Algorithm in SageMath to find determining equations of infinitesimals admitted by partial differential equations

Vishwas Khare [1] and M.G. Timol [2]

[1] *Department of Mathematics*
SSR College of ACS, Silvassa
vskssr@gmail.com

[2] *Former Professor, Department of Mathematics*
Veer Narmad South Gujarat University, Surat
mgtimol@gmail.com

Abstract

Algorithm in Computer algebra system (open-source software Sage Math) is developed to find determining equations of the infinitesimals admitted by *PDE* of order one and two involving four independent variables x,y,z,t and dependent variable u for the deductive group method of Bluman and Cole . The application of algorithm is illustrated by examples. The algorithm gives the set of determining equations by giving inputs as *PDE* under consideration written in solved form. The algorithm is universal and very useful for researcher working with linear/nonlinear *PDE* using Lie symmetry method.

Subject Classification:[2010]76M60, 58J70.
Keywords: Lie symmetry, Second order*PDE*, SageMath software, Infinitesimal.

1 Introduction

A similarity transformation reduces the number of independent variables in the partial differential equations. The transformed system of equations and auxiliary conditions is known as a similarity representation. In the deductive group method of Bluman and Cole a general infinitesimal group of transformations is considered initially . By invocation of invariance under the infinitesimal group the determining equations are derived. The determining equations are a set of linear differential equations, which on solving gives the transformation function or the infinitesimals of the dependent and independent variables [15]. The Lie symmetry technique is an efficient, reliable, and robust mathematical tool to solve nonlinear differential equations studied and developed by [1, 2, 3, 6, 7, 8, 13, 14, 15]. Symmetry method for solving non linear differential equations is extensively used and appeared in many research paper recently [4, 9, 10, 11, 12].
In this paper algorithm in open-source software Sage Math(Computer algebra system) is developed to find determining equations of the infinitesimals admitted by *PDE* of order one and two involving four independent variables. Open-source software is computer software with its source code made available by the developer to everybody which can be modified and enhance by user. On the other hand, commercial software has source code that only the person, team, or organization that created it can edit, inspect, change and enhance it.

There are packages to find symmetry of PDE available in commercial software but as per authors knowledge no such algorithm is not developed in open source SageMath.

The application of algorithm is illustrated by Fisher's equation, Laplace equation and Heat equation . The program finds the set of determining equations by giving the inputs as *PDE* written in solved form as explained in the examples. The codes given in the algorithm can be downloaded using the link `http://tiny.cc/vebvtz.` and can be used, using SageMath Cell, SageMath cloud .

2 Mathematical concepts

Consider [3] the kth order *PDE* written in solved form in terms of some k th order partial derivatives of u:

$$F(x,u,\underset{1}{u},\underset{2}{u},\dots\underset{k}{u}) = u_{i_1,i_2\dots i_l} - f(x,u,\underset{1}{u},\underset{2}{u},\dots,\underset{k}{u}) = 0\,. \tag{2.1}$$

where $x = (x_1, x_2, \dots, x_n)$, denotes n independent variables , u denotes the dependent variable, and $\underset{j}{u}$ denotes the set of coordinates corresponding to all jth order partial derivatives of u with respect to x. The coordinate of $\underset{j}{u}$ corresponding to $\dfrac{\partial^j}{\partial x_{i_1}, \partial x_{i_2} \dots \partial x_{i_j}}$ is dentoes by $u_{i_1,i_2,\dots i_j}$ $i_j = 1,2,\dots,n$ for $j = 1,2,\dots,k$.

Theorem 2.1. *(Infinitesimal Criterion for Invariance of a PDE) Let [3]*

$$\mathbb{X} = \xi_i \frac{\partial}{\partial x_i} + \eta \frac{\partial}{\partial u}. \tag{2.2}$$

be the infinitesimal generator of one parameter Lie group of transformations,

$$\begin{aligned} x^* &= \mathbf{X}(x,u;\varepsilon),\\ u^* &= \mathbf{U}(x,u;\varepsilon). \end{aligned} \tag{2.3}$$

where ξ_i and η are infinitesimals.
Let

$$\begin{aligned} \mathbb{X}^{(k)} =& \xi_i \frac{\partial}{\partial x_i} + \eta \frac{\partial}{\partial u} + \eta_i^{(1)}(x,u,\underset{1}{u}) \frac{\partial}{\partial u_i}\\ &+ \dots + \eta^{(k)}_{i_1,i_2,\dots i_j} \frac{\partial}{\partial u_{i_1,i_2,\dots i_j}}. \end{aligned} \tag{2.4}$$

be the kth extended infinitesimal generator of "(2.2)" where $\eta_i^{(1)}$ and $\eta^{(k)}_{i_1,i_2,\dots i_j}$ are given by

