Computational Algorithm for Obtaining Abelian Subalgebras in Lie Algebras

Manuel Ceballos,Juan Núñez,and Ángel F. Tenorio

Abstract—The set of all abelian subalgebras is computationally obtained for any given finite-dimensional Lie algebra, starting from the nonzero brackets in its law. More concretely, an algorithm is described and implemented to compute a basis for each non-trivial abelian subalgebra with the help of the symbolic computation package MAPLE. Finally, it is also shown a brief computational study for this implementation, considering both the computing time and the used memory.

Keywords—Solvable Lie algebra, maximal abelian dimension, algorithm.

I. Introduction

THERE exist an extensive research on Lie Theory due to its applications in Engineering, Physics and, above all, Applied Mathematics, in addition to its theoretical study. However, some aspects and properties of Lie algebras remain unknown. In fact, the classification of nilpotent and solvable Lie algebras is still an open problem, although the classification of other types of Lie algebras (such as semisimple and simple ones) was already obtained in 1890. In this way, the study of other structural properties of Lie algebras is very interesting for overcoming the difficulties involved in the unsolved problem of the classification of solvable and nilpotent Lie algebras.

This paper is devoted to the study of the abelian subalgebras of any given finite-dimensional Lie algebra $\mathfrak{g}$. In this way, a basis is obtained for all the non-trivial abelian subalgebras of $\mathfrak{g}$; that is, all the abelian subalgebras with dimension greater than 1 and less than or equal to the maximal abelian dimension of $\mathfrak{g}$ (the maximum among the dimensions of its abelian subalgebras). Note that the maximal abelian dimension can be usefully applied, for example, to characterize Lie algebras in several senses. So Tenorio [7] gave some criteria about properties of Lie algebras starting from this notion. Moreover, this topic has already been studied by different authors, being classical and fundamental the following references: Krawtchouk [4] ; Laffey [5], which computed the maximal abelian dimension of the algebra of $n \times n$ matrices over any field; or Suprunenko and Tyshkevich [6], which dealt with the problem of determining maximal abelian subalgebras of nilpotent type. However, in some cases like in [8] abelian ideals were considered instead of abelian subalgebras.

M. Ceballos and J. Núñez are with the Department of Geometry and Topology, Faculty of Mathematics, University of Seville, Aptdo. 1160, 41080-Seville, Spain, e-mail: {mceballos,jnvaldes}@us.es.

Á.F. Tenorio is with the Department of Economics, Quantitative Methods and Economic History, Polytechnical School, Pablo de Olavide University, Ctra. Utrera Km. 1, 41013-Seville, Spain, e-mail: aftenorio@upo.es.

Manuscript received April 19, 2005; revised January 11, 2007.

Previously, the authors have already studied abelian subalgebras by considering both points of view: Theoretical and practical. Moreover, the maximal abelian dimension was computed for two different families of complex Lie algebras: $\mathfrak{g}_n$, of $n \times n$ strictly upper-triangular matrices (see [1], [2]); and $\mathfrak{h}_n$, of $n \times n$ upper-triangular matrices (see [3]). To do it, an algorithmic procedure was introduced in [2].

At this time, the main goal of the article is to show an analogous algorithmic procedure which works for any arbitrary finite-dimensional complex Lie algebra (not necessarily of the types $\mathfrak{g}_n$ or $\mathfrak{h}_n$). This new algorithm, which generalizes the previous one, is formally given by indicating and commenting each of its steps. Besides, some computational data of its implementation with MAPLE are also shown and studied.

II. Theoretical background

This section is devoted to recall some concepts and results on Lie algebras to be applied later. For a general overview on such subjects, the interested reader can consult [9]. Note that, from here on, only finite-dimensional Lie algebras over the field $\mathbb{F}$, where $\mathbb{F}$ can be $\mathbb{R}$ or $\mathbb{C}$, are considered.

Given a finite-dimensional Lie algebra $\mathfrak{g}$, a vector subspace $\mathfrak{h}$ of $\mathfrak{g}$ is an abelian subalgebra if the following conditions hold:

$$[\mathfrak{h},\mathfrak{h}] \subseteq \mathfrak{h}; \quad \text{and} \quad [u,v]=0,\ \forall\, u,v \in \mathfrak{h}.$$

The *maximal abelian dimension* of a given and fixed d-dimensional Lie algebra $\mathfrak{g}$, which is denoted by $\mathcal{M}(\mathfrak{g})$, is the maximum among the dimensions of its abelian subalgebras. A *maximal abelian subalgebra* is an abelian subalgebra whose dimension is $\mathcal{M}(\mathfrak{g})$.

