

Learning about Differential Equations from Their Symmetries

Scott A. Herod

Symmetries form the basis of the packages `DSolve` and `PDSolve1`. A knowledge of symmetries facilitates the understanding and analysis of solutions of differential equations and an understanding of the techniques used to solve the equations. This article introduces a package, `MathSym`, that assists in the computation of symmetries. `MathSym`'s flexibility and utility are illustrated through four examples, two from ordinary differential equations and two from partial differential equations.

■ Introduction

Generally, we teach our sophomore engineering students a course in techniques for solving differential equations. It often appears to the students that we are offering a disjoint collection of methods that we expect them to apply to a sequence of contrived examples. Later in their education many of these students are introduced to partial differential equations and again they are besieged by a multitude of techniques and “special” solutions. Almost all of the methods that we teach can be derived from one basic idea: the existence of symmetries of differential equations.

In the late 1870s Sophus Lie, working at the University of Christiania (now Oslo), applied his theories of *transformation groups* to differential equations. It was well known at the time that Abel's theory on the roots of polynomials was best understood in terms of the theories of Galois regarding symmetries of polynomial equations. Lie believed that his theories of continuous transformation groups would lead to a similar description of solutions to differential equations. He soon realized that many of the standard techniques for solving differential equations could be subordinated to a general method. In [1] Lie wrote that “the foundation of this method is the concept of an infinitesimal transformation and closely related to it the concept of a one-parameter group” (translation by F. Schwartz in [2]).

The transformations that Lie was writing about are usually called *symmetries*. They are found by posing the question: “What are the transformations of the

variables in a differential equation that map solutions of the equation to other solutions of the equation?" This is the fundamental question that we should always keep in mind when discussing the symmetries of a differential equation.

A symmetry of an ordinary differential equation can be used to reduce the order of the equation. My first example shows how this may be done. Also, Alexei Bocharov discusses *Mathematica's* implementation of this technique in [3]. For partial differential equations, I will show how knowledge of a symmetry can be used to reduce the number of independent variables by one. So, for example, an equation in two independent variables can be converted to an ordinary differential equation.

Answering the question of what are the possible symmetries of any given differential equation leads to a massive computational problem. For this reason Lie's technique was rarely used until recently. The advent of powerful computer algebra systems has made it feasible to carry out such long calculations. *MathSym* is a *Mathematica* package that performs many of the computations necessary to apply Lie's technique. In addition, it incorporates ideas from Gröbner bases to reduce the equations, which must be solved in order to compute the symmetries of an equation.

I will not give a full description of the method of symmetry reduction of differential equations. Rather, I suggest the following references for someone interested in learning more about these techniques. The mathematical details of Lie's technique may be found in [4, 5, 6, 7]. A description of the use of symmetries in *Mathematica* appears in [3]. For a discussion of Gröbner bases, see [8] and [9]. Application of symmetry reduction to two equations from fluid mechanics may be found in [10].

■ Reductions of Differential Equations: Scaling Symmetries

As an example of Lie's technique, consider the standard linear second-order ordinary differential equation that we discuss in sophomore differential equations,

$$y''(x) + a y'(x) + b y(x) = 0. \quad (1)$$

If we rescale y by an arbitrary constant α , making the change of variable $\tilde{y} = \alpha y$, we are left with exactly the same equation,

$$\alpha \tilde{y}''(x) + \alpha a \tilde{y}'(x) + \alpha b \tilde{y}(x) = 0.$$

We say in this case that equation (1) is *scale invariant* and that we have identified a *scaling symmetry* of the equation.

In order to apply this scale symmetry to help us solve equation (1), we rewrite the equation as a pair of first-order equations,

$$u'(x) = v(x) \quad (2)$$

$$v'(x) = -a v(x) - b u(x), \quad (3)$$

and then ask: “Are there any quantities that are left invariant by the scaling symmetry?” Computation of such an invariant is algorithmic, but we will merely notice that $w = \tilde{v} / \tilde{u} = (\alpha v) / (\alpha u)$ is identical in both the original and in the new coordinate system. Assuming that w is a function of x , we next write the differential equations that $w(x)$ must satisfy. We do this by replacing v with the product $u w$ in equations (2) and (3) to get

$$u'(x) = v(x) = w(x) u(x) \quad (4)$$

and

$$v'(x) = u'(x) w(x) + w'(x) u(x) = u(x) w(x)^2 + w'(x) u(x) = -a u(x) w(x) - b u(x).$$

This implies that

$$w'(x) = -b - a w(x) - w(x)^2. \quad (5)$$

We have converted the original problem to two integrals, namely

$$\int \frac{dw}{b + a w + w^2} = - \int dx$$

and, once $w(x)$ is known,

$$\int \frac{du}{u} = \int w(x) dx.$$

However, we recognize that we can actually say a lot about equation (5). Namely, it is the Riccati equation that corresponds to equation (1). Equation (4) is the Riccati transformation. Furthermore, if we choose the two special solutions to equation (5) given by the roots of the right-hand side and then solve equation (4) for $u(x)$, we have recovered the standard technique for solving linear constant coefficient differential equations that is taught in the sophomore course.

