LA-UR-18-26586
Approved for public release; distribution is unlimited.

Title: SYMMETRY ANALYSIS USING SYMBOLIC COMPUTATION

Author(s): Albright, Eric Jason
McHardy, James David

Intended for: Report

Issued: 2018-07-17

SYMMETRY ANALYSIS USING SYMBOLIC COMPUTATION

E. JASON ALBRIGHT AND JAMES D. MCHARDY

Contents

Abstract. The symmetry group of a system of differential equations reveals basic invariant properties of the underlying model and its solutions. In many cases symmetry groups have simple geometric interpretations that can be applied to simplify the model and determine certain classes of solutions explicitly. The general procedure to compute the symmetry group of a given system of differential equations follows a well-known algorithm. However, this procedure involves a large number of laborious and usually error-prone calculations, especially in the case of higher-order systems. The purpose of this report is explain how to use an existing software program designed to automate the major steps of this algorithm in order to calculate the symmetry group of any system of differential equations more efficiently.

1. Background

The symmetry group of a system of differential equations has several important uses. These include classification of different solutions, generation of new solutions from existing solutions, and simplification of the original equations by the method of symmetry reduction. Additionally, for systems of equations describing an underlying physical system that possess certain symmetry properties, determination of the corresponding symmetry group can also be used as validation of a given model.

Symmetry analysis of differential equations dates back to Lie [6] and has been developed extensively by Ovsiannikov [8], Olver [7] and Bluman and Anco [4]. It is assumed that the reader is already familiar with symmetry analysis and has a basic understanding of its underlying mathematical justification. Otherwise, the reader may wish to consult [1] written in conjunction with the current report as an introduction to symmetry analysis. The majority of the examples treated in this report (using symbolic computation) are also worked out step-by-step in the aforementioned primer for comparison between the theoretical and the automated computational procedure.

In principle, the algorithm to determine the symmetry group of any differential equation can be carried out by hand. However, for larger systems of equations, especially involving higher order equations, the algorithm involves a large number of tedious calculations. Consequently, many computer algebra-assisted symmetry group software programs have been created to automate certain parts of the procedure. A more complete review of existing symmetry method software programs can be found in [2, Ch. 13]. Here, we will focus on an existing program called `symmgrp`

originally developed by Champagne, Hereman, and Winternitz [5]. In particular, their program has the advantage of extensive previous testing and has been widely cited in peer-reviewed literature. It is also freely available and is based on the open-source computer algebra system, Maxima. Additionally, its relative ease of use and overall flexibility were factors taken into consideration. We recommend consulting the original documentation for `symmgrp`, available in the aforementioned reference, in addition to this report. Here you will find a quick and updated demonstration of the capabilities of the software illustrated by some new examples pertaining to fluid motion and related applications.

2. Symmetry analysis

In this section, we briefly review the procedure to determine the symmetry group of any differential equation in order to introduce the notation required in later sections to explain how to use `symmgrp`. The symmetry group of any system of differential equations is the largest (local) Lie group of transformations acting on the space of independent and dependent variables that leaves the solutions to the system invariant. In other words, the symmetry group transforms solutions of the system into other solutions.

In general, a system of K-th order differential equations is denoted

$$F_n(x, u, p^{(1)}, p^{(2)}, \dots, p^{(K)}) = 0, \qquad n = 1, 2, \dots, N, \tag{1}$$

where F_n denotes a differential function of J independent and I dependent variables denoted

$$x = (x_1, x_2, \dots, x_J), \qquad u = (u^1, u^2, \dots, u^I), \tag{2}$$

and $p^{(k)}$ for $k = 1, 2, \dots, K$ denote vectors of all k-th order partial derivatives. The symmetry group of system (1) is determined by smooth, parametrized transformations of the form

$$\tilde{x} = \alpha(x, u; \epsilon), \qquad \tilde{u} = \beta(x, u; \epsilon), \tag{3}$$

such that the functions F_n are invariant with respect to α and β. The group of admitted transformations in (3) (parametrized by ϵ) is generated by an associated vector field of the form

$$V = \sum_{j=0}^{J} \eta_j(x, u) \frac{\partial}{\partial x_i} + \sum_{i=1}^{I} \phi^i(x, u) \frac{\partial}{\partial u_i} \tag{4}$$

where η_j and ϕ^i are called infinitesimals or coordinate functions, and correspond to the principal linear terms of the infinitesimal transformations of the group,

$$\tilde{x}_j = x_j + \epsilon \eta_j(x, u) + \mathcal{O}(\epsilon^2), \qquad \tilde{u}^i = u^i + \epsilon \phi^i(x, u) + \mathcal{O}(\epsilon^2). \tag{5}$$

Therefore, the objective of the procedure is then to find η_j and ϕ^i, which determine V, and hence generate the group defined by (3). The coordinate functions η_j and ϕ^i are determined by the condition for infinitesimal invariance given by

$$\mathrm{pr}^{(K)} V F_n \Big|_{F_n = 0} = 0, \qquad n = 1, 2, \dots, N, \tag{6}$$

which forms a system of equations whose solution defines the symmetry group of the system (1). The operator $\mathrm{pr}^{(k)} V$ in (6) denotes the K-th prolongation of the vector field V. A more detailed introduction to the general procedure, as well as the general formulas for the construction of the K-th prolongation, can be found in[1], as well as [5].

Therefore, the two major steps of the procedure are to construct the determining equations (6) and then to solve them for η_j and ϕ^i. In subsequent sections, we demonstrate how to automate both of these steps using symbolic computation with `symmgrp`.

3. Symbolic computation

One of the major advantages of `symmgrp` package is that it mirrors the procedure that would otherwise be carried out by hand, however, without the large number of laborious algebraic simplifications and substitutions, which are carried out automatically instead. Therefore, `symmgrp` can also aid in learning the underlying theoretical procedure with much less effort. There are several other advantages to using `symmgrp` (or any other automated procedure) to compute symmetries that are worth stating early on. First, the software can be used simply to check that existing calculations appearing in the literature for example, are in fact correct. Second, as described below, the individual files generated during the operation of `symmgrp` provide documentation for each successive step, which helps insure your work is reproducible and easy to return to later.

