AIP Conference Proceedings

An Algorithm to Compute Abelian Subalgebras in Linear Algebras of Upper-Triangular Matrices

Manuel Ceballos, Juan Núñez, and Ángel F. Tenorio

Citation: AIP Conference Proceedings **1148**, 53 (2009); doi: 10.1063/1.3225370

View online: http://dx.doi.org/10.1063/1.3225370

View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1148?ver=pdfcov

Published by the AIP Publishing

An Algorithm to Compute Abelian Subalgebras in Linear Algebras of Upper-Triangular Matrices

Manuel Ceballos*, Juan Núñez* and Ángel F. Tenorio†

*Departamento de Geometría y Topología. Facultad de Matemáticas.
Universidad de Sevilla. Aptdo. 1160. 41080-Seville (Spain).
†Departamento de Economía, Métodos Cuantitativos e H.ª Económica. Escuela Politécnica Superior.
Universidad Pablo de Olavide. Ctra. Utrera km. 1, 41013-Seville (Spain).

Abstract. This paper deals with the maximal abelian dimension of the Lie algebra $\mathfrak{h}_n$, of $n\times n$ upper-triangular matrices. Regarding this, we obtain an algorithm which computes abelian subalgebras of $\mathfrak{h}_n$ as well as its implementation (and a computational study) by using the symbolic computation package MAPLE, where the order n of the matrices in $\mathfrak{h}_n$ is the unique input needed. Let us note that the algorithm also allows us to obtain a maximal abelian subalgebra of $\mathfrak{h}_n$.

Keywords: Maximal abelian dimension, solvable Lie algebra, algorithmic procedure, programming
PACS: 02.20.Sv; 02.10.Hh; 02.10.Ud

INTRODUCTION

The maximal abelian dimension of a given finite-dimensional Lie algebra $\mathfrak{g}$ (i.e., the maximum among the dimensions of the abelian subalgebras of $\mathfrak{g}$) has been studied in previous papers. However, most of them (e.g. [5, 10]) consider abelian ideals instead of abelian subalgebras, being needed more restrictive hypotheses. As well as other papers like [4, 6], this does not assume such restrictions, but it considers all the subalgebras of the given Lie algebra $\mathfrak{g}$.

Jacobson [8] computed a classical bound for the dimension of any abelian subalgebra $\mathfrak{a}$ of the matrix algebra $M_n(\mathbf{K})$, of $n\times n$ square matrices over a field $\mathbf{K}$: $\dim(\mathfrak{a}) \leq \left[\frac{n^2}{4}\right]+1$, where $[x]$ denotes the integer part of x. Therefore the maximal abelian dimension $\mathscr{M}(\mathfrak{g})$ of any given subalgebra $\mathfrak{g}$ of $M_n(\mathbf{K})$ can be upper bounded by:

$$\mathscr{M}(\mathfrak{g}) \leq \left[\frac{n^2}{4}\right]+1 = \begin{cases} k^2+1, & \text{if } n=2k; \\ k^2+k+1, & \text{if } n=2k+1. \end{cases}$$

We have already studied the maximal abelian dimension of the Lie algebra $\mathfrak{g}_n$, of $n\times n$ strictly upper-triangular matrices, by using an algorithmic method which computed abelian subalgebras in [1, 3]. Besides, the law of $\mathfrak{g}_n$ was computed by means of another algorithmic procedure in [2]. Now we are studying the maximal abelian dimension of the Lie algebra $\mathfrak{h}_n$, of $n\times n$ upper-triangular matrices, by applying and adjusting the technics given in [3].

The Lie algebra $\mathfrak{h}_n$ is studied due to the following reasons: First, every finite-dimensional solvable Lie algebra is isomorphic to a subalgebra of some $\mathfrak{h}_n$ [11, Proposition 3.7.3]; and secondly, its applications to Physics are many and varied (e.g. [7, 9]).

PRELIMINARIES

Some preliminary concepts are recalled here, bearing in mind that the reader can consult [11] for a general overview on Lie algebras. This paper only considers finite-dimensional Lie algebras over the complex number field $\mathbf{C}$.