Our second example is the heat equation in one spatial dimension,

$$u_t = u_{xx}. \quad (6)$$

It is preserved by the change of variables given by the scalings $\tilde{u} = \beta u$, $\tilde{x} = \alpha x$, and $\tilde{t} = \alpha^2 t$. If we again search for quantities that are invariant under these scalings, we discover the new variables $s = -x^2 / (4t)$ and $w = u / t^{\gamma/2}$ where γ satisfies $\beta = \alpha^\gamma$. We assume that w is a function of s and compute the equation for w that arises by insisting that u be a solution of the heat equation. Setting $u(x, t) = t^{\gamma/2} w(s)$ and substituting into equation (6) yields an ordinary differential equation for w ,

$$s w''(s) + \left(\frac{1}{2} - s \right) w'(s) + \frac{\gamma}{2} w(s) = 0. \quad (7)$$

This equation is Kummer's equation and the solution of it may be expressed in terms of confluent hypergeometric functions, $M(-\frac{\gamma}{2}, \frac{1}{2}, s)$ and $U(-\frac{\gamma}{2}, \frac{1}{2}, s)$. *Mathematica* knows these two functions as `Hypergeometric1F1[- $\frac{\gamma}{2}$, $\frac{1}{2}$, s]` and `HypergeometricU[- $\frac{\gamma}{2}$, $\frac{1}{2}$, s]`. We can use solutions to Kummer's equation to plot scale invariant solutions of the heat equation. Figure 1 is the result of setting $\gamma = -12.1$. It was generated with the following routine.

```

In[1]:= dl[g_] := DensityPlot[
  t^(g/2) Hypergeometric1F1[-g/2, 1/2, (-x^2)/(4 t)], {x, -3, 3},
  {t, 0.001, 0.6}, PlotPoints -> 250, FrameLabel -> {"x", "t", "", ""},
  ColorFunction -> (GrayLevel[#] &), Mesh -> False];
dl[-12.1]

```

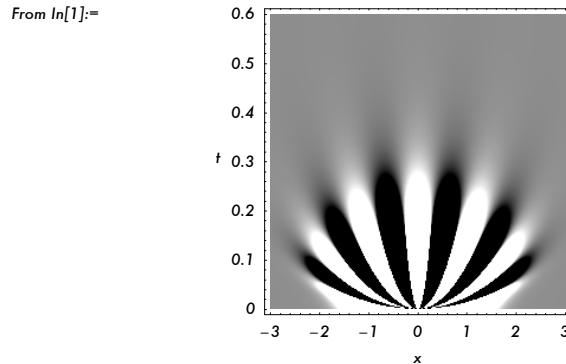


Figure 1. A scale invariant solution of the heat equation.

The horizontal axis gives the space variable x and the vertical axis is t . White sections are high temperature regions while dark regions are cooler.

■ Application of MathSym to Analyzing an Ordinary Differential Equation

In the previous section we used a scaling symmetry to help understand the solutions of a pair of differential equations. In each case, the scaling symmetry was found by inspection. Here I present the computation of the complete set of *point symmetries* for two additional differential equations. Our third example is a nonlinear ordinary differential that we analyze using its two symmetries. The final example is the partial differential equation known as the cubic nonlinear Schrödinger equation [11].

Example three is the ordinary differential equation

$$w''(x) + a w(x) w'(x) + b w(x)^3 = 0 \quad (8)$$

that arises in the study of nonlinear water wave equations. I also show that we can use its two symmetries to begin to learn something about the structure of its solutions.

