# Pseudo Differential Operators and Integrable Systems in AXIOM

*Werner M. Seiler**

Institut für Algorithmen und Kognitive Systeme
Universität Karlsruhe, D-76128 Karlsruhe, Germany
Email: kg04@dkauni2.bitnet

**Abstract**

An implementation of the algebra of pseudo differential operators in the computer algebra system AXIOM is described. In several examples the application of the package to typical computations in the theory of integrable systems is demonstrated.

## Program Summary

*Title of program:* PDO

*Computer:* IBM RS 6000, SUN

*Operating system:* AIX, SunOS

*Programming language:* AXIOM 1.2

*Memory required:* Depends on complexity of calculation

*No. of lines:* ca. 1800

*Keywords:* Pseudo differential operators, integrable systems, (generalized) Lax equation, Zakharov-Shabat equation, hierarchies, conserved densities

*Nature of physical problem:* Generation of integrable systems solvable by the Inverse Scattering Method and determination of conservation laws

*Method of solution:* Pseudo differential operators allow one to construct easily hierarchies of Lax pairs. The formalism yields also automatically an infinite number of conserved densities.

*Running times:* Depends on complexity of calculation

---

* Supported by Studienstiftung des deutschen Volkes

# 1 Introduction

Pseudo differential operators (sometimes also called micro differential operators) are a useful tool in the theory of integrable system [1]. It is easily possible to construct with their help whole hierarchies of integrable systems in Lax form starting with a linear differential operator. The problem here is to find possible partners for this operator to build a Lax pair [2].

It is interesting that this problem was already tackled by Schur [3] in 1904, however, not in the context of integrable systems. He was concerned with the question, when two linear differential operators commute, and was able to give a complete answer by introducing pseudo differential operators.

The original approach to integrable systems with pseudo differential operators [4] was restricted to systems in 1+1 dimensions. Using Sato theory [5] one can, however, also work in higher dimensions and derive similiarly hierarchies. In a recent paper, Oevel [6] showed, how pseudo differential operators can also be applied to Darboux transformations. We will, however, consider in this paper only Lax und Zakharov-Shabat equations.

Pseudo differential operators are meanwhile a standard technique in integrable system theory. Some operations like division or extraction of a root are, although straight forward, fairly tedious. It is thus natural to implement them in a computer algebra system. We have chosen AXIOM [7] because of its convenient stream data types which allow for the efficient manipulation of infinite series and for the possibility of generic programming.

Most implementations of infinite series are based on a truncation at a given, fixed order. In a stream the coefficients are also only computed up to a given order. A stream "knows", however, a rule to calculate additional terms, if they are needed. This so-called lazy evaluation is especially useful, if it is a priori not clear, how many terms will be needed. This is for instance the case, if leading terms may cancel.

Ito [8] presented a REDUCE program for the evaluation of Lax pairs. His pairs consisted of matrices of differential operators and the purpose of his program was to compute the commutator of generic matrices in order to be able to guess the Lax pair for a given equation. Our program can also handle easily such calculations, but it is not its main goal. The idea behind the use of pseudo differential operators is to *derive* operators which form together with a given one Lax pairs. Hence our program is much more general than the one of Ito.

This paper is organized as follows: The next section introduces pseudo differential operators and their algebraic properties. Section 3 explains their application to integrable systems. After a discussion of the implementation in Section 4, the last section shows some concrete examples, namely the Korteweg-de Vries and the Boussinesq hierarchy in 1+1 dimensions and the Kadomtsev-Petviashvili hierarchy in 2+1 dimensions.

## 2 Pseudo Differential Operators

We denote by $D$ the differentiation operator $\partial/\partial x$ with respect to the variable $x$. $D$ can be formally defined through the commutation relation

$$[D, f(x)] = f'(x)\,. \tag{1}$$

We can introduce, again formally, its inverse operator via the corresponding commutation relation

$$[D^{-1}, f(x)] = \sum_{k=1}^{\infty} (-1)^k f^{(k)}(x) D^{-k-1}\,. \tag{2}$$

Although this definition does not fix $D^{-1}$ completely, it suffices completely for all purely algebraic computations.