A Lie algebra $\mathfrak{g}$ is to be said *solvable* if its *commutator central series* becomes zero eventually:

$$\mathscr{C}_1(\mathfrak{g})=\mathfrak{g},\ \mathscr{C}_2(\mathfrak{g})=[\mathfrak{g},\mathfrak{g}],\ \ldots,\ \mathscr{C}_k(\mathfrak{g})=[\mathscr{C}_{k-1}(\mathfrak{g}),\mathscr{C}_{k-1}(\mathfrak{g})],\ \ldots,\ \mathscr{C}_m(\mathfrak{g})\equiv\{0\}$$

The maximal abelian dimension of a finite-dimensional Lie algebra $\mathfrak{g}$ is the maximum among the dimensions of its abelian subalgebras. This value will be denoted by $\mathscr{M}(\mathfrak{g})$.

Given $n\in\mathbf{N}$, the complex solvable Lie algebra $\mathfrak{h}_n$ is that whose vectors are the $n\times n$ upper-triangular matrices:

$$h_n(x_{r,s}) = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ 0 & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & x_{nn} \end{pmatrix}$$

CP1148, Vol. 2, *Computational Methods in Science and Engineering, Advances in Computational Science*
edited by T. E. Simos and G. Maroulis

where $x_{ij} \in \mathbf{C}$, for all $i, j \in \mathbf{N}$ with $1 \leq i \leq j \leq n$.

It is easy to prove that a basis of h_n is formed by the vectors $X_{ij} = h_n(x_{r,s})$ with $1 \leq i \leq j \leq n$ and such that:

$$x_{r,s} = \begin{cases} 1, & \text{if } (r,s) = (i,j), \\ 0, & \text{if } (r,s) \neq (i,j). \end{cases}$$

So the dimension of h_n is $\frac{n(n+1)}{2}$. Besides, the nonzero brackets with respect to this basis are the following:

$$\begin{array}{ll} [X_{i,j}, X_{j,k}] = X_{i,k}, & \forall\, i = 1, \ldots, n-2,\ \forall\, j = i+1, \ldots, n-1,\ \forall\, k = j+1, \ldots, n; \\ [X_{i,i}, X_{i,j}] = X_{i,j}, & \forall\, i = 1, \ldots, n-1, \quad \forall\, j = i+1, \ldots, n; \\ [X_{k,i}, X_{i,i}] = X_{k,i}, & \forall\, k = 1, \ldots, n-1, \quad \forall\, i = k+1, \ldots, n. \end{array}$$

The center $Z(\mathrm{h}_n)$ of h_n is generated by the vector $\sum_{i=1}^{n} X_{i,i}$. This vector has to belong to any abelian subalgebra which is not contained in another.

ALGORITHM TO OBTAIN ABELIAN SUBALGEBRAS

Next, we show an algorithmic procedure to obtain abelian subalgebras of the Lie algebra h_n. Before giving a general structure for this algorithm, we study the obtainment for $n < 4$.

Lie algebra h_n with $n < 4$

Case $n = 2$: The Lie algebra h_2 is generated by the basis $\{X_{1,1}, X_{1,2}, X_{2,2}\}$ and the nonzero brackets with respect to this basis are $[X_{1,1}, X_{1,2}] = X_{1,2}$ and $[X_{1,2}, X_{2,2}] = X_{1,2}$. In this case, it is easy to prove that the 2-dimensional subalgebra $\langle X_{1,1}, X_{2,2} \rangle$ is abelian (i.e. the two vectors coming from the main diagonal). Consequently, $\mathscr{M}(\mathrm{h}_2) = 2$.

Case $n = 3$: $\{X_{1,1}, X_{1,2}, X_{1,3}, X_{2,2}, X_{2,3}, X_{3,3}\}$ is a basis of the Lie algebra h_3, whose nonzero brackets are the following:

$$[X_{1,2}, X_{2,3}] = X_{1,3};\ [X_{1,1}, X_{1,2}] = X_{1,2};\ [X_{1,1}, X_{1,3}] = X_{1,3};\ [X_{2,2}, X_{2,3}] = X_{2,3};\ [X_{1,2}, X_{2,2}] = X_{1,2};\ [X_{1,3}, X_{3,3}] = X_{1,3};\ [X_{2,3}, X_{3,3}] = X_{2,3}.$$

Step 1: Take the three vectors coming from the 3rd column and remove the one coming from the 3rd row. The abelian subalgebra $\langle X_{1,3}, X_{2,3} \rangle$ is obtained.