MathSym returns a system of equations, the *determining equations*, whose solutions generate the symmetries of equation (8). Internally, the MathSym package denotes all independent variables in an equation as x_i and dependent variables as u_i . This way it can be run on systems of equations with arbitrary numbers of independent and dependent variables without needing to know how to treat

different variable names. Furthermore, constants are represented as $s[i]$ internally and printed as $s(i)$. With this notation, constants are treated correctly by *Mathematica*'s differentiation routine `Dt`. `MathSym`'s output is the following list of determining equations.

$$\begin{aligned} -\frac{\partial^2 \xi_1}{\partial u_1^2} &= 0 \\ 2 u_1 s(1) \frac{\partial \xi_1}{\partial u_1} + \frac{\partial^2 \eta_1}{\partial u_1^2} - 2 \frac{\partial^2 \xi_1}{\partial x_1} \partial u_1 &= 0 \\ 3 u_1^2 \eta_1 s(2) - u_1^3 s(2) \frac{\partial \eta_1}{\partial u_1} + u_1 s(1) \frac{\partial \eta_1}{\partial x_1} + 2 u_1^3 s(2) \frac{\partial \xi_1}{\partial x_1} + \frac{\partial^2 \eta_1}{\partial x_1^2} &= 0 \\ \eta_1 s(1) + 3 u_1^3 s(2) \frac{\partial \xi_1}{\partial u_1} + u_1 s(1) \frac{\partial \xi_1}{\partial x_1} + 2 \frac{\partial^2 \eta_1}{\partial x_1} \partial u_1 - \frac{\partial^2 \xi_1}{\partial x_1^2} &= 0 \end{aligned}$$

With the output from `MathSym` we can continue our analysis of equation (8). First, we solve the determining equations:

$$\xi_1 = c_1 + c_2 x_1 \quad (9)$$

$$\eta_1 = -c_2 u_1. \quad (10)$$

The functions ξ_1 and η_1 determine two symmetries that can be used to convert equation (8) into two integrals. The reader is directed to similar computations for the Blasius boundary layer equation which appear on pages 118–120 of [4].

We begin by considering the symmetry that occurs because of the c_1 term. Setting $\xi_1 = c_1$ and $\eta_1 = 0$ produces a transformation $\tilde{x} = x + c_1 \varepsilon$ and $\tilde{w} = w$. We next look for two quantities that do not change under this transformation. Obvious choices are $u = w$ and $v = w'$. If we assume that v is a function of u and write the differential equation for $v(u)$ that arises by insisting that $w(x)$ satisfy equation (8), we find

$$v(u) v'(u) + a u v(u) + b u^3 = 0. \quad (11)$$

This is a standard reduction of order for autonomous equations that may be found in a sophomore differential equations text such as [12].

This equation in u and v has a symmetry that is generated by the constant c_2 appearing in equations (9) and (10). From this symmetry we can derive new variables $r = v/u^2$ and $s = \ln v$ and consider s as a function of r . In terms of r and $s(r)$, equation (11) becomes

$$\frac{ds}{dr} = \frac{ar + b}{2r^3 + ar^2 + br}. \quad (12)$$

We have now converted the problem of solving the original equation into two integrations. First we find s as a function of r giving us a solution of equation (12) and hence of equation (11). Then we return to the original variables and have $w'(x)$ implicitly as a function of $w(x)$. Integrating again gives a relationship between w and x .

We can make *Mathematica* carry out some of these computations. First we will ask that it determine a solution to equation (12) by integrating both sides of the equation.

```
In[3]:= soln = (Integrate[s' [r], r] ==
             Integrate[(a r + b) / (2 r^3 + a r^2 + b r), r] + C[1])
```

$$\text{Out[3]= } s[r] == \frac{a \operatorname{ArcTan}\left[\frac{a+4r}{\sqrt{-a^2+8b}}\right]}{\sqrt{-a^2+8b}} + C[1] + \operatorname{Log}[r] - \frac{1}{2} \operatorname{Log}[b + ar + 2r^2]$$

In the equation for $s(r)$ we can return to the original variables $w(x)$ and $w'(x)$.

```
In[4]:= soln /. {s[r] -> Log[w' [x]], r -> w' [x] / w[x]^2}
```

```
Out[4]= Log[w' [x]] ==
```

$$\frac{a \operatorname{ArcTan}\left[\frac{a + \frac{4w'[x]}{w[x]^2}}{\sqrt{-a^2+8b}}\right]}{\sqrt{-a^2+8b}} + C[1] + \operatorname{Log}\left[\frac{w'[x]}{w[x]^2}\right] - \frac{1}{2} \operatorname{Log}\left[b + \frac{aw'[x]}{w[x]^2} + \frac{2w'[x]^2}{w[x]^4}\right]$$

What results is an implicit relationship between w and w' , and while *MathSym* has been successful in generating the symmetries of equation (8), it still is a challenge to solve this equation.

■ Application of *MathSym* to Analyzing a Partial Differential Equation

As a final example, I will discuss the computation of the symmetries of the cubic nonlinear Schrödinger equation, a complex valued partial differential equation. Using the symmetries of the equation it is possible to generate exact solutions by a couple of different methods. First since symmetries of an equation transform its solutions to other solutions, I will demonstrate a family of solutions that are the transforms of a spatially invariant solution. Also, I will show that the symmetries can be used to reduce the partial differential equation to an ordinary differential equation in a couple of different ways.

