

Algorithmic determination of infinite dimensional symmetry groups for integrable systems in 1+1 dimensions*

Benno Fuchssteiner[†], Sergei Ivanov
and Waldemar Wiwianka

Department of Mathematics
University of Paderborn
D 33098 Paderborn, Germany

[†] benno@uni-paderborn.de

September 12, 2007

Abstract

We present algorithms and describe CA-packages to compute the infinitesimal generators of infinite dimensional symmetry groups for integrable PDE's (evolution equations) in one space and one time dimension. Here, integrable is meant in the sense that the vector field defining the equation is a member of the abelian part of some infinite dimensional Virasoro algebra. The method of computation is completely different from the usual prolongation method, no determining equations are solved. Instead, all necessary generators of the finitely generated Virasoro algebra are computed from one given element by direct Lie algebra methods. The implementation of the algorithms in MuPAD is described. A sample session is included in which the recursion structures of the KdV and the Krichever-Novikov equations are computed.

1 Introduction

We are interested in dynamical systems

$$u_t = K(u) \tag{1.1}$$

for $u = u(x, t)$ in some suitable manifold of functions. We assume that all vector and tensor fields are C^∞ with respect to some suitable differential structure on the manifold.

We search for the generators of one-parameter symmetry groups and, when there are infinitely many, we want to find their recursion formulas. So what is required is the determination of $K^\perp := \{G \mid [G, K] = 0\}$, the commutant of the vector field K with respect to the vector field Lie algebra. We are mainly interested in integrable systems. By this we mean that in the vector field Lie

*Supported in part by SFB 376 of the Deutsche Forschungsgemeinschaft

algebra there is an infinite dimensional sub-Lie algebra spanned by elements $\{K_n, \tau_n \mid n \in \mathbb{N}\}$ with the following commutation relations:

$$\begin{aligned} [K_n, K_m] &= 0 \\ [\tau_n, K_m] &= (m + \rho) K_{m+n} \\ [\tau_n, \tau_m] &= (m - n) \tau_{n+m}. \end{aligned}$$

The scalar factor ρ describes the different scaling behaviour of the subalgebra under consideration. In particular, K in (1.1) is in the *abelian part* of that Lie algebra spanned by the elements K_n . Note that the relations above define that part of a Virasoro algebra (see for example [2] for the definition) with positive index. For simplicity, this positive part we also call a Virasoro algebra.

There are many different ways for the construction of the set of symmetry group generators such as hereditary operators, inverse scattering methods, Miura transformations or mastersymmetries. For computational purposes we choose the mastersymmetry approach, which is just the same as constructing the generators of the corresponding Virasoro algebra. Mastersymmetries exist for all known completely integrable systems¹, which is no surprise since they are closely connected to the existence of angle variables, at least on finite dimensional invariant submanifolds (see for example [7] and [12]). So mastersymmetries, or Virasoro algebras, seem to be the most general approach, they work in a simple and transparent way even when recursion operators do not exist, for example for the KP or the Benjamin-Ono equation [4]. Furthermore, from the computational point of view, mastersymmetries are simple to handle and give rise to elementary and efficient algorithms.

Other methods to describe the recursive structure of large abelian symmetry groups for integrable systems, like recursion operators, are easily retrieved from known mastersymmetries, at least for Hamiltonian systems: If the mastersymmetry is non-Hamiltonian, then the Lie derivative of the Hamiltonian operator (implectic form) with respect to that mastersymmetry is nonzero and gives a second invariant operator of the same kind. Combining it with the inverse of the Hamiltonian operator gives a recursion operator (see [5] for proofs). Indeed, all known recursion operators, as well as second Hamiltonian formulations, are of that nature. If the mastersymmetry is Hamiltonian, then generally recursion operators are not known².

The paper is organized in the following way: In the next section we justify, in the context of structure theory of integrable nonlinear systems, the assumptions which form the basis for our algorithms. Then the algorithm and its main ingredients are described. After that its implementation in MuPAD together with a sample session is presented.