Step 2: Add the vectors coming from the 2nd column and remove the ones coming from the 2nd row (to avoid nonzero brackets). The dimension of the abelian subalgebra obtained here does not increase with respect to Step 1.

Step 3: Add the vector $X_{1,1} + X_{2,2} + X_{3,3}$, coming from the main diagonal and the only generator of $Z(\mathrm{h}_3)$. Hence, the 3-dimensional abelian subalgebra$\langle X_{1,2}, X_{1,3}, X_{1,1} + X_{2,2} + X_{3,3} \rangle$ is obtained.

A general structure for the algorithm

Now we explain the algorithm to obtain abelian subalgebras. This depends on the parity of n, as it can be seen next. Hence, two possible cases have to be considered:

Case 1: n is even and $n \geq 4$ (i.e., $n = 2k$, with $k \in \mathbf{N} \setminus \{1\}$). The general reasoning consists on considering the vectors in the basis of h_n. When a vector coming from the i^{th} column is chosen, only the vectors coming from the i^{th} row lead to nonzero brackets. To avoid nonzero brackets, the vectors coming from the i^{th} row have to be removed.

Step 1: $(2k)^{\text{th}}$ column. Add the $2k$ vectors coming from the $(2k)^{\text{th}}$ column and remove the unique vector coming from the $(2k)^{\text{th}}$ row. In this way, the abelian subalgebra $\langle X_{1,2k}, \ldots, X_{2k-1,2k} \rangle$ is obtained.

Step $2k - i + 1$: i^{th} column, with $2k > i > k+1$. Add the i vectors coming from the i^{th} column to the generators of the previous step and remove the $2k - (i-1)$ vectors coming from the i^{th} row. In this way, we obtain an abelian subalgebra whose dimension increases $2i - 2k - 1$ with respect to the already obtained in the previous step. The dimension really increases if and only if $2i - 2k - 1 > 0$. Since this inequality is equivalent to $i > k + 1/2$, k is the last step in which the dimension of the abelian subalgebra increases.

Step k: $(k+1)^{\text{th}}$ column. Add the $k+1$ vectors coming from the $(k+1)^{\text{th}}$ column and remove the k ones coming from the $(k+1)^{\text{th}}$ row.

Step $k+1$**:** Add the vector $\sum_{i=1}^{n} X_{i,i}$, obtaining the (k^2+1)-dimensional abelian subalgebra generated by:

$$\begin{array}{ccc} X_{1,k+1} & \dots & X_{1,2k} \\ X_{2,k+1} & \dots & X_{2,2k} \\ \vdots & \ddots & \vdots \\ X_{k,k+1} & \dots & X_{k,2k} \end{array} \qquad \sum_{i=1}^{n} X_{i,i}$$

Case 2: n is odd and $n \geq 3$ (i.e., $n = 2k+1$, with $k \in \mathbf{N} \setminus \{1\}$). With a reasoning analogous to Case 1, we can settle the following algorithm to obtain abelian subalgebras up to a dimension as large as possible.

Step 1: $(2k+1)^{\text{th}}$ column. Add the $2k+1$ vectors coming from the $(2k+1)^{\text{th}}$ column and remove the unique vector coming from the $(2k+1)^{\text{th}}$ row, obtaining the abelian subalgebra $\langle X_{1,2k+1}, \dots, X_{2k,2k+1} \rangle$.

Step $2k-i+2$**:** i^{th} column, with $2k+1 > i > k+2$. Add the i vectors coming from the i^{th} column to the generators in the previous step and remove the $2k-(i-1)$ vectors coming from the i^{th} row. In this way, we obtain an abelian subalgebra whose dimension increases $2i-2k-2$ with respect to the obtained in the previous step. The dimension increases in each step if and only if $2i-2k-2 > 0$; which is equivalent to $i > k+1$. Hence, Step k is the last step in this adding-and-removing procedure.

