Computation of conservation laws for nonlinear lattices [1]

Ünal Göktaş [2], Willy Hereman [2,3]

Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401-1887, U.S.A.

Abstract

An algorithm to compute polynomial conserved densities of polynomial nonlinear lattices is presented. The algorithm is implemented in *Mathematica* and can be used as an automated integrability test. With the code **diffdens.m**, conserved densities are obtained for several well-known lattice equations. For systems with parameters, the code allows one to determine the conditions on these parameters so that a sequence of conservation laws exist.

Keywords: Conservation law; Integrability; Semi-discrete; Lattice

1 Introduction

There are several motives to find the explicit form of conserved densities of differential-difference equations (DDEs). The first few conservation laws have a physical meaning, such as conservation of momentum and energy. Additional ones facilitate the study of both quantitative and qualitative properties of solutions [0]. Furthermore, the existence of a sequence of conserved densities predicts integrability. Yet, the nonexistence of polynomial conserved quantities does not preclude integrability. Indeed, integrable DDEs could be disguised with a coordinate transformation in DDEs that no longer admit conserved densities of polynomial type [0].

Another compelling argument relates to the numerical solution of partial differential equations (PDEs). In solving integrable discretizations of PDEs, one should check that conserved quantities indeed remain constant. In particular,

[1] Research supported in part by NSF under Grant CCR-9625421.
[2] E-mail: {ugoktas,whereman}@mines.edu
[3] Corresponding author. Phone: (303) 273 3881; Fax: (303) 273 3875.

the conservation of a positive definite quadratic quantity may prevent nonlinear instabilities in the numerical scheme. The use of conservation laws in PDE solvers is well-documented in the literature [0,0,0].

Several methods to test the integrability of DDEs and for solving them are available. Solution methods include symmetry reduction [0] and solving the spectral problem [0] on the lattice. Adaptations of the singularity confinement approach [0], the Wahlquist-Estabrook method [0], and symmetry techniques [0,0,0] allow one to investigate integrability. The most comprehensive integrability study of nonlinear DDEs was done by Yamilov and co-workers (see e.g. [0,0]). Their papers provide a classification of semi-discrete equations possessing infinitely many local conservation laws. Using the formal symmetry approach, they derive the necessary and sufficient conditions for the existence of local conservation laws, and provide an algorithm to construct them.

In [0,0], we gave an algorithm to compute conserved densities for systems of nonlinear evolution equations. The algorithm is based on the concept of invariance of equations under dilation symmetries. Therefore, it is inherently limited to polynomial densities of polynomial systems. Recently an extension of the algorithm towards DDEs was outlined in [0]. Here we present a full description of our algorithm, which can be implemented in any computer algebra language. We also provide details about our software package **diffdens.m** [0] in *Mathematica* [0], which automates the tedious computation of closed-form expressions for conserved densities.

Following basic definitions in Section 2, the algorithm is given in Section 3. To illustrate the method we use the Toda lattice [0], a parameterized Toda lattice [0], and the discretized nonlinear Schrödinger (NLS) equation [0,0,0]. In Section 4, we list results for an extended Lotka-Volterra equation [0], a discretized modified KdV equation [0], a network equation [0], and some other lattices [0]. The features, scope and limitations of our code **diffdens.m** are described in Section 5, together with instructions for the user. In Section 6, we draw some conclusions.

2 Definitions

2.1 Conservation laws

Consider a system of DDEs which is continuous in time, and discretized in the single space variable,

$$\dot{\mathbf{u}}_n = \mathbf{F}(\ldots, \mathbf{u}_{n-1}, \mathbf{u}_n, \mathbf{u}_{n+1}, \ldots), \tag{1}$$

where $\mathbf{u}_n$ and $\mathbf{F}$ are vector dynamical variables with any number of components. For simplicity of notation, the components of $\mathbf{u}_n$ will be denoted by u_n, v_n, etc. We assume that $\mathbf{F}$ is polynomial with constant coefficients. If DDEs are non-polynomial or of higher order in t, we assume that they can be recast in the form (1).

A local *conservation law* is defined by

$$\dot{\rho}_n = J_n - J_{n+1}, \tag{2}$$

which is satisfied on all solutions of (1). The function ρ_n is the *conserved density* and J_n is the associated *flux.* Both are assumed to be polynomials in $\mathbf{u}_n$ and its shifts.

Obviously, $\frac{\mathrm{d}}{\mathrm{dt}}(\sum_n \rho_n) = \sum_n \dot{\rho}_n = \sum_n (J_n - J_{n+1})$, and this telescopic series vanishes for a bounded periodic lattice or a bounded lattice resting at infinity. In that case, $\sum_n \rho_n$ is constant in time. So, we have a conserved quantity.

Example 1 Consider the one-dimensional Toda lattice [0]

$$\ddot{y}_n = \exp{(y_{n-1} - y_n)} - \exp{(y_n - y_{n+1})}, \tag{3}$$

where y_n is the displacement from equilibrium of the nth unit mass under an exponential decaying interaction force between nearest neighbors.

With the change of variables,

$$u_n = \dot{y}_n, \qquad v_n = \exp{(y_n - y_{n+1})}, \tag{4}$$

lattice (3) can be written in polynomial form

$$\dot{u}_n = v_{n-1} - v_n, \qquad \dot{v}_n = v_n(u_n - u_{n+1}). \tag{5}$$

System (5) is completely integrable [0,0]. The first two density-flux pairs

$$\rho_n^{(1)} = u_n, \quad J_n^{(1)} = v_{n-1}, \quad \text{and} \quad \rho_n^{(2)} = \tfrac{1}{2}u_n^2 + v_n, \quad J_n^{(2)} = u_n v_{n-1}, \tag{6}$$

can be easily be computed by hand.

