Chapter for Book Michael Wester

Chapter for Book Michael Wester

First Edition, Wiley and Sons, New York (1999)

Willy Hereman[1] and Ünal Göktaş[2]

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto · Singapore

[1] Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401. Email: whereman@mines.edu
[2] Wolfram Research Inc., 100 Trade Center Drive, Champaign, IL 61820. Email: unalg@wolfram.com

wester 15/11/1999 16:45—PAGE PROOFS for John Wiley & Sons Ltd (31x47jw.sty v5.0, 15th April 1997)

Contents

1

Integrability Tests for Nonlinear Evolution Equations

1.1 Introduction

During the last three decades, the study of integrability of nonlinear ordinary and partial differential equations (ODEs and PDEs) has been the topic of major research projects (see, e.g., [1, 28]). This chapter presents a few symbolic algorithms to illustrate how computer algebra systems (CASs) can be effectively used in integrability investigations. We work with *Mathematica* [42], but our algorithms can be implemented in other languages.

Among the many alternatives [20] for investigating the integrability of systems of PDEs with symbolic software, the search for conserved densities, generalized symmetries, and recursion operators is particularly appealing [28]. Indeed, it turns out that these quantities can be computed without the use of sophisticated mathematical tools. As a matter of fact, not much beyond differentiation and solving of linear systems is needed. As a result, our algorithms are easy to implement. In fairness, our algorithms are restricted to the computation of polynomial quantities of polynomial equations. Yet, this covers the majority of the cases treated in the literature.

The algorithms in this chapter are based on a common principle: scaling (or dilation) invariance. Indeed, we observed that many known integrable systems are invariant under dilation symmetry, which is a special Lie point symmetry. The dilation symmetry can be computed by solving a linear system. Using dilation invariance, the plan is to first produce candidates for the polynomial densities, symmetries, and recursion operators in an efficient way. Once the candidate expressions are computed, their unknown constant coefficients follow from solving a linear system.

We focus our attention on explaining the strategy, at the cost of mathematical rigor and details, which can be found in [14, 15, 17, 23]. Rather than discussing the algorithms in general, we apply them to a few prototypical nonlinear evolution equations from nonlinear wave theory. Whenever appropriate, we address issues related to the implementation of the algorithms. For instance, we give explicit code for the Fréchet derivative, which is one of the key tools in our methods.

Our package *InvariantsSymmetries.m* [18] works for nonlinear evolution equations. Applied to a system with parameters, our package can determine the conditions on the parameters so that the system admits a sequence of conserved densities or generalized symmetries. Although we do not address it here, our package can also compute densities and symmetries of differential-difference equations (semi-discrete lattices). See [16, 17, 19, 23] for more information about that subject.

Due to memory constraints, our software can only compute a limited number of conserved densities and symmetries (half a dozen for systems; at best a dozen for scalar equations). To prove integrability, one must show that infinitely many independent densities or symmetries exist. Alternatively, one could construct the operator that connects the symmetries, and prove that it is a true recursion operator. Such proofs involve mathematical methods [7, 28, 30, 39] that are beyond the scope of this article. Although it is not yet implemented, we also present an algorithm for the computation of recursion operators, based on the knowledge of a few conserved densities and symmetries.

The computation of Lie point symmetries and generalized symmetries via prolongation techniques is purposely omitted. That topic and related software were covered extensively in [21, 22]. Space limitations also prevent the inclusion of the well-known Painlevé test, which is a widely applied and successful integrability detector for nonlinear ODEs and PDEs. We refer to [23] for survey papers, books, and software related to Painlevé analysis.

This chapter is organized as follows. In Section 2, we discuss scaling symmetries of PDEs and show how to compute them. Section 3 deals with conservation laws. We give the definition and the steps of our algorithm, and show how to implement and apply the algorithm. We do the same for generalized symmetries in Section 4. An algorithm to determine the recursion operator is given in Section 5. The leading examples in Sections 2 through 5 are the Korteweg-de Vries (KdV) and Sawada-Kotera (SK) equations, and a system of nonlinear Schrödinger-type equations. For the latter, we derive the recursion operator in Section 6. In Sections 7 and 8, we discuss our *Mathematica* package *InvariantsSymmetries.m* and review similar software. We draw some conclusions in Section 9.

1.2 Key Concept: Dilation Invariance

Our algorithms are based on the following observation: if a system of nonlinear evolution equations is invariant under a dilation (scaling) symmetry, then its conservation laws, generalized symmetries, and the recursion operator have the same scaling properties as the system. This is at least true for the polynomial case.

As leading example, we use the ubiquitous Korteweg-de Vries (KdV) equation [1],

$$u_t = 6uu_x + u_{3x}, \tag{1.1}$$

which describes water and plasma waves and lattice dynamics. Throughout this chapter, we will use the notations

$$u_t = \frac{\partial u}{\partial t}, \quad u_{tx} = \frac{\partial^2 u}{\partial t\, \partial x}, \quad u_{nx} = \frac{\partial^n u}{\partial x^n}. \tag{1.2}$$

Equation (1.1) is invariant under the dilation (scaling) symmetry

$$(t, x, u) \to (t/\lambda^3, x/\lambda, \lambda^2 u), \tag{1.3}$$

where λ is an arbitrary parameter. Indeed, replacement of (t, x, u) according to (1.3) allows one to cancel a factor λ^5 in (1.1). Note that, e.g., $\frac{\partial}{\partial t}$ is replaced by

$\lambda^3 \frac{\partial}{\partial t}$. Obviously, u corresponds to two derivatives in x, i.e., $u \sim \partial^2/\partial x^2$. Similarly, $\partial/\partial t \sim \partial^3/\partial x^3$.

We express all scalings in terms of $\frac{\partial}{\partial x}$. Introducing *weights*, denoted by w, we could say that $w(u) = 2$ and $w(\mathrm{D}_t) = 3$, if we set $w(\mathrm{D}_x) = 1$. We used D_t and D_x instead of $w(\frac{\partial}{\partial t})$ and $w(\frac{\partial}{\partial x})$ to cover cases where densities and symmetries depend explicitly on t and x (see [17, 23] for examples).

The *rank* R of a monomial is equal to the sum of all of its weights. Observe that (1.1) is *uniform in rank* since all the terms have rank $R = 5$, confirming that λ^5 was a common factor.

Computation of scaling symmetries. To compute the scaling symmetry of an equation, we compute the weights of all its terms, and *require* that the equation be uniform in rank. For (1.1), with $w(\mathrm{D}_x) = 1$, this yields

$$w(u) + w(\mathrm{D}_t) = 2w(u) + 1 = w(u) + 3. \tag{1.4}$$

The solution of this linear system is $w(u) = 2$, and $w(\mathrm{D}_t) = 3$.

As a second example, we consider a fifth-order PDE from soliton theory,

$$u_t = 5u^2u_x + 5u_xu_{2x} + 5uu_{3x} + u_{5x}, \tag{1.5}$$

due to Sawada and Kotera [38]. Scaling invariance requires that

$$w(u) + w(\mathrm{D}_t) = 3w(u) + 1 = 2w(u) + 3 = w(u) + 5. \tag{1.6}$$

Hence $w(u) = 2$ and $w(\mathrm{D}_t) = 5$.

Systems. Single PDEs like (1.1) are a special case of

$$\mathbf{u}_t = \mathbf{F}(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{2x}, \ldots, \mathbf{u}_{mx}), \tag{1.7}$$

where $\mathbf{u}$ and $\mathbf{F}$ are vector dynamical variables with n components. The number of components, the order m of the system, and its degree of nonlinearity are arbitrary.

To determine the scaling symmetry, we require that each equation in (1.7) be uniform in rank, and solve the resulting linear system for the weights of all the variables.

As an example, consider a vector nonlinear Schrödinger equation,

$$\mathbf{B}_t + (|\mathbf{B}|^2\mathbf{B})_x + (\mathbf{B}_0 \cdot \mathbf{B}_x)\mathbf{B}_0 + \mathbf{e} \times \mathbf{B}_{xx} = 0, \tag{1.8}$$

which occurs in plasma physics [5, 6]. With $\mathbf{B}_0 = (a, b)$ and $\mathbf{B} = (u, v)$ in the (y, z)-plane, and $\mathbf{e}$ along the x-axis, (1.8) can be written as

$$\begin{aligned} u_t + \left[u(u^2+v^2) + \beta u + \gamma v - v_x\right]_x &= 0, \\ v_t + \left[v(u^2+v^2) + \theta u + \delta v + u_x\right]_x &= 0, \end{aligned} \tag{1.9}$$

where $\beta = a^2, \gamma = \theta = ab$, and $\delta = b^2$ are nonzero parameters. With reference to [5], we call (1.8) or (1.9) the DMV equation. To start generally, we will consider the system (1.9) for arbitrary nonzero parameters β, γ, θ and δ.