Step k**:** $(k+2)^{\text{th}}$ column. Add the $k+2$ vectors coming from the $(k+2)^{\text{th}}$ column and remove the k ones coming from the $(k+2)^{\text{th}}$ row, obtaining a (k^2+k)-dimensional abelian subalgebra.

Step $k+1$**:** Add the vector $\sum_{i=1}^{n} X_{i,i}$, obtaining the (k^2+k+1)-dimensional abelian subalgebra generated by:

$$\begin{array}{ccc} X_{1,k+2} & \dots & X_{1,2k+1} \\ X_{2,k+2} & \dots & X_{2,2k+1} \\ \vdots & \ddots & \vdots \\ X_{k+1,k+2} & \dots & X_{k+1,2k+1} \end{array} \qquad \sum_{i=1}^{n} X_{i,i}$$

Summarizing the whole section, abelian subalgebras of $\mathfrak{h}_n$ can be computed for all $n \in \mathbf{N}$. Besides, the maximal dimension of these subalgebras can be expressed depending on n as follows:

$$B_n = \begin{cases} k^2+1, & \text{if } n = 2k; \\ k^2+k+1, & \text{if } n = 2k+1. \end{cases}$$

Hence, B_n is a lower bound of the maximal abelian dimension $\mathscr{M}(\mathfrak{h}_n)$ and is equal to Jacobson's upper bound (see [8]). So the algorithm computes abelian subalgebras, whose dimensions are increasing up to the value of $\mathscr{M}(\mathfrak{h}_n)$.

IMPLEMENTING THE ALGORITHM WITH MAPLE

Next, we explain a step-by-step implementation of the algorithm. The order n of the matrices in $\mathfrak{h}_n$ is the unique input, whereas the outputs are both the maximal abelian dimension and a maximal abelian subalgebra of $\mathfrak{h}_n$. The isomorphism classes of abelian subalgebras of $\mathfrak{h}_n$ can be also obtained by removing vectors from the maximal abelian subalgebra.

To implement the algorithm, a routine with two subroutines has been programmed. Two libraries are loaded to activate additional commands: `ListTools` and `numtheory`. The routine `mas` (from maximal abelian subalgebra) receives the order n of the matrices in $\mathfrak{h}_n$ and it returns a basis of a maximal abelian subalgebra. Moreover, an implicit list with all the isomorphism classes of abelian subalgebras can be obtained from this second output of the routine.

The first subroutine, `add_mas`, determines the vectors to be added in each step to the basis. The input is a natural number j which corresponds to the column considered by the routine `mas`. A list L is also defined as a local variable and its elements are the vectors to be added. This list is the output of the subroutine.

```
> add_mas:=proc(j)
> local L;
> L:=[];
> for k from 1 to j do L:=[op(L),X[k,j]];
> end do;
> return op(L[1..nops(L)]);
> end proc:
```

The second subroutine, `remove_mas`, computes the vectors which have to be removed in each step of the computation. By inputting two natural numbers, i and n, the subroutine removes the vectors coming from the i^{th} row in the matrices in $\mathfrak{h}_n$. To program this subroutine, a local variable M is defined to store a list with the vectors to be removed. The list M is the output of this subroutine.

```
> remove_mas:=proc(i,n)
> local M;
> M:=[];
> for k from i to n do
> M:=[op(M),X[i,k]];
> end do;
> return op(M[1..nops(M)]);
> end proc:
```

The routine `mas` computes both the basis of a maximal abelian subalgebra of $\mathfrak{h}_n$ and the dimension of this subalgebra (i.e. the maximal abelian dimension). Its unique input is the natural number n, corresponding to the order of the matrices in $\mathfrak{h}_n$. We have defined the local variables L, M, P, Q and i, where L, M, P and Q are sets and i is a natural number. L and M store the vectors to be added and removed, respectively. P stores the set difference $L \setminus M$. Finally, Q has a unique element: the generator of $Z(\mathfrak{h}_n)$. Hence, the routine computes these sets in each step, returning the union $P \cup Q$ (i.e. a maximal abelian subalgebra) and its cardinal (i.e. the maximal abelian dimension).

```
> mas:=proc(n)
> local L,M,P,i,Q;
> L:={};M:={};P:={};i:=n;Q:={X[1,1]};
> while i>iquo(n,2) do
> L:={op(L),add_mas(i)};M:={op(M),remove_mas(i,n)};i:=i-1;P:=L minus M;
> end do;
> for j from 2 to n do Q:={op(Q)+X[j,j]};
> end do;
> Q:=P union Q;
> return {Q,nops(Q)};
> end proc;
```

