Algorithmic procedure to compute abelian subalgebras and ideals of maximal dimension of Leibniz algebras

Article *in* International Journal of Computer Mathematics · December 2014

DOI: 10.1080/00207160.2014.884216

CITATIONS
3

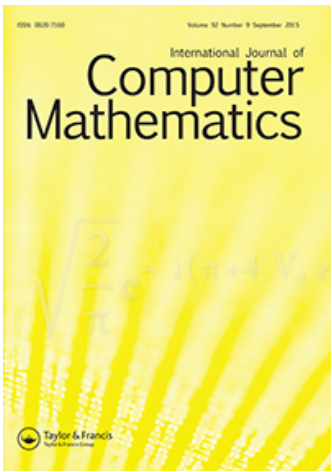
READS
29

3 authors:

Manuel Ceballos
Universidad Loyola Andalucía
50 PUBLICATIONS **325** CITATIONS
SEE PROFILE

Juan Núñez Valdés
Universidad de Sevilla
214 PUBLICATIONS **769** CITATIONS
SEE PROFILE

Angel F. Tenorio
Universidad Pablo de Olavide
105 PUBLICATIONS **405** CITATIONS
SEE PROFILE



International Journal of Computer Mathematics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gcom20

Algorithmic procedure to compute abelian subalgebras and ideals of maximal dimension of Leibniz algebras

Manuel Ceballos[a], Juan Núñez[a] & Ángel F. Tenorio[b]
[a] Departamento de Geometría y Topología, Facultad de Matemáticas. Universidad de Sevilla, Aptdo. 1160. 41080-Seville, Spain
[b] Dpto. de Economía, Métodos Cuantitativos e Historia Económica, Escuela Politécnica Superior, Universidad Pablo de Olavide. Ctra. Utrera km. 1. 41013 Seville, Sevilla, Spain
Accepted author version posted online: 05 Feb 2014.Published online: 27 Mar 2014.

To cite this article: Manuel Ceballos, Juan Núñez & Ángel F. Tenorio (2015) Algorithmic procedure to compute abelian subalgebras and ideals of maximal dimension of Leibniz algebras, International Journal of Computer Mathematics, 92:9, 1838-1854, DOI: 10.1080/00207160.2014.884216

To link to this article: http://dx.doi.org/10.1080/00207160.2014.884216

International Journal of Computer Mathematics, 2015
Vol. 92, No. 9, 1838–1854, http://dx.doi.org/10.1080/00207160.2014.884216

Algorithmic procedure to compute abelian subalgebras and ideals of maximal dimension of Leibniz algebras

Manuel Ceballos[a]*, Juan Núñez[a] and Ángel F. Tenorio[b]

[a]*Departamento de Geometría y Topología, Facultad de Matemáticas. Universidad de Sevilla, Aptdo. 1160. 41080-Seville, Spain;* [b]*Dpto. de Economía, Métodos Cuantitativos e Historia Económica, Escuela Politécnica Superior, Universidad Pablo de Olavide. Ctra. Utrera km. 1. 41013 Seville, Sevilla, Spain*

(*Received 5 August 2013; revised version received 6 November 2013; accepted 13 January 2014*)

In this paper, we show an algorithmic procedure to compute abelian subalgebras and ideals of a given finite-dimensional Leibniz algebra, starting from the non-zero brackets in its law. In order to implement this method, the symbolic computation package MAPLE 12 is used. Moreover, we also show a brief computational study considering both the computing time and the memory used in the two main routines of the implementation. Finally, we determine the maximal dimension of abelian subalgebras and ideals for 3-dimensional Leibniz algebras and 4-dimensional solvable ones over $\mathbb{C}$.

Keywords: Leibniz algebra; abelian subalgebra; abelian ideal; α invariant; β invariant; algorithm

2010 AMS Subject Classifications: 17A32; 17A60; 17–08; 68W30; 68Q25

1. Introduction

Nowadays, there exists an extensive research on Lie and Leibniz algebras due to both its own theoretical importance and its applications to many different fields, such as engineering, physics and applied mathematics. However, some general aspects of this theory remain unknown. For instance, the problem of classifying Lie algebras is still open. Due to Levi and Malcev's theorems (see [12,14], respectively), the study of finite-dimensional Lie algebras over a field of characteristic zero is reduced to the study of solvable and semisimple algebras and the classification of semisimple Lie algebras was obtained from works of Killing and Cartan (and see also [11]). However, solvable Lie algebras are only known in lower dimensions.

Leibniz algebras were introduced at the beginning of the 1990s by Loday [13]. They are a non-commutative generalization of Lie algebras and hence, inherit important properties of these algebras. There are two different types: left and right Leibniz algebras. In the case of right Leibniz algebras (which are those dealt with this paper), the right multiplication operator is a derivation (indeed, an inner derivation). On the contrary, when considering left Leibniz algebras, the right multiplication is not a derivation, but the left multiplication. Many results of the theory of Lie algebras can be extended to Leibniz algebras. In this sense, there exists an analogue of Levi's theorem for Leibniz algebras [3], which states that the classification of these algebras can be reduced to classify semisimple and solvable Leibniz algebras, where the semisimple part is a Lie

*Corresponding author. Email: mceballos@us.es

algebra. Moreover, Casas *et al.* [6] proved that the study of complex finite-dimensional solvable Leibniz algebras can be reduced to studying nilpotent ones. Current techniques do not allow researchers to face up successfully to the classification problem of solvable Leibniz algebras. Hence, to solve this and other problems, new and different properties of Leibniz algebras require to be studied naturally. Taking a simple related example, conditions on the lattice of subalgebras often lead to information about the algebra itself. In this way, studying abelian Leibniz subalgebras and ideals of a finite-dimensional Leibniz algebra constitutes the main goal of this paper, since this study would allow us to advance in the classification problem in the future.

In a previous paper [9], the authors developed an algorithmic method to compute abelian subalgebras and ideals of maximal dimension for Lie algebras. Now, in this paper, we generalize that method to Leibniz algebras. In that way, given a finite-dimensional Leibniz algebra $\mathcal{L}$, the maximal dimension of its abelian subalgebras and ideals is denoted by $\alpha(\mathcal{L})$ and $\beta(\mathcal{L})$, respectively. Note that both values are invariants of $\mathcal{L}$, being very important for many subjects; such as, for example, studying Leibniz algebra contractions, derivations and degenerations.

The paper is structured as follows: after reviewing some preliminaries in Section 2, Section 3 is devoted to give a step-by-step explanation of the algorithmic method focused on computing abelian subalgebras and ideals of Leibniz algebras, emphasizing the novelty in relation with the algorithm introduced in [9] and its implementation. Next, Section 4 shows an example of application for our algorithm, providing the computation of the abelian subalgebras and ideals of maximal dimension for 3-dimensional Leibniz algebras and 4-dimensional solvable ones, also including an example where the maximal dimension is different depending on considering subalgebras or ideals. Afterwards, a brief computational and statistical study of the algorithm is introduced in this paper. Finally, Section 6 is devoted to determine the complexity of the algorithm.

2. Preliminaries

We recall here some preliminary concepts on Leibniz algebras, bearing in mind that the reader can consult [13] for a general overview. Let us note that in this paper we only consider finite-dimensional Leibniz algebras over the complex number field $\mathbb{C}$.

A *Leibniz algebra* $\mathcal{L}$ over a field $\mathbb{K}$ is a vector space with a second inner bilinear composition law $[\cdot,\cdot]$, which verifies the so-called Leibniz identity

$$[[X,Y],Z] = [[X,Z],Y] + [X,[Y,Z]], \quad \forall X,Y,Z \in \mathcal{L}.$$

If, in addition, is verified that $[X,X] = 0$, for all $X \in \mathcal{L}$, then $\mathcal{L}$ is also a *Lie algebra*. In this case, it is satisfied that $[X,Y] = -[Y,X]$ and the Leibniz identity is equivalent to the Jacobi identity $([X,[Y,Z]] + [Y,[Z,X]] + [Z,[X,Y]] = 0)$.

Given a basis $\{e_i\}_{i=1}^n$ of a n-dimensional Leibniz algebra $\mathcal{L}$, its *structure constants* are defined by $[e_i,e_j] = \sum_{k=1}^n c_{i,j}^k e_k$, for $1 \leq i,j \leq n$.

The centre of a Leibniz algebra, $\mathcal{L}$, is given by

$$\mathrm{Cent}(\mathcal{L}) = \{X \in \mathcal{L} \mid [X,Y] = 0 = [Y,X],\ \forall Y \in \mathcal{L}\}.$$

The derived algebra of a Leibniz algebra, $\mathcal{L}$, is given by

$$\mathcal{D}(\mathcal{L}) = \langle [X,Y] \mid X,Y \in \mathcal{L}\}.$$

Given a Leibniz algebra $\mathcal{L}$, a vector subspace $\mathcal{M}$ of $\mathcal{L}$ is a subalgebra if $[\mathcal{M},\mathcal{M}] \subseteq \mathcal{M}$. Moreover, $\mathcal{M}$ is an abelian subalgebra if $[u,v] = 0$, for all $u,v \in \mathcal{M}$; that is, if $\mathrm{Cent}(\mathcal{M}) = \mathcal{M}$.