To compute the basis of a maximal abelian subalgebra of $\mathfrak{g}$, a basis $\mathcal{B}_d = \{X_i\}_{i=1}^d$ of $\mathfrak{g}$ is considered together with another basis $\mathcal{B} = \{v_h\}_{h=1}^r$ of an arbitrary r-dimensional (abelian) subalgebra $\mathfrak{h}$ (with $r \le d$). As each vector $v_h \in \mathcal{B}$ is a linear combination of the vectors in $\mathcal{B}_d$, $v_h = \sum_{i=1}^d a_{h,i} X_i$, the basis $\mathcal{B}$ can be translated to a matrix in which the h^{th} row records these coordinates of v_h with respect to the basis $\mathcal{B}_d$:

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

The rank of the matrix (1) is obviously equal to r and, hence, its echelon form is the following by using elementary row and column transformations:

$$\begin{pmatrix} b_{1,1} & 0 & \cdots & 0 & b_{1,r+1} & \cdots & b_{1,d} \\ 0 & b_{2,2} & \cdots & 0 & b_{2,r+1} & \cdots & b_{2,d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{r,r} & b_{r,r+1} & \cdots & b_{r,d} \end{pmatrix}. \qquad (2)$$

So, without loss of generality, any given basis $\mathcal{B}$ of $\mathfrak{h}$ can be expressed by (2). Hence, each vector in $\mathcal{B}$ is a linear combination of two different types of vectors X_i: The ones coming from the pivot positions and the remaining ones. The first are called *main vectors* of $\mathcal{B}$ with respect to $\mathcal{B}_d$, being called *non-main vectors* the rest.

III. Algorithm computing abelian subalgebras

An n-dimensional Lie algebra $\mathfrak{g}$ is considered together with the basis $\mathcal{B}_n = \{Z_1, \ldots, Z_n\}$. If n is lower, the abelian subalgebras of $\mathfrak{g}$ can be easily computed because the number of nonzero brackets with respect to $\mathcal{B}_n$ is quite greater in proportion with the dimension of $\mathfrak{g}$. To solve this computational problem, an algorithmic method has been implemented to compute a basis of each non-trivial abelian subalgebra of $\mathfrak{g}$. In this algorithm, the main and non-main vectors will be used to express any given basis of the subalgebra in order to determine the existence of nonzero brackets. The vectors in this basis will be expressed as a linear combination of the vectors in $\mathcal{B}_n$.

To implement the algorithm, the symbolic computation package MAPLE has been used. First the libraries `linalg` and `ListTools` are loaded to activate commands like `Flatten` and others related to Linear Algebra, since Lie algebras are vector spaces endowed with a second inner structure: The Lie bracket. Besides, the library `combinat` has to be also loaded to apply commands related to Combinatorial Algebra.

Now, an explanation is given for the different steps constituting the algorithm and its implementation. The structure of the algorithm is based on a main routine calling several other subroutines with different functions.

1) Implementing a subroutine which computes the Lie bracket between two arbitrary vectors in $\mathcal{B}_n$. This subroutine depends on the law of $\mathfrak{g}$.
   The subroutine, named `bracket`, receives two natural numbers as inputs. These numbers represent the subindexes of two vectors in $\mathcal{B}_n$. The subroutine returns the result of the bracket between these two vectors. Besides, conditional sentences are included to determine nonzero brackets (which are introduced in the subroutine) and the skew-symmetry property.

```
> bracket:=proc(i,j)
> if i=j then return 0;end if;
> if i>j then return -bracket(j,i);end if;
> if (i,j)=... then return ...;end if;
> ....
> else return 0;
> end if;
> end proc;
```

   The first two suspension points are associated with the computation of $[Z_i, Z_j]$: First, the value of the subindexes (i,j) and second, the result of $[Z_i, Z_j]$ with respect to $\mathcal{B}_n$. The third ellipsis denotes the rest of nonzero brackets. For each nonzero bracket, a new sentence `if` has to be included in the cluster.