This paper continues work of W. W. and B. F. ([14], [15]), which was interrupted for two years by the tragic death of Waldemar Wiwianka. The MuPAD

¹The only exception seems to be the quantum mechanical Hubbard model [10]. However, we believe that even there mastersymmetries of an extended type do exist.

²Recursion operators of a more general type do exist [3]. They can be retrieved from a correspondingly generalized type of mastersymmetries.

version, together with considerable improvements with respect to data structure and efficiency are due to S. I. Due to the object oriented nature of MuPAD some of the essential routines of the present package can be implemented such that they increase speed and efficiency, furthermore, MuPAD's parallel features will eventually allow to treat applications of a more demanding nature. In the long run, the present algorithms will be integrated in a tensor calculus package for infinite dimensional manifolds.

Other work, related to the algorithms presented here, was carried out in Paderborn. Our group has developed and implemented similar algorithms (based on MAPLE at that time) for the case of lattice-systems and for quantum mechanical spin- $\frac{1}{2}$ -chains (see [6], [13], [16], [11]). These algorithms are under revision and will eventually be improved and implemented in MuPAD. For the treatment of more demanding problems these algorithms will be also implemented in the parallel versions of MuPAD. MuPAD (Multi Processing Algebra Datatool) ([8], [9], [1]) is developed as a service to the scientific community and is distributed to Universities free of charge. Information is available on WWW under <http://math-www.uni-paderborn.de/MuPAD/>.

2 Basic assumption

The essential part of the package described below is the algorithmic solution of the division problem in the Lie algebra under consideration. As a result, for evolutionary systems in such algebras a symmetry can be computed if its highest order term is known. This computational method turns out to be particularly useful in cases of translation invariant integrable systems. For this case we present the necessary arguments in greater detail and express their algebraic structure in form of a basic assumption.

The **basic assumption**³ will be that the vector field $K(u)$ of the flow (1.1) is in the abelian part $Linearhull\{K_n \mid n \in \mathbb{N}\}$ of a Virasoro algebra. Further, it is assumed that its commutant K^\perp (restricted to a suitable type of vector fields) also is a subset of that abelian part. The elements of the non abelian part $Linearhull\{\tau_n \mid n \in \mathbb{N}\}$ shall be called *mastersymmetries*. The Lie algebra generated by all $\{\tau_n, K_n \mid n \in \mathbb{N}\}$ is denoted by $\mathcal{H}er := \mathcal{H}er(\tau_n, K_n \mid n \in \mathbb{N})$. It is a hereditary algebra [5].

For further simplification we treat only one-component systems of integro-differential equations of polynomial or rational type in the dependent variable u . The independent variable is denoted by x .

We consider $K(u)$ in the algebra $\mathcal{A}_{TI} := \mathcal{A}(D, D^{-1}, u)$ generated by D (differentiation with respect to the independent variable x), D^{-1} (inverse of D) and u (the field variable). The integration D^{-1} is considered in a formal sense, i.e., in this paper we do not discuss under which conditions (mostly at infinity) integrals over lines or half lines exist. So all our computations with derivatives

³As mentioned before, our programs also give relevant results in situations, where this basic assumption does not hold.

and integrals are performed in a formal way (this can be justified in terms of the density notion introduced in [4]).

In order to carry out more sophisticated computations we introduce the algebra $\mathcal{A} := \mathcal{A}(x, D, D^{-1}, u)$ generated by $\mathcal{A}(D, D^{-1}, u)$ and by multiplication with the independent variable x . Members of this Lie algebra are called *vector fields*.

We remark, that $K(u)$ is assumed to be translation invariant, i.e., it commutes with u_x , the generator of the translation group. Since u_x is a symmetry generator, it must be in the abelian part. Mastersymmetries, which in general do not commute with any of the symmetries, cannot be translation invariant. Hence, they are elements in \mathcal{A} . However, their commutator with u_x must be in the abelian part, which is a subspace of \mathcal{A}_{TI} . So we arrive at the crucial

Ansatz: All mastersymmetries are of the form $M = xG + Z$ with some expressions $G, Z \in \mathcal{A}_{TI}$. The field $G = [u_x, M]$ must be a symmetry, whereas Z is some unknown translation invariant vector field.

