Computer Physics Communications 274 (2022) 108284

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Weakly nonlocal Poisson brackets: Tools, examples, computations ☆,☆☆

M. Casati [a], P. Lorenzoni [b,c], D. Valeri [d], R. Vitolo [e,f,*]

[a] School of Mathematics and Statistics, Ningbo University, Ningbo, China
[b] Dipartimento di Matematica e Applicazioni, Università di Milano-Bicocca, Milano, Italy
[c] Sezione INFN, Milano-Bicocca, Italy
[d] Dipartimento di Matematica, Sapienza Università di Roma, 00185 Rome, Italy
[e] Dept. of Mathematics and Physics "E. De Giorgi", Università del Salento, Lecce, Italy
[f] Sezione INFN, Lecce, Italy

ARTICLE INFO

Article history:
Received 17 January 2021
Received in revised form 2 November 2021
Accepted 6 January 2022
Available online 19 January 2022

Keywords:
Poisson bracket
Hamiltonian operator
Schouten bracket
Partial differential equations
Integrable systems

ABSTRACT

We implement an algorithm for the computation of Schouten bracket of weakly nonlocal Hamiltonian operators in three different computer algebra systems: Maple, Reduce and Mathematica. This class of Hamiltonian operators encompass almost all the examples coming from the theory of (1+1)-integrable evolutionary PDEs.

Program summary
Program Title: `Jacobi` (Maple), `CDE` module `cde_weaklynl.red` (Reduce, official distribution), `nlPVA` (Mathematica)
CPC Library link to program files: https://doi.org/10.17632/synmrvr74g.1
Developer's repository link: https://gdeq.org/Weakly_nonlocal_Poisson_brackets
Licensing provisions: BSD 2-clause
Programming language: Maple, Reduce (Rlisp), Mathematica
Supplementary material: Example program files in the three languages (Maple, Reduce, Mathematica)
Nature of problem: Calculating the Jacobi identity for weakly nonlocal Poisson brackets
Solution method: Bringing the Jacobi identity to a canonical form
Additional comments including restrictions and unusual features: Use a 2020 Maple or Mathematica version, or a 2021 Reduce snapshot

1. Introduction

Poisson brackets are ubiquitous in Mathematical and Theoretical Physics. Besides their traditional role in Mechanics, they play an important role in the study of integrable evolutionary systems of partial differential equations (PDEs) in two independent variables t, x and n dependent variables $\mathbf{u} = (u^1, \dots, u^n)$

$$u^i_t = f^i(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \dots), \qquad i = 1, \dots, n, \tag{1}$$

where f^i are differential polynomials. We refer to the books [1,9,11,26,30] for an account of this theory. In this context a *local* Poisson bracket of two local functionals $F = \int f(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \dots)dx$ and $G = \int g(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \dots)\,dx$ is a local functional of the form

☆ The review of this paper was arranged by Prof. Hazel Andrew.
☆☆ This paper and its associated computer programs are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail addresses: matteo@nbu.edu.cn (M. Casati), paolo.lorenzoni@unimib.it (P. Lorenzoni), daniele.valeri@uniroma1.it (D. Valeri), raffaele.vitolo@unisalento.it (R. Vitolo).
URL: http://poincare.unisalento.it/vitolo (R. Vitolo).

https://doi.org/10.1016/j.cpc.2022.108284
0010-4655/

$$\{F, G\}_P = \int \frac{\delta F}{\delta u^i} P^{ij} \frac{\delta G}{\delta u^j}\, dx, \tag{2}$$

where

$$P^{ij} = \sum_\sigma P^{ij}_\sigma(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, ...)\partial_x^\sigma$$

is a suitable (matrix) differential operator and $\delta F/\delta u^i$ and $\delta G/\delta u^i$ are the variational derivatives of the functionals F and G respectively. The differential polynomials f and g are called the *densities* (of the functionals F and G).

The operator P cannot be arbitrarily chosen. The fulfillment of skew-symmetry and the Jacobi identity for the Poisson bracket imposes strict constraints on P: it must be formally skew-adjoint and the Schouten bracket $[P, P]$ must be identically zero (see *e.g.* [7]).[1]

A system of evolutionary PDEs of the form (1) is said to be Hamiltonian w.r.t. the Hamiltonian operator P if there exists a local functional $H = \int h\, dx$, called the *Hamiltonian functional* such that

$$f^i(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, ...) = P^{ij} \frac{\delta H}{\delta u^j} \tag{3}$$

The integrability of the system (3) is revealed in the existence of infinitely many local functionals (including H) in involution with respect to the Poisson bracket defined by P. A universal procedure to construct a sequence $H_0 = H, H_1, H_2, ...$ of Hamiltonian functionals in involution is Magri's bi-Hamiltonian recursion [21]:

$$P_1^{ij} \frac{\delta H_{k+1}}{\delta u^j} = P_2^{ij} \frac{\delta H_k}{\delta u^j}. \tag{4}$$

It relies on the existence of a pair of Hamiltonian operators (P_1, P_2) satisfying the compatibility condition

$$[P_1, P_2] = 0,$$

where the square bracket denotes (as above) the Schouten bracket.

In general, the computation of the Schouten bracket is a highly nontrivial task and most of the examples can be handled only with the help of computer algebra systems. Recently in [5,28] two computer algebra packages for calculations of Schouten bracket for *local* differential operators have been introduced and described. The packages have been developed following two different mathematical approaches to differential operators: the algebraic approach through Poisson Vertex Algebras [5] and the supermanifold approach [28]. A fairly complete list of packages for local operators can be found in [28].

Since the beginning of research on integrable systems, it was clear that Poisson brackets defined by local differential operators did not cover all interesting examples. Nonlocal differential operators (i.e. integro-differential operators) are much more common: they can be found for the KdV equation, the modified KdV equation, the AKNS equation, the Nonlinear Schrödinger equation, etc. (see [29]).

A systematic research on nonlocal operators was started by Ferapontov and Mokhov [13], then widely generalized by Ferapontov in subsequent works [12]. A further generalization was proposed by Maltsev and Novikov that introduced the general class of weakly nonlocal operators [22]:

$$P^{ij} = \sum_\sigma P^{ij}_\sigma(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, ...)\partial_x^\sigma + \sum_{\alpha,\beta=1}^{N} e^{\alpha\beta} w^i_\alpha(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, ...)\partial_x^{-1} w^j_\beta(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, ...), \tag{5}$$

where the coefficients P^{ij}_σ, w^i_α, w^j_β are differential polynomials, $e^{\alpha\beta}$ is a symmetric constant matrix and the operator ∂_x^{-1} is defined to be

$$\partial_x^{-1} = \frac{1}{2}\int_{-\infty}^{x} dy - \frac{1}{2}\int_{x}^{+\infty} dy. \tag{6}$$

All the above mentioned examples of nonlocal operators have this form, thus certifying the importance and generality of that class.

The computation of Schouten bracket for nonlocal operators is a more difficult task with respect to the local case. Indeed, until recently it was not even clear how to reach a divergence-free canonical form of the Schouten bracket in the presence of nonlocal terms. Such canonical (or normal) form is crucial in order to get a set of necessary and sufficient conditions for the vanishing of the Schouten bracket. It is worth to emphasize that the only publicly available computer algebra package for the Schouten bracket of nonlocal operators is the Maple package `JET` [23]. Its strongest limit is that it is not always able to simplify the nonlocal terms because it lacks for an algorithm to that purpose.

Recently, a canonical form of the Schouten bracket for weakly nonlocal operators has been achieved in three different formalisms (distributions, differential operators and Poisson Vertex Algebras) leading to three different (but equivalent) algorithms for the computation of the Schouten bracket [4] (see [20] for the supermanifold approach).

The aim of this paper is to implement the algorithms of [4] in three different computer algebra systems, respectively, Maple, Reduce and Mathematica. Let us describe more in detail such implementations.

[1] Explicit formulas for Schouten bracket will be given later.

- *Maple implementation.* The package for Maple (called `Jacobi`) is the straightforward implementation of the algorithm introduced in [4] for the computation of Schouten bracket using the language of distributions. The package provides the syntax and the command for computing the formula for the Schouten bracket given in [10] and reducing it to the normal form described in [19,4]. We provide both a classical and a parallel implementation of the algorithm. We used Maple's `Grid` package and we split the computation in few parallel independent branches when possible. We found that strategy to be optimal as there is an overhead related with the initialization of the independent processes. While parallel computing has no effect on relatively simple computations (or makes things even worse), it reduces the timing to half when the calculation is more complex.
- *Reduce implementation.* This implementation is within the software package CDE,[2] an official Reduce package [16] devoted to calculations in the geometry of differential equations and integrable systems [18]. Since version 2.0 (2015), CDE can quite easily compute the conditions for a nonlocal operator to be Hamiltonian for a given partial differential equation (in the sense of mapping conserved quantities into symmetries), see [18] for examples. It can also compute the Jacobi property for *local* operators [28]. The newly added library `cde_weaklynl.red` provides the syntax and the command for computing Schouten brackets of weakly nonlocal operators. Here, the command uses the CDE data structure for differential operators and CDE treatment of nonlocal variables in order to introduce the data and carry out the algorithm.
  At the moment, `Reduce` can be installed by means of a binary snapshot of the official distribution. Such snapshots can be found from the official website. The latest snapshot at the time of writing can be found at the following link: https://sourceforge.net/projects/reduce-algebra/files/snapshot_2021-07-16/ and it contains all examples that are discussed in our paper. The main reason for using `Reduce` were the fact that it is free software, so that we have been able to learn how to code in an efficient way, and the fact that it has an extremely fast computer algebra engine, which is competitive with commercial software. The real drawbacks of `Reduce` at the moment are that it has a limited Graphical User Interface and it has no parallel computing capabilities. However, a web player is currently being implemented at the official website.
- *Mathematica implementation.* This implementation extends the package `MasterPVA.m`[3] [5], devoted to the computation of λ-brackets in local PVAs, to the case of weakly non-local PVAs. The new added package `nlPVA.wl` (developed for Mathematica 11) provides the syntax and the command for computing the three terms appearing in the Jacobi identity (40) for weakly non-local λ-brackets. The computation is done using the Master Formula for PVAs as in [4], for the local part of the λ-bracket, and the algorithm in [5] for the non-local part. In particular, the package can be used to compute (40) for the local case as well. We provide both a classical and a parallel implementation of the algorithm; the latter is available in the package `nlPVA-par.wl`

The three implementations are different in that the mathematical formalism on which each of them is based is different *and* the computer algebra system on which each of them is based is different. Despite such differences, the canonical divergence-free form of the results can be translated from one formalism to another, see [4]. However, the three implementations follow different paths in order to achieve corresponding results. That is due to the different data structure that they have to handle (distributions, differential operators, Poisson Vertex Algebras).