COMPUTATIONAL STUDY

The algorithm was implemented by using MAPLE 9.5 and was run in an Intel Core 2 Duo T 5600 with a 1.83 GHz processor and 2.00 GB of RAM. Table 1 shows some computational data about this implementation for $n < 2000$. Starting from $n = 2$, the computational time is apparently double when the order n is increased fifty units. In this way, for $n = 1000$, the routine runs about 1 minute to compute a maximal abelian subalgebra of $\mathfrak{h}_{1000}$. For orders closer to 2000, some problems arise in relation to an insufficient computational capacity. These problems are motivated by the need of a very high capacity of memory (almost 2 GB). In any case, it does not suppose a serious problem because for this order the dimension of the Lie algebra would be about $2 \cdot 10^6$, which is not usually needed in practical situations.

TABLE 1. Computational time and used memory.

Input (order n)	Dimension of $\mathfrak{h}_n$	Computational time	Used memory	Input (order n)	Dimension of $\mathfrak{h}_n$	Computational time	Used memory
2	3	0 s	0 MB	400	80200	2.453 s	18.75 MB
50	1275	0 s	0 MB	500	125250	5.468 s	40.99 MB
100	5050	0.046 s	3.31 MB	600	180300	9.796 s	63.55 MB
150	11325	0.109 s	5.69 MB	800	320400	27 s	157.04 MB
200	20100	0.266 s	7.56 MB	1000	500500	59.266 s	353.81 MB
250	31375	0.562 s	11.88 MB	1250	781875	156.641 s	645.26 MB
300	45150	0.969 s	15.19 MB	1500	1125750	303.499 s	1071.24 MB
350	61425	1.609 s	16.31 MB	1750	1532125	522.061 s	1701.19 MB

After carrying out a brief statistical study about the relation between the computational time and the memory used to compute the maximal abelian dimension of $\mathfrak{h}_n$, we can assert that the memory used depends on the computational time and viceversa. Indeed, this dependence is expressed by a very strong positive linear correlation. Besides, both the memory used and the computational time seem to be exponentially related to the order n.

REFERENCES

1. J. C. Benjumea, F. J. Echarte, J. Núñez and A. F. Tenorio, *Extracta Math.* **19**, 269–277 (2004).
2. J. C. Benjumea, J. Núñez and A. F. Tenorio, "A computational method to obtain the law of the nilpotent Lie algebras $\mathfrak{g}_n$", in *Proceedings of Transgressive Computing 2006*, edited by J. G. Dumas. Granada, 2006, pp. 53–62.
3. J. C. Benjumea, J. Núñez and A. F. Tenorio, *Theor. Math. Phys.* **152**, 1225–1233 (2007).
4. R. Campoamor-Stursberg, *Acta Phys. Polon. B* **37**, 2745–2760 (2006).
5. C. Caranti, S. Mattarei and F. Newman, *Trans. Amer. Math. Soc.* **349**, 4021–4051 (1997).
6. D. Gorenstein, R. Lyons and R. Solomon, *The classification of the finite simple groups, vol. 3*, AMS, Providence, 1997.
7. S. Hervik, *J. Geom. Phys.* **52**, 298–312 (2004).
8. N. Jacobson, *Bull. Amer. Math. Soc.* **50**, 431–436 (1944).
9. P. J. Olver, *Applications of Lie Groups to Differential Equations*, Springer, New York, 1986.
10. J.-L. Thiffeault and P. J. Morrison, *Phys. D* **136**, 205–244 (2000).
11. V. S. Varadarajan, *Lie Groups, Lie Algebras and Their Representations*, Springer, New York, 1984.