

**SYMBOLIC COMPUTATION OF  
CONSERVED DENSITIES AND FLUXES  
FOR SYSTEMS OF PARTIAL  
DIFFERENTIAL EQUATIONS WITH  
TRANSCENDENTAL NONLINEARITIES**

by  
Paul J. Adams

ARTHUR LAKES LIBRARY  
COLORADO SCHOOL OF MINES  
GOLDEN, CO 80401

ProQuest Number: 10794698

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10794698

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 – 1346

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mathematical and Computer Sciences).

Golden, Colorado

Date 01/20/2003

Signed:   
Paul J. Adams

Approved:  1/20/2003  
Dr. Willy Hereman  
Mathematical and Computer Sciences  
Thesis Advisor

Golden, Colorado

Date 1/21/03

  
Dr. Graeme Fairweather  
Head of Department  
Mathematical and Computer Sciences

## ABSTRACT

An algorithm for the symbolic computation of polynomial conserved densities for systems of nonlinear evolution equations has previously been submitted by Ünal Göktaş and Willy Hereman. Here, we submit alterations to the algorithm when transcendental nonlinearities (such as the sine function) are introduced. The algorithm is implemented in *Mathematica* under the name **TransPDEDensityFlux.m**. The software automatically carries out the lengthy symbolic computations for the construction of conserved densities and associated fluxes, and was tested on many transcendental systems of partial differential equations involving well-known equations from soliton theory. The existence of a sequence of conserved densities is a predictor for integrability (solvability) of the PDE system, via the Inverse Scattering Transform.

## TABLE OF CONTENTS

<b>ABSTRACT</b> . . . . .	iii
<b>LIST OF TABLES</b> . . . . .	vi
<b>ACKNOWLEDGMENTS</b> . . . . .	vii
<b>Chapter 1 INTRODUCTION</b> . . . . .	<b>1</b>
<b>Chapter 2 SYSTEMS WITH TRANSC. NONLINEARITIES</b> . . . . .	<b>5</b>
2.1 Conservation Laws . . . . .	5
2.1.1 Notation for Partial Derivatives . . . . .	7
2.2 Algorithm . . . . .	8
2.2.1 Determining the Weights of Variables and Parameters . . . . .	8
2.2.2 Constructing the Form of the Density . . . . .	16
2.2.3 Determining the Unknown Coefficients for the Density . . . . .	27
2.2.4 Calculating the Corresponding Flux . . . . .	35
<b>Chapter 3 SCALAR EQUATIONS WITH MIXED DERIVATIVES</b>	<b>39</b>
<b>Chapter 4 FUNCTIONS ADDED TO EXISTING SOFTWARE</b> . . . . .	<b>43</b>
4.1 Alternative ‘Integration’ to Find the Flux . . . . .	44
4.2 Result of Transcendental Functions . . . . .	46
4.2.1 General Modifications . . . . .	46
4.2.2 Ordinary Differential Equation (ODE) Solver . . . . .	46
<b>Chapter 5 USING THE SOFTWARE</b> . . . . .	<b>49</b>

<b>Chapter 6</b>	<b>RESULTS</b>	<b>57</b>
6.1	Sine-Gordon Systems	57
6.1.1	Single-Equation Case, in 'Characteristic' Coordinates	59
6.1.2	Single-Equation Case, with Weighted Parameter	59
6.1.3	System Case, in 'Laboratory' Coordinates	60
6.2	Sinh-Gordon Systems	62
6.2.1	Single-Equation Case, in 'Characteristic' Coordinates	63
6.2.2	Single-Equation Case, with Weighted Parameter	64
6.2.3	System Case, in 'Laboratory' Coordinates	65
6.3	Liouville Systems	67
6.3.1	Single-Equation Case, in 'Characteristic' Coordinates	68
6.3.2	Single-Equation Case, with Weighted Parameter	69
6.3.3	System Case, in 'Laboratory' Coordinates	71
6.4	Multiple Sine-Gordon Equations	73
6.4.1	Double Sine-Gordon Equation	73
6.4.2	Another Multiple Sine-Gordon Equation	74
6.5	Multiple Liouville Systems	75
6.5.1	Tzetzica Equation without Weighted Parameter	75
6.5.2	Tzetzica Equation with Weighted Parameter	76
6.5.3	Mikhailov System	76
6.5.4	Double Liouville System	77
<b>Chapter 7</b>	<b>CONCLUSION</b>	<b>80</b>
	<b>REFERENCES</b>	<b>81</b>
<b>Appendix A</b>	<b>DATA FILES</b>	<b>85</b>

## LIST OF TABLES

6.1	Results for the sine-Gordon equation without weighted parameter . . .	60
6.2	Results for the sine-Gordon equation with weighted parameter . . . . .	61
6.3	Results for the sine-Gordon system . . . . .	62
6.4	Results for the sinh-Gordon equation without weighted parameter . . .	64
6.5	Results for the sinh-Gordon equation with weighted parameter . . . . .	65
6.6	Results for the sinh-Gordon system . . . . .	66
6.7	Results for the Liouville equation without parameter . . . . .	68
6.8	Results for the Liouville equation without parameter, continued . . .	69
6.9	Results for the Liouville equation with parameter . . . . .	70
6.10	Results for the Liouville equation with parameter, continued . . . . .	71
6.11	Results for the Liouville system . . . . .	72
6.12	Results for the double sine-Gordon equation . . . . .	74
6.13	Results for another multiple sine-Gordon equation . . . . .	74
6.14	Results for the Tzetzica equation without weighted parameter . . . . .	76
6.15	Results for the Tzetzica equation with weighted parameter . . . . .	77
6.16	Results for the Mikhailov system . . . . .	78
6.17	Results for the double Liouville system . . . . .	79

## ACKNOWLEDGMENTS

For his many hours of counseling and mathematical direction, as well as his friendship, I thank Dr. Willy Hereman, my thesis advisor.

Thanks also go to my thesis committee members, Dr. Paul Martin and Dr. Junping Wang for their time and probing beneath the surface of the material and asking the difficult questions.

Without the financial assistance of the National Science Foundation, under Grant CCR-9901929, this research and my studies would not have been possible.

For their ongoing and unwavering support of this and other endeavors, I thank my parents.

Finally, for her patience and understanding of the many late nights that went into my Masters work, I offer my heartfelt gratitude to my wife, Jami.

## Chapter 1

### INTRODUCTION

Investigations of solitary waves and solitons began with their discovery by John Scott Russell in 1834, as he rode his horse beside the narrow Union Canal near Edinburgh, Scotland [1]. His subsequent laboratory work, and that of Stokes, Boussinesq, and Rayleigh, further probed the nature of solitary waves (nonlinear waves that do not change shape as they travel), describing them in terms of equations from fluid dynamics.

The question as to whether equations for water waves allowed for the existence of solitary-wave solutions was finally answered in 1895, when the Dutch physicist Diederik Johannes Korteweg and his student Gustav de Vries derived an equation that supported the existence of solitary waves, which now bears their names. Despite this early derivation of the Korteweg-de Vries (KdV) equation, it was not until 1960 that any new application of the equation was discovered [16].

In 1965, from detailed numerical study, Zabusky and Kruskal [10, 22] found that stable pulse-like waves could exist in a system described by the KdV equation. A remarkable property of these solitary waves was that they could collide with each other and yet preserve their shapes and speeds after the collision. Solitary waves with that property are called solitons. This discovery created renewed interest in the equations for solitary waves and the special properties of their solutions. New more powerful methods for describing the waves mathematically have been developed, and many equations have been found to have solitary waves and solitons as solutions.

In the mid 1920's, Oskar Klein and Felix Gordon [9, 23] derived an equation for a charged particle in an electromagnetic field, using then-new ideas in the realm of quantum theory. From their work several equations addressed in this thesis arise. Perhaps the most well-known example, the sine-Gordon equation, has been seen in the propagation of a dislocation in a crystal, in the modulation of wave packets in a moving medium, and in the propagation of magnetic flux in superconductor equations, among other areas of modern research.

Solitary waves have been observed in a variety of natural realms: in the atmosphere, in oceans, in plasmas, and possibly in nervous systems of living organisms. Finally, solitons started playing an important technological role in modern telecommunications. Their persistent shape and immunity to distortion make them suitable carriers of long-distance signals [18].

Typically, autonomous evolution equations with translation invariance (such as the KdV equation) have only the three 'classical' conserved quantities, namely the mass, the momentum, and the energy. However, the KdV equation has infinitely many conserved quantities. The existence of an infinite sequence of conservation laws for a given system of PDEs suggests that it is completely integrable, though such a condition is not required [6]. Indeed, there are systems (such as the Burgers equation) that can be directly integrated, though possess only a finite number of conservation laws [34]. Additionally, conservation laws provide a simple and efficient method to study both quantitative and qualitative properties of solutions. A comprehensive definition of the term *integrable* is proving to be elusive. Integrable systems are in some sense *exactly solvable* and exhibit globally *regular* solutions for all initial conditions. In contrast, the term *nonintegrable* is, generally, taken to imply that a system can not be *solved exactly* and that its solutions can behave in an *irregular*

fashion due to sensitivity to initial conditions [34].

The method of solution, however, is complex. Though many physical problems are modeled by nonlinear partial differential equations (PDEs), the Fourier transform method is insufficient to solve the problem [1]. In fact, the method of solution for nonlinear PDEs, the Inverse Scattering Transform (IST), would come from a classical scattering problem of quantum mechanics [10]. The computational mechanics of the IST are similar to those involved in the Fourier transform (for solving linear equations), except that the final step of solving the IST is “highly nontrivial” [1].

In this thesis, we present an algorithm for computing conservation laws for systems of nonlinear evolution equations with transcendental nonlinearities. We also present a symbolic software package **TransPDEDensityFlux.M** [2] (in *Mathematica*), which automates the tedious algebra and differential calculus needed in the explicit computations. This is a nontrivial modification of the package **CONDENS.M**, developed by Göktaş and Hereman [16], for polynomial densities and fluxes for systems of polynomial PDEs without transcendental nonlinearities.

In Chapter 2, we describe the algorithm for the computation of conserved densities for systems of nonlinear PDEs with transcendental nonlinearities. In this chapter, we restrict the investigation to those expressed in ‘laboratory coordinates’ (systems without mixed derivatives on the dependent variables). The algorithm is implemented in *Mathematica*, and tested on well-known evolution systems of soliton theory.

In Chapter 3, the algorithm for mixed-derivative equations is addressed, though in less detail than Chapter 2, since many parts of the algorithm remain the same. These equations, though equivalent (through coordinate transformation) to cases addressed in Chapter 2, presented unique challenges for the software.

In Chapter 4, the major functions added and alterations to the original software

are presented and their methods explained. General modifications are discussed, along with specific new functions that were required. One function we added determines the flux associated with a density, without relying on built-in integration functions. This function uses a ‘trick’ to circumvent the sometimes weak built-in integration functions of *Mathematica*. Another function solves a system of linear ordinary differential equation, necessary to determine the density and flux. Due to the computational complexity, this function typically takes the longest of all the functions in the code.

In Chapter 5, we describe the usage of our program. Sample menus and input/output are provided, along with a description of how the user may test their own systems. Additionally, the capabilities and limitations of the code are indicated.

In Chapter 6, we give results for many transcendental PDE systems tested by our new code, along with a brief history and application for each equation or system, if available. In some cases, the efficiency of the software is discussed. Results for five major ‘classes’ of systems with transcendental nonlinearities are presented: the sine-Gordon, sinh-Gordon, Liouville, multiple sine-Gordon, and multiple Liouville equations.

We offer the scientific community a new extension to the symbolic package that carries out the tedious calculations of conserved densities for systems of nonlinear evolution equations. The software (with data and output files) is available from different sources: by anonymous ftp from `mines.edu` in the directory `pub/papers/math_cs_dept/software/TransPDEDensityFlux`, from the ‘Scientific Software’ section of Hereman’s web page at `http://www.mines.edu/fs_home/whereman`, and from the ‘Research’ section of Adams’ web page at `http://personal.idcomm.com/pjadams`.

## Chapter 2

### SYSTEMS WITH TRANSCENDENTAL NONLINEARITIES

#### 2.1 Conservation Laws

The focus of this research and the `TransPDEDensityFlux.M` code is the determination of conservation laws for nonlinear systems of PDEs of the form:

$$\mathbf{u}_t = \mathbf{F}(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{2x}, \dots, \mathbf{u}_{mx}) \quad (2.1)$$

in (single) space  $x$  and time  $t$  variables, where

$$\begin{aligned} \mathbf{u} &= (u_1, u_2, \dots, u_n), \quad \mathbf{F} = (F_1, F_2, \dots, F_n), \\ \mathbf{u}_t &= \frac{\partial \mathbf{u}}{\partial t}, \quad \mathbf{u}_{ix} = \frac{\partial^i \mathbf{u}}{\partial x^i}. \end{aligned}$$

A **conservation law** for (2.1) is an equation of the form

$$D_t \rho + D_x J = 0, \quad (2.2)$$

which is satisfied for all solutions of (2.1), where  $\rho$ , the conserved density, and  $J$ , the associated flux, in general are functions of  $x$ ,  $t$ ,  $\mathbf{u}$  and the partial derivatives of  $\mathbf{u}$  (with respect to  $x$ ).  $D_t$  denotes the total derivative with respect to  $t$  and  $D_x$  the total derivative with respect to  $x$  [1]. If  $\rho$  is a polynomial in  $\mathbf{u}$  and its  $x$  derivatives exclusively, then  $\rho$  is called a local polynomial conserved density. If  $J$  is also such a

polynomial, then (2.2) is called a local polynomial conservation law.

**Example 2.1** The most famous scalar evolution equation from soliton theory, the Korteweg-de Vries (KdV) equation [25],

$$u_t + uu_x + u_{3x} = 0, \quad (2.3)$$

is known to have infinitely many polynomial conservation laws. The first three polynomial conservation laws are given by:

$$\begin{aligned} (u)_t + \left(\frac{1}{2}u^2 + u_{2x}\right)_x &= 0, \\ (u^2)_t + \left(\frac{2}{3}u^3 - u_x^2 + 2uu_{2x}\right)_x &= 0, \\ \left(\frac{1}{3}u^3 - u_x^2\right)_t + \left(\frac{1}{4}u^4 - 2uu_x^2 + u^2u_{2x} + u_{2x}^2 - 2u_xu_{3x}\right)_x &= 0. \end{aligned}$$

The first two express conservation of momentum and energy, respectively, and are relatively easy to compute by hand. The third one, which is less obvious, requires more work.

**Example 2.2** In investigations into PDE systems with transcendental nonlinearities, the most well-known example is the sine-Gordon (SG) equation, here presented in ‘laboratory’ coordinates (later presented in ‘characteristic’ or ‘light-cone’ coordinates),

$$u_{2x} - u_{2t} = \sin u.$$

This equation is also known to have infinitely many polynomial conservation laws. Due to the design of the software (described later), we must introduce a second dependent variable,  $v$ , to transform this equation into the format of (2.1), giving the

coupled system:

$$\begin{aligned} u_t &= v, \\ v_t &= \alpha \sin u + u_{2x}, \end{aligned} \tag{2.4}$$

where  $\alpha$  is a weighted parameter, here equal to  $(-1)$ . The first three polynomial conservation laws for (2.4) are given by:

$$\begin{aligned} (f(u) u_x)_t + (-f(u) v)_x &= 0, \\ (2\alpha \cos(u) + v^2 + u_x^2)_t + (-2v u_x)_x &= 0, \\ (2v u_x)_t + (2\alpha \cos(u) - v^2 - u_x^2)_x &= 0. \end{aligned}$$

Here, we see something not encountered in systems without transcendental functions: coefficients which involve an arbitrary function ( $f(u)$ ) in the dependent variable  $u$ , not the simple constants seen previously in (2.4) [16].

### 2.1.1 Notation for Partial Derivatives

In equations (2.3) and (2.4), as well as all tabulated results in Chapter 4, we use subscripts to indicate partial derivatives, and distinct letters ( $u$  and  $v$ ) for dependent variables. This works well for display, but is cumbersome for algorithms, where it is advantageous to write (2.1) in terms of its component  $u_i$  and to denote partial derivatives with respect to  $x$  by superscripts.

Thus, for a system of  $N$  evolution equations,

$$u_{i,t} = F(u, u_x, u_{2x}, \dots, u_{mx}), \tag{2.5}$$

where

$$\mathbf{u} = (u_1, u_2, \dots, u_N), \quad \mathbf{F} = (F_1, F_2, \dots, F_N),$$

we define

$$u_{i,t} \stackrel{\text{def}}{=} \frac{\partial u_i}{\partial t} \quad \text{and} \quad u_i^{(k)} \stackrel{\text{def}}{=} \frac{\partial^k (u_i)}{\partial x^k}.$$

Also note that the dependent variables are  $u_i$  instead of  $u, v, w, \text{etc.}$

## 2.2 Algorithm

Next, we describe our method to compute conservation laws. The algorithm presented here is based on several sources [17, 22, 26, 30, 31]. Modifications to the algorithm (to handle the transcendental cases) will be noted accordingly, and are based on [29]. The algorithm is defined by four general steps:

- Determining the weights (scaling properties) of variables and parameters
- Constructing the form of the density
- Determining the unknown coefficients for the density,  $\rho$
- Determining the associated flux,  $J$

### 2.2.1 Determining the Weights of Variables and Parameters

Before describing this step, we present the topic of **scaling invariance**. Consider the KdV equation (2.3) [25],

$$u_t + uu_x + u_{3x} = 0. \tag{2.6}$$

Note that the KdV equation is invariant under the scaling symmetry

$$(x, t, u) \rightarrow (\lambda x, \lambda^3 t, \lambda^{-2} u). \quad (2.7)$$

To verify this, let us make the following substitutions into (2.6):

$$\begin{aligned} \tilde{x} &= \lambda x, \quad \tilde{t} = \lambda^3 t, \quad \tilde{u} = \lambda^{-2} u \\ \iff x &= \frac{\tilde{x}}{\lambda}, \quad t = \frac{\tilde{t}}{\lambda^3}, \quad u = \lambda^2 \tilde{u}. \end{aligned} \quad (2.8)$$

Upon substitution for  $u$ , the equation becomes

$$\begin{aligned} \frac{\partial(\lambda^2 \tilde{u})}{\partial t} + (\lambda^2 \tilde{u}) \frac{\partial(\lambda^2 \tilde{u})}{\partial x} + \frac{\partial^3(\lambda^2 \tilde{u})}{\partial x^3} &= 0 \\ \iff \lambda^2 \frac{\partial \tilde{u}}{\partial t} + \lambda^4 \tilde{u} \frac{\partial \tilde{u}}{\partial x} + \lambda^2 \frac{\partial^3 \tilde{u}}{\partial x^3} &= 0. \end{aligned} \quad (2.9)$$

But because  $\tilde{u}(\tilde{x}, \tilde{t})$  does not depend directly on  $t$ , we must apply the chain rule, thus the first term in (2.9) must be re-written as

$$\lambda^2 \frac{\partial \tilde{u}}{\partial \tilde{t}} \frac{\partial \tilde{t}}{\partial t}.$$

Similarly, equation (2.9) becomes:

$$\lambda^2 \frac{\partial \tilde{u}}{\partial \tilde{t}} \frac{\partial \tilde{t}}{\partial t} + \lambda^4 \tilde{u} \frac{\partial \tilde{u}}{\partial \tilde{x}} \frac{\partial \tilde{x}}{\partial x} + \lambda^2 \frac{\partial^3 \tilde{u}}{\partial \tilde{x}^3} \left( \frac{\partial \tilde{x}}{\partial x} \right)^3 = 0. \quad (2.10)$$

We can determine  $\frac{\partial \tilde{t}}{\partial t} = \lambda^3$  and  $\frac{\partial \tilde{x}}{\partial x} = \lambda$  (from our chosen substitutions given in (2.8)). Thus, after replacement, equation (2.10) becomes:

$$\begin{aligned} \lambda^5 \frac{\partial \tilde{u}}{\partial \tilde{t}} + \lambda^5 \tilde{u} \frac{\partial \tilde{u}}{\partial \tilde{x}} + \lambda^5 \frac{\partial^3 \tilde{u}}{\partial \tilde{x}^3} &= 0 \\ \iff \lambda^5 \left( \frac{\partial \tilde{u}}{\partial \tilde{t}} + \tilde{u} \frac{\partial \tilde{u}}{\partial \tilde{x}} + \frac{\partial^3 \tilde{u}}{\partial \tilde{x}^3} \right) &= 0 \\ \iff \frac{\partial \tilde{u}}{\partial \tilde{t}} + \tilde{u} \frac{\partial \tilde{u}}{\partial \tilde{x}} + \frac{\partial^3 \tilde{u}}{\partial \tilde{x}^3} &= 0, \end{aligned}$$

which is equivalent to (2.6), thus showing the equation to be scaling invariant under the scaling in (2.7). Another way to state (2.7) is to say that  $u$  carries the weight of two derivatives with respect to  $x$ , and  $t$  carries the weight of three derivatives with respect to  $x$ . For simplicity, we select the (free) weight of  $\frac{\partial}{\partial x}$  to be equal to 1 in all cases. We thus define the **weight** of a variable as the number of partial derivatives with respect to  $x$  that variable carries, and the **rank** of a monomial as the total weight of that monomial in terms of partial derivatives with respect to  $x$  [22, 26].