For ordinary differential equations, it is theoretically possible to reduce the order of the equation by one for every symmetry of the equation. So, a second-order equation can be reduced to two integrals if two symmetries are known. Of course, as we saw above, in practice it can be difficult to perform the necessary computations. For a partial differential equation, it is generally not possible to get the full solution set from just the knowledge of several symmetries. However, by looking for fixed points of a given symmetry, we can find reductions of the equation and often special solutions.

The cubic nonlinear Schrödinger equation is derived in descriptions of nonlinear optics, water waves, and plasma physics [11]. Usually, the equation is written in terms of a complex valued function $\Psi(x, t)$. The equation is

$$i\Psi_t + \Psi_{xx} + 2|\Psi|^2\Psi = 0.$$

When computing the symmetries, we set $\Psi(x, t) = u(x, t) + i v(x, t)$ and use the system of equations

$$-v_t + u_{x x} + 2(u^2 + v^2)u = 0$$

$$u_t + v_{x x} + 2(u^2 + v^2)v = 0$$

which we denote as NLS.

Again, `MathSym` returns a system of linear coupled partial differential equations for the generators of the symmetries. `MathSym` uses the convention that the independent variables x and t are stored and printed as x_1 and x_2 . The dependent variables u and v are represented by u_1 and u_2 . The generators of the symmetries are ξ_1 , ξ_2 , η_1 , and η_2 .

$$-(\eta_2 u_1) + \eta_1 u_2 - u_1^2 \frac{\partial \eta_1}{\partial u_2} - u_2^2 \frac{\partial \eta_1}{\partial u_2} = 0$$

$$\eta_1 u_1 + \eta_2 u_2 - u_1^2 \frac{\partial \eta_2}{\partial u_2} - u_2^2 \frac{\partial \eta_2}{\partial u_2} = 0$$

$$\frac{\partial \xi_1}{\partial u_2} = 0$$

$$\frac{\partial \xi_2}{\partial u_2} = 0$$

$$-(\eta_1 u_1) - \eta_2 u_2 + u_1^2 \frac{\partial \eta_1}{\partial u_1} + u_2^2 \frac{\partial \eta_1}{\partial u_1} = 0$$

$$\eta_2 u_1 - \eta_1 u_2 - u_1^2 \frac{\partial \eta_2}{\partial u_1} - u_2^2 \frac{\partial \eta_2}{\partial u_1} = 0$$

$$\frac{\partial \xi_1}{\partial u_1} = 0$$

$$\frac{\partial \xi_2}{\partial u_1} = 0$$

$$\frac{\partial \eta_1}{\partial x_2} = 0$$

$$\frac{\partial \eta_2}{\partial x_2} = 0$$

$$2 \eta_1 u_1 + 2 \eta_2 u_2 + u_1^2 \frac{\partial \xi_2}{\partial x_2} + u_2^2 \frac{\partial \xi_2}{\partial x_2} = 0$$

$$u_2 \frac{\partial \xi_1}{\partial x_2} + 2 \frac{\partial \eta_1}{\partial x_1} = 0$$

$$u_1 \frac{\partial \eta_1}{\partial x_1} + u_2 \frac{\partial \eta_2}{\partial x_1} = 0$$

$$\eta_1 u_1 + \eta_2 u_2 + u_1^2 \frac{\partial \xi_1}{\partial x_1} + u_2^2 \frac{\partial \xi_1}{\partial x_1} = 0$$

$$\frac{\partial \xi_2}{\partial x_1} = 0$$

$$\frac{\partial^2 \eta_1}{\partial x_1^2} = 0$$

MathSym has been successful in generating and reducing these equations. The solution of the determining equations is

$$\xi_1 = c_0 + c_1 x + 2 c_2 t$$

$$\xi_2 = c_3 + 2 c_1 t$$

$$\eta_1 = c_4 v - c_1 u - c_2 x v$$

$$\eta_2 = -c_4 u - c_1 v + c_2 x u$$

where the c_i are arbitrary constants. For clarity we have returned to the original variables x , t , u , and v . From these generators we can compute the symmetries for the nonlinear Schrödinger equation and with the symmetries we can derive solutions to NLS using at least two different strategies.

