Abstract. We present a computational toolkit for (local) Poisson-Nijenhuis calculus on manifolds. Our python module PoissonGeometry implements our algorithms, and accompanies this paper. We include two examples of how our methods can be used, one for gauge transformations of Poisson bivectors in dimension 3, and a second one that determines parametric Poisson bivector fields in dimension 4.

Key words. Poisson structures, Poisson-Nijenhuis calculus, Symbolic computation, Python.

AMS subject classifications. 68W30, 97N80, 53D17

1. Introduction. The origin of the concepts in this paper is the analysis of mechanical systems of Siméon Denis Poisson in 1809 [28]. A Poisson manifold is a pair $(M, \Pi)$, with $M$ a smooth manifold and $\Pi$ a contravariant 2-tensor field (bivector field) on $M$ satisfying the equation

\begin{equation}
[\Pi, \Pi] = 0,
\end{equation}

with respect to the Schouten-Nijenhuis bracket $[\ , \ ]$ for multivector fields [26, 10]. Suppose $m = \dim M$, fix a local coordinate system $x = (U; x^1, \ldots, x^m)$ on $M$. Then $\Pi$ has the following coordinate representation [22, 32]:

\begin{equation}
\Pi = \frac{1}{2} \Pi^{ij} \frac{\partial}{\partial x^i} \wedge \frac{\partial}{\partial x^j} = \sum_{1 \leq i < j \leq m} \Pi^{ij} \frac{\partial}{\partial x^i} \wedge \frac{\partial}{\partial x^j}
\end{equation}

Here, the functions $\Pi^{ij} = \Pi^{ij}(x) \in C^\infty_U$ are called the coefficients of $\Pi$, and $\{\partial/\partial x^i\}$ is the canonical basis for vector fields on $U \subseteq M$.

The Poisson bivector, and its associated bracket, are essential elements in the comprehension of Hamiltonian dynamics [23, 10]. We recommend interested readers consult the available surveys of this field [34, 18].

Table 1 below compiles the functions in our Python module PoissonGeometry$^1$ and their corresponding algorithm, and examples where such objects are used in the references. We describe all of our algorithms in section 2. In section 3 we present two applications that illustrate the usefulness of our computational methods. These are, a new result about gauge transformations of Poisson bivector fields in dimension 3 (Proposition 3.1), and a description of parametric families of Poisson bivectors in dimension 4 (Lemma 3.2).

---

$^*$This research was partially supported by CONACyT, “Programa para un Avance Global e Integrado de la Matemática Mexicana” FORDECYT 265667 and UNAM-DGAPA-PAPIIT-IN104819.

$^1$Instituto de Matemáticas, Universidad Nacional Autónoma de México (UNAM), México.

1 miguel.eva.alv @matem.unam.mx, 2 jcpanta @im.unam.mx, 3 pablo @im.unam.mx.

$^1$Our code repository is found at: https://github.com/appliedgeometry/poissongeometry.
| Function                        | Algorithm | Examples   |
|--------------------------------|-----------|------------|
| sharp_morphism                 | 2.1       | [10, 23, 6]|
| poisson_bracket                | 2.2       | [23, 6]    |
| hamiltonian_vf                 | 2.3       | [6, 31]    |
| lichnerowicz_poisson_operator  | 2.4       | [27, 2]    |
| curl_operator                  | 2.5       | [9, 2]     |
| bivector_to_matrix             | 2.6       | [10, 23, 6]|
| jacobiator                     | 2.7       | [10, 23, 6]|
| modular_vf                     | 2.8       | [1, 16, 2] |
| is_homogeneous_unimodular      | 2.9       | [9, 23, 2, 6]|
| one_forms_bracket              | 2.10      | [12, 18]   |
| gauge_transformation           | 2.11      | [7]        |
| linear_normal_form_R3          | 2.12      | [27, 6]    |
| isomorphic_lie_poisson_R3      | 2.13      | [27, 6]    |
| flaschka_ratiu_bivector        | 2.14      | [9, 14, 30, 11]|
| is_poisson_tensor              | 2.15      | [14, 30, 11]|
| is_in_kernel                   | 2.16      | [10, 23, 2, 6]|
| is_casimir                      | 2.17      | [9, 14, 30, 11]|
| is_poisson_vf                  | 2.18      | [27, 3]    |
| is_poisson_pair                | 2.19      | [4, 2]     |

Table 1: Functions, corresponding algorithms, and examples where each particular method can be, or has been, applied. The following diagram illustrates functional dependencies in PoissonGeometry.
2. Implementation of Functions in \texttt{PoissonGeometry}. In this section we describe the implementation of all functions of the module \texttt{PoissonGeometry}.

2.1. Key Functions. This subsection contains functions that serve as a basis for the implementation of almost all functions of \texttt{PoissonGeometry}.

2.1.1. Sharp Morphism. The function \texttt{sharp_morphism} computes the image of a differential 1-form under the vector bundle morphism $\Pi^\flat : \mathcal{T}^*M \to \mathcal{T}M$ induced by a bivector field $\Pi$ on $M$ and defined by

\begin{equation}
\langle \beta, \Pi^\flat(\alpha) \rangle := \Pi(\alpha, \beta),
\end{equation}

for any $\alpha, \beta \in \mathcal{T}^*M$ \cite{10, 23}. Here, $\langle \cdot, \cdot \rangle$ is the natural pairing for differential 1-forms and vector fields. Equivalently, $\Pi^\flat(\alpha) = i_\alpha \Pi$, with $i_\alpha$ the interior product of multivector fields and differential forms defined by the rule $i_{i_\alpha \alpha, \beta} := i_\alpha \circ i_\beta$ \cite{19}. Analogously for vector fields. In local coordinates, if $\alpha = \alpha_j \, dx^j$, $j = 1, \ldots, m$, then

\begin{equation}
\Pi^\flat(\alpha) = \sum_{1 \leq i < j \leq m} \alpha_i \Pi^{ij} \frac{\partial}{\partial x^j} - \alpha_j \Pi^{ij} \frac{\partial}{\partial x^i}.
\end{equation}
Algorithm 2.1 sharp_morphism(bivector, one_form)

Input: a bivector field and a differential 1-form

Output: a vector field which is the image of the differential 1-form under the vector bundle morphism (2.1) induced by the bivector field

1: **procedure**
2: \( m \leftarrow \) dimension of the manifold \( \triangleright \) Given by an instance of PoissonGeometry
3:  
4: \( \text{bivector} \leftarrow \) a dictionary \{ (1, 2): \( \Pi^{12} \), ..., \((m-1, m): \Pi^{m-1,m} \}\) that represents a bivector field according to (3.1)
5:  
6: \( \text{one\_form} \leftarrow \) a dictionary \{ (1): \( \alpha_1 \), ..., \((m): \alpha_m \}\) that represents a differential 1-form according to (3.1)
7:  
8: \( \text{sharp\_dict} \leftarrow \) the dictionary \{ (1): 0, ..., (m): 0 \}
9:  
10: **for** each \( 1 \leq i < j \leq m \) \( \triangleright \) Compute the sum in (2.1)
11:  
12: \( \text{sharp\_dict}(i) \leftarrow \text{sharp\_dict}(i) - \alpha_j \ast \Pi^{ij} \)
13:  
14: \( \text{sharp\_dict}(j) \leftarrow \text{sharp\_dict}(j) + \alpha_i \ast \Pi^{ij} \)
15:  
16: **end for**
17:  
18: if all values in \( \text{sharp\_dict} \) are equal to zero then
19:  
20: return \{0: 0\} \( \triangleright \) A dictionary with zero key and value
21:  
22: else
23:  
24: return \( \text{sharp\_dict} \)
25:  
26: end if
27:  
28: **end procedure**

Observe that the morphism (2.1) is defined, in particular, for Poisson bivector fields. So the function *sharp_morphism* can be applied on this class of bivector fields.