Now that we have established that the KdV equation is scaling invariant, we demonstrate a straightforward method by which the scaling symmetry can be determined. To achieve this, for a given system of PDEs, we first try to determine the weights (scaling properties) of the variables in the system.

Introducing some useful notations will make the explanation of the procedure much easier.

- $w$  returns the weight of its argument.

For example,  $w(u_1)$  is the unknown weight of variable  $u_1$ .

- $g$  returns the degree of nonlinearity of its argument in a term.

For example,  $g(u_1) = 3$  if  $u_1$  occurs cubically in a term.

- $d$  returns the number of partial derivatives with respect to its argument in a term. For example, in  $u_1^{(1)}(u_2^{(2)})^2$  we have  $d(x) = 5$ .
- $r_{i,k}$  denotes the rank of the  $k^{\text{th}}$  term in the  $i^{\text{th}}$  equation.

For example,  $r_{1,3}$  is the rank of the third term in the first equation.

Let us look at another example. If we work with the term  $u_{1,t} u_2^2 u_2^{(2)}$ , we have  $g(u_1) = 1$ ,  $g(u_2) = 3$ ,  $d(x) = 2$ , and  $d(t) = 1$ .

Recall that we have defined the weight of a variable in terms of  $x$ -derivatives. Therefore, we can choose  $w\left(\frac{\partial}{\partial x}\right) = 1$ . Hence,  $w\left(\frac{\partial^2}{\partial x^2}\right) = 2, \dots, w\left(\frac{\partial^n}{\partial x^n}\right) = n$ .

**Example 2.3** Returning to the KdV equation, in the algorithm notation, (2.6) becomes

$$u_{1,t} + u_1 u_1^{(1)} + u_1^{(3)} = 0. \quad (2.11)$$

To determine  $w(u_1)$  and  $w\left(\frac{\partial}{\partial t}\right)$  we first compute the ranks of the various terms. For this example, for the term  $u_{1,t}$ :

$$r_{1,1} = 1 w\left(\frac{\partial}{\partial t}\right) + 1 w(u_1),$$

for the term  $u_1 u_1^{(1)}$ :

$$r_{1,2} = 1 w\left(\frac{\partial}{\partial x}\right) + 2 w(u_1) = 1 + 2 w(u_1),$$

and for the term  $u_1^{(3)}$ :

$$r_{1,3} = 3 w\left(\frac{\partial}{\partial x}\right) + 1 w(u_1) = 3 + w(u_1).$$

Then uniformity in rank requires that  $r_{1,1} = r_{1,2} = r_{1,3}$ . Solving this linear system for the unknowns  $w(u_1)$  and  $w\left(\frac{\partial}{\partial t}\right)$  gives

$$w(u_1) = 2, \quad w\left(\frac{\partial}{\partial t}\right) = 3.$$

Alternatively, we could denote this by  $u_1 \sim \frac{\partial^2}{\partial x^2}$ , and  $\frac{\partial}{\partial t} \sim \frac{\partial^3}{\partial x^3}$ . We have thus recovered the scaling symmetry  $(x, t, u_1) \rightarrow (\lambda x, \lambda^3 t, \lambda^{-2} u_1)$  of (2.11).

Most of the time, only the variables  $u_i$  and  $\frac{\partial}{\partial t}$  will have weights. However, not every system of PDEs is scaling invariant. For PDEs that are not *uniform in rank* we use a trick to make them so. The trick is we allow for constant parameters to be introduced and for them to carry weights.

**Example 2.4** Consider the Boussinesq equation written in system form [1]:

$$\begin{aligned} u_{1,t} + u_2^{(1)} &= 0, \\ u_{2,t} + u_1^{(1)} - 3 u_1 u_1^{(1)} - \alpha u_1^{(3)} &= 0. \end{aligned}$$

It is easy to verify that the second equation (2.12) is not uniform in rank. To circumvent the problem we introduce an auxiliary parameter  $\beta$  with (unknown) weight and replace (2.12) by:

$$\begin{aligned} u_{1,t} + u_2^{(1)} &= 0, \\ u_{2,t} + \beta u_1^{(1)} - 3 u_1 u_1^{(1)} - \alpha u_1^{(3)} &= 0, \end{aligned}$$

and compute that  $w(u_1) = 2, w(u_2) = 3, w(\beta) = 2, w\left(\frac{\partial}{\partial x}\right) = 1$ , and  $w\left(\frac{\partial}{\partial t}\right) = 3$ .

Hence, (2.12) is scaling invariant under the transformation:

$$(x, t, u_1, u_2, \beta) \rightarrow (\lambda^1 x, \lambda^2 t, \lambda^{-2} u_1, \lambda^{-3} u_2, \lambda^{-2} \beta).$$

Let us assume that there are  $P$  such parameters in the system. We denote these parameters by  $p_i$ ,  $i = 1, 2, \dots, P$ . Thus, the *extended* list of variables that carry weights is  $\left\{ \frac{\partial}{\partial t}, u_1, u_2, \dots, u_N, p_1, p_2, \dots, p_P \right\}$ .

Next, we explain the procedure for general cases:

**Step (1)** Take the  $i^{\text{th}}$  equation. Suppose it has  $K_i$  terms.

**Step (2)** For the  $k^{\text{th}}$  term in the  $i^{\text{th}}$  equation compute the rank based on the following expression:

$$r_{i,k} = d(x) + d(t) w \left( \frac{\partial}{\partial t} \right) + \sum_{j=1}^N g(u_j) w(u_j) + \sum_{j=1}^P g(p_j) w(p_j), \quad k = 1, 2, \dots, K_i.$$

Recall that  $g$  returns the degree of nonlinearity of its argument in a term. If the variable  $u_j$  and/or the parameter  $p_j$  is missing in a term, then  $g(u_j) = 0$  or  $g(p_j) = 0$ , or both.

**Step (3)** Using uniformity in rank in the  $i^{\text{th}}$  equation, form the linear system:

$$A_i = \{r_{i,1} = r_{i,2} = \dots = r_{i,K_i}\}.$$

**Step (4)** Repeat steps (1) through (3) for all of the equations in the given system.

**Step (5)** Gather the equations  $A_i$  to form the global linear system:  $\mathcal{A} = \bigcup_{i=1}^N A_i$ ,

**Step (6)** Solve  $\mathcal{A}$  for the unknowns  $w(u_j)$ 's,  $w(p_j)$ 's and  $w \left( \frac{\partial}{\partial t} \right)$ .

**Example 2.5** Let us consider an example with a transcendental term, the SG system (2.4), re-stated here in the algorithm notation:

$$u_{1,t} - u_2 = 0, \quad (2.12)$$

$$u_{2,t} - \alpha \sin(u_1) - u_1^{(2)} = 0. \quad (2.13)$$

Before proceeding, we must stop to recognize a key change in the computation of weights due to the appearance of transcendental functions such as sine, cosine, sinh, cosh, and exp. Consider the power series representation of the function  $\sin(u_1)$ :

$$\sin(u_1) = \left( u_1 - \frac{u_1^3}{6} + \frac{u_1^5}{120} - \dots \right).$$

If this series was substituted into (2.13) we get the equation:

$$u_{2,t} - \alpha u_1 + \alpha \frac{u_1^3}{6} - \alpha \frac{u_1^5}{120} + \dots - u_1^{(2)} = 0.$$

For just this (modified) second equation, we have

$$r_{2,2} = 1 w(\alpha) + 1 w(u_1),$$

$$r_{2,3} = 1 w(\alpha) + 3 w(u_1),$$

$$r_{2,4} = 1 w(\alpha) + 5 w(u_1).$$

If we maintain the requirement for uniformity in rank, then  $r_{2,2} = r_{2,3} = r_{2,4}$  and

$$\begin{aligned} 1 w(\alpha) + 1 w(u_1) &= 1 w(\alpha) + 3 w(u_1) = 1 w(\alpha) + 5 w(u_1) \\ \iff 1 w(u_1) &= 3 w(u_1) = 5 w(u_1), \end{aligned}$$

which can only be true if  $w(u_1) = 0$ . Because each transcendental function has a similar power series expansion, we must immediately set to zero the weight of any variable that is the argument of a transcendental function. Consequently, because the  $\frac{\partial}{\partial x}$  has a default weight of 1, we cannot have terms such as  $u_k^{(n)}$  as arguments of these functions in the given systems.

Thus, since we see  $u_1$  as the argument of the sine function in (2.13), we immediately set  $w(u_1) = 0$ , then continue as before.

Now, the uniformity in rank requires that the rank of the second and third terms of (the original) (2.13) be equal:

$$\begin{aligned} r_{2,2} &= 1 w(\alpha) + w(\sin(u_1)) = w(\alpha), \\ r_{2,3} &= 1 w(u_1) + 2 w\left(\frac{\partial}{\partial x}\right) = 2, \\ r_{2,2} &= r_{2,3} \Rightarrow w(\alpha) = 2. \end{aligned}$$

Thus we establish that, for this and other similar systems, we must give a weight  $w(\alpha)$  to  $\alpha$ , in addition to weights on  $u_1$ ,  $u_2$ , and  $\frac{\partial}{\partial t}$ , else we cannot establish uniformity in rank. Doing so, we have (with our assumptions that  $w\left(\frac{\partial}{\partial x}\right) = 1$  and  $w(u_1) = 0$ ):

$$\begin{aligned} r_{1,1} &= 1 w\left(\frac{\partial}{\partial t}\right) + 1 w(u_1) = 1 w\left(\frac{\partial}{\partial t}\right), \\ r_{1,2} &= 1 w(u_2), \\ r_{2,1} &= 1 w\left(\frac{\partial}{\partial t}\right) + 1 w(u_2), \\ r_{2,2} &= 1 w(\alpha) + w(\sin(u_1)) = w(\alpha), \\ r_{2,3} &= 1 w(u_1) + 2 = 2. \end{aligned}$$

Then, we form

$$\begin{aligned} A_1 &= \{r_{1,1} = r_{1,2}\}, \\ A_2 &= \{r_{2,1} = r_{2,2} = r_{2,3}\}, \\ \mathcal{A} &= A_1 \cup A_2. \end{aligned}$$

Next, we solve  $\mathcal{A}$  for  $w(u_2)$ ,  $w\left(\frac{\partial}{\partial t}\right)$ , and  $w(\alpha)$ . This gives

$$w(u_1) = 0, \quad w(u_2) = w\left(\frac{\partial}{\partial t}\right) = 1, \quad w(\alpha) = 2.$$

Alternatively, we denote this by

$$u_2 \sim \frac{\partial}{\partial t} \sim \frac{\partial}{\partial x} \quad \text{and} \quad \alpha \sim \frac{\partial^2}{\partial x^2}.$$

In conclusion, system (2.12)-(2.13) is invariant under the scaling symmetry:

$$\{x, t, u, v, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-1} v, \lambda^{-2} \alpha\}.$$

### 2.2.2 Constructing the Form of the Density

The second step determines the form of the density with a prescribed rank. The conservation law (2.2) will “adopt” the scaling invariance of the PDE, because in the computation of  $D_t \rho$  all time derivative terms are removed using the PDE. In other words, all the terms that appear in the density need to have the same (given) rank. Each conservation law is therefore *uniform in rank*. Densities and fluxes are themselves *uniform in rank*, though their ranks may differ. The rank of the density is arbitrary and can be different from any of the ranks of the equations in the system.

**Example 2.6** Suppose that we work with the KdV equation (2.11) and we try to construct the form of the density with rank 8. Previously, we determined  $w(u_1) = 2$  for the KdV equation, thus we can only have the following powers of  $u_1$ , producing rank 8 or less:

$$\mathcal{G} = \{(1;0), (u_1;2), (u_1^2;4), (u_1^3;6), (u_1^4;8)\},$$

where for each pair the second component denotes the total weight of the first component. Any of the terms in the set  $\mathcal{G}$  involve powers of  $u_1$ . No derivatives with respect to  $x$  have been introduced yet. To create the terms that will have the desired rank, we apply  $\frac{\partial^\ell}{\partial x^\ell}$  to the first components of all of the pairs, where  $\ell = 8, 6, 4, 2, 0$ , respectively. Notice that for each term the number of derivatives,  $\ell$ , is such that it produces terms of exactly weight 8. For example, the term  $u_1^3$ , is already of weight 6, thus we need 2 extra derivatives to give the desired weight of 8. Partial differentiation takes care of the distribution of the derivatives. The result after differentiation is:

$$\begin{aligned} \frac{\partial^8(1)}{\partial x^8} &\rightarrow \{0\}, & \frac{\partial^6(u_1)}{\partial x^6} &\rightarrow \{u_1^{(6)}\}, & \frac{\partial^0(u_1^4)}{\partial x^0} &\rightarrow \{u_1^4\}, \\ \frac{\partial^4(u_1^2)}{\partial x^4} &\rightarrow \{(u_1^{(2)})^2, u_1^{(1)}u_1^{(3)}, u_1u_1^{(4)}\}, & \frac{\partial^2(u_1^3)}{\partial x^2} &\rightarrow \{u_1(u_1^{(1)})^2, u_1^2u_1^{(2)}\}. \end{aligned}$$

The union of these sets is

$$\mathcal{H} = \{0, u_1^{(6)}, (u_1^{(2)})^2, u_1^{(1)}u_1^{(3)}, u_1u_1^{(4)}, u_1(u_1^{(1)})^2, u_1^2u_1^{(2)}, u_1^4\}.$$

Next, we remove the terms from the set  $\mathcal{H}$  that can be written as a derivative with respect to  $x$ , or can be written as a derivative up to terms that were kept (prior) in

the set. Note that

$$\begin{aligned} u_1^2 u_1^{(2)} &= \frac{d}{dx}(u_1^2 u_1^{(1)}) - 2u_1(u_1^{(1)})^2, & 0 &= \frac{d}{dx}(\text{constant}), \\ u_1' u_1^{(3)} &= \frac{d}{dx}(u_1^{(1)} u_1^{(2)}) - (u_1^{(2)})^2, & u_1^{(6)} &= \frac{d}{dx}(u_1^{(5)}), \\ u_1 u_1^{(4)} &= \frac{d}{dx}(u_1 u_1^{(3)} - u_1^{(1)} u_1^{(2)}) + (u_1^{(2)})^2. \end{aligned}$$

Indeed, removing the redundant terms from the set  $\mathcal{H}$ , we have the final set

$$\mathcal{I} = \{(u_1^{(2)})^2, u_1(u_1^{(1)})^2, u_1^4\}.$$

With this simple procedure, we were able to find all the polynomial terms in  $u_1$  and its  $x$ -derivatives such that (i) they have a fixed rank (8 here), and (ii) they cannot be written as derivatives in  $x$ , or as derivatives modulo other terms in the set.

The density is then a linear combination with constant coefficients of the terms in the set  $\mathcal{I}$ . Hence,  $\rho = c_1 (u_1^{(2)})^2 + c_2 u_1(u_1^{(1)})^2 + c_3 u_1^4$ .

We now give a general procedure for constructing the density. Suppose  $\mathcal{V} = \{v_1, v_2, \dots, v_Q\}$  is the sorted list of all variables, including parameters (except  $\frac{\partial}{\partial t}$ ), that have positive weight. The variables are placed in order of descending weight. In order to determine the form of the density of rank  $R$ , we use the following procedure:

**Step (1)** We form all the combinations of the variables in  $\mathcal{V}$  that lead to rank  $R$  or less. Recursively, we form sets consisting of ordered pairs  $(T_{q,s}; W_{q,s})$ , where  $T_{q,s}$  denotes a combination of different powers of the variables, and  $W_{q,s}$  denotes the total weight of  $T_{q,s}$  such that  $W_{q,s} \leq R$ . Set  $\mathcal{B}_0 = \{(1;0)\}$  and proceed with the following.

**For  $q = 1$  through  $Q$  do**

**For  $m = 0$  through  $M - 1$ , where  $M$  is the number of pairs in  $\mathcal{B}_{q-1}$ , do**

Form  $B_{q,m} = \bigcup_{s=0}^{b_{q,m}} \{(T_{q,s}; W_{q,s})\}$  where  $b_{q,m} = \left\lfloor \frac{R-W_{q-1,m}}{w(v_q)} \right\rfloor$  is the maximum allowed power of  $v_q$ ,

$T_{q,s} = T_{q-1,m} v_q^s$ ,  $W_{q,s} = W_{q-1,m} + s w(v_q)$ , where  $(T_{q-1,m}; W_{q-1,m})$  is the  $(m+1)^{\text{th}}$  ordered pair in  $\mathcal{B}_{q-1}$ .

$$\text{Set } \mathcal{B}_q = \bigcup_{m=0}^{M-1} B_{q,m}.$$

**Step (2)** Set  $\mathcal{G} = \mathcal{B}_Q$ . Notice that  $\mathcal{G}$  has the all possible combinations of powers of the variables that produce rank  $R$  or less.

**Step (3)** Now we introduce partial derivatives with respect to  $x$ . For each pair  $(T_{Q,s}; W_{Q,s})$  of  $\mathcal{G}$ , apply  $\frac{\partial^\ell}{\partial x^\ell}$  to the term  $T_{Q,s}$ , if  $\ell = (R - W_{Q,s})$  is an integer, i.e., we introduce just enough partial derivatives so that all the pairs have weight  $R$ . Gathering the terms resulting from computing the various  $\frac{\partial^\ell(T_{Q,s})}{\partial x^\ell}$ , we form the set  $\mathcal{H}$ .

**Step (4)** Removing the terms from  $\mathcal{H}$  that can be written as a derivative with respect to  $x$ , or as a derivative up to terms kept (prior) in the set, we form the final set  $\mathcal{I}$ , which consists of the ‘building blocks’ of the density with the desired rank  $R$ .

**Step (5)** If  $\mathcal{I}$  has  $I$  elements, which are the building blocks of  $\rho$ , then the linear combination of these elements will produce the most general form of the density with rank  $R$ . Therefore,

$$\rho = \sum_{i=1}^I c_i \mathcal{I}(i), \quad (2.14)$$

where  $\mathcal{I}(i)$  is the  $i^{\text{th}}$  element in  $\mathcal{I}$ , and  $c_i$ 's are the coefficients to be determined. This sum applies only when the system is not transcendental. If it is, an adjustment will be made, as noted later.