2.2 Equivalence criterion

We introduce a few concepts that will be used in our algorithm. Let D denote the *shift-down* operator and U the *shift-up* operator. Both are defined on the set of all monomials in $\mathbf{u}_n$ and its shifts. If m is such a monomial then $Dm = m|_{n\to n-1}$ and $Um = m|_{n\to n+1}$. For example, $Du_{n+2}v_n = u_{n+1}v_{n-1}$ and $Uu_{n-2}v_{n-1} = u_{n-1}v_n$. It is easy to verify that compositions of D and U define an *equivalence relation* on monomials. Simply stated, all shifted monomials are *equivalent*, e.g. $u_{n-1}v_{n+1} \equiv u_{n+2}v_{n+4} \equiv u_{n-3}v_{n-1}$. This equivalence relation holds for any function of the dependent variables, but for the construction of conserved densities we will apply it only to monomials.

In the algorithm we will use the following *equivalence criterion*: if two monomials m_1 and m_2 are equivalent, $m_1 \equiv m_2$, then $m_1 = m_2 + [M_n - M_{n+1}]$ for some polynomial M_n that depends on $\mathbf{u}_n$ and its shifts. For example, $u_{n-2}u_n \equiv u_{n-1}u_{n+1}$ since $u_{n-2}u_n = u_{n-1}u_{n+1} + [u_{n-2}u_n - u_{n-1}u_{n+1}] = u_{n-1}u_{n+1} + [M_n - M_{n+1}]$, with $M_n = u_{n-2}u_n$.

The *main* representative of an equivalence class is the monomial of that class with n as *lowest* label on u (or v). For example, $u_n u_{n+2}$ is the main representative of the class with elements $u_{n-1}u_{n+1}, u_{n+1}u_{n+3}$, etc. Lexicographical ordering is used to resolve conflicts. For example, $u_n v_{n+2}$ (not $u_{n-2}v_n$) is the main representative in the class with elements $u_{n-3}v_{n-1}, u_{n+2}v_{n+4}$, etc.

3 Algorithm

Scaling invariance, which results from a special Lie-point symmetry, is an intrinsic property of many integrable nonlinear PDEs and DDEs. Indeed, observe that (5), and the couples $\rho_n^{(1)}, J_n^{(1)}$ and $\rho_n^{(2)}, J_n^{(2)}$ in (6) (after inserting them in (2)), are invariant under the dilation (scaling) symmetry

$$(t, u_n, v_n) \rightarrow (\lambda t, \lambda^{-1}u_n, \lambda^{-2}v_n), \tag{7}$$

where λ is an arbitrary parameter. Stated differently, u_n corresponds to one derivative with respect to t; denoted by $u_n \sim \frac{\mathrm{d}}{\mathrm{dt}}$. Similarly, $v_n \sim \frac{\mathrm{d}^2}{\mathrm{dt}^2}$.

Our three-step algorithm exploits the symmetry (7) to find conserved densities.

Step 1: Determine the weights of variables

The *weight*, w, of a variable is by definition equal to the number of derivatives with respect to t the variable carries. Weights are non-negative, rational, and independent of n. Without loss of generality we set $w(\frac{\mathrm{d}}{\mathrm{dt}}) = 1$.

The *rank* of a monomial is defined as the total weight of the monomial. For now we assume that all the terms (monomials) in a particular equation have the same rank. This property is called *uniformity in rank*. The uniformity in rank condition leads to a system of equations for the unknown weights. Different equations in the vector equation (1) may have different ranks.

Example 2 Requiring uniformity in rank for each equation in (5) allows one to compute the weights of the dependent variables. Indeed,

$$w(u_n) + 1 = w(v_n), \quad w(v_n) + 1 = w(u_n) + w(v_n), \tag{8}$$

yields $w(u_n) = 1$, $w(v_n) = 2$, which is consistent with (7).

Step 2: Construct the form of the density

This step involves finding the building blocks (monomials) of a polynomial density with prescribed rank R. All terms in the density must have the same rank R. Since we may introduce parameters with weights (see Example 6), the fact that the density will be a sum of monomials of uniform rank does not necessarily imply that the density must be uniform in rank with respect to the dependent variables.

Let $\mathcal{V}$ be the list of all the variables with positive weights, including parameters with weight. The following procedure is used to determine the form of the density of rank R:

- Form the set $\mathcal{G}$ of all monomials of rank R or less by taking all appropriate combinations of different powers of the variables in $\mathcal{V}$.
- For each monomial in $\mathcal{G}$, introduce the appropriate number of derivatives with respect to t so that all the monomials exactly have weight R. Gather in set $\mathcal{H}$ all the terms that result from computing the various derivatives.
- Identify the monomials that belong to the same equivalence classes and replace them by the main representatives. Call the resulting simplified set $\mathcal{I}$, which consists of the *building blocks* of the density with desired rank R.
- Linear combination of the elements in $\mathcal{I}$ with constant coefficients c_i gives the form of polynomial density of rank R.