The software packages and examples are released under the terms of the FreeBSD license. They can be downloaded at the webpage of this paper in the Geometry of Differential Equations website, the URL is in [15].

Structure of the paper. For each implementation we will shortly recall its mathematical procedure. Then, we will discuss the details of the corresponding software package. For each software package we will present the same three examples of (systems of) partial differential equations that can be written in Hamiltonian form through a weakly nonlocal operator, and compute the Schouten bracket of any such operator.

The examples are: the *modified KdV* equation, the *Heisenberg magnet* system, for which we will also compute the compatibility of its weakly nonlocal operator with a second local Hamiltonian operator, and one new example of weakly nonlocal operator for a *WDVV equation* in the form of a quasilinear first-order system of PDEs. Also the examples can be downloaded from the web page of this paper at [15].

We observe that examples with non-zero output are not available. Indeed, there are no known examples of an evolutionary PDE in two independent variables which have two Hamiltonian operators that are not compatible. For some PDEs that depends on the fact that there exists no non-trivial three-vectors, provided that their linearization fulfills a certain property (see [17]). It is also easy to check that simple changes in input (*e.g.* in a sign or in a coefficient) yield non-zero results, showing the 'rigidity' of the results.

2. Maple implementation

In what follows, let us denote the independent variable by x, the dependent variables by (u^i), $1 \leq i \leq n$, and derivatives by $u^i_\sigma = \partial^\sigma u^i / \partial x^\sigma$; note that if $\sigma = 0$ then $u^i_\sigma = u^i$. The *total derivative* ∂_x is defined as

$$\partial_x = \frac{\partial}{\partial x} + u^i_{\sigma+1} \frac{\partial}{\partial u^i_\sigma}, \tag{7}$$

(the summation convention holds as usual).

The Schouten bracket of weakly nonlocal Poisson bivectors is implemented in the Maple package `Jacobi` (filename `jacobi.mpl`) using the language of distributions (see, e.g. [4,10]). Let us briefly describe the mathematical algorithm. We will need three different labels for the independent variable: x, y, z. The σ-th power of ∂_x is denoted by ∂_x^σ, and analogously for other variables.

[2] This is an acronym standing for Calculus on Differential Equations.
[3] The acronym is motivated from the fact that the package computes the so-called Master Formula (see [2]) in PVAs.

A weakly nonlocal Poisson bivector has the form

$$P^{ij}_{x,y} = \sum_{k\geq 0} B^{ij}_k(u^h, u^h_\sigma)\delta^{(k)}(x-y) + c^{\alpha\beta} w^i_\alpha(u^k, u^k_\sigma)\nu(x-y)w^j_\beta(u^k, u^k_\sigma)$$

where $\nu(x-y) = \frac{1}{2}\mathrm{sgn}(x-y)$. In the above formula it is understood that

$$P^{ij}_{x,y} = -P^{ji}_{y,x}$$

This implies $c^{\alpha\beta} = c^{\beta\alpha}$.

Following [10] the Schouten bracket of two weakly nonlocal Poisson bivectors $P^{ij}_{x,y}$ and

$$Q^{ij}_{x,y} = \sum_{k\geq 0} C^{ij}_k(u^h, u^h_\sigma)\delta^{(k)}(x-y) + d^{\alpha\beta} z^i_\alpha(u^k, u^k_\sigma)\nu(x-y)z^j_\beta(u^k, u^k_\sigma)$$

can be defined as

$$\begin{aligned}
[P, Q]^{ijk}_{x,y,z} = &\frac{\partial P^{ij}_{x,y}}{\partial u^l_\sigma(x)}\partial^\sigma_x Q^{lk}_{x,z} + \frac{\partial P^{ij}_{x,y}}{\partial u^l_\sigma(y)}\partial^\sigma_y Q^{lk}_{y,z} + \frac{\partial P^{ki}_{z,x}}{\partial u^l_\sigma(z)}\partial^\sigma_z Q^{lj}_{z,y} + \\
&\frac{\partial P^{ki}_{z,x}}{\partial u^l_\sigma(x)}\partial^\sigma_x Q^{lj}_{x,y} + \frac{\partial P^{jk}_{y,z}}{\partial u^l_\sigma(y)}\partial^\sigma_y Q^{li}_{y,x} + \frac{\partial P^{jk}_{y,z}}{\partial u^l_\sigma(z)}\partial^\sigma_z Q^{li}_{z,x} + \\
&\frac{\partial Q^{ij}_{x,y}}{\partial u^l_\sigma(x)}\partial^\sigma_x P^{lk}_{x,z} + \frac{\partial Q^{ij}_{x,y}}{\partial u^l_\sigma(y)}\partial^\sigma_y P^{lk}_{y,z} + \frac{\partial Q^{ki}_{z,x}}{\partial u^l_\sigma(z)}\partial^\sigma_z P^{lj}_{z,y} + \\
&\frac{\partial Q^{ki}_{z,x}}{\partial u^l_\sigma(x)}\partial^\sigma_x P^{lj}_{x,y} + \frac{\partial Q^{jk}_{y,z}}{\partial u^l_\sigma(y)}\partial^\sigma_y P^{li}_{y,x} + \frac{\partial Q^{jk}_{y,z}}{\partial u^l_\sigma(z)}\partial^\sigma_z P^{li}_{z,x}
\end{aligned} \tag{8}$$

The vanishing of the Schouten bracket $[P, Q]^{ijk}_{x,y,z}$ means that for any choice of the test functions $p_i(x), q_j(y), r_k(z)$ the triple integral

$$\iiint [P, Q]^{ijk}_{x,y,z} p_i(x) q_j(y) r_k(z)\, dxdydz \tag{9}$$

should vanish. Using distributional identities it is possible to reduce the Schouten bracket to a normal form [4,19]:

1. Using the identities

$$\begin{aligned}
&\nu(z-y)\delta(z-x) = \nu(x-y)\delta(x-z) \\
&\nu(y-x)\delta(y-z) = \nu(z-x)\delta(z-y) \\
&\nu(x-z)\delta(x-y) = \nu(y-z)\delta(y-x)
\end{aligned} \tag{10}$$

and their differential consequences we can eliminate all terms containing $\nu(z-y)\delta^{(n)}(z-x)$, $\nu(y-x)\delta^{(n)}(y-z)$, $\nu(x-z)\delta^{(n)}(x-y)$ producing nonlocal terms containing $\nu(x-y)\delta^{(n)}(x-z)$, $\nu(z-x)\delta^{(n)}(z-y)$, $\nu(y-z)\delta^{(n)}(y-x)$ and additional local terms.

2. Using the identities

$$\begin{aligned}
f(z)\delta^{(n)}(x-z) &= \sum_{k=0}^{n}\binom{n}{k} f^{(n-k)}(x)\delta^{(n-k)}(x-z), \\
f(y)\delta^{(n)}(z-y) &= \sum_{k=0}^{n}\binom{n}{k} f^{(n-k)}(z)\delta^{(n-k)}(z-y), \\
f(x)\delta^{(n)}(y-x) &= \sum_{k=0}^{n}\binom{n}{k} f^{(n-k)}(y)\delta^{(n-k)}(y-x),
\end{aligned} \tag{11}$$

we can eliminate the dependence on z in the coefficients of the terms containing $\nu(x-y)\delta^{(n)}(x-z)$, the dependence on y in the coefficients of the terms containing $\nu(z-x)\delta^{(n)}(z-y)$ and the dependence on x in the coefficients of the terms containing $\nu(y-z)\delta^{(n)}(y-x)$. After the first two steps the Schouten bracket has the form

$$\begin{aligned}
&c_1(x,y,z)\nu(x-y)\nu(x-z) + c_2(x,y,z)\nu(y-z)\nu(y-x) \\
&\qquad + c_3(x,y,z)\nu(z-x)\nu(z-y) + \sum_{n\geq 0} a_n(x,y)\nu(x-y)\delta^{(n)}(x-z) \\
&\qquad\qquad + \sum_{n\geq 0} b_n(x,z)\nu(z-x)\delta^{(n)}(z-y) + \sum_{n\geq 0} c_n(y,z)\nu(y-z)\delta^{(n)}(y-x) + \dots
\end{aligned} \tag{12}$$

where $\dots$ are local terms.

3. The remaining local part can be reduced to the form

$$\sum_{m,n} f_{mn}(x)\delta^{(m)}(x-y)\delta^{(n)}(x-z) \tag{13}$$

using the identities (and their differential consequences)

$$\delta(z-x)\delta(z-y) = \delta(y-x)\delta(y-z) = \delta(x-y)\delta(x-z) \tag{14}$$

and the identities (11).

As proved in [4] no further simplifications are possible and the vanishing of the Schouten bracket turns out to be equivalent to the vanishing of each coefficient in the reduced form.

We provided two implementations of the above algorithm: a classical, non-parallel implementation in the library file `jacobi.mpl` and an implementation that uses Maple parallel programming routines in the library file `jacobi_p.mpl`. The difference in the execution times of the examples is discussed below, as well as some details of the implementation of the parallel code. The most important thing to be remembered when using the parallel code is that it *requires* 4 cores to run correctly. On a modern computer that is not a problem; however, errors could arise when trying to run the parallel code on an older version of Maple/older computer, or, for example, when running Maple on a virtual machine.