2.1.2. Poisson Brackets. A Poisson bracket on \( M \) is a Lie bracket structure \{,\} on the space of smooth functions \( C^\infty_M \) which is compatible with the pointwise product by the Leibniz rule [10, 23]. Explicitly, the Poisson bracket induced by a Poisson bivector field \( \Pi \) on \( M \) is given by the formula

\[
\{f, g\}_\Pi = \langle d\Pi \delta df, dg \rangle = (\Pi \delta df)^i \frac{\partial g}{\partial x^i}, \quad \forall f, g \in C^\infty_M;
\]

for \( i = 1, \ldots, m \). The function *poisson_bracket* computes the poisson bracket, induced by a Poisson bivector field, of two scalar functions.
Algorithm 2.2 poisson_bracket(bivector, function_1, function_2)

Input: a Poisson bivector field and two scalar functions
Output: the Poisson bracket of the two scalar functions induced by the Poisson bivector field

1: procedure
2: \( m \leftarrow \text{dimension of the manifold} \quad \triangleright \text{Given by an instance of PoissonGeometry} \)
3: \( \text{bivector} \leftarrow \text{a dictionary that represents a bivector field according to (3.1)} \)
4: \( \text{function}_1, \text{function}_2 \leftarrow \text{string expressions} \quad \triangleright \text{Represents scalar functions} \)
5: \( \text{if } \text{function}_1 == \text{function}_2 \text{ then} \)
6: \( \text{return } 0 \quad \triangleright \text{If } f = g \text{ in (2.3), then } \{ f, g \} = 0 \)
7: \( \text{else} \)
8: \( \text{Convert } \text{function}_1 \text{ and } \text{function}_2 \text{ to symbolic expressions} \)
9: \( \text{gradient}_\text{function}_1 \leftarrow \text{a dictionary that represents the gradient vector field of } \text{function}_1 \text{ according to (3.1)} \)
10: \( \text{sharp}_\text{function}_1 \leftarrow \text{sharp_morphism(bivector, gradient}_\text{function}_1) \quad \triangleright \text{See Algorithm 2.1} \)
11: \( \text{bracket} \leftarrow 0 \)
12: \( \text{for } i = 1 \text{ to } m \text{ do} \)
13: \( \text{bracket} \leftarrow \text{bracket} + \text{sharp}_\text{function}_1[i] * \partial(\text{function}_2)/\partial x_i \quad \triangleright \text{See (2.3)} \)
14: \( \text{end for} \)
15: \( \text{return } \text{bracket} \)
16: \( \text{end if} \)
17: \( \text{end procedure} \)

2.1.3. Hamiltonian Vector Fields. The function hamiltonian_vf computes the Hamiltonian vector field

\[ X_h := \Pi^\sharp(\text{dh}). \] (2.4)

of a function \( h \in C^\infty_M \) respect to a Poisson bivector field \( \Pi \) on \( M \) [10, 23].

Algorithm 2.3 hamiltonian_vf(bivector, hamiltonian_function)

Input: a Poisson bivector field and a scalar function
Output: the Hamiltonian vector field of the scalar function relative to the Poisson bivector field

1: procedure
2: \( m \leftarrow \text{dimension of the manifold} \quad \triangleright \text{Given by an instance of PoissonGeometry} \)
3: \( \text{bivector} \leftarrow \text{a dictionary that represents a bivector field according to (3.1)} \)
4: \( \text{hamiltonian_function} \leftarrow \text{a string expression} \quad \triangleright \text{Represents a scalar function} \)
5: \( \text{if } \text{hamiltonian_function} == \text{null} \text{ then} \)
6: \( \text{return } 0 \quad \triangleright \text{If } h = 0 \text{ in (2.3), then } X_h = 0 \)
7: \( \text{else} \)
8: \( \text{Convert } \text{hamiltonian_function} \text{ to symbolic expression} \)
9: \( \text{gradient}_\text{hamiltonian_function} \leftarrow \text{a dictionary that represents the gradient vector field of } \text{hamiltonian_function} \text{ according to (3.1)} \)
10: \( \text{sharp}_\text{hamiltonian_function} \leftarrow \text{sharp_morphism(bivector, gradient}_\text{hamiltonian_function}) \quad \triangleright \text{See Algorithm 2.1} \)
11: \( \text{bracket} \leftarrow 0 \)
12: \( \text{for } i = 1 \text{ to } m \text{ do} \)
13: \( \text{bracket} \leftarrow \text{bracket} + \text{sharp}_\text{hamiltonian_function}[i] * \partial(\text{hamiltonian_function})/\partial x_i \quad \triangleright \text{See (2.3)} \)
14: \( \text{end for} \)
15: \( \text{return } \text{bracket} \)
16: \( \text{end if} \)
17: \( \text{end procedure} \)
2.1.4. Coboundary Operator. The adjoint operator of a Poisson bivector field \( \Pi \) on \( M \) with respect to the Schouten-Nijenhuis bracket gives rise to a cochain complex \( (\Gamma \wedge TM, \delta) \), called the Lichnerowicz-Poisson complex of \((M, \Pi)\) [22, 10, 23]. Here \( \delta_\Pi : \Gamma \wedge^* TM \to \Gamma \wedge^{*+1} TM \) is the coboundary operator \((\delta_\Pi^2 = 0)\) defined by

\[
\delta_\Pi(A) := [\Pi, A], \quad \forall A \in \Gamma \wedge TM.
\]

Here, \( \Gamma \wedge TM \) denotes the \( C_M^\infty \)–module of multivector fields on \( M \). Explicitly, if \( a = \deg A \), then for any \( f_1, \ldots, f_{a+1} \in C_M^\infty \):

\[
[\Pi, A](df_1, \ldots, df_{a+1}) = \sum_{k=1}^{a+1} (-1)^{k+1} \{ f_k, A(df_1, \ldots, df_k, \ldots, df_{a+1}) \}_\Pi
\]

\[
+ \sum_{1 \leq k < l \leq a+1} (-1)^{k+l} A(df_k, df_l, \ldots, df_{a+1})
\]

Throughout this paper the symbol \( \wedge \) will denote the absence of the corresponding factor. In particular, if \( f_1 = x^1, \ldots, f_{a+1} = x^{a+1} \) are local coordinates on \( M \), we have that for \( 1 \leq i_1 < \cdots < i_{a+1} \leq m \):

\[
[\Pi, A]^{i_1 \cdots i_{a+1}} = \sum_{k=1}^{a+1} (-1)^{k+1} \{ x^{i_k}, A^{i_1 \cdots i_k \cdots i_{a+1}} \}_\Pi
\]

\[
+ \sum_{1 \leq k < l \leq a+1} (-1)^{k+l} \frac{\partial^{i_k i_l}}{\partial x^s} A^{s i_1 \cdots i_k \cdots i_{a+1}}
\]

Here \( [\Pi, A]^{i_1 \cdots i_{a+1}} \) and \( A^{i_1 \cdots i_{a+1}} \) are the coefficients of the coordinate expressions of \([\Pi, A] \), \( \hat{A} \) and \( \Pi \), in that order. The function \texttt{lichnerowicz_poisson_operator} computes the image of a multivector field under the coboundary operator induced by a Poisson bivector field.