Example 3 Continuing with (5), we compute the form of the density of rank $R = 3$. From $\mathcal{V} = \{u_n, v_n\}$ we build $\mathcal{G} = \{u_n^3, u_n^2, u_n v_n, u_n, v_n\}$. Next, introduce t-derivatives, so that each term exactly has rank 3. Thus, using (5),

$$\frac{\mathrm{d}^0}{\mathrm{dt}^0}(u_n^3) = u_n^3, \qquad \frac{\mathrm{d}^0}{\mathrm{dt}^0}(u_n v_n) = u_n v_n,$$

$$\frac{\mathrm{d}}{\mathrm{dt}}({u_n}^2) = 2u_n \dot{u}_n = 2u_n v_{n-1} - 2u_n v_n, \qquad \frac{\mathrm{d}}{\mathrm{dt}}(v_n) = \dot{v}_n = u_n v_n - u_{n+1} v_n,$$

$$\frac{\mathrm{d}^2}{\mathrm{dt}^2}(u_n) = \frac{\mathrm{d}}{\mathrm{dt}}(\dot{u}_n) = \frac{\mathrm{d}}{\mathrm{dt}}(v_{n-1} - v_n) = u_{n-1} v_{n-1} - u_n v_{n-1} - u_n v_n + u_{n+1} v_n.$$

Gather the resulting terms in set $\mathcal{H} = \{u_n^3, u_n v_{n-1}, u_n v_n, u_{n-1} v_{n-1}, u_{n+1} v_n\}$. Since $u_{n+1} v_n \equiv u_n v_{n-1}$ and $u_{n-1} v_{n-1} \equiv u_n v_n$ in $\mathcal{H}$, $u_{n+1} v_n$ is replaced by $u_n v_{n-1}$, and $u_{n-1} v_{n-1}$ by $u_n v_n$. Hence, we obtain $\mathcal{I} = \{u_n^3, u_n v_{n-1}, u_n v_n\}$.

Linear combination of the monomials in $\mathcal{I}$ gives the form of the density:

$$\rho_n = c_1\, u_n^3 + c_2\, u_n v_{n-1} + c_3\, u_n v_n. \tag{9}$$

Step 3: Determine the unknown coefficients in the density

The following procedure simultaneously determines the constants c_i and the form of the flux J_n:

- Compute $\dot{\rho}_n$ and use (1) to remove all the t-derivatives.
- Regarding (2), the resulting expression must match the pattern $J_n - J_{n+1}$. Use the equivalence criterion to modify $\dot{\rho}_n$. The goal is to introduce the main representatives and to identify the terms that match the pattern.
- The part that does not match the pattern must vanish identically for any combination of the components of $\mathbf{u}_n$ and their shifts. This leads to a *linear* system $\mathcal{S}$ in the unknowns c_i. If $\mathcal{S}$ has parameters, careful analysis leads to conditions on these parameters guaranteeing the existence of densities. See [0] for a description of this compatibility analysis.
- The flux is the first piece in the pattern $[J_n - J_{n+1}]$.

Example 4 Carrying on with (5), we determine the coefficients c_1 through c_3 in (9) by requiring that (2) holds. Simultaneously, we determine J_n.

Compute $\dot{\rho}_n$ using (9). Use (5) to remove $\dot{u}_n, \dot{v}_n$, etc. After grouping the terms

$$\begin{aligned}\dot{\rho}_n &= (3c_1 - c_2)u_n^2 v_{n-1} + (c_3 - 3c_1)u_n^2 v_n + (c_3 - c_2)v_{n-1}v_n \\ &\quad + c_2 u_{n-1}u_n v_{n-1} + c_2 v_{n-1}^2 - c_3 u_n u_{n+1} v_n - c_3 v_n^2.\end{aligned} \tag{10}$$

Use the equivalence criterion to modify $\dot{\rho}_n$. For instance, replace $u_{n-1}u_n v_{n-1}$ by $u_n u_{n+1} v_n + [u_{n-1}u_n v_{n-1} - u_n u_{n+1} v_n]$. In terms of main representatives,

$$\begin{aligned}\dot{\rho}_n &= (3c_1 - c_2)u_n^2 v_{n-1} + (c_3 - 3c_1)u_n^2 v_n \\ &\quad + (c_3 - c_2)v_n v_{n+1} + [(c_3 - c_2)v_{n-1}v_n - (c_3 - c_2)v_n v_{n+1}] \\ &\quad + c_2 u_n u_{n+1} v_n + [c_2 u_{n-1}u_n v_{n-1} - c_2 u_n u_{n+1} v_n] \\ &\quad + c_2 v_n^2 + [c_2 v_{n-1}^2 - c_2 v_n^2] - c_3 u_n u_{n+1} v_n - c_3 v_n^2.\end{aligned} \tag{11}$$

Next, group the terms outside of the square brackets and move the pairs inside the square brackets to the bottom. Rearrange the latter terms so that they match the pattern $[J_n - J_{n+1}]$. Hence,

$$\begin{aligned}\dot{\rho}_n &= (3c_1 - c_2)u_n^2 v_{n-1} + (c_3 - 3c_1)u_n^2 v_n \\ &\quad + (c_3 - c_2)v_n v_{n+1} + (c_2 - c_3)u_n u_{n+1} v_n + (c_2 - c_3)v_n^2 \\ &\quad + [\{(c_3 - c_2)v_{n-1}v_n + c_2 u_{n-1}u_n v_{n-1} + c_2 v_{n-1}^2\} \\ &\quad - \{(c_3 - c_2)v_n v_{n+1} + c_2 u_n u_{n+1} v_n + c_2 v_n^2\}].\end{aligned} \tag{12}$$