**Example 2.7** We turn to the Boussinesq equation [1], represented by the system:

$$\begin{aligned} u_{1,t} + u_2^{(1)} &= 0, \\ u_{2,t} + \beta u_1^{(1)} - u_1 u_1^{(1)} - \alpha u_1^{(3)} &= 0, \end{aligned} \quad (2.15)$$

where  $w(u_1) = 2$ ,  $w(\beta) = 2$ , and  $w(u_2) = 3$ . Let us construct the density with rank  $R = 6$ . So, we have  $\mathcal{V} = \{u_2, u_1, \beta\}$ ; hence  $v_1 = u_2$ ,  $v_2 = u_1$  and  $v_3 = \beta$  and, obviously,  $Q = 3$ . We now follow the procedure step by step:

**Step (1)** For  $q = 1$ ,  $m = 0$ :

$b_{1,0} = \left\lfloor \frac{6}{3} \right\rfloor = 2$ . Thus, with  $T_{1,s} = u_2^s$ ,  $W_{1,s} = 3s$  and  $s = 0, 1$ , and 2 we obtain

$$\mathcal{B}_1 = B_{1,0} = \{(1; 0), (u_2; 3), (u_2^2; 6)\}.$$

For  $q = 2$ :

- With  $(1; 0)$ , we have  $b_{2,0} = \left\lfloor \frac{6-0}{2} \right\rfloor = 3$ . So, with

$$T_{2,s} = 1 u_1^s, \quad W_{2,s} = 0 + 2s = 2s, \quad \text{and } s = 0, 1, 2, \text{ and } 3,$$

we get

$$B_{2,0} = \{(1; 0), (u_1; 2), (u_1^2; 4), (u_1^3; 6)\}.$$

- With  $(u_2; 3)$ , we obtain

$$B_{2,1} = \{(u_2; 3), (u_1 u_2; 5)\},$$

since  $b_{2,1} = \left\lfloor \frac{6-3}{2} \right\rfloor = 1$ , and  $T_{2,s} = u_2 u_1^s$ ,  $W_{2,s} = 3 + 2s$ ,  $s = 0, 1$ .

- The pair  $(u_2^2; 6)$  yields to  $b_{2,2} = \left\lfloor \frac{6-6}{2} \right\rfloor = 0$ . Therefore,  $B_{2,2} = \{(u_2^2; 6)\}$ .

Thus,

$$\mathcal{B}_2 = \{(1; 0), (u_1; 2), (u_1^2; 4), (u_1^3; 6), (u_2; 3), (u_1 u_2; 5), (u_2^2; 6)\}.$$

For  $q = 3$ :

Now, we introduce possible powers of  $\beta$ . An analogous procedure leads to

$$B_{3,0} = \{(1; 0), (\beta; 2), (\beta^2; 4), (\beta^3; 6)\},$$

$$B_{3,1} = \{(u_1; 2), (\beta u_1; 4), (\beta^2 u_1; 6)\},$$

$$B_{3,2} = \{(u_1^2; 4), (\beta u_1^2; 6)\},$$

$$B_{3,3} = \{(u_1^3; 6)\},$$

$$B_{3,4} = \{(u_2; 3), (\beta u_2; 5)\},$$

$$B_{3,5} = \{(u_1 u_2; 5)\},$$

$$B_{3,6} = \{(u_2^2; 6)\}.$$

Thus,

$$\begin{aligned} \mathcal{B}_3 = \{ & (1; 0), (\beta; 2), (\beta^2; 4), (\beta^3; 6), (u_1; 2), (\beta u_1; 4), (\beta^2 u_1; 6), (u_1^2; 4), \\ & (\beta u_1^2; 6), (u_1^3; 6), (u_2; 3), (\beta u_2; 5), (u_1 u_2; 5), (u_2^2; 6)\}. \end{aligned}$$

**Step (2)** Set  $\mathcal{G} = \mathcal{B}_3$ .

**Step (3)** Now we attach derivatives to the first components of pairs in  $\mathcal{G}$ . If we compute  $\ell$ , defined in step (3), for each pair of  $\mathcal{G}$ , we obtain

$$\ell = 6, 4, 2, 0, 4, 2, 0, 2, 0, 0, 3, 1, 1, 0,$$

respectively. Notice that in this case all  $\ell$  values are integers. Therefore, we obtain

$$\begin{aligned}
\frac{\partial^6(1)}{\partial x^6} &\rightarrow \{0\}, & \frac{\partial^4(\beta)}{\partial x^4} &\rightarrow \{0\}, & \frac{\partial^2(\beta^2)}{\partial x^2} &\rightarrow \{0\}, \\
\beta^3 &\rightarrow \{\beta^3\}, & \frac{\partial^4(u_1)}{\partial x^4} &\rightarrow \{u_1^{(4)}\}, & \frac{\partial^2(\beta u_1)}{\partial x^2} &\rightarrow \{\beta u_1^{(2)}\}, \\
\beta^2 u_1 &\rightarrow \{\beta^2 u_1\}, & u_2^2 &\rightarrow \{u_2^2\}, & \beta u_1^2 &\rightarrow \{\beta u_1^2\}, \\
u_1^3 &\rightarrow \{u_1^3\}, & \frac{\partial^3(u_2)}{\partial x^3} &\rightarrow \{u_2^{(3)}\}, & \frac{\partial(\beta u_2)}{\partial x} &\rightarrow \{\beta u_2^{(1)}\}, \\
\frac{\partial(u_1 u_2)}{\partial x} &\rightarrow \{u_1 u_2^{(1)}, u_1^{(1)} u_2\}, & \frac{\partial^2(u_1^2)}{\partial x^2} &\rightarrow \{(u_1^{(1)})^2, u_1 u_1^{(2)}\},
\end{aligned}$$

where the terms in the list on the right hand sides come from explicitly computing the partial derivatives. Gathering the sets on the right results in

$$\begin{aligned}
\mathcal{H} = & \{0, \beta^3, u_1^{(4)}, \beta u_1^{(2)}, \beta^2 u_1, (u_1^{(1)})^2, u_1 u_1^{(2)}, \beta u_1^2, \\
& u_1^3, u_2^{(3)}, \beta u_2^{(1)}, u_1 u_2^{(1)}, u_1^{(1)} u_2, u_2^2\}.
\end{aligned}$$

**Step (4)** Removing from  $\mathcal{H}$  the terms that can be written as a derivative with respect to  $x$ , or as a derivative up to terms retained earlier in the set, gives

$$\mathcal{I} = \{\beta^2 u_1, \beta u_1^2, u_1^3, u_2^2, u_1^{(1)} u_2, (u_1^{(1)})^2\},$$

since

$$\begin{aligned}
0 &= \frac{d}{dx}(\text{constant}), & \beta^3 &= \frac{d}{dx}(\beta^3 x), \\
u_1^{(4)} &= \frac{d}{dx}(u_1^{(3)}), & \beta u_1^{(2)} &= \frac{d}{dx}(\beta u_1^{(1)}), \\
u_1 u_1^{(2)} &= \frac{d}{dx}(u_1 u_1^{(1)}) - (u_1^{(1)})^2, & u_2^{(3)} &= \frac{d}{dx}(u_2^{(2)}), \\
\beta u_2^{(1)} &= \frac{d}{dx}(\beta u_2), & u_1 u_2^{(1)} &= \frac{d}{dx}(u_1 u_2) - u_1^{(1)} u_2.
\end{aligned}$$

**Step (5)** Therefore, the form of the density with rank 6 is

$$\rho = c_1 \beta^2 u_1 + c_2 \beta u_1^2 + c_3 u_1^3 + c_4 u_2^2 + c_5 u_1^{(1)} u_2 + c_6 (u_1^{(1)})^2.$$

**Example 2.8** We turn our attention again to the SG system (2.12)-(2.13), where  $w(u_1) = 0$ ,  $w(u_2) = 1$ , and  $w(\alpha) = 2$ . Because of its zero weight,  $u_1$  alone cannot be used in the construction. Instead, we must use  $u_1^{(1)}$ . Thus, the algorithm is as above, but with  $u_1^{(1)}$  (instead of  $u_1$ ) as a member of  $\mathcal{V}$ .

Now, let us construct the density with rank  $R = 2$  based on  $\mathcal{V} = \{\alpha, u_1^{(1)}, u_2\}$ . Thus  $v_1 = \alpha$ ,  $v_2 = u_1^{(1)}$ , and  $v_3 = u_2$ , with  $Q = (\text{length of } \mathcal{V}) = 3$ . We follow the procedure as outlined above:

**Step (1)** For  $q = 1$ ,  $M = 1$ :

Taking  $m = 0$ , we have

$$b_{1,0} = \left\lceil \frac{2-0}{2} \right\rceil = 1, \quad T_{1,s} = 1\alpha^s, \quad W_{1,s} = 2s, \quad \text{and } s = 0, 1$$

$$\Rightarrow \mathcal{B}_1 = B_{1,0} = \{(1; 0), (\alpha; 2)\}.$$

For  $q = 2, M = 2$ :

Taking  $m = 0$ , we work with  $(1; 0)$ , then

$$b_{2,0} = \left[ \frac{2-0}{1} \right] = 2, T_{2,s} = 1u_1^{(1)s}, W_{2,s} = 0 + 1s = 1s, \text{ and } s = 0, 1, 2$$

$$\Rightarrow B_{2,0} = \{(1; 0), (u_1^{(1)}; 1), ((u_1^{(1)})^2; 2)\}.$$

Taking  $m = 1$ , we work with  $(\alpha, 2)$ , then

$$b_{2,1} = \left[ \frac{2-2}{1} \right] = 0, T_{2,s} = \alpha^s, W_{2,s} = 2 + 1s, \text{ and } s = 0$$

$$\Rightarrow B_{2,1} = \{(\alpha; 2)\}$$

$$\Rightarrow B_2 = B_{2,0} \cup B_{2,1} = \{(1; 0), (u_1^{(1)}; 1), ((u_1^{(1)})^2; 2), (\alpha; 2)\}.$$

For  $q = 3, M = 4$ :

Taking  $m = 0$ , we work with  $(1; 0)$ , then

$$b_{3,0} = \left[ \frac{2-0}{1} \right] = 2, T_{3,s} = 1u_2^s, W_{3,s} = 0 + 1s = 1s, \text{ and } s = 0, 1, 2$$

$$\Rightarrow B_{3,0} = \{(1; 0), (u_2; 1), ((u_2)^2; 2)\}.$$

Taking  $m = 1$ , we work with  $(u_1^{(1)}; 1)$ , then

$$b_{3,1} = \left[ \frac{2-1}{1} \right] = 1, T_{3,s} = u_1^{(1)}u_2^s, W_{3,s} = 1 + 1s, \text{ and } s = 0, 1$$

$$\Rightarrow B_{3,1} = \{(u_1^{(1)}; 1), (u_1^{(1)}u_2; 2)\}.$$

Taking  $m = 2$ , we work with  $((u_1^{(1)})^2; 2)$ , then

$$b_{3,2} = \left[ \left[ \frac{2-2}{1} \right] \right] = 0, \quad T_{3,s} = (u_1^{(1)})^2 u_2^s, \quad W_{3,s} = 2 + 1s, \quad \text{and } s = 0$$

$$\Rightarrow B_{3,2} = \{((u_1^{(1)})^2; 2)\}.$$

Taking  $m = 3$ , we work with  $(\alpha; 2)$ , then

$$b_{3,3} = \left[ \left[ \frac{2-2}{1} \right] \right] = 0, \quad T_{3,s} = \alpha u_2^s, \quad W_{3,s} = 2 + 1s, \quad \text{and } s = 0$$

$$\Rightarrow B_{3,3} = \{(\alpha; 2)\}$$

$$\Rightarrow \mathcal{B}_3 = \bigcup_{m=0}^3 B_{3,m} =$$

$$\{(1; 0), (u_2; 1), ((u_2)^2; 2), (u_1^{(1)}; 1), (u_1^{(1)}u_2; 2), ((u_1^{(1)})^2; 2), (\alpha; 2)\}.$$

**Step (2)** Set  $\mathcal{G} = \mathcal{B}_3$ .

**Step (3)** Now we compute the necessary derivatives of the first components of pairs in  $\mathcal{G}$ . If we compute  $\ell$ , defined in step (3), for each pair of  $\mathcal{G}$ , we obtain

$$\ell = 2, 1, 0, 1, 0, 0, 0,$$

respectively. Notice that in this case all  $\ell$  values are integers. Therefore, we obtain

$$\frac{\partial^2(1)}{\partial x^6} \rightarrow \{0\}, \quad \frac{\partial(u_2)}{\partial x} \rightarrow \{u_2^{(1)}\}, \quad (u_2)^2 \rightarrow \{(u_2)^2\}, \quad \frac{\partial u_1^{(1)}}{\partial x} \rightarrow \{u_1^{(2)}\},$$

$$u_1^{(1)}u_2 \rightarrow \{u_1^{(1)}u_2\}, \quad (u_1^{(1)})^2 \rightarrow \{(u_1^{(1)})^2\}, \quad \alpha \rightarrow \{\alpha\},$$

where the terms in the list on the right hand sides come from explicitly computing the partial derivatives. Gathering the sets on the right results in

$$\mathcal{H} = \{0, u_2^{(1)}, (u_2)^2, u_1^{(2)}, u_1^{(1)}u_2, (u_1^{(1)})^2, \alpha\}.$$

**Step (4)** Removing from  $\mathcal{H}$  the terms that can be written as a derivative with respect to  $x$ , or as a derivative up to terms retained earlier in the set, gives

$$\mathcal{I} = \{\alpha, (u_2)^2, u_1^{(1)}u_2, (u_1^{(1)})^2\},$$

since

$$0 = \frac{\partial}{\partial x}(\text{constant}), \quad u_2^{(1)} = \frac{\partial}{\partial x}u_2, \quad u_1^{(2)} = \frac{\partial}{\partial x}u_1^{(1)}.$$

**Step (5)** In the original algorithm the form of the density,  $\rho$ , is a linear combination of the building blocks in  $\mathcal{I}$  with constant coefficients. We must alter it slightly due to the zero weight of the variable  $u_1$ . Constant coefficients were used previously because they would not alter the rank of the terms. Here, we must take a function of  $u_1$  as the coefficient of each term, not just a constant function, since any function of  $u_1$  also has weight of zero. Thus it does not alter the rank of the term to which it is attached. This observation was made by Sanders [29]. The general form of the density is then:

$$\rho = \sum_{i=1}^I h_i(u_k)\mathcal{I}(i). \quad (2.16)$$

Therefore, for rank 6 the candidate density is

$$\rho = h_1(u_1)\alpha + h_2(u_1)(u_2)^2 + h_3(u_1)u_1^{(1)}u_2 + h_4(u_1)(u_1^{(1)})^2. \quad (2.17)$$

We tacitly assume that only one of the dependent variables  $(u_1, u_2, \dots, u_N)$  has weight zero. In future generalization, one could consider systems where more than one dependent variable has weight zero. In such cases, the coefficients in (2.17) would be functions of all the variables with weight zero. It should be noted that because these functions would be in more than one dependent variable, that the linear system of ODEs (similar to those in (2.32)), which must be solved in the next section (also see Section 4.2.2), would become a system of PDEs, thus greatly increasing the complexity of the calculations. Because no such examples were found in the literature, this functionality was not included in `TransPDEDensityFlux.M`.

### 2.2.3 Determining the Unknown Coefficients for the Density

Recall that a conservation law is of the form

$$D_t \rho + D_x J = 0, \quad \iff \quad D_t \rho = -D_x J,$$

which implies that  $D_t \rho$  must be the total derivative of some functional  $(-J)$  with respect to  $x$ . After computation of  $D_t \rho$ , we remove all  $(u_{i,t})^{(j)}$  terms from the expression using the evolution equations in the system (2.5). This is done by solving the equations for  $u_{i,t}$  first, and then computing their partial derivatives with respect to  $x$ . The resulting expression, which we label  $E$ , must be the total derivative of  $(-J)$  with respect to  $x$ . There are two ways to compute  $J$ . The first alternative is to integrate the expression by parts as many times as possible. If there remains a non-integrable part, it must vanish; this will impose conditions on the coefficients  $c_i$  or functions  $h_i(u_k)$ . The second alternative is based on the Euler operator, which allows us to check whether or not an expression is a total derivative. Based on the work by Euler and Lagrange, given the continuously differentiable functions  $y_i(x)$ ,  $(i = 1, 2, \dots, n)$

and the function  $f(x, y_1, \dots, y_1^{(m)}, \dots, y_n, \dots, y_n^{(p)})$ , the following must be true

$$\frac{\partial f}{\partial y_i} - \frac{d}{dx} \left( \frac{\partial f}{\partial (y_i^{(1)})} \right) + \frac{d^2}{dx^2} \left( \frac{\partial f}{\partial (y_i^{(2)})} \right) + \dots + (-1)^n \frac{d^n}{dx^n} \left( \frac{\partial f}{\partial (y_i^{(n)})} \right) = 0, \quad i = 1, 2, \dots, n, \quad (2.18)$$

if and only if the function  $f$  can be integrated with respect to  $x$ . Because this is a necessary condition for the  $D_t \rho$  (after substitution from the system), we apply the **Euler operator** [27]:

$$\mathcal{L}_{u_i} = \frac{\partial}{\partial u_i} - \frac{d}{dx} \left( \frac{\partial}{\partial (u_i^{(1)})} \right) + \frac{d^2}{dx^2} \left( \frac{\partial}{\partial (u_i^{(2)})} \right) + \dots + (-1)^n \frac{d^n}{dx^n} \left( \frac{\partial}{\partial (u_i^{(n)})} \right), \quad (2.19)$$

to  $E$ , for each  $u_i$  separately, then set each result identically equal to zero. This imposes a set of restrictions on the constants  $c_i$  or functional coefficients  $h_i(u_k)$ . Once computed, we substitute the coefficients ( $c_i$  or  $h_i(u_k)$ ) into the candidate density to achieve the explicit conserved density,  $\rho$ .

**Example 2.9** For the KdV equation (2.11), the candidate density with rank 6 is

$$\begin{aligned} \rho &= c_1 u_1^3 + c_2 (u_1^{(1)})^2 \\ \Rightarrow D_t \rho &= 3 c_1 u_1^2 u_{1,t} + 2 c_2 u_1^{(1)} u_{1,t}^{(1)} \end{aligned}$$

After replacement of  $u_{1,t}$  and  $(u_{1,t})^{(1)}$  in (2.20) by  $-(u_1 u_1^{(1)} + u_1^{(3)})$  and  $-(u_1 u_1^{(1)} + u_1^{(3)})^{(1)}$ , respectively, we obtain

$$-D_x J = E = -3 c_1 u_1^2 (u_1 u_1^{(1)} + u_1^{(3)}) - 2 c_2 u_1^{(1)} (u_1 u_1^{(1)} + u_1^{(3)})_x.$$

Then, after (partial) integration with respect to  $x$ ,

$$-D_x J = \left[ -\frac{3}{4} c_1 u_1^4 + (3 c_1 - c_2) u_1 (u_1^{(1)})^2 - 3 c_1 u_1^2 u_1^{(2)} + c_2 (u_1^{(2)})^2 - 2 c_2 u_1^{(1)} u_1^{(3)} \right]_x - (3 c_1 + c_2) (u_1^{(1)})^3. \quad (2.20)$$

Obviously, the last term in (2.20) is non-integrable. We set it equal to zero and solve  $3 c_1 + c_2 = 0$  for  $c_1$ . In  $c_1 = -\frac{1}{3} c_2$ , we have the freedom of choosing  $c_2$ , since any density can only be determined up to an arbitrary multiplicative coefficient. We choose  $c_2 = -1$ . This leads to

$$\begin{aligned} \rho &= \frac{1}{3} u_1^3 - (u_1^{(1)})^2, \\ J &= \frac{1}{4} u_1^4 - 2 u_1 (u_1^{(1)})^2 + u_1^2 u_1^{(2)} + (u_1^{(2)})^2 - 2 u_1^{(1)} u_1^{(3)}. \end{aligned}$$

We are ready to generalize this procedure.

**Step (1)** After computing  $D_t \rho$ , replace all  $(u_{i,t})^{(j)}$ ,  $i = 1, \dots, N$  and  $j = 0, 1, 2, \dots$  from the equations (2.5) of the given system ( $N =$  the number of dependent variables in the system).