\begin{algorithm}[h]
\caption{\texttt{lichnerowicz_poisson_operator}(bivector, multivector)}
\begin{algorithmic}[1]
\Statex \textbf{Input:} a Poisson bivector field and a multivector field
\Statex \textbf{Output:} the image of the multivector field under the coboundary operator \((2.5)\)
\Statex \textbf{induced by the Poisson bivector field}
\Statex
\Procedure{} \Statex \texttt{Given} an instance of \texttt{PoissonGeometry}
\State \texttt{m} \leftarrow \text{dimension of the manifold}
\State \texttt{a} \leftarrow \text{degree of the multivector field}
\State \texttt{bivector} \leftarrow \text{a dictionary} \{ (1, 2): \Pi^{12}, \ldots, (m-1, m): \Pi^{m-1m} \} \text{ that represents a bivector field according to} \((3.1)\)
\State \texttt{multivector} \leftarrow \text{a dictionary} \{ (1, \ldots, a): A^{1 \cdots a}, \ldots, (m-a+1, \ldots, m): A^{m-a+1 \cdots m} \} \text{ that represents a bivector field according to} \((3.1)\) or a string expression
\EndProcedure
\end{algorithmic}
\end{algorithm}
6: if $a + 1 > m$ then \( \triangleright \deg [H, A] = \deg (A) + 1 \) in (2.5)
7: \quad return \{0: 0\} \( \triangleright \) A dictionary with zero key and value
8: else if multivector is a string expression then
9: \quad return hamiltonian_vf(bivector, str(-1*) + multivector)
\( \triangleright \) See Algorithm 2.3
10: else
11: \quad CONVERT each value in bivector and in multivector to symbolic expression
12: \quad image_multivector \( \leftarrow \) the dictionary \{\( (1, \ldots, a + 1): \ldots, (m - a, \ldots, m): 0 \}\}
13: \quad for each \( 1 \leq i_1 < \cdots < i_{a+1} \leq m \) do
14: \quad \quad for \( k = 1 \) to \( a+1 \) do \( \triangleright \) Compute first summation in (2.6)
15: \quad \quad \quad image_multivector[\( \{i_1, \ldots, i_{a+1}\}\}] \( \leftarrow \) image_multivector[\( \{i_1, \ldots, i_{a+1}\}\}]
16: \quad \quad \quad + (-1)^{k+1} \{x_{i_k}, A_{i_1 \cdots i_{k-1}i_{k+1}}\} \_\Pi \( \triangleright \) See Algorithm 2.2 to compute the Poisson bracket \{.\} \_\Pi
17: \quad \quad end for
18: \quad \quad for each \( 1 \leq k < l \leq a+1 \) do \( \triangleright \) Compute second summation in (2.6)
19: \quad \quad \quad image_multivector[\( \{i_1, \ldots, i_{a+1}\}\}] \( \leftarrow \) image_multivector[\( \{i_1, \ldots, i_{a+1}\}\}]
20: \quad \quad \quad + (-1)^{k+l} \{\frac{\partial A_{i_1 \cdots i_{k-1}i_{k+1}i_{l-1} \cdots i_{l+1}}}{\partial x_{i_k}}, \frac{\partial A_{i_1 \cdots i_{k-1}i_{l-1} \cdots i_{l+1}}}{\partial x_{i_k}}\} \_\Pi
21: \quad \quad end for
22: \quad return \{0: 0\} \( \triangleright \) A dictionary with zero key and value
23: else
24: \quad return image_multivector
25: end if
26: end if
27: end procedure

2.1.5. Curl (Divergence) Operator. Fix a volume form \( \Omega_0 \) on an oriented Poisson manifold \((M, \Pi, \Omega_0)\). The divergence (relative to \( \Omega_0 \)) of an \( a \)-multivector field \( A \) on \( M \) is the unique \((a - 1)\)-multivector field \( \mathcal{D}_{\Omega_0}(A) \) on \( M \) such that

\[
(2.8) \quad i_{\mathcal{D}_{\Omega_0}(A)} \Omega_0 = dA \Omega_0.
\]

This induces a (well defined, \( \Omega_0 \)-dependent) coboundary operator \( \mathcal{D}_{\Omega_0} : A \rightarrow \mathcal{D}_{\Omega_0}(A) \) on the module of multivector fields on \( M \), called the curl operator [19, 23]. As any other volume form on \( M \) is a multiple \( f \Omega_0 \) of \( \Omega_0 \) by a nowhere vanishing function \( f \in C_c^\infty(M) \), we have \( \mathcal{D}_{f \Omega_0} = \mathcal{D}_{\Omega_0} + \frac{1}{f} \Pi_d f \). In local coordinates, expressing \( \Omega_0 \) as

\[
(2.9) \quad \Omega_0 = dx^1 \wedge \cdots \wedge dx^m,
\]

then for any \( a \)-multivector field \( A = A_{i_1 \cdots i_a} \partial / \partial x^{i_1} \wedge \cdots \wedge \partial / \partial x^{i_a} \) on \( M \), with \( 1 \leq i_1 < \cdots < i_a \leq m \), the divergence of \( A \) with respect to the volume form \( f \Omega_0 \) is given by:

\[
(2.10) \quad \mathcal{D}_f \Omega_0(A) = \sum_{k=1}^m (-1)^{k+1} \left( \frac{\partial A_{i_1 \cdots i_k}}{\partial x^{i_k}} + \frac{1}{f} \frac{\partial f}{\partial x^{i_k}} A_{i_1 \cdots i_k} \right) \frac{\partial}{\partial x^{i_1}} \wedge \cdots \wedge \frac{\partial}{\partial x^{i_k}} \wedge \cdots \wedge \frac{\partial}{\partial x^{i_a}}
\]

Let \( f \) be a nonzero scalar function. The function curl_operator computes the divergence of a multivector field respect to the volume form \( f \Omega_0 \), for \( \Omega_0 \) in (2.9).
Algorithm 2.5 \texttt{curl\_operator}(\texttt{multivector, function})

Input: a multivector field and a nonzero scalar function $f_0$
Output: the divergence of the multivector field with respect to the volume form $f_0\Omega_0$

1: \textbf{procedure}
2: \hspace{1em} $m \leftarrow$ dimension of the manifold \hspace{1em} $\triangleright$ Given by an instance of \texttt{PoissonGeometry}
3: \hspace{1em} $a \leftarrow$ degree of the multivector field
4: \hspace{1em} \texttt{multivector} $\leftarrow$ a dictionary \{$(1,...,a)$: $A^{1\cdots a}$, ..., $(m-a+1,...,m)$: $A^{m-a+1\cdots m}$\}
   \hspace{1em} that represents a multivector field according to (3.1) or a string expression
5: \hspace{1em} \texttt{function} $\leftarrow$ a string expression \hspace{1em} $\triangleright$ Represents a nonzero function
6: \hspace{1em} \textbf{if} \texttt{multivector} is a string expression then
7: \hspace{2em} return \{0: 0\} \hspace{1em} \hspace{1em} $\triangleright$ A dictionary with zero key and value
8: \hspace{1em} \textbf{else}
9: \hspace{2em} \textbf{for} each $1 \leq i_1 < \cdots < i_a \leq m$ do \hspace{1em} $\triangleright$ Compute the summation in (2.10)
10: \hspace{3em} for $k = 1$ to $m$ do
11: \hspace{4em} \texttt{curl\_multivec}[$(i_1, \ldots, \hat{i}_k, \ldots, i_a)$] $\leftarrow$ \texttt{curl\_multivec}[$(i_1, \ldots, \hat{i}_k, \ldots, i_a)$] + $(-1)^{k+1} \left( \frac{\partial A^{1\cdots i_{a-1}}}{\partial x_{i_1}} + \frac{1}{\texttt{function}} \frac{\partial (\texttt{function})}{\partial x_{i_1}} \right) A^{i_1\cdots i_a}$
12: \hspace{3em} \textbf{end for}
13: \hspace{2em} \textbf{end for}
14: \hspace{1em} \textbf{if} all values in \texttt{curl\_multivec} are equal to zero then
15: \hspace{2em} return \{0: 0\} \hspace{1em} \hspace{1em} $\triangleright$ A dictionary with zero key and value
16: \hspace{1em} \textbf{else}
17: \hspace{2em} return \texttt{curl\_multivec}
18: \hspace{1em} \textbf{end if}
19: \textbf{end if}
20: \textbf{end procedure}

2.2. Matrix of a bivector field. The function \texttt{bivector\_to\_matrix} computes the (local) matrix $[\Pi^i]$ of a bivector field $\Pi$ on $M$ \cite{10, 23}, the coefficients of $\Pi$ in (1.2). In particular, it computes the matrix of a Poisson bivector field.

Algorithm 2.6 \texttt{bivector\_to\_matrix}(\texttt{bivector})

Input: a bivector field
Output: the (local) matrix of the bivector field

1: \textbf{procedure}
2: \hspace{1em} $m \leftarrow$ dimension of the manifold \hspace{1em} $\triangleright$ Given by an instance of \texttt{PoissonGeometry}
3: \hspace{1em} \texttt{bivector} $\leftarrow$ a dictionary \{$(1,2)$: $\Pi^{12}$, ..., $(m-1,m)$: $\Pi^{m-1\cdots m}$\} that represents a bivector field according to (3.1)
4: \hspace{1em} \texttt{matrix} $\leftarrow$ a symbolic $m \times m$-matrix
5: \hspace{1em} \textbf{for} each $1 \leq i < j \leq m$ do
6: \hspace{2em} \textbf{begin} \texttt{convert} $\Pi^{ij}$ to symbolic expression
7: \hspace{3em} \texttt{matrix}[$i-1, j-1$] $\leftarrow$ $\Pi^{ij}$
8: \hspace{3em} \texttt{matrix}[$j-1, i-1$] $\leftarrow$ $(-1)^{i+j} \texttt{matrix}[$i-1, j-1$]
9: \hspace{2em} \textbf{end for}
### 2.3. Jacobiator.