We define a *pseudo differential operator* of order $n$ as a formal series

$$F = \sum_{k=-\infty}^{n} f_k D^k \tag{3}$$

where the coefficients $f_k$ are arbitrary (smooth) functions out of some ring. The pseudo differential operators together with the usual addition and multiplication of operators form a non-commutative algebra over the coefficient ring [3].

For the coefficients of a product one can derive a similiar formula as for the product of two power series. If

$$F = \sum_{k=-\infty}^{n} f_k D^k\,,\quad G = \sum_{k=-\infty}^{m} g_k D^k\,, \tag{4}$$

then the product is given by

$$F \cdot G = \sum_{k=0}^{\infty} \left[ \sum_{i=0}^{k} \sum_{j=0}^{k-i} \binom{n-j}{k-i-j} f_j g_i^{(k-i-j)} \right] D^{m+n-k}\,. \tag{5}$$

One reason for the introduction of the negative powers in (3) lies in the fact that (left and right) division by a pseudo differential operator can be defined, if division in the coefficient ring is possible. A closer analysis of the product (5) reveals that the equation $H = F \cdot G$ leads to a linear system of equations for the coefficients of $F$ (or $G$), if $H$ and $G$ (or $H$ and $F$) are given. Thus they are rational functions of the coefficients of the two given operators.

Similiarly, it is possible to take the $k$-th root of a pseudo differential operator provided that $k$ divides the order $n$ of the operator. If we make the ansatz $G^k = F$ where $G$ is of order $m = n/k$, then we get for the leading coefficient $g_m^k = f_n$, i.e. it is determined only up to a $k$-th root of unity. The remaining coefficients are again given by linear systems of equations. This means that there exists $k$ different roots related by multiplication with a primitive $k$-th root of unity.

Finally, we introduce the *residue* of a pseudo differential operator

$$\mathrm{Res}\,(F) = f_{-1} \tag{6}$$

and its *positive* and *negative part*, respectively,

$$F_+ = \sum_{k=0}^{n} f_k D^k\,, \tag{7}$$

$$F_- = \sum_{k=-\infty}^{-1} f_k D^k\,. \tag{8}$$

$F_+$ is a usual linear differential operator. If one assumes that the coefficients tend sufficiently fast to zero as $|x| \to \infty$, one can show that

$$\int_{-\infty}^{\infty} dx\, \mathrm{Res}\,[F, G] = 0 \tag{9}$$

for arbitrary pseudo differential operators $F, G$.

## 3 Integrable Systems

One interesting application of pseudo differential operators is the construction of Lax pairs [2] and thus of completely integrable systems solvable by the Inverse Scattering Method. Furthermore, this approach yields conservation laws of the equation. For simplicity, we start with systems in 1+1 dimension.

The fundamental idea of Lax pairs is to associate with each function $u(t, x)$ a differential operator $L[u]$ such that if $u$ is a solution of the (nonlinear) evolution equation

$$u_t = K[u]\,, \tag{10}$$

the operators $L(t)$ ($L[u]$ at different times $t$) are unitarily equivalent. Then the eigenvalues of $L(t)$ are integrals of the equation (10).

Unitary equivalence means that there exists a family $U(t)$ of unitary operators with the properties

$$U_t = M \cdot U\,, \tag{11}$$

$$U(t) \cdot U(t)^\dagger = \mathbb{1}\,, \tag{12}$$

$$U(t)^{-1} \cdot L(t) \cdot U(t) = L(0)\,, \tag{13}$$

where $M$ is a skew-adjoint operator, i.e. $M^\dagger = -M$, which usually also depends on $u$. Note that $U$ has a *linear* evolution law (11).

Differentiating (13) with respect to $t$ and substituting (11) yields the famous *Lax equation*

$$\frac{\partial L}{\partial t} = [M, L]\,. \tag{14}$$

$L, M$ are called the Lax pair for the evolution equation (10).

There exists, however, no general method to construct the Lax pair to a given evolution equation. Conversely, it is not possible to choose any pair of operators as Lax pair. Pseudo differential operators provide a simple method to generate whole hierarchies of integrable systems starting with a linear differential operator $L$.