2.1. The mKdV equation

We consider the mKdV equation $u_t = u^3 u_x + u_{xxx}$ and its nonlocal Hamiltonian operator

$$P = \partial_x^3 + \frac{2}{3}(u^2\partial_x + uu_x - u_x\partial_x^{-1}u_x) \tag{15}$$

(see, *e.g.*, [29]). The Hamiltonian property of P is proved by checking that the Schouten bracket vanishes, $[P, P] = 0$.

First of all, we set up the number of components

```
N:=1:
```

and the maximal order of derivative coordinate $(u_x, u_{xx}, \ldots)$ appearing in the coefficients of the operator P

```
M1:=1:
```

and the maximal order of derivative of the δ function appearing in the operator P

```
D1:=3:
```

Since the two arguments of the Schouten bracket coincide we define

```
M2:=M1:D2:=D1:
```

The field variables u^i_σ are introduced as `u[i,x,σ]` (it is not possible to change the symbols `u` and `x`). We introduce the nonlocal part $w^i_j u^j_x \partial_x^{-1} w^k_h u^h_x$ of the operator, w^i_j is `W[i,j]` in the program:

```
W:=Matrix(N,N):
W[1,1]:=1:
```

Finally, we input the differential operator as a 3-dimensional array. The last index must be 0 in input; it will be increased throughout the calculation.

```
P := Array(1..N,1..N,0..M2);
P[1, 1, 0] := delta[x - y, 3] + 2/3*u[1, x, 0]^2*delta[x - y, 1]
 + 2/3*u[1, x, 0]*u[1, x, 1]*delta[x - y, 0]
 - 2/3*W[1, 1]*u[1, x, 1]*delta[x - y, -1]
  *subs(x = y, W[1, 1])*u[1, y, 1];
```

Again, the notation `delta` and the use of the second independent variable `y` are not modifiable by the user. Internally, the maximal order of derivative coordinate is set to be equal to

```
M:=M1+D1+M2+D2;
```

The main procedure is loaded by

```
read('jacobi.mpl');
```

where the file `jacobi.mpl` shall be in the current Maple directory. Alternatively, an implementation that makes some use of the parallel programming capabilities of Maple can be used by means of

```
read('jacobi_p.mpl');
```

The result of the calculation is a 3-vector, and it shall be defined beforehand:

```
T := Array(1..N,1..N,1..N):
```

We can now call the main procedure; the output will be stored in T, or $T = [P, P]$:

```
Schouten_bracket(P,P,T,N,M1,D1,M2,D2);
```

The program starts computing, and prints the stages of the procedure as follows. The first message

```
"Step 0: calculating Schouten bracket"
```

is printed when the program is computing formula (8). The iterated total derivatives of the components of P are calculated by induction up to the order $M-1$, then the results are multiplied by the coefficients of P and summed up and internally stored in a 3-vector `T0`. In the parallel version, since the calculation has to be repeated for P and Q (if they are different), two parallel processes are run through the `Grid` library. Then

```
"Step 1 of the algorithm"
```

is showed when the identities (10) are used in order to reduce the nonlocal part. A short notation for the product of `delta` and a step function is used:

```
delta[z-y,-1,z-x,0]:=delta[x-y,-1]*delta[x-z,0]:
```

Here the result is stored in the 3-vector `T1`. The reduction is completed by means of the identities (11) when the message

```
"Step 2 of the algorithm"
```

is issued, defining a new 3-vector `T2` and completing the reduction of the nonlocal part to the canonical form. Since the reduction process can be split into several independent calculations, Step 2 is also split into 4 parallel calculations in `jacobi_p.mpl`.

The reduction is completed by means of the identities (14) that act on the local part only, after the message

```
"Step 3 of the algorithm"
```

The final result is stored in the 3-vector `T3` that is returned to the command line and shown; 0 in this case. The user can show all components of the result one by one, instead of getting them altogether, by the command

```
for i to N do
  for j to N do
    for k to N do
      print(SB[i, j, k] = T[i, j, k]);
    end do;
  end do;
end do;
```

The execution time is 0.09 s for `jacobi` and 0.31 s for the parallel version `jacobi_p`, showing that the parallel programming model that we adopted has an overhead that makes its use in simple computations not so effective. The calculations for all Maple examples have been performed on a laptop with an Intel i7-8565U processor.

2.2. The Heisenberg magnet equation

The following operator is an example of nonlocal Hamiltonian operator of a class introduced by Ferapontov (see [12] and references therein):

$$P = f^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \partial_x + f \begin{pmatrix} u^1 u^1_x + u^2 u^2_x & u^1 u^2_x - u^2 u^1_x \\ u^2 u^1_x - u^1 u^2_x & u^1 u^1_x + u^2 u^2_x \end{pmatrix} + \begin{pmatrix} u^1_x \partial_x^{-1} u^1_x & u^1_x \partial_x^{-1} u^2_x \\ u^2_x \partial_x^{-1} u^1_x & u^2_x \partial_x^{-1} u^2_x \end{pmatrix} \tag{16}$$

where $f = (1/2)((u^1)^2 + (u^2)^2 + 1)$. The operator P is a Hamiltonian operator for the Heisenberg magnet equation [12]:

$$u^1_t = u^2_{xx} + \frac{1}{f}\left(u^2 (u^1_x)^2 - 2u^1 u^1_x u^2_x - u^2 (u^2_x)^2\right), \tag{17}$$

$$u^2_t = -u^1_{xx} - \frac{1}{f}\left(u^1 (u^2_x)^2 - 2u^2 u^1_x u^2_x - u^1 (u^1_x)^2\right). \tag{18}$$

The Schouten bracket $[P, P]$ is computed in Maple as follows. We introduce the number of components and a maximal order of x-derivatives of the dependent variables u^i and of the delta function

```
N:=2;
M1:=1;
D1:=1,
M2:=M1;
D2:=D1;
```

and we define the operator:

```
f := u[1,x,0]^2/2 + u[2,x,0]^2/2 + 1/2;
P := Array(1..N,1..N,0..M2);
P[1,1,0] := f^2*delta[x-y,1] + f*(u[1,x,0]*u[1,x,1]
 + u[2,x,0]*u[2,x,1])*delta[x-y,0] + u[1,x,1]*delta[x-y,-1]*u[1,y,1];
P[2,2,0] := f^2*delta[x-y,1] + f*(u[1,x,0]*u[1,x,1]
 + u[2,x,0]*u[2,x,1])*delta[x-y,0] + u[2,x,1]*delta[x-y,-1]*u[2,y,1];
P[1,2,0] := f*(u[1,x,0]*u[2,x,1]-u[2,x,0]*u[1,x,1])*delta[x-y,0]
 + u[1,x,1]*delta[x-y,-1]*u[2,y,1];
P[2,1,0] := f*(u[2,x,0]*u[1,x,1]-u[1,x,0]*u[2,x,1])*delta[x-y,0]
 + u[2,x,1]*delta[x-y,-1]*u[1,y,1];
```

after loading the program, the bracket is computed as in the previous example:

```
read('jacobi.mpl');
T := Array(1..N,1..N,1..N);
Schouten_bracket(P,P,T,N,M1,D1,M2,D2);
```

The components of the output 3-vector can be shown one by one:

```
for i to N do
  for j to N do
    for k to N do
      print(SB[i,j,k] = T[i,j,k]);
    end do;
  end do;
end do;
SB[1, 1, 1] = 0
SB[1, 1, 2] = 0
SB[1, 2, 1] = 0
SB[1, 2, 2] = 0
SB[2, 1, 1] = 0
SB[2, 1, 2] = 0
SB[2, 2, 1] = 0
SB[2, 2, 2] = 0
```

The Heisenberg magnet equation is indeed bi-Hamiltonian. The second Hamiltonian operator can be written in the above coordinates as the ultra-local operator

$$Q = f^2 \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \delta(x-y). \tag{19}$$

In this case $M_1 = D_1 = M_2 = D_2 = 0$. The Hamiltonian property is easily checked. First of all we load the operator:

```
Q := Array(1..N,1..N,0..M):
Q[1,1,0] := 0;
Q[2,2,0] := 0;
Q[1,2,0] := -f^2*delta[x-y,0];
Q[2,1,0] := f^2*delta[x-y,0];
```

Then we compute the Schouten bracket:

```
S := Array(1..N,1..N,1..N);
Schouten_bracket(Q,Q,S,N,M1.D1,M2,D2);
```

which yields zero. Finally, we calculate the compatibility condition between the two Poisson brackets:

```
U := Array(1..N,1..N,1..N);
Schouten_bracket(P,Q,U,N,M1,D1,M2,D2),
```

where $M_1 = 1, D_1 = 1, M_2 = 0, D_2 = 0$. The compatibility, in this case, has a nice geometric interpretation: the Hamiltonian operator Q turns out to be a Killing–Poisson tensor with respect to the metric (g_{ij}) that forms the leading coefficient of the Hamiltonian operator P (see [24]).

Here, the execution time for the Schouten bracket $[P, P]$ is 0.48 s for `jacobi` and 0.54 s for the parallel version `jacobi_p`. This is the most complicated between the brackets that we calculate for this example, although still not challenging. For a significant difference between the non-parallel and the parallel version, see the next example.