$$\begin{aligned} \eta_i^{(1)} &= D_i\eta - (D_i\xi_j)u_j, \quad i = 1,2,\dots,n;\\ \eta^{(k)}_{i_1,i_2,\dots i_k} &= D_{i_k}\eta^{(k-1)} - (D_{i_k}\xi_j)u_{i_1,i_2,\dots i_{k-1}j},\\ & i_l = 1,2,\dots,n;\ for\, l = 1,2,\dots,k\\ & with\, k = 2,3,\dots \end{aligned} \tag{2.5}$$

and $D_i = \dfrac{D}{Dx_i} = \dfrac{\partial}{\partial x_i} + u_i \dfrac{\partial}{\partial x_i} + u_{ij} \dfrac{\partial}{\partial u_j} + \cdots + u_{i_1,i_2,\ldots i_j} \dfrac{\partial}{\partial u_{i_1,i_2,\ldots i_j}}$.

Then "(2.3)" is admitted by PDE "(2.1)" if and only if

$$\mathbb{X}^{(k)} F(x, u, \underset{1}{u}, \underset{2}{u}, \ldots \underset{k}{u}) = 0.$$

when

$$F(x, u, \underset{1}{u}, \underset{2}{u}, \ldots \underset{k}{u}) = 0. \tag{2.6}$$

Proof. For proof see [3] □

Remark 2.2. *Equations "(2.6)" is called invariance condition or linearized symmetry condition.*

3 Symbols used in Algorithm

Tab. 1: Table for equivalent symbols used in algorithm and examples.

Symbols	Equivalent Symbol	Symbols	Equivalent Symbol	Symbols	Equivalent Symbol
$\frac{\partial u}{\partial x}$	`u_x`	$\frac{\partial u}{\partial y}$	`u_y`	$\frac{\partial u}{\partial z}$	`u_z`
$\frac{\partial u}{\partial t}$	`u_t`	$\frac{\partial^2 u}{\partial xx}$	`u_xx`	$\frac{\partial^2 u}{\partial xy}$	`u_xy`
$\frac{\partial^2 u}{\partial xz}$	`u_xz`	$\frac{\partial^2 u}{\partial yy}$	`u_yy`	$\frac{\partial^2 u}{\partial yz}$	`u_yz`
$\frac{\partial^2 u}{\partial yt}$	`u_yt`	$\frac{\partial^2 u}{\partial zz}$	`u_zz`	$\frac{\partial^2 u}{\partial zt}$	`u_zt`
$\frac{\partial^2 u}{\partial tt}$	`u_tt`				

Remark 3.1.

1. *For* $n = 2$, *use x,t as independent and u as dependent variable in the algorithm also* $\xi_1 = X, \xi_2 = T, \eta = U$ *in this case.*

2. *For* $n = 3$,*use x,y,t as independent and u as dependent variable in the algorithm also* $\xi_1 = X, \xi_2 = Y, \xi_3 = T, \eta = U$ *in this case.*

3. *For* $n = 4$,*use x,y,z,t as independent and u as dependent variable in the algorithm also* $\xi_1 = X, \xi_2 = Y, \xi_3 = Z, \xi_4 = T, \eta = U$ *in this case.*