**Step (2)** The resulting expression, say  $E$ , must be the total derivative of some functional  $(-J)$  with respect to  $x$ . At this stage we have two options:

1. Carry out all integrations by parts, isolate the non-integrable part, and set it equal to zero. The latter leads to a linear system for the coefficients  $c_i$  or a system of linear ODEs for the functions  $h_i(u_k)$ . This is the option used in Example 2.9.
2. Apply the Euler operator (2.19) [27]  $\mathcal{L}$  to  $E$ . If  $E$  is completely integrable no terms will be left, i.e.  $\mathcal{L}_{\vec{u}}(E) \equiv \vec{0}$ , where  $\vec{u} = [u_1, \dots, u_n]^T$  with  $T$  representing

the transpose; otherwise set the remaining terms equal to zero and form the linear system for the coefficients  $c_i$  or the linear system of ODEs for  $h_i(u_k)$  (as in (2.17)). This option is used in Examples 2.10 and 2.11.

For systems of PDEs without transcendental terms, with either procedure, we obtain a linear system for the coefficients  $\{c_1, c_2, \dots, c_I\}$  appearing in (2.14). For systems of PDEs with transcendental terms we obtain a linear system of ODEs for the functional coefficients  $\{h_1(u_k), h_2(u_k), \dots, h_I(u_k)\}$  appearing in (2.16). We will denote such systems by  $\mathcal{S}$ .

**Step (3)** We solve  $\mathcal{S}$  for the  $c_i$  or  $h_i(u_k)$  terms. Substitution of the nonempty solution into the candidate density (2.14) or (2.16) gives the explicit density.

**Example 2.10** Consider another system without transcendental terms, the coupled KdV equations due to Hirota and Satsuma [19]:

$$\begin{aligned} u_t - 3uu_x + 6vv_x - \frac{1}{2}u_{3x} &= 0, \\ v_t + 3uv_x + v_{3x} &= 0. \end{aligned} \tag{2.21}$$

In the notation of the algorithm, system (2.21) reads

$$\begin{aligned} u_{1,t} - 3u_1u_1^{(1)} + 6u_2u_2^{(1)} - \frac{1}{2}u_1^{(3)} &= 0, \\ u_{2,t} + 3u_1u_2^{(1)} + u_2^{(3)} &= 0. \end{aligned} \tag{2.22}$$

In the usual way, we determine the weights of  $u_1$  and  $u_2$ ;  $w(u_1) = w(u_2) = 2$ , and also compute the form of the candidate density of rank 6:

$$\rho = c_1 u_1^3 + c_2 u_1^2 u_2 + c_3 u_1 u_2^2 + c_4 u_2^3 + c_5 (u_1^{(1)})^2 + c_6 (u_2^{(1)})^2 + c_7 u_2 u_1^{(2)}. \tag{2.23}$$

To determine the coefficients  $c_i$  we first compute  $D_t \rho$ , and then replace  $(u_{1,t})$ ,  $(u_{2,t})$ ,  $(u_{1,t})^{(1)}$ ,  $(u_{2,t})^{(1)}$ ,  $(u_{1,t})^{(2)}$  from the equations in (2.22). This leads to

$$\begin{aligned}
E = & 9c_1 u_1^3 u_1^{(1)} + 6c_2 u_1^2 u_2 u_1^{(1)} + 3c_3 u_1 u_2^2 u_1^{(1)} + 6c_5 (u_1^{(1)})^3 \\
& - 3c_2 u_1^3 u_2^{(1)} - (18c_1 + 6c_3) u_1^2 u_2 u_2^{(1)} - (12c_2 + 9c_4) u_1 u_2^2 u_2^{(1)} \\
& - 6c_3 u_2^3 u_2^{(1)} - (12c_5 + 6c_6) u_1^{(1)} (u_2^{(1)})^2 + 6c_5 u_1 u_1^{(1)} u_1^{(2)} + 9c_7 u_2 u_1^{(1)} u_1^{(2)} \\
& - 3c_7 u_1 u_2^{(1)} u_1^{(2)} - 12c_5 u_2 u_1^{(1)} u_2^{(2)} - 6c_6 u_1 u_2^{(1)} u_2^{(2)} - 18c_7 u_2 u_2^{(1)} u_2^{(2)} \\
& + \frac{3c_1}{2} u_1^2 u_1^{(3)} + (c_2 + 3c_7) u_1 u_2 u_1^{(3)} + \frac{c_3}{2} u_2^2 u_1^{(3)} - c_2 u_1^2 u_2^{(3)} \\
& - 2c_3 u_1 u_2 u_2^{(3)} - (3c_4 + 6c_7) u_2^2 u_2^{(3)} - c_7 u_1^{(2)} u_2^{(3)} + c_5 u_1^{(1)} u_1^{(4)} \\
& - 2c_6 u_2^{(1)} u_2^{(4)} + \frac{c_7}{2} u_2 u_1^{(5)},
\end{aligned}$$

which must be the  $x$ -derivative of a flux  $(-J)$ . Setting  $\mathcal{L}_{\vec{u}}(E) \equiv \vec{0}$  yields the following system of equations  $\mathcal{S}$ :

$$\begin{aligned}
& \{c_2 = 0, \quad 2c_1 + c_3 = 0, \quad 4c_2 + 3c_4 = 0, \\
& c_3 - 4c_5 = 0, \quad c_1 + 2c_5 = 0, \quad c_3 - 12c_5 - 2c_6 = 0, \\
& c_3 + c_6 = 0, \quad c_7 = 0, \quad -c_4 + c_7 = 0, \quad c_2 + 2c_7 = 0\}.
\end{aligned}$$

Solving this linear system leads to the following solution:

$$\left\{c_1 = -\frac{c_3}{2}, c_4 = 0, c_6 = -c_3, c_2 = 0, c_5 = \frac{c_3}{4}, c_7 = 0\right\}.$$

Setting  $c_3 = 1$ , we obtain

$$\left\{c_1 = -\frac{1}{2}, c_2 = 0, c_3 = 1, c_4 = 0, c_5 = \frac{1}{4}, c_6 = -1, c_7 = 0\right\}.$$

After substitution of these coefficients into the form of the density (2.23), we obtain:

$$\rho = -\frac{1}{2}u_1^3 + u_1u_2^2 + \frac{1}{4}(u_1^{(1)})^2 - (u_2^{(1)})^2.$$

**Example 2.11** We turn now to the SG system (2.12)-(2.13). Since the system has a transcendental nonlinearity, Step 3 must be adjusted: instead of using constant coefficients (as in (2.14)) we use functions  $h_i(u_k)$  as coefficients in  $\rho$  (as in (2.16) and thus (2.17)):

$$\rho = h_1(u_1)\alpha + h_2(u_1)(u_2)^2 + h_3(u_1)u_1^{(1)}u_2 + h_4(u_1)(u_1^{(1)})^2, \quad (2.24)$$

where the functions  $h_i(u_1)$  are arbitrary with argument  $u_1$  (they cannot involve derivatives of  $u_1$ ). We now follow the algorithm as described above.

**Step (1)** We compute the total derivative of  $\rho$  with respect to  $t$ :

$$\begin{aligned} D_t\rho &= \alpha h_1'(u_1)u_{1,t} + u_2^2 h_2'(u_1)u_{1,t} + 2h_2(u_1)u_2u_{2,t} + \\ &u_2 h_3'(u_1)u_{1,t}u_1^{(1)} + h_3(u_1)u_{2,t}u_1^{(1)} + h_4'(u_1)u_{1,t}(u_1^{(1)})^2 + \\ &h_3(u_1)u_2(u_{1,t})^{(1)} + 2h_4(u_1)u_1^{(1)}(u_{1,t})^{(1)}. \end{aligned} \quad (2.25)$$

Then we use the original system (2.12)-(2.13) for replacements for all the terms  $(u_{i,t})^{(j)}$ ,  $i = 1, 2$  and  $j = 0$  or  $1$ :

$$\begin{aligned} u_{1,t} &= u_2, \\ u_{2,t} &= \alpha \sin(u_1) + u_1^{(2)}, \\ (u_{1,t})^{(1)} &= \frac{\partial}{\partial x}u_{1,t} = (u_2)^{(1)}. \end{aligned} \quad (2.26)$$

After substitution into (2.25), we get

$$\begin{aligned}
E &= 2\alpha \sin(u_1) h_2(u_1) u_2 + \alpha u_2 h_1'(u_1) + u_2^3 h_2'(u_1) + \\
&\alpha \sin(u_1) h_3(u_1) u_1^{(1)} + u_2^2 h_3'(u_1) u_1^{(1)} + u_2 h_4'(u_1) (u_1^{(1)})^2 + \\
&h_3(u_1) u_2 u_2^{(1)} + 2 h_4(u_1) u_1^{(1)} u_2^{(1)} + 2 h_2(u_1) u_2 u_1^{(2)} + \\
&h_3(u_1) u_1^{(1)} u_1^{(2)}. \tag{2.27}
\end{aligned}$$

**Step (2)** This expression,  $E$ , must be the total derivative of some functional  $(-J)$  with respect to  $x$ . We apply the Euler operator to (2.27), first with respect to  $u_1$ :

$$\begin{aligned}
\mathcal{L}_{u_1}(E) &= 2 h_2(u_1) \alpha \cos(u_1) u_2 + 2 h_2'(u_1) \alpha \sin(u_1) u_2 + h_1''(u_1) \alpha u_2 + \\
&h_2''(u_1) u_2^3 + 2 h_2''(u_1) u_2 (u_1^{(1)})^2 - h_4''(u_1) u_2 (u_1^{(1)})^2 + \\
&h_3''(u_1) (u_1^{(1)})^3 - h_3'(u_1) u_2 u_2^{(1)} + 4 h_2'(u_1) u_1^{(1)} u_2^{(1)} - \\
&2 h_4'(u_1) u_1^{(1)} u_2^{(1)} + 4 h_2'(u_1) u_2 u_1^{(2)} - 2 h_4'(u_1) u_2 u_1^{(2)} + \\
&3 h_3'(u_1) u_1^{(1)} u_1^{(2)} + 2 h_2(u_1) u_2^{(2)} - 2 h_4(u_1) u_2^{(2)}, \tag{2.28}
\end{aligned}$$

and then with respect to  $u_2$ :

$$\begin{aligned}
\mathcal{L}_{u_2}(E) &= 2 h_2(u_1) \alpha \sin(u_1) + \alpha h_1'(u_1) + 3 h_2'(u_1) u_2^2 + h_3'(u_1) u_2 u_1^{(1)} - \\
&h_4'(u_1) (u_1^{(1)})^2 + 2 h_2(u_1) u_1^{(2)} - 2 h_4(u_1) u_1^{(2)}. \tag{2.29}
\end{aligned}$$

For  $E$  to be completely integrable (with respect to  $x$ ), both (2.28) and (2.29) must vanish identically. We gather all like terms. Sorted, (2.28) becomes

$$[3 h_3'(u_1)] u_1^{(1)} u_1^{(2)} + [4 h_2'(u_1) - 2 h_4'(u_1)] u_1^{(1)} u_2^{(1)} +$$

$$\begin{aligned}
& [4 h_2'(u_1) - 2 h_4'(u_1)] u_2 u_1^{(2)} + [-h_3'(u_1)] u_2 u_2^{(1)} + \\
& [2 h_2''(u_1) - h_4''(u_1)] u_2 (u_1^{(1)})^2 + \\
& [2 \cos(u_1) h_2(u_1) + 2 \sin(u_1) h_2'(u_1) + h_1''(u_1)] \alpha u_2 + \\
& [2 h_2(u_1) - 2 h_4(u_1)] u_2^{(2)} + [h_3''(u_1)] (u_1^{(1)})^3 + [h_2''(u_1)] u_2^3 \equiv 0, \quad (2.30)
\end{aligned}$$

and (2.29) becomes

$$\begin{aligned}
& [h_3'(u_1)] u_2 u_1^{(1)} + [2 h_2(u_1) - 2 h_4(u_1)] u_1^{(2)} + [-h_4'(u_1)] (u_1^{(1)})^2 + \\
& [3 h_2'(u_1)] u_2^2 + [2 \sin(u_1) h_2(u_1) + h_1'(u_1)] \alpha \equiv 0. \quad (2.31)
\end{aligned}$$

Next, we set each coefficient (bracketed above), from both (2.30) and (2.31), equal to zero. We collect all the equations and discard duplicates. This results in the following ODE system:

$$\begin{aligned}
h_2'(u_1) &= 0, \\
h_3'(u_1) &= 0, \\
h_4'(u_1) &= 0, \\
h_2''(u_1) &= 0, \\
h_3''(u_1) &= 0, \\
h_2(u_1) - h_4(u_1) &= 0, \\
2 h_2'(u_1) - h_4'(u_1) &= 0, \\
2 h_2''(u_1) - h_4''(u_1) &= 0, \\
2 \sin(u_1) h_2(u_1) + h_1'(u_1) &= 0, \\
2 \cos(u_1) h_2(u_1) + 2 \sin(u_1) h_2'(u_1) + h_1''(u_1) &= 0. \quad (2.32)
\end{aligned}$$

For a detailed account of how the software solves such systems of ODEs, see Section 4.2.2. The solution of system (2.32) is:

$$\begin{aligned} h_1(u_1) &= 2 c_1 \cos(u_1) + c_3, \\ h_2(u_1) &= h_4(u_1) = c_1, \\ h_3(u_1) &= c_2. \end{aligned}$$

Upon substitution into the candidate density (2.24), we get the final conserved density of rank 2 (after removing the trivial piece  $c_3 \alpha$ ),

$$\rho = 2 c_1 \alpha \cos(u) + c_1 (u_2)^2 + c_2 u_1^{(1)} u_2 + c_1 (u_1^{(1)})^2. \quad (2.33)$$

#### 2.2.4 Calculating the Corresponding Flux

To review, by solving system (2.32), we have  $\mathcal{L}_{u_1}(E)$  and  $\mathcal{L}_{u_2}(E)$  (in equations (2.30) and (2.31) respectively) to be equal to zero. This guarantees that  $E = D_t \rho$  can be completely integrated with respect to  $x$ . Thus, it seems a simple matter to find the corresponding flux,  $J$ . We simply set

$$D_x J = -E = -D_t \rho$$

and then integrate  $(-E)$  with respect to  $x$  to get  $J$ . For short expressions a straight integration is possible.

For Example 2.11, the expression  $E$  (corresponding to one of the simplest conserved densities) has only 10 terms. Currently, the built-in integration (by parts) in *Mathematica* is not robust enough to integrate examples much larger than this. We let *Mathematica* attempt the integration, but more often than not the integration

fails and we must turn to another strategy to compute the flux.

We must exploit the uniformity of rank of the conservation law (see Section 2.2.2). Given the rank  $R$  of the density and the weights of  $\frac{\partial}{\partial x}$  and  $\frac{\partial}{\partial t}$ , we have that

$$\text{rank}(J) = R + w \left( \frac{\partial}{\partial t} \right) - w \left( \frac{\partial}{\partial x} \right).$$

So, we can use an algorithm similar to the one in Section 2.2.2, to construct the candidate for  $J$ . Only minor modifications are necessary:

1. In Step 4, no terms can be removed from  $\mathcal{H}$
2. In Step 5, we use functional coefficients  $g_i(u_k)$  (not to be confused with the  $h_i(u_k)$  used previously in  $\rho$ ), regardless of whether the PDE system has or does not have transcendental nonlinearities.

**Example 2.12** Continuing with the rank 2 conserved density from the SG system, the candidate for  $J$  (which coincidentally is also of rank 2) is then:

$$J = g_1(u_1) u_2^{(1)} + g_2(u_1) (u_2)^2 + g_3(u_1) u_1^{(2)} + g_4(u_1) u_1^{(1)} u_2 + g_5(u_1) (u_1^{(1)})^2 + g_6(u_1) \alpha. \quad (2.34)$$

Next, we take the  $x$ -derivative of  $J$ ,

$$\begin{aligned} D_x J = & g_2'(u_1) u_2^2 u_1^{(1)} + g_6'(u_1) \alpha u_1^{(1)} + g_4'(u_1) u_2 (u_1^{(1)})^2 + g_5'(u_1) (u_1^{(1)})^3 + \\ & 2 g_2(u_1) u_2 u_2^{(1)} + g_4(u_1) u_1^{(1)} u_2^{(1)} + g_1'(u_1) u_1^{(1)} u_2^{(1)} + g_4(u_1) u_2 u_1^{(2)} + \\ & 2 g_5(u_1) u_1^{(1)} u_1^{(2)} + g_3'(u_1) u_1^{(1)} u_1^{(2)} + g_1(u_1) u_2^{(2)} + g_3(u_1) u_1^{(3)}, \quad (2.35) \end{aligned}$$

and equate it to  $(-D_t \rho)$ . For this comparison, we use the explicit  $\rho$  from (2.33), take

the  $t$ -derivative, and use (2.26). The result is:

$$\begin{aligned}
 -D_t \rho &= -c_2 \sin(u_1) \alpha u_1^{(1)} - c_2 u_2 u_2^{(1)} - \\
 &2 c_1 u_1^{(1)} u_2^{(1)} - 2 c_1 u_2 u_1^{(2)} - c_2 u_1^{(1)} u_1^{(2)}. \tag{2.36}
 \end{aligned}$$

Alternatively, (2.36) can be obtained directly by substituting (2.33) into (2.27). Next, set  $D_x J = -D_t \rho$ , match up the terms, equate the coefficients, and solve the resulting ODEs for  $g_i(u_1)$ . We get the following system:

$$\text{From } \alpha u_1^{(1)} : g_6'(u_1) = -c_2 \sin(u_1), \tag{2.37}$$

$$\text{From } u_2 u_2^{(1)} : 2 g_2(u_1) = -c_2, \tag{2.38}$$

$$\text{From } u_1^{(1)} u_2^{(1)} : g_4(u_1) + g_1'(u_1) = -2 c_1, \tag{2.39}$$

$$\text{From } u_2 u_1^{(2)} : g_4(u_1) = -2 c_1, \tag{2.40}$$

$$\text{From } u_1^{(1)} u_1^{(2)} : 2 g_5(u_1) + g_3'(u_1) = -c_2, \tag{2.41}$$

$$\text{From } u_2^{(2)} : g_1(u_1) = 0, \tag{2.42}$$

$$\text{From } u_1^{(3)} : g_3(u_1) = 0, \tag{2.43}$$

$$\text{From } u_2^2 u_1^{(1)} : g_2'(u_1) = 0, \tag{2.44}$$

$$\text{From } u_2 (u_1^{(1)})^2 : g_4'(u_1) = 0, \tag{2.45}$$

$$\text{From } (u_1^{(1)})^3 : g_5'(u_1) = 0. \tag{2.46}$$

The resulting solution is:

$$g_1(u_1) = 0 \text{ from (2.42),}$$

$$g_2(u_1) = -\frac{1}{2} c_2 \text{ from (2.38),}$$

$$g_3(u_1) = 0 \text{ from (2.43),}$$

$$\begin{aligned}
g_4(u_1) &= -2 c_1 \quad \text{from (2.39), (2.40),} \\
g_5(u_1) &= -\frac{1}{2} c_2 \quad \text{from (2.41), (2.43),} \\
g_6(u_1) &= c_2 \cos(u_1) \quad \text{from (2.37).}
\end{aligned} \tag{2.47}$$

Some equations were not used to determine the  $g_i(u_1)$ , though it is easily verified that the solution (2.47) satisfies all equations (2.37)-(2.46). Now that we have found the  $g_i(u_1)$ , we substitute the results (2.47) into (2.34) to get the explicit flux,

$$J = -\frac{1}{2} c_2 (u_2)^2 - 2 c_1 u_1^{(1)} u_2 - \frac{1}{2} c_2 (u_1^{(1)})^2 + \alpha c_2 \cos(u_1).$$

Note the results here differ slightly from those reported in Table 6.3, as fractions were removed by replacing all  $c_2$  terms by  $2 c_2$ . This is allowed since any scalar multiple of a pair  $(\rho, J)$  still satisfies the conservation law.

