# Computing Theta-Stable Parabolic Subalgebras Using LiE

Alfred G. Noël

The University of Massachusetts, Department of Mathematics, Boston, MA 02125-3393, USA
Massachusetts Institute of Technology, Department of Mathematics, Cambridge, MA 02139-4307, USA

**Abstract.** This paper describes an algorithm for computing representatives of conjugacy classes of $\theta$-stable parabolic subalgebras of a semisimple complex Lie algebra $g_{\mathbb{C}}$ relative to any of its non-compact real forms $g$ of inner type. These subalgebras are important for studying representations of $g$.

## 1 Introduction

The notion of a $\theta$-stable parabolic subalgebra of $g_{\mathbb{C}}$ was introduced by Vogan [**1**] in the 1970's in order to study representations of semisimple Lie groups. Since then such subalgebras have been used by several authors to understand certain questions related to the theory of nilpotent orbits. In many instances, it is desirable to compute representatives of certain classes of such subalgebras under the action of a given Lie group. In this paper we propose an algorithm for computing such representatives under the action of a connected complex Lie group $K_{\mathbb{C}}$. The algorithm was implemented in the computer algebra system LiE [**2**] and was used to show that a theorem of Peter E. Tapa for classical real Lie groups does not extend to exceptional Lie groups [**3**]. It is also being used to study *polarization* in the exceptional Lie groups [**4**].

Let $g$ be a real semisimple Lie algebra with adjoint group $G$ and $g_{\mathbb{C}}$ its complexification. Also let $g = k \oplus p$ be the Cartan decomposition of $g$ where $k$ is a Lie algebra and $p$, a vector space. Finally, let $\theta$ be the corresponding Cartan involution of $g$. Then $g_{\mathbb{C}} = k_{\mathbb{C}} \oplus p_{\mathbb{C}}$ where $k_{\mathbb{C}}$ and $p_{\mathbb{C}}$ are obtained by complexifying $k$ and $p$ respectively. Denote by $K_{\mathbb{C}}$ the connected subgroup of the adjoint group $G_{\mathbb{C}}$ of $g_{\mathbb{C}}$, with Lie algebra $k_{\mathbb{C}}$. Then $k_{\mathbb{C}}$ and $p_{\mathbb{C}}$ are the $+1$-eigenspace and the $-1$-eigenspace of the extension of $\theta$ on $g_{\mathbb{C}}$ respectively. We shall call such an extension $\theta$ also. The subgroup $K_{\mathbb{C}}$ preserves $p_{\mathbb{C}}$ under the adjoint action.

Let $h_{\mathbb{C}}$ be a Cartan subalgebra and $\Phi = \Delta(g_{\mathbb{C}}, h_{\mathbb{C}})$ the root system of $g_{\mathbb{C}}$ determined by $h_{\mathbb{C}}$. A **Borel subalgebra** of $g_{\mathbb{C}}$ is a subalgebra $b = h_{\mathbb{C}} \oplus n$ where $n = \bigoplus_{\alpha \in \Phi^+} g_{\mathbb{C}}^{\alpha}$ for some positive system $\Phi^+$ within $\Phi$ and $g_{\mathbb{C}}^{\alpha}$ denotes the root space of $\alpha$. Any subalgebra $q$ of $g_{\mathbb{C}}$ containing a Borel subalgebra is called a **parabolic subalgebra** of $g_{\mathbb{C}}$. If $q = q \cap k_{\mathbb{C}} \oplus q \cap p_{\mathbb{C}}$ then we shall say that $q$ is a $\theta$-stable parabolic subalgebra of $g_{\mathbb{C}}$. Any parabolic subalgebra decomposes as $q = l + u$

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 335–342, 2004.
© Springer-Verlag Berlin Heidelberg 2004

where $l$ is a Levi subalgebra of $g_{\mathbb{C}}$ and $u$, the nilradical of $q$, is a vector space consisting of nilpotent elements of $g_{\mathbb{C}}$.

We shall give an algorithm to compute representatives of all the $K_{\mathbb{C}}$-conjugacy classes of $\theta$-stable parabolic subalgebras of $g_{\mathbb{C}}$ when $g$ is of inner type, that is, when $rank(g) = rank(k)$, and $g_{\mathbb{C}}$ simple. Slight modifications of the algorithm are necessary to handle the cases where $g$ is not of inner type. More will be said on this at the end.

