

# On Solving Large Systems of Polynomial Equations Appearing in Discrete Differential Geometry<sup>¶</sup>

T. Wolf

Department of Mathematics, Brock University, St. Catharines, Ontario, Canada L2S 3A1

E-mail: twolf@brocku.ca

Received September 10, 2007

**Abstract**—The paper describes methods for solving very large overdetermined algebraic polynomial systems on an example that appears from a classification of all integrable 3-dimensional scalar discrete quasilinear equations  $Q_3 = 0$  on an elementary cubic cell of the lattice  $\mathbb{Z}^3$ . The overdetermined polynomial algebraic system that has to be solved is far too large to be formulated. A “probing” technique, which replaces independent variables by random integers or zero, allows to formulate subsets of this system.

An automatic alteration of equation formulating steps and equation solving steps leads to an iteration process that solves the computational problem.

**DOI:** 10.1134/S0361768808020047

## 1. INTRODUCTION

The classification of integrable equations, whether Hamiltonian systems of ordinary differential equations (ODEs) [1], systems of evolutionary vector partial differential equations (PDEs) [2], hyperbolic systems [3], or supersymmetric systems [4–6] is a rich source of overdetermined systems of equations, which themselves pose interesting challenges to computer algebra. For example, the computational problem resulting from trying to find Hamiltonian ODE systems with higher degree first integrals is hard partly because of the large size of the overdetermined polynomial algebraic problem with, typically, 400–700 unknowns and twice as many equations but even more so because of the non-trivial nature of the solutions that included single polynomial equations solvable only in radicals, which complicated the solution process. To give another example, the classification of integrable hyperbolic systems does not lead to overdetermined *algebraic* systems but to overdetermined non-linear *PDE* systems.

This paper describes work on the classification of 3-dimensional integrable scalar discrete equations. The mathematical aspects of this project are discussed in detail in [7] and are based on a collaboration with S. Tsarev. In this article, the emphasis lies on the implied computational challenge following from the astronomical size of the overdetermined system.

Although there are only up to 22 unknowns, the algebraic polynomial conditions are hard because they are of degree 8, there are many ( $6.4 \times 10^9$ ) and they are large with a total number of at least  $10^{14}$  terms. Thus, the system of equations is far too big to be even only

formulated initially. A special ‘probing’ technique had been developed to generate at least subsets of the conditions so that an iterative algorithm, which alternates solution steps and problem formulation steps, was able to make progress and finally solve the complete problem.

Computations for this paper had been made in the computer algebra systems Reduce and Form [8], in particular with the Reduce package Crack [9, 10], which is the workhorse behind all the applications listed above.

In the following section, we introduce the mathematical problem and formulate the result of the computations. This section contains enough details to make this article selfconsistent, but it may be skipped by the reader who is exclusively interested in computer algebra issues.

An estimate for the initial size of the computational problem if it would be formulated ad hoc is derived in Section 3. In Section 4, we take a closer look at the computational challenge and motivate a “probing technique” to be introduced in Section 5. A dynamic alteration of the generation of equations and the solution of equations is described in Section 6. Although the random “probing” is very application specific, it was possible to design the dynamic iteration of solving and formulating equations essentially in a general form, being re-usable in future applications of the Crack package. In the last Section 7, we make comments about the time needed to complete the project.

<sup>¶</sup> The text was submitted by the author in English.

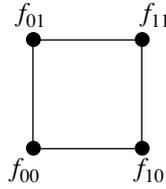


Fig. 1. Square  $K_2$ .

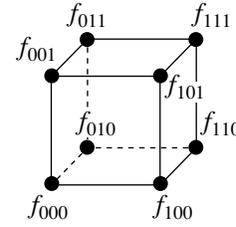


Fig. 2. Cube  $K_3$ .

## 2. INTEGRABLE SCALAR DISCRETE EQUATIONS

### 2.1. Face Formulas

In the relatively short period of about a decade, the field of discrete differential geometry already provided much insight to structures that are fundamental both to classical differential geometry and to the theory of integrable PDEs. Their objects of interest are classical surfaces and PDEs; both are smooth limits along some directions of discrete lattices built from elementary cubes. This paper deals with integrable fields on elementary cubes.

In the following, we consider an  $n$ -dimensional elementary cube  $K_n = \{(i_1, \dots, i_n) | i_s \in \{0, 1\}\}$  with coordinate values  $x_i$  at each corner being either zero or one. To each of the  $2^n$  corners  $(i_1, \dots, i_n)$  is attached a field variable  $f_{i_1, \dots, i_n}$ . We will use the short notation  $\mathbf{f}$  for the set  $(f_{00\dots 0}, \dots, f_{11\dots 1})$  of all these  $2^n$  variables.

By attaching cube cells to each other, one can assemble an  $n$ -dimensional cubic lattice  $\mathbb{Z}^n$  with field variables  $f_{i_1, \dots, i_n}$ ,  $i_j \in \mathbb{Z}$  assigned to each lattice point. Of interest are such discrete fields  $\mathbf{f}$  that satisfy on each single elementary cube one and the same relation

$$Q_n(\mathbf{f}) = 0 \tag{1}$$

between the  $f$ -values at the corners of this cube. Of special interest, and thus objects of intensive study, are Eq. (1) that are integrable in some sense. We are interested in an  $(n + 1)$ -dimensional consistency as this is a criterion underlying the integrability of many classical integrable nonlinear PDE that appear as continuous limits along some of the discrete directions.

An  $n$ -dimensional discrete Eq. (1) is called consistent if it may be imposed in a consistent way on all  $n$ -dimensional faces of an  $(n + 1)$ -dimensional cube.

Thus, Eqs. (1) are also called “face relations.” Two-dimensional face relations

$$Q_2(f_{00}, f_{10}, f_{01}, f_{11}) = 0 \tag{2}$$

were studied in detail in [11, 12]. Of special interest are face relations that have the following two extra properties.

