 2a_3a_4^2 + 2a_4a_5a_1 = 0, \frac{2a_6a_4+4a_3a_6+a_5^2}{a_6} = 0]]$.
The Optimal System is: $\{v_1, v_3, v_6, \frac{1}{4}v_3 - v_6, \frac{1}{4}v_3 + v_6, -\frac{1}{2}\sqrt{2}v_1 - v_6, v_2 + cv_3 - v_6, v_2 + cv_3 + v_6\}$.

According to Olver,[22] the one-dimensional optimal system of Heat Equation is

$$\text{(a) } v_4 + av_3 \quad \text{(b) } v_2 + v_6 + av_3 \quad (c_1)\ v_2 - v_5 \quad (c_2)\ v_2 + v_5 \quad \text{(d) } v_2 + av_3 \quad \text{(e) } v_1 \quad (f)\ v_3\,. \tag{18}$$

We can see (a) (c_1) (c_2) (d) is different from our answer. However, we can get these equivalent relationship as follows through the adjoint representative matrix A action in formula (5) (manual prove see Eq. (19))

$$v_4 + av_3 \sim v_2 + cv_3 - v_6\,, \quad v_2 - v_5 \sim v_2 + v_5 \sim -\frac{1}{2}\sqrt{2}v_1 - v_6\,, \quad v_2 + av_3 \sim \frac{1}{4}v_3 - v_6 \sim \frac{1}{4}v_3 + v_6 \sim v_6\,. \tag{19}$$

Thus, we get the equivalent answer of Heat Equation to Olver.

4.3 *NS (Navier-Stokes) Equation*

$$\phi_{xxt} + \phi_{yyt} + \phi_x\phi_{xxy} + \phi_x\phi_{yyy} - \phi_y\phi_{xxx} - \phi_y\phi_{xyy} - \gamma(\phi_{xxxx} + 2\phi_{xxyy} + \phi_{yyyy}) = 0\,, \tag{20}$$

which is generated by the vector fields (we only choose four of its generators)

$$v_1 = \frac{x}{2}\frac{\partial}{\partial x} + \frac{y}{2}\frac{\partial}{\partial y} + t\frac{\partial}{\partial t}\,, \quad v_2 = \frac{\partial}{\partial t}\,, \quad v_3 = -yt\frac{\partial}{\partial x} + xt\frac{\partial}{\partial y} + \frac{x^2+y^2}{2}\frac{\partial}{\partial u}\,, \quad v_4 = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}\,. \tag{21}$$

—Inputs:
> with (One Optimal System):
> alias ($\phi = \phi(x, y, t, u)$):
> ns := [$x * \text{diff}(\phi, x)/2 + y * \text{diff}(\phi, y)/2 + t * \text{diff}(\phi, t), \text{diff}(\phi, t), -y{*}t{*}\text{diff}(\phi, x) + x{*}t{*}\text{diff}(\phi, y) + (x^2 + y^2) * \text{diff}(\phi, u)/2, -y * \text{diff}(\phi, x) + x * \text{diff}(\phi, y)$]:
> One Optimal (ns,{}).
—Outputs:
The commutator table is as Table 9

Table 9 The commutator table of NS equation.

	v_1	v_2	v_3	v_4
v_1	0	$-v_2$	v_3	0
v_2	v_2	0	v_4	0
v_3	$-v_3$	$-v_4$	0	0
v_4	0	0	0	0

The origin invariants are: $[a_1, -a_3a_2 + a_4a_1]$.
The cases discussed: $[[a_1 = 1, -a_3a_2 + a_4a_1 = c]$, $[a_1 = 0, -a_3a_2 + a_4a_1 = 1]$, $[a_1 = 0, -a_3a_2 + a_4a_1 = -1]$, $[a_1 = 0, -a_3a_2 + a_4a_1 = 1] = 0]]$. The Optimal System is: $\{v_2, v_3, v_4, v_1 + cv_4, -v_2 - v_3, v_2 - v_3\}$.