## Chapter 3

### SCALAR EQUATIONS WITH MIXED DERIVATIVES

In the last chapter, we showed how the appearance of transcendental nonlinearities into the system of PDEs forced two changes in the algorithm:

1. Due to the zero weight of the variable ( $u_k$ ) that is the argument of the transcendental function, the form of the density must be generated from  $u_k^{(1)}$  rather than  $u_k$  itself (see Step 1 of ‘Constructing the Form of the Density’).
2. Functional coefficients ( $h_i(u_k)$ ) were used instead of constant coefficients in the candidate density,  $\rho$  (see Step 5 of ‘Constructing the Form of the Density’).

These rules are in effect for systems of the form:

$$\mathbf{u}_t = \mathbf{F}(\mathbf{u}, \mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(m)}) \quad (3.1)$$

in a (single) space variable  $x$  and time  $t$ , and involving transcendental functions, where

$$\begin{aligned} \mathbf{u} &= (u_1, u_2, \dots, u_n), & \mathbf{F} &= (F_1, F_2, \dots, F_n), \\ \mathbf{u}_t &= \frac{\partial \mathbf{u}}{\partial t}, & \mathbf{u}^{(j)} &= \frac{\partial^j \mathbf{u}}{\partial x^j}. \end{aligned}$$

Some systems, like the SG equation in ‘laboratory’ coordinates, which was worked completely in the previous chapter, also have a ‘characteristic’ (or ‘light cone’) coor-

dinate form. Indeed, the SG equation also occurs in the literature [32] as

$$u_{xt} = \alpha \sin(u), \quad (3.2)$$

(given here with weighted parameter,  $\alpha$ ). This ‘mixed derivative’ equation does not fit the class (3.1). Since there are many examples similar to (3.2), we must alter the algorithm to work for equations of the form:

$$u_{xt} = F(u, u^{(1)}, u^{(2)}, \dots, u^{(m)}),$$

where  $F$  is transcendental in  $u$  but polynomial in  $u^{(1)}, \dots, u^{(m)}$ . Note that we will not consider systems (of multiple equations) of this type (only scalar cases).

In fact, the algorithm for these cases is somewhat of a hybrid between the one used for equations like the KdV equation and the one used for systems like the SG equation in laboratory coordinates. It turns out that we will be able to follow the original algorithm for polynomial PDEs closely: we have to account for  $w(u) = 0$ , but the introduction of functional coefficients is unnecessary. In other words, of the two changes in the algorithm (discussed above) we need to use the first one only. The reasons will be explained below.

Due to the detail presented in the last chapter, tedious calculations will be omitted from now onwards.

### Determining the Weights of Variables and Parameters

We first re-write the example (3.2) in the short notation used previously:

$$u_{1,t}^{(1)} = \alpha \sin(u_1). \quad (3.3)$$

As noted in the last chapter, if a transcendental function is present in the system, we immediately set the weight of its argument equal to zero. Thus for (3.3), we immediately set  $w(u_1) = 0$ . Since we always use  $w\left(\frac{\partial}{\partial x}\right) = 1$ , we are left to determine  $w\left(\frac{\partial}{\partial t}\right)$  and  $w(\alpha)$ . We have:

$$\begin{aligned} r_{1,1} &= 1 w(u_1) + 1 w\left(\frac{\partial}{\partial x}\right) + 1 w\left(\frac{\partial}{\partial t}\right) = w\left(\frac{\partial}{\partial t}\right) + 1, \\ r_{1,2} &= 1 w(\alpha) + 1 w(u_1) = w(\alpha). \end{aligned}$$

Since we require  $r_{1,1} = r_{1,2}$ , the simplest solution is

$$\begin{aligned} w\left(\frac{\partial}{\partial t}\right) &= 1, \\ w(\alpha) &= 2. \end{aligned}$$

Note that if the weighted parameter,  $\alpha$ , is omitted from the system, we necessarily have that  $w\left(\frac{\partial}{\partial t}\right) = -1$ . Because we do not use  $w\left(\frac{\partial}{\partial t}\right)$  and  $\frac{\partial}{\partial t}$  in the construction of the candidate density, this would also be acceptable. Obviously, allowing a negative weight for any other variable would create tremendous problems in the determination of the form of the density (refer to Section 2.2.2, ‘Constructing the Form of the Density’).

### Constructing the Form of the Density

Again, because  $w(u_1) = 0$ , we must use  $u_1^{(1)}$  instead of  $u_1$  itself in the construction of the candidate  $\rho$ . Dispensing with the calculations, the form for the density of rank 4 is:

$$\rho = \alpha^2 c_1 + \alpha c_2 (u_1^{(1)})^2 + c_3 (u_1^{(1)})^4 + c_4 (u_1^{(2)})^2. \quad (3.4)$$

Note, even though we have a variable of weight zero, and thus a coefficient of the form  $h_i(u_1)$  in each term would not change the rank of the terms, the use of functional coefficients (instead of constant ones) would present another problem. Suppose that instead of (3.4), we had constructed

$$\rho = \alpha^2 h_1(u_1) + \alpha h_2(u_1) (u_1^{(1)})^2 + h_3(u_1) (u_1^{(1)})^4 + h_4(u_1) (u_1^{(2)})^2. \quad (3.5)$$

The  $t$ -derivative of (3.5) would then be:

$$\begin{aligned} D_t \rho = & \alpha^2 h'_1(u_1) u_{1,t} + \alpha h'_2(u_1) u_{1,t} (u_1^{(1)})^2 + h'_3(u_1) u_{1,t} (u_1^{(1)})^4 + \\ & 2 \alpha h_2(u_1) u_1^{(1)} u_{1,t}^{(1)} + 4 h_3(u_1) (u_1^{(1)})^3 u_{1,t}^{(1)} + h'_4(u_1) u_{1,t} (u_1^{(2)})^2 + \\ & 2 h_4(u_1) u_1^{(2)} u_{1,t}^{(2)}. \end{aligned} \quad (3.6)$$

The next step would be to remove all  $u_{i,t}^{(j)}$  terms from the expression using the original PDE system. Note that we have several occurrences of  $u_{1,t}$  in (3.6). However, there is no way to replace these terms using the original system (3.3). To avoid ever getting such  $u_{i,t}$  terms, we simply revert back to constant coefficients in the formulated density, as in (3.4). Since the form of  $\rho$  is constructed only from the variables  $u_1^{(1)}$  and  $\alpha$ , 'irreplaceable' terms  $u_{i,t}$  will not be generated if constant coefficients are used.

From this point, the final two steps of the general algorithm, 'Determining the Unknown Coefficients for the Density' and 'Determining the Corresponding Flux', remain the same as in the last chapter. For this example, the final result is:

$$\begin{aligned} \rho &= 4 \alpha^2 c_1 + 4 \alpha c_2 (u_1^{(1)})^2 - c_3 (u_1^{(1)})^4 + 4 c_3 (u_1^{(2)})^2, \\ J &= 8 \alpha^2 c_2 \cos(u_1) - 4 \alpha c_3 \cos(u_1) (u_1^{(1)})^2. \end{aligned}$$

## Chapter 4

### MAJOR FUNCTIONS ADDED TO EXISTING SOFTWARE

The **CONDENS.M** [12] code automates the lengthy computation of polynomial conserved densities and fluxes of polynomial systems of nonlinear PDEs. Previous to our contributions outlined here, the package **CONDENS.M** was limited to evolution-type systems of nonlinear PDEs. Indeed, for this code, only one space variable is allowed and the PDE system must be of first order in time (evolution equations).

For systems with parameters, the software can be used to determine the conditions on these parameters so that a sequence of conserved densities exists [14]. Since existence of a large number of conservation laws is a key feature of integrable equations, the code can be used as an integrability tester.

In [14], the algorithm is described and discussed its implementation in detail. Also addressed is the usage of the package, with its capabilities and limitations indicated, along with a comparison to similar programs of other researchers.

The code **CONDENS.M** was tested on many well-known PDEs from soliton theory, and two dozen of these test cases are built into the menu interface of the software. We used our code in a classification of integrable PDEs. In [14] the software is applied to classes of fifth- and seventh-order evolution equations with several free parameters. The software automatically retrieved all the known integrable cases.

Based on the dilation symmetry concept, a third, more comprehensive, *Mathematica* package for the computation of invariants and symmetries of systems of

nonlinear PDEs and DDEs was developed. The prototype program **InvariantsSymmetries.m** [13] is available from MathSource, the *Mathematica* program bank of Wolfram Research, Inc.

Our new software **TransPDEDensityFlux.M** [2] extends the functionality of the **CONDENS.M** package, allowing it to now consider systems with mixed-derivatives (in one space variable  $x$  and one time variable  $t$ ) and transcendental nonlinearities. The addition of these new cases required major modification of the existing software. These modifications are outlined here.

#### 4.1 Alternative ‘Integration’ to Find the Flux

While the integration function in *Mathematica* is robust for single-variable functions, it has considerable difficulty with long expressions in multi-variables, where integration by parts is required. The use of tools such as the Homotopy Operator [27] could greatly advance this area of symbolic computation. However, without such a tool, our ability to compute the flux,  $J$ , associated to a precomputed density,  $\rho$ , seemed uncertain. Take, for example, the conservation law of rank 8 for the SG system, which had (before simplification) 134 terms in the density and 244 terms in the flux. *Mathematica* had no hope to integrate  $J_x$ , which had 611 terms. Instead, to compute  $J$  we exploit the nature of the expected flux,  $J$ .

We go back to the idea of weights, and the general conservation law:

$$D_t \rho + D_x J = 0. \tag{4.1}$$

Just as the weights of all terms in the density,  $\rho$ , had to be equal, so do the terms in

(4.1). That is, uniformity in rank requires

$$\text{rank}(\rho) + w\left(\frac{\partial}{\partial t}\right) = \text{rank}(J) + w\left(\frac{\partial}{\partial x}\right),$$

resulting in

$$\text{rank}(J) = \text{rank}(\rho) + w\left(\frac{\partial}{\partial t}\right) - w\left(\frac{\partial}{\partial x}\right). \quad (4.2)$$

Since we fix the rank of  $\rho$ , and  $w\left(\frac{\partial}{\partial x}\right)$  and  $w\left(\frac{\partial}{\partial t}\right)$  are precomputed, we know the rank of  $J$ . The software re-uses the subroutine which determined the form of  $\rho$  (of a given rank), to compute the form of the corresponding  $J$ .

The subroutine had to be altered to not discard any terms, as it did in the construction of  $\rho$ . In the case of  $\rho$ , extra terms would result in an underdetermined system, thus too much freedom would exist in the final version of the density. However, for  $J$ , because we have no idea what the terms will be, we need to keep all possible terms.

Just as with  $\rho$ , functional coefficients are used in the candidate for  $J$ . Next, we compute  $D_x J$  and compare this with the  $(-D_t \rho)$  we previously sought to integrate. From here, it is a simple matter for the software to generate the system of ODEs for the unknown coefficient functions. Next, the software automatically solves the system of ODEs. As could be assumed, many of the unknown functions will be equal to zero, due to the extra terms in the candidate for  $J$ . For most systems, about half of the coefficient functions were equal to zero. The solutions are then substituted into the candidate for  $J$  to arrive at the final solution for the flux. The final pair  $(\rho, J)$  is tested with (4.1) before the results are reported to the user. See section 2.2.4 for an example of this procedure.

## 4.2 Result of Transcendental Functions

### 4.2.1 General Modifications

The CONDENS.M code, developed by Göktaş and Hereman [16], did not allow for transcendental functions in the original system. Allowing this addition required major modifications in the general algorithm.

First, before determining the weight of each variable, the software checks for any transcendental functions. Acceptable functions are: sine, cosine, sinh, cosh, and exp (at this time, csc, sec, tan, and cot are not allowed). If a transcendental function is detected, the argument of the function is immediately assigned a weight of zero (see section 2.2.1 on scaling invariance).

Second, because of the zero-weight of the dependent variable, functional coefficients were required in place of the constant coefficients in the form of  $\rho$ , for non-mixed-derivative cases (more on this in Section 4.2.2).

Finally, minor modifications were required for many operations from the original software to handle the introduction of the transcendental functions and cases with mixed-derivative terms.

### 4.2.2 Ordinary Differential Equation (ODE) Solver

In the systems investigated, any variable that is the argument of a transcendental function (sin, cos, exp, etc.) must necessarily have a weight of zero. Because of this, functional coefficients were needed in the candidate density. Recall from the algorithm that these functions must be determined to ascertain the final solution for the density. After applying the Euler operator and collecting like terms, instead of having a linear system for unknown constants to determine, the software produces an ODE system for the unknown functions,  $h_i(u_k)$ . The method of solution is described below.

**Example 4.1** Here is the (relatively short) system of ODEs generated by the SG system case, while computing the density of rank 2:

$$h'_2(u_1) = 0, \quad (4.3)$$

$$h'_3(u_1) = 0, \quad (4.4)$$

$$h'_4(u_1) = 0, \quad (4.5)$$

$$h''_2(u_1) = 0, \quad (4.6)$$

$$h''_3(u_1) = 0, \quad (4.7)$$

$$h_2(u_1) - h_4(u_1) = 0, \quad (4.8)$$

$$2 h'_2(u_1) - h'_4(u_1) = 0, \quad (4.9)$$

$$2 h''_2(u_1) - h''_4(u_1) = 0, \quad (4.10)$$

$$2 \sin(u_1) h_2(u_1) + h'_1(u_1) = 0, \quad (4.11)$$

$$2 \cos(u_1) h_2(u_1) + 2 \sin(u_1) h'_2(u_1) + h''_1(u_1) = 0. \quad (4.12)$$

Clearly, equations (4.6) and (4.7) are ‘derivative consequences’ of equations (4.3) and (4.4), respectively. Thus,  $h'_2(u_1) = 0$  implies  $h''_2(u_1) = 0$ , thus the latter can be removed. Similarly, (4.10) and (4.12) are consequences of (4.9) and (4.11). The ‘systemSolver’ function repeatedly searches for redundancies and removes them. This keeps the system small and reduces the possibility for error. The updated system is:

$$h'_2(u_1) = 0, \quad (4.13)$$

$$h'_3(u_1) = 0, \quad (4.14)$$

$$h'_4(u_1) = 0, \quad (4.15)$$

$$2 h'_2(u_1) - h'_4(u_1) = 0, \quad (4.16)$$

$$h_2(u_1) - h_4(u_1) = 0, \quad (4.17)$$

$$2 \sin(u_1) h_2(u_1) + h_1'(u_1) = 0. \quad (4.18)$$

The function will sort the ODEs according to length first and then work on one equation at a time, starting with the shortest. After the shortest equation is solved, the result is first sorted and then substituted into the system, the system is re-sorted, and the cycle repeats.

In the above example, (4.13) through (4.15) are one-term equations, so one is chosen and solved. From (4.13) we deduce that  $h_2(u_1) = c_1$  (some constant). Similarly,  $h_3(u_1) = c_2$  and  $h_4(u_1) = c_3$ , giving us (after substitution):

$$c_1 - c_3 = 0, \quad (4.19)$$

$$2 c_1 \sin(u_1) + h_1'(u_1) = 0. \quad (4.20)$$

Equation (4.19) does not give us any information about the  $h_i(u_1)$  functions, but it does provide important information. It is removed from the system, stored in a linear system with other such equations, and solved separately. Equation (4.20) leads to  $h_1(u_1) = 2 c_1 \cos(u_1) + c_3$ .

The results (after incorporating the linear system results):

$$h_1(u_1) = 2 c_1 \cos(u_1) + c_3,$$

$$h_2(u_1) = h_4(u_1) = c_1,$$

$$h_3(u_1) = c_2.$$

Because there is no restriction on the constants  $c_1$  and  $c_2$ , they are left in the final result.

## Chapter 5

### USING THE SOFTWARE

In this chapter, a brief demonstration of the **TransPDEDensityFlux.M** [2] program is given, along with an explanation of each step. The software is written in *Mathematica* [33] and is designed to carry out the calculations needed to determine conservation laws for systems of nonlinear evolution equations with or without transcendental functions. These systems may now include certain transcendental functions such as sine, exp, and sinh, subject to limitations noted below. The limitations of the software will be briefly discussed at the end of the chapter.

The software has its own menu interface, which should make its use relatively simple to the user. However, users will need to ensure that the **TransPDEDensityFlux.M** and all data files are located in the same directory. Familiarity with *Mathematica*, though not required, will prove useful. Those users unfamiliar with *Mathematica* should note that the activation of *Mathematica* commands requires holding the ⟨Shift⟩ key while pressing ⟨Enter⟩.

#### Starting the software

To initiate the software, open a new *Mathematica* session and enter the following:

```
SetDirectory["c:\target"];  
<<transpde.m
```

where `c:\target` is replaced by the location where the code and data files are stored.

After executing this command, all user interaction will be through prompts from the software.

### Selecting a case and rank

The first interaction will be the selection of a system from the menu:

```

*** MENU INTERFACE *** (page: 3)
-----
31) Sine-Gordon Equation (d_sineNP.m)
32) Sine-Gordon Equation w/ weighted parameter (d_sineP.m)
33) Sine-Gordon System (d_sineSY.m)
34) Hyperbolic Sine-Gordon Equation (d_sinhNP.m)
.
.
nn) Next Page
tt) Your System
qq) Exit the Program
-----

```

The user is prompted with ENTER YOUR CHOICE: , to which the user responds with the number in the menu corresponding to the system desired. For example, for the sine-Gordon equation, the user enters 31. Next, the user is offered the opportunity to store the session in a log file:

Save transcript in log file? (1 = yes):

If the user enters 1, the software prompts for the filename (this file will be stored in the directory specified at the initiation of the program):

Enter the name of the output file:

The software then proceeds to load the data file, determines the weight of the variables and parameters (if any), and prompts the user for the desired rank of the density,  $\rho$ .

```
*****
WELCOME TO THE MATHEMATICA PROGRAM
by UNAL GOKTAS, PAUL ADAMS, and WILLY HEREMAN
FOR THE COMPUTATION OF CONSERVED DENSITIES
WITH OR WITHOUT TRANSCENDENTAL NONLINEARITIES
Version 4.0 released on January 17, 2003
Copyright 2003
*****
```

Working with the data file for the Sine-Gordon Equation.

.

Enter the rank of rho:

After specifying the rank, typically no more action is required on the part of the user. The software will proceed with the computation of the density and flux and conclude with:

```
Testing final solution.
rho_t + J_x = 0, success!
```

```
***** SUMMARY *****
Total session time used in the current session is 3 seconds.
```

```
To see the density type: rho[x,t], or use pdeform[rho[x,t]]
To see the density (rho) sorted by coefficients, type 'sortedrho'
```

```
To see the flux type: J[x,t], or pdeform[J[x,t]]
To see the flux (J) sorted by coefficients, type 'sortedJ'
```

The first line shown here confirms that the  $(\rho, J)$  pair is valid. Otherwise, the software would report an error. Consult the section below on **Limitations** before taking further action if an error message appears.

Finally, the software reports the total time of the session (not just time of computation) and gives instructions for displaying and working with the result.

## Viewing the results

As stated, the density is stored as `rho[x,t]` and the associated flux as `J[x,t]` in standard *Mathematica* syntax. The user may perform mathematical operations on them, apply rules to them (e.g., to set constants to certain values), etc.

The user may also view a version of the density or flux, sorted according to the free constants and/or functions, and stored as lists under the names `sortedrho` and `sortedJ`. These sorted versions show the components of the densities and fluxes. Because the constants and functions are free, each may be individually isolated (e.g., set  $c_1$  to 1 and set all other constants to 0). Take note, however, that for systems that produce free functions  $f_i(u_k)$  in the conservation laws (see results for SG system, Table 6.3, to isolate the conservation laws one must collect all terms that involve the free function (e.g.,  $f_1(u_k)$ ) and all its derivatives (e.g.,  $f_1'(u_k)$ ,  $f_1''(u_k)$ , etc.).