(1) **Quasilinearity.** To be able to solve face relation (1) for any of the occurring  $f$ -values, it should be affine linear w.r.t. every  $f_{i\dots j}$ ; i.e.  $Q$  has degree one in any of its variables. In two dimensions, this means

$$\begin{aligned} Q_2 &= c_1(f_{10}, f_{01}, f_{11})f_{00} + c_2(f_{10}, f_{01}, f_{11}) \\ &= c_3(f_{00}, f_{01}, f_{11})f_{10} + c_4(f_{00}, f_{01}, f_{11}) \\ &= \dots = q_{1111}f_{00}f_{10}f_{01}f_{11} + q_{1110}f_{00}f_{10}f_{01} \\ &\quad + q_{1101}f_{00}f_{10}f_{11} + \dots + q_{0000} \end{aligned} \tag{3}$$

and, in  $n$  dimensions,

$$Q_n = \sum_{\mathcal{D}} q_{\mathcal{D}} \prod_{i_s=0,1} (f_{i_1\dots i_n})^{D_{i_1\dots i_n}} \tag{4}$$

with constant coefficients  $q_{\mathcal{D}}$ , where the summation is taken over all  $2^{2^n}$  many  $2^n$ -tuples  $\mathcal{D} = (D_{00\dots 0}, \dots, D_{11\dots 1})$ , each power  $D_{i_1\dots i_n}$  of the respective vertex variable  $f_{i_1\dots i_n}$  being either 0 or 1. In other words, the  $2^n$  indices of  $q_{\mathcal{D}}$  are the exponents of the  $2^n$  vertex field variables  $f_{i_1\dots i_n}$ , with each exponent  $D_{i_1\dots i_n}$  being 0 or 1.

(2) **Symmetry.** Equation (1) should be invariant w.r.t. the symmetry group of the  $n$ -dimensional cube. Subsection 2.3 is devoted to computational and conceptual simplifications resulting from it.

### 2.2. The Consistency Condition

In the case of a 2d face, formula (3) defined on the square of Fig. 1 to be 3d-consistent (also often called to be integrable) the above consistency condition takes the following form.

Let us assume that field value  $f_{000}$  in the lower left front corner in Fig. 2 and values  $f_{100}$ ,  $f_{010}$ , and  $f_{001}$  at the neighboring corners are arbitrarily given. By imposing (3) to hold on the three “initial 2d-faces”  $\{x_1 = 0\}$ :  $Q(f_{000}, f_{010}, f_{001}, f_{011}) = 0$ ,  $\{x_2 = 0\}$ :  $Q(f_{000}, f_{100}, f_{001}, f_{101}) = 0$ ,  $\{x_3 = 0\}$ :  $Q(f_{000}, f_{100}, f_{010}, f_{110}) = 0$ , the affine linearity of the relations allows us to find  $f_{011}$ ,  $f_{101}$ , and  $f_{110}$  situated diagonally opposite to  $f_{000}$  on those faces. By applying these relations again, now to the other three “final 2d-faces”  $\{x_1 = 1\}$ :  $Q(f_{100}, f_{110}, f_{101}, f_{111}) = 0$ ,  $\{x_2 = 1\}$ :  $Q(f_{010}, f_{110}, f_{011}, f_{111}) = 0$ ,  $\{x_3 = 1\}$ :  $Q(f_{001}, f_{101},$

$f_{011}, f_{111}) = 0$ , we obtain three rational expressions involving  $f_{111}$ , which, according to the *3d-consistency* condition, must be consistent for arbitrary “initial data”  $f_{000}, f_{100}, f_{010}, f_{001}$ . By eliminating  $f_{111}$  from one of the three expressions and substituting it in the other two, we obtain two polynomial identities, which have to be split w.r.t. the four independent initial field variables  $f_{000}, f_{100}, f_{010}, f_{001}$  resulting in a large overdetermined polynomial system for the unknown coefficients  $q_{ijkl}$  in the face formula (3).

In [11], a complete classification of *3d-consistent 2d-face* formulas (in a slightly different setting) was obtained; in [12], a similar classification was given for the case when one does *not* assume that the formula (2) is the same on all the six faces of the three-dimensional cube, but all face relations are still assumed to be affine linear (and in some sense nondegenerate).

### 2.3. Cubical Symmetries

In a first investigation, we classified 2-, 3- and 4-dimensional affine linear face relations according to their complete symmetry group of the respective  $n$ -dimensional cube.

Apart from simplifying the computational problem of classifying face relations (1), the symmetry requirement also simplifies the conceptual problem in which orientation to apply the  $n$ -dimensional formula (1) to each of the faces of the  $(n + 1)$ -dimensional cube. For example, in  $n = 2$  dimensions, the face formula (3) defined on the square in Fig. 1 can be attached in eight different ways to each of the six faces of the cube in Fig. 2, but, if (3) is symmetric, then all eight ways are equivalent.

We take as the symmetry generating set the reflection w.r.t. the plane  $x_1 = 1/2$  and, further,  $(n - 1)$  diagonal reflections w.r.t. the planes  $x_1 = x_s, s = 2, \dots, n$  (here,  $x_k$  denote the coordinates in  $\mathbb{R}^n$ ). Thus, the symmetry is characterized by  $n$  signs according to the requirement  $Q(\mathbf{f}) = \pm Q(R(\mathbf{f}))$ , which has to be satisfied identically in all  $\mathbf{f}$ . For  $n = 3$ , this generates eight cases, each with about 770 equations for the  $2^{2^3} = 256$  coefficients  $q_{D_1 \dots D_8}$  of the affine linear face formula.

An  $n = 4$  dimensional cube has  $2^4$  corners and as many  $\mathbf{f}$  variables giving  $2^{2^4} = 65\,536$  terms and as many undetermined coefficients in the affine linear face formula. In each of the 16 symmetry cases, a splitting w.r.t. the variables  $f_{i_1 \dots i_n}$  results in about 250000 linear (sparse) equations for the 65536 coefficients  $q_{D_1 \dots D_{16}}$ .