The Schouten-Nijenhuis bracket of a bivector field $\Pi$ with itself, $[\Pi, \Pi]$, is computed with the `jacobiator` function. This 3-multivector field is called the Jacobiator of $\Pi$. The Jacobi identity (1.1) for $\Pi$ follows from the vanishing of its Jacobiator [10, 23].

#### Algorithm 2.7 `jacobiator(bivector)`

**Input:** a bivector field

**Output:** the Schouten-Nijenhuis bracket of the bivector field with itself

1: procedure
2: \[ \text{bivector} \leftarrow \text{a dictionary that represents a bivector field according to (3.1)} \]
3: \[ \text{return lichnerowicz.poisson.operator(bivector, bivector)} \rightarrow \text{See Algorithm 2.4} \]
4: end procedure

### 2.4. Modular Vector Field.

For $(M, \Pi, \Omega)$ an orientable Poisson manifold, and a fixed volume form $\Omega$ on $M$, the map

\[
Z : h \mapsto \mathcal{D}_\Omega(X_h)
\]

is a derivation of $C^\infty_M$. Therefore it defines a vector field on $M$, called the **modular vector field** of $\Pi$ relative to $\Omega$ [33, 1, 10, 23]. Here, $\mathcal{D}_\Omega$ is the curl operator relative to $\Omega$ (2.8). Then, $Z$ is a Poisson vector field of $\Pi$ which is independent of the choice of a volume form, modulo Hamiltonian vector fields: $Z_{f\Omega} = Z - \frac{1}{f}X_f$. Here $Z_{f\Omega}$ is the modular vector field of $\Pi$ relative to the volume form $f\Omega$ and $f \in C^\infty_M$ a nowhere vanishing function. In this context, the Poisson bivector field $\Pi$ is said to be **unimodular** if $Z$ is a Hamiltonian vector field (2.4). Equivalently, if $Z$ is zero for some volume form on $M$. We can compute the modular vector field $Z$ of $\Pi$ (2.11) relative to a volume form $f\Omega$ as the (minus) divergence of $\Pi$ (2.10):

\[
Z_{f\Omega} = -\mathcal{D}_{f\Omega}(\Pi) = \mathcal{D}_{f\Omega}(-\Pi).
\]

Let $f_0$ be a nonzero scalar function. The function `modular_vectorfield` computes the modular vector field of a Poisson bivector field with respect to the volume form $f_0\Omega$.

#### Algorithm 2.8 `modular_vf(bivector, function)`

**Input:** a Poisson bivector field and a nonzero scalar function $f_0$

**Output:** the modular vector field of the Poisson bivector field (2.12) relative to the volume form $f_0\Omega$.

1: procedure
2: \[ \text{bivector} \leftarrow \text{a dictionary \{ (1, 2): } \Pi^{12}, \ldots, (m-1, m): \Pi^{m-1,m} \text{\} that represents} \]
   \[ \text{a bivector field according to (3.1)} \]
3: \[ \text{function} \leftarrow \text{a string expression} \rightarrow \text{Represents a scalar function} \]
4: \[ \text{for each } 1 \leq i < j \leq m \text{ do} \rightarrow \text{A dictionary for } -\Pi \text{ in (2.12)} \]
5: \[ \text{bivector}[(i, j)] \leftarrow -\Pi^{ij} \]
6: end for
2.5. Unimodularity of Homogeneous Poisson bivector fields. We can verify whether an homogeneous Poisson bivector field is unimodular or not with the is_homogeneous_unimodular function. A Poisson bivector field \( \Pi \) on \( \mathbb{R}^m \),

\[
\Pi = \frac{1}{2} \Pi^{ij} \frac{\partial}{\partial x_i} \wedge \frac{\partial}{\partial x_j}, \quad i, j = 1, \ldots, m;
\]

is said to be homogeneous if each coefficient \( \Pi^{ij} \) is an homogeneous polynomial \([23]\). To implement this function we use the following fact: an homogeneous Poisson bivector field on \( \mathbb{R}^m \) is unimodular on (the whole of) \( \mathbb{R}^m \) if and only if its modular vector field \((2.11)\) relative to the Euclidean volume form is zero \([20]\).

**Algorithm 2.9** is_homogeneous_unimodular(bivector)

**Input:** a homogeneous Poisson bivector field on \( \mathbb{R}^m \)

**Output:** verify if the modular vector field respect to the Euclidean volume form on \( \mathbb{R}^m \) of the Poisson bivector field is zero or not

1: procedure
2: bivector \( \leftarrow \) a dictionary that represents a bivector field according to (3.1)
3: if modularvf(bivector, 1) == \{0: 0\} then \( \triangleright \) See Algorithm 2.8
4: return True
5: else
6: return False
7: end if
8: end procedure

2.6. Bracket on Differential 1-Forms. The function one_forms_bracket computes the Lie bracket of two differential 1-forms \( \alpha, \beta \in \Gamma T^*M \) induced by a Poisson bivector field \( \Pi \) on \( M \) \([10, 23]\) and defined by

\[
\{ \alpha, \beta \}_\Pi := i_{\Pi^*(\alpha)} d\beta - i_{\Pi^*(\beta)} d\alpha + d\langle \beta, \Pi^*(\alpha) \rangle.
\]

Here, \( d \) is the exterior derivative for differential forms and \( \{ df, dg \}_\Pi = d\{ f, g \}_\Pi \), by definition for all \( f, g \in C^\infty_M \). The bracket on the right-hand side of this equality is the Poisson bracket for smooth functions on \( M \) induced by \( \Pi \) \((2.3)\). In coordinates, if \( \alpha = \alpha_k dx^k \) and \( \beta = \beta_l dx^l \), for \( k, l = 1, \ldots, m \):

\[
\begin{align*}
\{ \alpha, \beta \}_\Pi &= \sum_{1 \leq i < j \leq m} \left[ (\Pi^i \alpha)^j \left( \frac{\partial \beta_j}{\partial x^i} - \frac{\partial \beta_i}{\partial x^j} \right) - (\Pi^j \beta)^i \left( \frac{\partial \alpha_i}{\partial x^j} - \frac{\partial \alpha_j}{\partial x^i} \right) \right] dx^i \\
&\quad + \sum_{1 \leq i < j \leq m} \left[ (\Pi^i \alpha)^j \frac{\partial \beta_j}{\partial x^i} - (\Pi^j \beta)^i \frac{\partial \alpha_i}{\partial x^j} \right] dx^j + \partial \left[ (\Pi^i \alpha) \beta_i \right] dx^k
\end{align*}
\]
Algorithm 2.10 one_forms_bracket(bivector, one_form_1, one_form_2)

Input: a Poisson bivector field and two differential 1-forms
Output: a differential 1-form which is the Lie bracket induced by the Poisson bivector field of the two differential 1-forms

1: procedure 2:
2: m ← dimension of the manifold  ▶ Given by an instance of PoissonGeometry
3: bivector ← a dictionary that represents a bivector field according to (3.1)
4: one_form_1 ← a dictionary {(1): α_1, ..., (m): α_m} that represents a differential 1-form according to (3.1)
5: one_form_2 ← a dictionary {(1): β_1, ..., (m): β_m} that represents a differential 1-form according to (3.1)
6: sharp_1 ← sharp_morphism(bivector, one_form_1)  ▶ See Algorithm 2.1
7: sharp_2 ← sharp_morphism(bivector, one_form_2)
8: Convert each of one_form_1 and one_form_2 to a symbolic expression
9: forms_bracket ← the dictionary {(1): 0, ..., (m): 0}
10: for 1 ≤ i < j ≤ m do  ▶ Compute the first two summations in (2.14)
11:     forms_bracket[(i)] ← forms_bracket[(i)] + sharp_1[(j)] * (∂β_j/∂x_i − ∂β_i/∂x_j)
12:     forms_bracket[(j)] ← forms_bracket[(j)] + sharp_1[(i)] * (∂β_i/∂x_j − ∂β_j/∂x_i)
13: end for
14: for k, l = 1 to m do  ▶ Compute the last sum in (2.14)
15:     forms_bracket[(k)] ← forms_bracket[(k)] + ∂(sharp_1[(l)] * β_l)/∂x_k
16: end for
17: if all values in forms_bracket are equal to zero then
18:     return {0: 0}  ▶ A dictionary with zero key and value
19: else
20:     return forms_bracket
21: end if
22: end procedure

2.7. Gauge Transformations. Let Π be a bivector field on M. Suppose we are given a differential 2-form λ on M such that the vector bundle morphism

(2.15) (id_{T^∗M} − λ♭ ∘ Π^♯) : T^∗M → TM is invertible.