## 2 Algorithm Description and Proof of Correctness

The algorithm is divided into two main components:

i. Computation of the $K_{\mathbb{C}}$-conjugacy classes of systems of simple roots. This task will be performed by the function Compute_Chambers().

ii. Computation of representatives of conjugacy classes of $\theta$-stable parabolic subalgebras. The function Compute_Parabolics() will perform this computation.

The algorithm will be written in "pidgin" LiE, that is, we shall use a lot of built-in functions from LiE. Readers should consult [**2**] to find information on such functions. The built-in functions will be written in italics in the definition of the algorithm. We should point out that the algorithm can be implemented on any computer algebraic systems containing some Lie-theoretical capabilities. All variables will be of type integer. This is the only type that LiE accommodates. To see a very brief evaluation of LiE consult our ICCS 2003 paper [**5**]. Finally, we shall use the notation $\|S\|$ for the cardinality of a set $S$ and comments will be placed between square brackets.

**Description of the algorithm.**

**Input:**

G: semisimple type of $G_{\mathbb{C}}$

K: semisimple type of $K_{\mathbb{C}}$

$R_{nc}$: set of non compact roots of $G_{\mathbb{C}}$

**Output:**

P: one-dimensional array of integers containing the indices of the roots in each parabolic subalgebra.

P_addr: two-dimensional array of integers containing the start and the end of each parabolic subalgebra in P.

This variable plays an important role when processing the parabolic subalgebras.

num_parabolics: number of representatives computed

**Begin** [ Main]

n = *n_pos_roots*(G);

$l$ = *Lie_rank*(G);

[ Compute the number of $K_{\mathbb{C}}$-conjugacy classes of root systems ]

n_ch = ‖W(G)‖/‖W(K)‖; [ W(G) and W(K) are the Weyl groups of G and K ]

P = *null*(n_ch$*2^l, l$); [ Create and initialize P ]

P_addr = *null*(n_ch$*2^l$, 2);

```
    dim_par = null(n_ch*2^l);
    [ Initialize lists and queues ]
    ch = null (n_ch*l,l); qu = null ((n_ch*l),l);
    cur_ch =null(l,l); new_ch = null(l,l);
    [ Initialize counters ]
    l_count = 1; q_indx = l; l_indx = l;
    d_index = 0;
    Compute_Chambers();
    Compute_Parabolics();
End [ Main]
```

**Compute_Chambers();**

**begin**

[ This algorithm computes the $K_{\mathbb{C}}$ conjugacy classes of simple roots systems as follows: Starting with a Vogan system in the usual bourbaki system it looks for other non $K_{\mathbb{C}}$-conjugate systems by performing reflection along non compact imaginary roots ]

```
    for i = 1 to l do ch[i] = pos_roots(G)[i] ; qu[i] = pos_roots(G)[i]; od;
    [ Main Loop ]
    while l_count < n_ ch do
    [ reflect through all non compact roots in current chamber ]
    [ be sure that both queue and list are properly maintained ]
    for i =1 to l do cur_ch[i] = q[i];od; [ dequeue ]
    for  i = 1 to l do

    if  cur_ch[i] ∈ R_nc then
    [ this is a noncompact root reflect through it ]
    for k = 1 to l do
    new_ch[k] = cur_ch[k] - Cartan(cur_ch[k], cur_ch[i], G)
    *cur_ch[i]; od;
    [ check for duplicate ]
    for j =1 to l do if new_ ch[j] != ch[(k-1)*l + j] then
    uniq = 0; break; fi;  od;
    if uniq == 1 then break; fi; od;
    if  uniq ==0 then for k = 1 to l do ch[l_indx + k] = new_ch[k];
    qu[q_indx +k] = new_ch[k]; od; l_indx = l_indx + l;
    q_indx = q_indx + l;
    fi; fi; od;

    for i = 1 to (q_indx- l) do qu[i] = qu[i+l]; od; [ rebuild queue ]
    q_indx = q_indx - l;
    od; [Main Loop]
end [ Compute_Chambers() ]
```

**Compute_Parabolics();**

[ This algorithms computes a list which contains all the representatives of the $K_{\mathbb{C}}$-conjugacy classes of $\theta$-stable parabolic subalgebras of $g_{\mathbb{C}}$. Since each chamber given by the previous algorithm is equivalent to a $\theta$-stable Borel subalgebra the algorithm builds standard parabolic subalgebras in each chamber by using the subsets of the chamber. ]