The first piece inside the square brackets determines

$$J_n = (c_3 - c_2)v_{n-1}v_n + c_2 u_{n-1}u_n v_{n-1} + c_2 v_{n-1}^2. \tag{13}$$

The terms outside the square brackets must all vanish piece by piece, yielding

$$\mathcal{S} = \{3c_1 - c_2 = 0, c_3 - 3c_1 = 0, c_2 - c_3 = 0\}. \tag{14}$$

The solution is $3c_1 = c_2 = c_3$. Since densities can only be determined up to a multiplicative constant, we choose $c_1 = \frac{1}{3}$, $c_2 = c_3 = 1$, and substitute this into (9) and (13). The explicit forms of the density and the flux follow:

$$\rho_n = \tfrac{1}{3}\, u_n^3 + u_n(v_{n-1} + v_n), \qquad J_n = u_{n-1}u_n v_{n-1} + v_{n-1}^2. \tag{15}$$

Example 5 To illustrate how the algorithm works for DDEs with parameters, consider the parameterized Toda lattice

$$\dot{u}_n = \alpha\, v_{n-1} - v_n, \qquad \dot{v}_n = v_n\,(\beta\, u_n - u_{n+1}), \tag{16}$$

where α and β are *nonzero* parameters without weight. In [0] it was shown that (16) is completely integrable if and only if $\alpha = \beta = 1$; but then (16) is (5).

Using our algorithm, one can easily compute the *compatibility conditions* for α and β, so that (16) admits a polynomial conserved densities of, say, rank 3. The steps are the same as for (5). However, (14) must be replaced by

$$\mathcal{S} = \{3\alpha c_1 - c_2 = 0, \beta c_3 - 3c_1 = 0, \alpha c_3 - c_2 = 0, \beta c_2 - c_3 = 0, \alpha c_2 - c_3 = 0\}.$$

A non-trivial solution $3c_1 = c_2 = c_3$ will exist if and only if $\alpha = \beta = 1$.

Analogously, (16) has density $\rho_n^{(1)} = u_n$ of rank 1 if $\alpha = 1$, and density $\rho_n^{(2)} = \frac{\beta}{2}u_n^2 + v_n$ of rank 2 if $\alpha\,\beta = 1$. Only when $\alpha = \beta = 1$ will (16) have conserved densities of rank ≥ 3:

$$\rho_n^{(3)} = \tfrac{1}{3}u_n^3 + u_n(v_{n-1} + v_n), \tag{17}$$

$$\rho_n^{(4)} = \tfrac{1}{4}u_n^4 + u_n^2(v_{n-1} + v_n) + u_n u_{n+1} v_n + \tfrac{1}{2}v_n^2 + v_n v_{n+1}, \tag{18}$$

$$\begin{aligned} \rho_n^{(5)} = {} & \tfrac{1}{5}u_n^5 + u_n^3(v_{n-1} + v_n) + u_n u_{n+1} v_n(u_n + u_{n+1}) \\ & + u_n v_{n-1}(v_{n-2} + v_{n-1} + v_n) + u_n v_n(v_{n-1} + v_n + v_{n+1}). \end{aligned} \tag{19}$$

Ignoring irrelevant shifts in n, these densities agree with the results in [0].

Example 6 In [0,0], Ablowitz and Ladik studied the following integrable discretization of the NLS equation:

$$i\,\dot{u}_n = u_{n+1} - 2u_n + u_{n-1} + u_n^* u_n(u_{n+1} + u_{n-1}), \tag{20}$$

where u_n^* is the complex conjugate of u_n. Instead of splitting u_n into its real and imaginary parts, we treat u_n and $v_n = u_n^*$ as independent variables and augment (20) with its complex conjugate. Absorbing i in the scale on t,

$$\begin{aligned}\dot{u}_n &= u_{n+1} - 2u_n + u_{n-1} + u_n v_n (u_{n+1} + u_{n-1}),\\ \dot{v}_n &= -(v_{n+1} - 2v_n + v_{n-1}) - u_n v_n (v_{n+1} + v_{n-1}).\end{aligned} \tag{21}$$

Since $v_n = u_n^*$ we have $w(v_n) = w(u_n)$. Neither of the equations in (21) is uniform in rank. To circumvent this problem we introduce an auxiliary parameter α with weight, and replace (21) by

$$\begin{aligned}\dot{u}_n &= \alpha(u_{n+1} - 2u_n + u_{n-1}) + u_n v_n (u_{n+1} + u_{n-1}),\\ \dot{v}_n &= -\alpha(v_{n+1} - 2v_n + v_{n-1}) - u_n v_n (v_{n+1} + v_{n-1}).\end{aligned} \tag{22}$$

This extra freedom allows us to impose uniformity in rank:

$$w(u_n) + 1 = w(\alpha) + w(u_n) = 2w(u_n) + w(v_n) = 3w(u_n), \tag{23}$$
$$w(v_n) + 1 = w(\alpha) + w(v_n) = 2w(v_n) + w(u_n) = 3w(v_n), \tag{24}$$

which yields $w(u_n) = w(v_n) = \frac{1}{2}, w(\alpha) = 1$, or, $u_n^2 \sim v_n^2 \sim \alpha \sim \frac{\mathrm{d}}{\mathrm{dt}}$.