Then, there exists a bivector field Π on M (well) defined by the skew-symmetric morphism

(2.16) Π^♭ = Π^♯ ∘ (id_{T^∗M} − λ♭ ∘ Π^♯)^{-1}.

Here, λ^♭ : TM → T^∗M is the vector bundle morphism given by X ↦ i_X λ. The bivector field Π is called the λ–gauge transformation of Π [29, 7, 8]. A pair of bivector fields Π and Π on M are said to be gauge equivalent if they are related by (2.16) for some differential 2-form λ on M satisfying (2.15). If Π is a Poisson bivector field, then Π is a Poisson bivector field if and only if λ is closed along the symplectic leaves of Π. A gauge transformation modifies only the leaf-wise symplectic form of Π by means of the pull-back of λ, preserving the characteristic foliation. Furthermore, gauge
transformations preserve unimodularity. The function \texttt{gauge\_transformation} computes the gauge transformation of a bivector field.

### Algorithm 2.11 \texttt{gauge\_transformation}(\texttt{bivector, two\_form})

**Input:** a bivector field and a differential 2-form  
**Output:** a bivector field which is the gauge transformation induced by the differential 2-form of the given bivector field

1: procedure  
2: \hspace{1em} m ← dimension of the manifold \hspace{1em} \triangleright \text{Given by an instance of PoissonGeometry}  
3: \hspace{1em} \texttt{bivector} ← a dictionary that represents a bivector field according to (3.1)  
4: \hspace{1em} \texttt{two\_form} ← a dictionary that represents a differential 2-form according to (3.1)  
5: \hspace{1em} \texttt{bivector\_matrix} ← \texttt{bivector\_to\_matrix}(\texttt{bivector}) \hspace{1em} \triangleright \text{See Algorithm 2.6}  
6: \hspace{1em} \texttt{two\_form\_matrix} ← \texttt{bivector\_to\_matrix}(\texttt{two\_form})  
7: \hspace{1em} \texttt{identity} ← the \texttt{m \times m} identity matrix  
8: \hspace{1em} \textbf{if} \hspace{1em} \texttt{det(\texttt{identity} - \texttt{two\_form\_matrix} \ast \texttt{bivector\_matrix})} == 0 \hspace{1em} \textbf{then}  
9: \hspace{2em} \textbf{return} False \hspace{1em} \triangleright \text{Means that (2.15) is not invertible}  
10: \hspace{1em} \textbf{else}  
11: \hspace{2em} \texttt{gauge\_matrix} ← \texttt{bivector} \ast (\texttt{identity} - \texttt{two\_form\_matrix} \ast \texttt{bivector\_matrix})  
12: \hspace{2em} \texttt{gauge\_bivector} ← an empty dictionary \texttt{dict()}  
13: \hspace{2em} \textbf{for} \hspace{1em} 1 \leq i < j \leq m \hspace{1em} \textbf{do}  
14: \hspace{3em} \texttt{gauge\_bivector}[(i, j)] ← \texttt{gauge\_matrix}[i - 1, j - 1]  
15: \hspace{2em} \textbf{end for}  
16: \hspace{2em} \textbf{return} \texttt{gauge\_bivector}, \texttt{det(\texttt{identity} - \texttt{two\_form\_matrix} \ast \texttt{bivector\_matrix})}  
17: \hspace{1em} \textbf{end if}  
18: \textbf{end procedure}

Observe that the function \texttt{gauge\_transformation} can be used to compute the gauge transformation induced by a closed differential 2-form of a Poisson bivector field.

### 2.8. Classification of Lie-Poisson bivector fields on \(\mathbb{R}^3\). A Lie-Poisson bivector field is a homogeneous Poisson bivector field (2.13) for which each \(\Pi_{ij}\) is a linear polynomial \([17, 15, 10]\). A pair of homogeneous Poisson bivector fields \(\Pi\) and \(\widetilde{\Pi}\) on \(\mathbb{R}^m\) are said to be equivalent (or isomorphic) if there exists an invertible linear operator \(T : \mathbb{R}^m \rightarrow \mathbb{R}^m\) such that

\[
\widetilde{\Pi} = T^*\Pi.
\]

Under this equivalence relation in the 3-dimensional case there exist 9 non-trivial equivalence classes of Lie-Poisson bivector fields \([24]\).

The function \texttt{linear\_normal\_form\_R3} computes a normal form of a Lie-Poisson bivector field on \(\mathbb{R}^3\). The normal forms are based on well-known classifications of (real) 3-dimensional Lie algebra isomorphisms \([24]\).

### Algorithm 2.12 \texttt{linear\_normal\_form\_R3}(\texttt{bivector})

**Input:** a Lie-Poisson bivector field on \(\mathbb{R}^3\)  
**Output:** a linear normal form for the Lie-Poisson bivector field

1: procedure
2. bivector ← a dictionary \{(1, 2): \Pi^{12}, (1, 3): \Pi^{13}, (2, 3): \Pi^{23}\} that represents a Lie-Poisson bivector field on \(\mathbb{R}^3\) according to (3.1)

3. Convert each value in bivector to symbolic expression

4. parameter ← \(x_1 \cdot \Pi^{23} - x_2 \cdot \Pi^{13} + x_3 \cdot \Pi^{12}\)

5. hessian_parameter ← Hessian matrix of parameter

6. if modular of (bivector) == 0 then \(\triangleright\) See Algorithm 2.8

7. if rank(hessian_parameter) == 0 then

8. return \{0: 0\} \(\triangleright\) A dictionary with zero key and value

9. else if rank(hessian_parameter) == 1 then

10. return \{(1, 2): 0, (1, 3): 0, (2, 3): x_1\}

11. else if rank(hessian_parameter) == 2 then

12. if index(hessian_parameter) == 2 then \(\triangleright\) Index of quadratic forms

13. return \{(1, 2): 0, (1, 3): -x_2, (2, 3): x_1\}

14. else

15. return \{(1, 2): 0, (1, 3): x_2, (2, 3): x_1\}

16. end if

17. else

18. if index(hessian_parameter) == 3 then \(\triangleright\) Index of quadratic forms

19. return \{(1, 2): x_3, (1, 3): -x_2, (2, 3): x_1\}

20. else

21. return \{(1, 2): -x_3, (1, 3): -x_2, (2, 3): x_1\}

22. end if

23. end if

24. else

25. if rank(hessian_parameter) == 0 then

26. return \{(1, 2): 0, (1, 3): x_1, (2, 3): x_2\}

27. else if rank(hessian_parameter) == 1 then

28. return \{(1, 2): 0, (1, 3): x_1, (2, 3): 4*a*x_1 + x_2\}

29. else

30. if index(hessian_parameter) == 2 then \(\triangleright\) Index of quadratic forms

31. return \{(1, 2): 0, (1, 3): x_1 - 4*a*x_2, (2, 3): 4*a*x_1 + x_2\}

32. else

33. return \{(1, 2): 0, (1, 3): x_1 + 4*a*x_2, (2, 3): 4*a*x_1 + x_2\}

34. end if

35. end if

36. end if

37. end procedure

2.9. Isomorphic Lie-Poisson Tensors on \(\mathbb{R}^3\). Using the function isomorphic_lie_poisson_R3 we can verify whether two Lie-Poisson bivector fields on \(\mathbb{R}^3\) are isomorphic (2.17), or not.