The program `symmgrp` is built upon the open-source computer algebra system, Maxima[1]. Like any computer algebra system, Maxima can be used to manipulate symbolic and numerical expressions including differentiation, integration, series expansion, systems of linear equations etc. In addition to installing Maxima, you will need to obtain a copy of the `symmgrp` package, which has been made publicly available and can be downloaded directly from the webpage of one of its developers hosted by Colorado School of Mines[2].

As mentioned previously, the operation of `symmgrp` follows the same two steps that make up the general theoretical procedure. Namely, first constructing the determining equations and second using a built-in feedback routine to solve them. Moreover, these two steps can be applied repeatedly to reduce the system of determining equations by supplying additional information about the η's and ϕ's into the program's built-in feedback mechanism at the end of each successive step. At each stage, the program automatically removes trivial factors, duplications, and differential redundancies from the system of determining equations. During the final stage, the number of determining equations is reduced to zero, which then verifies the solution (the η's and ϕ's). The program can also be configured to operate on any given subset of the full system of equations to allow for the possibility of treating arbitrarily large systems of any order.

In Sections 4–5 we demonstrate the basic operations of `symmgrp` in the case of two first-order ordinary differential equation and a first-order partial differential equation respectively. The analytic calculation of the symmetries admitted by both of these examples can be found in [1] for comparison. In Section 6, we compute the symmetry group admitted by a second-order, nonlinear partial differential equation to demonstrate the procedure for higher-order equations. Finally, in Section 7, we illustrate the use `symmgrp` to compute the symmetries of a system of equations using the Euler equations of gas dynamics. We show that `symmgrp` can be used to reproduce the full set of equivalence transformations admitted by the Euler equations, which is a result originally due to Ovsiannikov [8].

4. Example 1 - The Ramsey equation

For the first example we consider the following first-order, nonlinear ordinary differential equation

$$\frac{dy}{dx} = e^{-x}y^2 + y + e^x, \tag{7}$$

widely known as the Ramsey equation. The symmetries admitted by (7) were determined analytically in [1, Ch. 3, Sec. 2]. This example will illustrate the basic operations required to use `symmgrp` in the case of ordinary differential equations.

First we create a file that contains all of the problem specific inputs saved as `ramsey_ode_run1.dat` with the `.dat` file extension. It contains the following

[1]Documentation for general Maxima usage and syntax can be found at http://maxima.sourceforge.net/documentation.html

[2]The most recent `symmgrp` package is available here https://inside.mines.edu/~whereman/software/symmetry

```
1  p:1$
2  q:1$
3  m:1$
4  parameters:[]$
5  warnings:true$
6  sublisteqs:[all]$
7  info_given:false$
8  highest_derivatives:1$
9  e1:u[1,[1]]-exp(-x[1])*(u[1]^2)-u[1]-exp(x[1])$
10 v1: u[1,[1]]$
```

Line 1: `p` specifies the number of independent variables, which for (7) corresponds to $x_1 = x$.

Line 2: `q` specifies the number of dependent variables, which in this example corresponds to $u^1 = y$.

Line 3: `m` specifies the number of equations in the system, which corresponds to (7).

Line 4: defines a list of any constant parameters that may appear in the equation.

Line 5: switches warnings messages on or off according to the values `true` or `false` respectively.

Line 6: specifies a list of equations in the form `e1`, `e2`, ..., or `all` that will be included in the calculation. In practice, this feature can be used to specify a special subset of equations that belong to a larger system in order to decompose the problem into smaller subsets. However, for modest systems of only a few equations this feature is rarely needed.

Line 7: specifies that feedback inputs are provided according to the values `true` or `false`. We will explain how to use this feature in order to provide feedback for successive runs in order to simplify and eventually solve the resulting system of determining equations. For the initial run, its value is always set to `false`.

Line 8: specifies the the highest order derivatives k to be included in the prolongation $\mathrm{pr}^{(k)}$, such as `k=1`, `k=2`, ..., or `k=all`. We set this to `all` by default in subsequent examples.

Line 9: specifies the equation that defines the system, the first equation is always labeled `e1`. Additional equations can be included to treat systems and are labeled `e2`, `e3`, etc.

The following describes the required syntax to specify equations. Independent variables are accessed using `x[j]` where the index `j` specifies their position in the list of independent variables `x`. All dependent variables are accessed in a similar way using `u[i]` where `i` is the list position of the i-th dependent variable. Derivatives of the dependent variables are specified by including additional arguments in nested brackets. For ordinary differential equations

$$\frac{\mathrm{d}u}{\mathrm{d}x} = u[1,[1]]. \tag{8}$$

The integer value in brackets is also used to specify the order of the derivative.

$$\frac{\mathrm{d}^2u}{\mathrm{d}x_1^2} = u[1,[2]]. \tag{9}$$

Similarly, partial derivatives are indicated by using the position of the additional arguments to specify different coordinate direction of the derivatives. For example,

$$\frac{\partial u^1}{\partial x_2} = u[1,[0,1]], \qquad \frac{\partial^2 u^2}{\partial x_1^2} = u[2,[2,0]]. \tag{10}$$

Line 10: specifies the free variables for the subsequent substitution steps that result from evaluating the determining equation under the provision that the differential equation is satisfied. In the case of the Ramsey equation we select

$$\frac{\mathrm{d}y}{\mathrm{d}x} = u[1,[1]], \tag{11}$$

and consequently the substitution $\mathrm{d}y/\mathrm{d}x = e^{-x}y^2 + y + e^x$ will be made during the automated simplification steps in Maxima.

The next required file is a batch script, which calls the `symmgrp` Maxima script and the initial inputs for the problem from `ramsey_ode_run1.dat`. This batch script is saved as `ramsey_run1.mac` in this example, and consists of

```
1 batchload("~/symmgrp.max");
2 writefile("~/RamseyODE_Maxima/ramsey_ode_run1.412");
3 batch("~/RamseyODE_Maxima/ramsey_ode_run1.dat");
4 symmetry(1,0,0);
5 printeqn(lode);
6 save ("~/RamseyODE_Maxima/loderamseyode.lsp", lode);
7 closefile();
8 quit();
```

Line 1: loads the Maxima `symmgrp.max` script into the Maxima session using the batchload command under the provision of the script file path.