2) For each k-dimensional subalgebra $\mathfrak{h}$ of $\mathfrak{g}$, computing the bracket between two arbitrary vectors in the basis of $\mathfrak{h}$. Those vectors are linear combinations of a main vector (with nonzero coefficient) and the $n-k$ non-main ones. These expressions depend on the dimension of $\mathfrak{h}$. After the law of $\mathfrak{g}$ is introduced, the brackets in an arbitrary subalgebra $\mathfrak{h}$ are computed by the subroutine `ex`, which requires four inputs: The dimension `n` of $\mathfrak{g}$; the subindexes `i` and `l`, indicating the main vectors in the bracket to be computed; and a list `M` with the subindexes of the non-main vectors in $\mathfrak{h}$. To do it, three local variables `exp`, `L` and `P` are defined. For computing the brackets between the vectors in $\mathcal{B}_n$, the subroutine `ex` calls the subroutine `bracket`, which is necessary to obtain each bracket in the law of $\mathfrak{h}$. Whereas the variable `exp` saves the expression of the bracket belonging to the law of $\mathfrak{h}$, the list `P` takes the elements of `M` two by two and finally, `L` is a list containing all the coefficients in the expression of `exp` with respect to $\mathcal{B}_n$. Precisely, the list `L` is the first term of the output of the subroutine `ex`. The second is a list with the subindexes `i` and `l` corresponding to `L`. Note that the subindexes of the main vectors has to be saved together with the coefficients in order to use them in another subroutine.
   Each vector in the subalgebra $\mathfrak{h}$ can be expressed as a linear combination of one main vector and the $n-k$ non-main ones according to expression (2), where each row represents the coefficients of one vector in the basis of $\mathfrak{h}$. Obviously, the coefficient of each main vector can be assumed to be equal to 1, because the row of (2) corresponding to that main vector can be divided by its coefficient. To implement the subroutine `ex`, the coefficients of the non-main vectors are denoted by `a[i,k]`.

```
> ex:=proc(n,i,l,M::list)
> local exp,L,P;
> L:=[];
> if nops(M)=1 then P:=[[M[1],M[1]]] else
  P:=choose (M,2);
> end if;
> exp:=bracket(i,l);
> for k from 1 to nops(M) do
> exp:=exp+a[l,M[k]]*bracket(i,M[k])
  +a[i,M[k]]*bracket(M[k],l);
> end do;
> for j from 1 to nops(P) do
> exp:=exp+(a[i,P[j][1]]*a[l,P[j][2]]
  -a[i,P[j][2]]*a[l,P[j][1]])
  *bracket(P[j][1],P[j][2]);
> end do;
> for m from 1 to n do
> L:=[op(L),coeff(exp,Z[m])];
> end do;
> return L,[i,l];
> end proc;
```

3) Solving a system whose equations are obtained by imposing the abelian law to the brackects computed in

the previous step for the subalgebra $\mathfrak{h}$.
This subroutine, named `sys`, receives two inputs: The dimension `n` of $\mathfrak{g}$ and a list `M` with the subindexes of the non-main vectors in the basis of $\mathfrak{h}$. This subroutine solves the system of equations generated by the subroutine `ex`. Four local variables L, P, R and S have been defined for its implementation. L is a list with the subindexes of the main vectors. The list R contains the expressions computed by the subroutine `ex`. P is defined as in the previous subroutine. Finally, S is a set where the equations of the system are saved:

```
> sys:=proc(n,M::list)
> local L,P,R,S;
> L:=[]; R:=[]; S:={};
> 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[j]:=[ex(n,P[j][1],P[j][2],M)];
> end do; R:=[seq(r[i][1],i=1..nops(P))];
> for y from 1 to nops(R) do
> for k from 1 to n do
> S:={op(S),R[y][k]=0};
> end do; end do;
> return {solve(S)};
> end proc;
```