For  $n = 3$  (the case of interest in this paper), only three symmetries allow face formulas that are not identically zero:

- ( $---$ ):  $Q$  has 1 parameter and 24 terms,
- ( $-++$ ):  $Q$  has 13 parameters and 186 terms,

( $+++$ ):  $Q$  has 22 parameters and 256 terms.

Results for dimensions 2 and 4, as well as a classification of face formulas that are invariant under the action of the more restrictive  $SL_2(\mathbb{C})$  group of fractional-linear transformations

$$\mathbf{f} \mapsto (a\mathbf{f} + b)/(c\mathbf{f} + d) \tag{5}$$

(all group parameters  $a, b, c, d$  are the same for all the vertices of the cube), are given in [7].

To solve the sparse but rather extensive linear systems of symmetry conditions for the coefficients  $q_{\mathcal{D}}$  appearing after splitting w.r.t. the variables  $f_{i_1 \dots i_n}$ , a special linear equation solver StreamSolve had to be written. It can be downloaded together with other material related to this publication from <http://lie.math.brocku.ca/twolf/papers/TsWo2007/readme>.

For example, when computing for  $n = 4$  from the face formula that already has the symmetry ( $-+++$ ) (and thus had already reduced the number of unknown coefficients) a specialization which in addition has the  $SL_2(\mathbb{C})$ -symmetry, then this program reads 24556 equations from a 5.2 MB input file, solves this linear algebraic system, and writes the solution into a file in an 80 MB session of Reduce 3.8 running under Linux on a 32 bit 1.7 GHz Pentium IV dual CPU PC in 16 sec CPU. The default Reduce solver SOLVE was not able to solve this or similar systems but has been improved recently using ideas from StreamSolve.

### 2.4. Results

Before turning to the central content of this article in the following section, which concerns the size of the consistency conditions to be solved, we want to discuss their solutions as they are an indicator of the size of computations too.

The first case ( $---$ ) is totally skew-symmetric and has only one parameter (a constant overall factor); so, this symmetry case specifies  $Q$  completely:

$$\begin{aligned} Q_{(---)} &= (f_{100} - f_{001})(f_{010} - f_{111}) \\ &\times (f_{101} - f_{110})(f_{011} - f_{001}) \\ &- (f_{001} - f_{010})(f_{111} - f_{100}) \\ &\times (f_{000} - f_{101})(f_{110} - f_{011}), \end{aligned} \tag{6}$$

which is the so-called discrete Schwarzian bi-Kadomtsev–Petviashvili system (dBKP-system), an integrable discrete system found in [13, 14] and studied in [11], where the fact of its *4d-consistency* was first established.

For the two other cases, ( $-++$ ) and ( $+++$ ), the remaining 13 and 22 parameters had to be determined from consistency conditions. The complete solution consists, in the case ( $-++$ ), of three solutions with 1, 2, and 3 free parameters and, in the case ( $+++$ ), of five solutions with 2, 2, 3, 3, and 4 free parameters, all

shown at <http://lie.math.brocku.ca/twolf/papers/TsWo2007/SolutionsOfConsistency>. All these solutions (in contrast to (6)) turned out to be mathematically trivial, as, for each of them,  $SL_2(\mathbb{C})$ -transformations (5) were found that transform the face formula into one of the three trivial versions

$$Q^{(1)} = f_{000}f_{001}f_{010}f_{011}f_{100}f_{101}f_{110}f_{111} - \sigma, \quad (7)$$

$$Q^{(2)} = f_{001}f_{010}f_{100}f_{111} - \sigma f_{000}f_{011}f_{101}f_{110}, \quad (8)$$

$$\begin{aligned} Q^{(3)} &= (f_{001} + f_{010} + f_{100} + f_{111}) \\ &\quad - \sigma(f_{000} + f_{011} + f_{101} + f_{110}), \quad (9) \\ \sigma &= \pm 1. \end{aligned}$$

Although being mathematically trivial, some solutions involve extended rational expressions and are surely not computationally trivial. For example, for solution case+++/s4 from the web page given above, the corresponding face formula is too big to be shown in this article.<sup>1</sup> It is also too large to be checked to be a solution directly, even with the computer algebra system FORM (which was used to check the consistency of (6)). A correctness test had to be done through finding  $SL_2(\mathbb{C})$ -transformations converting them to the forms (7), (8), and (9).

The non-trivial size of some solutions has some relevance for Section 4, which deals with the main computational problem: the formulation and solution of the consistency conditions. The complexity in solving a system of equations (not in formulating it) is, typically, closely related to the number and complexity of the solutions themselves.

Before, we will make general considerations about the problem of formulating the full set of consistency conditions.

### 3. THE SIZE OF CONSISTENCY CONDITIONS

The following considerations are made under the assumption of generic unknown coefficients  $q_{\mathcal{D}}$  in the face formula (4) not satisfying additional symmetry conditions.

Any  $(n+1)$ -dimensional hypercube built from  $2^{n+1}$  vertices  $f_{i_1 \dots i_{n+1}}$ ,  $i_k \in \{0, 1\}$  has  $2(n+1)$  faces located in the (logical) planes  $x_k = 0$  and  $x_k = 1$ ,  $k = 1, \dots, (n+1)$ . The face relations for the  $n+1$  faces that correspond to  $x_k = 0$  are

$$0 = \sum_{\mathcal{D}} q_{\mathcal{D}} \prod_{i_s = 0, 1} (f_{i_1 \dots i_{k-1} 0 i_{k+1} \dots i_{n+1}})^{D_{i_1 \dots i_{k-1} i_{k+1} \dots i_{n+1}}}. \quad (10)$$

They can be used to determine  $f_{1 \dots 1 0 1 \dots 1}$  with the 0 being in the  $k$ th index position. Each of these face relations

<sup>1</sup> With  $Q$  in ASCII code taking 200 kB.