System (1.9) is not uniform in rank, unless we allow that the parameters β through δ have weights. Doing so, with $w(\mathrm{D}_x) = 1$, we obtain

$$\begin{aligned} w(u) + w(\mathrm{D}_t) &= 3w(u) + 1 = w(u) + 2w(v) + 1 = w(u) + w(\beta) + 1 \\ &= w(v) + w(\gamma) + 1 = w(v) + 2, \end{aligned} \tag{1.10}$$

$$\begin{aligned} w(v) + w(\mathrm{D}_t) &= 2w(u) + w(v) + 1 = 3w(v) + 1 = w(u) + w(\theta) + 1 \\ &= w(v) + w(\delta) + 1 = w(u) + 2. \end{aligned} \tag{1.11}$$

Hence,

$$w(u) = w(v) = \frac{1}{2}, \quad w(\beta) = w(\gamma) = w(\theta) = w(\delta) = 1, \quad w(\mathrm{D}_t) = 2. \tag{1.12}$$

Remark. For scaling-invariant equations like (1.1) and (1.5), it suffices to consider the dilation symmetry on the space of independent and dependent variables. For systems like (1.9) that are inhomogeneous for scaling, we give weights to the parameters to circumvent the problem. For systems that lack scaling invariance and have no parameters, introducing one (or more) auxiliary parameter(s) with appropriate scaling provides a solution.

The trick is to extend the action of the dilation symmetry to the space of independent and dependent variables, *including* the parameters. Doing so, our algorithms apply to a larger class of polynomial PDEs. The extra parameters are only used in the first step of the algorithms: that is, in producing the candidate densities and generalized symmetries. Beyond that first step, parameters are no longer treated as dependent variables! Details and examples are given in [15, 17, 23].

1.3 Conservation Laws

Definition. A conservation law for (1.7),

$$\mathrm{D}_t \rho + \mathrm{D}_x J = 0, \tag{1.13}$$

connects the *conserved density* ρ and the *associated flux* J. As usual, D_t and D_x are total derivatives, and (1.13) holds for all solutions of (1.7). Hence, density-flux pairs only depend on $\mathbf{u}, \mathbf{u}_x$, etc., not on $\mathbf{u}_t$. With a few exceptions, densities and fluxes do not explicitly depend on t and x.

For the scalar case, $u_t = F$, the computations are carried out as follows:

$$\mathrm{D}_t \rho = \frac{\partial \rho}{\partial t} + \sum_{k=0}^{n} \frac{\partial \rho}{\partial u_{kx}} \mathrm{D}_x^k u_t, \tag{1.14}$$

where n is the order of ρ. Upon replacement of u_t, u_{xt}, etc. from $u_t = F$, one gets

$$\mathrm{D}_t \rho = \frac{\partial \rho}{\partial t} + \rho'(u)[F], \tag{1.15}$$

where $\rho'(u)[F]$ is the Fréchet derivative of ρ in the direction of F (see Section 1.4). Furthermore,

$$\mathrm{D}_x J = \frac{\partial J}{\partial x} + \sum_{k=0}^{m} \frac{\partial J}{\partial u_{kx}} u_{(k+1)x}, \tag{1.16}$$

where m is the order of J. Integrating both terms in (1.13) with respect to x yields

$$\mathrm{D}_t \int_{-\infty}^{+\infty} \rho \, dx = -J|_{-\infty}^{+\infty} = 0, \tag{1.17}$$

provided that J vanishes at infinity. In that case,

$$P = \int_{-\infty}^{+\infty} \rho \, dx = \text{constant in time.} \tag{1.18}$$

So, P is the true conserved quantity. For ODEs, the quantities P are called constants of motion.

Examples. The first three (of infinitely many) independent conservation laws [1, 29] for (1.1) are

$$\mathrm{D}_t(u) - \mathrm{D}_x(3u^2 + u_{2x}) = 0, \tag{1.19}$$

$$\mathrm{D}_t(u^2) - \mathrm{D}_x(4u^3 - u_x^2 + 2uu_{2x}) = 0, \tag{1.20}$$

$$\mathrm{D}_t\left(u^3 - \frac{1}{2}u_x^2\right) - \mathrm{D}_x\left(\frac{9}{2}u^4 - 6uu_x^2 + 3u^2u_{2x} + \frac{1}{2}u_{2x}^2 - u_xu_{3x}\right) = 0. \tag{1.21}$$

The first two conservation laws correspond to conservation of momentum and energy. Note that the above conservation laws are indeed invariant under (1.3). The terms in the conservation laws have ranks $5, 7$, and 9. The densities

$$\rho^{(1)} = u, \quad \rho^{(2)} = u^2, \quad \text{and} \quad \rho^{(3)} = u^3 - \frac{1}{2}u_x^2 \tag{1.22}$$

have ranks $2, 4$ and 6, respectively. The associated fluxes have ranks $4, 6$ and 8.

Equation (1.1) also has a density-flux pair that depends explicitly on t and x :

$$\tilde{\rho} = tu^2 + \frac{1}{3}xu, \tag{1.23}$$

$$\tilde{J} = t(4u^3 + 2uu_{2x} - u_x^2) + x\left(u^2 + \frac{1}{3}u_{2x}\right) - \frac{1}{3}u_x. \tag{1.24}$$

$\tilde{\rho}$ has rank 1, $\tilde{J}$ has rank 3, since $w(t) = -3$ and $w(x) = -1$. To accommodate this case, we used the total derivative notation D_t and D_x, in (1.13).

For (1.5), the first two (of infinitely many) conserved densities [15] are

$$\rho^{(1)} = u \quad \text{and} \quad \rho^{(2)} = \frac{1}{3}u^3 - u_x^2. \tag{1.25}$$

We will use them in the construction of the recursion operator for (1.5) in Section 5.

We now describe how to compute *polynomial* densities that are (explicitly) independent of t and x. We refer to [14, 15] for the general algorithm covering systems as well as (x, t)-dependent densities.

Algorithm for Polynomial Conserved Densities

Step 1: Determine the form of the density

Select the rank R of ρ, say, $R = 6$. Make a list $\mathcal{L}$ of all monomials in the components of $\mathbf{u}$ and their x-derivatives that have rank R. Remove from $\mathcal{L}$ all monomials where the power of the highest derivative is 1. This is done to remove terms in ρ that are in Image (D_x), and therefore belong to the flux J. After all, densities are equivalent if they only differ by terms that are total derivatives with respect to x.

Make a linear combination with constant coefficients c_i of the monomials that eventually remain in the list $\mathcal{L}$.

For (1.1), $\mathcal{L} = \{u^3, u_x^2, uu_{2x}, u_{4x}\}$. Next, u_{4x} and uu_{2x} are removed. Obviously, $u_{4x} = \mathrm{D}_x u_{3x}$, and $uu_{2x} = \frac{1}{2}\mathrm{D}_x^2 u^2 - u_x^2$. So, uu_{2x} and u_x^2 only differ by a total x-derivative. From $\mathcal{L} = \{u^3, u_x^2\}$, one constructs $\rho = c_1 u^3 + c_2 u_x^2$, which has rank $R = 6$.

Step 2: Determine the unknown coefficients

Substitute ρ into the conservation law (1.13), and compute $\mathrm{D}_t\rho$ via (1.14). Use the PDE system to eliminate all t-derivatives of $\mathbf{u}$, and require the resulting expression E to be a total x-derivative.

To avoid integration by parts, apply the Euler operator (also called the variational derivative) [30]

$$\begin{aligned} L_u &= \sum_{k=0}^{m} (-\mathrm{D}_x)^k \frac{\partial}{\partial u_{kx}} \\ &= \frac{\partial}{\partial u} - \mathrm{D}_x \left(\frac{\partial}{\partial u_x}\right) + \mathrm{D}_x^2 \left(\frac{\partial}{\partial u_{2x}}\right) + \cdots + (-1)^m \mathrm{D}_x^m \left(\frac{\partial}{\partial u_{mx}}\right) \end{aligned} \tag{1.26}$$

to E of order m. If $L_u(E) = 0$ immediately, then E is a total x-derivative. If $L_u(E) \neq 0$, then the remaining expression must vanish identically. This yields a linear system for the constants c_i. Solve the system. Carrying out these operations for (1.1), one gets $c_1 = 1, c_2 = -\frac{1}{2}$.

Remark. With (1.15), the system for the c_i follows from $L_u(\rho'(u)[F]) = 0$ by equating to zero the coefficients of monomials in u and their x-derivatives.

Implementation in Mathematica

In *Mathematica*, `D` is the total derivative operator and the variational derivative (Euler operator) can be found in the Standard Add-on Package *Calculus‘VariationalMethods‘*.