Recall that symmetries map solutions to solutions so knowledge of a solution and a symmetry allows us to generate a family of new solutions. As an example, let us use the *Galilean boost*, which is represented above by the constant c_2 . We can compute a transformation by setting $c_2 = 1$ and all of the other constants in the list of generators for the symmetries to zero. The transformation corresponding to this choice of constants is given by the change to the new variables \tilde{x} , \tilde{t} , \tilde{u} , and \tilde{v} where

$$\tilde{x} = x + 2 \varepsilon t \quad (13)$$

$$\tilde{t} = t \quad (14)$$

$$\tilde{u} = \sqrt{u^2 + v^2} \cos\left(\frac{1}{2} t \varepsilon^2 + x \varepsilon + \tan^{-1}(v/u)\right) \quad (15)$$

$$\tilde{v} = \sqrt{u^2 + v^2} \sin\left(\frac{1}{2} t \varepsilon^2 + x \varepsilon + \tan^{-1}(v/u)\right). \quad (16)$$

The point in this exercise was to generate transformations of the variables appearing in the nonlinear Schrödinger equation that sent solutions of the equation to other solutions. This means that any solution to NLS can be used to generate a family of solutions.

It is straightforward to check that a solution of NLS is

$$u(t) = k \cos(2 k^2 t) \quad (17)$$

$$v(t) = k \sin(2 k^2 t) \quad (18)$$

where k is an arbitrary constant. This solution is a prototypical nonlinear oscillator where the frequency is a function of the amplitude.

The transformation in equations (13) to (16) gives new solutions to NLS when applied to equations (17) and (18). If we carry out these substitutions, we find

$$\begin{aligned}\tilde{u}(\tilde{t}) &= k \cos\left(2 k^2 \tilde{t} + \varepsilon \tilde{x} - \frac{1}{2} \varepsilon^2 \tilde{t}\right) \\ \tilde{v}(\tilde{t}) &= k \sin\left(2 k^2 \tilde{t} + \varepsilon \tilde{x} - \frac{1}{2} \varepsilon^2 \tilde{t}\right)\end{aligned}$$

which can be shown to satisfy NLS for any choices of k and ε . Images of the real parts for these two solutions are generated in the following cells.

```
In[5]:= Plot3D[Cos[2 t],{x,-10,10},{t,0,10},PlotPoints->50,
  AxesLabel->{"x","t","u(x,t)",Mesh->False,
  PlotRange->{-2,2}]
```

From In[5]:=

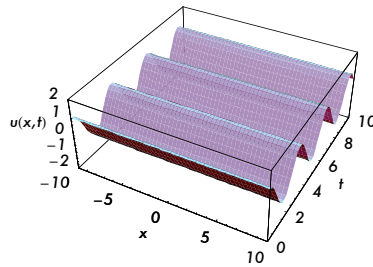


Figure 2. The real part of a spatially invariant solution of the nonlinear Schrödinger equation.

```
In[6]:= Plot3D[Cos[2 t + .3 x - (.3)^2/2 t],{x,-10,10},{t,0,10},
  PlotPoints->50,
  AxesLabel->{"x","t","u(x,t)",Mesh->False,
  PlotRange->{-2,2}]
```

From In[6]:=

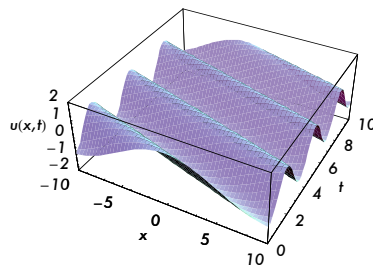


Figure 3. The real part of a transformed solution of the nonlinear Schrödinger equation.

Finally, we will use the symmetries to derive ordinary differential equations. In order to compute the reductions, we pose the question: “What are the solutions

of NLS that are invariant under a given symmetry?” Notice that this question is similar to the one that we asked to define the symmetries. A symmetry is a mapping of the solution set of an equation to itself. Reductions arise from looking at the invariant subset of that transformation.

To find these invariant solutions we look for the intersection of the solution sets of the original equation, here the NLS equation, and a pair of first-order, quasi-linear partial differential equations, which are

$$\begin{aligned}\xi_1 \frac{\partial u}{\partial x} + \xi_2 \frac{\partial u}{\partial t} &= \eta_1 \\ \xi_2 \frac{\partial v}{\partial x} + \xi_2 \frac{\partial v}{\partial t} &= \eta_2.\end{aligned}$$

These equations are known as the *invariant surface condition equations*. We will solve this new pair of equations using the method of characteristics and then substitute the solutions into the NLS equation.

We start with $\xi_1 = c_0$, $\xi_2 = c_3$, and $\eta_1 = \eta_2 = 0$. When we solve the invariant surface equations, we find that along characteristic curves, $s = x - ct$, u and v must be constant. Thus, $u(x, t) = f(s)$ and $v(x, t) = g(s)$ for arbitrary functions f and g . Here $c = c_0/c_3$. The functions f and g must be determined by the NLS equation. You perhaps recognize f and g as being the functions that one must find in order to compute the traveling wave solutions of the NLS equation.