**begin**

chamb = null (l,l); i = 1; par_index = 0;
**for** j = i **to** (i+l-1) **do** count= count+1; chamb[count]= ch[j];**od**;
[ initialize the nilradical of the Borel subalgebra generated by chamb ]
u = null (*n_pos_roots*(G), l);
**for** u_indx = 1 **to** *n_pos_roots*(G) **do**
**for** r_indx = 1 **to** l **do** u[u_indx] = u[u_indx]+ pos_roots(G)[u_indx,r_indx]*
chamb[r_indx]; **od**; **od**;
[ find all subsets of the simple roots in the class
and build the resulting parabolic $q = l \oplus u$ ]
cc[ii] = ii; **od**;
null(kk,l);
[ retrieve the subset of simple roots defining the Levi subalgebra ]
chamb[cc[ii]];**od**;
l_matrix= null (kk,l); [ Cartan matrix for the Levi subalgebra ]
**for** i_l = 1 **to** kk **do for** j_l =1 **to** kk **do**
l_matrix[i_l,j_l] = *Cartan*(G)(lev_roots[i_l],lev_roots[j_l]); **od**; **od**;
g = *Cartan_type*(G)(lev_roots); m_cartan = *Cartan*(G)(g);
**if** l_matrix $\neq$ m_cartan **then**
good_lev = lev_roots; nnn = kk; pi =null(nnn+2); p = null(nnn+1);
d = null(nnn+1); previous = null(nnn); current = null(nnn);
**for** iii = 2 **to** nnn+1 **do** pi[iii] = iii; p[iii] = iii; d[iii] = -1; **od**;
d[1] = 0; m = nnn+2; pi[1] = m; pi[m] = m; counter = 0; **for** c = 2 **to** nnn+1
**do** current[c-1] = pi[c] - 1; **od**;
lev_roots[current[i_l]] **od**; **to** kk **do** l_matrix[i_l,j_l] =
*Cartan*(G)(good_lev[i_l],good_lev[j_l]); **od**; **od**;
**if** l_matrix == m_cartan **then** lev_roots =good_lev ; break; fi;
[ right order found ]
m = nnn+1; **while** pi[p[m]+d[m] ] >m **do** d[m] = -d[m]; m = m-1;
**if** m ==1 **then** break; **fi**; **od**;
bucket = pi[p[m]]; pi[p[m]] = pi[p[m]+d[m]]; pi[p[m]+d[m]] = bucket;
previous = current;
**od**; **fi**;
[ end permutation, continue to process subsets ]
jj=kk;
**while** cc[jj]== (l-kk +jj) **do** jj = jj-1; **if** jj == 0
**then** break **fi**; **od**;
**if** jj !=0 **then** cc[jj] = cc[jj] +1 **fi**;
**for** ii = jj+1 **to** kk **do if** ii == 1 **then** cc[ii] = 1 **else**
cc[ii] = cc[ii-1] +1; **fi** ;**od**;

```
    n_pieces = n_comp(g);
    nilp_u = null(n,l); levi_index = 0; nilp_u_index = 0; lev_ptr = 0;
    sg = g[ii]; l_p = Append ( pos_roots(sg),  (-pos_roots(sg)));
    for jk = 1 to 2 n_pos_roots(sg) do
    levi_index = levi_index +1;
    for ll=1 to Lie_rank(sg) do
    levi_subalg[levi_index] = levi_subalg[levi_index] +
    l_p[jk,ll]*lev_roots[lev_ptr + ll]; od;od;
    lev_ptr = lev_ptr + Lie_rank(sg); od;
    for ik = 1 to n do trouver = 0;
    break;fi; od;
    if trouver == 0  then nilp_u_index = nilp_u_index + 1;
    nilp_u[nilp_u_index] = u[ik]; fi; od;
    [ check for duplicate subalgebras and build the list]
    found = 0; u_qq = null(nilp_u_index,l);
    for ik = 1 to nilp_u_index do u_qq[ik] = nilp_u[ik]; od;
    q = sort(Append (levi_subalg,u_qq));  [ q = l ⊕ u ]
    dimq = levi_index+nilp_u_index;
    data = Append (levi_subalg u_qq);
    if n_parabolics == 0 then
    n_parabolics = n_parabolics + 1; P_addr[1,1] = 1; P_addr[1,2] = dimq;
    for ik =1 to dimq do par_index = par_index +1;
    P[par_index] = data[ik]; od;
    d_index = d_index +1; dim_par[d_index] = dimq+l;
    else found = 0; [ Check for duplicates ]
    P[P_addr [ik,1] +jk -1] od; break; fi; fi; ood;
    dimq do par_index = par_index +1;
    P[par_index] = data[ik]; od;
    d_index = d_index +1; dim_par[d_index] = dimq+l;
    fi; fi; od; od; od;
end  [ Compute_Parabolics() ]
```