Line 2: specifies the path to the output `.412` file that we have labelled `ramsey_ode_run1.412`.

Line 3: specifies the path to the `.dat` file `ramsey_ode_run1.dat`.

Line 4: specifies extra run mode options of the `symmetry()` function defined in `symmgrp.max`. The first argument dictates whether to run in interactive or batch mode by `0` or `1` respectively. The second argument dictates whether to clear the array of the coefficients of the prolongation created on the initial run or to store and update the existing array when necessary by `0` or `1` respectively. The final argument suppresses or outputs a trace of the calculations performed by `0` or `1` respectively.

Line 5: prints the list of differential equations called `lode` to the Maxima session.

Line 6: saves the session output to a newly created `.lsp` file.

Recall that in the case of a first-order differential equation, relabeling $x_1 = x$ and $u^1 = y$, the symmetries are determined by infinitesimal transformations of the form

$$\tilde{x}_1 = x_1 + \epsilon\eta_1(x_1,u^1) + \mathcal{O}(\epsilon^2), \qquad \tilde{u}^1 = u^1 + \epsilon\phi^1(x_1,u^1) + \mathcal{O}(\epsilon^2). \tag{12}$$

Hence, our objective below is to determine η_1 and ϕ^1. This is always accomplished in two major steps. First, we calculate the determining equations resulting from condition (6). Second, we apply the feedback mechanism, which we will explain below, to simplify and reduce the determining equations to find η_1 and ϕ^1 explicitly.

The output of the first run in `ramsey_ode_run1.dat` produces the determining equation for η_1 and ϕ^1 in (12), in the form

$$\frac{\partial \phi^1}{\partial x_1}\left[e^{3x_1}+e^{2x_1}(1+2u^1)+e^{x_1}(u^1)^2\right]-e^{3x_1}\eta_1-e^{2x_1}\phi^1+e^{x_1}\left(\eta_1(u^1)^2-2\phi^1u^1\right)$$
$$-\frac{\partial \eta_1}{\partial x_1}\left[e^{4x_1}+e^{3x_1}(1+2u^1)+e^{2x_1}(u^1+3(u^1)^2)+e^{x_1}((u^1)^2+2(u^1)^3)-(u^1)^4\right]=0. \tag{13}$$

As we observed in [1, Ch. 3, Sec. 2], first-order ordinary differential equations always admit an infinite number of symmetries[3]. Therefore, below we will illustrates how to isolate particular symmetries using the feedback mechanism in `symmgrp`.

We start with the following simplifying assumptions regarding η_1

$$\frac{\partial \eta_1}{\partial x_1}=0, \qquad \frac{\partial \eta_1}{\partial u^1}=0. \tag{14}$$

In other words, η_1 is an undetermined constant, which we will denote t. Additionally, we assume that the infinitesimal ϕ^1 is independent of the variable x_1. This information can then supplied as additional input for the next run, which will invoke the feedback mechanism to simplify the determining equations mentioned previously.

The second run is comprised of new `.mac` and `.dat` files with the following additions to the previous run. In the `.mac` file, now labelled `ramsey_run2.mac`, the file paths are changed appropriately to ensure the updated calculation `ramsey_run2` is carried out

```
writefile("~/RamseyODE_Maxima/ramsey_run2.412");
batch("~/RamseyODE_Maxima/ramsey_run2.dat");
```

Creating new files for each run ensures that at the end of the procedure you also have a step-by-step record of the calculation leading to the final solution. In the `ramsey_run2.dat` file the following lines are added and `info_given:` is set to `true`

```
depends(phi1,u[1])$
eta1:t$
```

The result is a much simpler determining equation

$$\left(\frac{\partial \phi^1}{\partial u^1}-\eta_1\right)e^{2x_1}+\left(\frac{\partial \phi^1}{\partial u^1}u^1-\phi^1\right)e^{x_1}+\frac{\partial \phi^1}{\partial u^1}(u^1)^2+\eta_1(u^1)^2-2\phi^1u^1=0, \tag{15}$$

which is a quadratic polynomial in e^x. This equation is satisfied if and only if each of the coefficients are zero. This yields three new equations

$$\frac{\partial \phi^1}{\partial u^1}-\eta_1=0, \tag{16}$$
$$\frac{\partial \phi^1}{\partial u^1}u^1-\phi^1=0, \tag{17}$$
$$\frac{\partial \phi^1}{\partial u^1}(u^1)^2+\eta_1(u^1)^2-2\phi^1u^1=0. \tag{18}$$

Substituting (16) into (18) yields

$$\phi^1=u^1\eta_1, \tag{19}$$

[3]A proof of this fact can be found in in [9, Chap. 4, p.27])

which satisfies the substitution of (16) into (17). One solution is therefore $\eta_1 = 1$ and $\phi^1 = u^1$. The infinitesimal generator obtained in terms of x and y is therefore

$$V = \frac{\partial}{\partial x} + y\frac{\partial}{\partial y}, \tag{20}$$

which is a combination of a scaling and a translation transformation. In [1, Ch. 3, Sec. 2], we used this symmetry to solve (7) by the method of symmetry reduction.

5. Example 2 - Inviscid Burgers's equation

The next example illustrates how to compute and solve the determining equations of the symmetry group corresponding to a first-order, nonlinear partial differential equation. In contrast to the previous example, this example only admits a finite number of symmetries, and hence we will compute its entire symmetry group.