For instance, returning to $\rho^{(3)}$ in (1.22), with (1.1) and (1.14), one computes

$$\begin{aligned} E &= \mathrm{D}_t\rho^{(3)} = {\rho^{(3)}}'(u)[u_t] = (3u^2 - u_x \mathrm{D})u_t \\ &= 18u^3 u_x - 6u_x^3 - 6uu_x u_{2x} + 3u^2 u_{3x} - u_x u_{4x}. \end{aligned} \tag{1.27}$$

Application of the variational derivative, `VariationalD[E,u[x,t],{x,t}]` gives zero. That means that E is a total x-derivative of a polynomial $J^{(3)}$. Integration of $E = -\mathrm{D}_x J^{(3)}$ gives

$$J^{(3)} = -(\frac{9}{2}u^4 - 6uu_x^2 + 3u^2 u_{2x} + \frac{1}{2}u_{2x}^2 - u_x u_{3x}). \tag{1.28}$$

Example. With our package *InvariantsSymmetries.m*, we searched for conserved densities of (1.9). Obviously, (1.9) is a conservation law; thus, $\rho^{(1)} = u$ and $\rho^{(2)} = v$, without conditions on the parameters. Additional conserved densities only exist if $\gamma = \theta$. The first few are:

$$\rho^{(3)} = u^2 + v^2, \tag{1.29}$$

$$\rho^{(4)} = \frac{1}{2}(u^2+v^2)^2 + (\beta-\delta)u^2 + 2\theta uv + 2vu_x, \tag{1.30}$$

$$\begin{aligned}\rho^{(5)} = &\ \frac{1}{4}(u^2+v^2)^3 + \frac{1}{2}(u_x^2+v_x^2) + \theta uv(u^2+v^2) \\ &+\frac{1}{4}(\beta-\delta)(u^4-v^4) + 3u^2vu_x + v^3u_x,\end{aligned} \tag{1.31}$$

and

$$\begin{aligned}\rho^{(6)} = &\ \frac{5}{32}(u^2+v^2)^4 + \frac{3}{4}(u^2+v^2)(u_x^2+v_x^2) + \frac{1}{2}(uu_x+vv_x)^2 \\ &+\frac{1}{8}(\beta-\delta)^2u^4 + \frac{1}{4}(\beta-\delta)u^6 + \frac{1}{2}(\beta-\delta)\theta u^3v + \frac{3}{4}(\beta-\delta)\theta^2v^2 \\ &+\frac{3}{8}(\beta-\delta)u^4v^2 - \frac{1}{4}\theta^2(u^4+v^4) + \frac{3}{4}\theta(u^5v+uv^5) \\ &-\frac{1}{8}(\beta-\delta)v^6 - \frac{3}{2}\theta^3uv + \frac{3}{2}\theta u^3v^3 - \frac{3}{2}\theta^2vu_x \\ &+\frac{3}{2}(\beta-\delta)u^2vu_x + \frac{15}{4}u^4vu_x + \frac{3}{2}\theta uv^2u_x + \frac{5}{2}u^2v^3u_x \\ &+\frac{3}{4}v^5u_x + \frac{1}{4}(\beta-\delta)u_x^2 + \frac{1}{2}\theta u_xv_x + \frac{1}{2}v_xu_{2x}.\end{aligned} \tag{1.32}$$

Via integration by parts, $\rho^{(4)}$ is equivalent to

$$\tilde{\rho}^{(4)} = \frac{1}{2}(u^2+v^2)^2 + (\beta-\delta)u^2 + 2\theta uv + vu_x - uv_x. \tag{1.33}$$

Indeed, $\rho^{(4)} = \tilde{\rho}^{(4)} + D_x(uv)$.

Judged from (1.29)–(1.32), the complexity of the expressions dramatically increases as the rank increases. The fact that we were able to compute 6 independent densities for (1.9) is an indicator that the system presumably is completely integrable, as was later proved in [40].

Only the conserved densities $\rho^{(1)}$ through $\rho^{(3)}$ will be used in the construction of the recursion operator for (1.9) in Section 6.

1.4 Generalized Symmetries

Definition. A vector function $\mathbf{G}(x,t,\mathbf{u},\mathbf{u}_x,\mathbf{u}_{2x},\ldots)$ is called a *symmetry* of (1.7) if and only if it leaves (1.7) invariant under the replacement $\mathbf{u} \to \mathbf{u} + \epsilon\mathbf{G}$ within order ϵ. Hence,

$$\mathrm{D}_t(\mathbf{u}+\epsilon\mathbf{G}) = \mathbf{F}(\mathbf{u}+\epsilon\mathbf{G}) \tag{1.34}$$

must hold up to order ϵ on any solution of (1.7). Consequently, $\mathbf{G}$ must satisfy the linearized equation [7, 28]

$$\mathrm{D}_t\mathbf{G} = \mathbf{F}'(\mathbf{u})[\mathbf{G}], \tag{1.35}$$

where $\mathbf{F}'$ is the Fréchet derivative of $\mathbf{F}$, i.e.,

$$\mathbf{F}'(\mathbf{u})[\mathbf{G}] = \frac{\partial}{\partial \epsilon}\mathbf{F}(\mathbf{u}+\epsilon\mathbf{G})_{|\epsilon=0}. \tag{1.36}$$

In (1.34) and (1.36), we infer that $\mathbf{u}$ is replaced by $\mathbf{u}+\epsilon\mathbf{G}$, and $\mathbf{u}_{nx}$ by $\mathbf{u}_{nx}+\epsilon \mathrm{D}_x^n\mathbf{G}$. As usual, D_t and D_x are total derivatives, and $\mathbf{G} = (G_1, G_2, \ldots, G_n)$ if the system (1.7) has n components.

Once higher-order symmetries have been found, these vector fields can be used to obtain fundamental information about the integrability of the equation. In many cases, conserved quantities, Hamiltonian structures, and recursion operators follow readily from the knowledge of generalized symmetries [7].

Examples. The first three (of infinitely many) symmetries [30] of (1.1) are

$$\begin{aligned} G^{(1)} &= u_x, \quad G^{(2)} = 6uu_x + u_{3x}, \\ G^{(3)} &= 30u^2u_x + 20u_xu_{2x} + 10uu_{3x} + u_{5x}. \end{aligned} \tag{1.37}$$

All the terms in these symmetries have rank $3, 5$ and 7, respectively.

With higher-order symmetries one can generate new integrable PDEs. For example, $u_t = G^{(3)}$ is the Lax equation in the completely integrable KdV hierarchy [1].

Note that (1.1) also admits symmetries that explicitly depend on t and x. Indeed, the symmetries

$$\tilde{G}^{(1)} = 1 + 6tu_x \quad \text{and} \quad \tilde{G}^{(2)} = 4u + 2xu_x + 36tuu_x + 6tu_{3x} \tag{1.38}$$

are of rank 0 and 2. They linearly (and explicitly) depend on t and x.

The algorithm presented in this paper can easily be extended to cover this type of symmetries (see [14, 17] for details).

The situation for (1.5), which also has infinitely many polynomial symmetries, is more complicated. The symmetries of (1.5) originate from two distinct "seeds":

$$G^{(1)} = u_x \quad \text{and} \quad G^{(2)} = 5u^2u_x + 5u_xu_{2x} + 5uu_{3x} + u_{5x}. \tag{1.39}$$

We have also computed symmetries of higher rank, but we do not show them here. A detailed computer-aided study showed that the symmetries $G^{(2i-1)}$ with rank $6i-3$ come from the seed $G^{(1)}$, whereas $G^{(2i)}$ with rank $6i+1$ originate from $G^{(2)}$, where $i = 1, 2, \ldots$ (see [36] for details).

For systems of type (1.7), the symmetry $\mathbf{G}$ is a vector with n components. Our computer search with *InvariantsSymmetries.m* revealed that (1.9) is invariant under the transformation $(u, v) \to (v, -u)$, which is a Lie point symmetry, provided the conditions $\beta = \delta$ and $\gamma = -\theta$ hold. However, these conditions do not lead to a hierarchy of integrable equations. We therefore continued our search with arbitrary nonzero parameters β through δ.

The first two symmetries of (1.9) are $\mathbf{G}^{(1)} = (G_1^{(1)}, G_2^{(1)})$ and $\mathbf{G}^{(2)} = (G_1^{(2)}, G_2^{(2)})$, where

$$\begin{aligned} G_1^{(1)} &= u_x, \\ G_2^{(1)} &= v_x, \end{aligned} \tag{1.40}$$

$$\begin{aligned} G_1^{(2)} &= (\beta-\delta)u_x + 3u^2u_x + v^2u_x + \gamma v_x + 2uvv_x - v_{2x}, \\ G_2^{(2)} &= \theta u_x + 2uvu_x + u^2v_x + 3v^2v_x + u_{2x}. \end{aligned} \tag{1.41}$$

Note that the sum of symmetries is still a symmetry. Remembering $(u_t, v_t) = (F_1, F_2)$ in (1.9), we then have $G_1^{(2)} + \delta G_1^{(1)} = -F_1$ and $G_2^{(2)} = -F_2$.

The next symmetry, $\mathbf{G}^{(3)} = (G_1^{(3)}, G_2^{(3)})$, only exists if $\gamma = \theta$. It is

$$
\begin{aligned}
G_1^{(3)} &= 3(\beta - \delta)u^2 u_x + \frac{15}{2}u^4 u_x + 6\theta uvu_x + 9u^2v^2u_x + \frac{3}{2}v^4u_x + 3\theta(u^2+v^2)v_x \\
&\quad + 6(u^3v + uv^3)v_x - 3(u^2+v^2)v_{2x} - 6uu_xv_x - 6vv_x^2 - u_{3x}, \qquad (1.42) \\
G_2^{(3)} &= 3\theta(u^2+v^2)u_x + 6(u^2+v^2)uvu_x + 6uu_x^2 + \frac{3}{2}u^4v_x + 6\theta uvv_x \\
&\quad + 9u^2v^2v_x - 3(\beta-\delta)v^2v_x + \frac{15}{2}v^4v_x + 6vu_xv_x + 3(u^2+v^2)u_{2x} - v_{3x}. \qquad (1.43)
\end{aligned}
$$

The algorithm for symmetries is similar to the one for conserved densities. The only difference is that monomials that differ by a total x-derivative are no longer removed from the list $\mathcal{L}$.