Substituting $u(x, t) = f(s)$ and $v(x, t) = g(s)$ into the NLS equation gives us two equations that we must solve in order to find f and g . These two new equations are now ordinary differential equations rather than the original partial differential equations. They are

$$\begin{aligned}c g' + f'' + 2(f^2 + g^2) f &= 0 \\ -c f' + g'' + 2(f^2 + g^2) g &= 0.\end{aligned}$$

We can approximate solutions to this pair of ordinary differential equations by changing to polar coordinates $r(s)^2 = f(s)^2 + g(s)^2$ and $\theta(s) = \tan^{-1}(g(s)/f(s))$ and integrating twice. The new equations are

$$\begin{aligned}\frac{1}{2} (r')^2 + \frac{1}{2} \left[r^4 + \left(\frac{c}{2}\right)^2 r^2 + \left(\frac{k}{r}\right)^2 \right] &= T \\ \theta' &= \frac{c}{2} + \frac{k}{r^2}\end{aligned}$$

with k and T constants of the integration. The form of the equation for $r(s)$ was chosen on purpose to remind us of the equation from classical mechanics,

$$\frac{1}{2} m v^2 + \text{P.E.} = \text{Energy.} \quad (19)$$

If we let $\xi_1 = c_1 x$, $\xi_2 = 2 c_1 t$, $\eta_1 = -c_1 u$, and $\eta_2 = -c_1 v$, we find that, along characteristic curves $s = x^2/t$, invariant solutions of the NLS equation multiplied by x must be constant. That is, $x u(x, t) = f(s)$ and $x v(x, t) = g(s)$. Note that these are different f , g , and s than those that we just used. Substituting into the

NLS equation gives a pair of coupled ordinary differential equations that f and g must solve. This time these equations are

$$\begin{aligned} s^2 g' + 4s^2 f'' + 2f - 2s f' + (f^2 + g^2) f &= 0 \\ -s^2 f' + 4s^2 g'' + 2g - 2s g' + (f^2 + g^2) g &= 0. \end{aligned}$$

Our last reduction will come from letting $\xi_1 = 2t$, $\xi_2 = 0$, $\eta_1 = -xv$, and $\eta_2 = xu$. We will switch to a polar representation immediately here. The resulting pair of ordinary differential equations that we derive are

$$\begin{aligned} 2sr'(s) + r(s) &= 0 \\ \left(\frac{\theta(s)}{s}\right)' + 8r(s)^2 &= 0. \end{aligned}$$

We can solve these two equations to get

$$\begin{aligned} r(s) &= \frac{c_1}{\sqrt{s}} \\ \theta(s) &= -4c_2 s - 8c_1 s \ln s. \end{aligned}$$

In terms of the original variables this is

$$\begin{aligned} u(x, t) &= \frac{c_1}{\sqrt{t}} \cos\left(\frac{x^2}{4t} + c_2 + 2c_1 \ln t\right) \\ v(x, t) &= \frac{c_1}{\sqrt{t}} \sin\left(\frac{x^2}{4t} + c_2 + 2c_1 \ln t\right). \end{aligned}$$

Here it is nice to have *Mathematica* check our computations. First we will derive the equations. Notice that we are using a little i instead of the capital I that *Mathematica* usually uses. This is done so that we can separate using a `CoefficientList`.

```
In[7]:= SetAttributes[{i, x, t}, Constant];
eqn = (i Dt[psi, t] + Dt[psi, x, x] + 2 Abs[psi]^2 psi);
eqn = (eqn /. psi -> u[x, t] + i v[x, t]) // Factor;
eqn = eqn /. Abs[_] -> Sqrt[u[x, t]^2 + v[x, t]^2];
eqn = CoefficientList[(eqn /. i^2 -> -1), i]

Out[11]= {2 u[x, t] (u[x, t]^2 + v[x, t]^2) - v^(0,1)[x, t] + u^(2,0)[x, t],
          2 v[x, t] (u[x, t]^2 + v[x, t]^2) + u^(0,1)[x, t] + v^(2,0)[x, t]}
```

Before we substitute for u and v , we need to “undo” the differentiation that occurred. It would also be nice to strip the arguments x and t from u and v on the cubic term. We will do this using two functions `UnDt` and `UnArg`. Here is their syntax.

```
In[12]:= UnList[{x__}] := Sequence[x];
UnDt[f_] := f /. Derivative[order__][func_][vars__] =>
Dt[func, UnList[Transpose[{List[vars], List[order]}]]];
UnArg[stuff_, func_] := stuff /. func[___] => func;
UnArg[stuff_, {func_}] := UnArg[stuff, func];
UnArg[stuff_, {func_, morefuncs__}] :=
UnArg[UnArg[stuff, func], {morefuncs}];
```