We consider the inviscid Burgers's equation in the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0. \tag{21}$$

We determined the symmetry group for inviscid Burgers's equation in [1, Ch. 4, Sec. 1], which the reader may wish to consult for a step-by-step comparison against the approach with `symmgrp` below. Similarly to the previous example, recall the symmetries are determined by infinitesimal transformations of the form

$$\tilde{x}_1 = x_1 + \epsilon\eta_1(x,y) + \mathcal{O}(\epsilon^2), \qquad \tilde{x}_2 = x_2 + \epsilon\eta_2(x,y), \qquad \tilde{u}^1 = u^1 + \epsilon\phi^1(x,y) + \mathcal{O}(\epsilon^2), \tag{22}$$

where $x_1 = t$, $x_2 = x$, and $u^1 = u$. Hence, our objective below is again to determine the η's and ϕ's from the determining equations resulting from the condition for infinitesimal invariance (6).

As in the previous section, we start by constructing the file `inviscid_burgers_run1.mac` containing

```
batchload("~/symmgrp.max");
writefile("~/Inviscid_Burgers_Eqn/inviscid_burgers_run1.412");
batch("~/Inviscid_Burgers_Eqn/inviscid_burgers_run1.dat");
symmetry(1,0,0);
printeqn(lode);
save ("~/Inviscid_Burgers_Eqn/lodeburgers.lsp", lode);
closefile();
quit();
```

and the input file `inviscid_burgers_run1.dat` consisting of

```
p:2$
q:1$
m:1$
parameters:[]$
warnings:true$
sublisteqs:[all]$
info_given:false$
highest_derivatives:all$
e1:u[1,[1,0]]+u[1]*u[1,[0,1]]$
v1: u[1,[1,0]]$
```

From the first run, we then obtain the determining equations for the symmetries of the inviscid Burgers's equation (21) given by

$$\frac{\partial \phi^1}{\partial x_1} + (u^1)\frac{\partial \phi^1}{\partial x_2} = 0, \tag{23}$$

$$\phi^1 + (u^1)^2\frac{\partial \eta_1}{\partial x_2} + u^1\left(\frac{\partial \eta_2}{\partial x_2} - \frac{\partial \eta_1}{\partial x_1}\right) - \frac{\partial \eta_2}{\partial x_1} = 0. \tag{24}$$

Note this set of equations is identical to those obtained analytically in [1, Ch. 4, Sec. 1]. From (24), we infer that ϕ^1 is at most a quadratic polynomial in terms of u^1 provided we additionally assume that its coefficients have no dependence on u^1. Therefore, we include

$$\eta_1 := \eta_1(x_1, x_2), \qquad \eta_2 := \eta_2(x_1, x_2), \tag{25}$$

and express ϕ^1 as

$$\phi^1 = -(u^1)^2\frac{\partial \eta_1}{\partial x_2} + u^1\left(\frac{\partial \eta_2}{\partial x_2} - \frac{\partial \eta_1}{\partial x_1}\right) + \frac{\partial \eta_2}{\partial x_1}, \tag{26}$$

in the `inviscid_burgers_run2.dat` file.

```
7  info_given:true$
8  highest_derivatives:all$
9  depends(eta1,[x[1],x[2]])
10 depends(eta2,[x[1],x[2]])$
11 phi1:-u[1]^2*diff(eta1,x[2])+u[1]*(diff(eta2,x[2])-diff(eta1,x[1]))+
      diff(eta2,x[1])$
```

The second run reduces the set of determining equations (23)–(24) to

$$(u^1)^2\left(\frac{\partial^2 \eta_2}{\partial x_2^2}\right) + \frac{\partial^2 \eta_2}{\partial x_1^2} + 2u^1\left(\frac{\partial^2 \eta_2}{\partial x_1 \partial x_2}\right) - (u^1)^3\left(\frac{\partial^2 \eta_1}{\partial x_2^2}\right) - u^1\left(\frac{\partial^2 \eta_1}{\partial x_1^2}\right) - 2(u^1)^2\left(\frac{\partial^2 \eta_1}{\partial x_1 \partial x_2}\right) = 0. \tag{27}$$

Due to the previous assumptions in (25) on η_1 and η_2, the left-hand side of the previous equation is again just a polynomial in terms of u^1. Hence, for the previous equation to be satisfied for all smooth solutions u^1 of the original equation (21), the coefficients of the terms on the left-hand side must be identically zero. This conclusion yields the following reduced system of equations from (27)

$$\frac{\partial^2 \eta_1}{\partial x_2^2} = 0, \tag{28}$$

$$\frac{\partial^2 \eta_2}{\partial x_1^2} = 0, \tag{29}$$

$$\frac{\partial^2 \eta_2}{\partial x_2^2} - 2\frac{\partial^2 \eta_1}{\partial x_1 \partial x_2} = 0, \tag{30}$$

$$\frac{\partial^2 \eta_1}{\partial x_1^2} - 2\frac{\partial^2 \eta_2}{\partial x_1 \partial x_2} = 0, \tag{31}$$

which defines an overdetermined system of linear equations for the remaining inifinitesimals η_1 and η_2. We solved this system previously in [1, Ch. 4, Sec. 1], we state the result below

$$\eta_1 = (a_1 x_1 + a_2)\, x_2 + a_3 x_1^2 + a_4 x_1 + a_5, \tag{32}$$

$$\eta_2 = (a_3 x_2 + a_6)\, x_1 + a_1 x_2^2 + a_7 x_2 + a_8, \tag{33}$$

where $a_1, a_2, \ldots, a_8$ are arbitrary constants.