4) Programming a subroutine which determines the existence of abelian subalgebras in the dimension considered in the previous two steps.
This subroutine, called `absub`, is implemented by introducing two natural numbers `n` and `k`, namely: `n` is the dimension of $\mathfrak{g}$ and `k` is less than `n`. This subroutine determines the existence of abelian subalgebras with dimension `k`. Two local variables are used by the subroutine: `L` and `S`. The first variable, `L`, is a list whose elements are lists with the subindexes of the `n`$-$`k` non-main vectors. The variable `S` is a set with the solutions given by the subroutine `sys` and the main vectors. In this way, `absub` returns a message indicating the non-existence of `k`-dimensional abelian subalgebras or the set S if there exist k-dimensional abelian subalgebras. Since the coefficient of each main vector is 1, the system given by the subroutine `sys` has not solutions when `S` vanishes. When the system has some solution, the family of computed vectors is linearly independent and forms a basis of the subalgebra. Note that, if all the solutions in `S` contain some complex coefficient, there are no real solutions for the system solved by `sys` and there do not exist any abelian subalgebras of dimension `k` for the case $\mathbb{F} = \mathbb{R}$. For this field, it would be necessary 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:={ };
> 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 do not exist
  any abelian subalgebras of dimension k."
> else return S;
> end if;
> end proc;
```

5) Computing the basis of an abelian subalgebra for a fixed set of non-main vectors and some restrictions given by the previous subroutine.
This subroutine, named `basabsub`, receives three inputs: the dimension `n` of $\mathfrak{g}$ and two sets, `S` and `T`, with the subindexes of the non-main vectors in the basis of $\mathfrak{h}$ and some equations. This subroutine will be used with the solution given by `sys`. Four local variables `R`, `B`, `M` and `N` have been defined for its implementation. First, a conditional sentence `if`, for the sets `M` and `N`, is introduced in the cluster to find out wether `S` or `T` is the set of non-main vectors. This is due to the fact that MAPLE sometimes returns the solutions in different order. `R` is a set with the subindexes of the main vectors and, in the set `B`, we compute the basis for the abelian subalgebra. In this way, `B` is the output of this subroutine.

```
> basabsub:=proc(n,S::set,T::set)
> local R,B,M,N;
> R:={};B:={};
> if type(S,set(integer))=true then M:=S;
  N:=T else M:=T; N:=S;
> end if;
> for x from 1 to n do
> if member(x,M)=false then R:={op(R),x};
> end if; end do;
> for i from 1 to nops(R) do
> b[i]:=Z[R[i]];
> end do;
> for i from 1 to nops(R) do
> for j from 1 to nops(M) do
> b[i]:=b[i] + a[R[i],M[j]]*Z[M[j]];
> end do; end do;
> B:={seq(b[i],i=1..nops(R))};
> return eval(B,N);
> end proc:
```

6) Programming a subroutine which computes a list with all the abelian subalgebras of $\mathfrak{g}$ with certain dimension k.
This subroutine, named `listabsub`, requires two inputs: The dimension `n` of $\mathfrak{g}$ and a natural number k, less than n and which corresponds to the dimension of the abelian subalgebra. To implement it, two local variables `S` and `L` are considered. This subroutine calls the subroutine `basabsub` for computing the basis for each k-dimensional abelian subalgebra. Whereas this value is saved in the local variable `S`, `L` is a set with the basis of each abelian subalgebra of $\mathfrak{g}$ with dimension k. Precisely, the list `L` is the output of the subroutine `listabsub`.

```
> listabsub:=proc(n,k)
> local S,L;
> S:=absub(n,k);L:={};
> 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:
```

7) Computing the maximal abelian dimension of $\mathfrak{g}$ by ruling out dimensions for abelian subalgebras.
The subroutine mad receives the dimension n of $\mathfrak{g}$ as its unique input and returns the maximal abelian dimension of $\mathfrak{g}$. The subroutine starts studying if the maximal abelian dimension is n by using the subroutine absub. Then a loop is programmed to stop when absub does not find abelian subalgebras:

```
> mad:=proc(n)
> if type(absub(n,n-1),set)=true then
  return n-1
> end if;
> for i from 2 to n-1 do
> if absub(n,i)="There do not exist any abelian
  subalgebras."
> then return i-1;
> end if; end do;
> end proc;
```

8) Programming the main routine to compute a list with the basis of all the non-trivial abelian subalgebras of $\mathfrak{g}$ by using the previous subroutines.
The main routine allabsub receives the dimension n of $\mathfrak{g}$ as its unique input. The routine allabsub returns a set with the basis of all the abelian subalgebras of $\mathfrak{g}$ with dimension greater than one and less than or equal to the maximal abelian dimension of $\mathfrak{g}$. In this way, the routine starts computing the maximal abelian dimension and then, the output is defined by using the previous subroutine listabsub.