## Testing user-defined systems

### Creating a user-defined data-file

First, refer to Appendix A for examples of pre-generated data-files. Here, the required lines of the data-files are explained briefly. For the SG system the data file has the following two lines:

```
eq[1][x,t] = D[u[1][x,t],t] - u[2][x,t];
eq[2][x,t] =
    D[u[2][x,t],t] - D[u[1][x,t],{x,2}] - alpha*Sin[u[1][x,t]];
```

corresponding to the system:

$$u_t = v,$$

$$v_t = u_{2x} + \alpha \sin(u).$$

To enter, collect terms on one side of the equation, then set

`eq[n][x,t]` equal to each expression, iterating 'n' with each line. Note that `u[1][x,t]` is used for the function  $u$  (dependent on  $x$  and  $t$ ) and `u[2][x,t]` is used for the function  $v(x,t)$ . Subsequent dependent variables should be added with similar notation in the data file.

This specifies the number of equations in the system:

```
noeqs = 2;
```

Though not necessary for calculations, a short name needs to be specified (the quotes are mandatory):

```
name = "Sine-Gordon System";
```

Next, the parameters are specified. The list `parameters` is for non-weighted parameters. If none exist, specify an empty list as shown in the example above. The list `weightpars` is for weighted parameters. The software will determine the weight of these parameters and also use them in constructing the density,  $\rho$ .

```
parameters = {};
```

```
weightpars = {alpha};
```

The software allows the user to specify a form for the density. This could be used to test an expression of  $\rho$  found in literature. Typically, the software is run without a form of  $\rho$  specified. If running in this mode, use the line:

```
formrho[x,t] = {};
```

If specifying the form of  $\rho$ , one would alter the line similar to these cases (note the braces; they are essential):

```
formrho[x,t] = { c[1]*u[1][x,t]^3 + c[2]*D[u[1][x,t],x]^2 };
```

if the PDE system has mixed derivatives or no transcendental nonlinearities, or

```
formrho[x,t] =
    {h[1][u[1][x,t]]*alpha + h[2][u[1][x,t]]*D[u[1][x,t],x]^2 };
```

if the PDE has transcendental nonlinearities and no mixed-derivatives. The constant coefficients must be denoted by  $c[i]$ , functional coefficients by  $h[i][u[k][x,t]]$ , where  $u[k][x,t]$  is the argument of the transcendental function.

Any additional comments or information may be included in the data file, as long as it is enclosed by (\* and \*).

### Importing the user-defined data file

The Menu Interface, as displayed previously, gives the options:

```
*** MENU INTERFACE *** (page: 1)
```

```
-----
.
.
nn) Next Page
tt) Your System
qq) Exit the Program
-----
```

Here, the user enters `tt` instead of a number when prompted with

```
ENTER YOUR CHOICE:
```

The user will then be asked

Is your file is ready? (1 = yes):

The only acceptable answer here is 1 (any other response will exit the program, with a warning to Prepare your data file first, then start over), after which the user will be asked to

Enter the name of your data file:

If for any reason the data file cannot be opened, the computations will immediately be aborted. Otherwise, the software will continue exactly as though the system was selected from those in the top menu.

### Other possible prompts

If there is too much freedom in the determination of the weights of the variables and parameters, the software may prompt the user to:

Enter your values for the free weights,

followed by instructions on how to do so. Alternatively, the software will provide choices for the assignment these weights.

### Limitations

- Fractional weights and a zero weight (on one dependent variable,  $u_k$ ) are allowed, and  $w \left( \frac{\partial}{\partial t} \right)$  can be negative. Other dependent variables and parameters must have strictly positive weights, as must  $\frac{\partial}{\partial x}$ .
- There is no limit to the value given for rank of the density,  $\rho$ . For high ranks, the speed and memory of the computer will be a limiting factor. For some PDE

systems, computations of densities of rank 8 required seven hours on the latest hardware (1.8 GHz processor, 512 MB of DDR memory).

- The only transcendental functions allowed are: sin, cos, sinh, cosh, exp.
- For PDE systems with transcendental nonlinearities, no unweighted parameters are allowed.
- Only one dependent variable may be the argument of a transcendental function. A variable with partial derivative (e.g.,  $u_x$ ) may not be the argument of the function, and ‘nesting’ of functions (e.g.,  $\exp(\sin(u))$ ) is not allowed.
- Only  $x$  and  $t$  are allowed as independent variables, and PDEs should not depend on  $x$  and  $t$  explicitly.
- There is no limit to the number of dependent variables or number of equations in the system.

### Obtaining the software

The software (TransPDEDensityFlux.M) and all related data files (D\_sineNP.M, etc.) are available via anonymous FTP from mines.edu. The login name is *anonymous* and password is your name or email address. The files are in the directory `pub/papers/math_cs_dept/software/TransPDEDensityFlux`.

The software is also available from the ‘Scientific Software’ section of Hereman’s home page with URL:

[http://www.mines.edu/fs\\_home/whereman](http://www.mines.edu/fs_home/whereman).

If there is any difficulty obtaining the files or using the software, users may contact the authors via email to `pjadams@idcomm.com` or `whereman@mines.edu`.

## Chapter 6

### RESULTS

In this chapter, the most notable systems with transcendental nonlinearities will be discussed. Where applicable and available, the physical or mathematical context of the given system will be discussed. For each equation or system, a table with the first few conservation laws found will be included. For brevity, long results of higher ranks are omitted. The computations were carried out with the software package **TransPDEDensityFlux.M**.

#### 6.1 Sine-Gordon Systems

The prototypical example for investigations of systems of nonlinear PDEs that include transcendental functions, the sine-Gordon equation derived its name as a pun on Klein-Gordon, after the class of equations named after its discoverers, Oskar Klein and Felix Gordon. Indeed, Klein and Gordon derived a relativistic equation for a charged particle in an electromagnetic field, using the recently discovered ideas of quantum theory. Their *Klein-Gordon equation* for the special case of a free particle in three dimensions reduces to:

$$\frac{1}{c^2} \frac{\partial^2 \psi}{\partial t^2} - \nabla^2 \psi + \left( \frac{mc}{\hbar} \right)^2 \psi = 0.$$

The mathematical generalization of this equation was later determined and termed the *nonlinear Klein-Gordon equation*:

$$\frac{1}{c^2} \frac{\partial^2 \psi}{\partial t^2} - \nabla^2 \psi + V'(\psi) = 0,$$

where  $V'$  is a nonlinear function. Restricted to one spatial dimension and setting  $V'(\psi) = \sin \psi$  we get the sine-Gordon equation in ‘laboratory’ coordinates [10]:

$$\frac{1}{c^2} \psi_{tt} - \psi_{xx} + \sin \psi = 0. \quad (6.1)$$

To arrive at the ‘characteristic’ or ‘light cone’ coordinates [4] we apply the coordinate transformations  $T = \frac{1}{2}(x - ct)$  and  $X = \frac{1}{2}(x + ct)$  to (6.1):

$$U_{XT} = \sin U, \quad (6.2)$$

where  $U(X(x, t), T(x, t)) = \psi(x, t)$ .

The Sine-Gordon arises as a direct model of physical phenomena such as the propagation of a dislocation in a crystal whose periodicity is represented by  $\sin \psi$ , as discovered by Frenkel and Kontorova in 1939 [5, 10]. As another example of the experimental observation of soliton interactions, it was also determined that the equation governs the propagation of magnetic flux in a long Josephson-junction transmission line, where  $\sin \psi$  is the Josephson current across an insulator between two superconductors and the voltage is proportional to  $\psi$  [5, 7]. In 1979, Gibbon, James, and Moroz showed that the SG equation governs the modulation of a weakly unstable baroclinic wave packet in a two-layer fluid, and therefore similar wave packets in a moving medium [10]. Other areas where the SG equation appears is in the theory

of pseudospherical surfaces (that is, surfaces of constant negative curvature), in dislocation theory, model field theories, superconductivity, and in mechanical models of nonlinear wave propagation [23]. A mechanical model of the SG equation can be constructed as a type of ‘analog computer’ of wave processes [5].

The SG equation is the first of the standard forms of the nonlinear Klein-Gordon equation [1, 3].

### 6.1.1 Single-Equation Case, in ‘Characteristic’ Coordinates

We calculated conservation laws for  $u_{xt} = \sin(u)$ , without the inclusion of the weighted parameter,  $\alpha$ , and tabulated the results in Table 6.1. In this system, note that  $w\left(\frac{\partial}{\partial t}\right) = -1$  in order to equalize the rank of each monomial in the equation. Because the scaling of  $x$  and  $t$  are not equal here,  $\text{rank}(\rho) \neq \text{rank}(J)$ . The ‘Rank’ here (in the first column of the table) refers to the rank of the density. Indeed, for all tabulated results in this chapter, the ‘Rank’ refers to the rank of the conserved density ( $\rho$ ).

### 6.1.2 Single-Equation Case, with Weighted Parameter

Here we consider  $u_{xt} = \alpha \sin(u)$ , where the weighted parameter,  $\alpha$ , is included to force  $w\left(\frac{\partial}{\partial t}\right)$  to be positive. The results are not surprising; in fact, substitution of  $\alpha = 1$  in the fluxes in Table 6.2 will produce the same results as in Table 6.1. These results are included merely as a demonstration of the software for the SG equation with a parameter ( $\alpha$ ) with weight.

Table 6.1. Results for the sine-Gordon equation without weighted parameter

$u_{xt} = \sin(u)$		
Scaling symmetry: $\{x, t, u\} = \{\lambda^1 x, \lambda^{-1} t, \lambda^0 u\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$2 \cos(u)$
4	$u_x^4 - 4 u_{2x}^2$	$4 \cos(u) u_x^2$
6	$u_x^6 - 20 u_x^2 u_{2x}^2 + 8 u_{3x}^2$	$6 \cos(u) u_x^4 + 16 \sin(u) u_x^2 u_{2x} - 8 \cos(u) u_{2x}^2$
8	$5 u_x^8 - 280 u_x^4 u_{2x}^2 -$ $112 u_{2x}^4 + 224 u_x^2 u_{3x}^2 -$ $64 u_{4x}^2$	$40 \cos(u) u_x^6 + 320 \sin(u) u_x^4 u_{2x} +$ $160 \cos(u) u_x^2 u_{2x}^2 + 128 \sin(u) u_{2x}^3 -$ $128 \cos(u) u_x^3 u_{3x} -$ $384 \sin(u) u_x u_{2x} u_{3x} + 64 \cos(u) u_{3x}^2$

### 6.1.3 System Case, in ‘Laboratory’ Coordinates

To arrive at a system from the single-equation in laboratory coordinates,  $u_{2x} - u_{2t} = \sin(u)$ , we introduce a new dependent variable,  $v$ . Setting  $v = u_t$  allows us to have a system of equations with only first order derivatives in time on any given dependent variable ( $u_t = v$  and  $v_t = \alpha \sin(u) + u_{2x}$ ), as required by the software. The results for the sine-Gordon system are given in Table 6.3. Here, the weighted parameter,  $\alpha$ , is required, else no scaling invariance (other than the trivial) is possible. To make the system coincide with (6.2), we should set  $\alpha = -1$  after processing. Again, we leave the parameter unchanged in the results, to illustrate its distribution during processing.

In these results, note the results for Ranks 1 and 6, which involve not only the free constants (seen in the results for the other ranks) but free functions in  $u$ . When separating the conservation laws for other ranks, one constant can be set equal to 1

Table 6.2. Results for the sine-Gordon equation with weighted parameter

$u_{xt} = \alpha \sin(u)$		
Scaling symmetry: $\{x, t, u, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$2 \alpha \cos(u)$
4	$u_x^4 - 4 u_{2x}^2$	$4 \alpha \cos(u) u_x^2$
6	$u_x^6 - 20 u_x^2 u_{2x}^2 + 8 u_{3x}^2$	$\alpha [6 \cos(u) u_x^4 + 16 \sin(u) u_x^2 u_{2x} - 8 \cos(u) u_{2x}^2]$
8	$5 u_x^8 - 280 u_x^4 u_{2x}^2 - 112 u_{2x}^4 + 224 u_x^2 u_{3x}^2 - 64 u_{4x}^2$	$\alpha [40 \cos(u) u_x^6 + 320 \sin(u) u_x^4 u_{2x} + 160 \cos(u) u_x^2 u_{2x}^2 + 128 \sin(u) u_{2x}^3 - 128 \cos(u) u_x^3 u_{3x} - 384 \sin(u) u_x u_{2x} u_{3x} + 64 \cos(u) u_{3x}^2]$

while the others are set equal to 0. To separate the conservation laws for rank 6, one equation can be set equal to 0 and the other to any function of  $u$  (without time- or space-derivatives).

The conservation law for rank 6 for the SG system required 158 seconds on a new computer. To find the density, the software solved a system of 217 linear ODEs and then ‘integrated’ (see Section 4.1) an expression with 130 terms to find the flux. For comparison, the previous conservation law, of rank 4, required only 9 seconds. The next conservation law, of rank 8, required several hours on the same computer. This demonstrates the exponential increase in difficulty in finding higher conservation laws.

Table 6.3. Results for the sine-Gordon system

$u_t = v$ $v_t = \alpha \sin(u) + u_{2x}$ Scaling symmetry: $\{x, t, u, v, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-1} v, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
1	$f_1(u) u_x$	$-f_1(u) v$
2	$c_1 [2\alpha \cos(u) + v^2 + u_x^2] +$ $c_2 [2v u_x]$	$c_1 [-2v u_x] +$ $c_2 [2\alpha \cos(u) - v^2 - u_x^2]$
4	$c_1 [2\alpha^2 \cos^2(u) - 2\alpha^2 \sin^2(u) +$ $4\alpha \cos(u) v^2 + v^4 +$ $20\alpha \cos(u) u_x^2 + 6v^2 u_x^2 +$ $u_x^4 - 16v_x^2 - 16u_{2x}^2] +$ $c_2 [24\alpha \cos(u) v u_x + 4v^3 u_x +$ $4v u_x^3 - 32v_x u_{2x}]$	$c_1 [-8\alpha \cos(u) v u_x - 4v^3 u_x -$ $4v u_x^3 + 32v_x u_{2x}] +$  $c_2 [6\alpha^2 \cos^2(u) - 6\alpha^2 \sin^2(u) -$ $12\alpha \cos(u) v^2 - v^4 + 4\alpha \cos(u) u_x^2 -$ $6v^2 u_x^2 - u_x^4 + 16v_x^2 + 16u_{2x}^2]$
6	$f_1(u) [v_x^2 u_{2x} + 2u_x v_x v_{2x}] +$ $f_1'(u) [u_x^2 v_x^2] +$  $f_2(u) [v_x u_{2x}^2 + 2v u_{2x} u_{3x}] +$ $f_2'(u) [v u_x u_{2x}^2]$	$f_1(u) [-2\alpha \cos(u) u_x^2 v_x - v_x^3 -$ $2u_x v_x u_{3x}] +$ $f_1'(u) [-v u_x v_x^2] +$ $f_2(u) [-\alpha \sin(u) u_{2x}^2 - u_{2x}^3 -$ $2v u_{2x} v_{2x}] +$ $f_2'(u) [-v^2 u_{2x}^2]$
8	(110 terms)	(218 terms)

## 6.2 Sinh-Gordon Systems

The sinh-Gordon equation,  $u_{xt} = \sinh(u)$ , arises as a special case of the Toda lattice equation, another well-known soliton equation in one space and one time dimension [1], which can be used to model the interaction of neighboring particles of equal mass in a lattice formation within a crystal. The sinh-Gordon equation

describes generic properties of string dynamics for strings and multi-strings in constant curvature spacetimes [24]. Another application of the sinh-Gordon equation is in the field of thermodynamics, where it can be used to exactly calculate partition and correlation functions, and thus support the high-resolution Langevin simulations performed on such systems [20].

The sinh-Gordon equation is the second of the four standard forms of the nonlinear Klein-Gordon equation [1, 3]. While some publications do not list them explicitly [1], their format is nearly identical, and other (more recent) papers name them specifically [3].

Here, we present the conservation laws for the various formats of the equation. Note the results are similar, though not identical, to those for their sine-Gordon ‘cousins’.

### 6.2.1 Single-Equation Case, in ‘Characteristic’ Coordinates

Table 6.4 lists the results for the sinh-Gordon equation without weighted parameter,  $u_{xt} = \sinh(u)$ . The calculations for the equations in ‘characteristic’ coordinates (with or without parameter) are much simpler than those for their ‘laboratory’ coordinate counterparts. Here, for the rank 10 conservation law, the code had only to solve a system of 29 equations, requiring only 11 seconds to compute the conserved density and flux. Even the conserved density of ranks 12 and 14 (and their fluxes) required only 21 and 59 seconds (respectively), though the results are too long to include here. One explanation for this reduced time is the absence of functional coefficients ( $h_i(u_k)$ ).

Because the scaling of  $x$  and  $t$  are not equal here,  $\text{rank}(\rho) \neq \text{rank}(J)$ . The ‘Rank’ here (in the first column of the table) refers to the rank of the density.

Table 6.4. Results for the sinh-Gordon equation without weighted parameter

$u_{xt} = \sinh(u)$ Scaling symmetry: $\{x, t, u\} = \{\lambda^1 x, \lambda^{-1} t, \lambda^0 u\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$-2 \cosh(u)$
4	$u_x^4 + 4 u_{2x}^2$	$-4 \cosh(u) u_x^2$
6	$u_x^6 + 20 u_x^2 u_{2x}^2 + 8 u_{3x}^2$	$-6 \cosh(u) u_x^4 - 16 \sinh(u) u_x^2 u_{2x} - 8 \cosh(u) u_{2x}^2$
8	$5 u_x^8 + 280 u_x^4 u_{2x}^2 - 112 u_{2x}^4 + 224 u_x^2 u_{3x}^2 + 64 u_{4x}^2$	$-40 \cosh(u) u_x^6 - 320 \sinh(u) u_x^4 u_{2x} + 160 \cosh(u) u_x^2 u_{2x}^2 + 128 \sinh(u) u_{2x}^3 - 128 \cosh(u) u_x^3 u_{3x} - 384 \sinh(u) u_x u_{2x} u_{3x} - 64 \cosh(u) u_{3x}^2$
10	$7 u_x^{10} + 840 u_x^6 u_{2x}^2 - 2128 u_x^2 u_{2x}^4 + 1008 u_x^4 u_{3x}^2 - 3264 u_{2x}^2 u_{3x}^2 - 1280 u_x u_{3x}^3 + 576 u_x^2 u_{4x}^2 + 128 u_{5x}^2$	$-70 \cosh(u) u_x^8 - 1120 \sinh(u) u_x^6 u_{2x} + 1680 \cosh(u) u_x^4 u_{2x}^2 + 1792 \sinh(u) u_x^2 u_{2x}^3 + 672 \cosh(u) u_{2x}^4 - 896 \cosh(u) u_x^5 u_{3x} - 896 \sinh(u) u_x^3 u_{2x} u_{3x} + 3840 \cosh(u) u_x u_{2x}^2 u_{3x} + 704 \cosh(u) u_x^2 u_{3x}^2 + 1280 \sinh(u) u_{2x} u_{3x}^2 - 256 \sinh(u) u_x^4 u_{4x} - 1536 \cosh(u) u_x^2 u_{2x} u_{4x} - 768 \sinh(u) u_{2x}^2 u_{4x} - 1024 \sinh(u) u_x u_{3x} u_{4x} - 128 \cosh(u) u_{4x}^2$

### 6.2.2 Single-Equation Case, with Weighted Parameter

Results for the sinh-Gordon equation with weighted parameter,  $u_{xt} = \alpha \sin(u)$ , are given in Table 6.5. Because the system (and results) are nearly identical to those in Table 6.4, the computation times and length of results are the same.