Algorithm for Polynomial Generalized Symmetries

Step 1: Determine the form of the symmetry

Select the rank R of the symmetry. Make a list $\mathcal{L}$ of all monomials involving $\mathbf{u}$ and its x-derivatives of rank R. To obtain the form of the symmetry, make a linear combination of these monomials with constant coefficients c_i. For example, for (1.1), $G = c_1\, u^2u_x + c_2\, u_xu_{2x} + c_3\, uu_{3x} + c_4\, u_{5x}$ is the form of the generalized symmetry of rank $R = 7$.

Step 2: Determine the unknown coefficients

Compute $\mathrm{D}_t\mathbf{G}$. Use the PDE system to remove all t-derivatives. Equate the result to the Fréchet derivative $\mathbf{F}'(\mathbf{u})[\mathbf{G}]$. Treat the different monomial terms in $\mathbf{u}$ and its x-derivatives as independent to get the linear system for the c_i. Solve that system. For (1.1), one obtains the symmetry of rank 7 :

$$
G = 30u^2u_x + 20u_xu_{2x} + 10uu_{3x} + u_{5x}. \qquad (1.44)
$$

Symmetries of lower or higher rank are computed similarly. See [14, 17, 23] for details about the algorithm and its implementation.

Remark. Starting with a conserved density ρ, the symmetries for a Hamiltonian system can be obtained from $\mathrm{D}_x(L_u(\rho))$, where L_u is defined in (1.26). See, for example, [31] for a study of the connection between densities and symmetries for Lagrangian and non-Lagrangian systems.

Implementation in Mathematica

The key tool to compute symmetries is the Fréchet derivative, which is implemented as follows:

```
frechet[funcF_List,funcU_List,indVars_List,funcG_List]:=
```

```
Module[{eps,resultlist={},i},
Do[resultlist=Append[resultlist,
 Expand[(D[Part[funcF,i] /.
 {(f_/; MemberQ[funcU,f])[Sequence @@ indVars]:>
 f[Sequence @@ indVars]+eps*funcG[[Flatten[Position[funcU,f]][[1]]]],
 Derivative[k_,0][f_][Sequence @@ indVars]:>
 D[f[Sequence @@ indVars]+eps*funcG[[Flatten[Position[funcU,f]][[1]]]],
 {indVars[[1]],k}]},eps]/. eps :> 0)]],
 {i,Length[funcF]}];
Return[resultlist]];
```

To compute the Fréchet derivative of $u^3 - \frac{1}{2}u_x^2$ in the direction of $k(x,t)$, type

```
frechet[{u[x,t]^3-(1/2)*D[u[x,t],x]^2},{u},{x,t},{k[x,t]}];
```

This gives the answer $3u^2k - u_xk_x$.

1.5 Recursion Operators for Scalar Equations

Definition. A *recursion operator* is a linear operator $\mathbf{\Phi}$ on the space of differential functions with the property that whenever $\mathbf{G}$ is a symmetry of (1.7), so is $\hat{\mathbf{G}}$ with $\hat{\mathbf{G}} = \mathbf{\Phi}\mathbf{G}$.

The equation for the recursion operator [7, 30, 39] is

$$\mathrm{D}_t\mathbf{\Phi} + [\mathbf{\Phi}, \mathbf{F}'(u)] = \frac{\partial \mathbf{\Phi}}{\partial t} + \mathbf{\Phi}'[\mathbf{F}] + \mathbf{\Phi} \circ \mathbf{F}'(u) - \mathbf{F}'(u) \circ \mathbf{\Phi} = 0, \tag{1.45}$$

where [,] means commutator, $\circ$ indicates for composition, and the variational derivative of the operator $\mathbf{\Phi}$ is defined in, e.g., [7].

For n-component systems like (1.7), the symmetries $\mathbf{G}$ are vectors with n components, and the recursion operator $\mathbf{\Phi}$ is an $n \times n$ matrix.

Examples. The recursion operator for (1.1) is

$$\Phi_{\mathrm{KdV}} = \mathrm{D}^2 + 4u + 2u_x\mathrm{D}^{-1} = \mathrm{D}^2 + 2u + 2\mathrm{D}u\mathrm{D}^{-1} \;, \tag{1.46}$$

where, for simplicity of notation, $\mathrm{D} = \mathrm{D}_x$ and $\mathrm{D}^{-1} = \mathrm{D}_x^{-1}$. Here, $\Phi_{\mathrm{KdV}}\, G^{(1)} = G^{(2)}$ and $\Phi_{\mathrm{KdV}}\, G^{(2)} = G^{(3)}$ for the symmetries listed in (1.37). The recursion operator Φ_{KdV} also connects the (x,t)-dependent symmetries (1.38), i.e., $\Phi_{\mathrm{KdV}}\tilde{G}^{(1)} = \tilde{G}^{(2)}$, but $\Phi_{\mathrm{KdV}}\tilde{G}^{(2)}$ is no longer local.

Since $w(\mathrm{D}^{-1}) = -w(\mathrm{D}) = -1$, the three terms in (1.46) have rank $R = 2$. The recursion operator (1.46) is uniform in rank. Clearly, the rank of Φ_{KdV} is the difference in rank between consecutive symmetries in (1.37).

In view of the symmetries (1.39), the recursion operator for (1.5) must have rank 6. Indeed, the recursion operator [9, 10] has rank 6 :

$$\begin{aligned}\Phi_{\mathrm{SK}} =\, & \mathrm{D}^6 + 2u\mathrm{D}^4 + 2\mathrm{D}u\mathrm{D}^3 + \mathrm{D}^2u\mathrm{D}^2 + 3u\mathrm{D}u\mathrm{D} + 3u\mathrm{D}^2u - 2\mathrm{D}u\mathrm{D}u - 2u^3 \\ & + \mathrm{D}^5u\mathrm{D}^{-1} + 5\mathrm{D}u\mathrm{D}^2u\mathrm{D}^{-1} + 5u^2\mathrm{D}u\mathrm{D}^{-1} + \mathrm{D}u\mathrm{D}^{-1}(u^2 - 2u_x\mathrm{D}),\end{aligned} \tag{1.47}$$

which can also be written as

$$\begin{aligned}\Phi_{\mathrm{SK}} = & \mathrm{D}^6 + 3u\mathrm{D}^4 - 3\mathrm{D}u\mathrm{D}^3 + 11\mathrm{D}^2u\mathrm{D}^2 - 10\mathrm{D}^3u\mathrm{D} + 5\mathrm{D}^4u \\ & +12u^2\mathrm{D}^2 - 19u\mathrm{D}u\mathrm{D} + 8u\mathrm{D}^2u + 8\mathrm{D}u\mathrm{D}u + 4u^3 \\ & +u_x\mathrm{D}^{-1}(u^2 - 2u_x\mathrm{D}) + G^{(2)}\mathrm{D}^{-1},\end{aligned} \tag{1.48}$$

with $G^{(2)}$ in (1.39).

Our algorithm for the computation of polynomial recursion operators is based on the following observations.

Key observations. All terms in (1.46) and (1.48) are monomials in $\mathrm{D}, \mathrm{D}^{-1}, u$, and u_x. Depending on the form of the recursion operator, u_{2x}, u_{3x}, etc. can also appear, as is the case in (1.47).

Recursion operators split naturally in $\Phi = \Phi_0 + \Phi_1$, where Φ_0 is a differential operator (without D^{-1} terms), and Φ_1 is an integral operator (with D^{-1} terms).

Furthermore, application of Φ to any symmetry should not leave any integrals unresolved, since all symmetries are polynomial (see [34]). This is where the connection between conserved densities and symmetries comes into play.

For instance, for (1.1) it is clear that $\mathrm{D}^{-1}(6uu_x + u_{3x}) = 3u^2 + u_{2x}$ is polynomial. Similarly, for (1.5), using (1.48), we have

$$\mathrm{D}^{-1}(5u^2u_x + 5u_xu_{2x} + 5uu_{3x} + u_{5x}) = \frac{5}{3}u^3 + 5uu_{2x} + u_{4x} \tag{1.49}$$

and

$$\begin{aligned}&\mathrm{D}^{-1}(u^2 - 2u_x\mathrm{D})(5u^2u_x + 5u_xu_{2x} + 5uu_{3x} + u_{5x}) \\ &= u^5 - 10u^2u_x^2 + \cdots - 2u_xu_{5x}.\end{aligned} \tag{1.50}$$

The first two conserved densities of (1.5) are $\rho^{(1)} = u$ and $\rho^{(2)} = \frac{1}{3}u^3 - u_x^2$. Thus, with (1.13), we get $\mathrm{D}_tu = u_t = -\mathrm{D}_xJ^{(1)}$ and

$$\mathrm{D}_t\left(\frac{1}{3}u^3 - u_x^2\right) = \rho'(u)[u_t] = (u^2 - 2u_x\mathrm{D})u_t = -\mathrm{D}_xJ^{(2)}. \tag{1.51}$$

So, the factor $(u^2 - 2u_x\mathrm{D})$ in (1.48) comes from $\rho^{(2)}$, and $\mathrm{D}^{-1}[(u^2 - 2u_x\mathrm{D})u_t]$ will be polynomial, namely $-J^{(2)}$.