In case of several independent variables x_1, \dots, x_n the expression u_x has to be replaced by the generators u_{x_1}, \dots, u_{x_n} , i.e., $M = \sum_j x_j G_j + Z$, where each G_j is a linear combination of the commuting symmetries spanning the abelian part of the Virasoro algebra. However, in this case the normal form algorithms become more difficult, especially the one for the inverse of differentiation.

3 The essential parts of the algorithm

The algorithm consists of two main ingredients which are complemented by a projection method in order to reduce complexity and speed up the programs. One is a normal form algorithm for vector fields, in particular a procedure for normalizing nested terms containing the operator D^{-1} . This is nontrivial, because via integration by parts certain terms may be written in different ways and no *obvious* normal form is available. The second essential algorithm is a solver in the Lie algebra of vector fields, again nontrivial because of the many zero divisors in that algebra. A strategy is needed to pick out certain solutions from the (usually) infinite solution set. This is done by means of a grading, which reduces the division problem to a finite dimensional one and also serves for splitting the problem into a series of simpler approximate tasks.

3.1 Highest order terms

For the algorithmic search of symmetries the *highest-order-term projection* is an essential tool. It is based on a grading which should be thought of as a grading with respect to the highest x -derivatives. Actually, it is a little bit more subtle than that, because integration is an admissible operation. This results in terms

which may be nested in a complicated way. More details go beyond the scope of this paper, so it may be sufficient at this point to understand the basic principle that differentiation increases the grade. The grading introduces an order \prec among the elements of $\mathcal{A} = \mathcal{A}(x, D, D^{-1}, u)$ having the following properties:

1. For every fixed $A \in \mathcal{A}$ any intersection of the space $\{B \mid B \prec A\}$ with the polynomials in u up to some fixed degree is finite-dimensional.
2. The order is semi-compatible with the Lie algebra structure in \mathcal{A} , i.e., whenever $[A, X] = B$ then there is some $\tilde{X} \prec B$ such that $X - \tilde{X} \in A^\perp$.
3. \prec is a linear order on $\mathcal{M} \subset \mathcal{A}$, where \mathcal{M} is the set generated by x, D, D^{-1} and u . Multiplication is the only algebraic operation (no addition). So, \mathcal{A} is the set of sums over elements from \mathcal{M} .

Once a suitable order is found, a solution X of the Lie algebra *division problem*

$$\text{given } A, B, \text{ find } X \text{ solving } [A, X] = B \quad (3.1)$$

can be found among the elements of order less than some C with $A, B \prec C$. Working with a basis of the *finite dimensional* linear space given by the intersection $\{Y \mid Y \prec C\}$ with suitable polynomials in u , this is achieved algorithmically by means of elementary linear algebra (assuming that such a solution exists). However, such an algorithm based on solving linear equations is much too slow to yield an efficient tool. Therefore the division problem is replaced by an *approximate division problem*. In order to do that we have introduced the *highest-order-projection* $\text{hop}(\mathbf{A})$ mapping every sum $A = A_1 + A_2 + \dots + A_n$ in \mathcal{A} onto its maximal summand A_k in \mathcal{M} , i.e., $A_j \prec A_k$ for all $j \neq k$.

The order relation sketched in this section also is the basis for the normal form algorithm for the nested D^{-1} -terms. Here the basic principle is that integration by parts is performed, whenever the order can be decreased by that operation.

To give a better understanding of the order structure under consideration we present some elementary examples:

$$\begin{aligned} u_{xx} &\prec u_{xxx} + u_x \\ u_{xx}u_{xx} &\prec u_{xxx} \\ u_{xxx}u_x &\prec u_{xxx}u_{xx} \\ D^{-1}(u_{xx}u) &\prec u_{xx} \\ u_xu &\prec D^{-1}(u_{xx}u) \end{aligned}$$

and so on. These examples come from a set of rules which comprise: The highest order is determined by the highest derivative occurring in the expression. If these highest derivatives are equal then the remaining factors of the term with the highest derivative are considered. Integration counts as a negative derivative with respect to its integrands, however not as much as if it were really carried out. As a result of such rules in $D^{-1}(u_{xx}u)$ an integration by parts is performed since the result $u_xu - D^{-1}(u_xu_x)$ is of lower order.