4.4 *ZK (Zakharove-Kuznetsov) Equation*

$$u_t + auu_{xx} + bu_{xxx} + u_{xyy} + u_{xzz} = 0\,, \tag{22}$$

which is generated by the vector fields

$$v_1 = \frac{\partial}{\partial x},\ v_2 = \frac{\partial}{\partial y}\,,\ \ v_3 = \frac{\partial}{\partial t}\,,\ \ v_4 = \frac{\partial}{\partial z}\,,$$
$$v_5 = z\frac{\partial}{\partial y} - y\frac{\partial}{\partial z}\,,\ \ v_6 = at\frac{\partial}{\partial x} + \frac{\partial}{\partial u}\,,$$
$$v_7 = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + z\frac{\partial}{\partial z} + 3t\frac{\partial}{\partial t} - 2u\frac{\partial}{\partial u}\,. \tag{23}$$

—Inputs:
> With (One Optimal System):
> Alias($\phi = \phi(x, y, z, t, u)$):
> zk := [$\text{diff}(\phi, x)$, $\text{diff}(\phi, y)$, $\text{diff}(\phi, t)$, $\text{diff}(\phi, z)$, $z * \text{diff}(\phi, y) - y{*}\text{diff}(\phi, z)$, $a * t{*}\text{diff}(\phi, x) + \text{diff}(\phi, u)$, $x{*}\text{diff}(\phi, x) + y{*}\text{diff}(\phi, y) + z{*}\text{diff}(\phi, z) + 3 * t{*}\text{diff}(\phi, t) - 2 * u{*}\text{diff}(\phi, u)$]:
> One Optimal (zk, {a}).
—Outputs:
The commutator table is as Table 10.

Table 10 The commutator table of ZK equation.

	0	0	0	0	0	0	v_1
v_1	0	0	0	0	$-v_4$	0	v_2
v_2	0	0	0	0	0	av_1	$3v_3$
v_3	0	0	0	0	v_2	0	v_4
v_4	0	v_4	0	$-v_2$	0	0	0
v_5	0	0	$-av_1$	0	0	0	$-2v_6$
v_6	$-v_1$	$-v_2$	$-3v_3$	$-v_4$	0	$2v_6$	0

The origin invariants are: $[a_5, a_7]$
The cases discussed: $[[a_5 = 1, a_7 = c], [a_5 = 0, a_7 = 1]$, $[a_5 = 0, a_7 = 0, a_6a_3^{2/3} = 1, (a_4^2 + a_2^2)/a_3(2/3) = c]$, $[a_5 = 0, a_7 = 0, a_6a_3^{2/3} = 0, (a_4^2 + a_2^2)/a_3(2/3) = 1]$, $[a_5 = 0, a_7 = 0, a_6a_3^{2/3} = 0, (a_4^2 + a_2^2)/a_3(2/3) = 0]]$.
The Optimal System is: $\{v_7, v_3 + v_2, v_5 + cv_7, v_2 + c^{3/2}v_3 + cv_6\}$.

5 Conclusions

Group invariant solutions have been used to describe general solutions to PDE systems. They can be illustrated by the invariance of PDE symmetry group. We need to find group invariant solutions due to the existence of infinite different symmetry groups. To classify invariant solutions in Lie algebra, we propose a new *Maple* package called *One Optimal System*. This has a significant mean-

ing in the mathematics mechanization and machine learning. The mechanized realization efficiently make complicated manual computation much more simple. Especially, to many important partial differential equations, many researchers have paid substantial concentration to the theme. Compared with the previous software package,[33] this new software package is obviously more efficient than the previous one, which costs less time in operation. And we can get a perfect result that is precisely consistent with Olver's result. And the algorithm of the new software package is more direct and programming than the previous one without many manual settings. Furthermore, the algorithm structure can be used to r-parameter ($r \geq 2$) optimal systems for further developing, which is difficult for the previous one. In this article, firstly, we use a new algorithm to calculate the adjoint transformation matrix, which effectively makes the mechanized realization much easier. Secondly, we find all the origin invariants (new ones are found in the process of classification) of a Lie algebra in spite of Killing form, then the classification rules of assigning different values to invariants regularly ensure the completeness of one optimal system. Last but not least, we check constraints of the chosen representative generator in simplest form to avoid overlooking some special situations. Simultaneously, we propose an important conception of invariance set. The software package can be applied to not only single partial differential equations, but also to ODEs and systems of differential equations. The original $(1 + 1)$-dimensional system of differential equations can be reduced to inequivalent ODEs, as a result of the one-dimensional optimal system of the symmetry Lie algebra with corresponding group invariant solutions recovered.