In addition, if the subalgebra $\mathcal{M}$ satisfies the conditions $[\mathcal{M}, \mathcal{L}] \subseteq \mathcal{M}$ and $[\mathcal{L}, \mathcal{M}] \subseteq \mathcal{M}$, then we say that $\mathcal{M}$ is a (two-sided) ideal of $\mathcal{L}$. If only one of these two conditions holds, then $\mathcal{M}$ is named left- and right-ideal, respectively.

In order to compute the basis of an abelian subalgebra of maximal dimension of $\mathcal{L}$, we apply the technique proposed in [2] by considering a basis $\mathcal{B}_n = \{e_i\}_{i=1}^n$ of the n-dimensional Leibniz algebra $\mathcal{L}$ and another basis $\mathcal{B} = \{v_h\}_{h=1}^k$ of an arbitrary k-dimensional (abelian) subalgebra $\mathcal{M}$ (with $k \leq n$). Since each vector $v_h \in \mathcal{B}$ is a linear combination of the vectors in $\mathcal{B}_n$, we can express it as $v_h = \sum_{i=1}^n a_{h,i} e_i$. In this way, the basis $\mathcal{B}$ is translated into the $k \times n$ matrix in which the h^{th} row saves the coordinates of v_h with respect to the basis $\mathcal{B}_n$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k,1} & a_{k,2} & \cdots & a_{k,n} \end{pmatrix}. \tag{1}$$

The rank of the matrix (1) is obviously equal to k. Consequently, after applying elementary row and column transformations, its associated echelon form is as follows:

$$\begin{pmatrix} b_{1,1} & 0 & \cdots & 0 & b_{1,k+1} & \cdots & b_{1,n} \\ 0 & b_{2,2} & \cdots & 0 & b_{2,k+1} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{k,k} & b_{k,k+1} & \cdots & b_{k,n} \end{pmatrix}. \tag{2}$$

So, without loss of generality, we can assume that any basis $\mathcal{B}$ of $\mathcal{M}$ can be expressed by the matrix (2). Therefore, each vector in $\mathcal{B}$ is a linear combination of two different types of vectors e_i: the ones coming from the pivot positions and the remaining ones. The first are called *main vectors* of $\mathcal{B}$ with respect to $\mathcal{B}_n$, with the rest being called *non-main vectors*.

Let us note that we can also assume that the coefficient of each main vector is equal to 1. Effectively, each basis vector in the subalgebra $\mathcal{M}$ is expressed as a linear combination of one main vector and the $n - k$ non-main ones, with their coefficients being saved as a row in the matrix (2). Hence, as the pivot coefficient corresponds to the main vector, the whole row can be divided by this coefficient. This fact will be used in the next section to implement the fourth step of our algorithm to find out abelian ideals and subalgebras.

3. Algorithm to compute abelian subalgebras and ideals

In this section, we give a generalization of the algorithmic procedure introduced by ourselves in [9]. In that reference, we gave the procedure to compute all the abelian subalgebras and ideals of a given Lie algebra starting from its law. Now, our goal is to adapt and adjust this procedure to make possible the computation of every abelian subalgebra and (two-sided) ideal for a given Leibniz algebra also described by its law. In addition, the algorithm also finds out the maximum among those subalgebras and ideals in order to determine the value of the invariants α and β. If the dimension of the algebra is small, its abelian subalgebras and ideals can be easily computed because the number of non-zero brackets is quite greater in proportion to the dimension. To solve this computational problem, we have implemented an algorithmic procedure which computes a basis of each non-trivial abelian subalgebra. To do so, we use the main and non-main vectors to express an arbitrary basis of the subalgebra and, hence, to determine the existence of non-zero brackets for each candidate subalgebra.

Before describing the algorithmic procedure for Leibniz algebras, we give previously an example to show how the main and non-main vector are going to be used in our procedure.

Example 1 Let us consider the 3-dimensional Leibniz algebra $\mathcal{L}$ with law $[e_1, e_1] = e_1 + e_2$, $[e_1, e_2] = -e_1 - e_2$. Obviously, $\mathcal{L}$ is non-abelian and both span(e_2) and span(e_3) are abelian subalgebras of dimension 1. Consequently, $1 \leq \alpha(\mathcal{L}) \leq 2$. Now, we check the existence of 2-dimensional abelian subalgebras in $\mathcal{L}$. The general expression of these subalgebras is the following by using main and non-main vectors:

$$\mathfrak{a}_s = \langle \{e_i + \lambda_{i,s} e_s \mid 1 \leq i \leq 3 \wedge i \neq s\} \rangle,$$

where e_s is the non-main vector. In this case, there exist three possible expressions for such subalgebras.

- For $s = 1$, the subalgebra should be span$(e_2 + \lambda_{2,1}e_1, e_3 + \lambda_{3,1}e_1)$ with brackets.

$$\begin{aligned}
[e_2 + \lambda_{2,1}e_1, e_2 + \lambda_{2,1}e_1] &= \lambda_{2,1}(\lambda_{2,1} - 1)(e_1 + e_2),\\
[e_2 + \lambda_{2,1}e_1, e_3 + \lambda_{3,1}e_1] &= \lambda_{2,1}\lambda_{3,1}(e_1 + e_2),\\
[e_3 + \lambda_{3,1}e_1, e_2 + \lambda_{2,1}e_1] &= \lambda_{3,1}(\lambda_{2,1} - 1)(e_1 + e_2),\\
[e_3 + \lambda_{3,1}e_1, e_3 + \lambda_{3,1}e_1] &= \lambda_{3,1}\lambda_{3,1}(e_1 + e_2).
\end{aligned}$$

  So, we only obtain the abelian subalgebras span$(e_2 + e_1, e_3)$ and span(e_2, e_3).
- For $s = 2$, the subalgebra should be span$(e_1 + \lambda_{1,2}e_2, e_3 + \lambda_{3,2}e_2)$, leading to the brackets.

$$[e_1 + \lambda_{1,2}e_2, e_1 + \lambda_{1,2}e_2] = (1 - \lambda_{2,1})(e_1 + e_2), \quad [e_1 + \lambda_{1,2}e_2, e_3 + \lambda_{3,2}e_2] = -\lambda_{3,2}(e_1 + e_2),$$
$$[e_3 + \lambda_{3,2}e_2, e_1 + \lambda_{1,2}e_2] = [e_3 + \lambda_{3,2}e_2, e_3 + \lambda_{3,2}e_2] = 0.$$

  Therefore, we only obtain the abelian subalgebra span$(e_1 + e_2, e_3)$.
- For $s = 3$, we should consider the subalgebra span$(e_1 + \lambda_{1,3}e_3, e_2 + \lambda_{2,3}e_3)$. In this case, we obtain no abelian subalgebras since $[e_1 + \lambda_{1,3}e_3, e_2 + \lambda_{2,3}e_3] = -e_1 - e_2 \neq 0$.

Therefore, we obtain that $\alpha(\mathcal{L}) = 2$ and the abelian subalgebras of maximal dimension are $\mathfrak{a} = \text{span}(e_1 + e_2, e_3)$ and $\mathfrak{b} = \text{span}(e_2, e_3)$. Now, we study which of those abelian subalgebras are (two-sided) ideals of $\mathcal{L}$. First, $\mathfrak{b}$ is not an ideal since $[e_1, e_2] = -e_1 - e_2 \notin \mathfrak{b}$. However, $\mathfrak{a}$ is an ideal because $\mathcal{D}(\mathcal{L}) = \langle e_1 + e_2 \rangle \subset \mathfrak{a}$.

Now, we show the general algorithmic procedure: let us consider a n-dimensional vector space $\mathcal{L}$ with basis $\mathcal{B}_n = \{e_i\}_{i=1}^n$ and with a second bilinear inner operation $[\cdot, \cdot]$, determined by certain products $[e_i, e_j] = \sum_{k=1}^n c_{ij}^k e_k$ $(1 \leq i, j \leq n)$. The algorithmic procedure consists of the following steps, starting with inserting the law of $\mathcal{L}$ and checking that the Leibniz identities hold.

Step 1 Introducing the bracket between two arbitrary basis vectors in $\mathcal{B}_n$.
Step 2 Computing the bracket between two vectors expressed as a linear combination of vectors from the basis $\mathcal{B}_n$.
Step 3 Checking the Leibniz identities for the vector space $\mathcal{L}$.
Step 4 For each k-dimensional subalgebra $\mathcal{M}$ of $\mathcal{L}$, computing the bracket between two arbitrary vectors in the basis $\mathcal{B}$. These vectors will be expressed as linear combinations of main and non-main vectors.
Step 5 Solving a system whose equations are obtained by imposing the abelian law to the brackets computed in the previous step for the subalgebra $\mathcal{M}$.
Step 6 Determining the existence of abelian subalgebras in a fixed dimension, starting from the solutions of the system solved in the previous step.
Step 7 Computing the value of the α invariant for the algebra $\mathcal{L}$ by ruling out dimensions for abelian subalgebras.