2.3. The equations of associativity

In [8] the geometric theory of the equations of associativity, or Witten–Dijkgraaf–Verlinde–Verlinde equation, is developed in detail. One of the cases of equations of associativity can be rewritten as a quasilinear system of PDEs of the form $u^i_t = (V^i(\mathbf{u}))_x$ (Example 3 in [14]), with

$$u^1_t = (u^2 + u^3)_x, \quad u^2_t = \left(\frac{u^2(u^2+u^3)-1}{u^1}\right)_x, \quad u^3_t = u^1_x. \tag{20}$$

It can be proved that the above system admits the following operator of Ferapontov type [27]:

$$P = g^{ij}\partial_x + \Gamma^{ij}_k u^k_x + \alpha_1 \frac{\partial V^i}{\partial u^q} u^q_x \partial_x^{-1} \frac{\partial V^j}{\partial u^p} u^p_x + \gamma_1 u^i_x \partial_x^{-1} u^j_x, \tag{21}$$

where the metric in *upper* indices is

$$g^{ij} = \begin{pmatrix} u_1^2+u_2^2+2u_2u_3+u_3^2+1 & \frac{u_1^2u_2+u_2^3+2u_2^2u_3+u_2u_3^2-u_2-u_3}{u_1} & u_1(u_2+2u_3) \\ \frac{u_1^2u_2+u_2^3+2u_2^2u_3+u_2u_3^2-u_2-u_3}{u_1} & \frac{u_1^2u_2^2+4u_1^2+u_2^4+2u_2^3u_3+u_2^2u_3^2-2u_2^2-2u_2u_3+1}{u_1^2} & u_2^2+2u_2u_3-3 \\ u_1(u_2+2u_3) & u_2^2+2u_2u_3-3 & u_1^2+u_3^2+4 \end{pmatrix} \tag{22}$$

the Christoffel symbols are

$$\Gamma^{ij}_1 = \begin{pmatrix} u_1 & u_2 & u_3 \\ \frac{-(u_2^2+u_2u_3-1)(u_2+u_3)}{u_1^2} & \frac{-(u_2^2+u_2u_3-1)^2}{u_1^3} & \frac{-(u_2^2+u_2u_3-1)}{u_1} \\ u_2+u_3 & \frac{u_2^2+u_2u_3-1}{u_1} & u_1 \end{pmatrix} \tag{23}$$

$$\Gamma^{ij}_2 = \begin{pmatrix} u_2+u_3 & \frac{u_2^2+u_2u_3-1}{u_1} & u_1 \\ \frac{(2u_2+u_3)(u_2+u_3)+u_1^2}{u_1} & \frac{2u_2^3+3u_2^2u_3-u_3+u_1^2u_2+(u_3^2-2)u_2}{u_1^2} & 2(u_2+u_3) \\ 0 & 0 & 0 \end{pmatrix} \tag{24}$$

$$\Gamma^{ij}_3 = \begin{pmatrix} u_2+u_3 & \frac{u_2^2+u_2u_3-1}{u_1} & u_1 \\ \frac{(u_2+u_3)u_2}{u_1} & \frac{(u_2^2+u_2u_3-1)u_2}{u_1^2} & u_2 \\ u_1 & u_2 & u_3 \end{pmatrix} \tag{25}$$

and the value of the constants is $\alpha_1 = -1$, $\gamma_1 = -1$.

In Maple, we need to introduce the coefficients as three arrays

```
g := Matrix(N,N);
g[1,1] := u[1,x,0]^2 + (u[2,x,0] + u[3,x,0])^2 + 1;
```

and so on, and

```
Gamma := Array(1..N,1..N,1..N);
Gamma[1,1,1] := u[1,x,0];
```

and so on, and finally

```
W := Matrix(N,N);
W[1,1] := 0;
W[1,2] := 1;
W[1,3] := 1;
W[2,1] := -(u[2, x, 0]*(u[2, x, 0] + u[3, x, 0]) - 1)/u[1, x, 0]^2;
W[2,2] := (2*u[2, x, 0] + u[3, x, 0])/u[1, x, 0];
W[2,3] := u[2, x, 0]/u[1, x, 0];
W[3,1] := 1;
W[3,2] := 0;
W[3,3] := 0;
```

where `W[i,j]` is equal to $\partial V^i/\partial u^j$. Indeed, in this example the nonlocal part is made by two matrices, the other being the identity. Then, we define the operator as

```
P := Array(1..N,1..N,0..M);
for i to N do
  for j to N do
    P[i,j,0] := g[i,j]*delta[x - y,1]
      + add(Gamma[i,j,k]*u[k,x,1],k = 1..N)*delta[x - y,0]
      - add(add(W[i,s]*u[s,x,1]*delta[x - y,-1]
       *subs(x = y,W[j,t])*u[t,y,1],s = 1..N),t = 1..N)
      - u[i,x,1]*delta[x - y,-1]*u[j,y,1];
  end do;
end do;
```

The Schouten bracket is computed as usual and the result is 0.

Here, the execution time for the Schouten bracket $[P, P]$ is about 18 s for `jacobi` and about 9 s for the parallel version `jacobi_p`, showing that the parallel calculations are giving an effective reduction of the computation time with respect to the non-parallel implementation.

3. Reduce implementation

Now, we describe the implementation of the Schouten bracket for nonlocal differential operators in Reduce, as a part of the package CDE. The package CDE has been developed by one of us (RV) for calculations in the geometric theory of PDEs and integrability [18].

Consider two weakly nonlocal **skew-adjoint** operators P, Q:

$$P(\psi)^i = P^{ij}\psi_j = B^{ij\sigma}\partial_\sigma\psi_j + c^{\alpha\beta}w^i_\alpha\partial_x^{-1}(w^j_\beta\psi_j), \tag{26}$$

$$Q(\psi)^i = Q^{ij}\psi_j = C^{ij\sigma}\partial_\sigma\psi_j + d^{\alpha\beta}z^i_\alpha\partial_x^{-1}(z^j_\beta\psi_j). \tag{27}$$

For the nonlocal summands, the skew-adjointness is fulfilled if $c^{\alpha\beta}$ and $d^{\alpha\beta}$ are symmetric matrices.

Let us introduce the *nonlocal variables*

$$\tilde\psi^a_\alpha = \partial_x^{-1}(w^i_\alpha\psi^a_i), \qquad \tilde\chi^a_\beta = \partial_x^{-1}(z^i_\beta\psi^a_i), \tag{28}$$

where $a = 1, 2, 3$ and α and β run in the same finite set of indices (see [3,18] for a geometric theory of nonlocal variables).

We have the formula:

$$\ell_{P,\psi}(\varphi)^i = \frac{\partial B^{ij\sigma}}{\partial u^k_\tau}\partial_\sigma\psi^1_j\partial_\tau\varphi^k + c^{\alpha\beta}\frac{\partial w^i_\alpha}{\partial u^k_\tau}\partial_\tau\varphi^k\partial_x^{-1}(w^j_\beta\psi_j) + c^{\alpha\beta}w^i_\alpha\partial_x^{-1}\left(\frac{\partial w^j_\beta}{\partial u^k_\tau}\partial_\tau\varphi^k\psi_j\right), \tag{29}$$

where we used the fact that ∂_x^{-1} commutes with linearization. Then, we have the following expression for the Schouten bracket [7]:

$$\begin{aligned}[P, Q](\psi^1,\psi^2,\psi^3) =& [\ell_{P,\psi^1}(Q(\psi^2))(\psi^3) + \text{cyclic}(\psi^1,\psi^2,\psi^3)\\ &+ \ell_{Q,\psi^1}(P(\psi^2))(\psi^3) + \text{cyclic}(\psi^1,\psi^2,\psi^3)]\end{aligned} \tag{30}$$

where square brackets stand for horizontal cohomology [3], i.e. the expression should be considered up to total x-derivatives.

A single summand of the above formula has the expression

$$\begin{aligned}\ell_{P,\psi^1}(Q(\psi^2))(\psi^3) =& \frac{\partial B^{ij\sigma}}{\partial u^k_\tau}\partial_\sigma\psi^1_j\partial_\tau\left(C^{kp\sigma}\partial_\sigma\psi^2_p + d^{\alpha\beta}z^k_\alpha\tilde\chi^2_\beta\right)\psi^3_i\\ &+ c^{\alpha\beta}\frac{\partial w^i_\alpha}{\partial u^k_\tau}\partial_\tau\left(C^{kp\sigma}\partial_\sigma\psi^2_p + d^{\gamma\delta}z^k_\gamma\tilde\chi^2_\delta\right)\tilde\psi^1_\beta\psi^3_i\\ &- c^{\alpha\beta}\tilde\psi^3_\alpha\left(\frac{\partial w^j_\beta}{\partial u^k_\tau}\partial_\tau(C^{kp\sigma}\partial_\sigma\psi^2_p + d^{\gamma\delta}z^k_\gamma\tilde\chi^2_\delta)\psi^1_j\right)\end{aligned} \tag{31}$$

Here, the last summand has been obtained by integration by parts:

$$c^{\alpha\beta}w^i_\alpha\partial_x^{-1}\left(\frac{\partial w^j_\beta}{\partial u^k_\tau}\partial_\tau(C^{kp\sigma}\partial_\sigma\psi^2_p + d^{\gamma\delta}z^k_\gamma\tilde\chi^2_\delta)^k\psi^1_j\right)\psi^3_i = -c^{\alpha\beta}\tilde\psi^3_\alpha\left(\frac{\partial w^j_\beta}{\partial u^k_\tau}\partial_\tau(C^{kp\sigma}\partial_\sigma\psi^b_p + d^{\gamma\delta}z^k_\gamma\tilde\chi^2_\delta)\psi^1_j\right) + \partial_x(T), \tag{32}$$

where the term $\partial_x(T)$ is ignored in the bracket formula.

In Reduce, the above formulae (30) and (32) and the subsequent algorithm is implemented through CDE in the library file `cde_weaklynl.red`. We illustrate the implementation and the usage by means of the following program files, which provide the same computations as in the previous Section. The examples accompany the paper as Reduce program files. They can be run by the command

```
in "filename.red";
```

issued as the first command in a Reduce window.

3.1. The mKdV equation

The calculation is carried out with the operator P in (15). First of all, we load the package CDE:

```
load_package cde;
```

then we define the list of variables:

```
indep_var:={x};
dep_var_equ:={u};
loc_arg:={psi};
total_order:=10;
```

Here `indep_var` is the name of the independent variable, `dep_var_equ` is the list of dependent variables that are involved in the equation and `loc_arg` is a list of symbols that will be used for the local arguments of the three-vector (one symbol for each dependent variable in the equation). Finally, `total_order` is the maximal order of derivative that can appear in all expressions that will be input in and/or calculated by the program. The order can be computed more sharply with the formula that has been given in the Maple package: `M:=M1+D1+M2+D2`, but in any case CDE will issue an error and quit if the order turns out to be too low.