3.2 The solver

Instead of solving (3.1) we first solve the following approximate division problem: Given A, B , find X such that

$$\text{hop}([\text{hop}(A), \text{hop}(X)]) = \text{hop}(B).$$

The sub-routine that does the job is called $\text{cs}(B, A)$, its efficiency is the crucial point of the algorithm since a great number of special cases has to be considered. Basically it could be realized by linear algebra alone. However, in order to speed up things, its implementation uses a lot of expertise on the structure of commutators in the algebra under consideration. Once we have this routine, it is easy to find symmetries and mastersymmetries.

4 The algorithm

We start with finding a symmetry G for the vector field K , i.e., $[G, K] = 0$, where G and K are both assumed to be translation invariant. We may choose $G = K$, if no further symmetry is known. There is a mastersymmetry M with $[u_x, M] = G$, hence M is of the form $M = xG + Z$ with some translation invariant Z .

Since M is a mastersymmetry, the commutator with K must be another symmetry $K_1 = [M, K]$. A crucial property of the order relation is that the highest-order-term of $[M, K]$ is of the same order as that of $[xG, K]$. Therefore, the known term G yields the highest-order-term of the commutator $[M, K]$ (up to multiplication by a suitable factor). Thus the highest-order-term of the unknown mastersymmetry M as well as the corresponding term in K_1 is independent of the unknown quantity Z . Therefore, specifying G to fix the highest-order-term of the unknown M , we may construct the new symmetry $K_1 = \text{symmetry}(K, G)$ directly by calling the sub-routine symmetry of our program package which is based on the following

Algorithm	symmetry
Call	$\text{symmetry}(K, G)$
Step 0	$K_1 := [xG, K]$
Step 1	$P := [K_1, K]$ <i>if</i> $P \neq 0$ <i>then</i> Goto Step 2 <i>else</i> Output $\rightarrow \text{symmetry}(K, G) = K_1$
Step 2	$X := \text{cs}(P, K)$ <i>if</i> there is no solution <i>then</i> Output \rightarrow <i>no solution</i> <i>else</i> Goto Step 3
Step 3	$K_1 := K_1 - X$ Goto Step 1

Let us point out again that the algorithm is successful even if no symmetry G is known, because at the beginning of the computation we can choose $G = K$. Further symmetries are computed recursively from previously found generators.

However, the implementation of this routine is not really necessary, since it is a special case of a more general algorithm. We only presented it because the method of the approximate solution of the division problem is more transparent in **symmetry** than in the more general procedure called **gho** (abbreviation for *given highest order*).

Algorithm	gho
Call	gho(E, K, X)
Step 0	$S := X$
Step 1	$P := [S, E]$ if $P \neq K$ then Goto Step 2 else Output \rightarrow gho(E, K, X) = S
Step 2	$X := \text{cs}(\text{hop}(E), \text{hop}(P - K))$ if there is no solution then Output \rightarrow no solution else Goto Step 3
Step 3	$S := S - X$
	Goto Step 1

The procedure **cs**() denotes the crucial algorithm which produces the solution of our approximate division problem. For given E, K and X this algorithm produces, if existent, the solution Y of $[Y, E] = K$ with $\text{hop}(Y) = \text{hop}(X)$.

Obviously, given some symmetry G , we obtain another symmetry by

$$\text{symmetry}(K, G) = \text{gho}(K, 0, [K, x * G])$$

and the corresponding mastersymmetry by the call

$$\text{mastersymmetry}(K, G) = \text{gho}(K, \text{symmetry}(K, G), x * G).$$

At the beginning, when a second symmetry G is not yet known, we use K instead. One should observe that the computation of a symmetry does not require the computation of a mastersymmetry first. That observation is helpful for those cases where the normal form procedures are not yet able to perform normalization in the algebraic structure where the mastersymmetry is to be expected, but are fully able to normalize the resulting symmetries (see, e.g., the Novikov-Krichever equation [2] used in the sample session of section 5).