involves  $2^n f$ -variables and, thus,  $2^{2^n}$  terms, half of them include  $f_{1 \dots 1 0 1 \dots 1}$  as a factor and the other half does not. Solving the face relation  $x_k = 0$ ,

$$0 = A_k f_{1 \dots 1 0 1 \dots 1} + B_k, \quad (11)$$

where  $A_k, B_k$  are expressions in  $q_{\mathcal{D}}, f_{\beta}$  for  $f_{1 \dots 1 0 1 \dots 1}$ , and substituting  $f_{1 \dots 1 0 1 \dots 1} = -B_k/A_k$  in any expression that involves  $f_{1 \dots 1 0 1 \dots 1}$  linearly (like other face relations) and taking the numerator over the common denominator amounts to multiplying all terms that involve  $f_{1 \dots 1 0 1 \dots 1}$  by  $-B_k$  and all other terms by  $A_k$ . As  $A_k$  and  $B_k$  involve each  $2^{2^n}/2 = 2^{2^n-1}$  terms, this means that a substitution of  $f_{1 \dots 1 0 1 \dots 1}$  increases the number of terms by a factor of  $2^{2^n-1}$ , before cancellations and reductions will be made.

The 2nd half of face relations for the  $n+1$  faces that correspond to  $x_k = 1$  are

$$0 = \sum_{\mathcal{D}} q_{\mathcal{D}} \prod_{i_s = 0, 1} (f_{i_1 \dots i_{k-1} 1 i_{k+1} \dots i_{n+1}})^{D_{i_1 \dots i_{k-1} i_{k+1} \dots i_{n+1}}}. \quad (12)$$

Each one of them involves  $f_{11 \dots 1}$  and  $n$  of those  $f$ -variables which have exactly one 0 as index in any one of the  $n+1$  index positions apart from the  $k$ th position. Replacing each one of these  $n$   $f$ -variables by using the corresponding  $x_l = 0$  face relation increases the number

of terms by a factor  $2^{2^n-1}$  each time, giving in total  $2^{2^n} (2^{2^n-1})^n = 2^{2^n(n+1)-n}$  terms. In each substitution, the degree of the coefficients  $q_{\mathcal{D}}$  increases by one, reaching finally  $n+1$ .

Solving one of the  $n+1$  many  $x_k = 1$  face relations

$$0 = G_k f_{11 \dots 1} + H_k \quad (13)$$

for  $f_{11 \dots 1}$  and substituting  $f_{11 \dots 1} = -H_k/G_k$  in the other face relations, gives  $n$  independent consistency conditions

$$G_j H_k = G_k H_j, \quad j = 1, \dots, k-1, k+1, \dots, n+1 \quad (14)$$

with each  $G_i$  and  $H_i$  having  $2^{2^n(n+1)-n}/2$  terms, i.e., each consistency condition involving  $2(2^{2^n(n+1)-n-1})^2$  terms. The total number of terms of the  $n$  consistency conditions is thus  $n 2^{\{2^{n+1}(n+1)-2n-1\}}$ .

To compute an upper bound of the number of conditions that result from splitting each consistency condition with respect to the independent  $f$ -variables, we note that their highest degree is equal to the total degree of all  $q_{\mathcal{D}}$ ; i.e., it is  $2n+2$ . The only exception is  $f_{00 \dots 0}$ , which does not occur in the face relations (12). It enters only through substitutions, so its highest degree in the constraints is  $2n$ . We thus get for an upper bound of the number of different products of different powers  $f_{00 \dots 0}$  and the other  $2^{n+1} - n - 3$  independent  $f$ -variables the

Faces and consistency conditions in different dimensions

Dimension of face	$n$	2	3	4	5
# of $f$ -variables in face formula	$2^n$	4	8	16	32
# of terms in face formula (= # of undermined coefficients $q_D$ in $Q_n$ )	$2^{2^n}$	16	256	65536	$4.3 \times 10^9$
# of all $f$ -variables in $(n + 1)$ -dim. hypercube	$2^{n+1}$	8	16	32	64
# of indep. $f$ -variables in $(n + 1)$ -dim. hypercube	$2^{n+1} - n - 2$	4	11	26	57
# of $n$ -dim. faces in $(n + 1)$ -dim. hypercube	$2(n + 1)$	6	8	10	12
# of consistency conditions	$n$	2	3	4	5
Upper bound on the # of terms of each condition	$2^{\{2^{n+1}(n+1)-2n-1\}}$	$5.2 \times 10^5$	$1.4 \times 10^{17}$	$2.8 \times 10^{45}$	$1.9 \times 10^{112}$
Total degree of the $q_D$ in each condition	$2n + 2$	6	8	10	12
Upper bound estimate of the # of equations resulting from splitting each condition	$2n(2n + 2)^{(2^{n+1}-n-3)}$	864	$6.4 \times 10^9$	$8.0 \times 10^{25}$	$2.7 \times 10^{61}$
Estimated average # of terms in each equation	$\frac{2^{2^{n+1}(n+1)-2n-1}}{2n(2n + 2)^{(2^{n+1}-n-3)}}$	606	$2.2 \times 10^7$	$3.5 \times 10^{19}$	$7 \times 10^{50}$

value  $2n(2n + 2)^{(2^{n+1}-n-3)}$ . With this number and the number of terms of each constraint, we get with their quotient an estimate of the average number of terms in each equation (see the table).

*Remark.* Although, strictly speaking, these are upper bounds for the size of conditions, one must keep in mind that any computer algebra system would have to generate all these terms by expanding the brackets before searching for possible cancellations or reductions. As test runs with Form had shown, for the 3-dimensional case (+++) described in Section 2.3 (with a much smaller number of independent  $q_{\mathcal{Q}}$  than  $2^2$  but the same number of terms in  $Q_3$ ), the total number of terms to be generated for each of the three consistency conditions is around  $10^{14}$  (compared to the upper bound of  $1.4 \times 10^{17}$  in the table).

4. THE COMPUTATIONAL PROBLEM

As outlined before (cf. Section 2.2), the task consists of the following steps.