Algorithm 2.13 isomorphic_lie_poisson_R3(bivector_1, bivector_2)

Input: two Lie-Poisson bivector fields

Output: verify if the Lie-Poisson bivector fields are isomorphic or not

1. procedure

2. bivector_1 ← a dictionary that represents a bivector field according to (3.1)
2.10. Flaschka-Ratiu Bivector Fields. Given \( m-2 \) functions \( K_1, \ldots, K_{m-2} \in C^\infty_\mathcal{M} \) on an oriented \( m \)-dimensional manifold \((\mathcal{M}, \Omega)\), with volume form \( \Omega \), we can construct a Poisson bivector field \( \Pi \) on \( \mathcal{M} \) defined by

\[
\iota_\Pi \Omega := \mathrm{d}K_1 \wedge \cdots \wedge \mathrm{d}K_{m-2}.
\]

Clearly, \( \Pi \) is non-trivial on the open subset of \( \mathcal{M} \) where \( K_1, \ldots, K_{m-2} \) are (functionally) independent. Moreover, by construction, each \( K_i \) is a Casimir function of \( \Pi \). These class of Poisson bivector fields are called Flaschka-Ratiu bivector fields \([9]\). In coordinates, if \( \Omega = \mathrm{d}x^1 \wedge \cdots \wedge \mathrm{d}x^m \), then

\[
(2.18) \quad \Pi = (-1)^{i+j} \det P_{[i,j]} \cdot \frac{\partial}{\partial x^i} \wedge \frac{\partial}{\partial x^j}, \quad 1 \leq i < j \leq m.
\]

Here \( P \) denotes the \( (m-2) \times m \)-matrix whose \( k \)-th row is \((\partial K_k/\partial x^1, \ldots, \partial K_k/\partial x^m)\), for \( k = 1, \ldots, m-2 \); and \( P_{[i,j]} \) the matrix \( P \) without the columns \( i \) and \( j \). Moreover, the symplectic form \( \omega_S \) of \( \Pi \) on a 2-dimensional (symplectic) leaf \( S \subseteq \mathcal{M} \) is given by

\[
(2.19) \quad \omega_S = \left. \frac{\Pi}{\|\Pi\|^2} \left[ (-1)^{i+j+1} \det P_{[i,j]} \cdot \mathrm{d}x^i \wedge \mathrm{d}x^j \right] \right|_S, \quad \|\Pi\|^2 := \sum_{1 \leq i < j \leq m} \left( \det P_{[i,j]} \right)^2.
\]

The function \texttt{flaschka\_ratiu\_bivector} computes the Flaschka-Ratiu bivector field and the corresponding symplectic form of a ‘maximal’ set of scalar functions \([9, 14, 30, 11]\).

---

**Algorithm 2.14 flaschka\_ratiu\_bivector(casimir\_list)**

**Input:** \( m-2 \) scalar functions  
**Output:** the Flaschka-Ratiu bivector field induced by the \( m-2 \) functions and the symplectic form of this Poisson bivector field

1: procedure  
2: \( m \leftarrow \) dimension of the manifold \( \triangleright \) Given by an instance of \texttt{PoissonGeometry}  
3: \texttt{casimir\_list} \( \leftarrow \) a list \([K1, \ldots, 'K\{m-2\}]\) with \( m-2 \) string expressions \( \triangleright \) Each string expression represents a scalar function  
4: if at least two functions in \texttt{casimir\_list} are functionally dependent then  
5: \( \text{return} \) \{0: 0\} \( \triangleright \) A dictionary with zero key and value  
6: else  
7: \texttt{matrix\_gradients} \( \leftarrow \) a symbolic \((m-2) \times m\)-matrix  
8: for \( i = 1 \) to \( m-2 \) do  
9: \( \text{CONVERT} \) \( K_i \) to symbolic expressions  
10: \( \text{COMPUTE} \) the gradient vector \( \nabla K_i \) of \( K_i \)
2.11. Test Type Functions. In this section we describe our implementation of some useful functions in the PoissonGeometry module which allow us to verify whether a given geometric object on a Poisson manifold satisfies certain property. The algorithms for each of these functions are similar, as they are decision-making processes.

2.11.1. Jacobi Identity. We can verify in PoissonGeometry if a given bivector field Π is a Poisson bivector field or not.

Algorithm 2.15  is_poisson_tensor(bivector)

Input: a bivector field
Output: verify if the bivector field is a Poisson bivector field or not

1: procedure
2:  bivector ← a dictionary that represents a bivector field according to (3.1)
3:  if lichnerowicz_poisson_operator(bivector, bivector) == {0; 0} then
4:      return True
5:  else
6:      return False
7:  end if
8: end procedure

2.11.2. Kernel of a Bivector Field. The kernel of a bivector field Π is the subspace ker Π := \{α ∈ T^∗M | Π^∗(α) = 0\} of T^∗M. It is defined as the kernel of its sharp morphism (2.1), and is defined likewise for Poisson bivector fields [10, 23].
Algorithm 2.16  \textit{is\_in\_kernel}(\textit{bivector}, \textit{one\_form})

\textbf{Input:} a bivector field and a differential 1–form  
\textbf{Output:} verify if the differential 1–form belongs to the kernel of the (Poisson) bivector field

\begin{verbatim}
1: procedure
2: \textbf{bivector} ← a dictionary that represents a bivector field according to (3.1)
3: \textbf{one\_form} ← a dictionary that represents a differential 1-form according to (3.1)
4: if \textbf{sharp\_morphism}(\textbf{bivector}, \textbf{one\_form}) == \{0: 0\} then \Comment{See Algorithm 2.1}
5: \hspace{1em} return True
6: else
7: return False
8: end if
9: end procedure
\end{verbatim}

2.11.3. Casimir Functions. A function $K \in C^\infty_M$ is said to be a Casimir function of a Poisson bivector field $\Pi$ if its Hamiltonian vector field (2.4) is zero. Equivalently, if its exterior derivative $dK$ belongs to the kernel of $\Pi$ \cite{10, 9, 23}.

Algorithm 2.17  \textit{is\_casimir}(\textit{bivector}, \textit{function})

\textbf{Input:} a Poisson bivector field and a scalar function  
\textbf{Output:} verify if the scalar function is a Casimir function of the Poisson bivector field

\begin{verbatim}
1: procedure
2: \textbf{bivector} ← a dictionary that represents a bivector field according to (3.1)
3: \textbf{function} ← a string expression \Comment{Represent a scalar function}
4: if \textbf{hamiltonian\_vf}(\textbf{bivector}, \textbf{function}) == \{0: 0\} then \Comment{See Algorithm 2.3}
5: \hspace{1em} return True
6: else
7: return False
8: end if
9: end procedure
\end{verbatim}

2.11.4. Poisson Vector Fields. A vector field $W$ on $M$ is said to be a Poisson vector field of a Poisson bivector field $\Pi$ if it commutes with respect to the Schouten-Nijenhuis bracket, $[\left[ W, \Pi \right]] = 0$ \cite{10, 23}.

Algorithm 2.18  \textit{is\_poisson\_vf}(\textit{bivector}, \textit{vector\_field})

\textbf{Input:} a Poisson bivector field and a vector field  
\textbf{Output:} verify if the vector field is a Poisson vector field of the Poisson bivector field

\begin{verbatim}
1: procedure
2: \textbf{bivector} ← a dictionary that represents a bivector field according to (3.1)
3: \textbf{vector\_field} ← a dictionary that represents a vector field according to (3.1)
\end{verbatim}
2.11.5. Poisson Pairs. We can verify whether a couple of Poisson bivector fields \( \Pi \) and \( \Psi \) form a Poisson pair. That is, if the sum \( \Pi + \Psi \) is again a Poisson bivector field or, equivalently, if \( \Pi \) and \( \Psi \) commute with respect to the Schouten-Nijenhuis bracket, \([\Pi, \Psi] = 0\) \[10, 23]\.