We show how to get the building blocks of the density of rank $\frac{3}{2}$. In this case $\mathcal{V} = \{u_n, v_n, \alpha\}$ and $\mathcal{G} = \{u_n, v_n, \alpha, u_n^2, u_n v_n, v_n^2, \alpha u_n, u_n^3, \alpha v_n, u_n^2 v_n, u_n v_n^2, v_n^3\}$. The monomials $\alpha u_n, u_n^3, \alpha v_n, u_n^2 v_n, u_n v_n^2$ and v_n^3 have already rank $\frac{3}{2}$, so no derivatives are needed. The monomials u_n and v_n will have rank $\frac{3}{2}$ after introducing $\frac{\mathrm{d}}{\mathrm{dt}}$. There is no way for the remaining monomials $\alpha, {u_n}^2, u_n v_n$ and ${v_n}^2$ to have rank $\frac{3}{2}$ after differentiation with respect to t. Therefore, they are rejected. The remaining intermediate steps lead to

$$\mathcal{I} = \{\alpha u_n, u_n^3, \alpha v_n, u_n^2 v_n, u_n u_{n+1} v_n, u_n v_{n-1} v_n, u_n v_n^2, v_n^3, u_n u_{n+1} v_{n+1}, u_n v_n v_{n+1}\}.$$

Although uniformity in rank is essential for the first two steps of the algorithm, after the second step, we may already set $\alpha = 1$. The computations now proceed as in the previous example. The trick of introducing one or more extra parameters with weights can always be attempted if any equation in (1) lacks uniformity in rank.

We list some conserved densities of (21):

$$\rho_n^{(1)} = c_1 u_n v_{n-1} + c_2 u_n v_{n+1}, \tag{25}$$

$$\begin{aligned}\rho_n^{(2)} &= c_1\left(\tfrac{1}{2} u_n^2 v_{n-1}^2 + u_n u_{n+1} v_{n-1} v_n + u_n v_{n-2}\right)\\ &\quad + c_2\left(\tfrac{1}{2} u_n^2 v_{n+1}^2 + u_n u_{n+1} v_{n+1} v_{n+2} + u_n v_{n+2}\right),\end{aligned} \tag{26}$$

$$\begin{aligned}\rho_n^{(3)} &= c_1\big[\tfrac{1}{3} u_n^3 v_{n-1}^3 + u_n u_{n+1} v_{n-1} v_n (u_n v_{n-1} + u_{n+1} v_n + u_{n+2} v_{n+1})\\ &\quad + u_n v_{n-1}(u_n v_{n-2} + u_{n+1} v_{n-1}) + u_n v_n (u_{n+1} v_{n-2} + u_{n+2} v_{n-1}) + u_n v_{n-3}\big]\\ &\quad + c_2\big[\tfrac{1}{3} u_n^3 v_{n+1}^3 + u_n u_{n+1} v_{n+1} v_{n+2} (u_n v_{n+1} + u_{n+1} v_{n+2} + u_{n+2} v_{n+3})\\ &\quad + u_n v_{n+2}(u_n v_{n+1} + u_{n+1} v_{n+2}) + u_n v_{n+3}(u_{n+1} v_{n+1} + u_{n+2} v_{n+2}) + u_n v_{n+3}\big].\end{aligned} \tag{27}$$

Our results confirm those in [0]. Also, if defined on an infinite interval, (20) admits infinitely many independent conserved densities [0]. Although it is a constant of motion, we cannot find the Hamiltonian of (20),

$$H = -i \sum_{n} [u_n^*(u_{n-1} + u_{n+1}) - 2\ln(1 + u_n u_n^*)], \tag{28}$$

for it has a logarithmic term [0].

4 More examples

4.1 An extended Lotka-Volterra equation

In [0], Hu and Bullough considered an extended version of the Lotka-Volterra equation:

$$\dot{u}_n = \sum_{r=1}^{k-1} (u_{n-r} - u_{n+r}) u_n. \tag{29}$$

For $k = 2$, (29) is the well-known Lotka-Volterra equation, for which the densities were presented in [0].

We computed five densities of (29) for the case $k = 3$. The first three are:

$$\rho_1 = u_n, \qquad \rho_2 = \frac{1}{2}u_n^2 + u_n(u_{n+1} + u_{n+2}), \tag{30}$$

$$\begin{aligned} \rho_3 = {} & \frac{1}{3}u_n^3 + u_n^2(u_{n+1} + u_{n+2}) + u_n(u_{n+1}^2 + u_{n+2}^2) \\ & + u_n u_{n+1}(u_{n+2} + u_{n+3}) + u_n u_{n+2}(u_{n+1} + u_{n+3} + u_{n+4}). \end{aligned} \tag{31}$$

For $k = 4$, we also computed five densities of (29). The first three are:

$$\rho_1 = u_n, \qquad \rho_2 = \frac{1}{2}u_n^2 + u_n(u_{n+1} + u_{n+2} + u_{n+3}), \tag{32}$$

$$\begin{aligned} \rho_3 = {} & \frac{1}{3}u_n^3 + u_n^2(u_{n+1} + u_{n+2} + u_{n+3}) + u_n(u_{n+1} + u_{n+2} + u_{n+3})^2 \\ & + u_n u_{n+3}(u_{n+4} + u_{n+5} + u_{n+6}) + u_n u_{n+4}(u_{n+1} + u_{n+2}) + u_n u_{n+2} u_{n+5}. \end{aligned} \tag{33}$$

We computed four densities of (29) for $k = 5$. To save space we do not list them here. The integrability and other properties of (29) are discussed in [0].