A similar situation happens for (1.1), where $\rho^{(1)} = u, \rho^{(2)} = u^2$, and $\rho^{(3)} = u^3 - \frac{1}{2}u_x^2$. Then, with (1.13) and (1.15),

$$\begin{aligned}&\mathrm{D}_t\rho^{(1)} = \mathrm{D}_tu = u_t = -\mathrm{D}_xJ^{(1)}, \quad \mathrm{D}_t\rho^{(2)} = \mathrm{D}_tu^2 = 2uu_t = -\mathrm{D}_xJ^{(2)}, \quad \text{and} \\ &\mathrm{D}_t\rho^{(3)} = \mathrm{D}_t(u^3 - \frac{1}{2}u_x^2) = \rho^{(3)\prime}(u)[u_t] = (3u^2 - u_x\mathrm{D}_x)u_t = -\mathrm{D}_xJ^{(3)},\end{aligned} \tag{1.52}$$

for polynomial $J^{(i)},\ i = 1, 2, 3$. Thus, application of D^{-1}, or $\mathrm{D}^{-1}u$, or $\mathrm{D}^{-1}(3u^2 - u_x\mathrm{D})$ to $6uu_x + u_{3x}$ leads to a polynomial result. However, as will be shown below, the terms involving $\mathrm{D}^{-1}u$ and $\mathrm{D}^{-1}(3u^2 - u_x\mathrm{D})$ are not needed in the construction of the recursion operator (1.46).

Since our algorithm for recursion operators is the most elaborate, we give the steps that lead to (1.46) and (1.48). We consider the scalar case first. Systems are dealt with in Section 6.

Algorithm for Polynomial Recursion Operators

Step 1: Construct the form of the recursion operator

(i) Determine the rank of the operator

Compute the rank R of the operator based on the known ranks of consecutive symmetries. For example, from (1.37), we compute

$$R = \text{rank}\ \Phi = \text{rank}\ G^{(3)} - \text{rank}\ G^{(2)} = \text{rank}\ G^{(2)} - \text{rank}\ G^{(1)} = 2. \tag{1.53}$$

Obviously, rank $\Phi_0 =$ rank $\Phi_1 =$ rank $\Phi = R$.

(ii) Determine the pieces of the operator Φ_0

Make a list $\mathcal{L}$ of all permutations of $\mathrm{D}^j u^k$, with j and k nonnegative integers, that have the rank R. For (1.1),

$$\mathcal{L} = \{\mathrm{D}^2, u\}. \tag{1.54}$$

(iii) Determine the pieces of the operator Φ_1

It can be shown [2, 39] that

$$\Phi_1 = \sum_j \sum_k G^{(j)} \mathrm{D}^{-1} {\rho^{(k)}}'(u), \tag{1.55}$$

where the symmetries $G^{(j)}$ are combined with D^{-1} and ${\rho^{(k)}}'(u)$ in such a way that every term is exactly of rank $\Phi_1 = R$. That is, the indices j and k are taken so that rank $(G^{(j)})$ + rank $({\rho^{(k)}}'(u)) - 1 = R$ for every term in (1.55).

Using the densities and symmetries, make a list $\mathcal{M}$ of the pieces involving D^{-1}.

For (1.1), from Table 1 it should be clear that D^{-1} can only be sandwiched between u_x and 1. Any other combination would exceed rank 2. Hence,

$$\mathcal{M} = \{u_x \mathrm{D}^{-1}\}. \tag{1.56}$$

(iv) Build the operator Φ

Next, produce $\mathcal{R} = \mathcal{L} \bigcup \mathcal{M}$, which has the building blocks of the recursion operator. To get Φ, linearly combine the pieces in $\mathcal{R}$ with constant coefficients c_i. For (1.1), we obtain

$$\mathcal{R} = \{\mathrm{D}^2, u, u_x \mathrm{D}^{-1}\}. \tag{1.57}$$

Thus,

$$\Phi_{\mathrm{KdV}} = c_1\, \mathrm{D}^2 + c_2\, u + c_3\, u_x \mathrm{D}^{-1}. \tag{1.58}$$

We now repeat steps (i)–(iv) for the SK equation (1.5).

Table 1 Building blocks of Φ_1 for the KdV equation.

Rank	Symmetry $G^{(j)}$	Density $\rho^{(k)}$	${\rho^{(k)}}'(u)$
0	—	—	1
1	—	—	—
2	—	u	u
3	u_x	—	—
4	—	u^2	$3u^2 - u_x\mathrm{D}$
5	$6uu_x + u_{3x}$	—	—
6	—	$u^3 - \frac{1}{2}u_x^2$	

(i) Using the symmetries (1.39), we get

$$\text{rank } \Phi = \text{rank } G^{(3)} - \text{rank } G^{(1)} = \text{rank } G^{(4)} - \text{rank } G^{(2)} = 6. \tag{1.59}$$

(ii) The operator Φ_0 will be built from

$$\begin{aligned}\mathcal{L} \;=\; &\{\mathrm{D}^6, u\mathrm{D}^4, \mathrm{D}u\mathrm{D}^3, \mathrm{D}^2u\mathrm{D}^2, \mathrm{D}^3u\mathrm{D}, \mathrm{D}^4u, u^2\mathrm{D}^2, u\mathrm{D}u\mathrm{D},\\ &u\mathrm{D}^2u, \mathrm{D}u^2\mathrm{D}, \mathrm{D}u\mathrm{D}u, \mathrm{D}^2u^2, u^3\}.\end{aligned} \tag{1.60}$$

Table 2 Building blocks of Φ_1 for the SK equation.

Rank	Symmetry $G^{(j)}$	Density $\rho^{(k)}$	${\rho^{(k)}}'(u)$
0	—	—	1
1	—	—	—
2	—	u	—
3	u_x	—	—
4	—	—	$u^2 - 2u_x\mathrm{D}$
5	—	—	—
6	—	$\frac{1}{3}u^3 - u_x^2$	—
7	$5u^2u_x + 5u_xu_{2x} + 5uu_{3x} + u_{5x}$	—	—

(iii) From Table 2, which list the building blocks for Φ_1 for (1.25), we obtain

$$\mathcal{M} = \{u_x\mathrm{D}^{-1}(u^2 - 2u_x\mathrm{D}), (5u^2u_x + 5u_xu_{2x} + 5uu_{3x} + u_{5x})\mathrm{D}^{-1}\}. \tag{1.61}$$

All other combinations of the form $G^{(j)}\mathrm{D}^{-1}{\rho^{(k)}}'(u)$ exceed rank 6.

(iv) Combining the monomials from $\mathcal{R} = \mathcal{L} \bigcup \mathcal{M}$, we get

$$\begin{aligned}\Phi_{\mathrm{SK}} \;=\; & c_1\mathrm{D}^6 + c_2 u\mathrm{D}^4 + c_3\mathrm{D}u\mathrm{D}^3 + c_4\mathrm{D}^2u\mathrm{D}^2 + c_5\mathrm{D}^3u\mathrm{D} \\ & + c_6\mathrm{D}^4u + c_7u^2\mathrm{D}^2 + c_8u\mathrm{D}u\mathrm{D} + c_9u\mathrm{D}^2u + c_{10}\mathrm{D}u^2\mathrm{D} \\ & + c_{11}\mathrm{D}u\mathrm{D}u + c_{12}\mathrm{D}^2u^2 + c_{13}u^3 + c_{14}u_x\mathrm{D}^{-1}(u^2 - 2u_x\mathrm{D}) \\ & + c_{15}(5u^2u_x + 5u_xu_{2x} + 5uu_{3x} + u_{5x})\mathrm{D}^{-1}. \qquad (1.62)\end{aligned}$$

Step 2: Determine the unknown coefficients

To determine the coefficients c_i, require that

$$\Phi G^{(k)} = G^{(k+s)}, \quad k = 1, 2, 3, \ldots, \qquad (1.63)$$

where s is the number of seeds. In practice, it suffices to use $k = 1$ and 2 in (1.63) to fix all coefficients c_i. Solve the resulting linear system(s) for the unknown c_i.

For (1.1), $\Phi_{\mathrm{KdV}}G^{(2)} = G^{(3)}$ with Φ_{KdV} in (1.58) and its symmetries (1.37), we obtain

$$\mathcal{S} = \{c_1 - 1 = 0, 18c_1 + c_3 - 20 = 0, 6c_1 + c_2 - 10 = 0, 2c_2 + c_3 - 10 = 0\}.$$

The solution is $c_1 = 1, c_2 = 4$, and $c_3 = 2$. Substituting it into (1.58), we get

$$\Phi_{\mathrm{KdV}} = \mathrm{D}^2 + 4\,u + 2\,u_x\mathrm{D}^{-1}. \qquad (1.64)$$

The explicit computation on page 260 in [7] shows that (1.64) satisfies (1.45).