**Algorithm 2.19** \( \text{is\_poisson\_pair}(\text{bivector}_1, \text{bivector}_2) \)

**Input:** two Poisson bivector fields.

**Output:** verify if the bivector fields commute with respect to the Schouten-Nijenhuis bracket

1. procedure
2. \( \text{bivector}_1 \leftarrow \) a dictionary that represents a bivector field according to (3.1)
3. \( \text{bivector}_2 \leftarrow \) a dictionary that represents a bivector field according to (3.1)
4. if lichnerowicz\_poisson\_operator(\text{bivector}_1, \text{bivector}_2) == \{0: 0\} then
5. \text{return True}
6. else
7. \text{return False}
8. end if
9. end procedure

3. PoissonGeometry: Syntax and Applications. PoissonGeometry is our python module for local calculus on Poisson manifolds. First we define a tuple of symbolic variables that emulate local coordinates on a finite (Poisson) smooth manifold \( M \). By default, these symbolic variables are just the juxtaposition of the symbol \( x \) and an index of the set \( \{1, \ldots, m = \dim M\}: (x_1, \ldots, x_m) \).

**Scalar Functions.** A local representation of a scalar function in PoissonGeometry is written using string literal expressions. For example, the function \( f = a(x^1)^2 + b(x^2)^2 + c(x^3)^2 \) should be written exactly as follows: \( 'a * x1**2 + b * x2**2 + c * x3**2' \). It is important to remember that all characters that are not local coordinates are treated as (symbolic) parameters: \( a, b \) and \( c \) for the previous example.

**Multivector Fields and Differential forms.** Both multivector fields and differential forms are written using dictionaries with tuples of integers as keys and string type values. If the coordinate expression of an \( a \)-multivector field \( A \) on \( M \), with \( a \in \mathbb{N} \), is given by,

\[
A = \sum_{1 \leq i_1 < i_2 < \cdots < i_a \leq m} A^{i_1 i_2 \cdots i_a} \frac{\partial}{\partial x^{i_1}} \wedge \frac{\partial}{\partial x^{i_2}} \wedge \cdots \wedge \frac{\partial}{\partial x^{i_a}}, \quad A^{i_1 \cdots i_a} = A^{i_1 \cdots i_a}(x),
\]
then \( A \) should be written using a dictionary, as follows:

\[
(3.1) \left\{ (1,...,a) : A^{1\cdots a}, \ldots, (i_1,...,i_a) : A^{i_1\cdots i_a}, \ldots, (m-a+1,...,m) : A^{m-a+1\cdots m} \right\}.
\]

Here each key \((i_1,...,i_a)\) is a tuple containing ordered indices \(1 \leq i_1 < \cdots < i_a \leq m\) and the corresponding value \(A^{i_1\cdots i_a}\) is the string expression of the scalar function (coefficient) \(A^{i_1\cdots i_a}\) of \(A\).

The syntax for differential forms is the same. It is important to remark that we can only write the keys and values of non-zero coefficients. See the documentation for more details.

### 3.1. Applications

We will now describe two applications. One of gauge transformation, used here to derive a characterization of gauge transformations on \(\mathbb{R}^3\) (see, Subsection 2.7), and a second one of jacobitator, used here to construct a family of Poisson bivector fields on \(\mathbb{R}^4\) (1.1).

**Gauge Transformations on \(\mathbb{R}^3\).** For an arbitrary bivector field on \(\mathbb{R}^3\),

\[
(3.2) \quad \Pi = \Pi^{12} \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^2} + \Pi^{13} \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^3} + \Pi^{23} \frac{\partial}{\partial x^2} \wedge \frac{\partial}{\partial x^3},
\]

and an arbitrary differential 2-form

\[
(3.3) \quad \lambda = \lambda_{12} \, dx^1 \wedge dx^2 + \lambda_{13} \, dx^1 \wedge dx^3 + \lambda_{23} \, dx^2 \wedge dx^3,
\]

we compute:

```python
>>> pg3 = PoissonGeometry(3)
>>> P = {(1,2): 'P12', (1,3): 'P13', (2,3): 'P23'}
>>> lambda = {(1,2): 'L12', (1,3): 'L13', (2,3): 'L23'}
>>> (gauge_bivector, determinant) = pg3.gauge_transformation(P, lambda)
>>> print(gauge_bivector)
>>> print(determinant)
```

\[
\{ (1,2): P12/(L12*P12 + L13*P13 + L23*P23 + 1),
(1,3): P13/(L12*P12 + L13*P13 + L23*P23 + 1),
(2,3): P23/(L12*P12 + L13*P13 + L23*P23 + 1) \}
\]

\[(L12*P12 + L13*P13 + L23*P23 + 1)**2\]

The symbols \(P12, P13, P23,\) and \(L12, L13, L23\) stand for the coefficients of \(\Pi\) and \(\lambda\), in that order. Then, (see (2.15)):

\[
(3.4) \quad \det(I - \lambda^i \circ \Pi^i) = (\lambda_{12} \Pi^{12} + \lambda_{13} \Pi^{13} + \lambda_{23} \Pi^{23} + 1)^2
\]

So, for \(1 \leq i < j \leq 3\), the \(\lambda\)-gauge transformation \(\Pi\) of \(\Pi\) is given by:

\[
(3.5) \quad \Pi = \frac{\Pi^{12}}{\lambda_{ij} \Pi^{ij} + 1} \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^2} + \frac{\Pi^{13}}{\lambda_{ij} \Pi^{ij} + 1} \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^3} + \frac{\Pi^{23}}{\lambda_{ij} \Pi^{ij} + 1} \frac{\partial}{\partial x^2} \wedge \frac{\partial}{\partial x^3}
\]

With these ingredients, we can now show:
Proposition 3.1. Let $\Pi$ be a bivector field on a 3–dimensional smooth manifold $M$. Then, given a differential 2–form $\lambda$ on $M$, the $\lambda$–gauge transformation $\overline{\Pi}$ (2.16) of $\Pi$ is well defined on the open subset,

$$\{ F := \langle \lambda, \Pi \rangle + 1 \neq 0 \} \subseteq M. \tag{3.6}$$

Moreover, $\overline{\Pi}$ is given by

$$\overline{\Pi} = \Pi \overleftarrow{\lambda}. \tag{3.7}$$

If $\Pi$ is Poisson and $\lambda$ is closed along the leaves of $\Pi$, then $\overline{\Pi}$ is also Poisson.

Proof. Suppose (3.2) and (3.3) are coordinate expressions of $\Pi$ and $\lambda$ on a chart $(U; x^1, x^2, x^3)$ of $M$. Observe that the pairing of $\Pi$ and $\lambda$ is given by

$$\langle \lambda, \Pi \rangle = \lambda^{12} \Pi_{12} + \lambda^{13} \Pi_{13} + \lambda^{23} \Pi_{23}. \tag{3.4}$$

Hence (3.4) yields $\det(\text{Id} - \lambda^\flat \circ \Pi^\sharp) = (\langle \lambda, \Pi \rangle + 1)^2$. This implies that the morphism $\text{Id} - \lambda^\flat \circ \Pi^\sharp$ is invertible on the open subset in (3.6) and, in consequence, the $\lambda$–gauge transformation of $\Pi$ (2.15). Finally, formula (3.7) follows from (3.5) as

$$\Pi = \frac{\Pi^{12}}{F} \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^2} + \frac{\Pi^{13}}{F} \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^3} + \frac{\Pi^{23}}{F} \frac{\partial}{\partial x^2} \wedge \frac{\partial}{\partial x^3}, \tag{3.7}$$

for $F$ in (3.6).

Parametrized Poisson Bivector Fields. Poisson bivector fields also play an important role in the theory of deformation quantization, which is linked to quantum mechanics [5]. They appear in star products, that is, in general deformations of the associative algebra of smooth functions of a symplectic manifold [13]. Our module PoissonGeometry can be used to study particular problems around deformations of Poisson bivector fields and star products.