Table 6.5. Results for the sinh-Gordon equation with weighted parameter

$u_{xt} = \alpha \sinh(u)$ Scaling symmetry: $\{x, t, u, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$-2 \alpha \cosh(u)$
4	$u_x^4 + 4 u_{2x}^2$	$-4 \alpha \cosh(u) u_x^2$
6	$u_x^6 + 20 u_x^2 u_{2x}^2 + 8 u_{3x}^2$	$\alpha [-6 \cosh(u) u_x^4 - 16 \sinh(u) u_x^2 u_{2x} - 8 \cosh(u) u_{2x}^2]$
8	$5 u_x^8 + 280 u_x^4 u_{2x}^2 - 112 u_{2x}^4 + 224 u_x^2 u_{3x}^2 + 64 u_{4x}^2$	$\alpha [-40 \cosh(u) u_x^6 - 320 \sinh(u) u_x^4 u_{2x} + 160 \cosh(u) u_x^2 u_{2x}^2 + 128 \sinh(u) u_{2x}^3 - 128 \cosh(u) u_x^3 u_{3x} - 384 \sinh(u) u_x u_{2x} u_{3x} - 64 \cosh(u) u_{3x}^2]$
10	$7 u_x^{10} + 840 u_x^6 u_{2x}^2 - 2128 u_x^2 u_{2x}^4 + 1008 u_x^4 u_{3x}^2 - 3264 u_{2x}^2 u_{3x}^2 - 1280 u_x u_{3x}^3 + 576 u_x^2 u_{4x}^2 + 128 u_{5x}^2$	$\alpha [-70 \cosh(u) u_x^8 - 1120 \sinh(u) u_x^6 u_{2x} + 1680 \cosh(u) u_x^4 u_{2x}^2 + 1792 \sinh(u) u_x^2 u_{2x}^3 + 672 \cosh(u) u_{2x}^4 - 896 \cosh(u) u_x^5 u_{3x} - 896 \sinh(u) u_x^3 u_{2x} u_{3x} + 3840 \cosh(u) u_x u_{2x}^2 u_{3x} + 704 \cosh(u) u_x^2 u_{3x}^2 + 1280 \sinh(u) u_{2x} u_{3x}^2 - 256 \sinh(u) u_x^4 u_{4x} - 1536 \cosh(u) u_x^2 u_{2x} u_{4x} - 768 \sinh(u) u_{2x}^2 u_{4x} - 1024 \sinh(u) u_x u_{3x} u_{4x} - 128 \cosh(u) u_{4x}^2]$

### 6.2.3 System Case, in ‘Laboratory’ Coordinates

The results for the sinh-Gordon system ( $u_t = v$  and  $v_t = \alpha \sinh(u) + u_{2x}$ ) are found in Table 6.6. Here again we see the arbitrary functions ( $f_i(u)$ ) appear in the results for ranks 1 and 6, similar to the sine-Gordon system (Table 6.3). The

computational difficulty (and thus time required) for the sinh-Gordon system is very similar to that of the sine-Gordon system.

Table 6.6. Results for the sinh-Gordon system

$u_t = v$ $v_t = \alpha \sinh(u) + u_{2x}$ Scaling symmetry: $\{x, t, u, v, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-1} v, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
1	$f_1(u) u_x$	$-f_1(u) v$
2	$c_1 [2\alpha \cosh(u) - v^2 - u_x^2] +$ $c_2 [2v u_x]$	$c_1 [2v u_x] +$ $c_2 [-2\alpha \cosh(u) - v^2 - u_x^2]$
4	$c_1 [2\alpha^2 \cosh^2(u) + 2\alpha^2 \sinh^2(u) -$ $4\alpha \cosh(u) v^2 + v^4 -$ $20\alpha \cosh(u) u_x^2 + 6v^2 u_x^2 +$ $u_x^4 + 16v_x^2 + 16u_{2x}^2] +$ $c_2 [24\alpha \cosh(u) v u_x + 4v^3 u_x +$ $4v u_x^3 + 32v_x u_{2x}]$	$c_1 [8\alpha \cosh(u) v u_x - 4v^3 u_x -$ $4v u_x^3 - 32v_x u_{2x}] +$ $c_2 [6\alpha^2 \cosh^2(u) + 6\alpha^2 \sinh^2(u) +$ $12\alpha \cosh(u) v^2 - v^4 -$ $4\alpha \cosh(u) u_x^2 - 6v^2 u_x^2 -$ $u_x^4 - 16v_x^2 - 16u_{2x}^2]$
6	$f_1(u) [v_x^2 u_{2x} + 2u_x v_x v_{2x}] +$ $f_1'(u) [u_x^2 v_x^2] +$ $f_2(u) [v_x u_{2x}^2 + 2v u_{2x} u_{3x}] +$ $f_2'(u) [v u_x u_{2x}^2]$	$f_1(u) [-2\alpha \cosh(u) u_x^2 v_x - v_x^3 -$ $2u_x v_x u_{3x}] +$ $f_1'(u) [-v u_x v_x^2] +$ $f_2(u) [-\alpha \sinh(u) u_{2x}^2 - u_{2x}^3 -$ $2v u_{2x} v_{2x}] +$ $f_2'(u) [-v^2 u_{2x}^2]$

### 6.3 Liouville Systems

The Liouville equation is the third of four standard forms of the nonlinear Klein-Gordon equation [1, 3]. Certain forms of the Liouville equation play an important role in modern field theory, namely in the theory of strings, where the quantum Liouville field appears as a conformal anomaly [21].

Note is the length of the density and flux reported in the text. These results are ‘raw’, without the lower-ranked results removed. In some cases, previous (lower-ranked) conserved densities and fluxes were found recurring in higher-ranked conservation laws. For the results given here, lower-ranked conserved densities and fluxes are removed from the higher-ranked results. This is done for simplicity, since it is given than scalar multiples of conservation laws can be combined to produce ‘new’ conservation laws.

### 6.3.1 Single-Equation Case, in 'Characteristic' Coordinates

For the 'light-cone' or 'characteristic' coordinate form of the Liouville equation,  $u_{xt} = e^u$ , results are in Tables 6.7 and 6.8. Because the scaling of  $x$  and  $t$  are not equal here,  $\text{rank}(\rho) \neq \text{rank}(J)$ . The 'Rank' here (in the first column of the table) refers to the rank of the density.

Table 6.7. Results for the Liouville equation without parameter

$u_{xt} = e^u$		
Scaling symmetry: $\{x, t, u\} = \{\lambda^1 x, \lambda^{-1} t, \lambda^0 u\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$-2 e^u$
4	$u_x^4 + 4 u_{2x}^2$	$-4 e^u u_x^2$
6	$c_1 [u_x^6 + 12 u_x^2 u_{2x}^2 - 8 u_{2x}^3] +$ $c_2 [u_x^6 + 20 u_x^2 u_{2x}^2 + 8 u_{3x}^2]$	$c_1 [-6 e^u u_x^4] +$ $c_2 [-6 e^u u_x^4 - 16 e^u u_x^2 u_{2x} -$ $8 e^u u_{2x}^2]$
8	$c_1 [u_x^8 + 24 u_x^4 u_{2x}^2 -$ $32 u_x^2 u_{2x}^3 + 16 u_{2x}^4] +$ $c_2 [u_x^8 + 36 u_x^4 u_{2x}^2 -$ $20 u_x^2 u_{2x}^3 + 12 u_x^2 u_{3x}^2 -$ $24 u_{2x} u_{3x}^2] +$ $c_3 [3 u_x^8 + 112 u_x^4 u_{2x}^2 -$ $56 u_x^2 u_{2x}^3 + 56 u_x^2 u_{3x}^2 +$ $16 u_{4x}^2]$	$c_1 [-8 e^u u_x^6] +$ $c_2 [-8 e^u u_x^6 - 24 e^u u_x^4 u_{2x} +$ $12 e^u u_x^2 u_{2x}^2 + 16 e^u u_{2x}^3] +$ $c_3 [-24 e^u u_x^6 - 80 e^u u_x^4 u_{2x} +$ $40 e^u u_x^2 u_{2x}^2 + 32 e^u u_{2x}^3 -$ $32 e^u u_x^3 u_{3x} - 96 e^u u_x u_{2x} u_{3x} -$ $16 e^u u_{3x}^2]$

Table 6.8. Results for the Liouville equation without parameter, continued

$u_{xt} = e^u$		
Scaling symmetry: $\{x, t, u\} = \{\lambda^1 x, \lambda^{-1} t, \lambda^0 u\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
10	$c_1 [u_x^{10} + 40 u_x^6 u_{2x}^2 -$ $80 u_x^4 u_{2x}^3 + 80 u_x^2 u_{2x}^4 -$ $32 u_{2x}^5] +$ $c_2 [u_x^{10} + 56 u_x^6 u_{2x}^2 -$ $64 u_x^4 u_{2x}^3 + 16 u_x^2 u_{2x}^4 +$ $16 u_x^4 u_{3x}^2 - 64 u_x^2 u_{2x} u_{3x}^2 +$ $64 u_{2x}^2 u_{3x}^2] +$ $c_3 [5 u_x^{10} + 312 u_x^6 u_{2x}^2 -$ $288 u_x^4 u_{2x}^3 - 80 u_x^2 u_{2x}^4 +$ $144 u_x^4 u_{3x}^2 - 64 u_x u_{3x}^3 -$ $288 u_x^2 u_{2x} u_{3x}^2 +$ $32 u_x^2 u_{4x}^2 - 64 u_{2x} u_{4x}^2] +$ $c_4 [29 u_x^{10} + 1848 u_x^6 u_{2x}^2 -$ $1632 u_x^4 u_{2x}^3 - 656 u_x^2 u_{2x}^4 +$ $912 u_x^4 u_{3x}^2 - 640 u_x u_{3x}^3 -$ $1632 u_x^2 u_{2x} u_{3x}^2 +$ $288 u_x^2 u_{4x}^2 + 64 u_{5x}^2]$	$c_1 [-10 e^u u_x^8] +$ $c_2 [-10 e^u u_x^8 - 32 e^u u_x^6 u_{2x} +$ $48 e^u u_x^4 u_{2x}^2 - 32 e^u u_{2x}^4] +$ $c_3 [-50 e^u u_x^8 - 224 e^u u_x^6 u_{2x} +$ $336 e^u u_x^4 u_{2x}^2 + 128 e^u u_x^2 u_{2x}^3 -$ $96 e^u u_{2x}^4 - 64 e^u u_x^5 u_{3x} -$ $64 e^u u_x^3 u_{2x} u_{3x} +$ $384 e^u u_x u_{2x}^2 u_{3x} -$ $32 e^u u_x^2 u_{3x}^2 + 64 e^u u_{2x} u_{3x}^2] +$ $c_4 [-290 e^u u_x^8 - 1376 e^u u_x^6 u_{2x} +$ $2064 e^u u_x^4 u_{2x}^2 + 896 e^u u_x^2 u_{2x}^3 -$ $480 e^u u_{2x}^4 - 448 e^u u_x^5 u_{3x} -$ $448 e^u u_x^3 u_{2x} u_{3x} + 352 e^u u_x^2 u_{3x}^2 +$ $1920 e^u u_x u_{2x}^2 u_{3x} +$ $640 e^u u_{2x} u_{3x}^2 - 128 e^u u_x^4 u_{4x} -$ $768 e^u u_x^2 u_{2x} u_{4x} - 384 e^u u_{2x}^2 u_{4x} -$ $512 e^u u_x u_{3x} u_{4x} - 64 e^u u_{4x}^2]$

### 6.3.2 Single-Equation Case, with Weighted Parameter

Adding the weighted parameter,  $\alpha$ , makes the equation  $u_{xt} = \alpha e^u$ . The conservation laws for this equation are in Tables 6.9 and 6.10.

Similar to previous 'characteristic' coordinate forms, the results here were ob-

tained quickly by the software. To solve the 48 equation and find the 54-term density of rank 10, along with its 49-term associated flux, required only 35 seconds. Again, the absence of the coefficient functions ( $h_i(u_k)$ ) greatly reduced the computation time.

Table 6.9. Results for the Liouville equation with parameter

$u_{xt} = \alpha e^u$ Scaling symmetry: $\{x, t, u, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$-2\alpha e^u$
4	$u_x^4 + 4u_{2x}^2$	$-4\alpha e^u u_x^2$
6	$c_1 [u_x^6 + 12u_x^2 u_{2x}^2 - 8u_{2x}^3] +$ $c_2 [u_x^6 + 20u_x^2 u_{2x}^2 + 8u_{3x}^2]$	$c_1 \alpha [-6e^u u_x^4] +$ $c_2 \alpha [-6e^u u_x^4 - 16e^u u_x^2 u_{2x} -$ $8e^u u_{2x}^2]$
8	$c_1 [u_x^8 + 24u_x^4 u_{2x}^2 -$ $32u_x^2 u_{2x}^3 + 16u_{2x}^4] +$ $c_2 [u_x^8 + 36u_x^4 u_{2x}^2 -$ $20u_x^2 u_{2x}^3 + 12u_x^2 u_{3x}^2 -$ $24u_{2x} u_{3x}^2] +$ $c_3 [3u_x^8 + 112u_x^4 u_{2x}^2 -$ $56u_x^2 u_{2x}^3 + 56u_x^2 u_{3x}^2 +$ $16u_{4x}^2]$	$c_1 \alpha [-8e^u u_x^6] +$ $c_2 \alpha [-8e^u u_x^6 - 24e^u u_x^4 u_{2x} +$ $12e^u u_x^2 u_{2x}^2 + 16e^u u_{2x}^3] +$ $c_3 \alpha [-24e^u u_x^6 - 80e^u u_x^4 u_{2x} +$ $40e^u u_x^2 u_{2x}^2 + 32e^u u_{2x}^3 -$ $32e^u u_x^3 u_{3x} - 96e^u u_x u_{2x} u_{3x} -$ $16e^u u_{3x}^2]$

Table 6.10. Results for the Liouville equation with parameter, continued

$u_{xt} = \alpha e^u$		
Scaling symmetry: $\{x, t, u, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
10	$c_1 [u_x^{10} + 40 u_x^6 u_{2x}^2 -$ $80 u_x^4 u_{2x}^3 + 80 u_x^2 u_{2x}^4 -$ $32 u_{2x}^5] +$ $c_2 [u_x^{10} + 56 u_x^6 u_{2x}^2 -$ $64 u_x^4 u_{2x}^3 + 16 u_x^2 u_{2x}^4 +$ $16 u_x^4 u_{3x}^2 - 64 u_x^2 u_{2x} u_{3x}^2 +$ $64 u_{2x}^2 u_{3x}^2] +$ $c_3 [5 u_x^{10} + 312 u_x^6 u_{2x}^2 -$ $288 u_x^4 u_{2x}^3 - 80 u_x^2 u_{2x}^4 +$ $144 u_x^4 u_{3x}^2 - 64 u_x u_{3x}^3 -$ $288 u_x^2 u_{2x} u_{3x}^2 +$ $32 u_x^2 u_{4x}^2 - 64 u_{2x} u_{4x}^2] +$ $c_4 [29 u_x^{10} + 1848 u_x^6 u_{2x}^2 -$ $1632 u_x^4 u_{2x}^3 - 656 u_x^2 u_{2x}^4 +$ $912 u_x^4 u_{3x}^2 - 640 u_x u_{3x}^3 -$ $1632 u_x^2 u_{2x} u_{3x}^2 +$ $288 u_x^2 u_{4x}^2 + 64 u_{5x}^2]$	$c_1 \alpha [-10 e^u u_x^8] +$ $c_2 \alpha [-10 e^u u_x^8 - 32 e^u u_x^6 u_{2x} +$ $48 e^u u_x^4 u_{2x}^2 - 32 e^u u_{2x}^4] +$ $c_3 \alpha [-50 e^u u_x^8 - 224 e^u u_x^6 u_{2x} +$ $336 e^u u_x^4 u_{2x}^2 + 128 e^u u_x^2 u_{2x}^3 -$ $96 e^u u_{2x}^4 - 64 e^u u_x^5 u_{3x} -$ $64 e^u u_x^3 u_{2x} u_{3x} +$ $384 e^u u_x u_{2x}^2 u_{3x} -$ $32 e^u u_x^2 u_{3x}^2 + 64 e^u u_{2x} u_{3x}^2] +$ $c_4 \alpha [-290 e^u u_x^8 - 1376 e^u u_x^6 u_{2x} +$ $2064 e^u u_x^4 u_{2x}^2 + 896 e^u u_x^2 u_{2x}^3 -$ $480 e^u u_{2x}^4 - 448 e^u u_x^5 u_{3x} -$ $448 e^u u_x^3 u_{2x} u_{3x} + 352 e^u u_x^2 u_{3x}^2 +$ $1920 e^u u_x u_{2x}^2 u_{3x} +$ $640 e^u u_{2x} u_{3x}^2 - 128 e^u u_x^4 u_{4x} -$ $768 e^u u_x^2 u_{2x} u_{4x} - 384 e^u u_{2x}^2 u_{4x} -$ $512 e^u u_x u_{3x} u_{4x} - 64 e^u u_{4x}^2]$

### 6.3.3 System Case, in 'Laboratory' Coordinates

Again, changing the variables puts the equation into its 'laboratory' coordinate form,  $u_{2x} - u_{2t} = \alpha e^u$ , which we then convert into two equations, each with only one derivative in time:  $u_t = v$  and  $v_t = \alpha e^u + u_{2x}$ . The results for this system are found

in Table 6.11.

Computational complexity here was, as expected, similar to the previous 'laboratory coordinate' systems investigated. The 'raw' rank 6 density had 32 terms while its flux had 27 terms. The computations, including the solution of a system of 217 linear ODEs and the 'integration' (see Section 4.1) of a 170-term expression, required 148 seconds.

Table 6.11. Results for the Liouville system

$u_t = v$ $v_t = \alpha e^u + u_{2x}$ Scaling symmetry: $\{x, t, u, v, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-1} v, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
1	$f_1(u) u_x$	$-f_1(u) v$
2	$c_1 [2\alpha e^u - v^2 - u_x^2] +$ $c_2 [2v u_x]$	$c_1 [2v u_x] +$ $c_2 [-2\alpha e^u - v^2 - u_x^2]$
4	$4\alpha^2 e^{2u} - 4\alpha e^u v^2 + v^4 - 20\alpha e^u u_x^2 +$ $6v^2 u_x^2 + u_x^4 + 16v_x^2 + 16u_{2x}^2$	$8\alpha e^u v u_x - 4v^3 u_x -$ $4v u_x^3 - 32v_x u_{2x}$
6	$c_1 [\alpha^2 e^{2u} u_x^2 + 2\alpha e^u u_x^4 -$ $2\alpha e^u v u_x v_x + v^2 v_x^2 + u_x^2 v_x^2 +$ $4v u_x v_x u_{2x} + 6v_x^2 u_{2x} -$ $4\alpha e^u u_{2x}^2 + v^2 u_{2x}^2 + u_x^2 u_{2x}^2 +$ $2u_{2x}^3 + 4v_{2x}^2 + 4u_{3x}^2] +$ $f_1(u) [v_x^2 u_{2x} + 2u_x v_x v_{2x}] +$ $f_1'(u) [u_x^2 v_x^2] +$  $f_2(u) [v_x u_{2x}^2 + 2v u_{2x} u_{3x}] +$ $f_2'(u) [v u_x u_{2x}^2]$	$c_1 [-2\alpha e^u v u_x^3 - 8\alpha e^u u_x^2 v_x -$ $2v u_x v_x^2 - 2v_x^3 +$ $2\alpha e^u v u_x u_{2x} - 2v^2 v_x u_{2x} -$ $2u_x^2 v_x u_{2x} - 2v u_x u_{2x}^2 -$ $6v_x u_{2x}^2 - 8v_{2x} u_{3x}] +$ $f_1(u) [-2\alpha e^u u_x^2 v_x - v_x^3 -$ $2u_x v_x u_{3x}] +$ $f_1'(u) [-v u_x v_x^2] +$ $f_2(u) [-\alpha e^u u_{2x}^2 - u_{2x}^3 -$ $2v u_{2x} v_{2x}] +$ $f_2'(u) [-v^2 u_{2x}^2]$

## 6.4 Multiple Sine-Gordon Equations

Previous work on multiple sine-Gordon equations, which have application in the theory of short optical pulse propagation, revealed that they do not have an infinite number of conservation laws. In fact, they have essentially three conservation laws, only one of which is a polynomial conserved density, of rank 2. The conclusion is that multiple sine-Gordon equations are not soluble by present formulations of the inverse scattering method, despite numerical solutions which show soliton-like behavior [9].