For (1.5), we express that

$$\Phi_{\mathrm{SK}}G^{(1)} = G^{(3)} \quad \text{and} \quad \Phi_{\mathrm{SK}}G^{(2)} = G^{(4)}, \qquad (1.65)$$

with Φ_{SK} in (1.62) and the symmetries (1.39). Then we solve for the constants c_i. This eventually yields

$$\begin{aligned}\Phi_{\mathrm{SK}} =\; & \mathrm{D}^6 + 3u\mathrm{D}^4 - 3\mathrm{D}u\mathrm{D}^3 + 11\mathrm{D}^2u\mathrm{D}^2 - 10\mathrm{D}^3u\mathrm{D} + 5\mathrm{D}^4u \\ & + 12u^2\mathrm{D}^2 - 19u\mathrm{D}u\mathrm{D} + 8u\mathrm{D}^2u + 8\mathrm{D}u\mathrm{D}u + 4u^3 \\ & + u_x\mathrm{D}^{-1}(u^2 - 2u_x\mathrm{D}) + G^{(2)}\mathrm{D}^{-1}, \qquad (1.66)\end{aligned}$$

with $G^{(2)}$ in (1.39). A lengthy computation shows that this recursion operator satisfies (1.45).

After integration by parts, (1.47) or, equivalently, (1.66) can also be written [36] as

$$\begin{aligned}\Phi_{\mathrm{SK}} \;=\; & \mathrm{D}^6 + 6u\mathrm{D}^4 + 9u_x\mathrm{D}^3 + 9u^2\mathrm{D}^2 + 11u_{2x}\mathrm{D}^2 + 10u_{3x}\mathrm{D} + 21uu_x\mathrm{D} \\ & + 4u^3 + 16uu_{2x} + 6u_x^2 + 5u_{4x} + u_x\mathrm{D}^{-1}(u^2 + 2u_{2x}) + G^{(2)}\mathrm{D}^{-1}. \qquad (1.67)\end{aligned}$$

1.6 Recursion Operators for Systems

We show how to construct the recursion operators for systems (1.7) with n components. The symmetry $\mathbf{G}$ has n components and the recursion operator $\mathbf{\Phi}$ is a $n \times n$ matrix.

We used *Mathematica* interactively to compute the recursion operator for (1.9) with $\gamma = \theta$. The recursion operator has the form

$$\mathbf{\Phi} = \begin{pmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{pmatrix}, \tag{1.68}$$

From $\mathbf{G}^{(2)} = \mathbf{\Phi}\mathbf{G}^{(1)}$, the rank of the entries Φ_{ij} is determined by

$$\begin{aligned} \text{rank } G_1^{(2)} &= \text{rank } \Phi_{11} + \text{rank } G_1^{(1)} = \text{rank } \Phi_{12} + \text{rank } G_2^{(1)}, \\ \text{rank } G_2^{(2)} &= \text{rank } \Phi_{21} + \text{rank } G_1^{(1)} = \text{rank } \Phi_{22} + \text{rank } G_2^{(1)}. \end{aligned} \tag{1.69}$$

For (1.9), the difference in rank between consecutive symmetries is 1, so rank $\Phi_{ij} = 1,\ i,j = 1,2$.

(i) As for the scalar case, we first construct the differential operator $\mathbf{\Phi}_0$. In view of the weights (1.12),

$$\mathcal{L}_{ij} = \{u^2, v^2, uv, \mathrm{D}, \beta, \delta, \theta\}. \tag{1.70}$$

Hence,

$$\mathbf{\Phi}_0 = \begin{pmatrix} c_1 u^2 + c_2 v^2 + c_3 uv + c_4 \mathrm{D} + c_5 & c_6 u^2 + c_7 v^2 + c_8 uv + c_9 \mathrm{D} + c_{10} \\ c_{11} u^2 + c_{12} v^2 + c_{13} uv + c_{14} \mathrm{D} + c_{15} & c_{16} u^2 + c_{17} v^2 + c_{18} uv + c_{19} \mathrm{D} + c_{20} \end{pmatrix}, \tag{1.71}$$

where c_5, c_{10}, c_{15}, and c_{20} will be linear in β, δ, and θ.

(ii) Using the conserved densities $\rho^{(1)} = u, \rho^{(2)} = v$, and $\rho^{(3)} = u^2 + v^2$, we have

$$\mathrm{D}_t \rho^{(1)} = \mathrm{D}_t u = \frac{\partial u}{\partial u} F_1 + \frac{\partial u}{\partial v} F_2 = (1,0) \cdot (u_t, v_t) = -\mathrm{D}_x J^{(1)}, \tag{1.72}$$

$$\mathrm{D}_t \rho^{(2)} = \mathrm{D}_t v = \frac{\partial v}{\partial u} F_1 + \frac{\partial v}{\partial v} F_2 = (0,1) \cdot (u_t, v_t) = -\mathrm{D}_x J^{(2)}, \tag{1.73}$$

$$\begin{aligned} \mathrm{D}_t \rho^{(3)} &= \mathrm{D}_t (u^2 + v^2) = \frac{\partial (u^2+v^2)}{\partial u} F_1 + \frac{\partial (u^2+v^2)}{\partial v} F_2 \\ &= 2(u,v) \cdot (u_t, v_t) = -\mathrm{D}_x J^{(3)}, \end{aligned} \tag{1.74}$$

where the dot $(\cdot)$ refers to the standard inner product of vectors. Therefore, introducing the symmetry $(u_x, v_x)^{\mathrm{T}}$ on the left of D^{-1} gives

$$\mathcal{M} = \left\{ (u_x, v_x)^{\mathrm{T}} \odot \mathrm{D}^{-1} (u, v) \right\}, \tag{1.75}$$

where $\odot$ stands for the tensor product of matrices and T for transpose. So,

$$\begin{aligned} \mathbf{\Phi}_1 &= c_{21}\ (u_x, v_x)^{\mathrm{T}} \odot \mathrm{D}^{-1} (u, v) \\ &= \begin{pmatrix} c_{21} u_x \mathrm{D}^{-1} u & c_{21} u_x \mathrm{D}^{-1} v \\ c_{21} v_x \mathrm{D}^{-1} u & c_{21} v_x \mathrm{D}^{-1} v \end{pmatrix}. \end{aligned} \tag{1.76}$$

Note that $(u_x, v_x)^{\mathrm{T}} \odot \mathrm{D}^{-1}(1,0)$ and $(u_x, v_x)^{\mathrm{T}} \odot \mathrm{D}^{-1}(0,1)$ are of rank $\frac{1}{2}$. They cannot be used in $\mathbf{\Phi}_1$, where all pieces must have rank 1.

To determine the unknown constants in $\mathbf{\Phi} = \mathbf{\Phi}_0 + \mathbf{\Phi}_1$, we use

$$\mathbf{\Phi}\mathbf{G}^{(k)} = \mathbf{G}^{(k+1)} + \sum_{l=1}^{k} \alpha_{kl}\mathbf{G}^{(l)}, \quad k = 1, 2, \ldots, \tag{1.77}$$

where α_{kl} are unknown coefficients (which can be zero). In contrast to the examples in the previous sections, the α_{kl} play a role when dealing with systems with weighted parameters like (1.9).

It suffices to take $k = 1$ and 2 in (1.77) to fix all coefficients c_i, and the extra unknowns α_{11}, α_{21} and α_{22}. By solving a linear system, we obtain $\alpha_{11} = 0, \alpha_{21} = \theta^2$ and $\alpha_{22} = \beta - \delta$, and the values for the coefficients c_1 through c_{21}. The recursion operator then follows readily:

$$\mathbf{\Phi} \;=\; \begin{pmatrix} \beta - \delta + 2u^2 + 2u_x \mathrm{D}^{-1} u & \theta + 2uv - \mathrm{D} + 2u_x \mathrm{D}^{-1} v \\ \theta + 2uv + \mathrm{D} + 2v_x \mathrm{D}^{-1} u & 2v^2 + 2v_x \mathrm{D}^{-1} v \end{pmatrix}. \tag{1.78}$$

The recursion operator for the case $\gamma = \theta = \delta = 0$ was computed analytically in [40].

1.7 About the Integrability Package InvariantsSymmetries.m

We briefly describe our package *InvariantsSymmetries.m*, which automatically performs the computation of conservation laws (invariants) and symmetries based on the algorithms in Sections 3 and 4.

Users must have access to *Mathematica* 3.0 [42]. The files for our package are available from *MathSource* [18]. The package includes instructions for installation, on-line help, documentation, and built-in examples.

After proper installation, it is advisable to first run our notebook (called *Examples*), which is accessible through the browser as part of the Add-on Package *Integrability*. The interactive examples in the notebook will help familiarize the user with the syntax of our functions (see also [18]).

To use the package as part of a new notebook, start *Mathematica* and type `In[1]:= <<Integrability‘` to read in the package. You will get the following message:

```
Loading init.m for Integrability from AddOns.
```

The key functions for conservation laws and symmetries of PDEs are `PDEInvariants` and `PDESymmetries`. These functions take the following arguments: the equations in the system, the dependent and independent variables, and the range for the rank.