4.2 A discretized modified KdV equation

In [0], we found the following integrable discretization of the MKdV equation:

$$\dot{u}_n = (1 + u_n^2)(u_{n+1} - u_{n-1}). \tag{34}$$

We computed four densities of (34). The first three are:

$$\rho_n^{(1)} = u_n u_{n+1}, \qquad \rho_n^{(2)} = \tfrac{1}{2} u_n^2 u_{n+1}^2 + u_n u_{n+2}(1 + u_{n+1}^2), \tag{35}$$

$$\begin{aligned}\rho_n^{(3)} = {} & \tfrac{1}{3} u_n^3 u_{n+1}^3 + u_n u_{n+1} u_{n+2}(u_n + u_{n+2})(1 + u_{n+1}^2) \\ & + u_n u_{n+3}(1 + u_{n+1}^2)(1 + u_{n+2}^2).\end{aligned} \tag{36}$$

4.3 Self-dual network equations

The integrable nonlinear self-dual network equations [0,0] can be written as:

$$\dot{u}_n = (1 + u_n^2)(v_n - v_{n-1}), \qquad \dot{v}_n = (1 + v_n^2)(u_{n+1} - u_n). \tag{37}$$

We computed the first four densities of (37). The first three are

$$\rho_n^{(1)} = u_n v_{n-1} + u_n v_n, \tag{38}$$

$$\rho_n^{(2)} = \tfrac{1}{2} u_n^2 (v_{n-1}^2 + v_n^2) + u_n u_{n+1}(1 + v_n^2) + v_n(u_n^2 v_{n-1} + v_{n+1}), \tag{39}$$

$$\begin{aligned}\rho_n^{(3)} = {} & \tfrac{1}{3} u_n^3 (v_{n-1}^3 + v_n^3) + u_n u_{n+1}(u_n v_{n-1} + u_{n+1} v_n + u_n v_n)(1 + v_n^2) \\ & + u_n v_{n-2}(1 + v_{n-1}^2) + u_n v_{n-1} v_n (v_{n-1} + v_n)(1 + u_n^2) \\ & + u_n v_{n+1}(1 + u_{n+1}^2)(1 + v_n^2).\end{aligned} \tag{40}$$

4.4 Generalized lattices

Shabat and Yamilov [0] studied the following integrable Volterra lattice:

$$\dot{u}_n = u_n(v_{n+1} - v_n), \qquad \dot{v}_n = v_n(u_n - u_{n-1}). \tag{41}$$

With our program we computed the first four densities for this system:

$$\rho_n^{(1)} = u_n + v_n, \qquad \rho_n^{(2)} = \tfrac{1}{2}(u_n^2 + v_n^2) + u_n(v_n + v_{n+1}), \tag{42}$$

$$\rho_n^{(3)} = \tfrac{1}{3}(u_n^3 + v_n^3) + u_n^2(v_n + v_{n+1}) + u_n(v_n^2 + v_{n+1}^2) + u_n v_{n+1}(u_{n+1} + v_n), \tag{43}$$

$$\begin{aligned}\rho_n^{(4)} = {} & \tfrac{1}{4}(u_n^4 + v_n^4) + u_n^3(v_n + v_{n+1}) + \tfrac{3}{2} u_n^2 (v_n^2 + v_{n+1}^2) \\ & + u_n(v_n^3 + v_{n+1}^3) + 2 u_n v_{n+1}(u_n + u_{n+1}) \\ & + u_n u_{n+1} v_{n+1}(u_n + u_{n+1} + v_n + v_{n+2}) + u_n v_n v_{n+1}(v_n + v_{n+1}).\end{aligned} \tag{44}$$

In [0], the following Hamiltonian lattice is also listed:

$$\dot{u}_n = u_{n+1} + u_n^2 v_n, \qquad \dot{v}_n = -v_{n-1} - u_n v_n^2. \tag{45}$$

Four densities of (45) are:

$$\rho_n^{(1)} = u_n v_n, \qquad \rho_n^{(2)} = \tfrac{1}{2} u_n^2 v_n^2 + u_n v_{n-1}, \tag{46}$$

$$\rho_n^{(3)} = \tfrac{1}{3} u_n^3 v_n^3 + u_n v_n (u_n v_{n-1} + u_{n+1} v_n) + u_n v_{n-2}, \tag{47}$$

$$\begin{aligned}\rho_n^{(4)} &= \tfrac{1}{4} u_n^4 v_n^4 + u_n^2 v_n^2 (u_n v_{n-1} + u_{n+1} v_n) + \tfrac{1}{2} u_n^2 v_{n-1}^2 + u_n v_{n-3} \\ &\quad + u_n v_n (u_n v_{n-2} + 2u_{n+1} v_{n-1} + u_{n+2} v_n + u_{n+1}^2 v_n v_{n+1}).\end{aligned} \tag{48}$$

5 The Mathematica code diffdens.m

We describe the features, scope and limitations of our program **diffdens.m**, which is written in *Mathematica* syntax [0]. The program has its own *menu* interface with a dozen data files. Users should have access to *Mathematica*. The code **diffdens.m** and the data files [0] must be in the same directory.