Additionally, from the Painlevé ODE test, researchers have concluded that the only PDEs of the form

$$u_{xt} = f(u)$$

that might be completely integrable are equivalent to one of four standard forms (the sine-Gordon, sinh-Gordon, Liouville equation, or the double Liouville equations) [3]. The multiple sine-Gordon equations do not fit this format, and are therefore not completely integrable, as is supported by the existence of only one conservation law [1].

Our findings support these claims, as we could compute only one conservation law for each of the multiple sine-Gordon equations tested. Each density is of rank 2.

### 6.4.1 Double Sine-Gordon Equation

The first multiple sine-Gordon equation,  $u_{xt} = \sin(u) + \sin(2u)$ , gives the one conservation law in Table 6.12. Because the scaling of  $x$  and  $t$  are not equal here,  $\text{rank}(\rho) \neq \text{rank}(J)$ . The ‘Rank’ here (in the first column of the table) refers to the rank of the density.

Table 6.12. Results for the double sine-Gordon equation

$u_{xt} = \sin(u) + \sin(2u)$		
Scaling symmetry: $\{x, t, u\} = \{\lambda^1 x, \lambda^{-1} t, \lambda^0 u\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$2 \cos(u) + \cos(2u)$

#### 6.4.2 Another Multiple Sine-Gordon Equation

The other tested multiple sine-Gordon equation,  $u_{xt} = \sin(u) + \frac{1}{2} \sin\left(\frac{1}{2}u\right)$ , also gives just one conservation law, as seen in Table 6.13. Again, because the scaling on  $x$  and  $t$  here are not equal, the rank of  $\rho$  differs from that of  $J$ . As before, the 'Rank' here refers to the rank of the density.

Table 6.13. Results for another multiple sine-Gordon equation

$u_{xt} = \sin(u) + \frac{1}{2} \sin\left(\frac{1}{2}u\right)$		
Scaling symmetry: $\{x, t, u\} = \{\lambda^1 x, \lambda^{-1} t, \lambda^0 u\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$2 \cos\left(\frac{1}{2}u\right) + 2 \cos(u)$

## 6.5 Multiple Liouville Systems

The multiple Liouville equation,

$$u_{xt} = e^u - e^{-2u}, \quad (6.3)$$

is the fourth standard form of the nonlinear Klein-Gordon equation [1, 3].

One of the modern applications of these types of equations is in the field of “laser-induced vibrational predesorption of molecules physisorbed on insulating substrates.” Briefly, (6.3) is related to the investigation of the dynamics of energy flow of excited admolecules on insulating substrates [28]. Double-Liouville equations are also relevant in studies on the global properties of scalar-vacuum configurations in general relativity and similarly systems in some alternative theories of gravity [8].

Here, we present the conservation laws for three different cases that fall under the classification of multiple Liouville systems.

### 6.5.1 Tzetzzeica Equation without Weighted Parameter

The Tzetzzeica equation,  $u_{xt} = e^u - e^{-2u}$ , is found with either a plus or a minus sign before the  $e^{-2u}$  term [3] in the literature. The results for this form are in Table 6.14. Because the scaling of  $x$  and  $t$  are not equal here,  $\text{rank}(\rho) \neq \text{rank}(J)$ . The ‘Rank’ here (in the first column of the table) refers to the rank of the density.

The rank 8 conservation law given is calculated by the software in only 10 seconds. The rank 12 results (a 21-term density and 60-term flux) only required 39 seconds. This is similar to previous single-equation cases.

Table 6.14. Results for the Tzetzzeica equation without weighted parameter

$u_{xt} = e^u - e^{-2u}$		
Scaling symmetry: $\{x, t, u\} = \{\lambda^1 x, \lambda^{-1} t, \lambda^0 u\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$2 \cos(u)$
4	$u_x^4 - 4 u_{2x}^2$	$4 \cos(u) u_x^2$
6	$u_x^6 - 20 u_x^2 u_{2x}^2 + 8 u_{3x}^2$	$6 \cos(u) u_x^4 + 16 \sin(u) u_x^2 u_{2x} - 8 \cos(u) u_{2x}^2$
8	$5 u_x^8 - 280 u_x^4 u_{2x}^2 -$ $112 u_{2x}^4 + 224 u_x^2 u_{3x}^2 -$ $64 u_{4x}^2$	$40 \cos(u) u_x^6 + 320 \sin(u) u_x^4 u_{2x} +$ $160 \cos(u) u_x^2 u_{2x}^2 + 128 \sin(u) u_{2x}^3 -$ $128 \cos(u) u_x^3 u_{3x} - 384 \sin(u) u_x u_{2x} u_{3x} +$ $64 \cos(u) u_{3x}^2$

### 6.5.2 Tzetzzeica Equation with Weighted Parameter

Just as before, these results for  $u_{xt} = \alpha e^u - \alpha e^{-2u}$  are included as a demonstration of the software for a double-Liouville equation with weighted parameter,  $\alpha$ , which is added merely to force  $w \left( \frac{\partial}{\partial t} \right)$  to be positive. Setting this parameter  $\alpha = 1$  in the results for this equation (in Table 6.15) will produce the same results seen in Table 6.14.

### 6.5.3 Mikhailov System

The next two systems,

$$\begin{aligned}
 u_t &= v, \\
 v_t &= -\alpha e^u \pm \alpha e^{-2u} + u_{2x},
 \end{aligned} \tag{6.4}$$

Table 6.15. Results for the Tzetzzeica equation with weighted parameter

$u_{xt} = \alpha e^u - \alpha e^{-2u}$ Scaling symmetry: $\{x, t, u, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
2	$u_x^2$	$2 \alpha \cos(u)$
4	$u_x^4 - 4 u_{2x}^2$	$4 \alpha \cos(u) u_x^2$
6	$u_x^6 - 20 u_x^2 u_{2x}^2 + 8 u_{3x}^2$	$6 \alpha \cos(u) u_x^4 + 16 \alpha \sin(u) u_x^2 u_{2x} -$ $8 \alpha \cos(u) u_{2x}^2$
8	$5 u_x^8 - 280 u_x^4 u_{2x}^2 -$ $112 u_{2x}^4 + 224 u_x^2 u_{3x}^2 -$ $64 u_{4x}^2$	$40 \alpha \cos(u) u_x^6 + 320 \alpha \sin(u) u_x^4 u_{2x} +$ $160 \alpha \cos(u) u_x^2 u_{2x}^2 + 128 \alpha \sin(u) u_{2x}^3 -$ $128 \alpha \cos(u) u_x^3 u_{3x} -$ $384 \alpha \sin(u) u_x u_{2x} u_{3x} + 64 \alpha \cos(u) u_{3x}^2$

differ by only a sign, and correspond to the Tzetzzeica equations, put into ‘laboratory’ coordinates. Some authors refer to (6.4) as the Mikhailov system (results in Table 6.16) or double-Liouville system (results in Table 6.17) [11].

Though the results here for rank 6 are fairly short, they come as a result of 306 seconds of computations, after solving a 217-equation system of ODEs. Before removing the lower-ranked conservation laws, the density had 12 terms and the flux had 16 terms, though the expression for  $D_x J$  had 148 terms.

#### 6.5.4 Double Liouville System

The results for the double-Liouville system,  $u_t = v$  and  $v_t = -\alpha e^u + \alpha e^{-2u} + u_{2x}$  (found in Table 6.17), similar to those for the Mikhailov system, had a mix of conservation laws with free constants (see rank 2), free functions (see rank 6) and those that were linear combinations of lower-level conservation laws, and thus not

Table 6.16. Results for the Mikhailov system

$u_t = v$ $v_t = -\alpha e^u - \alpha e^{-2u} + u_{2x}$ Scaling symmetry: $\{x, t, u, v, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-1} v, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
1	$f_1(u) u_x$	$-f_1(u) v$
2	$c_1 [\alpha e^{-2u} - 2\alpha e^u - v^2 - u_x^2] +$ $c_2 [2v u_x]$	$c_1 [2v u_x] +$ $c_2 [-\alpha e^{-2u} + 2\alpha e^u - v^2 - u_x^2]$
4	(No new solution)	(No new solution)
6	$f_1(u) [v_x^2 u_{2x} + 2u_x v_x v_{2x}] +$ $f_1'(u) [u_x^2 v_x^2] +$  $f_2(u) [v_x u_{2x}^2 + 2v u_{2x} u_{3x}] +$ $f_2'(u) [v u_x u_{2x}^2]$	$f_1(u) [-4\alpha u_x^2 v_x e^{-2u} +$ $2\alpha e^u u_x^2 v_x - v_x^3 - 2u_x v_x u_{3x}] +$ $f_1'(u) [-v u_x v_x^2] +$ $f_2(u) [\alpha u_{2x}^2 e^{-2u} + \alpha e^u u_{2x}^2 -$ $u_{2x}^3 - 2v u_{2x} v_{2x}] +$ $f_2'(u) [-v^2 u_{2x}^2]$

re-listed (see rank 4).

Due to their extreme similarity, the Mikhailov system (see Table 6.16) and this system required similar amounts of computation-time for equivalently-ranked conservation laws. The rank 8 case, run for this system and not the Mikhailov system, required about seven hours to compute. Again, this is typical for system cases (with transcendental nonlinearities), due to the inclusion of coefficient functions ( $h_i(u_k)$ ) in the calculations.

Table 6.17. Results for the double Liouville system

$u_t = v$ $v_t = -\alpha e^u + \alpha e^{-2u} + u_{2x}$ Scaling symmetry: $\{x, t, u, v, \alpha\} = \{\lambda^1 x, \lambda^1 t, \lambda^0 u, \lambda^{-1} v, \lambda^{-2} \alpha\}$		
Rank	Density ( $\rho$ )	Flux ( $J$ )
1	$f_1(u) u_x$	$-f_1(u) v$
2	$c_1 [\alpha e^{-2u} + 2\alpha e^u + v^2 + u_x^2] +$ $c_2 [2v u_x]$	$c_1 [-2v u_x] +$ $c_2 [\alpha e^{-2u} + 2\alpha e^u - v^2 - u_x^2]$
4	(No new solution)	(No new solution)
6	$f_1(u) [v_x^2 u_{2x} + 2u_x v_x v_{2x}] +$ $f_1'(u) [u_x^2 v_x^2] +$ $f_2(u) [v_x u_{2x}^2 + 2v u_{2x} u_{3x}] +$ $f_2'(u) [v u_x u_{2x}^2]$	$f_1(u) [4\alpha u_x^2 v_x e^{-2u} + 2\alpha e^u u_x^2 v_x -$ $v_x^3 - 2u_x v_x u_{3x}] +$ $f_1'(u) [-v u_x v_x^2] +$ $f_2(u) [-\alpha u_{2x}^2 e^{-2u} + \alpha e^u u_{2x}^2 -$ $u_{2x}^3 - 2v u_{2x} v_{2x}] +$ $f_2'(u) [-v^2 u_{2x}^2]$
8	(119 terms)	(298 terms)

## Chapter 7

### CONCLUSION

Presented in this thesis was an algorithm and code for the symbolic computation of conserved densities for systems of evolution equations with transcendental nonlinearities. The code, `TransPDEDensityFlux.M` implemented in *Mathematica*, has been tested on many equations from the current literature on soliton theory.

Due to the correlation in nonlinear partial differential equation systems between large (that is, theoretically infinite) numbers of conserved densities and complete integrability, the software can be used to test the integrability of such systems. Additionally, the conservation laws themselves can be used further in numerical computations of solutions.

Designing algorithms for a larger class of evolution equations (involving more than one space variable, more types of transcendental functions) would be a possible topic for future research. Also, algorithms for systems with multiple dependent variables with zero weight are currently unavailable. One non-trivial difficulty in attempting algorithms for these types of systems is the location of a reasonable number of examples (and their associated conservation laws) in the literature.

## REFERENCES

- [1] M.J. Ablowitz and P.A. Clarkson, Solitons, Nonlinear Evolution Equations and Inverse Scattering, London Mathematical Society Lecture Note Series **149**, Cambridge University Press, Cambridge (1991).
- [2] P. Adams and W. Hereman, **TransPDEDensityFlux.M**: A Mathematica program for the symbolic computation of conserved densities for systems of partial differential equations with transcendental nonlinearities (2002).  
Available via anonymous FTP from mines.edu in directory  
pub/papers/math\_cs\_dept/software/TransPDEDensityFlux; or via Internet URL:  
[http://www.mines.edu/fs\\_home/whereman/](http://www.mines.edu/fs_home/whereman/).
- [3] S.C. Anco and G. Bluman, Euro. Jnl. of Applied Mathematics **13**, 545-585 (2002).
- [4] B. Bagchi, A. Lahiri, and P.K. Roy, Physical Review D **39** (4), 1186-1189 (1989).
- [5] A. Barone, F. Esposito, C.J. Magee, and A.C. Scott, Rivista del Nuovo Cimento **1** (2), 227-267 (1971).
- [6] R.D. Benguria and M.C. Depassier, J. Phys. A: Math. Gen. **22**, 4135-4142 (1989).
- [7] P.L. Bhatnagar, Nonlinear Waves in One-dimensional Dispersive Systems, Clarendon Press, Oxford (1979).
- [8] K.A. Bronnikov, Acta Physica Polonica B **11**, 3571-3592 (2001).
- [9] R.K. Dodd and R.K. Bullough, Proc. R. Soc. Lond. A. **352**, 481-503 (1977).

- [10] P.G. Drazin and R.S. Johnson, Solitons: an Introduction, Cambridge University Press, Cambridge (1989).
- [11] N. Euler, O. Lindblom, M. Euler, and L.E. Persson, Symmetry in Nonlinear Mathematical Physics **1**, 185-192 (1997).
- [12] Ü. Göktaş and W. Hereman, **CONDENS.M**: A Mathematica program for the symbolic computation of conserved densities for systems of nonlinear evolution equations (1996). Available via anonymous FTP from mines.edu in directory pub/papers/math\_cs\_dept/software/condens; or via Internet URL: [http://www.mines.edu/fs\\_home/whereman/](http://www.mines.edu/fs_home/whereman/).
- [13] Ü. Göktaş and W. Hereman, **InvariantsSymmetries.m**: A Mathematica integrability package for the computation of invariants and symmetries (1997). Available from MathSource (Item: 0208-932, Applications/Mathematics) via FTP: [mathsource.wolfram.com](http://mathsource.wolfram.com) or via Internet URL: <http://www.mathsource.com/cgi-bin/MathSource/Applications/0208-932>.
- [14] Ü. Göktaş and W. Hereman, J. Symbolic Computation, **24**, 591-621 (1997).
- [15] Ü. Göktaş and W. Hereman, Proc. Fourth International Conference on Mathematical and Numerical Aspects of Wave Propagation, Colorado School of Mines, Golden, Colorado, June 1-5, 1998, SIAM, Philadelphia, 403-407 (1998).
- [16] Ü. Göktaş and W. Hereman, J. Symbolic Computation **11**, 1-31 (2001).
- [17] W. Hereman and W. Zhuang, Acta Applicandae Mathematicae **39**, 361-378 (1995). Also: Proceedings of KdV '95 Conference, April 1995, Amsterdam, The Netherlands. Eds.: M. Hazewinkel, H.W. Capel and E.M. de Jager, Kluwer Academic Publishers, Dordrecht, The Netherlands; 361-378 (1995).

- [18] R. Herman, *American Scientist* **80**, 350-361 (1992).
- [19] R. Hirota and J. Satsuma, *Phys. Lett. A* **85**, 407-408 (1981).
- [20] A. Khare, S. Habib, and A. Saxena, *Phys. Rev. Lett.* **79**, 3797 (1997).
- [21] A.V. Kiselev, *Acta Applicandae Mathematicae* **72**, 33-49 (2002).
- [22] M.D. Kruskal, R.M. Miura, C.S. Gardner, and N.J. Zabusky, *J. Math. Phys.* **11**, 952-960 (1970).
- [23] G.L. Lamb, Jr., *Reviews of Modern Physics* **43** (2) part I, 99-124 (1971).
- [24] A.L. Larsen and N.Sanchez, *Phys. Rev.* **D54**, 2801-2807 (1996).
- [25] R.M. Miura, *J. Math. Phys.* **9**, 1202-1204 (1968).
- [26] R. Miura, C.S. Gardner, and M.D. Kruskal, *J. Math. Phys.* **9**, 1204-1209 (1968).
- [27] P.J. Olver, *Applications of Lie Groups to Differential Equations*, Graduate Texts in Math., Springer-Verlag, Berlin (1986).
- [28] Y.Ohtsuki, T. Kato, Y. Fujimura, and S.H. Lin, *J. Chem. Phys.* **106**, 4339-4352 (1997).
- [29] J.A. Sanders and W. Hereman, Private Communication (2001).
- [30] F. Verheest and W. Hereman, *Physica Scripta* **50**, 611-614 (1995).
- [31] R. Willox, W. Hereman, and F. Verheest, *Physica Scripta* **52**, 21-26 (1995).
- [32] T. Wolf, A comparison of four approaches to the calculation of conservation laws, Brock University, Ontario (2001).

- [33] S. Wolfram, *Mathematica*, Addison-Wesley Publishing Company, Inc., Massachusetts (1992).
- [34] V.E. Zakharov (Ed.), What is Integrability?, Springer Series in Nonlinear Dynamics, Springer-Verlag, Berlin (1990).

## Appendix A

### DATA FILES

The following pages give examples of data files used to produce the results in this thesis. The users are allowed to use their own data files, though the format must be similar to those shown here. In particular, all lines not 'commented out' (by `(*` and `*)` ) *must* be present in new data files. We recommend that existing data files be copied and modified.

Two data files are included here, one for the sine-Gordon equation, with a weighted parameter, the other for the sine-Gordon system. Softcopy forms of the data files are available from the same location as the **TransPDEDensityFlux.M** code itself.

```
(* Data file d_sineP.m *)

(* Sine-Gordon Equation w/ weighted parameter *)
eq[1][x,t] = D[D[u[1][x,t],t],x] - alpha*Sin[u[1][x,t]];

noeqs = 1;
name = "Sine-Gordon Equation w/ weighted parameter";
parameters = {};
weightpars = {alpha};

(**** User can supply the rhorank and/or
                                the name for the output file. ****)
(* rhorank = 4; *)
(* myfile = "filename.o"; *)

(**** User can give the weights of u[1] and partial t. ****)
(**** Make sure they are correct! If not, you will see! ****)
(* givenscalerules = {weight[d/dt] -> 1, weightu[1] -> 0}; *)

(**** User can supply the form of rho. ****)
formrho[x,t] = {};

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

```

(* Data file d_sineSY.m *)

(* Sine-Gordon System *)
eq[1][x,t] = D[u[1][x,t],t] - u[2][x,t];
eq[2][x,t] =
    D[u[2][x,t],t] - D[u[1][x,t],{x,2}] - alpha*Sin[u[1][x,t]];

noeqs = 2;
name = "Sine-Gordon System";
parameters = {};
weightpars = {alpha};

(**** User can supply the rhorank and/or
                                the name for the output file. ****)
(* rhorank = 4; *)
(* myfile = "filename.o"; *)

(**** User can give the weights of u[1] and partial t. ****)
(**** Make sure they are correct! If not, you will see! ****)
(* givenscalerules = {weight[d/dt] -> 1, weightu[1] -> 0}; *)

(**** User can supply the form of rho. ****)
formrho[x,t] = {};

(* End of data file d_sineSY.m *)

```