Now that we have determined η_1 and η_2, we can evaluate the remaining infinitesimal ϕ^1 in the form of (26), which yields

$$\phi^1 = -(a_1x_1 + a_2)(u^1)^2 + (a_1x_2 - a_3x_1 + a_7 - a_4)u^1 + a_3x_2 + a_6. \tag{34}$$

As a final step, we can utilize `symgrp` to verify the solution in (32)–(34). We include the following in the `.dat` file for the third and final run

```
highest_derivatives:all$
eta1:(a1*x[1]+a2)*x[2]+a3*x[1]^2+a4*x[1]+a5$
eta2:(a3*x[2]+a6)*x[1]+a1*x[2]^2+a7*x[2]+a8$
phi1:-(a1*x[1]+a2)*u[1]^2+(a1*x[2]-a3*x[1]+a7-a4)*u[1]+a3*x[2]+a6$
```

The result produced by this run states there are zero remaining determining equations, from which we can conclude that the infinitesimals (32)–(34) are admitted by inviscid Burgers's equation. In terms of the original variables, t, x, and u, the corresponding infinitesimal generator of the corresponding eight-parameter Lie group is given by

$$\begin{aligned} V = &\left[(a_1t + a_2)\,x + a_3t^2 + a_4t + a_5\right]\frac{\partial}{\partial t} \\ &+ \left[(a_3x + a_6)\,t + a_1x^2 + a_7x + a_8\right]\frac{\partial}{\partial x} \\ &+ \left[-(a_1t + a_2)u^2 + (a_1x - a_3t + a_7 - a_4)u + a_3x + a_6\right]\frac{\partial}{\partial u}, \end{aligned} \tag{35}$$

which matches the result determined completely analytically in [1, Ch. 4, Sec. 1].

6. Example 3 - Viscous Burgers's equation

In this example we compute the symmetries of a second-order, nonlinear partial differential equation to illustrate the treatment of higher-order equations. Namely, we find the symmetry group for the viscous Burgers's equation in the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} = 0, \tag{36}$$

which includes an additional second-order dissipative term in relation to the previous equation. However, we proceed analogously to inviscid Burgers's equation. Namely, we first determine an expression for ϕ^1 that we use to simplify the remaining determining equations down to an overdetermined system for η_1 and η_2.

The `.mac` file for the initial run contains

```
batchload("~/symmgrp.max");
writefile("~/Viscous_Burgers_Eqn_Maxima/viscous_burgers_run1.412");
batch("~/Viscous_Burgers_Eqn_Maxima/viscous_burgers_run1.dat");
symmetry(1,0,0);
printeqn(lode);
save ("~/Viscous_Burgers_Eqn_Maxima/lodeburgers.lsp", lode);
closefile();
quit();
```

and the `.dat` file with the initial inputs contains

```
p:2$
q:1$
m:1$
parameters:[]$
warnings:true$
sublisteqs:[all]$
info_given:false$
highest_derivatives:all$
e1:u[1,[1,0]]+u[1]*u[1,[0,1]]-u[1,[0,2]]$
v1: u[1,[1,0]]$
```

The output from the initial run includes the full system of determining equations, including the the following

$$\frac{\partial \eta_1}{\partial u^1} = 0, \qquad \frac{\partial \eta_1}{\partial x_2} = 0, \qquad \frac{\partial \eta_2}{\partial u^1} = 0, \tag{37}$$

from which we infer

$$\eta_1 := \eta_1(x_1), \qquad \eta_2 := \eta_2(x_1, x_2). \tag{38}$$

The system of determining equations also include the following

$$\frac{\partial^2 \phi^1}{\partial (u^1)^2} = 0, \tag{39}$$

$$2\left(\frac{\partial \eta_2}{\partial x_2}\right) - \frac{\partial \eta_1}{\partial x_1} = 0. \tag{40}$$

As we will see below, (39) implies that ϕ^1 is at most linear in u^1 and the Maxima gradient command `gradef()` can used to encode equation (40) into the next run.

The second run then consists of

```
info_given:true$
highest_derivatives:all$
depends(eta1,x[1])$
depends(eta2,[x[1],x[2]])$
gradef(eta1,x[1],2*diff(eta2,x[2]))$
```

From the second run, we obtain the following new output

$$\phi^1 = -\frac{\partial \eta_2}{\partial x_2} u^1 - \frac{\partial^2 \eta_2}{\partial x_2^2} + \frac{\partial \eta_2}{\partial x_1}, \tag{41}$$

which confirms the previous observation from (39) that ϕ^1 is at most linear in u^1. Now that we have an expression for ϕ^1, we can again utilize it in subsequent runs in order to determine the remaining expressions for η_1 and η_2. To that end, we include the expression for ϕ^1 in the next run as

```
phi1:-diff(eta2,x[2])*u[1]-diff(eta2,x[2],2)+diff(eta2,x[1])$
```

from which we then obtain the following output

$$\frac{\partial^2 \eta_2}{\partial x_1^2} = 0, \qquad \frac{\partial^2 \eta_2}{\partial x_2^2} = 0. \tag{42}$$

Together the previous two equations imply that η_2 is at most bilinear in x_1 and x_2, which gives

$$\eta_2 = (ax_1 + b)x_2 + cx_1 + d. \tag{43}$$

Then going back to equation equation (40), from the previous expression (43) for η_2 we also determine that

$$\eta_1 = ax_1^2 + bx_1 + e. \tag{44}$$

Finally, from (41) for ϕ^1, we obtain that

$$\phi^1 = -(ax_1 + b)u^1 + ax_2 + c, \tag{45}$$

where a, b, c, d, and e are arbitrary constants.

In terms of the original variables the corresponding infinitesimal generator is then

$$V = \left(at^2 + bt + e\right)\frac{\partial}{\partial t} + \left((at + b)x + ct + d\right)\frac{\partial}{\partial x} + \left(-(at + b)u + ax + c\right)\frac{\partial}{\partial u}, \tag{46}$$

which generates the five-parameter symmetry group admitted by viscous Burgers's equation. Moreover, note that this is a subgroup of the eight-parameter group we obtained in the previous example for inviscid Burgers's equation where

$$a = a_3, \qquad b = a_4, \qquad c = a_6, \qquad d = a_8, \qquad e = a_5. \tag{47}$$

7. Example 4 - The Euler equations of gas dynamics

To illustrate the analogous procedure in `symmgrp` for systems of equations, we analyze the symmetries of the Euler equations of gas dynamics. As we will see below, this example also illustrates how to apply `symmgrp` to compute the symmetries of differential equations containing undetermined constitutive functions, which in turn produce constraints on their admissible functional form as discussed for this example in [1, Ch. 4, Sec. 3].