4 Algorithm

```
# Program for finding determining equation for PDE OF ORDER ONE
AND TWO FOR TWO THREE AND FOUR VARIBALES
print ("Program to find determining equ of the type
u_i=f(u_k,u,x,t) where i,k can take value x,y,z,t,xx,xy,xz,xt,
yy,yz,yt,zz,zt,tt and u_i is not equal to u_k.")
print("Use x,y,z,t for 4 independent varibles, x,y,t
for 3 independent  varibles,and ,x,t for 2
independent  varibales")
var('x,t,y,z,u,u_x,u_y,u_z,u_t, u_xx,u_xy,u_xz,u_xt,u_yy,u_yz
,u_yt,u_zz,u_zt,u_tt,u_yx,u_zx,u_tx,u_zy,u_ty,u_tz')
function('X,Y,f,F,U,T,V,w,Z') # Define function
import itertools
@interact
def partial_4variablesymmetry(n=input_box(default=
[4,u_t,u_xx+u_yy+u_zz],label='No of independent variable n,
LHS,RHS of eq')):
  A=n[1](u_yx=u_xy,u_zx=u_xz,u_tx=u_xt,u_zy=u_yz,
u_ty=u_yt,u_tz=u_zt)
  w=n[2](u_yx=u_xy,u_zx=u_xz,u_tx=u_xt,u_zy=u_yz
,u_ty=u_yt,u_tz=u_zt)
  W=A-(w)
  r=n[0]
  if(r==2):
    B=[X(x,t,u),0,0,T(x,t,u),U(x,t,u)]
    C1,C2,C3,C4=[u_x],[0],[0],[u_t]
    N=[u_x,u_t,u_xx,u_xt,u_tt]
  elif (r==3):
    B=[X(x,y,t,u),Y(x,y,t,u),0,T(x,y,t,u),U(x,y,t,u)]
    C1,C2,C3,C4=[u_x],[u_y],[0],[u_t]
    N=[u_x,u_y,u_t,u_xx,u_xy,u_xt,u_yy,u_yt,u_tt]
  elif (r==4):
    B=[X(x,y,z,t,u),Y(x,y,z,t,u),Z(x,y,z,t,u),
    T(x,y,z,t,u),U(x,y,z,t,u)]
    C1,C2,C3,C4=[u_x],[u_y],[u_z],[u_t]
    N=[u_x,u_y,u_z,u_t,u_xx,u_xy,u_xz,u_xt,u_yy,
    u_yz,u_yt,u_zz,u_zt,u_tt]
    L=[]
    for j in range(0,5):
      for k in range(0,1):
          a1=diff(B[j],x)+C1[k]*diff(B[j],u)
          a2=diff(B[j],y)+C2[k]*diff(B[j],u)
          a3=diff(B[j],z)+C3[k]*diff(B[j],u)
          a4=diff(B[j],t)+C4[k]*diff(B[j],u)
```

```
L.append(a1)
L.append(a2)
L.append(a3)
L.append(a4)
# A)———————————(For first order pde)
U_x=L[16]-(u_x*L[0]+u_y*L[4]+u_z*L[8]+u_t*L[12])
U_y=L[17]-(u_x*L[1]+u_y*L[5]+u_z*L[9]+u_t*L[13])
U_z=L[18]-(u_x*L[2]+u_y*L[6]+u_z*L[10]+u_t*L[14])
U_t=L[19]-(u_x*L[3]+u_y*L[7]+u_z*L[11]+u_t*L[15])
#print(" Value of U_x,U_y,U_z,U_t is")
# B )———————————(For second order pde)
M=[]
D1=[U_x,U_y,U_z,U_t]
for i in range(0,4):
    b1=(diff(D1[i],x)+u_x*diff(D1[i],u)+(u_xx*
     diff(D1[i],u_x)+u_xy*diff(D1[i],u_y)+u_xz*
     diff(D1[i],u_z)+u_xt*diff(D1[i],u_t)))

    b2=(diff(D1[i],y)+u_y*diff(D1[i],u)+(u_xy*
     diff(D1[i],u_x)+u_yy*diff(D1[i],u_y)+u_yz*
     diff(D1[i],u_z)+u_yt*diff(D1[i],u_t)))

    b3=(diff(D1[i],z)+u_z*diff(D1[i],u)+(u_xz*
     diff(D1[i],u_x)+u_yz*diff(D1[i],u_y)+u_zz*
     diff(D1[i],u_z)+u_zt*diff(D1[i],u_t)))

    b4=(diff(D1[i],t)+u_t*diff(D1[i],u)+(u_xt*
     diff(D1[i],u_x)+u_yt*diff(D1[i],u_y)+u_zt*
     diff(D1[i],u_z)+u_tt*diff(D1[i],u_t)))

    M.append(b1)
    M.append(b2)
    M.append(b3)
    M.append(b4)


# C )———————————(For second order pde)
U_xx=M[0]-(L[0]*u_xx+L[4]*u_xy+L[8]*u_xz+L[12]*
u_xt)
U_xy=M[4]-(L[0]*u_xy+L[4]*u_yy+L[8]*u_yz+L[12]*
u_yt)
U_xz=M[8]-(L[0]*u_xz+L[4]*u_yz+L[8]*u_zz+L[12]*
u_zt)
U_xt=M[12]-(L[0]*u_xt+L[4]*u_yt+L[8]*u_zt+L[12]*
u_tt)
```

```
U_yy=M[5]-(L[1]*u_xy+L[5]*u_yy+L[9]*u_yz+L[13]*
u_yt)
U_yz=M[9]-(L[1]*u_xz+L[5]*u_yz+L[9]*u_zz+L[13]*u_zt)
U_yt=M[13]-(L[1]*u_xt+L[5]*u_yt+L[9]*u_zt+L[13]*u_tt)

U_zz=M[10]-(L[2]*u_xz+L[6]*u_yz+L[10]*u_zz+L[14]*u_zt)
U_zt=M[14]-(L[2]*u_xt+L[6]*u_yt+L[10]*u_zt+L[14]*u_tt)

U_tt=M[15]-(L[3]*u_xt+L[7]*u_yt+L[11]*u_zt+L[15]*u_tt)

X4=(B[0]*diff(W,x)+B[1]*diff(W,y)+B[2]*diff(W,z)+B[3]
*diff(W,t))+B[4]*diff(W,u)+(U_x*diff(W,u_x)+U_y*diff(W,u_y)
+U_z*diff(W,u_z)+U_t*diff(W,u_t))+((U_xx)*diff(W,u_xx)
+(U_xy)*diff(W,u_xy)+(U_xz)*diff(W,u_xz)+(U_xt)*diff(W,u_xt)
+(U_yy)*diff(W,u_yy)+(U_yz)*diff(W,u_yz
)+(U_yt)*diff(W,u_yt)+(U_zz)*diff(W,u_zz)+(U_zt)
*diff(W,u_zt)+(U_tt)*diff(W,u_tt))

if (A==u_x):
    K=X4(u_x=w).simplify_full()
elif (A==u_y):
    K=X4(u_y=w).simplify_full()
elif (A==u_z):
    K=X4(u_z=w).simplify_full()
elif (A==u_t):
    K=X4(u_t=w).simplify_full()
elif (A==u_xx):
    K=X4(u_xx=w).simplify_full()
elif (A==u_xy):
    K=X4(u_xy=w).simplify_full()
elif (A==u_xz):
    K=X4(u_xz=w).simplify_full()
elif (A==u_xt):
    K=X4(u_xt=w).simplify_full()
elif (A==u_yy):
    K=X4(u_yy=w).simplify_full()
elif (A==u_yz):
    K=X4(u_yz=w).simplify_full()
elif (A==u_yt):
    K=X4(u_yt=w).simplify_full()
elif (A==u_zz):
    K=X4(u_zz=w).simplify_full()
elif (A==u_zt):
    K=X4(u_zt=w).simplify_full()
```

```
  elif (A==u_tt):
     K=X4(u_tt=w).simplify_full()
  K=(numerator(K))
  print("The determining equations are given by")
  F=[1,2,3,4]
  E=[1,2]
  I=[]
  J=[]
  v=[]
  for i,j,k in itertools.product(N,N,N):
      if i!=j and j!=k:
          for a,b,c in itertools.product(F,F,F):
              s=i^a*j^b*k^c
              e=i^a*j^b
              d=i^a
              I.append(s)
              J.append(e)
              v.append(d)
  for m in I :
      if ((K.coefficient(m))!=0):
      #print("The coefficient of " ,m, "is")
      show(K.coefficient(m)==0)
      K=(K-(K.coefficient(m)*m)).simplify_full()
  for m in J:
      if ((K.coefficient(m))!=0):
      #print("The coefficient of " ,m, "is")
      show(K.coefficient(m)==0)
      K=(K-(K.coefficient(m)*m)).simplify_full()
  for m in v:
      if ((K.coefficient(m))!=0   ):
      #print("The coefficient of " ,m, "is")
      show(K.coefficient(m)==0)
      K=(K-(K.coefficient(m)*m)).simplify_full()
  #print("The coefficient of u_x^0","is")
  show(K==0)
  print("All determining equations have been found.")
  print("End of the program.")
```