Schur [3] showed that every pseudo differential operator $M$ commuting with a given linear differential operator $L$ of order $n$ can be written as

$$M = f + \sum_{k=1}^{m} c_k L^{k/n} \tag{15}$$

where $f$ is an arbitrary function and the $c_k$ arbitrary constants. Furthermore, it is easy to see that for any $M$ of this form $\mathrm{order}\big([M_+, L]\big) \leq n$ holds.

Hence we can use these operators as partner of $L$ in Lax pairs to define a hierarchy of Lax equations through

$$\frac{\partial L}{\partial t} = \left[(L^{k/n})_+, L\right], \quad k = 1, 2, \ldots \tag{16}$$

The corresponding flows are called *Gelfand-Dikii flows*. Since from (16) follows $\partial/\partial t\,(L^{j/n}) = \left[(L^{k/n})_+, L^{j/n}\right]$ for all $j$, we can use (9) and find for this hierarchy an infinite number of conserved quantities, namely

$$Q_j = \int_{-\infty}^{\infty} dx\, \mathrm{Res}\, L^{j/n}\,, \qquad j = 1, 2, \ldots \tag{17}$$

It is possible to extend this approach to integrable systems in more than 1+1 dimensions using Sato theory [5]. We introduce the "dressing" operator

$$P = 1 + w_{-1}D^{-1} + w_{-2}D^{-2} + \cdots \tag{18}$$

and the differential operators

$$C = \gamma D\,,\ A_n = \alpha_n D^n\,,\ n \in \mathbb{N}\,. \tag{19}$$

The coefficients $w_i$ will be the fields of the theory satisfying nonlinear evolution equations. They depend on an infinite number of "time variables" $t_k$ . $\gamma, \alpha_n$ are arbitrary but fixed functions independent of the $t_k$.

We assign now an evolution to the fields $w_i$ by requiring

$$\frac{\partial P}{\partial t_n} = -\left(PA_nP^{-1}\right)_-\,P\,. \tag{20}$$

Since we have on both sides of this equation a pseudo differential operator of order $-1$, this yields a consistent dynamic. Introducing the operators

$$L = PCP^{-1}\,, \tag{21}$$

$$B_n = \left(PA_nP^{-1}\right)_+ \tag{22}$$

we derive the *generalized Lax equation*

$$\frac{\partial L}{\partial t_n} = [B_n, L] \tag{23}$$

and as its compatibility conditions the *Zakharov-Shabat equation*

$$\frac{\partial B_m}{\partial t_n} - \frac{\partial B_n}{\partial t_m} = [B_n, B_m] \; . \tag{24}$$

## 4 Implementation in AXIOM

For an implementation of pseudo differential operators, we have chosen the computer algebra system AXIOM [7] (formerly known as Scratchpad II). It provides a very powerful, object-oriented programming language which allows one to introduce easily abstract data types for the representation of algebraic structures.

For infinite series like (3), the stream types [9] with their lazy evaluation mechanism proved to be very useful. For a stream, it is not necessary to define a cut-off order. Every operation is done only up to a prescribed order, which can be changed any time. If suddenly more terms are needed, they are automatically computed, as a stream "knows" a rule to determine its next term.

Several domains and packages have been developed for pseudo differential operators and linear differential operators. There exists also a package for their application to integrable systems as indicated in Section 3. Especially for Sato theory, we have added a domain with functions depending on an infinite number of variables.

The fundamental domain is `PseudoDifferentialOperator`. It takes as parameters the coefficient domain, a `PartialDifferentialRing`, the name of the differential operator and the name of the variable with respect to which this operator differentiates. It implements besides some procedures to generate pseudo differential operators the basic algebraic operations of a `PartialDifferential-Ring`.

A pseudo differential operator is internally represented as a record with three slots: the coefficient stream, the highest and the lowest occuring exponent. The latter one is minus infinity in the generic case (AXIOM provides the data type `OrderedCompletion Integer` which allows for manipulations of $\pm\infty$). It is included to faciliate checks whether or not a given pseudo differential operator is finite.

The only non-trivial operation is the multiplication. Since the number of terms which have to be computed grows rapidly with each further coefficient, a careful management of the lazy evaluation is necessary to avoid superfluous calculations. We have also implemented a special version which tries to check whether the result has only a finite number of non-vanishing terms.