We consider the Euler equations, expressed in terms of the isentropic bulk modulus in the form

$$\frac{\partial \rho}{\partial t} + v\frac{\partial \rho}{\partial r} + \rho\left(\frac{\partial v}{\partial r} + \frac{(n-1)v}{r}\right) = 0, \tag{48}$$

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial r} + \frac{\partial P}{\partial r} = 0, \tag{49}$$

$$\frac{\partial P}{\partial t} + v\frac{\partial P}{\partial r} + A(P,\rho)\left(\frac{\partial v}{\partial r} + \frac{(n-1)v}{r}\right) = 0, \tag{50}$$

where v denotes the velocity and the parameter $n = 1, 2, 3$ specifies $1D$ planar, cylindrical or spherical symmetry respectively. The isentropic bulk modulus is assumed to be an arbitrary function of pressure and density, $A := A(P, \rho)$. The symmetries of the Euler equations written in terms of A have been obtained previously by Ovsiannikov [8] and Axford [3].

It is worth mentioning before proceeding that calculating and solving the determining equations of the symmetry group admitted by the above system of equations (48)–(50) mirrors the approach applied in the two previous examples. However, due to the presence of the unspecified constitutive information introduced through the bulk modulus A, in addition to the η's and ϕ's, we will also obtain an additional partial differential equation from the system of determining equations constraining the class of isentropic bulk moduli admitted by the symmetry group for the Euler equations.

As before, to calculate the determining equations, we first construct an `Eulers_run1.mac` file containing

```
batchload("~/Desktop/symmgrp.max");
writefile("~/Desktop/Eulers_Eqns/Eulers_run1.412");
batch("~/Desktop/Eulers_Eqns/Eulers_run1.dat");
symmetry(1,0,0);
printeqn(lode);
save ("~/Desktop/Eulers_Eqns/lodeeulers.lsp", lode);
closefile();
quit();
```

and a `Eulers_run1.dat` data file with the initial inputs, which consists of

```
p:2$
q:3$
m:3$
parameters:[]$
warnings:true$
sublisteqs:[all]$
info_given:true$
highest_derivatives:all$
depends(A,[u[2],u[3]])$
depends([eta1,eta2,phi1,phi2,phi3],[x[1],x[2],u[1],u[2],u[3]])$
e1:u[3,[1,0]]+u[1]*u[3,[0,1]]+u[3]*(u[1,[0,1]]+((n-1)*u[1])/(x[2]))$
e2:u[3]*(u[1,[1,0]]+u[1]*u[1,[0,1]])+u[2,[0,1]]$
e3:u[2,[1,0]]+u[1]*u[2,[0,1]]-A*((u[3,[1,0]]+u[1]*u[3,[0,1]])/(u[3])
   )$
v1:u[3,[1,0]]$
v2:u[1,[1,0]]$
v3:u[2,[1,0]]$
```

where $u[1] = v$, $u[2] = P$, and $u[3] = \rho$.

Note that for systems of equations each equation is entered on a new line, as illustrated in lines 11 to 13. Moreover, for each equation, we must specify the variable to be used in subsequent automatic substitution steps. This enforces the determining equations to be evaluated under the constraint that ensures the system in (48)– (50) is satisfied. In this example, we assign the time derivative in each equation for this pupose, which is a general strategy that can be applied to any evolutionary system in mechanics of the form (48)–(50). As final note, equation (48) has been substituted into (50) to obtain the form of conservation of energy entered in line 13.

The initial run results in a system of 18 determining equations. The first seven of which are

$$\begin{aligned} &\frac{\partial \eta_1}{\partial u^1} = 0, \qquad \frac{\partial \eta_1}{\partial u^2} = 0, \qquad \frac{\partial \eta_1}{\partial u^3} = 0, \qquad \frac{\partial \eta_2}{\partial u^1} = 0, \\ &\frac{\partial \eta_2}{\partial u^2} = 0, \qquad \frac{\partial \eta_2}{\partial u^3} = 0, \qquad \frac{\partial \phi^2}{\partial u^3} = 0. \end{aligned} \tag{51}$$

These equations imply that the infinitesimals η_1 and η_2 are independent of the dependent variables, and likewise that ϕ^2 is independent of u_3. Assuming the isentropic bulk modulus is not identically zero, we rearrange the ninth equation to obtain the following expression for ϕ^1

$$\phi^1 = -u_1^2 \frac{\partial \eta_1}{\partial x_2} + u_1 \left(\frac{\partial \eta_2}{\partial x_2} - \frac{\partial \eta_1}{\partial x_1} \right) + \frac{\partial \eta_2}{\partial x_1}. \tag{52}$$

We also obtain an expression for ϕ^3

$$\phi^3 = u^3\left(2u^1\frac{\partial\eta_1}{\partial x_2} + \frac{\partial\phi^2}{\partial u^2} - \frac{\partial\phi^1}{\partial u^1} - \frac{\partial\eta_2}{\partial x_2} + \frac{\partial\eta_1}{\partial x_1}\right). \tag{53}$$

This information is fed into the second run in the form

```
depends(A,[u[2],u[3]])$
depends(eta1,[x[1],x[2]])$
depends(eta2,[x[1],x[2]])$
depends(phi2,[x[1],x[2],u[1],u[2]])$
phi1:-u[1]^2*diff(eta1,x[2])+u[1]*(diff(eta2,x[2])-diff(eta1,x[1]))+
   diff(eta2,x[1])$
phi3:u[3]*(2*u[1]*diff(eta1,x[2])+diff(phi2,u[2])-diff(phi1,u[1])-
   diff(eta2,x[2])+diff(eta1,x[1]))$
```

The output from the new run consists of the following

$$\frac{\partial^2\phi^2}{\partial(u^2)^2} = 0, \tag{54}$$

$$\frac{\partial\eta_1}{\partial x_2}A + \frac{\partial\phi^2}{\partial u^1} = 0, \tag{55}$$

which we apply to determine ϕ^2.