Remark 4.1. *The algorithm ends with message "All determining equations have been found", "End of program".*

5 Examples

Consider the Fisher's equation [3]

$$u_{xx} = u_t + u(u-1). \tag{5.1}$$

The invariance condition in this case is

$$\mathbb{X}^{(2)}(u_{xx} - u_t - u(u-1)) = 0 \text{ when } u_{xx} = u_t + u(u-1)\,. \tag{5.2}$$

where $\mathbb{X} = X\dfrac{\partial}{\partial x} + T\dfrac{\partial}{\partial t} + U\dfrac{\partial}{\partial u},\ \mathbb{X}^{(1)} = \mathbb{X} + U_{[x]}\dfrac{\partial}{\partial u_x} + U_{[t]}\dfrac{\partial}{\partial u_t}$,

$$\mathbb{X}^{(2)} = \mathbb{X}^{(1)} + U_{[xx]}\frac{\partial}{\partial u_{xx}} + U_{[xt]}\frac{\partial}{\partial u_{xt}} + U_{[tt]}\frac{\partial}{\partial u_{tt}}. \tag{5.3}$$

Input: We give input [2, `u_xx`, `u_t+u(u-1)`] where 2 represents number of independent variables , `u_xx`, is LHS and `u_t+u(u-1)`, is RHS of equation `u_{xx}=u_t+u(u-1)`, written in solved form.