5.1 The menu interface

After launching *Mathematica*, type

```
In[1]:= <<diffdens.m
```

to read in the code **diffdens.m** and start the program. Via its menu interface, the program will prompt the user for answers.

The density is available at the end of the computations. To view it in standard *Mathematica* notation, type `rho`. To display it in a more elegant subscript notation, type `subscriptform[rho]`.

5.2 Preparing data files

To test systems that are not in the menu, one should prepare a data file in the format of the data files that are provided with the code. Of course, the name for a new data file should not coincide with any name already listed in the menu, unless one intended to modify those data files.

Example 7 For the parameterized Toda lattice (16) the data file reads:

```
(* Start of data file d_ptoda.m with parameters. *)
(* Toda Lattice with parameters aa and bb *)

u[1][n_]'[t]:= aa*u[2][n-1][t]-u[2][n][t];
u[2][n_]'[t]:= u[2][n][t]*(bb*u[1][n][t]-u[1][n+1][t]);

noeqs = 2;
name = "Toda Lattice (parameterized)";
parameters = {aa,bb};
weightpars = {};

(* The user may give the rank of rho *)
(* and a name for the output file. *)
(* rhorank = 3; *)
(* myfile = "ptodar3.o"; *)

(* The user can give the weights of u[1] and u[2], *)
(* and of the parameters with weight (if applicable). *)
(* weightu[1] = 1; weightu[2] = 2; weight[aa] = 1; *)

formrho = 0;

(* The user can give the form of rho. *)
(* For example, for the density of rank 3: *)
(* formrho = c[1]*u[1][n][t]^3+c[2]*u[1][n][t]*u[2][n-1][t]+
             c[3]*u[1][n][t]*u[2][n][t]; *)

(* End of data file d_ptoda.m *)
```

A brief explanation of the lines in the data file now follows.

```
u[i][n_]'[t] := ...;
```

Give the i^{th} equation of the system in *Mathematica* notation.

```
noeqs = 2;
```

Specify the number of equations in the system.

```
name = "Toda Lattice (parameterized)";
```

Pick a short name for the system under investigation. The quotes are essential.

```
parameters = {aa,bb};
```

Give the list of parameters in the system. If none, set *parameters* = { }.

```
weightpars = {};
```

Give the list of the parameters that are assumed to have weights. Note that weighted parameters are *not* listed in *parameters*, the latter is the list of parameters without weight.

```
(* rhorank = 3; *)
```

Optional. Give the desired rank of the density, if less interactive use of the program is preferred (batch mode).

```
(* myfile = "ptodar3.o"; *)
```

Optional. Give a name of the output file, again to bypass interaction with the program.

```
(* weightu[1] = 1; weightu[2] = 2; *)
```

Optional. Specify a choice for *some or all* of the weights. The program then skips the computation of the weights, but still checks for consistency. Particularly useful if there are several free weights and non-interactive use is preferred.

```
formrho = 0;
```

If *formrho* is set to zero, the program will *compute* the form of ρ_n.

```
formrho = c[1]*u[1][n][t]^3+c[2]*u[1][n][t]*u[2][n-1][t]+
          c[3]*u[1][n][t]*u[2][n][t];
```

Alternatively, one could give a form of ρ_n (here of rank 3). The density must be given in expanded form and with coefficients c[i]. If form of ρ_n is given, the program skips both the computation of the weights and the form of the density. Instead, the code uses what is given and computes the coefficients c[i]. This option allows one, for example, to test densities from the literature.

Anything within `(*` and `*)` is a comment and ignored by *Mathematica.*

Once the data file is ready, pick the choice "`tt) Your System`" in the menu.

5.3 Scope and limitations

Our program can handle systems of first order DDEs that are polynomial in the dependent variables. Only one independent variable (t) is allowed. No terms in the DDEs should have coefficients that *explicitly* depend on t or n. The program only computes polynomial conserved densities in the dependent variables and their shifts, without explicit dependencies on t or n.

Theoretically, there is no limit on the number of DDEs. In practice, for large systems, the computations may take a long time or require a lot of memory. The computational speed depends primarily on the amount of memory.

By design, the program constructs only densities that are uniform in rank. The uniform rank assumption for the monomials in ρ_n allows one to compute *independent* conserved densities piece by piece, without having to split linear combinations of conserved densities. Due to the superposition principle, a linear combination of conserved densities of unequal rank is still a conserved density. This situation arises frequently when parameters with weight are introduced in the DDEs.

The input systems can have one or more parameters, which are assumed to be nonzero. If a system has parameters, the program will attempt to compute the compatibility conditions for these parameters such that densities (of a given rank) exist. The assumption that all parameters in the given DDE must be nonzero is essential. As a result of setting parameters to zero in a given system of DDEs, the weights and the rank of ρ_n might change.

In general, the compatibility conditions for the parameters could be highly nonlinear, and there is no general algorithm to solve them. The program automatically generates the compatibility conditions, and solves them for parameters that occur linearly. Gröbner basis techniques could be used to reduce complicated nonlinear systems into equivalent, yet simpler, non-linear systems. For DDEs with parameters and when the linear system for the unknown coefficients c_i has many equations, the program saves that system and its coefficient matrix, etc., in the file *worklog.m*. Independent from the program, the worklog files can later be analyzed with appropriate *Mathematica* functions.