- (1) Formulate the relations for the  $n + 1$  “initial faces”  $x_k = 0$ .
- (2) Solve them for  $f_{11\dots 101\dots 1}$ .
- (3) Formulate the relations for the  $n + 1$  “final faces”  $x_k = 1$ .
- (4) Perform the substitutions obtained under (2) in the relations of (3).
- (5) Solve one of the resulting relations for  $f_{11\dots 1}$ .
- (6) Substitute  $f_{11\dots 1}$  in all other  $n$  face relations of 4.

(7) Split these consistency conditions with respect to all the occurring independent  $f$ -variables to obtain an overdetermined system of equations for the unknowns  $q_{\mathcal{Q}}$ .

(8) Find the general solution of this system.

(9) Reduce the number of free parameters of the solutions using  $SL_2(\mathbb{C})$ -transformations (5).

Although the algebraic system for the unknown coefficients  $q_{\mathcal{Q}}$  is heavily overdetermined the following difficulties appear.

(1) Strictly speaking, in order to formulate even only the smallest subset of conditions, one would have to formulate at least one consistency condition (by performing steps 1 and 2 completely and 3–6 for at least two  $x_k = 1$  face relations (12) before step 7), i.e., to generate an expression with  $2^{\{2^{n+1}(n+1)-2n-1\}}$  terms according to Section 3.

(2) If one found a way around this hurdle, then the resulting equations are of high degree  $2n + 2$  with, on average, many terms.

(3) Even if one were able to generate 100000’s of equations and, thus, find shorter ones, which one could solve for some unknowns in terms of others, one would face the problem that many cases and sub-sub-cases have to be investigated due to the high degree of the equations and

(4) that one has to generate billions of equations to find some that are independent of the ones generated so far. As we explain below, due to the “near triangular” form of this huge system (see Section 6), one cannot hope that the first, say,  $10^6$  conditions will be equivalent to the full system of equations for  $q_{\mathcal{Q}}$ , even though we have only very few unknown  $q_{\mathcal{Q}}$ ’s.

In the computations to be described in this paper, we have  $n = 3$ . The full problem (without cubical symmetry) would mean to generate three consistency conditions involving each about  $10^{17}$  terms that split into an estimate of  $10^{10}$  polynomial equations, each being homogeneous of degree 8 for 256 unknowns and involving on average over  $10^7$  terms.

To make progress, we introduce different solving techniques but also simplify the problem.

(1) We restrict ourselves to face formulas that obey the full cubical symmetry (cf. Section 2.3). In the third case (+ + +) (the hardest case), the fact that only 22 of the  $q_{\mathcal{Q}}$  are parametric simplifies the problem, but the simplification is limited because none of the other 234  $q_{\mathcal{Q}}$  needs to vanish just because of the extra cubical symmetry.

(2) We ease the formulation of the large consistency conditions (before splitting them into smaller equations) by replacing  $z$  of the  $v := 2^{n+1} - n - 2$  independent  $f$ -variables by zero, and replacing  $u$  of them by random integer values un-equal zero, leaving  $s := v - z - n$  of them in symbolic form. As a consequence, we only get a set of necessary and not sufficient equations for the  $q_{\mathcal{Q}}$ , but we can repeat this procedure on a gradually increasing level of generality (by increasing  $s$  and lowering  $z$ ). A crucial feature is the use of knowledge accumulated so far about the solution to formulate new necessary systems of equations (cf. Section 5).

(3) We design a dynamic process that automatically organizes an iteration process that generates and solves/simplifies/'makes use of' necessary equations automatically. As an important feature, a set of newly generated necessary equations is not read into the ongoing computation at once, but gradually on demand (cf. Section 6).

(4) A recent extension is the parallelization of different case investigations on a computer cluster.

We discuss points 2 and 3 in more detail. Transparencies of a talk given about point 4 are available from [15].

## 5. RANDOM PROBING

The method under point 2 of the above list is based on replacing a number  $z$  of the  $v := 2^{n+1} - n - 2$  independent  $f$ -variables by zero (for  $n = 3$ ,  $v = 11$ ) and replacing a number  $u$  of them by random integer non-zero values, leaving  $s := v - z - u$  of them in symbolic form.

The computation has 3 phases: finding solutions, verifying the obtained solutions probabilistically and again rigorously. The first two phases use the probing technique. Still, it makes sense to distinguish them because optimal values of  $z$ ,  $u$  and  $s$  differ in both cases.

In the probing technique, the two types of replacements, i.e. replacing  $f_{\alpha}$  by zero or by a non-zero integer, share the same disadvantage: the number of indepen-

dent parameters, i.e., of symbolic  $f_{\alpha}$  which allow to split the consistency condition into many smaller equations, is reduced by one. The advantage of replacing an  $f_{\alpha}$  by zero instead of a non-zero integer is that expressions shrink more. Also, replacements by non-zero integers often introduce extra solutions to the generated conditions and it appears to be costly to eliminate these spurious temporary solutions by leading them to a contradiction with more conditions to be generated based on other random replacements. Therefore, it is more productive to replace, for example, *one*  $f_{\alpha}$  by zero than to replace *two*  $f_{\alpha}$  by non-zero integers.

Consequently, in the first phase of *finding solutions* at most one replacement by a non-zero integer is made (i.e.,  $u \leq 1$ ), and, after starting with  $z = 9$ ,  $u = 0$ ,  $s = 2$ , one increases generality gradually by either changing  $u$  from 0 to 1 and decreasing  $z$  by 1, or by decreasing  $u$  from 1 to 0 and increasing  $s$  by 1 until  $z = 1$ ,  $u = 0$ ,  $s = 10$ . We need the occasional substitution by one non-zero integer, because we want to increase the generality in as small as possible steps in order to avoid the generation of too many too high degree equations with too many terms. This would happen if we decrease  $z$  by 1, keep  $u = 0$ , and increase  $s$  by 1. A run in full generality  $z = 0$ ,  $u = 0$ ,  $s = 11$  is computationally prohibitive; therefore, in a second phase of *confirming the found solution probabilistically*, one starts with  $z = 4$ ,  $u = 1$ ,  $s = 6$  and increases generality by decreasing  $z$ , increasing  $u$  and keeping  $s$  constant until  $z = 0$ ,  $u = 5$ ,  $s = 6$ . By testing a hypothetical solution with this final setting many times, the correctness of the solution is confirmed with an arbitrarily high probability.