For example (1.9), the first two lines below define the system (1.9). The third line produces the densities (1.29)–(1.31). The fourth line gives the symmetries (1.40)–(1.42).

```
In[2]:= pde1 := D[u[x,t],t]+D[u[x,t]*(u[x,t]^2+v[x,t]^2)+
        beta*u[x,t]+gamma*v[x,t]-D[v[x,t],x],x] == 0;

In[3]:= pde2 := D[v[x,t],t]+D[v[x,t]*(u[x,t]^2+v[x,t]^2)+
        theta*u[x,t]+delta*v[x,t]+D[u[x,t],x],x] == 0;
```

```
In[4]:= PDEInvariants[{pde1,pde2}, {u,v}, {x,t}, {1,3}];

In[5]:= PDESymmetries[{pde1,pde2}, {u,v}, {x,t}, {3/2,5/2}];
```

Help about the functions and their options is available on-line. For instance, type `In[6]:= ??PDEInvariants` to obtain the function description. Part of it reads:

```
PDEInvariants[eqn, u, {x,t}, R, opts] finds the invariant with rank R
of a partial differential equation for the function u.
PDEInvariants[{eqn1,eqn2,...}, {u1,u2,...}, {x,t}, {Rmin,Rmax}, opts]
finds the invariants with rank Rmin through Rmax.
x is understood as the space variable and t as the time variable.
```

Typing `In[7]:= ??PDESymmetries` produces descriptions like:

```
PDESymmetries[eqn, u, {x,t}, R, opts]
finds the symmetry with rank R of a partial differential equation
for the function u.
```

Information about the options of PDESymmetries is obtained by typing `In[8]:= ??WeightedParameters`. It returns:

```
WeightedParameters is an option that determines the parameters with
weight. If WeightedParameters -> {p1,p2,...}, then p1, p2, ... are
considered as constant parameters with weight.
The default is WeightedParameters -> {}.
```

The option `WeightedParameters` is useful when working with systems that lack uniformity in rank. In such cases, our software tries to resolve the problem by itself and prints appropriate messages. When unsuccessful, the program will suggest the use of the `WeightedParameters` option. Therefore, the option `WeightedParameters` should not be used until it is explicitly recommended by the software.

Rules for the weights of variables can be entered via the option `WeightRules`:

```
WeightRules is an option that determines the rules for weights of
the variables. If WeightRules -> {Weight[u] -> val,...}, then scaling
properties are determined under these rules. There is a built in
checking mechanism to see if the given rules cause inconsistency.
```

For PDEs, the `MaxExplicitDependency` option allows one to compute conserved densities or symmetries that explicitly depend on the independent variables:

```
MaxExplicitDependency is an option for finding the invariant and
generalized symmetries of PDEs and DDEs.
If MaxExplicitDependency -> Max_Integer, then the program allows for
explicit dependency of independent variables of maximum degree Max.
The default is MaxExplicitDependency -> 0.
```

1.8 Software Review

In this section, we briefly review software for the computation of conservation laws, higher-order symmetries and recursion operators. In Table **??**, we give a summary and contact information.

Higher-order symmetries can be computed with prolongation methods, and numerous software packages are available that can aid in the tedious computations inherent to such methods. A 50 page survey of software for Lie symmetry computations, including generalized symmetries, can be found in [21], and a short update in [22]. We will not repeat these software reviews here. A survey of packages for conservation laws was first given in [15]. However, to keep this chapter self-contained, we present a summary of that survey.

Based on dilation invariance, Ito's programs in REDUCE (see [25, 26, 27]) compute polynomial higher-order symmetries and conserved densities for systems of evolution equations that are uniform in rank (no weighted parameters can be introduced). Ito's latest program, called *SYMCD*, cannot be used to compute symmetries and densities that depend explicitly on the independent variables t and x, nor can it handle systems with parameters. More details are given in [15].

In [8, 10], Fuchssteiner et al. present algorithms to compute higher-order *symmetries* of evolution equations. Their algorithm in [8] is based on Lie-algebraic techniques and uses commutator algebra on the Lie algebra of vector fields. Their approach is different from the usual prolongation method in that no determining equations are solved. Instead, all necessary generators of the finitely generated Virasoro algebra are computed from one given element by direct Lie-algebraic methods. Their code is available in MuPAD. In [10], Fuchssteiner et al. give code to verify that recursion operators are hereditary. In [8], it is shown how to compute *mastersymmetries* from which the recursion operators can be retrieved.

The REDUCE program *FS* for "formal symmetries" was written by Gerdt and Zharkov [13] (see also [11, 12]). *FS* computes higher-order symmetries and conservation laws of polynomial type. The algorithm requires that the evolution equations be of order two or higher in the spatial variable. However, this approach does not require that the evolution equations be uniform in rank. With *FS*, one cannot compute symmetries that depend explicitly on the independent variables t and x. Applied to equations with parameters, *FS* computes the conditions on the parameters using the symmetry approach.

The PC package *DELiA*, written in Turbo Pascal by Bocharov [3] and co-workers, is a commercial computer algebra system for investigating differential equations using Lie's approach. The program deals with higher-order symmetries, conservation laws, integrability and equivalence problems. It has a special routine for systems of evolution equations. The program requires the presence of second- or higher-order spatial derivative terms in all equations. For systems with parameters, *DELiA* does not automatically compute the densities and symmetries corresponding to the (necessary) conditions on the parameters. One has to use *DELiA*'s integrability test first, to determine the conditions. Once the parameters are fixed, one can compute the densities and symmetries.

Sanders and Wang have Maple and FORM code for the computation of symmetries in the scalar case, allowing zero and negative weights [35, 36, 37] and nonpolynomial equations and symmetries. This code relies on the Maple package *diffalg* [4] to do the reductions of solutions of ODEs (PDEs). See [32] for theoretical foundations of the computation of conservation laws, and [33] for the use of their algorithms in the integrability classification of KdV-type higher order PDEs.

Wolf et al. [41] have three packages, called *CONLAW 1/2/3*, in REDUCE for the computation of conservation laws. There is no limitation on the number of independent variables. The approach uses Wolf's program *CRACK* for solving overdetermined systems of PDEs (see [21, 22]). Wolf's algorithm is particularly efficient for showing the non-existence of conservation laws of high order. In contrast to our program, it also allows one to compute nonpolynomial conservation laws.

Hickman [24] at the University of Canterbury, Christchurch, New Zealand has implemented a slight variation of our algorithm for conserved densities in Maple. Instead of computing the differential monomials in the density by repeated differentiation, Hickman uses a tree structure combining the appropriately weighted building blocks.

1.9 Conclusions

The *Mathematica* package *InvariantsSymmetries.m* presented in this chapter can be used for computer-aided integrability detection of systems of nonlinear PDEs as they occur in various branches of science and engineering.

More precisely, our package is a tool to search for the first half a dozen conservation laws and symmetries. If our programs succeed in finding a large set of independent conservation laws or symmetries, there is a good chance that the system has infinitely many of these quantities. For instance, if the number of conservation laws is 4 or less, most likely the system is not integrable—at least not in its current coordinate representation.

Applied to a system with parameters, our package can determine the conditions on the parameters so that the system admits a sequence of conserved densities or generalized symmetries.

An actual proof of integrability, by showing the existence of an infinity of conservations laws or symmetries, must be done analytically (see [36] for results in this direction). On the other hand, constructing the recursion operator, and showing that it indeed satisfies the defining equation, provides conclusive proof of integrability.

Acknowledgements

We acknowledge helpful discussions with Professors Jan Sanders and Frank Verheest, and thank them for careful reading of the manuscript.

This research project was supported in part by the National Science Foundation of the United States of America under Grant CCR-9625421.

Table 3 List of Software and Contact Information

Name & System	Scope	Developer(s) & Address	Email Address
CONLAW 1/2/3 (REDUCE)	Conservation Laws	T. Wolf *et al.* School of Math. Sci. Queen Mary & Westfield College University of London London E1 4NS, U.K.	T.Wolf@maths.qmw.ac.uk
DELiA (Pascal)	Conservation Laws and Generalized Symmetries	A. Bocharov *et al.* Saltire Software P.O. Box 1565 Beaverton, OR 97075 U.S.A.	alexeib@saltire.com
FS (REDUCE)	Conservation Laws and Generalized Symmetries	V. Gerdt & A. Zharkov Laboratory of Computing Techniques & Automation Joint Institute for Nuclear Research 141980 Dubna, Russia	gerdt@jinr.dubna.su
Invariants Symmetries.m (Mathematica)	Conservation Laws and Generalized Symmetries	Ü. Göktaş & W. Hereman Dept. of Math. Comp. Sci. Colorado School of Mines Golden, CO 80401, U.S.A.	unalg@wolfram.com whereman@mines.edu

Table 3 cont. List of Software and Contact Information

Name & System	Scope	Developer(s) & Address	Email Address
SYMCD (REDUCE)	Conservation Laws and Generalized Symmetries	M. Ito Dept. of Appl. Maths. Hiroshima University Higashi-Hiroshima 724 Japan	ito@puramis.amath. hiroshima-u.ac.jp
symmetry & mastersymmetry (MuPAD)	Generalized Symmetries	B. Fuchssteiner *et al.* Dept. of Mathematics Univ. of Paderborn D-33098 Paderborn Germany	benno@uni-paderborn.de
Tests for Integrability (Maple & FORM)	Conservation Laws, Genera- lized Symmetries, and Recursion Operators	J. Sanders & J.P. Wang Dept. of Math. & Comp. Sci. Vrije Universiteit 1081 HV Amsterdam The Netherlands	jansa@cs.vu.nl
Tools for Conservation Laws (Maple)	Conservation Laws	M. Hickman Dept. of Maths. & Stats. University of Canterbury Private Bag 4800 Christchurch New Zealand	M.Hickman @math.canterbury.ac.nz

References

[1] M. J. Ablowitz and P. A. Clarkson, *Solitons, Nonlinear Evolution Equations and Inverse Scattering*, Cambridge University Press, Cambridge, 1991.