**Remark**. At the end of Compute_Parabolics() the list P will contain representatives of all classes of parabolic subalgebras except those of the Borel subalgebras. However, the Borel subalgebras are completely determined by the roots stored in the variable **chamb** which defines the Cartan subalgebra equivalent to the Levi subalgebra in this case. The variable **u** contains the appropriate positive roots and is in fact the nilradical of the Borel representative.

### Proof of correctness

**Theorem.** *The above algorithm is correct.*

**Proof.** Maintaining the above notations, it is known that the parabolic subalgebras $q$ containing a Borel subalgebra $b$ of $g_{\mathbb{C}}$ are parametrized by the set of subsets of $\Delta$ the set of simple roots that defines $b$ ( See **[7]** Proposition 5.90 for a proof). Let $\Phi$ be the root system generated by $\Delta$ and let $\Gamma$ be a subset of $\Delta$. Define $q_\Gamma$ to be the subalgebra of $g_{\mathbb{C}}$ generated by $h_{\mathbb{C}}$ and all of the root

spaces $g_{\mathbb{C}}^{\alpha}$ such that $\alpha \in \Delta$ or $-\alpha \in \Gamma$. Let $\langle \Gamma \rangle$ denote the subroot system of $\Phi$ generated by $\Gamma$ and put $\langle \Gamma \rangle^+ = \langle \Gamma \rangle \cap \Phi^+$. Define

$$l = h_{\mathbb{C}} \oplus \bigoplus_{\alpha \in \langle \Gamma \rangle} g_{\mathbb{C}}^{\alpha} \qquad u = \bigoplus_{\alpha \in \Phi^+ \setminus \langle \Gamma \rangle^+} g_{\mathbb{C}}^{\alpha}$$

Then $q_\Gamma = l \oplus u$ is a parabolic subalgebra containg $b$ and is said to be a standard parabolic subalgebra. Moreover every parabolic subalgebra of $g_{\mathbb{C}}$ is conjugate to a standard parabolic subalgebra of $g_{\mathbb{C}}$. Since we assume that $g$ is of inner type we conclude that all parabolic subalgebras are $\theta$-stable. The above argument is valid for each $k_{\mathbb{C}}$-conjugacy class of Borel subalgebras. Hence, the algorithm generates a list containing representatives of all the $k_{\mathbb{C}}$-conjugacy classes of $\theta$-stable parabolic subalgebras of $g_{\mathbb{C}}$.

In order to finish the proof we need to show that the computation $\langle \Gamma \rangle$ is correct. This is done in Compute_Parabolics() by obtaining subsets of $\Delta$ and permuting the set of roots in such subsets when necessary. We generate the permutations using minimal change order as described in [**6**]. To compute subsets of $\Delta$ we use an implementation of Algorithm 5.8 in [**6**] also. The proofs of correctness of both algorithmic schemes are found in [**6**]. Hence, the theorem follows.

□

**Complexity**

The complexity of the algorithm depends on that of the built-in functions. In LiE such functions are well designed and seem to perform in an optimal manner. Since LiE is not a multipurpose software package, the designers were able to use clever and faster algorithms to enhance performance. The reader should realize that most of the work in this algorithm is done by Compute_Parabolics() which computes subsets and permutations of elements of $\Delta$ which is of size $l$ the rank of $g_{\mathbb{C}}$. The analysis of both schemes is done in [**6**] and it is not too difficult to see that the permutation scheme will take $\mathcal{O}(k!)$ to permute $k$ elements of $\Delta$ and the determination of the subsets of size $k$ is proportional to the number of combinations of subsets of $k$ elements of $\Delta$ that is $\mathcal{O}(\binom{l}{k})$. Hence both schemes perform quasi-optimally. Of course, this is a worst case analysis. For $l \leq 8$ the algorithm performs very well on average. We are not in a position to give an average case analysis at this time. However the reader should know that we were able to compute the representatives of the classes of theta-stable parabolic subalgebras for all the exceptional non compact simple Lie groups relative to all their real forms of inner type. This is not trivial [**4**]. The computations were carried on an IMac G4 with speed 1GHz and 1Gb SDRAM of memory.