In both phases, we want to generate as few and as simple equations as possible in each generation step. So, we continue using the same setting of  $z$ ,  $u$ ,  $s$  as long as possible (i.e., as long as there are new conditions still resulting after randomly choosing other sets of  $z$  many  $f_{\alpha}$  to be 0,  $s$  of  $f_{\alpha}$  to be kept symbolic and randomly assigning integer non-zero values to the other  $u$  parametric  $f_{\alpha}$ ) before generalizing it, i.e., making  $s$  larger and  $z$  smaller. This is regulated by one parameter, which specifies the maximum number of consecutive times that a "probing" (generation of conditions) attempt yielded only identities before changing  $z$ ,  $u$ ,  $s$ .

In a third phase, after all hypothetical solutions have been obtained and been checked probabilistically, they are checked again, now *rigorously*. This has been done either by a brute force check using the computer algebra system Form or by using  $SL_2(\mathbb{C})$ -transformations on the field variables  $\mathbf{f}$  to reduce solutions to integrable trivial forms (7), (8), and (9) in Section 2.4.

A helpful and initially unexpected feature of the probing technique is that the resulting equations appear to be somehow triangularized in the following sense. Each unknown  $q_{i_1 i_2 \dots i_m}$ ,  $m = 2^n$ ,  $i_j \in \{0, 1\}$  is the coefficient of a product of  $i_1 + i_2 + \dots + i_m$  many different factors  $f_{\alpha}$ . That means that, at the beginning of the computation when many  $f_{\alpha}$  are replaced by zero, the  $q_{\mathcal{Q}}$  with a

high index sum do not occur. Only later on, as fewer  $f_\alpha$  are replaced by zero, gradually  $q_{\mathfrak{Q}}$  with higher index sum appear in the equations. On the one hand, this is a good feature providing a partially triangularized system of equations. On the other hand, this means that, although we only want to compute a relatively small number of unknowns (for  $n = 3$  and case  $(+ + +)$ , these are 22  $q_{\mathfrak{Q}}$ ), it is not enough to formulate only a comparable number of the huge total set of equations (about  $6.4 \times 10^9$  equations for  $n = 3$ ). A set of equations that is equivalent to the complete set of equations is only obtained towards the end of generalizations. For example,  $q_{11\dots 1}$  is the coefficient of the product of all the  $2^n$  many  $\mathbf{f}$  occurring in a face formula. For  $q_{11\dots 1}$  to occur in a consistency condition, it must occur in at least one face formula; i.e., all the  $2^n$  many  $\mathbf{f}$  must occur in at least one face formula (i.e., none of them can be replaced by zero). The condition is slightly weaker, because only  $2^n - 1$  of them belong to the independent  $f_\alpha$ , so the number of non-zero  $\mathbf{f}$  must be at least  $2^n - 1$ , i.e.,  $(u + s) \geq (2^n - 1)$  for  $q_{11\dots 1}$  to have a chance to appear in a consistency condition.

That triangular dependence of generated equations, in turn, means that millions of the early equations are redundant, which implies a large inefficiency in generating equations. This can be avoided by using known relations  $q_{\mathfrak{Q}} = h_{\mathfrak{Q}}(q'_{\mathfrak{Q}})$ , which were derived in the solution process so far as automatic simplification rules when generating new equations.

Three problems remain to be considered.

(1) By replacing  $f_\alpha$  through numbers, it may happen that  $A_k$  in any one of the  $x_k = 0$  face relations (11) becomes zero. Then, a new equation generation attempt with different or more general random replacements has to be made.

(2) Similarly, it may happen that the coefficient of  $q_{11\dots 1}$  in all  $n + 1$  many  $x_k = 1$  face relations (13) is zero. Then a different replacement has to be tried as well.

(3) Even if none of the  $A_k$  in (13) becomes zero, it may happen that  $A_k$  and  $B_k$  in one face relation are not prime and, then, a solution for the  $q_{\mathfrak{Q}}$ , which makes the greatest common divisor  $GCD(A_k, B_k)$  to zero, is potentially lost when performing substitutions  $f_{1\dots 101\dots 1} = -B_k/A_k$ . The same applies to common factors of  $G_k$  and  $H_k$  in the single face relation (13) that is applied to replace  $f_{11\dots 1}$ . Therefore, each consistency condition has to be multiplied with a product of all common factors of any pair  $A_k, B_k$  and of all common factors of the pair  $G_k, H_k$ , which is used to replace  $f_{11\dots 1}$ . To lower the computational cost, one drops multiplicities of the factors. These factors involve, in general,  $q_{\mathfrak{Q}}$ , as well as  $f_\alpha$ , and, therefore, the multiplication has to be done before splitting the consistency condition with respect to the independent  $f_\alpha$ . Alternatively, one can split the consistency

condition before multiplication and, instead, multiply and duplicate the equations in the following way.

Let  $P(q_{\mathfrak{Q}}, f_\alpha)$  be one of the above-mentioned factors and let  $0 = P$  be split into a system  $0 = \hat{P}_i(q_{\mathfrak{Q}})$ , where redundant equations are dropped.<sup>2</sup> Instead of multiplying a constraint  $0 = C(q_{\mathfrak{Q}}, f_\alpha)$  with  $P$ , splitting the equation  $0 = PC$  into individual equations, and factorizing all of them afterwards, it is equivalent, but much more efficient, to split  $0 = C$  into a system of equations  $0 = \hat{C}_j(q_{\mathfrak{Q}})$  and  $0 = P$  into a system  $0 = \hat{P}_i(q_{\mathfrak{Q}})$  and to consider the equivalent system  $0 = \hat{P}_i \hat{C}_j, \forall i, j$ .