```
> allabsub:=proc(n)
> local B,k;
> k:=mad(n);B:={};
> for i from 2 to k-1 do
> B:={op(B), listabsub(n,i)};
> end do;
> return B;
> end proc:
```

IV. Statistical and computational data

In this section, a computational study is developed for the previous algorithm, which has been implemented with MAPLE 9.5, in an Intel Core 2 Duo T 5600 with a 1.83 GHz processor and 2.00 GB of RAM. Table I shows some computational data about both the computing time and the memory used to return the outputs according to the value of the dimension n of the algebra.

This computational study was done considering a particular family of Lie algebras: The Lie algebras $\mathfrak{s}_n$ generated by $\{e_1, e_2, \ldots, e_n\}$ with the following nonzero brackets:

$$[e_i, e_n] = e_i \quad \text{for } i < n.$$

This family has been chosen because they constitute a special subclass of non-nilpotent solvable Lie algebras, which allow to check empirically the computational data given for both the computing time and the used memory.

In Table I, the set of all non-trivial abelian subalgebras has been computed for the algebras in this family up to dimension $n = 13$ inclusive. Starting from $n = 8$, the computing time is about three times greater when the dimension n is increased in one unit.

TABLE I
Computing time and used memory.

Input	Computing time	Used memory
$n = 3$	0 s	0 MB
$n = 4$	0 s	0 MB
$n = 5$	0 s	0 MB
$n = 6$	0.53 s	3.44 MB
$n = 7$	0.69 s	4.25 MB
$n = 8$	1.49 s	4.31 MB
$n = 9$	3.56 s	4.5 MB
$n = 10$	10.19 s	4.81 MB
$n = 11$	26.94 s	5.19 MB
$n = 12$	77.99 s	6.25 MB
$n = 13$	238.12 s	9.88 MB

Next some brief statistics are shown about the relation between the computing time and the memory used by the implementation for the Lie algebras $\mathfrak{s}_n$.

Figure 1 shows the behavior of both the computing time and the used memory according to the dimension n of $\mathfrak{s}_n$. Note that the computing time increases more quickly than the used memory. Besides, whereas the increase of the computing time corresponds to a positive exponential model, the used memory does not follows such a model. This can be observed in Figure 2.

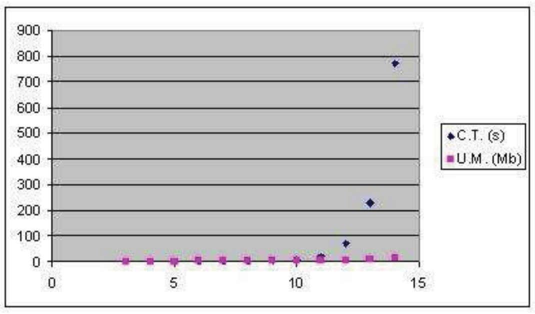
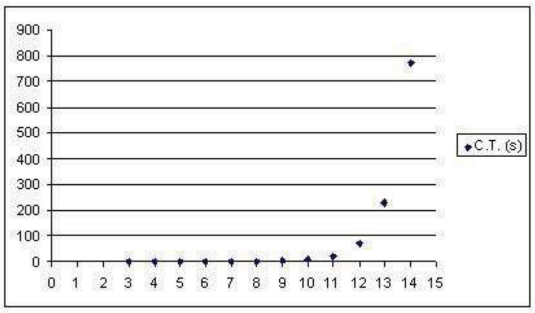


Fig. 1. Comparative graph between C.T. and U.M. with respect to dimension.

The quotients between used memory and computing time were also studied in this section. The resulting data can be observed in the frequency diagram shown in Figure 3. In this case, the behavior can be also considered exponential, although this time is negative.

V. Conclusion

In this paper the authors have described new algorithmic techniques to deal with abelian subalgebras of finite dimensional Lie algebras. We hope to continue with this research in the future in order to provide new classifications for nilpotent and solvable Lie algebras.



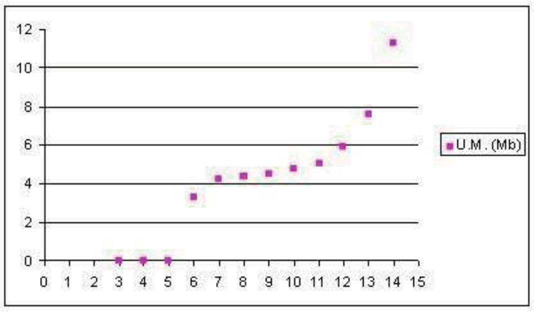


Fig. 2. Graphs for C.T. and U.M. with respect to dimension.