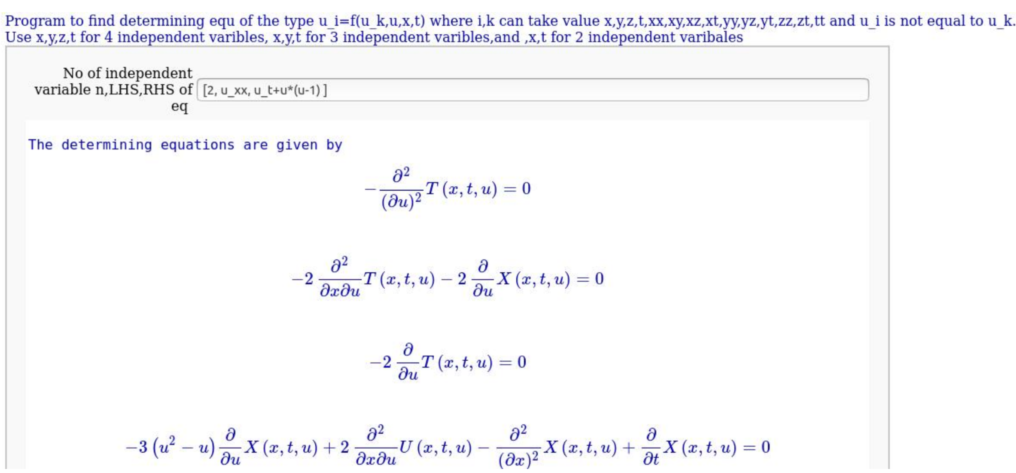

Fig. 1

$$\frac{\partial^2}{(\partial u)^2}U(x,t,u) - 2\,\frac{\partial^2}{\partial x \partial u}X(x,t,u) = 0$$

$$-\frac{\partial^2}{(\partial u)^2}X(x,t,u) = 0$$

$$-\left(u^2-u\right)\frac{\partial}{\partial u}T(x,t,u) - \frac{\partial^2}{(\partial x)^2}T(x,t,u) + \frac{\partial}{\partial t}T(x,t,u) - 2\,\frac{\partial}{\partial x}X(x,t,u) = 0$$

$$-2\,\frac{\partial}{\partial x}T(x,t,u) = 0$$

$$-(2\,u-1)U(x,t,u) + \left(u^2-u\right)\frac{\partial}{\partial u}U(x,t,u) - 2\left(u^2-u\right)\frac{\partial}{\partial x}X(x,t,u) + \frac{\partial^2}{(\partial x)^2}U(x,t,u) - \frac{\partial}{\partial t}U(x,t,u) = 0$$

```
All determining equations have been found.
End of the program.
```

Fig. 2

Output: We get following set of determining equations using algorithm.

(5.4)
$$\begin{aligned} T_u = 0, X_u &= 0, \\ -T_t + 2Y_z &= 0, \\ 2U_{xu} - X_{xx} + X_t &= 0, \\ U_{uu} &= 0, \\ T_t - 2X_x &= 0 \\ T_x &= 0 \\ -(2u-1)U + (u^2-u)U_u - 2(u^2-u)X_x + U_{xx} - U_t &= 0. \end{aligned}$$

Solving above determining equations we get infinitesimals.

$$T = c_1, X = c_2, U = 0.$$

where c_1, c_2 are arbitrary constants.

Consider the Laplace equation [3]

(5.5)
$$u_{tt} + u_{xx} + u_{yy} = 0\,.$$

The invariance condition in this case is

(5.6)
$$\mathbb{X}^{(2)}(u_{tt} + u_{xx} + u_{yy}) = 0 \text{ when } u_{tt} = -u_{xx} - u_{yy}\,.$$

where $\mathbb{X} = X\dfrac{\partial}{\partial x} + Y\dfrac{\partial}{\partial y} + T\dfrac{\partial}{\partial t} + U\dfrac{\partial}{\partial u}$, $\mathbb{X}^{(1)} = \mathbb{X} + U_{[x]}\dfrac{\partial}{\partial u_x} + U_{[y]}\dfrac{\partial}{\partial u_y} + U_{[t]}\dfrac{\partial}{\partial u_t}$,

(5.7)
$$\mathbb{X}^{(2)} = \mathbb{X}^{(1)} + U_{[xx]}\frac{\partial}{\partial u_{xx}} + U_{[xy]}\frac{\partial}{\partial u_{xy}} + U_{[xt]}\frac{\partial}{\partial u_{xt}} + U_{[yy]}\frac{\partial}{\partial u_{yy}} + U_{[yt]}\frac{\partial}{\partial u_{yt}} + U_{[tt]}\frac{\partial}{\partial u_{tt}}.$$

Input: We give input $[3,$ `u_tt`, `-u_xx-u_yy` $]$ where 3 represents number of independent variables, `u_tt`,is LHS and `-u_xx-u_yy`, is RHS of equation `u_tt =-u_xx-u_yy`, written in solved form.