The first of these equations implies ϕ^2 is at most linear in u^2. From the second equation, since the first term does not depend on u^1, we conclude that ϕ^2 is at most linear in u^1. Therefore

$$\phi^2 = (au^1 + b)u^2 + cu^2 + d, \tag{56}$$

where a, b, c and d are arbitrary functions of x_1 and x_2. Now that we have obtained general expressions for ϕ^1, ϕ^2, and ϕ^3, we will use this information to determine η_1 and η_2 from the remaining determining equations.

We use the previous equation as input for the next run

```
depends(a,[x[1],x[2]])$
depends(b,[x[1],x[2]])$
depends(c,[x[1],x[2]])$
depends(d,[x[1],x[2]])$
phi1:-u[1]^2*diff(eta1,x[2])+u[1]*(diff(eta2,x[2])-diff(eta1,x[1]))+
   diff(eta2,x[1])$
phi2:(a*u[1]+b)*u[2]+c*u[1]+d$
phi3:u[3]*(2*u[1]*diff(eta1,x[2])+diff(phi2,u[2])-diff(phi1,u[1])-
   diff(eta2,x[2])+diff(eta1,x[1]))$
```

From the first three equations of the new output

$$3\frac{\partial\eta_1}{\partial x_2} + a = 0, \tag{57}$$

$$\frac{\partial\eta_1}{\partial x_2}A + c + u^2a = 0, \tag{58}$$

$$4\frac{\partial\eta_1}{\partial x_2}A + aA - c - u^2a = 0, \tag{59}$$

we find that

$$a = 0, \qquad c = 0, \qquad \frac{\partial\eta_1}{\partial x_2} = 0. \tag{60}$$

The last equality implies that η_1 is independent of x_2. The next run consists of the following.

```
depends(eta1,x[1])$
depends(eta2,[x[1],x[2]])$
depends(b,[x[1],x[2]])$
depends(d,[x[1],x[2]])$
phi1:u[1]*(diff(eta2,x[2])-diff(eta1,x[1]))+diff(eta2,x[1])$
phi2:b*u[2]+d$
phi3:u[3]*(diff(phi2,u[2])-diff(phi1,u[1])-diff(eta2,x[2])+diff(eta1
   ,x[1]))$
```

Then we obtain the equation

$$\frac{\partial^2\eta_2}{\partial x_2^2}(u^1)^2u^3+\left[2\frac{\partial^2\eta_2}{\partial x_1\partial x_2}-\frac{\partial^2\eta_1}{\partial x_1^2}\right]u^1u^3+\frac{\partial^2\eta_2}{\partial x_1^2}u^3+\frac{\partial b}{\partial x_2}u^2+\frac{\partial d}{\partial x_2}=0. \tag{61}$$

Treating the left-hand side of the previous as a polynomial in u^1, u^2, and u^3, the previous equation is satisfied for all solutions u^1, u^2 and u^3 provided the coefficients are identically zero. Therefore, we obtain the following system of equations

$$\frac{\partial^2\eta_2}{\partial x_2^2}=0, \qquad \frac{\partial^2\eta_2}{\partial x_1^2}=0, \tag{62}$$

$$\frac{\partial b}{\partial x_2}=0, \qquad \frac{\partial d}{\partial x_2}=0. \tag{63}$$

The first two equations imply that η_2 is bilinear in x_1 and x_2,

$$\eta_2=(e+fx_1)x_2+gx_1+h. \tag{64}$$

The next two equations dictate that b and d are only functions of x_1. We also obtain the equation

$$2\frac{\partial^2\eta_2}{\partial x_1\partial x_2}-\frac{\partial^2\eta_1}{\partial x_1^2}=0, \tag{65}$$

from which, we infer

$$\eta_1=fx_1^2+kx_1+l. \tag{66}$$

The next run contains the previous information

```
depends(b,x[1])$
depends(d,x[1])$
eta1:f*x[1]^2+k*x[1]+l$
eta2:(e+f*x[1])*x[2]+g*x[1]+h$
```

The output of the above contains the following equation

$$-\left(\frac{\partial b}{\partial x_1}+(n+2)f\right)x_2^2-g(n-1)x_2+u^1(n-1)(h-x_1g)=0. \tag{67}$$

The left-hand side is a polynomial in x_2, from which we can obtain

$$\frac{\partial b}{\partial x_1}+f(n+2)=0, \tag{68}$$

$$g(n-1)=0, \tag{69}$$

$$(n-1)(h-x_1g)=0. \tag{70}$$

Integrating the first equation with respect to x_1,

$$b=-f(n+2)x_1+m, \tag{71}$$

where m is a constant. The remaining pair of equations imply that h and g must be zero, for $n > 1$. We use the expression for b as the new input in the next run

```
depends(d,x[1])$
eta1:f*x[1]^2+k*x[1]+l$
eta2:(e+f*x[1])*x[2]+g*x[1]+h$
phi1:u[1]*(diff(eta2,x[2])-diff(eta1,x[1]))+diff(eta2,x[1])$
phi2:(-f*(n+2)*x[1]+m)*u[2]+d$
```

We can treat the third equation obtained from the output again as a polynomial in x_2, and therefore in order to satisfy the equation, the following coefficient must be zero

$$\frac{\partial d}{\partial x_1} + f\left(nA - (n+2)u^2\right) = 0. \tag{72}$$

Recognizing that d is a function of only x_1, the second term must be independent of u_2, which requires either

$$f = 0, \text{ or } A = \frac{n+2}{n}u^2. \tag{73}$$

In either case,

$$\frac{\partial d}{\partial x_1} = 0, \tag{74}$$

in other words d is constant. We have now explicitly determined the infinitesimals η_1 and η_2, which then fully determine ϕ^1, ϕ^2, and ϕ^3. The complete expressions are

$$\eta_1 = fx_1^2 + kx_1 + l, \tag{75}$$
$$\eta_2 = (e + fx_1)x_2 + gx_1 + h, \tag{76}$$
$$\phi^1 = g + (e-k)u^1 + f\left(x_2 - x_1u^1\right), \tag{77}$$
$$\phi^2 = (-fx_1(n+2) + m)u^2 + d, \tag{78}$$
$$\phi^3 = [m + 2(k-e)]u^3 - fnx_1u^3. \tag{79}$$