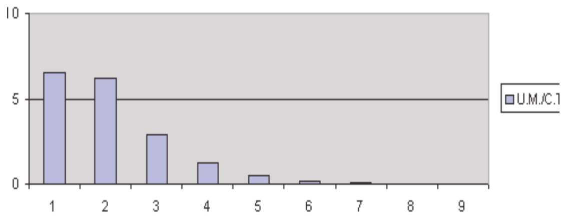


Fig. 3. Graph for quotients C.T./U.M. with respect to dimension.

References

[1] J.C. Benjumea, F.J. Echarte, J. Núñez and A.F. Tenorio, "An Obstruction to Represent Abelian Lie Algebras by Unipotent Matrices," Extracta Math., vol. 19, pp. 269–277, 2004.

[2] J.C. Benjumea, J. Núñez and A.F. Tenorio, "The Maximal Abelian Dimension of Linear Algebras formed by Strictly Upper Triangular Matrices," Theor. Math. Phys., vol. 152, pp. 1225–1233, 2007.

[3] M. Ceballos, J. Núñez and A.F. Tenorio, "The Computation of Abelian Subalgebras in the Lie Algebra of Upper-Triangular Matrices," An. St. Univ. Ovidius Constanta, vol. 16, pp. 59–66, 2008.

[4] M. Krawtchouk, "Über vertauschbare Matrizen," Rend. Circolo Mat. Palermo Serie I, vol. 51, pp. 126–130, 1927.

[5] T.J. Laffey, "The minimal dimension of maximal commutative subalgebras of full matrix algebras," Linear Alg. Appl., vol. 71, pp. 199–212, 1985.

[6] D.A. Suprunenko and R.I. Tyshkevich, "Commutative Matrices". New York: Academic Press, 1968.

[7] A.F. Tenorio, "Solvable Lie Algebras and Maximal Abelian Dimensions," Acta Math. Univ. Comenian. (N.S.), vol. 77, pp. 141–145, 2008.

[8] J.-L. Thiffeault and P.J. Morrison, "Classification and Casimir Invariants of Lie-Poisson Brackets," Phys. D, vol. 136, pp. 205–244, 2000.

[9] V.S. Varadarajan, "Lie Groups, Lie Algebras and Their Representations". New York: Springer, 1984.

Manuel Ceballos B.Sc in Mathematics from the University of Seville. He has studied a Master in Advanced Studies on Mathematics and, now, he is working in his Ph.D in this University. Besides, he held a Collaboration Fellowship in the Department of Geometry and Topology for the academic year 2006/2007 and a Research Fellowship for the Cámara Foundation of the University of Seville for the course 2008/2009. He has just been awarded with a Research Fellowship to conclude his Ph.D. for the University of Seville during this year. His contributions are related to Lie Theory.

Juan Núñez B.Sc. and Ph.D. in Mathematics from the University of Seville, where he works as a "Profesor Titular de Universidad" in the Department of Geometry and Topology (placed in the Faculty of Mathematics). His research is mainly referred to Lie groups and algebras and Discrete Mathematics, having published papers about both of them in several impact factor journals. He has also written papers about Recreational Mathematics, as well as the History and Popularization of Mathematics.

Ángel F. Tenorio B.Sc., M.Sc. and Ph.D. in Mathematics from the University of Seville. Actually he is a Lecturer in Pablo de Olavide University, in Seville (Department of Economics, Quantitative Methods and Economic History). His contributions are related to Lie Theory, Graph Theory applied to Economics, History, Popularization and Didactics of Mathematics (especially the implementation of the European Higher Education Area).