4.1 An application to nonintegrable systems

One should observe that by use of the package, or by the Lie algebra solver given in it, also time-dependent symmetries for integrable systems and Lie-point symmetries for nonintegrable systems may be computed. One has to

recall [4] that whenever $M(u)$ is a mastersymmetry (of first degree, no others are considered in this paper), then

$$T(u, t) := M(u) + t[M(u), K(u)]$$

is a time-dependent symmetry for $u_t = K(u)$, i.e., it fulfills

$$\frac{\partial}{\partial t}T(u, t) + [K(u), T(u, t)] = 0.$$

Hence, whenever a mastersymmetry is found, we obtain a time dependent symmetry out of it by one further commutation.

Usually for nonintegrable systems `symmetry(K, u_x)` exists, so we may use `mastersymmetry(K, u_x)` to find the corresponding time-dependent symmetry which usually is a Lie-point symmetry.

5 Sample session

The programs are implemented in MuPAD, a computer algebra system developed at the Automath Institute of the University of Paderborn. The package will be contained in the next release of MuPAD. Eventually it will be integrated in a more comprehensive package for computations with tensor bundles on infinite dimensional manifolds.

First a remark on the notation: throughout the following `u_(.1,n)` denotes the n -th derivative of $u = u_(.1,0)$ with respect to the independent variable x . Although only one-component field functions are considered the notation uses a field index (the first argument of `u`) for extensions of the package to several components. The functions `symmetry` and `mastersymmetry` are those described above, the function `gho` is accessible by the user. It is called by the routines `symmetry` and `mastersymmetry`. Comments, not produced by the MuPAD system, will start with `comment:.` Input given by the user and output coming from the system start with `input>` and `output>`, respectively. Operations like loading libraries and exportation of domains are suppressed, for these the reader is referred to the MuPAD documentation [8], [9]. Throughout the sample session we use both the pretty-print mode and the line-print mode.

```
comment: First, we define the right hand side of the KdV:

input > K1:=u_(.1, 3) + u_(.1, 0)*u_(.1, 1)*6;

comment: We call the procedure "symmetry" with arguments
comment: given by the known symmetry (the KdV itself):

input > G2:=symmetry(K1,K1);

comment: Indeed, as output the known first symmetry is obtained:

output> u_(.1, 5)*(-3) + u_(.1, 0)*u_(.1, 3)*(-30)
```

```

output> + u(_1, 1)*u(_1, 2)*(-60)
output> + u(_1, 0)^2*u(_1, 1)*(-90)

comment: We check its correctness by computing the commutator:

input > commutator(K1,G2);

output> 0

comment: We look for the first nontrivial mastersymmetry:

input > M1:=mastersymmetry(K1,K1);

output> u(_1, 2)*4 + x*u(_1, 3) + x*u(_1, 0)*u(_1, 1)*6
output> + u(_1, 1)*J(u(_1, 0), x)*2 + u(_1, 0)^2*8

comment: J represents the integration D^{-1}.
comment: Commuting M1 with K1 must give a symmetry:

input > commutator(K1,M1);

output> u(_1, 5)*(-3) + u(_1, 0)*u(_1, 3)*(-30)
output> + u(_1, 1)*u(_1, 2)*(-60) + u(_1, 0)^2*u(_1, 1)*(-90)

comment: Indeed, this is the same as G2. The second mastersymmetry:

input > M2:=mastersymmetry(K1,G2);

output> u(_1, 4)*(-18) + x*u(_1, 5)*(-3)
output> + u(_1, 0)*u(_1, 2)*(-144) + x*u(_1, 0)*u(_1, 3)*(-30)
output> + x*u(_1, 1)*u(_1, 2)*(-60)
output> + u(_1, 3)*J(u(_1, 0), x)*(-6)
output> + u(_1, 0)^3*(-96) + u(_1, 1)^2*(-108)
output> + u(_1, 0)*u(_1, 1)*J(u(_1, 0), x)*(-36)
output> + x*u(_1, 0)^2*u(_1, 1)*(-90)
output> + u(_1, 1)*J(u(_1, 0)^2, x)*(-18)

comment: Its commutator with K1 yields the 7-th order symmetry:

input > commutator(K1,M2);

output> u(_1, 7)*9 + u(_1, 0)*u(_1, 5)*126
output> + u(_1, 1)*u(_1, 4)*378 + u(_1, 2)*u(_1, 3)*630
output> + u(_1, 1)^3*630 + u(_1, 0)*u(_1, 1)*u(_1, 2)*2520
output> + u(_1, 0)^3*u(_1, 1)*1260 + u(_1, 0)^2*u(_1, 3)*630

comment: Commuting this last expression with K1 checks its invariance:

input > commutator(%,K1);