The generation of a hierarchy using (16) requires higher and higher powers of the same operator $M$. For efficiency a special multiplication procedure `repeatedMult` is implemented which stores the derivatives of the coefficients of this operator. Otherwise they would be recomputed in each step. The output of `repeatedMult(L,M)` is the stream `(L,L·M,L·M·M,...)`.

The package `PDOAlgebra` implements division by a pseudo differential operator and extraction of roots. In both cases explicit formulae for the coefficients would be very complicated. It is much simpler to compute them as solutions of systems of linear equations by making a generic ansatz for the wanted pseudo differential operator. In order to be able to set up and solve these systems, the coefficient domain must now belong additionally to the category `FunctionSpace Integer` [2].

AXIOM provides in its huge library already a domain for linear differential operators: `LinearOrdinaryDifferentialOperator (LODO)`. It is, however, for several reasons not convenient for our purposes. Especially awkward is the fact that it requires as parameter a left module of the coefficient domain representing the definition domain on which the operators are acting. Since in our applications these two domains are always identical, we implemented the somewhat strange domain constructor `SelfLeftModuleWithDerivation (SLMD)` which turns a differential ring into a left module over itself.

Furthermore, a few needed operations are missing in `LODO`. Consequently, we have introduced the new domain `LinearOrdinaryDifferentialOperator2 (LODO2)`. It is based on the old one, but takes automatically the same domain as definition and coefficient domain using the above mentioned constructor `SLMD` and is augmented by a few procedures. Especially, it is now easier to see such an operator as a polynomial in the differentiation operator.

For generic pseudo differential operators, as they are for instance used in `PDOAlgebra` to make an ansatz for a quotient or a root, one needs indexed functions with prescribed arguments to represent the coefficients. Similiarly, we need in Sato theory indexed variables. The package `IndexedFunction` provides both. It takes as argument a domain from the category `FunctionSpace Integer` elements of which the generated functions are.

For our applications we have implemented the domain `DependentFunction` for the coefficients of the differential operators. It provides functions with implicit dependencies, i.e. they can depend on variables which do not explicitly occur in their arguments. This is on one side often convenient to avoid lengthy argument lists in the output, on the other side this is necessary, if functions shall depend at least formally on an infinite number of variables.

`DependentFunction` is based on the domain `Expression Integer` for general symbolic expressions. The only difference lies in the differentiation which now also respects the implicit dependencies. The procedure `function` generates kernels with a special entry in their property list. This entry contains all variables on which the function depends implicitly. Such a function can also depend on indexed variables, if their name (without index!) appears in the list.

The package `LaxEquation` provides several procedures to evaluate (general-

[2] Actually, it might appear more naturally to require it to be in the category `Field`. But in view of our applications, where the coefficients are usually symbolic expressions, this is more convenient.

ized) Lax and Zakharov-Shabat equations. It takes two domains as parameters: one for the operators and one for their coefficients. This allows one to use the same package for different kind of operators. Especially, it is also possible to apply it to matrix operators.

Finally, the package `LaxHierarchy` implements the generation of a hierarchy of Lax equations as described in Section 3. Such a hierarchy is again represented as a stream. Similiarly on can produce a stream of the conserved densities $Q_r$ associated with the hierarchy. This package is restricted to linear differential operators from `LODO2`. Only the coefficient domain for the operators is taken as parameter.

## 5 Examples

As examples we present here some standard equations in integrable system theory. Most of the computations for them could be done by hand. But the purpose of this section is only to demonstrate the possibilities of our program by showing some well-known calculations and not to introduce some new results.

The classical example of an integrable system is the Korteweg-de Vries equation. Miura, Gardner and Kruskal [10] were the first to show that the eigenvalues of the Schrödinger operator

$$L = D^2 + u(x,t) \tag{25}$$

are invariant, if $u$ satisfies the KdV equation

$$4u_t = u_{xxx} + 6uu_x \,. \tag{26}$$

Thus we can construct the KdV hierarchy along the lines of Section 3 by starting with the above $L$. With

$$L^{1/2} = D + \frac{u}{2}D^{-1} - \frac{u_x}{4}D^{-2} + \frac{u_{xx} - u^2}{8}D^{-3} + \cdots \,, \tag{27}$$