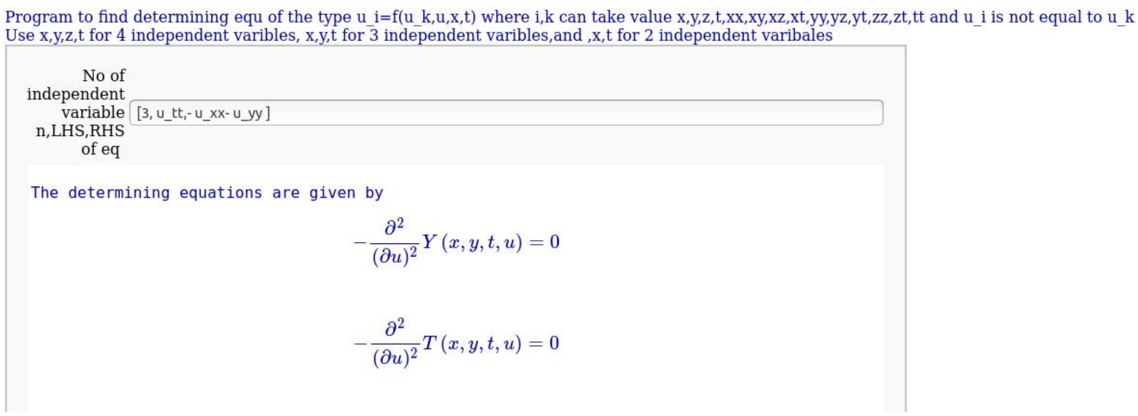

Fig. 3

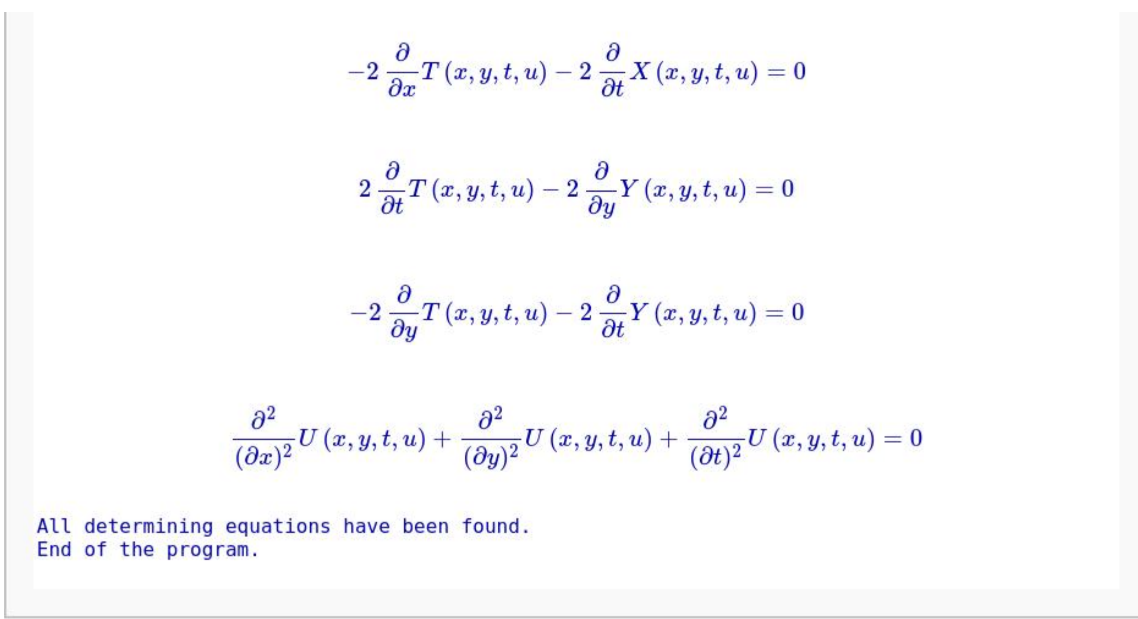

Fig. 4

Output: We get following set of determining equations using algorithm.

$$
\begin{aligned}
X_u = 0, Y_u = 0, T_u &= 0, \\
2U_{xu} - X_{xx} - X_{yy} - X_{tt} &= 0, \\
U_{uu} &= 0, \\
2U_{yu} - Y_{xx} - Y_{yy} - Y_{tt} &= 0, \\
-T_{xx} - T_{yy} - T_{tt} + 2U_{tu} &= 0, \\
T_t - X_x &= 0, \\
-X_y - Y_x &= 0, \\
-T_x - X_t &= 0, \\
T_t - Y_y &= 0, \\
T_y - Y_t &= 0, \\
U_{xx} + U_{yy} + U_{tt} &= 0.
\end{aligned} \tag{5.8}
$$

Solving above determining equations we get infinitesimals X,Y,T, U..