## 3 Some Applications of Representation Theory

A representation of a group is a mathematical map which associates a matrix to each element of the group. Matrices are very concrete objects that facilitate difficult computations which would be impossible otherwise. This was recognized

after the discovery of quantum mechanics. Hence, given a group, if all or a lot of its representations are available then the investigator has a better chance of finding one which fits the problem at hand. Finding all the representations of a given reductive real Lie group is one the most important unsolved problems in Mathematics.

An other incentive to study Representation theory comes from Number theory. Here, we should point out that the field of Number theory, although considered one of the purest branches of mathematical thought, turns out to have very important and concrete applications in our industrial world. One such application is the design and development of almost unbreakable codes in cryptography allowing the possibility of making transactions on the internet using credit cards.

Representation theory is used in quantum chemistry, quantum computing, construction of telephone networks, radar and antenna design, robotics, coding theory, computer vision and many other branches of science and engineering. Readers who are interested in real world applications of Representation theory should visit the following website:

*http* : *//web.usna.navy.mil/wdj/repn_thry_appl.htm*

The work presented in this paper is part of a program whose aim is to compute new representations of reductive real Lie groups. See [**8**] for more details.

## 4 Conclusion

In this paper we proposed and used and algorithm which produces a list containing representatives of all the $K_{\mathbb{C}}$-conjugacy classes of theta-stable parabolic subalgebras of a complex simple Lie algebra $g_{\mathbb{C}}$ relative any of its real non-compact forms $g$ of inner type. We proved the correctness of the algorithm and gave a worst case analysis of its complexity. We also mentioned that the average performance of the algorithm is quite good because we were able to use it to compute data from all the exceptional simple Lie groups. However we still have more work to do. First, we need to extend the algorithm to the cases where $g$ is not of inner type. This can be done as follows: the group $K_{\mathbb{C}}$ should be replaced by $G^{\theta}_{\mathbb{C}}$ the subgroup of $G_{\mathbb{C}}$ that fixes $k_{\mathbb{C}}$ and the computation of the theta stable parabolic will be more complicated. We should be able to do this soon. In order to manage space more effectively we need to develop a formula for the number of $K_{\mathbb{C}}$-conjugacy classes of theta-stable parabolic subalgebras. We have not been able to find such a formula in the literature. From our conversation with experts there are reasons to believe that the formula is not known. One way to circumvent this issue is to use the Weyl group of $K_{\mathbb{C}}$. This solution does not scale well because the Weyl group grows fast as the rank of $g$ increases and traversing it becomes a challenging problem. We are currently developing some new strategies to solve this problem.

There are also some software issues. LiE provides essentially two types of data structures, the **vector**, a one-dimensional array of type integer, and the **matrix**, a two-dimensional array of type integer and does not allow dynamic allocation. These two factors complicate the handling of large data sets. Since the LiE source

code is available we plan to solve these problems in the future. We believe that in general the mathematical algorithms in LiE are well designed. However we would welcome some serious work on a good user-interface.

Many mathematicians are currently using Computer Algebra Systems in their research not only as simulation tools but also as a way of generating important counterexamples and conjectures. As these systems become more and more powerful we should expect a stronger cooperation between mathematicians, system designers and computer scientists.

**Acknowledgment.** The author wishes to thank the referees for their helpful and insightful comments.

## References

1. Vogan D. jr:*The algebraic structure of the representation of semisimple Lie groups I*, Annals of Math. **109** (1979), 1-60
2. Van Leeuwen M. A. A., Cohen A. M., Lisser B.:*LiE A package for Lie Group Computations*, Computer Algebra Nederland, Amsterdam The Netherlands (1992)
3. Noël A. G.: *Appendix to "Richardson Orbits for Real Classical Groups" by Peter E. Trapa ( Counterexamples in $F_4$)* , to appear in Journal of Algebra
4. Noël A. G.: *Some remarks on Richardson Orbits in Complex Symmetric Spaces*, (preprint)
5. Noël A. G.: *Computing maximal tori using LiE and Mathematica*, Lectures Notes in Computer Science, Springer-Verlag. **2657** (2003) 728-736
6. Reingold E. M., Nievergelt J., Deo N.: *Combinatorial Algorithms Theory and Practice*, Prentice-Hall (1977)
7. Knapp A. W.:Lie Groups Beyond and introduction second edition, Birkhaüser Progress in Mathematics **140** (2002)
8. Peter E. Trapa.: *Richardson Orbits for Real Classical Groups* , to appear in Journal of Algebra