```

output> 0

comment: Let us turn our attention to the Krichever-Novikov equation
comment: which is interesting insofar as the normalization procedure
comment: is not yet powerful enough to perform normalization in the
comment: algebraic structure where the mastersymmetry is to be
comment: expected. However, this does not prevent us from computing
comment: the symmetries. We start by defining K1 to be:

input > K1:= u(_1, 3) + 1/u(_1, 1)*u(_1, 2)^2*(-3/2)

comment: For the output now the pretty-printer is used:

output>
output>
$$u_{(-1, 3)} - \frac{3 u_{(-1, 2)}^2}{2 u_{(-1, 1)}}$$

comment: We find the first non-trivial symmetry by:

input > G2:=symmetry(K1,K1);

output>
output>
$$- 3 u_{(-1, 5)} + \frac{15 u_{(-1, 2)} u_{(-1, 4)} u_{(-1, 3)}^2}{u_{(-1, 1)}} + \frac{15 u_{(-1, 3)}^2}{2 u_{(-1, 1)}}$$

output>
output>
$$+ \frac{135 u_{(-1, 2)}^4}{8 u_{(-1, 1)}^3} - \frac{75 u_{(-1, 2)}^2 u_{(-1, 3)}^2}{2 u_{(-1, 1)}^2}$$

comment: The second symmetry is a bit more involved:

input > G3:=symmetry(K1,G2);

output>
output>
$$9 u_{(-1, 7)} - \frac{63 u_{(-1, 2)} u_{(-1, 6)}}{u_{(-1, 1)}}$$

output>
output>
$$- \frac{126 u_{(-1, 3)} u_{(-1, 5)}}{u_{(-1, 1)}} - \frac{189 u_{(-1, 4)}^2}{2 u_{(-1, 1)}}$$

output>
output>
$$+ \frac{441 u_{(-1, 3)}^3}{2 u_{(-1, 1)}^2} - \frac{14175 u_{(-1, 2)}^6}{16 u_{(-1, 1)}^5}$$

```

output>
output>      2
output>      1071 u_(1, 2) u_(1, 3) u_(1, 4)  567 u_(1, 2)  u_(1, 5)
output>      ----- + -----
output>      2      2
output>      u_(1, 1)      2 u_(1, 1)
output>
output>      2      3
output>      567 u_(1, 2)  u_(1, 5)  2079 u_(1, 2)  u_(1, 4)
output>      + ----- - -----
output>      2      3
output>      2 u_(1, 1)      2 u_(1,1)
output>
output>      4      2      2
output>      22491 u_(1, 2)  u_(1, 3)  8757 u_(1, 2)  u_(1, 3)
output>      + ----- - -----
output>      4      3
output>      8 u_(1, 1)      4 u_(1, 1)

```

comment: Let us compute a Lie point symmetry of an equation which
comment: certainly is non-integrable. We define K1 to be:

```
input > K1:=u_(1, 3) + u_(1, 0)^3*u_(1, 1)*6 ;
```

```
output> u_(1, 3) + u_(1, 0)^3*u_(1, 1)*6
```

comment: The lowest mastersymmetry is obtained by:

```
input > M0:=mastersymmetry(K1,u_(1, 1));
```

```
output> u_(1, 0)*2/3 + x*u_(1, 1)
```

comment: Now, we generate the time-dependent part by taking the
comment: commutator with K1 and obtain a Lie point symmetry of
comment: that equation:

```
input > M0+t*commutator(M0,K1);
```

```
output> u_(1, 0)*2/3 + x*u_(1, 1) + t*(u_(1, 3)*3
output> + u_(1, 0)^3*u_(1, 1)*18)
```