Step 8 Computing the basis of an abelian subalgebra for a fixed set of non-main vectors and restrictions given by the previous subroutines.
Step 9 Computing a list of all the abelian subalgebras of $\mathcal{L}$ with dimension k.
Step 10 Computing a list with the bases of all the non-trivial abelian subalgebras of $\mathcal{L}$, including those of dimension $\alpha(\mathcal{L})$.
Step 11 Determining the existence of an abelian (two-sided) ideal associated with a given abelian subalgebra.
Step 12 Computing $\beta(\mathcal{L})$, starting from the value of $\alpha(\mathcal{L})$ and the previous step.
Step 13 Computing a list with the bases of all the non-trivial abelian ideals of $\mathcal{L}$, including those of dimension $\beta(\mathcal{L})$.

The main structure of this algorithm can be almost automatically translated from the case of Lie algebras (introduced in [9]) into that of Leibniz algebras. However, several significant changes must be carried out for its application to Leibniz algebras. First of all, Leibniz algebras do not satisfy both commutative and skew-symmetric properties in general (i.e. $[X, X] = 0$ and $[X, Y] = -[Y, X]$ for all $X, Y \in \mathcal{L}$, respectively). Hence, all these brackets must be inserted in the law, taken into account in the intermediate computations and checked when imposing the abelian condition to the subalgebras of $\mathcal{L}$ (in Steps 1–5). Moreover, when checking if an abelian subalgebra is a (two-sided) ideal in Step 11, we must use the two-sided condition to assure that the subalgebra is a right- and lef-ideal of $\mathcal{L}$.

In a second level, we have obtained several improvements related to simplifications in the implementation, which are also valid for the case of Lie algebras in [9]. For example, we have used the command `elif` instead of `if` to reduce the complexity of the implementation. In this sense, we will only explain those steps in the implementation given in [9], which had to be generalized and adapted in order to make applicable the implementation to Leibniz algebras. In this sense, the description and implementation of Steps 7, 8, 10, 12 and 13 can be found exactly in [9] and there are no difference between considering Lie and Leibniz algebras for these steps.

We have implemented our algorithm by using the symbolic computation package MAPLE 12, although the implementation works properly in higher versions. In the first place, we load the libraries `linalg` and `ListTools` to activate commands like `Flatten` and others related to linear algebra. Moreover, we also have to load the libraries `combinat` and `Maplets[Elements]`. The first is used to apply commands of combinatorial algebra and the second to display a message reminding the user that it is required to introduce the input in the first subroutine. Let us note that all the routines explained hereafter must be written in the same worksheet in order to run it after introducing the data asked for by the dialog window built with the library `Maplets[Elements]`. Now, we explain the steps with differences in our algorithm.

Step 1 We have define the subroutine `law` to introduce the bracket between two arbitrary basis vectors in $\mathcal{B}_n$. It receives two natural numbers as inputs, which represent the subindexes of two basis vectors in $\mathcal{B}_n$. The subroutine returns the result of the bracket between these two vectors. A conditional sentence is introduced to determine each non-zero bracket. Let us note that the user has to complete the implementation of this subroutine depending on the law of $\mathcal{L}$, so we have added a sentence at the beginning of the implementation to remind this fact. Notice that before running any other sentence, we must restart all the variables and delete all the computations saved by using the command `restart`. Moreover, we must update the value of the dimension in the variable `dim` with the command `assign`.

```
> restart:
> maplet:=Maplet(AlertDialog("Remind to introduce the non-zero brackets
 and the dimension in subroutine law",'onapprove'=Shutdown("Continue"),
 'oncancel'=Shutdown("Aborted"))):
> Maplets[Display](maplet):
```

```
> assign(dim,...):
> law:=proc(i,j)
>   if (i,j)=... then ...;
>   elif ....
>   else 0; end if;
> end proc;
```

The ellipsis in command `assign` corresponds to writing the dimension of $\mathcal{L}$. The following two suspension points are associated with the computation of $[e_i, e_j]$: the value of the pair of subindexes (i, j) and the result of $[e_i, e_j]$ with respect to $\mathcal{B}_n$. The last ellipsis denotes the rest of non-zero brackets. For each non-zero bracket, a new sentence `elif` has to be included in the cluster.

Notice that the main difference between this implementation and the one considered in [9] is that here we cannot consider the skew-symmetry property.

Step 2 The computation of a bracket between two arbitrary vectors of $\mathcal{L}$ expressed as linear combinations of the basis $\mathcal{B}_n$ is carried out by the subroutine `bracket`. The subroutine `law` is called in the implementation.

```
> bracket:=proc(u,v,n)
>   local exp; exp:=0;
>   for i from 1 to n do
>      for j from 1 to n do
>         exp:=exp + coeff(u,e[i])*coeff(v,e[j])*law(i,j);
>      end do;
>   end do;
>   return exp;
> end proc:
```

Step 3 To check the Leibniz identities in the vector space $\mathcal{L}$, the subroutine `Leibniz` is implemented. Its unique input is the dimension n of the vector space $\mathcal{L}$ and its output is `true` if $\mathcal{L}$ is really a Leibniz algebra and `false` otherwise. This implementation uses the subroutine `bracket` and requires four local variables: first, the lists `L` and `M` are used respectively to save the list of the first n numbers repeated three times and the list of all the possible permutations of the previous list taken three by three; the variable `N` is defined in order to save all the expressions from the Leibniz identities; and, finally, the list `P` solves the resulting system of equations. The output of this subroutine is `false` in case that the system has no solutions and `true` otherwise.

```
> Leibniz:=proc(n)
>   local L,M,N,P;L:=[];M:=[];N:=[];P:=[];
>   for i from 1 to n do
>     L:=[op(L),i,i,i];
>   end do;
>   M:=permute(L,3);
>   for j from 1 to nops(M) do
>     eq[j]:=bracket(bracket(e[M[j][1]],e[M[j][2]],n),e[M[j][3]],n)-
>     bracket(bracket(e[M[j][1]],e[M[j][3]],n),e[M[j][2]],n)-
>     bracket(e[M[j][1]],bracket(e[M[j][2]],e[M[j][3]],n),n);
>   end do;
>   N:=[seq(eq[k], k=1..nops(M))];
>   for i from 1 to nops(N) do
>     if N[i]<>0 then P:=[op(P),N[i]];
>     end if;
>   end do;
>   if P=[] then return "True" else return "False";
>   end if;
> end proc:
```

Once we have checked that $\mathcal{L}$ is a Leibniz algebra, the algorithm can continue running with $\mathcal{L}$.

Step 4 For each k-dimensional subalgebra $\mathcal{M}$ of $\mathcal{L}$, the bracket between two arbitrary vectors in the basis of $\mathcal{M}$ is computed, being both vectors expressed as linear combinations of main and non-main vectors. This step is implemented by the subroutine `eq`. Each vector in the subalgebra $\mathcal{M}$ is expressed as a linear combination of one main vector (with coefficient equals 1) and the $n-k$ non-main ones. Obviously, all these expressions depend on the dimension of $\mathcal{M}$.

This subroutine is executed after introducing the law of $\mathcal{L}$, requiring four inputs: the dimension `n` of $\mathcal{L}$; the subindexes `i` and `l`, to fix the main vectors in the bracket to be computed; and a list `M` with the subindexes of all the non-main vectors in $\mathcal{M}$. To do so, we define seven local variables `eqt1`, `eqt2`, `eqt3`, `eqt4`, `L`, `u` and `v`. The subroutine `bracket` is called in the implementation of `eq`. First, `u` and `v` correspond to the vectors of $\mathcal{L}$ expressed using the main and non-main vectors. Then, we have to compute the brackets $[u,u]$, $[u,v]$, $[v,u]$ and $[v,v]$. In this way, the variables `eqt1`, `eqt2`, `eqt3` and `eqt4` save these four expressions. Finally, the list `L` contains all the coefficients of those expressions with respect to $\mathcal{B}_n$. Precisely, the list `L` is the first term of the output of the subroutine `eq`. The second is a list with the main-vector subindexes `i` and `l` used to generate `L`. Let us note that these two subindexes have to be saved together with the coefficients in order to use them in a later subroutine.