We load the operator P as follows. First of all, we define the local part of P using the CDE syntax [18]:

```
mk_cdiffop(ham_l,1,{1},1);
for all psi let ham_l(1,1,psi)=
  td(psi,x,3) + (2/3)*(u**2*td(psi,x) + u*u_x*psi);
```

Then we enter the nonlocal part by the commands

```
mk_wnlop(c,w,1);
c(1,1):= - 2/3;
w(1,1):=u_x;
```

The command `mk_wnlop` defines the coefficient matrix of the nonlocal summand $c^{\alpha\beta}$ as a Reduce operator `c`, where the value at the indices `alpha`, `beta` is `c(alpha,beta)`. It also defines the vectors w^i_α (for every index α) as the reduce operator `w`, where the value at the indices `i`, `alpha` is `w(i,alpha)`. The third entry of the command `mk_wnlop` is the number of summands w^i_α (for every index i), or the range of the index α.

The first weakly nonlocal operator is arranged in a list as follows:

```
ham1:={ham_l,c,w};
```

Since we are computing $[P, P]$, the second operator is equal to the first one:

```
ham2:=ham1;
```

The Reduce implementation makes use of nonlocal variables in order to calculate the Schouten bracket. For this reason, we need to form a list of all distinct non-local variables. There is one nonlocal variable for any value of α; in the mKdV case, the list has the format

```
nloc_var:={{tpsi,w,1}};
```

The jet space (*i.e.* all variables and their x-derivatives) is prepared by the command

```
dep_var_tot:=cde_weaklyn1(indep_var,dep_var_equ,loc_arg,nloc_var,
  total_order);
```

creates the jet space with:

1. derivative symbols like `u_x`, `u_2x`, ...;
2. three times the symbols in the list `loc_arg`, by appending `1, 2, 3` to the names of the arguments. In our case: `psi1, psi2, psi3` and their derivatives `psi1_x, psi2_x, ...`
3. three times the nonlocal variables in `nloc_arg`, i.e. `tpsi_11,tpsi_12,tpsi_13`, which correspond to the nonlocal variables $\tilde\psi^1_1$, $\tilde\psi^2_1$, $\tilde\psi^3_1$.

```
tpsi1_x = w(1,1)*psi1;
tpsi2_x = w(1,1)*psi2;
tpsi3_x = w(1,1)*psi3;
```

The above jet space is defined using well-tested CDE routines, after an initial consistency check on the input data (for example, the number of elements of `dep_var_equ` should be the same as `loc_arg`); the constraints are CDE differential equations and allow an immediate and automatic replacement with the right-hand side during the calculations.

The return argument is a list of all dependent variables, like `u, psi1, ...`, introduced by the command `cde_weaklynl`, in this case:

```
dep_var_tot := {{u},{psi1,psi2,psi3},{tpsi_11,tpsi_12,tpsi_13}};
```

Finally, we need a list of the two names that are used for nonlocal variables for the two operators; in this case, since we have only one operator, we will repeat the nonlocal variable two times:

```
nloc_arg:={{tpsi,w},{tpsi,w}};
```

The Schouten bracket is readily computed:

```
sb_res:=schouten_bracket_wnl(ham1,ham2,dep_var_equ,loc_arg,nloc_arg);
```

The return variable `sb_res` contains the expression of the three-vector $[P, P]$ after the reduction to the divergence-free normal form. The list of variables `dep_var_tot` might be used to extract coefficients of the three-vector `sb_res` and set them to zero if needed by the problem that is under consideration. In the mKdV case it is obviously zero.

Let us briefly explain how CDE calculates the divergence-free normal form.

The expression (30) is a three-vector which is defined up to total x-derivatives. It is processed by the software following the algorithm developed in [4] in the following steps.

1. The command `schouten_bracket_wnl` makes several consistency checks on the input, which should be declared as above.
2. Then, the control is taken by the symbolic procedure[4] `sb_wnl_algorithm` which, as a first step, generates from `nloc_arg` nonlocal variables which are functions of the arguments `psi1, psi2, psi3`. It is of utmost importance that `nloc_arg` contains all non-identical nonlocal variables; using two different names for the same nonlocal variable would result in no simplification and wrong results.
3. The control is taken now by the symbolic procedure `dubrovin_zhang_expr`, which computes the formula (30). The formula is built by blocks, one for each of the summands of (29) plus one for the argument $Q(\psi^2)$, with four distinct symbolic procedures used to that purpose, and the result is recomputed with cyclically permuted arguments ψ^1, ψ^2, ψ^3 two times, with the roles of P and Q exchanged.
4. The result is fed into a symbolic procedure, `nonlocal_reduction`, that collects all terms that consist of a coefficient C^{pi} (which can be an arbitrary function of u^i and its derivatives) that multiplies one of the terms

$$\tilde{\psi}^1_\alpha \psi^2_p \partial_x^k \psi^3_i, \qquad \tilde{\psi}^2_\alpha \psi^3_p \partial_x^k \psi^1_i, \qquad \tilde{\psi}^3_\alpha \psi^1_p \partial_x^k \psi^2_i, \tag{33}$$

   with $k > 0$, or similar terms with $\tilde{\chi}^a_\beta$, and replace them by

$$(-1)^k \partial_x^k (C^{pi} \tilde{\psi}^1_\alpha \psi^2_p) \psi^3_i \tag{34}$$

   and similar terms. In the end, the nonlocal part of the three-vector will be generated by

$$\tilde{\psi}^1_\alpha \partial_x^h \psi^2_p \psi^3_i, \qquad \tilde{\psi}^2_\alpha \partial_x^h \psi^3_p \psi^1_i, \qquad \tilde{\psi}^3_\alpha \partial_x^h \psi^1_p \psi^2_i, \tag{35}$$

   with $h \geq 0$, or similar terms with $\tilde{\chi}^a_\beta$.
5. Another symbolic procedure, `local_reduction`, does a similar job on terms of the form

$$\partial_x^k \psi^1_j \partial_x^h \psi^2_p \partial_x^l \psi^3_i \tag{36}$$

   with respect to ψ^3 when $l > 0$, so to obtain a local part which is of order 0 with respect to ψ^3.

The output of the command `schouten_bracket_wnl` is the three-vector $[P, Q]$ in the normal, divergence-free form. In this example, the output is zero. The elapsed time (measured by activating the Reduce switch `on time;`) is 30 ms. We stress that here and for the following examples we used CSL as the underlying Lisp system for Reduce (the other possibility being the PSL Lisp system, see [16]) on a laptop with an Intel i7-8565U processor.

3.2. The Heisenberg magnet equation

We describe a program file to calculate the Schouten bracket of P as in (16). We will only mention the differences between the program for the mKdV and this one. We introduce the dependent variables and the arguments of the operator:

```
dep_var_equ:={u1,u2};
loc_arg:={psi1,psi2};
```

Then we define the local part of the operator:

```
p:=(1/2)*(u1**2 + u2**2 + 1)$

mk_cdiffop(ham_l,1,{2},2);
```

[4] This is a Reduce concept meaning that the procedure operates on raw Rlisp data rather than algebraic data, see [25].

```
for all psi11 let ham_l(1,1,psi11)=
  p**2*td(psi11,x) + p*(u1*u1_x + u2*u2_x)*psi11;

for all psi12 let ham_l(1,2,psi12)=
  p*(u1*u2_x - u2*u1_x)*psi12;

for all psi21 let ham_l(2,1,psi21)=
  p*(u2*u1_x - u1*u2_x)*psi21;

for all psi22 let ham_l(2,2,psi22)=
  p**2*td(psi22,x) + p*(u1*u1_x + u2*u2_x)*psi22;
```

The nonlocal part contains only one 'tail' vector:

```
mk_wnlop(c,w,1);
c(1,1):=1;
w(1,1):=u1_x;
w(2,1):=u2_x;
```

The first and second operators in the bracket are loaded as

```
ham1:={ham_l,c,w};
ham2:=ham1;
```

There is only one nonlocal variable:

```
nloc_var:={{tpsi,w,1}};
```

The space of all variables and derivatives is generated by

```
dep_var_tot:=cde_weaklynl(indep_var,dep_var_equ,
  loc_arg,nloc_var,total_order);
```

The list of the two names of the nonlocal variables is

```
nloc_arg:={{tpsi,w},{tpsi,w}};
```

and the Schouten bracket $[P, P] = 0$ is calculated by:

```
sb_res:=schouten_bracket_wnl(ham1,ham2,dep_var_equ,
  loc_arg,nloc_arg);
```

The last command returns `0`; the elapsed time is 810 ms, plus 40 ms of garbage collection.

The Schouten bracket for the second operator Q (19) of the Heisenberg magnet equation (17) is calculated as follows. The operator is input as

```
p:=(1/2)*(u1**2 + u2**2 + 1);
mk_cdiffop(ham_2,1,{2},2);
for all psi11 let ham_2(1,1,psi11) = 0;
for all psi12 let ham_2(1,2,psi12) = - p**2*psi12;
for all psi21 let ham_2(2,1,psi21) = p**2*psi21;
for all psi22 let ham_2(2,2,psi22) = 0;
```

The operator is local, hence we need to set its nonlocal data to 0:

```
mk_wnlop(d,z,1);
d(1,1):=1;
z(1,1):=0;
z(2,1):=0;
```

The input for the two arguments of the Schouten bracket is formed as

```
ham1:={ham_2,d,z};
ham2:=ham1;
```

The jet space is generated by

```
nloc_var:={{tpsi,z,1}};
dep_var_tot:=cde_weaklynl(indep_var,dep_var_equ,
  loc_arg,nloc_var,total_order);
```

and the Schouten bracket is calculated by

```
nloc_arg:={{tpsi,z},{tpsi,z}};
sb_res:=schouten_bracket_wnl(ham1,ham2,dep_var_equ,
  loc_arg,nloc_arg);
```

with result 0 in just 30 ms.