To summarize, in order not to lose solutions for the  $q_{\mathfrak{Q}}$ , the procedure is

- to collect all common factors  $P_r$  of all pairs  $A_k, B_k$  and of the pair  $G_k, H_k$  used to substitute  $f_{11\dots 1}$ ;
- to drop duplicate factors;
- to split all consistency conditions giving a system  $S$  of equations  $0 = \hat{C}_j$ ; and
- to split, for each factor  $P_r$ , the equation  $0 = P_r$  into a system  $0 = \hat{P}_{ri}, i = 1 \dots i_r$ , where, again, equations that are redundant within one such system are dropped.
- If a system  $0 = \hat{P}_{ri}$  includes a non-vanishing  $\hat{P}_{ri}$  either because  $\hat{P}_{ri} = 1$  or because  $\hat{P}_{ri}(q_{\mathfrak{Q}})$  is known to be non-zero based on the inequalities that are known for some  $q_{\mathfrak{Q}}$ ,<sup>3</sup> then this system is ignored because the corresponding  $P_r$  is non-zero. For every other such system  $0 = \hat{P}_{ri}$ , replace the system  $S$  of conditions  $0 = \hat{C}_j$  by the new system  $\hat{S}$  consisting of the equations  $0 = \hat{P}_{ri} \hat{C}_j, \forall i, j$ .

## 6. A SUCCESSION OF GENERATING AND SOLVING EQUATIONS

The system of algebraic equations for the  $q_{\mathfrak{Q}}$  is investigated by the computer algebra package Crack, which aims at solving polynomial algebraic or differential systems, typically systems that are overdetermined and very large. It offers various degrees of interactivity, from fully automatic to fully interactive. The package consists of about 40 modules, which perform different steps, like substitutions, factorizations, shortenings, Gröbner basis steps, integrations, separations, etc.,

<sup>2</sup> The definition of “redundant” depends of the effort one wants to spend at this stage. In the implementation of this algorithm, the polynomials  $\hat{P}_i$  are divided by the coefficients of their leading terms with respect to some ordering of the  $q_{\mathfrak{Q}}$ , and, then, duplicate  $\hat{P}_i$  are dropped.

<sup>3</sup> In the process of solving the non-linear system of conditions for the  $q_{\mathfrak{Q}}$ , many cases and sub-cases are considered that lead to extra equations  $q_{\mathfrak{Q}} = 0$  and inequalities  $q_{\mathfrak{Q}} \neq 0$  (see Section 3). All such information, i.e., evaluations  $q_{\mathfrak{Q}} = \dots$  and inequalities  $q_{\mathfrak{Q}} \neq 0$ , is used when formulating new consistency conditions where inequalities become useful.

which can be executed in any order. In automatic computations, their application is governed by a priority list where highly beneficial, low-cost and low-risk (of exploding the size of equations) steps come first. Modules are tried from the beginning of this list to its end until an attempt is successful and then execution returns to the start of the list, and modules are tried again in that order. This simple principle is refined in a number of ways. For more details, see [9, 10].

In order to accommodate the dynamic generation of equations and their successive use, all that had to be done was to add two modules:

- one for generating a new set of necessary conditions using the “probing” technique from Section 5 and writing the generated equations into a buffer file, and
- another module for reading one non-trivial equation from this buffer file (i.e., for continuously reading equations until one is obtained that is not instantly simplified to an identity modulo the known equations or until the end of this file is reached),

and to determine the place of these modules in the priority list. The need for a buffer file arose because the number of equations generated in each “probing” is unpredictable, especially in view of the large impact that factors  $\hat{P}_{ri}$  can have on the number of equations (see the end of Section 5).<sup>4</sup>

*Some more miscellaneous comments.* As shown in Section 5, the generated equations often take the form of products set equal to zero. This leads to many case distinctions of factors being either zero or non-zero and, consequently, other factors being zero. The depth of sub-case levels sometimes reaches 20. Because buffer files are only valid for the case in which they were generated (because they make use of case-dependent known substitutions  $q_{\mathfrak{g}} = h_{\mathfrak{g}}(q'_{\mathfrak{g}})$  and case-dependent inequalities) and because of these deep levels of sub-cases, the number of buffer files easily reaches 100000 and more (e.g., too many to be deleted with the simple UNIX command `rm *`). Therefore the case label is encoded in the buffer file name allowing to delete buffer files automatically when the case in which the file was created and all its sub-cases are solved.

There is much room for experimenting with the place of the two new modules within the priority list.<sup>5</sup> On the one hand, one wants to read and create early, so that the ongoing computation has many equations to

<sup>4</sup> This arrangement is similar to the design of CPU chips that have not only access to their own register memory (~ the equations known within Crack) and access to the hard disk (~ the possibility to call the “probing” module) but also have access to cache memory (~ the buffer file).

<sup>5</sup> Apart from the necessity to give the “reading from the buffer” module a higher priority than the “creating of a buffer” module, in order to try reading and emptying a buffer file first (if available and not already read completely) and to create a new buffer file only if none is available or if the available one is already completely read in.

choose from when looking for the most suitable substitutions, shortenings, .... The problem is that these equations are all generated with the same limited information on relations between  $q_{\mathfrak{g}}$ , and, thus, they have a high redundancy. Also, dealing with many long equations does slow down Crack. On the other hand, giving the branching of the computation into sub-cases a higher priority generates an exponential growth of sub- and sub-sub-cases, which drastically increases the number of buffer files to be generated, because they are only valid for the case they were generated for or for its sub-cases.

With each investigated case, say  $q_5 = 0$ , the other case  $q_5 \neq 0$  generates inequalities, which the package Crack collects, updates, and makes heavily use of to avoid further case distinctions, and in this computation also to drop factors of the  $\hat{P}_{mi}$  as mentioned in Section 5. The individual cases can either be investigated sequentially or in parallel.