We apply these two functions.

```
In[17]:= eqn = UnArg[UnDt[eqn], {u, v}]
Out[17]= {2 u (u^2 + v^2) + Dt[u, {x, 2}] - Dt[v, t],
          2 v (u^2 + v^2) + Dt[u, t] + Dt[v, {x, 2}]}
```

We can now substitute our solution in and run the result through Simplify.

```
In[18]:= eqn /. {u -> (c[1] / Sqrt[t]) Cos[c[2] + 2 c[1]^2 Log[t] + x^2 / (4 t)], v ->
             (c[1] / Sqrt[t]) Sin[c[2] + 2 c[1]^2 Log[t] + x^2 / (4 t)]} // Simplify
Out[18]= {0, 0}
```

So it works. Let us see what some pictures look like. The constant $c[2]$ is only a phase shift so we will set it to zero. $c[1]$ affects the amplitude and frequency and we will let it be one. Here is the command to generate a density plot of the real part of the solution that is given by $u(x, t)$.

```
In[19]:= DensityPlot[(1 / Sqrt[t]) Cos[2 Log[t] + x^2 / (4 t)],
                    {x, -10, 10}, {t, 0.01, 3.0}, PlotPoints -> 250, AxesLabel -> {"x", "t"},
                    FrameLabel -> {"x", "t", "", ""},
                    ColorFunction -> (GrayLevel[#] &), Mesh -> False]
```

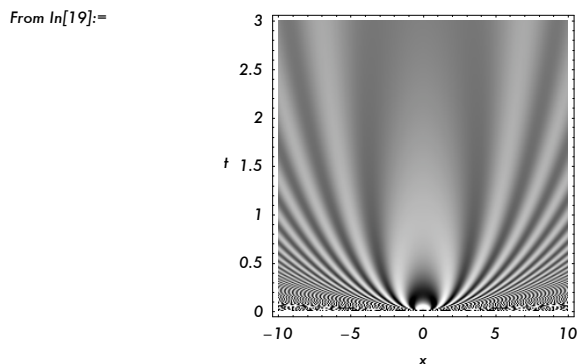


Figure 4. The real part of a boost invariant solution of the nonlinear Schrödinger equation.

■ Closing Comments

Symmetries of differential equations are useful in understanding their solutions and as a way to understand the techniques that we use to find the solutions. We have shown how some of these techniques can be derived from the symmetries of the equations and how generalizations of the methods for solving ordinary differential equations lead to exact solutions for a couple of partial differential equations.

To assist in determining the symmetries for equations, we have also introduced a package `MathSym` that performs many of the calculations. In addition to the

calculation of the Lie symmetries for differential equations that we have demonstrated here, `MathSym` can use ideas from differential Gröbner bases to generate a reduced form of the determining equations. For a discussion of algorithms similar to the ones that we have used, see [9].

`MathSym` can also generate the determining equations for the conditional symmetries first introduced by Bluman and Cole in 1969 [13]. Reductions with respect to conditional symmetries include the reductions derived using the *direct method* of Clarkson and Kruskal [14].

Finally, I do not consider `MathSym` a finished product. There are generalizations of Lie's technique that the package does not currently address. A lot of recent work in symmetry techniques for differential equations focuses on *generalized* symmetries and symmetries of difference equations. It is my intention to incorporate some of these ideas into future versions of the `MathSym` package.

■ Input Files

`MathSym` uses input files for the information that it needs. I decided to do this because the amount of information that the package needs is reasonably large, and I have found it useful to keep a record of all of the necessary information needed to run a particular problem.