References

- [1] *Computeralgebra in Deutschland - Bestandsaufnahme, Möglichkeiten, Perspektiven*, Herausgegeben von der Fachgruppe Computeralgebra der GI, DMV und GAMM, Passau und Heidelberg, 1993.
- [2] I. Dorfman, *Dirac Structures and Integrability of Nonlinear Equations*, Nonlinear Science: Theory and Applications, John Wiley and Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1993.

- [3] A.S. Fokas and P.M. Santini, The recursion operator of the Kadomtsev-Petviashvili equation and the squared eigenfunctions of the Schrödinger operator, *Studies in Appl. Math.*, 75:179–186, 1986.
- [4] B. Fuchssteiner, Mastersymmetries, higher-order time-dependent symmetries and conserved densities of nonlinear evolution equations, *Progr. Theor. Phys.*, 70:1508–1522, 1983.
- [5] B. Fuchssteiner. Hamiltonian structure and integrability, in: *Nonlinear Equations in the Applied Sciences, Mathematics in Science and Engineering Vol. 185*, C. Rogers and W.F. Ames (eds.), Academic Press, Boston - San Diego - New York - London - Sydney - Tokyo - Toronto, pages 211–256, 1991.
- [6] B. Fuchssteiner and U. Falck, Computer algorithms for the detection of completely integrable quantum spin chains, in: *Symmetries and nonlinear phenomena*, D. Levi and P. Winternitz (eds.), World Scientific Publishers, Singapore, pages 22–50, 1988.
- [7] B. Fuchssteiner and G. Oevel, Geometry and action-angle variables of multisoliton systems, *Rev. Math. Phys.*, 1:415–479, 1990.
- [8] K. Gottheil, A. Kemper, O. Kluge, K. Morisse, H. Naundorf, G. Oevel, T. Schulze, W. Wiwianka and B. Fuchssteiner, *MuPAD Benutzerhandbuch*, Birkhäuser Verlag, Basel, 1993.
- [9] K. Gottheil, A. Kemper, O. Kluge, K. Morisse, H. Naundorf, G. Oevel, T. Schulze, W. Wiwianka and B. Fuchssteiner, *MuPAD Multi Processing Algebra Data Tool Version 1.2: Tutorial*, Birkhäuser Verlag, Basel, 1994.
- [10] M.P. Grabowski and P. Mathieu, Structure of the conservation laws in quantum integrable spin chains with short range interaction, *preprint LAVAL-PHY-21/94*, Université Laval, Quebec, 1994.
- [11] W. Oevel, H. Zhang, G. Tu and B. Fuchssteiner, Symmetries, conserved quantities and hierarchies for some lattice systems with soliton structure, *J. Math. Phys.*, 32:1908–1918, 1991.
- [12] G. Oevel and B. Fuchssteiner, Unified approach to action-angle representation of real and complex multisolitons, *Physica*, A 181:364–384, 1992.
- [13] W. Oevel, H. Zhang, and B. Fuchssteiner, Mastersymmetries and multi-hamiltonian formulations for some integrable lattice systems, *Progr. Theor. Phys.*, 81:294–308, 1989.
- [14] W. Wiwianka and B. Fuchssteiner, Algorithms to detect complete integrability in 1+1 - dimension, in: *Research Reports in Physics - Nonlinear Dynamics*, S. Carillo and O. Ragnisco (eds.), Springer Verlag, Berlin-Heidelberg-New York, pages 131–135, 1990.

- [15] W. Wiwianka, Algorithmen zur rekursiven Berechnung von Lie-Bäcklund Symmetrien nichtlinearer partieller Differentialgleichungen, *Dissertation*, Paderborn 1990.
- [16] H. Zhang, Computeralgebra-Algorithmen zur Bestimmung der vollständigen Integrabilität der Dynamik auf nichtlinearen Gittersystemen, *Dissertation*, Paderborn, 1990.