For the implementation, the coefficients of the non-main vectors are denoted by `b[i,k]`.

```
> eq:=proc(n,i,l,M::list)
>   local eqt1,eqt2,eq3,eqt4,L,u,v;
>   L:=[]; eqt1:=0; eqt2:=0; eqt3:=0; eqt4:=0; u:=e[i]; v:=e[l];
>   for k from 1 to nops(M) do
>     u:=u+b[i,M[k]]*e[M[k]]; v:=v+b[l,M[k]]*e[M[k]];
>   end do;
>   eqt1:=bracket(u,u,n); eqt2:=bracket(u,v,n);
>   eqt3:=bracket(v,u,n); eqt4:=bracket(v,v,n);
>   for m from 1 to n do
>    L:=[op(L),coeff(eqt1,e[m]),coeff(eqt2,e[m]),coeff(eqt3,e[m]),
     coeff(eqt4,e[m])];
>   end do;
>   return L,[i,l];
> end proc:
```

Let us note that the main difference between this implementation for Leibniz algebras and that given in [9] for Lie algebras is that we now have to consider the two-sided condition for abelian subalgebras. So, we need to impose the condition $[u,v]=[v,u]=0$.

Step 5 We have implemented the subroutine `sys` to solve the system of equations resulting from imposing the abelian condition to the brackets computed in the previous step for the subalgebra $\mathcal{M}$; i.e. `sys` solves the system generated by the output of the subroutine `eq`, requiring the following two inputs: the dimension `n` of $\mathcal{L}$ and a list `M` with the subindexes of the non-main vectors in the basis of $\mathcal{M}$. To do so, we have defined three local variables for the implementation: list `L` saves the subindexes of the main vectors; list `P` takes all the elements of `L` two by two; and, finally, list `R` saves all the equations of the system when imposing the abelian condition by using the subroutine `eq`. Let us note that the abelian condition must be checked on both sides for the Leibniz subalgebra as this was considered in the implementation of `eq`.

```
> sys:=proc(n,M::list)
>   local L,P,R;
>   L:=[];R:=[];
>   for x from 1 to n do
>     if member(x,convert(M,set))=false then L:=[op(L),x]; end if;
>   end do;
>   if nops(L)=1 then P:=[[L[1],L[1]]] else P:=choose(L,2); end if;
>   for j from 1 to nops(P) do
```

```
>   R:=[op(R),eq(n,P[j][1],P[j][2],M)[1]];
>   end do;
>   return {solve(Flatten(R))};
> end proc:
```

Notice that this implementation improves that given in [9] for this step for the case of Lie algebras. Moreover, we have used less local variables and the command `Flatten` in order to join all the lists.

Step 6 To determine the existence of abelian subalgebras with a certain dimension k, which starts from the solutions of the system solved in Step 5, we have implemented the subroutine `absub`. The input of this subroutine consists of two natural numbers, namely `n` and `k`: `n` is the dimension of $\mathcal{L}$; and `k` is less than `n`, representing the dimension of one of its subalgebras. The first case to study is when $k = 1$, for which we have evaluated if $[e_i, e_i] = 0$ by using the subroutine `law`. From this result, we have constructed the output of the subroutine. In the implementation, we have used two local variables `L` and `S`: `L` is a list consisting of lists with the subindexes of the `n`−`k` non-main vectors; whereas `S` is a set with the solutions given by the subroutine `sys`. In this way, `absub` returns either a message indicating the non-existence of `k`-dimensional abelian subalgebras or, if there exist k-dimensional abelian subalgebras, the set `S`. All this is necessary because the coefficient of each main vector is 1, which implies that the system given by the subroutine `sys` has no solutions when `S` vanishes. Conversely, when the system has a solution, the family of computed vectors forms a basis of the subalgebra, because it is linearly independent. Additionally, if every solution in `S` contains some complex coefficient, real solutions cannot be found for the system, which implies the non-existence of abelian subalgebras of dimension `k` for the field $\mathbb{K} = \mathbb{R}$. Consequently, if we want to execute this subroutine for the real field $\mathbb{R}$ instead of the complex one $\mathbb{C}$, we would need to include a conditional sentence for determining if such complex coefficients appear.

```
> absub:=proc(n,k)
>   local L,S; L:=choose(n,n-k); S:={ };
>   if k=1 then
>      for i from 1 to n do
>         if law(i,i)=0 then S:={op(S),e[i]};
>         end if;
>      end do;
>   return S;
>   end if;
>   for i from 1 to nops(L) do
>     if sys(n,L[i])={{}} then S:=S else
>       for j from 1 to nops(sys(n,L[i])) do
>         S:={op(S),{convert(L[i],set),sys(n,L[i])[j]}};
>       end do;
>     end if;
>   end do;
>   if S={} then return "There is no abelian subalgebra"; end if;
>   if S={{}} then return "There is no abelian subalgebra" else return S;
     end if;
> end proc;
```

Let us note that the case $k = 1$ was trivial in the implementation for Lie algebras in [9].

Step 9 To compute the list of abelian subalgebras of $\mathcal{L}$ with dimension $k \leq \alpha(\mathcal{L})$, we have programmed the subroutine `listabsub`. This subroutine requires two inputs: the value `n` of the dimension of $\mathcal{L}$ and a natural number k less than `n`, corresponding to the dimension of the abelian subalgebra. For the case $k = 1$, we have used the output obtained from the subroutine `absub`. The implementation requires two local variables `S` and `L`. This subroutine calls the subroutine `absub` to determine the existence of k-dimensional abelian subalgebras, saving the output of the latter in the variable `S`. Then, the subroutine `basabsub` is executed

to compute a basis for each k-dimensional abelian subalgebra. Precisely, the output of the subroutine `listabsub` is the list `L`, consisting of the basis of each abelian subalgebra of $\mathcal{L}$ with dimension k.

```
> listabsub:=proc(n,k)
>   local S,L; S:=absub(n,k);L:={};
>   if k=1 then {seq({absub(n,k)[i]},i=1..nops(absub(n,k)))};end if;
>   if S="There is no abelian subalgebra" then {}; end if;
>   for i from 1 to nops(S) do
>     L:={op(L),basabsub(n,S[i][1],S[i][2])};
>   end do;
>   return L;
> end proc:
```

Notice that, as in Step 6, the case $k = 1$ was trivial in the implementation for Lie algebras given in [9].

Step 11 To determine the existence of an abelian ideal associated with a given abelian subalgebra, we implement the subroutine `abideal`. For this subroutine, two inputs are required: a set `S` with the basis of an abelian subalgebra and the dimension `n` of $\mathcal{L}$. The subroutine determines if an abelian ideal can be associated with the basis `S` given by the subroutine `listabsub` for a fixed dimension. To do so, we impose that `S` has to be also a basis of an abelian ideal. Then, we solve the system of equations resulting from imposing all these conditions. Let us note that here we have to use the two-sided condition in order to obtain an abelian Leibniz ideal. If the system has no solutions, the output of `abideal` is the message 'There is no abelian ideal'; otherwise, the subroutine returns the basis of an abelian ideal. In order to implement this subroutine, we have used several local variables. First, `w` corresponds to an arbitrary vector in the subalgebra with basis `S`. The coefficient of each basis vector in `S` is denoted by `a[i]`. Next, we define the lists `R`, `L`, `Q` and `M`, which are used to save all the resulting expressions when we impose the ideal conditions. The list `L` saves all the non-zero brackets between the basis `S` and $\{e_i\}_{i=1}^n$. Let us note that we have to consider the brackets in both ways to assure that the subalgebra is a two-sided ideal. Next, in the list `R` we express arbitrary vectors from the basis `S` with coefficients `b[i,j]` in order to impose the ideal condition and the lists `Q` and `M` are used to solve the resulting system, since `M` saves the basis of the ideal associated with the subalgebra and `Q` saves the coefficients of each vector in `M`.

```
> abideal:=proc(S,n)
>   local w, R, L, Q, M; w:=0; R:=[]; L:=[]; Q:={}; M:={};
>   for i from 1 to nops(S) do
>     w:=w + a[i]*S[i];
>   end do;
>   for i from 1 to nops(S) do
>     for j from 1 to n do
>       if bracket(e[j],S[i],n)<>0 then
>         L:=[op(L),bracket(e[j],S[i],n)]; else L:=L; end if;
>       if bracket(S[i],e[j],n)<>0 then
>         L:=[op(L),bracket(S[i],e[j],n)]; else L:=L; end if;
>     end do;
>   end do;
>   for i from 1 to nops(L) do r[i]:=0;
>     for j from 1 to nops(S) do
>       r[i]:=r[i]+b[i,j]*S[j];
>     end do;
>   end do;
>   R:=[seq(r[i],i=1..nops(L))];
>   M:={seq(L[k]-R[k], k=1..nops(L))};
>   for i from 1 to nops(M) do
>     Q:={op(Q),seq(coeff(M[i],e[j])=0,j=1..n)};
>   end do;
>   if {solve(Q)}={} then return "There is no abelian ideal" else
```

```
>     return eval(S,solve(Q));
>   end if;
> end proc:
```

We conclude this section showing an example of application of our algorithm and routines.