non-trivial equations result, if the positive part of an odd power of $L^{1/2}$ is taken as second operator in the Lax pair. The classical KdV equation (26) is for instance the third equation of the hierarchy and obtained using

$$M = (L^{3/2})_+ = D^3 + \frac{3u}{2}D + \frac{3u_x}{4} \,. \tag{28}$$

Figure 1 shows the complete AXIOM session for this computation. Only some commands to load needed domains and packages are omitted. The `)set stream calculate` command determines the number of terms calculated in a stream. The next five lines defines the needed data types and packages: `ctyp` for the coefficients, `ltyp` for linear operators like $L$ and `ptyp` for the pseudo differential operators. `algpack` contains the procedures for divisions and roots; `laxpack` the ones for Lax equations.

The following lines perform the actual computation. Because AXIOM allows operations to be overloaded, it is often necessary to indicate explicitly either the

```
)set stream calculate 5

ctyp:=DependentFunction()
ltyp:=LinearOrdinaryDifferentialOperator2(ctyp,'x,'D)
ptyp:=PseudoDifferentialOperator(ctyp,'x,'D)
algpack:=PDOAlgebra(ctyp,'x,'D)
laxpack:=LaxEquation(ctyp,ltyp,'t,'x,'D)

u:ctyp:=function('u,['x,'t])

   (6)  u(x,t)

L:ltyp:=D()**2+u

          2
   (7)  D  + u(x,t)

M:ptyp:=nthRoot(L)$algpack

                                                                    2
                          u  (x,t)           u    (x,t) - u(x,t)
             u(x,t)   - 1  ,1          - 2    ,1,1                    - 3
   (8)  D + ------ D    - -------- D    + -------------------- D
               2              4                    8

              - 4
        + O(D    )

evalLax(L,pos(M**3))$laxpack

        - u      (x,t) + 4u  (x,t) - 6u(x,t)u  (x,t)
           ,1,1,1          ,2                 ,1
   (9)  ---------------------------------------------
                              4

conservedDensities(M)$laxpack

   (10)
                                         2
                  u    (x,t) + 3u(x,t)
    u(x,t)         ,1,1
   [------, 0, ---------------------, 0,
      2                 8
                                                           2          3
    u         (x,t) + 10u(x,t)u    (x,t) + 5u  (x,t)  + 10u(x,t)
     ,1,1,1,1                  ,1,1            ,1
    ----------------------------------------------------------, ...]
                                   32
```

**Fig. 1.** Session for KdV equation.

types of some of their arguments, the type of their result or the package in which they are defined. Otherwise it may be impossible for the AXIOM interpreter to find the correct operation.

`D()` represents the differentation operator; **`nthRoot`** computes the $n$-th root of a linear differential operator of order $n$ (it is a special case of a similiar procedure for pseudo differential operators). The calculation of the root is the most time-consuming step in the whole derivation. It takes about 7 sec on an IBM RS 6000 Model 530 with 64MB memory. The evaluation of the Lax form is of course trivial. Because of the power it takes about 1.5 sec.

The final line computes the first five conserved densities using (17). Again only the odd ones are non-trivial. Assuming that $u$ and all its derivatives tend sufficiently fast to zero as $|x| \to \infty$ we can simplify the corresponding conserved quantities by partial integration and obtain after multiplication with suitable factors

$$\begin{aligned} Q_1 &= \int_{-\infty}^{\infty} dx\, u\,, \\ Q_3 &= \int_{-\infty}^{\infty} dx\, u^2\,, \\ Q_5 &= \int_{-\infty}^{\infty} dx\, \left(u^3 - u_x^2/2\right)\,. \end{aligned} \tag{29}$$

We can compute the complete KdV hierarchy using the command `generate-Hierarchy(L)`. The first seven equations are shown in Figure 2. It was obtained using the TeX interface of AXIOM. Derivatives are here denoted by a comma preceding the number of the argument with respect to which the function is differentiated. We see, that indeed every second equation is trivial and that the third line in the figure is equation (26). The time needed for this computation varies between 90 and 100 sec.