The above Hamiltonian operators for the Heisenberg magnet equation are compatible: $[P, Q] = 0$. The operators are loaded as above in a single program file, with

```
ham1:={ham_1,c,w};
ham2:={ham_2,d,z};
```

there are two nonlocal variables (even if one is trivial, it must be fed into the program):

```
nloc_var:={{tpsi,w,1},{tchi,z,1}};
dep_var_tot:=cde_weaklynl(indep_var,dep_var_equ,
  loc_arg,nloc_var,total_order);
```

The Schouten bracket $[P, Q]$ is calculated by

```
nloc_arg:={{tpsi,w},{tchi,z}};
sb_res:=schouten_bracket_wnl(ham1,ham2,dep_var_equ,
  loc_arg,nloc_arg);
```

with 0 as a result in 300 ms.

3.3. The equations of associativity

Here we will compute the Schouten bracket $[P, P]$ for P as in (21). This provides an example of computation where the nonlocal part of the operator has two distinct nonlocal summands.

After loading the variables and the arguments

```
dep_var_equ:={u1,u2,u3};
loc_arg:={psi1,psi2,psi3};
```

we define the velocity matrix of the system of PDEs; in particular, the Reduce matrix element `av(i,j)` corresponds to $\partial V^i/\partial u^j_x$ in the system of PDEs $u^i_t = (V^i)_x$ (20).

```
% right-hand side of the system of PDEs
de:={
  u2_x + u3_x,
 - ( - 2*u1*u2*u2_x - u1*u2*u3_x - u1*u2_x*u3
     + u1_x*u2**2 + u1_x*u2*u3 - u1_x)/u1**2,
  u1_x
    }$

nc:=length(dep_var_equ)$
dv1:={u1_x,u2_x,u3_x}$

matrix av(nc,nc);
for i:=1:nc do
  for j:=1:nc do
    av(i,j):=df(part(de,i),part(dv1,j));
```

Then, we define the local part of the operator by

```
mk_cdiffop(ham_1,1,{3},3);
for all i,j,psi let ham_1(i,j,psi)=
  gu1(i,j)*td(psi,x) + b(i,j)*psi;
```

Here, `gu1` is a Reduce operator containing the metric of the operator P in upper indices, and `b` is a Reduce operator containing the value of $\Gamma^{ij}_k u^k_x$ that has been separately computed (see Subsection 2.3). The nonlocal part of the operator is loaded by

```
mk_wnlop(c,w,2);
c(1,1):= - 1;
c(2,2):= - 1;
c(1,2):=0;
c(2,1):=0;
for i:=1:nc do w(i,1):=(for j:=1:nc sum av(i,j)*part(dv1,j));
w(1,2):=u1_x;
w(2,2):=u2_x;
w(3,2):=u3_x;
```

The Reduce operator `c` contains the coefficients of the nonlocal summands. The second index of the Reduce operator `w` labels the 'tail' vector, and it corresponds to α in w^i_α. In this case we have just one operator in the bracket with two distinct nonlocal variables:

```
nloc_var:={{tpsi,w,1},{tpsi,w,2}};
```

The Schouten bracket is computed by

```
nloc_arg:={{tpsi,w},{tpsi,w}};
sb_res:=schouten_bracket_wnl(ham1,ham2,dep_var_equ,
  loc_arg,nloc_arg);
```

The result is `0`. Here the execution time is much higher, due to the algebraic complexity of the example: 21330 ms, with 1020 ms of garbage collection.

4. Mathematica implementation

We describe the implementation of the computation of the Jacobi identity for nonlocal Poisson vertex algebras in Mathematica. To this aim, let us start by reviewing some basic facts from [6] (see also [4]).

Let $\mathcal{V}$ be an algebra of differential functions in the variables $\{u^i\}_{i\in I}$ (cf. [2]). Recall from [6] that to a matrix pseudodifferential operator $P=\left(P^{ij}(\partial)\right)_{i,j\in I}\in \mathrm{Mat}_{\ell\times\ell}\,\mathcal{V}((\partial^{-1}))$ we associate a map, called (nonlocal) λ-bracket, $\{\cdot\,{}_\lambda\,\cdot\}_P:\mathcal{V}\times\mathcal{V}\to\mathcal{V}((\lambda^{-1}))$, given by the following *Master Formula*:

$$\{f_\lambda g\}_P=\sum_{\substack{i,j\in I\\ m,n\in\mathbb{Z}_+}}\frac{\partial g}{\partial u_j^{(n)}}(\lambda+\partial)^n P^{ji}(\lambda+\partial)(-\lambda-\partial)^m\frac{\partial f}{\partial u_i^{(m)}}\in\mathcal{V}((\lambda^{-1}))\,. \tag{37}$$

In particular, $\{u_{i\lambda}u_j\}_P=P^{ji}(\lambda)$, the symbol of the (j,i)-entry of the matrix pseudodifferential operator P. For arbitrary P, the λ-bracket (37) satisfies the following sesquilinearity conditions ($f,g\in\mathcal{V}$):

$$\{\partial f_\lambda g\}_P=-\lambda\{f_\lambda g\}_P\,,\qquad \{f_\lambda\partial g\}_P=(\lambda+\partial)\{f_\lambda g\}_P\,,$$

and left and right Leibniz rules ($f,g,h\in\mathcal{V}$):

$$\{f_\lambda gh\}_P=\{f_\lambda g\}_P h+\{f_\lambda h\}_P g\,,\qquad \{fg_\lambda h\}_P=\{f_{\lambda+\partial}h\}_{P\to}g+\{g_{\lambda+\partial}h\}_{P\to}f\,.$$

An expression of the form $\{f_{\lambda+\partial}h\}_{P\to}g$ is interpreted as follows: if $\{f_\lambda h\}_P=\sum_{n=-\infty}^N c_n\lambda^n$, then $\{f_{\lambda+\partial}h\}_{P\to}g=\sum_{n=-\infty}^N c_n(\lambda+\partial)^n g$, where we expand $(\lambda+\partial)^n$ in non-negative powers of ∂.

The matrix pseudodifferential operator P is skew adjoint if and only if ($f,g\in\mathcal{V}$)

$$\{f_\lambda g\}_P=-\{g_{-\lambda-\partial}f\}_P\,. \tag{38}$$

The RHS of the skewsymmetry condition should be interpreted as follows: we move $-\lambda-\partial$ to the left and we expand its powers in non-negative powers of ∂, acting on the coefficients on the λ-bracket.

In general, for all $f,g,h\in\mathcal{V}$ we have that $\{f_\lambda\{g_\mu h\}\}_P\in\mathcal{V}((\lambda^{-1}))((\mu^{-1}))$. Let $\mathcal{V}_{\lambda,\mu}:=\mathcal{V}[[\lambda^{-1},\mu^{-1},(\lambda+\mu)^{-1}]][\lambda,\mu]$, namely the quotient of the $\mathbb{C}[\lambda,\mu,\nu]$-module $\mathcal{V}[[\lambda^{-1},\mu^{-1},\nu^{-1}]][\lambda,\mu,\nu]$ by the submodule $(\nu-\lambda-\mu)\mathcal{V}[[\lambda^{-1},\mu^{-1},\nu^{-1}]]\,[\lambda,\mu,\nu]$. We have the natural embedding $\iota_{\mu,\lambda}:\mathcal{V}_{\lambda,\mu}\hookrightarrow V((\lambda^{-1}))((\mu^{-1}))$ defined by expanding the negative powers of $\nu=\lambda+\mu$ by geometric series in the domain $|\mu|>|\lambda|$.

The λ-bracket $\{\cdot\,{}_\lambda\,\cdot\}_P$ defined by (37) is called *admissible* if ($f,g,h\in\mathcal{V}$):

$$\{f_\lambda\{g_\mu h\}_P\}_P\in\mathcal{V}_{\lambda,\mu}\,, \tag{39}$$

where we identify the space $\mathcal{V}_{\lambda,\mu}$ with its image in $\mathcal{V}((\lambda^{-1}))((\mu^{-1}))$ via the embedding $\iota_{\mu,\lambda}$. It is proved in [6] that, if skewsymmetry (38) and admissibility (39) conditions hold, then we also have $\{g_\mu\{f_\lambda h\}_P\}_P\in\mathcal{V}_{\lambda,\mu}$ and $\{\{f_\lambda g\}_{P\lambda+\mu}h\}_P\in\mathcal{V}_{\lambda,\mu}$, for every $f,g,h\in\mathcal{V}$.

A skewsymmetric and admissible λ-bracket (37) defines a *(nonlocal) Poisson vertex algebra structure* on $\mathcal{V}$ if the following Jacobi identity is satisfied ($f,g,h\in\mathcal{V}$):

$$\{f_\lambda\{g_\mu h\}_P\}_P-\{g_\mu\{f_\lambda h\}_P\}_P-\{\{f_\lambda g\}_{P\lambda+\mu}h\}_P=0\,, \tag{40}$$

where the equality is understood in the space $\mathcal{V}_{\lambda,\mu}$. In this case P is called a *nonlocal Hamiltonian operator*.

Note that Jacobi identity (40) holds for all $f, g, h \in \mathcal{V}$ if and only if it holds for any triple of generators u^i, u^j, u^k, $i, j, k \in I$:

$$\{u^i{}_\lambda \{u^j{}_\mu u^k\}_P\}_P - \{u^j{}_\mu \{u^i{}_\lambda u^k\}_P\}_P - \{\{u^i{}_\lambda u^j\}_{P\,\lambda+\mu} u^k\}_P = 0\,. \tag{41}$$

Equation (41) is equivalent to the equation $[P, P] = 0$, where $[\cdot, \cdot]$ denotes the Schouten bracket of matrix pseudodifferential operators defined in (30).