Example 2 Next, we show an example with the 3-dimensional Leibniz algebra with law $[e_1, e_1] = e_1 + e_3, [e_1, e_2] = e_1 + e_3, [e_1, e_3] = -e_1 - e_3$. First, we complete the implementation of `law` as follows:

```
> restart:
> maplet:=Maplet(AlertDialog("Don't forget to introduce non-zero brackets of the
  algebra and its dimension in subroutine law",'onapprove'=Shutdown("Continue"),
 'oncancel'=Shutdown("Aborted"))):
> Maplets[Display](maplet):
> assign(dim,3):
> law:=proc(i,j)
>   if (i,j)=(1,1) then e[1]+e[3];
>   elif (i,j)=(1,2) then e[1]+e[3];
>   elif (i,j)=(1,3) then -e[1]-e[3];
>     else 0;
>   end if;
> end proc:
```

After that, we run all the subroutines. Now, we can compute α and β invariants as well as the set of abelian subalgebras and ideals of the Leibniz algebra $\mathcal{L}$ by using the commands defined by the above-mentioned subroutines.

```
> alpha(dim);
                                        2
> listabsub(dim,alpha(dim));
        {{e[2],e[3]},{e[1]-e[2],e[3]+e[2]},{e[1]+e[3],e[2]-e[1]},{e[1]+e[3],
          e[3]+e[2]}}
> allabsub(dim);
      {{{e[2]},{e[3]}},{{e[2],e[3]},{e[1]-e[2],e[3]+e[2]},{e[1]+e[3],
         e[2]-e[1]},{e[1]+e[3],e[3]+e[2]}}}
> beta(dim);
                                        2
> allabideal(dim);
        {{e[1]-e[2],e[3]+e[2]},{e[1]+e[3],e[2]-e[1]},{e[1]+e[3],e[3]+e[2]}}
```

Table 1. 3-dimensional non-Lie, Leibniz algebras.

$\mathcal{L}$	Brackets	$\alpha(\mathcal{L})$	$\beta(\mathcal{L})$
$\mathcal{L}_1^a$	$[e_2, e_2] = ae_1, [e_3, e_2] = e_1, [e_3, e_3] = e_1$	2	2
$\mathcal{L}_2$	$[e_3, e_3] = e_1$	2	2
$\mathcal{L}_3$	$[e_2, e_2] = e_1, [e_3, e_3] = e_1$	2	2
$\mathcal{L}_4$	$[e_1, e_3] = e_1$	2	2
$\mathcal{L}_5^a$	$[e_1, e_3] = ae_1 (a \neq 0), [e_2, e_3] = e_2, [e_3, e_2] = -e_2$	2	2
$\mathcal{L}_6$	$[e_2, e_3] = e_2, [e_3, e_2] = -e_2, [e_3, e_1] = e_1$	2	2
$\mathcal{L}_7$	$[e_1, e_3] = 2e_1, [e_2, e_2] = e_1, [e_2, e_3] = e_2,$ $[e_3, e_2] = -e_2, [e_3, e_3] = e_1$	1	1
$\mathcal{L}_8$	$[e_1, e_3] = ae_1 (a \neq 0), [e_2, e_3] = e_2$	2	2
$\mathcal{L}_9$	$[e_1, e_3] = e_1 + e_2, [e_2, e_3] = e_2$	2	2
$\mathcal{L}_{10}$	$[e_1, e_3] = e_2, [e_3, e_3] = e_1$	2	2
$\mathcal{L}_{11}$	$[e_1, e_3] = e_2, [e_2, e_3] = e_2, [e_3, e_3] = e_1$	2	2

We would like to point out that in this case, the abelian subalgebra span(e_2, e_3) is an abelian left-ideal, but not an abelian right-ideal. Hence, this subalgebra does not appear in the list of (two-sided) ideals.

4. Application of the algorithmic method

As example of application of the algorithm, we study and compute α and β invariants for low-dimensional Leibniz algebras. These invariants have not been previously studied for Leibniz algebras. Indeed, we determine the value of α and β invariants for 3-dimensional Leibniz algebras and 4-dimensional solvable ones over $\mathbb{C}$. To do so, we have used the classifications given in [1,4,5]. Let us note that we have not considered the case of Lie algebras in these classifications, since the value of α and β for them was already computed in [7, Proposition 4.1].

PROPOSITION 1 *Let $\mathcal{L}$ be a 3-dimensional non-Lie, Leibniz algebra. Then, the values of $\alpha(\mathcal{L})$ and $\beta(\mathcal{L})$ are given in Table 1.*

Table 2. 4-dimensional nilpotent, non-Lie and Leibniz algebra.

$\mathcal{L}$	Brackets	$\alpha(\mathcal{L})$	$\beta(\mathcal{L})$
$\mathcal{L}_1$	$[e_1,e_1]=e_2, [e_2,e_1]=e_3, [e_3,e_1]=e_4$	3	3
$\mathcal{L}_2$	$[e_1,e_1]=e_3, [e_1,e_2]=e_4, [e_2,e_1]=e_3, [e_3,e_1]=e_4$	3	3
$\mathcal{L}_3$	$[e_1,e_1]=e_3, [e_2,e_1]=e_3, [e_3,e_1]=e_4$	3	3
$\mathcal{L}_4^a$	$[e_1,e_1]=e_3, [e_1,e_2]=ae_4, [e_2,e_1]=e_3,$ $[e_2,e_2]=e_4, [e_3,e_1]=e_4 (a\in\{0,1\})$	2	2
$\mathcal{L}_5$	$[e_1,e_1]=e_3, [e_1,e_2]=e_4, [e_3,e_1]=e_4$	3	3
$\mathcal{L}_6$	$[e_1,e_1]=e_3, [e_2,e_2]=e_4, [e_3,e_1]=e_4$	2	2
$\mathcal{L}_7$	$[e_1,e_1]=e_4, [e_2,e_1]=e_3, [e_3,e_1]=e_4,$ $[e_1,e_2]=-e_3, [e_1,e_3]=-e_4$	3	3
$\mathcal{L}_8$	$[e_1,e_1]=e_4, [e_2,e_1]=e_3, [e_3,e_1]=e_4$ $[e_1,e_2]=-e_3+e_4, [e_1,e_3]=-e_4$	3	3
$\mathcal{L}_9$	$[e_1,e_1]=e_4, [e_2,e_1]=e_3, [e_2,e_2]=e_4$ $[e_3,e_1]=e_4, [e_1,e_2]=-e_3+2e_4, [e_1,e_3]=-e_4$	2	2
$\mathcal{L}_{10}$	$[e_1,e_1]=e_4, [e_2,e_1]=e_3, [e_2,e_2]=e_4$ $[e_3,e_1]=e_4, [e_1,e_2]=-e_3, [e_1,e_3]=-e_4$	2	2
$\mathcal{L}_{11}$	$[e_1,e_1]=e_4, [e_1,e_2]=e_3,$ $[e_2,e_1]=-e_3, [e_2,e_2]=-2e_3+e_4$	2	2
$\mathcal{L}_{12}$	$[e_1,e_2]=e_3, [e_2,e_1]=e_4, [e_2,e_2]=-e_3$	3	3
$\mathcal{L}_{13}^a$	$[e_1,e_1]=e_3, [e_1,e_2]=e_4$ $[e_2,e_1]=-ae_3, [e_2,e_2]=-e_4$	3 if $a=1$ 2 otherwise	3 if $a=1$ 2 otherwise
$\mathcal{L}_{14}^a$	$[e_1,e_1]=e_4, [e_1,e_2]=ae_4,$ $[e_2,e_1]=-ae_4, [e_2,e_2]=e_4, [e_3,e_3]=e_4$	2	2
$\mathcal{L}_{15}$	$[e_1,e_2]=e_4, [e_1,e_3]=e_4, [e_2,e_1]=-e_4$ $[e_2,e_2]=e_4, [e_3,e_1]=e_4$	2	2
$\mathcal{L}_{16}$	$[e_1,e_1]=e_4, [e_1,e_2]=e_4, [e_2,e_1]=-e_4, [e_3,e_3]=e_4$	2	2
$\mathcal{L}_{17}$	$[e_1,e_2]=e_3, [e_2,e_1]=e_4$	3	3
$\mathcal{L}_{18}$	$[e_1,e_2]=e_3, [e_2,e_1]=-e_3, [e_2,e_2]=e_4$	3	3
$\mathcal{L}_{19}$	$[e_2,e_1]=e_4, [e_2,e_2]=e_3$	3	3
$\mathcal{L}_{20}^a$	$[e_1,e_2]=e_4, [e_2,e_1]=\dfrac{1+a}{1-a}e_4, [e_2,e_2]=e_3$	3	3
$\mathcal{L}_{21}$	$[e_1,e_2]=e_4, [e_2,e_1]=-e_4, [e_3,e_3]=e_4$	2	2

Table 3. 4-dimensional solvable, non-nilpotent, non-Lie and Leibniz algebra (I).