$$u_{,2}(x,\, t) - u_{,1}(x,\, t)$$

$$u_{,2}(x,\, t)$$

$$\left[-u_{,1,1,1}(x,\, t) + 4\ u_{,2}(x,\, t) - 6\ u(x,\, t)\ u_{,1}(x,\, t)\right]/4$$

$$u_{,2}(x,\, t)$$

$$\begin{aligned}&\left[-u_{,1,1,1,1,1}(x,\, t) - 10\ u(x,\, t)\ u_{,1,1,1}(x,\, t) - 20\ u_{,1}(x,\, t)\ u_{,1,1}(x,\, t) + 16\ u_{,2}(x,\, t)\right.\\ &\qquad \left. - 30\ u(x,\, t)^2\ u_{,1}(x,\, t)\right]/16\end{aligned}$$

$$u_{,2}(x,\, t)$$

$$\begin{aligned}&\left[-u_{,1,1,1,1,1,1,1}(x,\, t) - 14\ u(x,\, t)\ u_{,1,1,1,1,1}(x,\, t) - 42\ u_{,1}(x,\, t)\ u_{,1,1,1,1}(x,\, t)\right.\\ &\qquad + \left(-70\ u_{,1,1}(x,\, t) - 70\ u(x,\, t)^2\right)\ u_{,1,1,1}(x,\, t) - 280\ u(x,\, t)\ u_{,1}(x,\, t)\ u_{,1,1}(x,\, t)\\ &\qquad \left. + 64\ u_{,2}(x,\, t) - 70\ u_{,1}(x,\, t)^3 - 140\ u(x,\, t)^3\ u_{,1}(x,\, t)\right]/64\end{aligned}$$

**Fig. 2.** Output for KdV hierarchy.

As a second example we treat the Boussinesq hierarchy (Figure 3). The procedure is like in the previous example except that we start this time with the operator

$$L = D^3 + p(x,t)D + q(x,t)\,. \tag{30}$$

Thus each level of the hierarchy consists now of a system of two equations for the two functions $p, q$. The second system yields the Boussinesq equation. After the transformation $t \to -t$, it is possible to eliminate $q$ by cross-differentiating and one gets

$$3\,p_{tt} + p_{xxxx} + 2\,(p^2)_{xx} = 0\,. \tag{31}$$

To show the behavior of the program in longer calculations, we computed the first ten systems of this hierarchy. This took about 800 sec. Nearly 200 of them were used for garbage collection. Figure 3 contains only the first five systems, as they become quickly very large.

$$\left\{p_{,2}\,(x,\,t) - p_{,1}\,(x,\,t),\; q_{,2}\,(x,\,t) - q_{,1}\,(x,\,t)\right\}$$

$$\begin{aligned}&\big\{p_{,1,1}\,(x,\,t) - 2\;q_{,1}\,(x,\,t) + p_{,2}\,(x,\,t),\\ &\;[2\;p_{,1,1,1}\,(x,\,t) - 3\;q_{,1,1}\,(x,\,t) + 3\;q_{,2}\,(x,\,t) + 2\;p\,(x,\,t)\;p_{,1}\,(x,\,t)]/3\big\}\end{aligned}$$

$$\left\{p_{,2}\,(x,\,t),\; q_{,2}\,(x,\,t)\right\}$$

$$\begin{aligned}&\big\{[p_{,1,1,1,1}\,(x,\,t) - 2\;q_{,1,1,1}\,(x,\,t) + 2\;p\,(x,\,t)\;p_{,1,1}\,(x,\,t) - 4\;p\,(x,\,t)\;q_{,1}\,(x,\,t) +\\ &\qquad 3\;p_{,2}\,(x,\,t) + 2\;p_{,1}\,(x,\,t)^2 - 4\;q\,(x,\,t)\;p_{,1}\,(x,\,t)]/3,\\ &\;[2\;p_{,1,1,1,1,1}\,(x,\,t) - 3\;q_{,1,1,1,1}\,(x,\,t) + 6\;p\,(x,\,t)\;p_{,1,1,1}\,(x,\,t) - 6\;p\,(x,\,t)\;q_{,1,1}\,(x,\,t) +\\ &\qquad 12\;p_{,1}\,(x,\,t)\;p_{,1,1}\,(x,\,t) + 9\;q_{,2}\,(x,\,t) + (-6\;p_{,1}\,(x,\,t) - 12\;q\,(x,\,t))\;q_{,1}\,(x,\,t) +\\ &\qquad 4\;p\,(x,\,t)^2\;p_{,1}\,(x,\,t)]/9\big\}\end{aligned}$$