Relabeling the free constants in terms of $a_i (i = 1, 2, \ldots 8)$

$$a_1 = l, \qquad a_2 = e, \qquad a_3 = h, \qquad a_4 = g, \tag{80}$$
$$a_5 = k - e, \qquad a_6 = m, \qquad a_7 = d, \qquad a_8 = f, \tag{81}$$

the infinitesimals determined above are given by

$$\eta_1 = a_1 + (a_2 + a_5)t + a_8t^2, \tag{82}$$
$$\eta_2 = a_3 + a_2r + a_4t + a_8tr, \tag{83}$$
$$\phi^1 = a_4 - a_5v + a_8(r - vt), \tag{84}$$
$$\phi^2 = a_7 + a_6p - a_8(n+2)pt, \tag{85}$$
$$\phi^3 = (2a_5 + a_6)\rho - a_8n\rho t. \tag{86}$$

In terms of the original variables, the corresponding eight-parameter symmetry group of the Euler equations is generated by the vector field

$$V = \left[a_1 + (a_2 + a_5)t + a_8t^2\right]\frac{\partial}{\partial t} + (a_3 + a_2r + a_4t + a_8tr)\frac{\partial}{\partial r} \tag{87}$$
$$+ [a_4 - a_5u + a_8(r - vt)]\frac{\partial}{\partial v} + [a_7 + a_6p - a_8(n+2)pt]\frac{\partial}{\partial P} \tag{88}$$
$$+ [(2a_5 + a_6)\rho - a_8n\rho t]\frac{\partial}{\partial \rho}, \tag{89}$$

with the additional restrictions on a_3, a_4, a_8 and A imposed by (69), (70) and (73).

In the final run, we consider the special case where $A \neq (n+2)/n$, and therefore $a_8 = 0$. The data file for the final consists of the following

```
eta1:a1+(a2+a5)*x[1]$
eta2:a3+a2*x[2]+a4*x[1]$
phi1:a4-a5*u[1]$
phi2:a7+a6*u[2]$
phi3:(2*a5+a6)*u[3]$
```

In conclusion, the remaining determining equation yields a partial differential equation

$$a_6 A - \frac{\partial A}{\partial P}(a_6 P + a_7) - \frac{\partial A}{\partial \rho}(a_6 + 2a_5)\rho = 0, \tag{90}$$

defining the admissible functional forms of the isentropic bulk modulus, as we mentioned previously. In terms of ϕ^2 and ϕ^3 from (82)–(86), the previous equation can be written

$$A\frac{\partial \phi^2}{\partial P} - \frac{\partial A}{\partial P}\phi^2 - \frac{\partial A}{\partial \rho}\phi^3 = 0. \tag{91}$$

Equation (91) is identical to the original result obtained by Ovsiannikov [8, p.131]. Moreover, equation (91) imposes a constraint on the maximum number of symmetries admitted by the Euler equations based on the particular functional form of the isentropic bulk modulus. To illustrate further, if the bulk modulus corresponds to an ideal gas, in other words,

$$A = \gamma P, \tag{92}$$

where γ is the ratio of specific heat capacities, equation is (91) is only satisfied provided $a_7 = 0$. Thus, in the case of ideal gases, the corresponding symmetry group admitted by the Euler equations is is a reduced six-parameter group for planar symmetry, and likewise a reduced four-parameter group for the cylindrical ($n = 2$) and spherical ($n = 3$) cases, which is a result originally due to Axford [3, p.5-6].

8. Conclusions

The purpose of this report was to illustrate how to find the symmetry group of a system of differential equations more efficiently using symbolic computation. To illustrate, we used `symmgrp` to compute the symmetry group admitted by several example problems including, an ordinary differential equation, first-order and second-order partial differential equations, and a system of partial differential equations. For comparison with the analytic procedure, complete step-by-step calculations for two of the examples treated here are presented in the accompanying primer [1]. Finally, although we illustrated how to use the Maxima-based `symmgrp` program, the strategy and analysis we described here are applicable with only minor modification to most other existing symmetry analysis software programs.

References

[1] E. Jason Albright, James D. McHardy, Scott D. Ramsey, and Joseph H. Schmidt, *Symmetry analysis of differential equations: A primer*, Los Alamos National Laboratory, 2018.

[2] R. L. Anderson, V. A. Baikov, R. K. Gazizov, W. Hereman, N. H. Ibragimov, F. M. Mahomed, S. V. Meleshko, M. C. Nucci, P. J. Olver, M. B. Sheftel′, A. V. Turbiner, and E. M. Vorob′ ev, *CRC handbook of Lie group analysis of differential equations. Vol. 3*, CRC Press, Boca Raton, FL, 1996. New trends in theoretical developments and computational methods.

[3] Roy Axford, *Solutions of the Noh problem for various equations of state using lie grous*, Los Alamos National Laboratory, 1998.

[4] George W. Bluman and Stephen C. Anco, *Symmetry and integration methods for differential equations*, Applied Mathematical Sciences, vol. 154, Springer-Verlag, New York, 2002.

[5] B. Champagne, W. Hereman, and P. Winternitz, *The computer calculation of Lie point symmetries of large systems of differential equations*, Comput. Phys. Comm. **66** (1991), no. 2-3, 319–340.
[6] Sophus Lie, *Zur allgemeinen theorie der partiellen differentialgleichungen beliebiger ordnung*, Leipzig Bereich. **1** (1895), 53–128.
[7] Peter J. Olver, *Applications of Lie groups to differential equations*, Second, Graduate Texts in Mathematics, vol. 107, Springer-Verlag, New York, 1993.
[8] L. V. Ovsiannikov, *Group analysis of differential equations*, Academic Press, Inc.,New York-London, 1982. Translated from the Russian by Y. Chapovsky, Translation edited by William F. Ames.
[9] Hans Stephani, *Differential equations: Their solution using symmetries*, Cambridge University Press, Cambridge, 1989.

E-mail address: ejalbright@lanl.gov

X-Theoretical Design Division, Los Alamos National Laboratory, Los Alamos, NM, USA