$\mathcal{L}$	Brackets	$\alpha(\mathcal{L})$	$\beta(\mathcal{L})$
$\mathcal{L}_1$	$[e_1,e_3]=e_1,[e_2,e_4]=e_2$	2	2
$\mathcal{L}_2$	$[e_1,e_3]=e_1,[e_2,e_4]=e_2,[e_4,e_2]=-e_2$	2	2
$\mathcal{L}_3$	$[e_1,e_2]=e_3,[e_2,e_1]=-e_3,[e_1,e_4]=e_1,[e_2,e_4]=-e_2$ $[e_4,e_1]=-e_1,[e_4,e_2]=e_2,[e_4,e_4]=e_3$	2	2
$\mathcal{L}_4^{\gamma}$	$[e_2,e_1]=e_3,[e_1,e_4]=e_1,[e_2,e_4]=\gamma e_2,$ $[e_3,e_4]=(1+\gamma)e_3,[e_4,e_1]=-e_1$	2	2
$\mathcal{L}_5$	$[e_2,e_1]=e_3,[e_1,e_4]=e_1,[e_2,e_4]=-e_2$ $[e_4,e_1]=-e_1,[e_4,e_4]=e_3$	2	2
$\mathcal{L}_6$	$[e_2,e_1]=e_3,[e_1,e_4]=e_1+e_3,[e_3,e_4]=e_3$ $[e_4,e_1]=-e_1,[e_4,e_4]=-e_2$	2	2
$\mathcal{L}_7$	$[e_2,e_1]=e_3,[e_2,e_4]=e_2,[e_3,e_4]=e_3$	2	2
$\mathcal{L}_8$	$[e_2,e_1]=e_3,[e_1,e_2]=\beta e_3,[e_1,e_4]=e_1,[e_2,e_4]=\beta e_2$ $[e_3,e_4]=(\beta+1)e_3,[e_4,e_1]=-e_1,[e_4,e_2]=-\beta e_2$ $\beta=\dfrac{\sqrt{1-4a}-1}{\sqrt{1-4a}+1}$, where $a\neq\{0,1/4\}$	2	2
$\mathcal{L}_9^{\gamma}$	$[e_1,e_1]=e_3,[e_1,e_4]=e_1,[e_2,e_4]=\gamma e_2$ $[e_3,e_4]=2e_3,[e_4,e_1]=-e_1,[e_4,e_2]=-\gamma e_2$	2	2
$\mathcal{L}_{10}^{\delta}$	$[e_1,e_1]=e_3,[e_1,e_4]=e_1,[e_2,e_4]=\delta e_2$ $[e_3,e_4]=2e_3,[e_4,e_1]=-e_1\ (\delta\neq 0)$	2	2
$\mathcal{L}_{11}$	$[e_1,e_1]=e_3,[e_1,e_4]=e_1,[e_3,e_4]=2e_3,$ $[e_4,e_1]=-e_1,[e_4,e_4]=e_2$	2	2
$\mathcal{L}_{12}$	$[e_1,e_1]=e_3,[e_1,e_4]=e_1,[e_2,e_4]=2e_2+e_3$ $[e_3,e_4]=2e_3,[e_4,e_1]=-e_1$	2	2
$\mathcal{L}_{13}$	$[e_1,e_1]=e_3,[e_1,e_4]=e_1+e_2,[e_2,e_4]=e_2$ $[e_3,e_4]=2e_3,[e_4,e_1]=-e_1-e_2,[e_4,e_2]=-e_2$	2	2
$\mathcal{L}_{14}$	$[e_1,e_1]=e_3,[e_2,e_4]=e_2,[e_4,e_2]=-e_2$	2	2
$\mathcal{L}_{15}^{\lambda}$	$[e_1,e_1]=e_3,[e_2,e_4]=e_2,[e_4,e_1]=e_3$ $[e_4,e_2]=-e_2,[e_4,e_4]=\lambda e_3$	2	2
$\mathcal{L}_{16}$	$[e_1,e_1]=e_3,[e_2,e_4]=e_2,[e_4,e_2]=-e_2,[e_4,e_4]=-2e_3$	2	2
$\mathcal{L}_{17}$	$[e_1,e_1]=e_3,[e_2,e_4]=e_2$	2	2
$\mathcal{L}_{18}^{\mu}$	$[e_1,e_1]=e_3,[e_2,e_4]=e_2,[e_4,e_1]=e_3,[e_4,e_4]=\mu e_3$	2	2
$\mathcal{L}_{19}$	$[e_1,e_1]=e_3,[e_2,e_4]=e_2,[e_4,e_1]=e_3,[e_1,e_4]=e_3$	2	2
$\mathcal{L}_{20}^{\mu_2,\mu_3}$	$[e_1,e_4]=e_1,[e_2,e_4]=\mu_2e_2,[e_3,e_4]=\mu_3e_3$ $[e_4,e_1]=-e_1,[e_4,e_2]=-\mu_2e_2\ (\mu_3\neq 0)$	3	3
$\mathcal{L}_{21}^{\mu_2,\mu_3}$	$[e_1,e_4]=e_1,[e_2,e_4]=\mu_2e_2$ $[e_3,e_4]=\mu_3e_3,[e_4,e_1]=-e_1\ (\mu_2,\mu_3\neq 0)$	3	3

PROPOSITION 2 *Let $\mathcal{L}$ be a 4-dimensional nilpotent, non-Lie, Leibniz algebra. Then, the values of $\alpha(\mathcal{L})$ and $\beta(\mathcal{L})$ are given in Table 2.*

PROPOSITION 3 *Let $\mathcal{L}$ be a 4-dimensional solvable, non-nilpotent, non-Lie, Leibniz algebra. Then, the values of $\alpha(\mathcal{L})$ and $\beta(\mathcal{L})$ are given in Tables 3 and 4.*

Although the value of α and β invariants are the same in all these classifications over $\mathbb{C}$, it is possible to find examples of (both nilpotent and solvable) Leibniz algebras over $\mathbb{R}$ having different values for these invariants.

Table 4. 4-dimensional solvable, non-nilpotent, non-Lie and Leibniz algebra (II).

$\mathcal{L}$	Brackets	$\alpha(\mathcal{L})$	$\beta(\mathcal{L})$
$\mathcal{L}_{22}^{\mu_2,\mu_3}$	$[e_1,e_4]=e_1,[e_2,e_4]=\mu_2e_2,[e_3,e_4]=\mu_3e_3$	3	3
$\mathcal{L}_{23}^{\mu_2}$	$[e_1,e_4]=e_1,[e_2,e_4]=\mu_2e_2,[e_4,e_1]=-e_1$ $[e_4,e_2]=-\mu_2e_2,[e_4,e_4]=e_3$	3	3
$\mathcal{L}_{24}^{\mu_2}$	$[e_1,e_4]=e_1,[e_2,e_4]=\mu_2e_2,$ $[e_4,e_1]=-e_1,[e_4,e_4]=e_3$ $(\mu_2\neq 0)$	3	3
$\mathcal{L}_{25}^{\mu_2}$	$[e_1,e_4]=e_1,[e_2,e_4]=\mu_2e_2,[e_4,e_4]=e_3$	3	3
$\mathcal{L}_{26}$	$[e_1,e_4]=e_1,[e_4,e_1]=-e_1,[e_4,e_2]=e_3$	3	3
$\mathcal{L}_{27}$	$[e_1,e_4]=e_1,[e_4,e_2]=e_3$	3	3
$\mathcal{L}_{28}^{\mu_3}$	$[e_1,e_4]=e_1+e_2,[e_2,e_4]=e_2,[e_3,e_4]=\mu_3e_3$ $[e_4,e_1]=-e_1-e_2,[e_4,e_2]=-e_2$ $(\mu_3\neq 0)$	3	3
$\mathcal{L}_{29}$	$[e_1,e_4]=e_1+e_2,[e_2,e_4]=e_2,[e_4,e_1]=-e_1-e_2$ $[e_4,e_2]=-e_2,[e_4,e_4]=e_3$	3	3
$\mathcal{L}_{30}^{\mu_3}$	$[e_1,e_4]=e_1+e_2,[e_2,e_4]=e_2,[e_3,e_4]=\mu_3e_3$	3	3
$\mathcal{L}_{31}$	$[e_1,e_4]=e_1+e_2,[e_2,e_4]=e_2,[e_4,e_4]=e_3$	3	3
$\mathcal{L}_{32}^{\mu_3}$	$[e_1,e_4]=e_1+e_2,[e_2,e_4]=e_2,$ $[e_3,e_4]=\mu_3e_3,[e_4,e_3]=-\mu_3e_3$ $(\mu_3\neq 0)$	3	3
$\mathcal{L}_{33}^{\alpha}$	$[e_1,e_4]=e_2,[e_3,e_4]=e_3,$ $[e_4,e_1]=\alpha e_2,[e_4,e_3]=-e_3$ $(\alpha\neq -1)$	3	3
$\mathcal{L}_{34}$	$[e_1,e_4]=e_2,[e_3,e_4]=e_3,[e_4,e_1]=-e_2$ $[e_4,e_3]=-e_3,[e_4,e_4]=e_2$	3	3
$\mathcal{L}_{35}$	$[e_1,e_4]=e_2,[e_3,e_4]=e_3,[e_4,e_3]=-e_3,[e_4,e_4]=e_1$	3	3
$\mathcal{L}_{36}^{\alpha}$	$[e_1,e_4]=e_2,[e_3,e_4]=e_3,[e_4,e_1]=\alpha e_2$	3	3
$\mathcal{L}_{37}$	$[e_1,e_4]=e_2,[e_3,e_4]=e_3,[e_4,e_1]=-e_2,[e_4,e_4]=e_2$	3	3
$\mathcal{L}_{38}$	$[e_1,e_4]=e_2,[e_3,e_4]=e_3,[e_4,e_4]=e_1$	3	3
$\mathcal{L}_{39}$	$[e_1,e_4]=e_1+e_2,[e_2,e_4]=e_2+e_3,[e_3,e_4]=e_3$	3	3
$\mathcal{L}_{40}$	$[e_1,e_2]=e_3,[e_2,e_1]=e_3,[e_4,e_1]=-e_1,[e_4,e_2]=-e_2,$ $[e_1,e_4]=e_1,[e_2,e_4]=e_2,[e_3,e_4]=2e_3$	2	2
$\mathcal{L}_{41}$	$[e_1,e_1]=e_2,[e_2,e_1]=e_3,[e_4,e_1]=-e_1,$ $[e_1,e_4]=e_1,[e_2,e_4]=2e_2,[e_3,e_4]=3e_3$	2	2