The assumption that the DDEs are uniform in rank is not very restrictive. If the uniform rank condition is violated, the user can introduce one or more parameters with weights. This also allows for some flexibility in the form of the densities. Although built up with terms that are uniform in rank in the dependent variables and parameters, the densities do no longer have to be uniform in rank with respect to the dependent variables alone. Conversely, when the uniform rank condition *is* fulfilled, the introduction of extra parameters (with weights) in the given DDE leads to a linear combination of conservation laws, not to new ones.

In cases where it is not clear whether or not parameters with weight should be introduced, one should start with parameters without weight. If this causes incompatibilities in the assignment of weights (due to non-uniformity), the program may provide a suggestion. Quite often, it recommends that one or more parameters be moved from the list *parameters* into the list *weightpars* of weighted parameters.

For systems with two or more free weights, the user will be prompted to enter values for the free weights. If only one weight is free, the program will automatically compute some choices for the free weight, and continue with the lowest integer or fractional value. The program selects this value for the free weight; it is just one choice out of possibly infinitely many. If the algorithm fails to determine a suitable value, the user will be prompted to enter a value for the free weight.

Negative weights are not allowed. Zero weights are allowed, but at least one of the dependent variables must have positive weight. The code checks this requirement, and if it is violated the computations are aborted. Note that *fractional weights* and densities of *fractional rank* are permitted.

Our program is a tool in the search of the first half-dozen conservation laws. An existence proof (showing that there are indeed infinitely many conservation laws) must be done analytically. If our program succeeds finding a large set of independent conservation laws, there is a good chance that the system of DDEs has infinitely many conserved densities. If the number of conserved densities is 3 or less, the DDE may have other than polynomial conserved densities, or may not be integrable (in the chosen coordinate representation).

6 Conclusions

We offer the scientific community a *Mathematica* package to carry out the tedious calculations of conserved densities for systems of nonlinear DDEs. The code **diffdens.m**, together with several data and output files, is available via Internet URL [0].

For lattices with parameters, the code automatically determines the compatibility conditions on these parameters so that a sequence of polynomial conserved densities exists.

The existence of a large number of conservation laws is an indicator of integrability of the lattice. Therefore, by generating the compatibility conditions, one can analyze classes of parameterized DDEs and filter out the candidates for complete integrability.

Future generalizations of the algorithm will exploit other symmetries in the hope to find conserved densities of non-polynomial form.

Acknowledgements

Ünal Göktaş thanks Wolfram Research, Inc. for an internship. We acknowledge helpful comments from Drs. B. Herbst, S. Mikhailov, W.-H. Steeb, Y. Suris, P. Winternitz, and R. Yamilov. We also thank G. Erdmann for his help with this project.

References

[1] E.G.B. Hohler and K. Olaussen, Int. J. Mod. Phys. A 11 (1996) 1831.

[2] A.B. Shabat and R.I. Yamilov, Leningrad Math. J. 2 (1991) 377.

[3] F.J. Hickernell, Stud. Appl. Math. 69 (1983) 23.

[4] R.J. LeVeque, Numerical methods for conservation laws (Birkhäuser Verlag, Basel, 1992).

[5] J.M. Sanz-Serna, J. Comput. Phys. 47 (1982) 199.

[6] D. Levi and P. Winternitz, J. Math. Phys. 34 (1993) 3713.

[7] D. Levi and O. Ragnisco, Lett. Nuovo Cimento 22 (1978) 691.

[8] A. Ramani, B. Grammaticos and K.M. Tamizhmani, J. Phys. A: Math. Gen. 25 (1992) L883.

[9] B. Deconinck, Phys. Lett. A 223 (1996) 45.

[10] I. Cherdantsev and R. Yamilov, in: Symmetries and integrability of difference equations, eds. D. Levi, L. Vinet and P. Winternitz (American Mathematical Society, Providence, Rhode Island, 1996) 51.

[11] D. Levi and R. Yamilov, J. Math. Phys. 38 (1997) 6648.

[12] V.V. Sokolov and A.B. Shabat, in: Soviet Sci. Rev. Sec. C, Vol. 4 (Harwood Academic Publishers, New York, 1984) 221.

[13] R. Yamilov, in: Proc. 8th int. workshop on nonlinear evolution equations and dynamical systems (World Scientific Publishing, Singapore, 1993) 423.

[14] Ü. Göktaş and W. Hereman, J. Symb. Comp. 24 (1997) 591.

[15] Ü. Göktaş and W. Hereman. The program condens.m is available via Internet URL: http://www.mines.edu/fs_home/whereman/.

[16] Ü. Göktaş, W. Hereman and G. Erdmann, Phys. Lett. A 236 (1997) 30.

[17] Ü. Göktaş and W. Hereman. The program diffdens.m is available from the WWW site in [15].

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

[19] M. Toda, Theory of nonlinear lattices (Springer Verlag, Berlin, 1981).

[20] M.J. Ablowitz and J.F. Ladik, J. Math. Phys. 17 (1976) 1011.

[21] M.J. Ablowitz and J.F. Ladik, Stud. Appl. Math. 55 (1976) 213.

[22] M.J. Ablowitz and B.M. Herbst, SIAM J. Appl. Math. 50 (1990) 339.

[23] X-B. Hu and R.K. Bullough, J. Phys. A: Math. Gen. 30 (1997) 3635.

[24] M.J. Ablowitz and P.A. Clarkson, Solitons, nonlinear evolution equations and inverse scattering (Cambridge University Press, Cambridge, U.K., 1991).

[25] M. Hénon, Phys. Rev. B 9 (1974) 1921.