$$\begin{aligned}&\big\{[p_{,1,1,1,1,1}\,(x,\,t) + 5\;p\,(x,\,t)\;p_{,1,1,1}\,(x,\,t) + (5\;p_{,1}\,(x,\,t) + 15\;q\,(x,\,t))\;p_{,1,1}\,(x,\,t) +\\ &\qquad (15\;p_{,1}\,(x,\,t) - 30\;q\,(x,\,t))\;q_{,1}\,(x,\,t) + 9\;p_{,2}\,(x,\,t) + 5\;p\,(x,\,t)^2\;p_{,1}\,(x,\,t)]/9,\\ &\;[q_{,1,1,1,1,1}\,(x,\,t) + 5\;p\,(x,\,t)\;q_{,1,1,1}\,(x,\,t) + 10\;q\,(x,\,t)\;p_{,1,1,1}\,(x,\,t) + (15\;p_{,1}\,(x,\,t) -\\ &\qquad 15\;q\,(x,\,t))\;q_{,1,1}\,(x,\,t) + 20\;q_{,1}\,(x,\,t)\;p_{,1,1}\,(x,\,t) + 9\;q_{,2}\,(x,\,t) - 15\;q_{,1}\,(x,\,t)^2 +\\ &\qquad 5\;p\,(x,\,t)^2\;q_{,1}\,(x,\,t) + 10\;p\,(x,\,t)\;q\,(x,\,t)\;p_{,1}\,(x,\,t)]/9\big\}\end{aligned}$$

**Fig. 3.** Output for the Boussinesq hierarchy.

As a simple example of a higher-dimensional system we consider the derivation of the Kadomtsev-Petviashvili hierarchy [11]. We set in (19) $\gamma = \alpha_n = 1$. Evaluating now the Zakharov-Shabat equation (24) with $n = 2$ and $m = 3$ yields two equations for $u_{-1}$ and $u_{-2}$

$$\begin{gathered}u_{-1,t_2} = u_{-1,xx} + 2u_{-2,x}\,,\\ 3u_{-2,t_2} + 3u_{-1,xt_2} + 6u_{-1}u_{-1,x} = 3u_{-2,xx} + 2u_{-1,t_3}\,.\end{gathered} \tag{32}$$

If we eliminate $u_{-2}$ by cross-differentiation and identify $t_2 = y, t_3 = t, u_{-1} = u$ we get the KP equation in the form

$$(4u_t - 12uu_x - u_{xxx})_x - 3u_{yy} = 0\,. \tag{33}$$

It is also possible to obtain the KP equation from the generalized Lax equation (23) with $n = 2$ and $n = 3$, but this requires a more complicated elimination.

Figure 4 shows the corresponding AXIOM session. The procedure `functionStream` generates a stream of kernels representing indexed functions with explicit and implicit dependencies of variables. Here, we have chosen to take all dependencies implicit. Note that there is no difference between indexed and normal variables; `x` and `t` appear on the same footing, although the latter one is used as name for the indexed time variables. The time needed for this computation varies between 1 and 2 sec.

```
L : ptyp := D()+generate(functionStream(u,[],['x,'t],-1,-1),-1)

                   - 1      - 2
   (5)  D + u   ()D    + O(D   )
             - 1

B : NonNegativeInteger -> ltyp
B := n +-> pos(L**n)$ptyp

evalZakharovShabat(B,2,3)$laxpack

   (8)
     (- 3u          () + 3u        () - 6u       ())D - u          ()
          - 1,x,x          - 1,t          - 2,x          - 1,x,x,x
                                2

    + 3u          () - 3u          () - 2u        () + 6u    ()u     ()
        - 1,x,t          - 2,x,x          - 1,t          - 1   - 1,x
               2                               3

    + 3u        ()
        - 2,t
             2
```