Two nonlocal Hamiltonian operators P and Q are called compatible if their linear combination (or, equivalently, their sum) is a nonlocal Hamiltonian operator. According to the above discussion, this is equivalent to check that the λ-bracket $\{\cdot_\lambda \cdot\}_{P+Q} = \{\cdot_\lambda \cdot\}_P + \{\cdot_\lambda \cdot\}_Q$ defined by equation (37) satisfies Jacobi identity (41) on generators. This condition reads ($i, j, k \in I$):

$$\begin{aligned} &\{u^i{}_\lambda \{u^j{}_\mu u^k\}_P\}_Q + \{u^i{}_\lambda \{u^j{}_\mu u^k\}_Q\}_P - \{u^j{}_\mu \{u^i{}_\lambda u^k\}_P\}_Q - \{u^j{}_\mu \{u^i{}_\lambda u^k\}_Q\}_P \\ &- \{\{u^i{}_\lambda u^j\}_{P\,\lambda+\mu} u^k\}_Q - \{\{u^i{}_\lambda u^j\}_{Q\,\lambda+\mu} u^k\}_P = 0\,. \end{aligned} \tag{42}$$

Equation (42) is equivalent to the equation $[P, Q] = 0$.

Consider the weakly nonlocal skew-adjoint matrix pseudodifferential operators P and Q defined in equations (26) and (27). The corresponding λ-brackets on $\mathcal{V}$, defined by equation (37), on a pair of generators u^i, u^j, $i, j \in I$, are

$$\{u^j{}_\lambda u^i\}_P = P^{ij}(\lambda) = B^{ij\sigma}\lambda^\sigma + c^{\alpha\beta} w^i_\alpha (\lambda + \partial)^{-1} w^j_\beta\,, \tag{43}$$

$$\{u^j{}_\lambda u^i\}_Q = Q^{ij}(\lambda) = C^{ij\sigma}\lambda^\sigma + d^{\alpha\beta} z^i_\alpha (\lambda + \partial)^{-1} z^j_\beta\,. \tag{44}$$

In both equations (43) and (44) we should use the expansion

$$(\lambda + \partial)^{-1} = \sum_{n \in \mathbb{Z}_+} \lambda^{-n-1} (-\partial)^n\,,$$

where ∂ acts on coefficients on the right, to get elements in $\mathcal{V}((\lambda^{-1}))$.

Since the nonlocal matrix pseudodifferential operators P and Q, defined by equations (26) and (27) respectively, are rational, it follows from [6] that the λ-brackets (43) and (44) are admissible. In order to check that P (or Q) is a Hamiltonian operator we need to verify that equation (41) holds. To check that $[P, Q] = 0$, we need to verify that equation (42) holds.

In Mathematica the Master Formula (37) is implemented to compute each summand in the LHS of equations (41) and (42). Then the algorithm described in [4] to write these summands in a convenient basis of $\mathcal{V}_{\lambda,\mu}$ is implemented in order to check whether equations (41) and (42) hold. We illustrate the implementation and the usage in the three examples considered in Section 3.

4.1. The mKdV equation

Consider the mKdV equation $u_t = u^3 u_x + u_{xxx}$ and its nonlocal λ-bracket (cf. equation (15))

$$\{u_\lambda u\}_P = \lambda^3 + \frac{1}{3}(2\lambda + \partial)\, u^2 - \frac{2}{3} u_x (\lambda + \partial)^{-1} u_x. \tag{45}$$

The Hamiltonian property of P is proved by checking that the Jacobi identity (41) vanishes for the triple u, u, u.

We program the above calculation as follows. The package `nlPVA.wl` (with the classical implementation of the algorithm) or `nlPVA-par.wl` (with the parallel implementation) can be either stored locally, in a path where Mathematica is able to find them, or accessed remotely from the repository https://github.com/mcasati/nlPVA. This makes it possible using the packages on the web-based Wolfram Cloud and Mathematica online, as well as in any desktop version of the software (from 11.0) without installation. We import the package

```
Import["https://raw.githubusercontent.com/mcasati/nlPVA/main/nlPVA.wl"]
```

(or the parallel implementation `nlPVA-par.wl`). The classical and parallel implementation have the same input and the same final output. The parallelization of the code is performed automatically by Mathematica using the command `Parallelize[]`. We initialize the package settings by introducing the name u^i for the dependent variables and x for the independent variable using the commands `SetGenName` and `SetVarName`, and introducing a formal parameter β (shadowing the formal variable λ of the λ-bracket) using the command `SetFormalParameter` as follows:

```
SetGenName[u];
SetVarName[x];
SetFormalParameter[\[Beta]];
```

The formal parameter introduced will be used throughout the algorithm for internal computations and should be different from the variable λ that will be used for outputs.

In the case of the mKdV equation we have only one dependent variable $u := u^1$ and the corresponding λ-bracket (45) has a tail of order 1 (using the terminology in [12]). Moreover, we need to specify the highest derivative of the generators which will appear in the input with the command `SetMaxO` (its value is 5 by default, but in many examples this is unnecessarily high). We pass this information to the program as follows:

```
SetN[1];
SetTail[1];
SetMaxO[1];
```

Then we can load the λ-bracket (45). This is done in two steps. We start by defining its local part (the polynomial part in the variable λ):

```
PLoc = {{\[Beta]^3 + 2/3 gen[[1]] TD[gen[[1]]] + 2/3 gen[[1]]^2 \[Beta]}};
```

Note that it is important to define `PLoc` as a (in this case 1×1) matrix for the program to work. In the above definition, the variable β is used as the auxiliary variable to λ, `gen` is the vector containing the dependent variables, and the command `TD` is built-in in the program and it corresponds to the operator of total derivative (in this case, is the total derivative with respect to x). Then we enter the components of the nonlocal part of (45) (the term containing negative powers of λ) by the commands

```
c = {{-2/3}};
w = {{D[gen[[1]], x]}};
```

This is done by defining the coefficient matrix of the tail summand $c^{\alpha\beta}$ as a matrix `c` and of w^i_α as a matrix `w`, cf. (43). We combine the local and nonlocal part of the λ-bracket (45) using the command `BuildBracket`:

```
P = BuildBracket[PLoc, c, w, Nw];
```

The nonlocal part of (45) is understood by the program as a vector containing the matrices `c` and `w`, and a symbol `Nw` corresponding to the term $(\lambda + \partial)^{-1} w^j_\beta$. Note that, in the case of a purely local (respectively, purely nonlocal) λ-bracket, its nonlocal (respectively, local) part has to be initialized to 0. It is possible to visualize a matrix containing the full expression of the λ-brackets among generators in an explicit form as follows:

```
GetBracket[P,\[Lambda]];
```

The LHS of equation (41) can be computed in this case as follows:

```
CompatCheck[P, P, {\[Lambda],\[Mu],\[Nu]}];
```

The variables λ and μ appearing as input correspond to the variables appearing in the Jacobi identity (41). The variable ν is hidden from that identity since in the definition of $\mathcal{V}_{\lambda,\mu}$ in Section 4 we set $\nu = \lambda + \mu$.

The command `CompatCheck` implements the computation of the LHS of Jacobi identity (41) as follows. First, it computes all the three terms of that identity by using the Master Formula (37). This implementation is the same used in the Mathematica package `MasterPVA`, see [5]. Then, it applies a normalization procedure consisting in writing each summand in a convenient basis of $\mathcal{V}_{\lambda,\mu}$ implementing the algorithm described in [4]. After expressing each summand in the same basis, the program carries an easy computation of the LHS of equations (41). The running time of the procedure is, in general, heavily dependent on the maximal order of the generators' derivatives. It is worthy noticing, moreover, that the first time the command `CompatCheck` is used in the parallel implementation of the algorithm it takes significantly longer because Mathematica needs to initialize the additional kernels. It is possible to avoid this using the function `LaunchKernels[]` beforehand. The time required is computed by the Mathematica function `RepeatedTiming[]`. In a Mathematica 11.3 worksheet running on a laptop with an Intel i7-6500U processor and two available Mathematica kernels, the classical implementation takes 0.020 s, while the parallel one 0.028 s

4.2. The Heisenberg magnet equation

Let $P(\partial)$ be the matrix pseudodifferential operator defined in (16). The corresponding λ-brackets among generators are given by the following matrix equation

$$\begin{aligned}&\begin{pmatrix} \{u^1{}_\lambda u^1\}_P & \{u^2{}_\lambda u^1\}_P \\ \{u^1{}_\lambda u^2\}_P & \{u^2{}_\lambda u^2\}_P \end{pmatrix} = P(\lambda) \\ &= \begin{pmatrix} f(\lambda+\partial)f + u^1_x(\lambda+\partial)^{-1}u^1_x & f(u^1u^2_x - u^2u^1_x) + u^1_x(\lambda+\partial)^{-1}u^2_x \\ f(u^2u^1_x - u^1u^2_x) + u^2_x(\lambda+\partial)^{-1}u^1_x & f(\lambda+\partial)f + u^2_x(\lambda+\partial)^{-1}u^2_x \end{pmatrix},\end{aligned} \tag{46}$$

where $f = (1/2)((u^1)^2 + (u^2)^2 + 1)$.

Let us explain how to check that P is a nonlocal Hamiltonian operator. After loading the package and initializing the package settings as in Section 4.1 we introduce the number of dependent variables

```
SetN[2];
```

This command builds up a vector `gen` with two components corresponding to the dependent variables u^1 and u^2. The maximal order of derivatives of the generators in the two operators is 1 as in the previous example, so we do not need to reassign the value of `MaxO` (we can check that `GetMaxO[]` outputs 1). Then we define the local part of the operator:

```
f = 1/2(gen[[1]]^2+gen[[2]]^2+1);
```

```
PLoc = {{f^2 \[Beta]+f TD[P],
  f (gen[[1]] TD[gen[[2]]] - gen[[2]] TD[gen[[1]]])},
 {f (gen[[2]] TD[gen[[1]]] - gen[[1]] TD[gen[[2]]]),
  f^2 \[Beta]+f TD[f]} ;
```

The components of the nonlocal part are defined as follows:

```
c = {{1}};
w = {{TD[gen[[1]]], TD[gen[[2]]]}};
```

Then we combine the local and nonlocal parts of the λ-brackets (46):

```
P = BuildBracket[PLoc, c, w, Nw];
```

To check that P satisfies Jacobi identity we use

```
CompatCheck[P, P, {\[Lambda], \[Mu], \[Nu]}];
```

to compute the LHS of equation (41). Note that, by default, this commands prints as output the results of the partial steps of this computation corresponding to the different choices of the indices $i, j, k \in I$. This computation returns a set of four zeroes, corresponding to the four sets of indices (1,1,1), (1,1,2), (1,2,2), (2,2,2) and it is performed in 0.23 s using the classical implementation and 0.15 s using the parallel implementation.