[2] A. H. Bilge, On the equivalence of linearization and formal symmetries as integrability tests for evolution equations, Journal of Physics A: Mathematics and General **26** (1993) 7511–7519.

[3] A. V. Bocharov, DELiA: a system for exact analysis of Differential Equations using S. Lie Approach, DELiA 1.5.1 User Guide, Beaver Soft Programming Team, New York, 1991.

[4] F. Boulier, D. Lazard, F. Ollivier and M. Petitot, Representation for the radical of a finitely generated differential ideal, in: *Proceedings of ISSAC '95 International Symposium on Symbolic and Algebraic Computation*, Ed.: A. H. M. Levelt, ACM Press, New York, 1995, pp. 158–166.

[5] B. Deconinck, P. Meuris and F. Verheest, Oblique nonlinear Alfvén waves in strongly magnetized beam plasmas. Part 1. Nonlinear vector evolution equation, Journal of Plasma Physics **50** (1993) 445–455.

[6] B. Deconinck, P. Meuris and F. Verheest, Oblique nonlinear Alfvén waves in strongly magnetized beam plasmas. Part 2. Soliton solutions and integrability, Journal of Plasma Physics **50** (1993) 457–476.

[7] A. S. Fokas, Symmetries and integrability, Studies in Applied Mathematics **77** (1987) 253–299.

[8] B. Fuchssteiner, S. Ivanov and W. Wiwianka, Algorithmic determination of infinite-dimensional symmetry groups for integrable systems in 1+1 dimensions, Mathematical and Computer Modelling **25** (1997) 91–100.

[9] B. Fuchssteiner and W. Oevel, The bi-Hamiltonian structure of some nonlinear fifth- and seventh-order differential equations and recursion formulas for their symmetries and conserved covariants, Journal of Mathematical Physics **23** (1982) 358–363.

[10] B. Fuchssteiner, W. Oevel and W. Wiwianka, Computer-algebra methods for investigation of hereditary operators of higher order soliton equations, Computer Physics Communications **44** (1987) 47–55.

[11] V. P. Gerdt, Computer algebra, symmetry analysis and integrability of nonlinear evolution equations, International Journal of Modern Physics C **4** (1993) 279–286.

[12] V. P. Gerdt, Homogeneity of integrability conditions for multi-parametric families of polynomial non-linear evolution equations, Mathematics and Computers in Simulation **42** (1996) 399–408.

[13] V. P. Gerdt and A. Y. Zharkov, Computer generation of necessary integrability conditions for polynomial-nonlinear evolution systems, in: *Proceedings of ISSAC '90 International Symposium on Symbolic and Algebraic Computation*, Eds.: S. Watenabe and M. Nagata, ACM Press, New York, 1990, pp. 250–254.

[14] Ü. Göktaş, *Algorithmic Computation of Symmetries, Invariants and*

Recursion Operators for Systems of Nonlinear Evolution Equations and Differential-difference Equations, Ph.D. Thesis, Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, 1998.

[15] Ü. Göktaş and W. Hereman, Symbolic computation of conserved densities for systems of nonlinear evolution equations, Journal of Symbolic Computation 24 (1997) 591–621.

[16] Ü. Göktaş and W. Hereman, Computation of conservation laws for nonlinear lattices, Physica D 123 (1998) 425–436.

[17] Ü. Göktaş and W. Hereman, Algorithmic computation of generalized symmetries for nonlinear evolution and lattice equations, to appear in Advances in Computational Mathematics (1999).

[18] Ü. Göktaş and W. Hereman, The *Mathematica* package *InvariantsSymmetries.m* is available at http://www.mathsource.com/cgi-bin/msitem?0208-932. *MathSource* is the electronic library of Wolfram Research, Inc., Champaign, Illinois, 1998.

[19] Ü. Göktaş, W. Hereman and G. Erdmann, Computation of conserved densities for systems of nonlinear differential-difference equations, Physics Letters A 236 (1997) 30–38.

[20] B. Grammaticos and A. Ramani, Integrability – and how to detect it, in: Integrability of Nonlinear Systems, Eds.: Y. Kosmann-Schwarzbach, B. Grammaticos and K. M. Tamizhmani, Springer Verlag, Berlin, 1997, pp. 30–94.

[21] W. Hereman, Symbolic software for Lie symmetry analysis, in: CRC Handbook of Lie Group Analysis of Differential Equations, Volume 3: New Trends in Theoretical Developments and Computational Methods, Ed.: N. H. Ibragimov, CRC Press, Boca Raton, Florida, 1996, pp. 367–413.

[22] W. Hereman, Review of symbolic software for Lie symmetry analysis, Mathematical and Computer Modelling **25** (1997) 115–132.

[23] W. Hereman, Ü. Göktaş, M. Colagrosso and A. Miller, Algorithmic integrability tests for nonlinear differential and lattice equations, Computer Physics Communications **115** (1998) 428–446.

[24] M. Hickman, Private communication, 1998.

[25] M. Ito, A REDUCE program for finding symmetries of nonlinear evolution equations with uniform rank, Computer Physics Communications **42** (1986) 351–357.

[26] M. Ito, SYMCD - a REDUCE package for finding symmetries and conserved densities of systems of nonlinear evolution equations, Computer Physics Communications **79** (1994) 547–554.

[27] M. Ito and F. Kako, A REDUCE program for finding conserved densities of partial differential equations with uniform rank, Computer Physics Communications **38** (1985) 415–419.

[28] A. V. Mikhailov, A. B. Shabat and V. V. Sokolov, The symmetry approach to classification of integrable equations, in: *What Is Integrability?*, V. E. Zakharov ed., Springer-Verlag, Berlin Heidelberg, 1991, pp. 115–184.

[29] R. M. Miura, C. S. Gardner and M. D. Kruskal, Korteweg-de Vries equations and generalizations. II. Existence of conservation laws and constants of motion, Journal Mathematical Physics **9** (1968) 1204–1209.

[30] P. J. Olver, *Applications of Lie Groups to Differential Equations,* 2nd Edition, Springer Verlag, New York, 1993.

[31] V. Rosenhaus and G. H. Katzin, On symmetries, conservation laws, and variational problems for partial differential equations, Journal of Mathematical Physics **35** (1994) 1998–2012.

[32] J. A. Sanders and J. P. Wang, Hodge decomposition and conservation laws, Mathematics and Computers in Simulation **44** (1997) 483–493.

[33] J. A. Sanders and J. P. Wang, Classification of conservation laws for KdV-like equations, Mathematics and Computers in Simulation **44** (1997) 471–481.

[34] J. A. Sanders and J. P. Wang, On hereditary recursion operators, Report WS-472, Department of Mathematics and Computer Sciences, Vrije Universiteit of Amsterdam, Amsterdam, The Netherlands, 1997.

[35] J. A. Sanders and J. P. Wang, On the classification of integrable systems, in: *Proceeding of the Fourth International Conference on Mathematical and Numerical Aspects of Wave Propagation*, Ed.: J. A. DeSanto, SIAM, Philadelphia (1998) pp. 393–397.

[36] J. A. Sanders and J. P. Wang, On the integrability of homogeneous scalar evolution equations, Journal of Differential Equations **147** (1998) 410–434.

[37] J. A. Sanders and J. P. Wang, Combining Maple and Form to decide on integrability questions, Computer Physics Communications 115 (1998) 447-459.

[38] K. Sawada and T. Kotera, A method for finding N-Soliton solutions of the K.d.V. equation and K.d.V.-like equation, Progress in Theoretical Physics **51** (1974) 1355–1367.

[39] J. P. Wang, *Symmetries and Conservation Laws of Evolution Equations*, Ph.D. Thesis, Department of Mathematics and Computer Sciences, Vrije Universiteit of Amsterdam, Amsterdam, The Netherlands, 1998.

[40] R. Willox, W. Hereman and F. Verheest, Complete integrability of a modified vector derivative nonlinear Schrödinger equation, Physica Scripta **52** (1995) 21–26.

[41] T. Wolf, A. Brand and M. Mohammadzadeh, Computer algebra algorithms and routines for the computation of conservation laws and fixing of gauge in differential expressions, Journal of Symbolic Computation **27** (1999) 221-238.

[42] S. Wolfram, *The Mathematica book*, 3rd Edition, Wolfram Media, Urbana-Champaign, Illinois & Cambridge University Press, London, 1996.