Consider the heat equation in three dimensions [3]

$$u_t = u_{xx} + u_{yy} + u_{zz}\,. \tag{5.9}$$

We find the 13-parameter Lie group of transformations admitted by *PDE* by finding determining equations using algorithm.
The invariance condition in this case is

$$\mathbb{X}^{(2)}(u_t - u_{xx} - u_{yy} - u_{zz}) = 0 \text{ when } u_t = u_{xx} + u_{yy} + u_{zz}\,. \tag{5.10}$$

where $\mathbb{X} = X\frac{\partial}{\partial x} + Y\frac{\partial}{\partial y} + T\frac{\partial}{\partial t} + Z\frac{\partial}{\partial z} + U\frac{\partial}{\partial u}$, $\mathbb{X}^{(1)} = \mathbb{X} + U_{[x]}\frac{\partial}{\partial u_x} + U_{[y]}\frac{\partial}{\partial u_y} + U_{[z]}\frac{\partial}{\partial u_z} + U_{[t]}\frac{\partial}{\partial u_t}$,

$$
\begin{aligned}
\mathbb{X}^{(2)} = \mathbb{X}^{(1)} &+ U_{[xx]}\frac{\partial}{\partial u_{xx}} + U_{[xy]}\frac{\partial}{\partial u_{xy}} + U_{[xz]}\frac{\partial}{\partial u_{xz}} + U_{[xt]}\frac{\partial}{\partial u_{xt}} \\
&+ U_{[yy]}\frac{\partial}{\partial u_{yy}} + U_{[yz]}\frac{\partial}{\partial u_{yz}} + U_{[yt]}\frac{\partial}{\partial u_{yt}} + U_{[zz]}\frac{\partial}{\partial u_{zz}} + U_{[zt]}\frac{\partial}{\partial u_{zt}} + U_{[tt]}\frac{\partial}{\partial u_{tt}}.
\end{aligned} \tag{5.11}
$$

Input: We give input $[4,$ `u_t`, `u_xx+u_yy+u_zz` $]$ where 4 represent number of independent variables, `u_t`, is LHS and `u_xx+u_yy+u_zz`, is RHS of equation `u_t = u_xx+u_yy+u_zz`, written in solved form.

Program to find determining equ of the type u_i=f(u_k,u,x,t) where i,k can take value x,y,z,t,xx,xy,xz,xt,yy,yz,yt,zz,zt,tt and u_i is not equal to u_k.
Use x,y,z,t for 4 independent varibles, x,y,t for 3 independent varibles,and ,x,t for 2 independent varibales

No of independent variable n,LHS,RHS of eq [4, u_t, u_xx + u_yy + u_zz]

```
The determining equations are given by
```

$$\frac{\partial^2}{(\partial u)^2} Y(x,y,z,t,u) = 0$$

$$\frac{\partial^2}{(\partial u)^2} Z(x,y,z,t,u) = 0$$

$$\frac{\partial^2}{(\partial u)^2} T(x,y,z,t,u) = 0$$

Fig. 5

$$\frac{\partial^2}{(\partial x)^2} T(x,y,z,t,u) + \frac{\partial^2}{(\partial y)^2} T(x,y,z,t,u) + \frac{\partial^2}{(\partial z)^2} T(x,y,z,t,u) - \frac{\partial}{\partial t} T(x,y,z,t,u) + 2\frac{\partial}{\partial z} Z(x,y,z,t,u) = 0$$

$$2\frac{\partial}{\partial z} T(x,y,z,t,u) = 0$$

$$-\frac{\partial^2}{(\partial x)^2} U(x,y,z,t,u) - \frac{\partial^2}{(\partial y)^2} U(x,y,z,t,u) - \frac{\partial^2}{(\partial z)^2} U(x,y,z,t,u) + \frac{\partial}{\partial t} U(x,y,z,t,u) = 0$$

```
All determining equations have been found.
End of the program.
```

Fig. 6

Output: We get following set of determining equations using algorithm.

$$
\begin{aligned}
T_u = 0, T_t = 0, T_x = 0, T_y = 0,\\
X_u = 0, Y_u = 0, Z_u = 0,\\
-2U_{xu} + X_{xx} + X_{yy} + X_{zz} - X_t = 0,\\
-U_{uu} = 0,\\
-2U_{yu} + Y_{xx} + Y_{yy} + Y_{zz} - Y_t = 0,\\
-2U_{zu} + Z_{xx} + Z_{yy} + Z_{zz} - Z_t = 0,\\
-T_t + 2X_x = 0,\\
2X_y + 2Y_x = 0,\\
2X_z + 2Z_x = 0,\\
-T_t + 2Y_y = 0,\\
2Y_z + 2Z_y = 0,\\
-T_t + 2Y_z = 0,\\
-2U_{xx} - 2U_{yy} - 2U_{zz} + U_t = 0,
\end{aligned}
\tag{5.12}
$$