Example 3 Let $\mathcal{L}$ be the 4-dimensional solvable non-nilpotent Leibniz algebra over $\mathbb{R}$ with law

$$[x_1,x_2]=x_2-x_3,\quad [x_1,x_4]=2x_4,\quad [x_1,x_3]=x_2+x_3,$$
$$[x_2,x_1]=x_3-x_2,\quad [x_2,x_3]=x_4,$$
$$[x_3,x_1]=-x_2-x_3,\quad [x_3,x_2]=-x_4,\quad [x_4,x_1]=-2x_4.$$

Over $\mathbb{R}$, we can prove that $\alpha(\mathcal{L})=2$, but $\beta(\mathcal{L})=1$. Obviously, $\operatorname{span}(x_3,x_4)$ is an abelian subalgebra of dimension 2. Assume that $\alpha(\mathcal{L})=3$. Then $\mathcal{L}$ is almost abelian, and hence 2-step solvable. This is impossible, since $\mathcal{L}$ is 3-step solvable. Hence $\alpha(\mathcal{L})=2$.

Assume that I is a 2-dimensional abelian ideal. It is easy to see that we can represent I as $\operatorname{span}(ax_2+bx_3,x_4)$. Obviously both x_2 and x_3 cannot belong to I. Hence $a\neq 0$ and $b\neq 0$. We have $ax_2+bx_3\in I$ and $[x_1,ax_2+bx_3]=(a+b)x_2-(a-b)x_3\in I$. This implies $a^2+b^2=0$. This is a contradiction over $\mathbb{R}$, so that $\beta(\mathcal{L})=1$ in this case. Over $\mathbb{C}$, we can take $a=1$ and $b=i$, obtaining the 2-dimensional abelian ideal $I=\operatorname{span}(x_2+ix_3,x_4)$.

Example 4 Given a field $\mathbb{K}$ of characteristic 2, the 9-dimensional nilpotent Leibniz algebra shown in [8, Example 4.1] provides an example in which invariants α and β take values 6 and 5, respectively.

Table 5. Computing time (CT) and used memory (UM) for `allabsub`.

Input	CT (s)	UM (MB)
$n = 4$	0.1	3.24
$n = 5$	0.37	5.31
$n = 6$	1.06	5.54
$n = 7$	3.84	5.74
$n = 8$	13.07	5.99
$n = 9$	44.75	6.87
$n = 10$	151.67	8.12
$n = 11$	506.51	9.62

Table 6. CT and UM for `allabideal`.

Input	CT (s)	UM (MB)
$n = 4$	0.67	6.12
$n = 5$	2.31	6.24
$n = 6$	10.5	6.43
$n = 7$	58.34	6.93
$n = 8$	358.26	7.87
$n = 9$	2146.97	8.96

5. Statistical and computational data

Now, we present a computational study of the algorithm introduced in the previous section and run in an Intel Core 2 Duo T 5600 with a 1.83 GHz processor and 2.00 GB of RAM. Tables 5 and 6 reproduce computational data about both the CT and the memory used to obtain the outputs of `allabsub` and `allabideal` with respect to the dimension n of the algebra. For this study, we have considered the Leibniz algebra $\mathfrak{f}_n$ generated by $\{e_i\}_{i=1}^n$ with non-zero brackets

$$[e_i, e_1] = e_{i+1}, \quad \forall 1 \leq i \leq n-2,$$

and the algebra $\mathfrak{f}_n^*$ consisting of adding the non-zero bracket $[e_1, e_1] = e_3$.

Both algebras, $\mathfrak{f}_n$ and $\mathfrak{f}_n^*$, correspond to filiform Leibniz algebras. More concretely, $\mathfrak{f}_n$ can be decomposed into the direct sum of a $(n-1)$-dimensional nulfiliform Leibniz algebra and a 1-dimensional abelian algebra. Note that the classification of complex filiform Leibniz algebras were already obtained in [10].

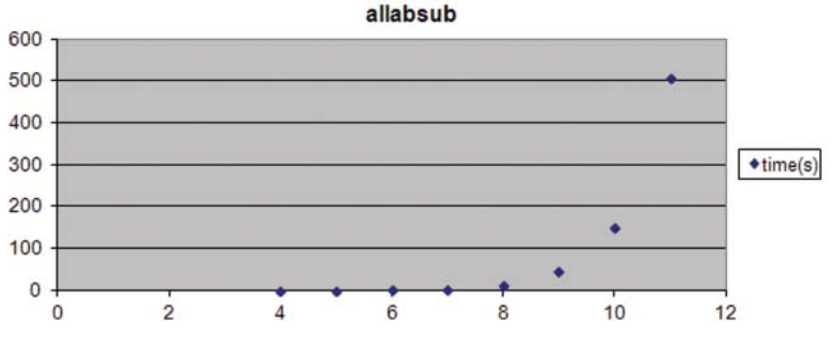


Figure 1. CT for allabsub.

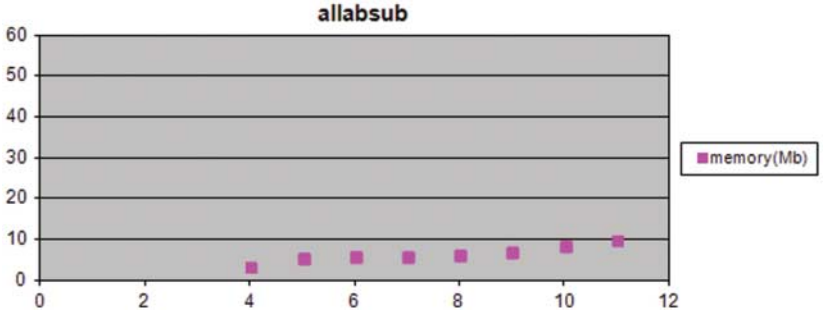


Figure 2. UM for allabsub.

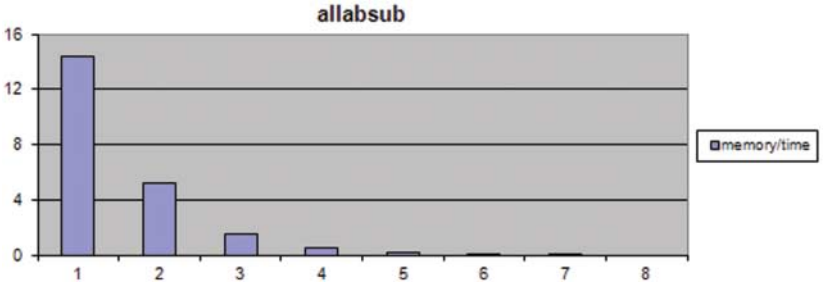


Figure 3. Quotient between UM and CT for allabsub.

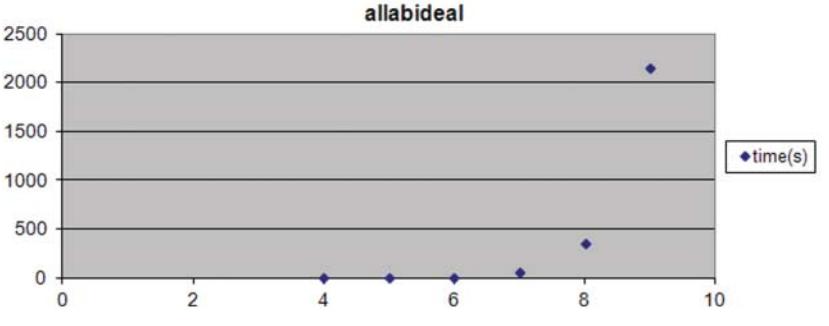


Figure 4. CT for allabideal.