Let $Q()$ denote the second Hamiltonian operator of the Heisenberg magnet equation defined in equation (19). The corresponding λ-brackets among generators are

$$\begin{pmatrix} \{u^1{}_\lambda u^1\}_Q & \{u^2{}_\lambda u^1\}_Q \\ \{u^1{}_\lambda u^2\}_Q & \{u^2{}_\lambda u^2\}_Q \end{pmatrix} = Q(\lambda) = \begin{pmatrix} 0 & -f^2 \\ f^2 & 0 \end{pmatrix}. \tag{47}$$

The compatibility condition between the non-local λ-bracket (46) and the local λ-bracket (47) is checked as follows. First, we define the λ-brackets (47) by

```
d = {{0, 0}, {0, 0}};
y = {{0, 0, 0}, {0, 0, 0}};
Q = BuildBracket[Qloc, d, y, Nz];
```

where `Qloc` is the matrix (47) separately introduced in Mathematica. As previously mentioned, in the case that the λ-brackets are local we still need to define their nonlocal tail to be zero for the program to work. Then, using the command

```
CompatCheck[P, Q, {\[Lambda], \[Mu], \[Nu]}];
```

one can check that the compatibility condition (42) is satisfied. The running time of this function is 0.11 s with the classical implementation and 0.081 s with the parallel one.

4.3. The equations of associativity

Consider the λ-brackets on generators associated to the weakly nonlocal matrix pseudodifferential operator $P(\partial)$ defined in (21) ($i, j = 1, 2, 3$)

$$\{u^j{}_\lambda u^i\}_P = g^{ij}\lambda + \Gamma^{ij}_k u^k_x + \alpha_1 \frac{\partial V^i}{\partial u^q} u^q_x (\lambda+\partial)^{-1} \frac{\partial V^j}{\partial u^p} u^p_x + \gamma_1 u^i_x (\lambda+\partial)^{-1} u^j_x, \tag{48}$$

where the metric g^{ij} is defined in (22), the Christoffel symbols Γ^{ij}_k are defined in (23), (24) and (25), and the value of the constants is $\alpha_1 = -1$, $\gamma_1 = -1$.

Let us show the required steps to check that P is a Hamiltonian operator. After loading the package and initializing its settings as in Section 4.1 we introduce the number of dependent variables and the length of the tail (which in this case is 2)

```
SetN[3];
SetTail[2];
```

We define the local part of the λ-brackets (48) by

```
PLoc = g \[Beta]  + \[CapitalGamma]1 TD[gen[[1]]] + \[CapitalGamma]2 TD[gen[[2]]]
       + \[CapitalGamma]3 TD[gen[[3]]];
```

Here, `g` is the matrix in (22), `Γ1` the matrix in (23), `Γ2` the matrix in (24) and `Γ3` the matrix in (25), which were previously separately introduced in Mathematica. The nonlocal part of the operator is loaded by

```
c = {{-1, 0}, {0, -1}};
w = {Table[Sum[D[V[[i]], gen[[s]]] TD[gen[[s]]],
    {s, $d}], {i, $d}],
   Table[TD[gen[[i]]], {i, $d}]};
```

The matrix `c` contains the coefficients of the nonlocal summands. The second index of matrix `w` labels the tail vector. The vector `V` contains the components V^i, needed to compute the velocity matrix $\partial V^i/\partial u^j$ of the system of PDEs (20). We combine the local and nonlocal part of the λ-brackets (48) as follows

```
P = BuildBracket[PLoc, c, w, Nw];
```

We check that Jacobi identity (41) holds using the command

```
CompatCheck[P, P, {\[Rho], \[Sigma], \[Tau]}];
```

(the choice of the output formal variables ρ, σ and τ does not affect the results provided that they are all distinct). The function outputs the expected list of ten zeroes (corresponding to independent entries of the Jacobi identity for a 3-components system) in 2.8 s when using the classical implementation and 1.8 s with the parallel one.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank A. Carbotti, E.V. Ferapontov, S. Perletti for scientific discussions.

This research has been funded by the Dept. of Mathematics and Physics “E. De Giorgi” of the Università del Salento, Istituto Naz. di Fisica Nucleare IS-CSN4 *Mathematical Methods of Nonlinear Physics*, GNFM of Istituto Nazionale di Alta Matematica. P. L. is supported by MIUR - FFABR funds 2017 and by funds of H2020-MSCA-RISE-2017 Project No. 778010 IPaDEGAN.

References

[1] M.J. Ablowitz, P.A. Clarkson, Solitons, Nonlinear Evolution Equations and Inverse Scattering, London Mathematical Society Lecture Note Series, vol. 149, Cambridge University Press, 1991.
[2] A. Barakat, A. De Sole, V.G. Kac, Jpn. J. Math. 4 (2009) 141–252.
[3] A.V. Bocharov, V.N. Chetverikov, S.V. Duzhin, N.G. Khor'kova, I.S. Krasil'shchik, A.V. Samokhin, Yu.N. Torkhov, A.M. Verbovetsky, A.M. Vinogradov, Symmetries and Conservation Laws for Differential Equations of Mathematical Physics, in: I.S. Krasil'shchik, A.M. Vinogradov (Eds.), Translations of Math. Monographs, vol. 182, Amer. Math. Soc., 1999.
[4] M. Casati, P. Lorenzoni, R. Vitolo, Stud. Appl. Math. 144 (4) (2020) 412–448, https://doi.org/10.1111/sapm.12302, https://arxiv.org/abs/1903.08204.
[5] M. Casati, D. Valeri, Boll. Unione Mat. Ital. 11 (4) (2018) 503–531, https://arxiv.org/abs/1603.05028.
[6] A. De Sole, V.G. Kac, Jpn. J. Math. 8 (2) (2013) 233–347.
[7] I.Ya. Dorfman, Dirac Structures and Integrability of Nonlinear Evolution Equations, John Wiley & Sons, 1993.
[8] B. Dubrovin, in: Integrable Systems and Quantum Groups, in: Lecture Notes in Math., vol. 1620, Springer, Berlin, 1996, pp. 120–348, arXiv:https://.org/abs/hep-th/9407018.
[9] B.A. Dubrovin, I.M. Krichever, S.P. Novikov, in: Dynamical Systems IV, 2 edition, in: Encyclopaedia of Mathematical Sciences, vol. 4, Springer-Verlag, Berlin, 2001, pp. 173–280.
[10] B.A. Dubrovin, Y. Zhang, Normal forms of integrable PDEs, Frobenius manifolds and Gromov-Witten invariants, arXiv:math.DG/0108160.
[11] L. Faddeev, L. Takhtajan, Hamiltonian Methods in the Theory of Solitons, Springer Verlag, 2007.
[12] E.V. Ferapontov, Am. Math. Soc. Transl. (2) 170 (1995) 33–58.
[13] E.V. Ferapontov, O.I. Mokhov, Uspekhi Math. Nauk 45 (3) (1990) 191–192, English translation in: Russ. Math. Surv. 45 (1990) 218–219.
[14] E.V. Ferapontov, M.V. Pavlov, R.F. Vitolo, Int. Math. Res. Not. 22 (2016) 6829–6855.
[15] The geometry of differential equations website, webpage of the software packages and examples discussed in this paper, https://gdeq.org/Weakly_nonlocal_Poisson_brackets, 2021.
[16] A.C. Hearn, Reduce, http://reduce-algebra.sourceforge.net/, 2004, version 3.8 edition, 2004. Computer algebra system, currently in development after that it has been released in 2008 as free software at Sourceforge. The manual is available at the website.
[17] P. Kersten, J. Krasil'shchik, A. Verbovetsky, Acta Appl. Math. 83 (2004) 167–173.
[18] J. Krasil'shchik, A. Verbovetsky, R. Vitolo, The Symbolic Computation of Integrability Structures for Partial Differential Equations, Texts and Monographs in Symbolic Computation, Springer, ISBN 978-3-319-71654-1, 2018; see http://gdeq.org/Symbolic_Book for downloading program files that are discussed in the book.
[19] P. Lorenzoni, Lett. Math. Phys. 67 (2004) 83–94.
[20] P. Lorenzoni, R. Vitolo, J. Geom. Phys. 149 (2020) 103573, https://arxiv.org/abs/1909.07695.
[21] F. Magri, J. Math. Phys. 19 (1978) 1156–1162.
[22] A.Ya. Maltsev, S.P. Novikov, Physica D 156 (2001) 53–80.
[23] A.G. Meshkov, Differ. Equ. Control Process. 1 (2002), http://www.math.spbu.ru/diffjournal/.
[24] O.I. Mokhov, Symplectic and Poisson Geometry on Loop Spaces of Smooth Manifolds and Integrable Equations, in: S.P. Novikov, I.M. Krichever (Eds.), Reviews in Mathematics and Mathematical Physics, vol. 11, Harwood Academic Publishers, 1998, pp. 1–128.
[25] A.C. Norman, R. Vitolo, Inside Reduce, 2014, part of the official Reduce documentation, included in the source code of Reduce. See also http://reduce-algebra.sourceforge.net/lisp-docs/insidereduce.pdf.
[26] S. Novikov, S.V. Manakov, L.P. Pitaevskii, V.E. Zakharov, Theory of Solitons. The Inverse Scattering Method, Springer, 1984.
[27] J. Vašíček, R. Vitolo, J. High Energy Phys. 2021 (2021) 129, https://arxiv.org/abs/2104.13206.
[28] R. Vitolo, Comput. Phys. Commun. 244 (2019) 228–245, https://arxiv.org/abs/1808.03902.
[29] J.P. Wang, J. Nonlinear Math. Phys. 9 (2002) 213–233.
[30] V.E. Zakharov (Ed.), What Is Integrability?, Springer-Verlag, Berlin, 1991.