Solving above determining equations we get infinitesimals.

$$
\begin{aligned}
X &= \frac{1}{2}[c_{10}t + c_{11}]x + c_1 y + c_3 z + [c_6 t + c_7],\\
Y &= -[c_1 x] + \frac{1}{2}[c_{10}t + c_{11}]y + c_2 z + [c_4 t + c_5],\\
Z &= -[c_2 y] + c_3 x + \frac{1}{2}[c_{10}t + c_{11}]z + [c_8 t + c_9],\\
T &= c_{10}\frac{t^2}{2} + c_{11}t + c_{12},\\
U &= u[-\frac{1}{8}c_{10}x^2 - \frac{1}{8}c_{10}y^2 - \frac{1}{8}c_{10}z^2 - \frac{1}{2}c_6 x - \frac{1}{2}c_4 y - \frac{1}{2}c_8 z - \frac{3}{4}c_{10}t + c_{13}].
\end{aligned}
$$

where $c_1, \ldots, c_{13}$ are constants,and F is any function satisfies $u_t = u_{xx} + u_{yy} + u_{zz}$.

Remark 5.1. *If PDE contains functions other than F, f then codes given in the algorithm can be modified and the new functions can be added in* # *Define function code.*

6 Conclusion

In symmetry method for *PDE*s, finding infinitesimals associated with *PDE* is the key step. The algorithm given in the paper gives the output as the determining equations for finding infinitesimals by giving inputs as *PDE* in two three and four independent variables x,y,z,t and dependent variable u, of order one and two, written in the solved form. The algorithm is very useful for researchers working with *PDE* using Lie symmetry method and open source SageMath software. The results can be extended for higher-order *PDE*s.

References

[1] Arigo Daniel J. (2015), Symmetry Analysis of Differential Equations An Introduction, John Wiley and Sons.

[2] Bluman G.W., Cheviakov A.F. and Anco S.C.(2000) , Applications of symmetry methods to partial differential equations, Springer.

[3] Bluman G.W. and, Kumei S.(1989), Symmetry and Differential equations, Springer-Verlag New York, Inc.

[4] Chaudry Masood Khalique and Karabo Plaatjie and Oageng Lawrence Diteho (2021), Symmetry Solutions and Conservation Laws for the 3D Generalized Potential Yu-Toda-Sasa-Fukuyama Equation of Mathematical Physics, Symmetry, 2058, 13.

[5] Ghanbari B., Kumar S., Niwas M. and Baleanu D.(2021), The Lie symmetry analysis and exact Jacobi elliptic solutions for the Kawahara-KdV type equations Results, Phys, 23, pp. 1-15.

[6] Hans Stephani (1989), Differential equations their solutions using symmetry, Cambridge University Press.

[7] Hydon Peter E. (2005), Symmetry Methods for Differential Equations, Cambrige University Press.

[8] Ibragimov N.H. (1994), CRC Handbook of Lie Group Analysis of Differential Equations Symmetries, Exact Solutions, and Conservation Laws, Boca Raton,Vol.I, CRC Press.

[9] Kumar S. and Kumar A. (2019), Lie symmetry reductions and group invariant solutions of (2+1)-dimensional modified Veronese web equation, Nonlinear Dynam, 98, pp.1891-1903.

[10] Kumar S., Ma W.X. and Kumar A.(2021), Lie symmetries, optimal system and group-invariant solutions of the (3+1)-dimensional generalized KP equation, Chinese Journal of Physics, 69, pp. 1-23.

[11] Kumar Sachin and Setu Rani (2021), Lie symmetry analysis, group-invariant solutions and dynamics of solitons to the (2 + 1)-dimensional BogoyavlenskiiSchieff equation, Pramana Journal of Physics, Springer, 95.

[12] Kumar Sachin and Setu Rani (2021), Invariance analysis, optimal system, closed-form solutions and dynamical wave structures of a (2+1)-dimensional dissipative long wave system, Physica Scripta, 96 .

[13] Lawrence Dresner (1998), Applications of Lie's Theory of Ordinary and Partial Differential Equations, CRC Press.

[14] Olver P.J. (1993), Applications of Lie groups to differential equations (2nd ed.), Graduate Texts in Mathematics, 107, Springer-Verlag, New York.

[15] Seshadri R. and Na T.Y.(1985), Group invariance in engineering boundary value problems, Springer-Verlag.