If two solutions of two different cases, for example, the solutions for  $q_5 = 0$  and  $q_5 \neq 0$  can be merged into one analytic form, if necessary by a re-parameterization, then this is achieved by one of the modules of Crack [16].

## 7. THE COMPUTATION

The computation was not performed in a single run. Simple subcases were solved initially, whereas harder ones were completed only after the probing technique from Section 5 and its automatic interplay with the package Crack were developed within several months. Even then, it took some time to fine-tune parameters and put the new modules in the right place within the priority list of procedures in Crack. Finally, it was nearly possible to do the computation fully automatically; only a few times, the proper case distinctions had to be initiated manually at the right time to be able to complete the computation. If one would add up purely the necessary computation time without runs following a poor manual choice of case distinctions leading to computations that generated too large systems and that could not be completed, then this would amount to about two weeks of CPU time on a 3 GHz AMD64 PC.

## ACKNOWLEDGMENTS

For this work, facilities of the Shared Hierarchical Academic Research Computing Network (SHARC-NET: [www.sharcnet.ca](http://www.sharcnet.ca)) were used.

TW thanks the Konrad Zuse Institute at Freie Universität Berlin and the Technische Universität Berlin where part of the work was done.

The author also wants to thank A. Bobenko for suggesting this topic, S. Tsarev for the smooth collaboration on the mathematical and computational aspects, and, especially, W. Neun for frequent systems support, which lead to a number of improvements of the

REDUCE computer algebra system as they are listed under “FAQ/Problems” on [10].

#### REFERENCES

1. Sokolov, V.V. and Wolf, T., Integrable Quadratic Hamiltonians on  $so(4)$  and  $so(3,1)$ , *J. Phys. A: Math. Gen.*, 2006, vol. 39, pp. 1915–1926. Also preprint at [arXiv:nlin.SI/0405066](http://arXiv:nlin.SI/0405066).
2. Tsuchida, T. and Wolf, T., Classification of Polynomial Integrable Systems of Mixed Scalar and Vector Evolution Equations. I, *J. Phys. A: Math. Gen.*, 2005, vol. 38, pp. 7691–7733. Also preprint at [arXiv:nlin.SI/0412003](http://arXiv:nlin.SI/0412003).
3. Anco, S. and Wolf, T., Some Symmetry Classifications of Hyperbolic Vector Evolution Equations, *JNMP*, 2005, vol. 12 (supplement 1), pp. 13–31. Also preprint at [arXiv:nlin.SI/0412015](http://arXiv:nlin.SI/0412015).
4. Kiselev, A. and Wolf, T., On Weakly Non-Local, Nilpotent, and Super-Recursion Operators for  $N=1$  Homogeneous Superequations, *Proc. of Dubna Int. Workshop “Supersymmetries and Quantum Symmetries” (SQS’05)*, JINR, 2005, pp. 231–237. Online: <http://theor.jinr.ru/~sqs05/SQS05.pdf>. Also preprint at [arXiv:math-ph/051107](http://arXiv:math-ph/051107).
5. Kiselev, A. and Wolf, T., Supersymmetric Representations and Integrable Super-Extensions of the Burgers and Businesq Equations, *SIGMA*, 2006, no. 2, pp. 19, Paper 030. Also preprint at [arXiv:math-ph/0511071](http://arXiv:math-ph/0511071).
6. Kiselev, A. and Wolf, T., Classification of Integrable Super-systems Using the SsTools Environment, *Comp. Phys. Commun.*, 2007, vol. 177, no. 3, pp. 315–328. Also preprint at [arXiv:nlin.SI/0609065](http://arXiv:nlin.SI/0609065).
7. Tsarev, S.P. and Wolf, T., *Classification of 3-dimensional Integrable Scalar Discrete Equations*, 2007. Preprint at [arXiv:0706.2464](http://arXiv:0706.2464).
8. Vermaseren, J.A.M., *New Features of Form*, 2000. Also preprint at [arXiv:math-ph/0010025](http://arXiv:math-ph/0010025) (a complete distribution can be downloaded from <http://www.nikhef.nl/~form/>).
9. Wolf, T., Applications of Crack in the Classification of Integrable Systems, *CRM Proc. and Lecture Notes*, Montreal, Centre de Recherches Mathematiques, 2004, vol. 37, pp. 283–300. Also preprint at [arXiv:nlin.SI/0301032](http://arXiv:nlin.SI/0301032).
10. Wolf, T., An Online Tutorial for the Package CRACK, 2004, <http://lie.math.brocku.ca/crack/demo>.
11. Adler, V.E., Bobenko, A.I., and Suris, Yu.B., Classification of Integrable Equations on Quad-graphs. The Consistency Approach, *Comm. Math. Phys.*, 2003, vol. 233, pp. 513–543.
12. Adler, V.E., Bobenko, A.I., and Suris, Yu.B., *Discrete Nonlinear Hyperbolic Equations. Classification of Integrable Cases*, 2007. Preprint at [arXiv:nlin.SI/0705.1663](http://arXiv:nlin.SI/0705.1663).
13. Konopelchenko, B.G. and Schief, W.K., Reciprocal Figures, Graphical Statics and Inverse Geometry of the Schwarzian BKP Hierarchy, *Stud. Appl. Math.*, 2002, vol. 109, pp. 89–124. Also preprint at [arXiv:nlin.SI/0107001](http://arXiv:nlin.SI/0107001).
14. Nimmo, J.J.C. and Schief, W.K., An Integrable Discretization of a 2+1-Dimensional Sine-Gordon Equation, *Stud. Appl. Math.*, 1998, vol. 100, pp. 295–309.
15. Wolf, T., Neun, W., and Tsarev, S.P., About Parallelizing the Search for 3-dim. Scalar Discrete Integrable Equations, *Talk given at PASC0*, London, Ontario, 2007. <http://lie.math.brocku.ca/twolf/papers/paratalk.pdf>.
16. Wolf, T., *Merging Solutions of Polynomial Algebraic Systems*, 2003, Preprint <http://lie.math.brocku.ca/twolf/papers/merge-sig.ps>.