For example, we can modify the following 4–parametric bivector field on $\mathbb{R}^4$

$$\Pi = a_1 x^2 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^2} + a_2 x^3 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^3} + a_3 x^4 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^4} + a_4 x^1 \frac{\partial}{\partial x^2} \wedge \frac{\partial}{\partial x^3},$$

using the jacobiator function to construct a family of Poisson bivector fields on $\mathbb{R}^4$:

```python
>>> pg4 = PoissonGeometry(4)
>>> P = {(1,2): 'a1*x2', (1,3): 'a2*x3', (1,4): 'a3*x4', (2,3): 'a4*x1'}
>>> pg4.jacobiator(P)
{(1,2,3): -2*a4*x1*(a1 + a2), (2,3,4): -2*a3*a4*x4}
```

Therefore

$$[[\Pi, \Pi]] = -2a_4(a_1 + a_2) x^1 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^2} \wedge \frac{\partial}{\partial x^3} - 2a_3 a_4 x^4 \frac{\partial}{\partial x^2} \wedge \frac{\partial}{\partial x^3} \wedge \frac{\partial}{\partial x^4}$$

Hence, we have two cases, explained in the following:
Lemma 3.2. If $a_4 = 0$, then $\Pi$ determines a 3-parametric family of Poisson bivector fields on $\mathbb{R}^4$.

\[
\Pi = a_1 x^2 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^2} + a_2 x^3 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^3} + a_3 x^4 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^4}.
\]

If $a_2 = -a_1$ and $a_3 = 0$, then $\Pi$ determines a 2-parametric family of Poisson bivector fields on $\mathbb{R}^4$.

\[
\Pi = a_1 x^2 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^2} - a_1 x^3 \frac{\partial}{\partial x^1} \wedge \frac{\partial}{\partial x^3} + a_4 x^1 \frac{\partial}{\partial x^2} \wedge \frac{\partial}{\partial x^3}.
\]

Related Work. Notable contributions in similar directions include: computations of normal forms in Hamiltonian dynamics (in Maxima) [31], symbolic tests of the Jacobi identity for generalized Poisson brackets and their relation to hydrodynamics [21], and an implementation of the Schouten-Bracket for multivector fields (in Sage).

Our work here is, to the best of our knowledge, the first comprehensive implementation of routine computations used in Poisson geometry, and in Python (based on SymPy[25]).

Future Directions. With the algorithms in this paper, numerical extensions for the same methods can be developed. Explicit computations of Poisson cohomology can also be explored. These are the subjects of ongoing, and forthcoming work.

Acknowledgments. We thank Luis A. Gómez-Telesforo for many fruitful discussions.

REFERENCES

[1] A. Abouqateb and M. Boucetta. The Modular Class of a Regular Poisson Manifold and the Reeb Class of its Symplectic Foliation, C. R. Math. Acad. Sci. Paris 337, 1, 6166 (2003)

[2] M. Ammar, G. Kass, M. Masmoudi, N. Poncin, Strongly R-Matrix Induced Tensors, Koszul Cohomology, and Arbitrary-Dimensional Quadratic Poisson Cohomology, Pac. J. Math. 245, 1-23 (2010)

[3] M. Avendaño-Camacho, J. A. Vallejo and Yu. Vorobiev, A Perturbation Theory Approach to the Stability of the Pais-Uhlenbeck Oscillator, J. Math. Phys. 58 (9), (2017)

[4] M. Avendaño-Camacho and Yu. Vorobiev, Deformations of Poisson Structures on Fibered Manifolds and Adiabatic Slow-Fast Systems, Int. J. Geom. Methods Mod. Phys., 14 (6), (2017)

[5] F. Bayen, M. Flato, C. Fronsdal, A. Lichnerowicz, and D. Sternheimer, Deformation theory and quantization I and II, Ann. Physics, 111 (1), 6151 (1978)

[6] E. Bayro-Corrochano, Geometric Algebra Applications Vol. I: Computer Vision, Graphics and Neurocomputing, Springer International Publishing, (2019)

[7] H. Bursztyn, On Gauge Transformations of Poisson Structures, Quantum Field Theory and Noncommutative Geometry, Lecture Notes in Phys., 662. Springer.

[8] H. Bursztyn, O. Radko, Gauge equivalence of Dirac structures and symplectic groupoids. Ann. Inst. Fourier (Grenoble) 53, 309-337 (2003)

[9] P. A. Damianou and F. Petalidou, Poisson Brackets with Prescribed Casimirs, Canad. J. Math. 64, 991-1018 (2012)

[10] J. P. Dufour and N. T. Zung, Poisson Structures and their Normal Forms, Progress in Mathematics, 242, Birkhäuser Verlag, Basel, (2005)

[11] M. Evangelista-Alvarado, P. Suárez-Serrato, J. Torres-Orozco , R. Vera, On Bott-Morse Folitations and their Poisson Structures in Dimension 3, Jour. Singular. 19, 19-33 (2019)

https://trac.sagemath.org/ticket/23429
[12] R. L. Fernandes, Lie Algebroids, Holonomy, and Characteristic Classes, Adv. in Math. 170, 119-179 (2002)
[13] M. Flato, A. Lichnerowicz and D. Sternheimer, Déformations 1-différentiables des algèbres de Lie attachées à une variété symplectique ou de contact, Comp. Math. 31, 47-82 (1975)
[14] L. C. García-Naranjo, P. Suárez-Serrato and R. Vera, Poisson Structures on Smooth 4-manifolds, Lett. Math. Phys. 105, 1533-1550 (2015)
[15] V. L. Ginzburg and A. Weinstein, Lie-Poisson Structure On Some Poisson Lie Groups, J. Amer. Math. Soc 5 no. 2, 445-453 (1992)
[16] V. Guillemin, E. Miranda and A. R. Pires, Codimension One Symplectic Foliations and Regular Poisson Structures, Bull. Braz. Math. Soc., 42 no. 4, 607-623 (2011)
[17] B. Konstant, Orbits, Symplectic Structures, and Representation Theory, Proc. U.S.-Japan Sem. Differ. Geom., Kyoto, Nippon Hyoronsha Tokyo 71, Japan, (1965)
[18] Y. Kosmann-Schwarzbach, Poisson Manifolds, Lie Algebroids, Modular Classes: a Survey, SIGMA 4, 005-034 (2008)
[19] J. L. Koszul, Crochet de Schouten-Nijenhuis et Cohomologie, Astérisque, S131, 257-271 (1985)
[20] V. V. Kozlov, Symmetries, Topology and Resonances in Hamiltonian Dynamics, Udmurtian State University, Izhevsk, (1996); English Transl., Springer-Verlag, Berlin, (1995)
[21] M. Kröger, M. Hütter, H. C. Öttinger, Symbolic Test of the Jacobi Identity for Given Generalized ‘Poisson’ Bracket, Comput. Phys. Commun. 137, Issue 2, 325-340 (2001)
[22] A. Lichnerowicz, Les variétés de Poisson et leurs algèbres de Lie associées. J. Differential Geom. 12, 253-300 (1977)
[23] C. Laurent-Gengoux, A. Pichereau and P. Vanhaecke, Poisson Structures, Grundlehren der mathematischen Wissenschaften, 347, Springer-Verlag Berlin Heidelberg, (2013)
[24] Z. J. Liu and P. Xu, On Quadrature Poisson Structures, Lett. Math. Phys. 26, 33-42 (1992)
[25] A. Meurer et al., SymPy: Symbolic Computing in Python, PeerJ Comput. Sci. 3:e103 https://doi.org/10.7717/peerj-cs.103, (2017)
[26] P. Michor, Topics in Differential Geometry, Graduate Studies in Mathematics, American Mathematical Society, Providence 93 (2008)
[27] N. Nakanishi, On the Structure of Infinitesimal Automorphisms of Linear Poisson Manifolds I, J. Math. Kyoto Univ. 31, 71-82 (1991)
[28] S.-D.Poisson, Sur la variation des constantes arbitraires dans les questions de mécanique, J. École Polytechnique 8 Cah. 15 (1809), 266-344.
[29] P. Severa and A. Weinstein, Poisson Geometry with a 3-Form Background, Progr. Theoret. Phys. Suppl. 144, 145-154 (2001)
[30] P. Suárez-Serrato and J. Torres-Orozco, Poisson Structures on Wrinkled Fibrations, Bol. Soc. Mat. Mex. 22, 263-280 (2016)
[31] S. Takato, J.A. Vallejo, Hamiltonian Dynamical Systems: Symbolical, Numerical and Graphical Study, Math. in Comp. Sci. 13, 281-295 (2019)
[32] A. Weinstein, The Local Structure of Poisson Manifolds, J. Diff. Geom. 18, 523-557 (1983)
[33] A. Weinstein, The Modular Automorphism Group of a Poisson Manifold, J. Geom. Phys. 23, 379-394 (1997)
[34] A. Weinstein, Poisson Geometry, Diff. Geom. Appl. 9, 213-238 (1998)