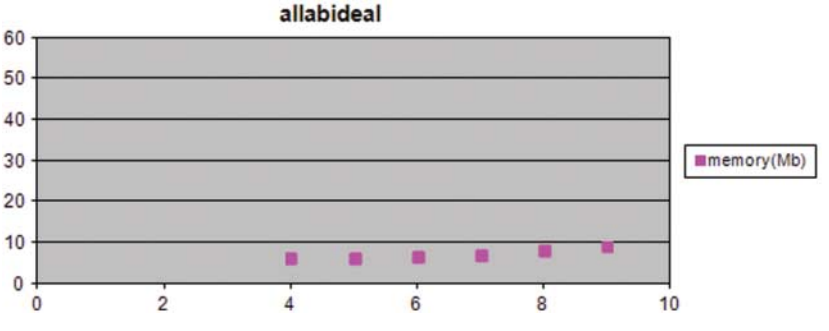


Figure 5. UM for allabideal.

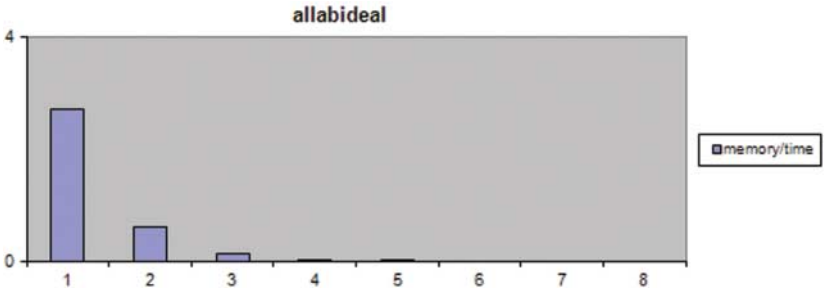


Figure 6. Quotient between UM and CT for allabideal.

Table 7. Complexity and number of operations.

Step	Routine	Complexity	Operations
1	`law`	$O(n^2)$	$N_1(n) = O(\frac{n(n-1)}{2})$
2	`bracket`	$O(n^4)$	$N_2(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} N_1(n)$
3	`Leibniz`	$O(n^7)$	$N_3(n) = O(n) + O(n^3) + \sum_{i=1}^{n^3} N_2(n) + \sum_{j=1}^{n^3} \sum_{k=1}^{n} 1$
4	`eq`	$O(n^4)$	$N_4(n) = \sum_{j=1}^{\frac{n(n-1)}{2}} N_1(n)$
5	`sys`	$O(n^6)$	$N_5(n) = \sum_{i=1}^{\frac{n(n-1)}{2}} N_4(n)$
6	`absub`	$O(n^{10})$	$N_6(n) = \sum_{i=1}^{\frac{n(n-1)}{2}} \sum_{j=1}^{n^2} (N_5(n))$
7	`alpha`	$O(n^{11})$	$N_7(n) = \sum_{i=1}^{n} N_6(n)$
8	`basabsub`	$O(n^2)$	$N_8(n) = O(n^2) + \sum_{i=1}^{n} O(n) + \sum_{i=1}^{n} \sum_{j=1}^{n} O(1)$
9	`listabsub`	$O(n^{10})$	$N_9(n) = N_6(n) + \sum_{i=1}^{n} N_8(n)$
10	`allabsub`	$O(n^{11})$	$N_10(n) = N_7(n) + \sum_{i=1}^{n} N_9(n)$
11	`abideal`	$O(n^6)$	$N_{11}(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} N_2(n)$
12	`beta`	$O(n^{12})$	$N_{12}(n) = N_7(n) + \sum_{k=0}^{n-2} \sum_{i=1}^{n} (N_9(n) + N_{11}(n))$
13	`allabideal`	$O(n^{12})$	$N_{13}(n) = N_{12}(n) + \sum_{i=1}^{n} \sum_{j=1}^{n} (N_9(n) + N_{11}(n))$

Table 5 has been obtained from computing the set of all non-trivial abelian subalgebras for the algebras $\mathfrak{f}_n$ and $\mathfrak{f}_n^*$ up to dimension $n = 11$ inclusive. Note that the CT is about four times greater when the dimension n is increased in one unit starting from $n = 7$.

Analogously, Table 6 shows the same variables when computing the set of all non-trivial abelian ideals for the algebras $\mathfrak{f}_n$ and $\mathfrak{f}_n^*$, but up to dimension $n = 9$ inclusive, due to computational issues. We want to remark that the CT is about five times greater when the dimension n is increased in one unit.

Next, we show brief statistics about the relation between the CT and the memory used by the implementation of the main routines `allabsub` and `allabideal` for the Leibniz algebras $\mathfrak{f}_n$ and $\mathfrak{f}_n^*$.

In this sense, Figures 1 and 4 show the behaviour of the CT for both routines with respect to the value n for the dimension of both $\mathfrak{f}_n$ and $\mathfrak{f}_n^*$. For its part, Figures 2 and 5 graphically represent the behaviour of the UM for both routines with respect to the value n for the dimension of both $\mathfrak{f}_n$ and $\mathfrak{f}_n^*$. Note that the CT increases more quickly than the UM in both cases. Additionally, whereas the increase of the CT fits a positive exponential model, the UM does not follow such a model.

Finally, we have studied the quotients between UM and CT, obtaining the frequency diagram shown in Figures 3 and 6. In both cases, the behaviour also fits an exponential model, but being negative this time.

6. Complexity of the algorithm

This section is devoted to compute the complexity of the algorithm, considering the number of operations carried out in the worst case. We have used the big O notation to express the complexity. To recall the big O notation, the reader can consult [15]: given two functions $f, g : \mathbb{R} \to \mathbb{R}$, it is said that $f(x) = O(g(x))$ if and only if there exist $M \in \mathbb{R}^+$ and $x_0 \in \mathbb{R}$ such that $|f(x)| < M \cdot g(x)$, for all $x > x_0$.

We denote by $N_i(n)$ the order of the operations when running Step i. This function depends on the dimension n of the Leibniz algebra. Table 7 shows the number of computations and the complexity of each step, as well as indicating the name of the routine corresponding to each step. In fact, we determine that the complexity of the algorithm has a polynomial order, where the two last routines are the most computationally expensive.

References

[1] S. Albeverio, B.A. Omirov, and I.S. Rakhimov, *Classification of 4-dimensional Nilpotent complex Leibniz algebras*, Extracta Math. 21(3) (2006), pp. 197–210.
[2] J.C. Benjumea, J. Núñez, and Á.F. Tenorio, *The maximal abelian dimension of linear algebras formed by strictly upper triangular matrices*, Theoret. Math. Phys. 152(3) (2007), pp. 1225–1233.
[3] D.W. Barnes, *On Levi's theorem for Leibniz algebras*, Bull. Aust. Math. 86(2) (2012), pp. 184–185.
[4] E.M. Cañete and A.K. Khudoyberdiyev, *The classification of* 4*-dimensional Leibniz algebras*, Linear Algebr. Appl. 439(1) (2013), pp. 273–288.
[5] J.M. Casas, M.A. Insua, M. Ladra, and S. Ladra, *An algorithm for the classification of* 3*-dimensional complex Leibniz algebras*, Linear Algebr. Appl. 436(9) (2012), pp. 3747–3756.
[6] J.M. Casas, M. Ladra, B.A. Omirov, and I.A. Karimjanov, *Classification of solvable Leibniz algebras with null-filiform nilradical*, Linear Multilinear Algebra 61(6) (2012), pp. 758–774.
[7] M. Ceballos, *Abelian subalgebras and ideals of maximal dimension in Lie algebras*, Ph.D. Thesis, University of Seville, 2012.
[8] M. Ceballos and D. Towers, *On abelian subalgebras and ideals of maximal dimension in supersolvable Lie algebras*, J. Pure Appl. Algebra 218(3) (2014), pp. 497–503.
[9] M. Ceballos, J. Núñez, and Á.F. Tenorio, *Algorithmic method to obtain abelian subalgebras and ideals in lie algebras*, Int. J. Comput. Math. 89(10) (2012), pp. 1388–1411.
[10] J.R. Gómez and B.A. Omirov, *On classification of complex filiform Leibniz algebras*, arXiv:math/0612735v2 [math.RA].
[11] N. Jacobson, *Lie Algebras*, Interscience Publishers, Wiley, New York, 1962.
[12] E.E. Levi, *Sulla struttura dei gruppi finiti e continui*, Atti. Accad. Sci. Torino 40 (1905), pp. 551–565.
[13] J.L. Loday, *Une version non commutative des algèbres de Lie: les algèbres de Leibniz*, Enseign. Math. (2), 39 (1993), pp. 269–293.
[14] A.I. Malcev, *Solvable Lie algebras*, Trans. Am. Math. Soc. Transl. 9 (1962), pp. 228–262.
[15] H.S. Wilf, *Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, 1986.