We realize calculation process of the one-dimensional optimal system by developing *One Optimal System* software package, which is demonstrated effective, precise and systemic. This is proved to simplify the manual computation significantly and has made a contribution to mathematics mechanization. To achieve more pervasive mechanized application of this method, we will devote to developing new software package of r-parameter ($r \geq 2$) optimal systems in the future.

Acknowledgments

We would like to express our sincere thanks to S. Y. Lou, X. R. Hu, and Q. Miao for their valuable comments and suggestions.

References

[1] S. Lie, Arch. Math. **6** (1881) 328.
[2] J. Lin and H. Wang, Optics Commun. **298** (2013) 185.
[3] A. L. Guo and J. Lin, Commun. Theor. Phys. **57** (2012) 523.
[4] Y. K. Liu and B. Li, Chin. J. Phys. **54** (2016) 718.
[5] W. G. Cheng, B. Li, and Y. Chen, Commun. Nonl. Sci. Numer. Simu. **29** (2015) 198.
[6] X. R. Hu and Y. Chen, Appl. Math. Comput. **215** (2009) 1141.
[7] Z. Z. Dong, F. Huang, and Y. Chen, Z. Naturforsch. **66a** (2011) 75.
[8] X. R. Hu, Z. Z. Dong, F. Huang, and Y. Chen, Z. Naturforsch. **65a** (2010) 1.
[9] Z. Z. Dong, Y. Chen, D. X. Kong, and Z. G. Wang, Chin. Ann. Math. **33** (2012) 309.
[10] N. H. Ibragimov, *CRC Handbook of Lie Group Analysis of Differential Equations*, CRC Press, Boca Raton (1994).
[11] X. R. Hu, Y. Chen, and L. J. Qian, Commun. Theor. Phys. **55** (2011) 737.
[12] Z. Z. Dong and Y. Chen, Commun. Theor. Phys. **54** (2010) 389.
[13] F. Galas and E. W. Richter, Physica D **50** (1991) 297.
[14] L. Gagnon and P. Winternitz, J. Phys. A **22** (1989) 469.
[15] L. Gagnon, B. Grammaticos, A. Ramani, and P. Winternitz, J. Phys. A **22** (1989) 499.
[16] L. Gagnon, P. Winternitz, J. Phys. A **21** (1988) 1493.
[17] K. S. Chou, G. X. Li, and C. Z. Qu, J. Math. Anal. Appl. **261(2)** (2001) 741.
[18] X. R. Hu and Y. Chen, Commun. Theor. Phys. **52** (2009) 997.
[19] S. V. Coggeshall and J. Meyer-Ter-Vehn, J. Math. Phys. **33** (1992) 3585.
[20] J. C. Fuchs, J. Math. Phys. **32** (1991) 1703.
[21] L. V. Ovsiannikov, *Group Analysis of Differential Equations*, Academic, New York (1982).
[22] P. J. Olver, *Applications of Lie Groups to Differential Equations*, Springer, New York (1993).
[23] J. Patera, R. T. Sharp, P. Winternitz, and H. Zassenhaus, J. Math. Phys. **17** (1976) 986.
[24] L. Weisner, Canad. J. Math. Phys. **11** (1959) 141.
[25] A. K. Head, *Program BIGLIE for Lie Analysis of Differential Equations on IBM Type PCs*, Users Manual (2000).
[26] J. Carminati and K. Vu, J. Symbolic Comput. **29** (2000) 95.
[27] A. K. Head, *Program LIE for Lie Analysis of Differential Equations on IBM Type PCs*, User's Manual (2000).
[28] Baumann, *Symmetry Analysis of Differential Equations with Mathematica*, Springer, New York (2000).
[29] E. S. Cheb-Terrab, and K. von Bulow, Comp. Phys. Comm. **90** (1995) 116.
[30] K. T. Vu, J. Butcher, and J. Carminati, Comp. Phys. Comm. **176** (2007) 682.
[31] F. Schwarz, SIAM Rev. **30** (1988) 450.
[32] X. R. Hu, Y. Q. Li, and Y. Chen, J. Math. Phys. **56** (2015) 053504
[33] Q. Miao, X. R. Hu, and Y. Chen, Commun. Theor. Phys. **61** (2014) 160.