The first two input files do not include the three optional arguments that occur in the input file for the nonlinear Schrödinger equation. The defaults for these variables are discussed in the documentation, which is part of the standard distribution of `MathSym`. The first listing is for the example of the heat equation discussed in the first section, Reductions of Differential Equations: Scaling Symmetries. It is also included in Additional Material in the file named `Heat_eqn.m`. The second listing contains the information for the ordinary differential equation described in Application of `MathSym` to Analyzing an Ordinary Differential Equation, and is contained in the file `ODE_eqn.m`. Finally, the input file for the nonlinear Schrödinger equation from the section Application of `MathSym` to Analyzing a Partial Differential Equation is listed in the third input cell. This information is also contained in the file `NLS_eqn.m` in Additional Material.

```
In[20]:= eqname = "the 1x1 Dimensional Heat Equation equation";
         numeqn = 1;
         numind = 2;
         numdep = 1;
         numpar = 0;
         diffeqn = {Dt[u1, x[2]] - Dt[u1, {x[1], 2}]};
         removedvar = {Dt[u1, {x[1], 2}]};
```

```

In[27]:= eqname = "a nonlinear ODE ";
         numeqn = 1;
         numind = 1;
         numdep = 1;
         numpar = 0;
         diffeqn = {Dt[u1, x[1], x[1]] + s[1] u1 Dt[u1, x[1]] + s[2] u1^3};
         removedvar = {Dt[u1, x[1], x[1]]};

In[34]:= eqname = "the Nonlinear Schroedinger equation";
         tech = "Classical";
         splitlist = True;
         sortstyle = "Lexicographical";
         numeqn = 2;
         numind = 2;
         numdep = 2;
         numpar = 0;
         diffeqn = {Dt[u1, {x[1], 2}] - Dt[u2, x[2]] + 2 (u1 * u1 + u2 * u2) * u1,
                   Dt[u2, {x[1], 2}] + Dt[u1, x[2]] + 2 (u1 * u1 + u2 * u2) * u2};
         removedvar = {Dt[u1, {x[1], 2}], Dt[u2, {x[1], 2}]};

```

■ Acknowledgments

I wish to thank Willy Hereman and Elizabeth Mansfield for discussions of computer techniques used when computing symmetries and Mark J. Ablowitz and Peter Clarkson for comments on the use of symmetries of differential equations. Thanks to James H. Curry for reading and commenting on numerous versions of this manuscript. Finally, I wish to thank the staff at Wolfram Research, especially Alexei Bocharov, for help with the intricacies of *Mathematica*.

Supported in part by the Department of Energy under grant DOE DE-FG03-94ER25194.

■ References

- [1] S. Lie, "Über die Integration durch bestimmte Integrale von einer Klasse linearer partieller Differentialgleichungen," *Arch. for Math.*, **VI**, Heft 3, 1881 S. 328–368.
- [2] F. Schwartz, "Symmetries of Differential Equations: From Sophus Lie to Computer Algebra," *Siam Review*, **30**, 1988 pp. 450–481.
- [3] A. V. Bocharov, "Symbolic Solvers for Nonlinear Differential Equations," *The Mathematica Journal*, **3**(2), 1993 pp. 63–69.
- [4] G. W. Bluman and S. Kumei, *Symmetries and Differential Equations*, New York: Springer-Verlag, 1989.
- [5] P. J. Olver, *Applications of Lie Groups to Differential Equations*, New York: Springer-Verlag, 1986.
- [6] L. V. Ovsiannikov, *Group Analysis of Differential Equations* (W. F. Ames, ed.), New York: Academic Press, 1982.

- [7] N. H. Ibragimov, *Lie Group Analysis of Differential Equations*, Boca Raton: CRC Press, 1994.
- [8] G. Helzer, "Grobner Bases," *The Mathematica Journal*, **5**(1), 1995 pp. 61–73.
- [9] F. Schwartz, "An Algorithm for Determining the Size of Symmetry Groups," *Computing*, **49**, 1992 pp. 95–115.
- [10] S. A. Herod, "Families of Exact Solutions for the Barotropic Vorticity Equation," *PAM Technical Report*, 1995.
- [11] M. J. Ablowitz and P. A. Clarkson, *Solitons, Nonlinear Evolution Equations, and Inverse Scattering*, Cambridge: Cambridge University Press, 1991.
- [12] A. L. Rabenstein, *Elementary Differential Equations with Linear Algebra*, New York: Academic Press, 1975.
- [13] G. W. Bluman and J. D. Cole, "The General Solution of the Heat Equation," *Journal of Mathematical Mechanics*, **18**(11), 1969 pp. 1025–1042.
- [14] P. A. Clarkson and M. D. Kruskal, "New Similarity Solutions of the Boussinesq Equation," *Journal of Mathematical Physics*, **30**, 1989 pp. 2201–2213.

■ Additional Material

MathSym11.m
Heat_eqn.m
ODE_eqn.m
NLS_eqn.m
Heat_run.nb
NLS_run.nb
README
Documentation.nb

Available at www.mathematica-journal.com.

About the Author

Scott A. Herod did postdoctoral research at the University of Colorado in Boulder where his interests included differential equations, dynamical systems, and computational mathematics. He now works as a software consultant specializing in the video industry.

Scott A. Herod
Department of Applied Mathematics
University of Colorado, Boulder
Campus Box 526
Boulder, Colorado 80309-0526
scott.herod@colorado.edu