**Fig. 4.** Session for KP equation.

Finally, we want to comment briefly on more general operators. It was already mentioned in Section 4 that `LaxEquation` can be used with many different kinds of operators. For instance it is thus also possible to evaluate the Lax pair for the Nonlinear Schrödinger Equation [12]

$$\mathrm{i}u_t + u_{xx} + \frac{2}{1-k^2}u^2u = 0 \tag{34}$$

consisting of matrices of linear differential operators with complex coefficients

$$\begin{aligned} L &= \mathrm{i}\begin{pmatrix} 1+k & 0 \\ 0 & 1-k \end{pmatrix} D + \begin{pmatrix} 0 & \bar{u} \\ u & 0 \end{pmatrix}, \\ M &= \mathrm{i}k \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} D^2 + \begin{pmatrix} -\dfrac{\mathrm{i}u\bar{u}}{1+k} & \bar{u}_x \\ -u_x & \dfrac{\mathrm{i}u\bar{u}}{1-k} \end{pmatrix}. \end{aligned} \tag{35}$$

To perform this calculation with our program is straight-forward.

```
ctyp := Complex DependentFunction()
ltyp := LinearOrdinaryDifferentialOperator2(ctyp,'x,'D)
mtyp := SquareMatrix(2,ltyp)
laxpack := LaxEquation(ctyp,mtyp,'t,'x,'D)
u:ctyp := function('u,['x,'t])
uq:ctyp := function('uq,['x,'t])
d:ltyp := D()
k:ctyp := K
L := matrix [[%i*(1+k)*d,uq], [u,%i*(1-k)*d]]
M := matrix [[%i*k*d**2-%1*u*uq/(1+k), differentiate(uq,x)], _
             [-differentiate(u,x), %i*k*d**2+%i*u*uq/(1-k)]]
evalLax(L,M)$laxpack
```

Because of the complicated data types entering `L` and `M` is fairly clumsy. As already mentioned above, the overloading makes it difficult for the interpreter to identify the correct operation. In such cases it is often quite useful to introduce auxiliary variables like `d` or `k` and to give their types explicitly.

## References


1. J. Hoppe. *Lectures on Integrable Systems.* Lecture Notes in Physics m10. Springer Verlag, Berlin, 1992.
2. P.D. Lax. Integrals of nonlinear equations of evolution and solitary waves. *Comm. Pure Appl. Math.*, 21:467–490, 1968.
3. J. Schur. Über vertauschbare lineare Differentialausdrücke. *Sitzungsber. Berliner Math. Ges., 29. Sitz.*, pages 2–8, 1904. (App. to Archiv Math. Phys. 8, 1905).
4. V.G. Drinfeld and V.V. Sokolov. Lie algebras and equations of Korteweg-de Vries type. *J. Sov. Math.*, 30:1975–2036, 1985.
5. Y. Ohta, J. Satsuma, D. Takahashi, and T. Tokihiro. An elementary introduction to Sato theory. *Prog. Theor. Phys. Supp.*, 94:210–241, 1988.
6. W. Oevel. Darboux theorems and Wronskian formulas for integrable systems I. Constrained KP flows. *Physica A*, 195:533–576, 1993.
7. D. Jenks and R.S. Sutor. *AXIOM – The Scientific Computation System.* Springer-Verlag, New York, 1992.
8. M. Ito. A REDUCE program for evaluating a Lax pair form. *Comp. Phys. Comm.*, 34:325–331, 1985.
9. W.H. Burge and S.M. Watt. Infinite structures in Scratchpad II. In J.H. Davenport, editor, *Proc. EUROCAL '87*, Lecture Notes in Computer Science 378, pages 138–148. Springer-Verlag, Berlin 1987.
10. R.M. Miura, C.S. Gardner, and M.D. Kruskal. Korteweg-de Vries equation and generalizations II: Existence of conservation laws and constants of motion. *J. Math. Phys.*, 9:1204–1209, 1968.
11. E. Date, M. Jimbo, M. Kashiwara, and T. Miwa. Transformation groups for soliton equations. In M. Jimbo and T. Miwa, editors, *Nonlinear Integrable Systems – Classical Theory and Quantum Theory.* World Scientific, Singapore, 1983.
12. V.E. Zakharov and A.B. Shabat. Exact theory of two-dimensional self-focusing and of one-dimensional waves in nonlinear media. *Sov. Phys. JETP*, 34:62–69, 1972.