```
# The computation presented here are related to the article:
# M. Bulois, On the normality of the null-fiber of the moment map for
theta- and tori representations, https://arxiv.org/abs/1706.05456
#
#They were made using GAP (version 4.8.3) and W. deGraaf's GAP package
"sla" (http://www.science.unitn.it/~degraaf/sla.html, version 1.2)
#
# More precisely, it contains the  computations justifying the claims
made
#   -in Proposition 4.10 (list of locally free stable theta-
representations of rank 1 in the exceptional cases), see function
"Stable_rk1_Inner" for the inner cases and "Stable_rk1_Outer" for the
outer cases
#   -in Lemma 4.11 (classification of the locally free stable theta-
representations of rank 1 where property (P) holds), see functions
"Has_regs_check" and "Really_Non_reg"
#   -in Lemma 4.12 (the non-existence of locally free theta-rep having a
locally free stable proper Levi of rank 1 where property (P) does not
hold.), see function "ListOfpossLocfreec"
#
# The file is divided in 3 parts. In the first part "Functions" we define
the functions used in the second part "Computations"
#
#
#-----------------------------------------
# A. Functions
#-----------------------------------------

LoadPackage("sla"); #Loading the "sla" package


prodscal:=function(l1,l2) #computes the scalar product of two lists
local i,k;
k:=0;
for i in [1..Length(l1)] do
k:=k+(l1[i]*l2[i]);
od;
return k;
end;;


deuxn:=function(n) #returns a list of all non-zero (n+1)-uples with value
in {0,1}
local l,i,s;
l:=[List([1..n],x->0)];
Add(l[1],1);
for i in [2..(2^(n+1)-1)] do
    s:=n+1;
    Add(l,StructuralCopy(l[i-1]));
    while l[i-1][s]=1 do
      l[i][s]:=0;
      s:=s-1;
    od;
    l[i][s]:=l[i-1][s]+1;
od;
return l;
end;;
```

```
ListOfTheta:=function(R) #R is a Root system, the inner gradings on the
corresponding Lie algebra g are classified by labellings of the extended
Dynkin diagrams.
                         #The function computes the list of labellings in
{0,1} (see above Lemma 4.9 in the article to see why restricting to such
labellings) giving rise to a grading on g such that $dim g_1-dim g_0=1$.
local S,P,n,l,s,i,g0,g1,k,m,wp,p,hir,res,lindep;
n:=Length(SimpleSystem(R));  #rank of the root system
P:=PositiveRootsNF(R);   #list of positive roots, written as linear
combinations of the simple roots.
hir:=ShallowCopy(P[Length(P)]); #coordinates of the highest root
lindep:=[1];
Append(lindep,hir);      #1 added on the extended node of the affine
Dynkin diagram (arbitrarily sent at the start of the list),
                         #thus the coefficients in lindep are those of
the linear dependency among simple roots of the roots in the extended
Dynkin diagram.
l:=deuxn(n);             #list of all possible labellings (for untwisted
Kac diagram giving rise to an action with positive rank).
res:=[];                 #the selected gradings will be stored in res
for s in l do
    g1:=0;               #g1 will store the dimension of g_1
    g0:=n;               #g0 will store the dimension of g_0. Since g_0
contains a Cartan subspace, g0 is initialized to the rank before looking
at root_spaces.
    m:=prodscal(lindep,s);  #m stores the order of the grading
    for p in P do
      wp:=(prodscal(s{[2..(n+1)]},p) mod m);                  #the root p
lies (modulo m) in the wp-th part of the grading induced by the labelling
s.
      if wp=1 then g1:=g1+1; fi;       #1 dimension added to g1 if the
root space corresponding to p lies in g_1.
        if wp=m-1 then g1:=g1+1; fi;     #1 dimension added to g1 if the
root space corresponding to p lies in g_{-1} (the root spaces for the
corresponding negative root lie in g_1).
      if wp=0 then g0:=g0+2; fi;         #2 dimension added to g0 if the
root space corresponding to p lies in g_0 (since the opposite root space
then also lies in g0).
    od;
    if (g1-g0=1) then Add(res,s); fi;    #[s,g0,g1,m] is stored if and
only if $dim g_1-dim g_0=1$.
od;
return(res);
end;;

basesg0g1:=function(g)   #g is a Simple Lie algebra.
                         #For each grading selected by the above
function, a basis of g_0 and g_1 is computed.
                         #The output is a list of quadruples (g0,g1,s,m)
where g0 (resp. g1) is a basis of g_0 (resp. g_1), s is the labelling and
m the order of the automorphism.
                         #The name of the variables and the structure of
the program is globally the same as in ListOfTheta, except that g0 and g1
store basis instead of dimension.
                         #The advantage of lauching "ListOfTheta" firstly
is that it deals quickly with many labellings, thus allowing the
remaining of the program to deal with few.
local R,S,n,res,P,hir,Pv,Pvm,B,Car,r,s,i,wp,g0,g1,bases,m,p,lindep;
```

```
R:=RootSystem(g);
n:=Length(SimpleSystem(R));
res:=ListOfTheta(R);
P:=PositiveRootsNF(R);
hir:=ShallowCopy(P[Length(P)]);
lindep:=[1];
Append(lindep,hir);
Pv:=PositiveRootVectors(R);
Pvm:=NegativeRootVectors(R);
B:=Basis(g);
Car:=B{[(Length(B)-(n-1))..Length(B)]}; #Basis of the Cartan subspace of
g
bases:=[];
for s in res do           #s is a labelling giving rise to a grading with
$dim g_1-dim g_0=1$
    g0:=StructuralCopy(Car);
    g1:=[];
    m:=prodscal(lindep,s);
    for i in [1..Length(P)] do
      p:=P[i];
      wp:=(prodscal(s{[2..n+1]},p) mod m);
      if (wp = 1) then Add(g1,Pv[i]); fi;
      if (wp = m-1) then Add(g1,Pvm[i]); fi;
      if (wp = 0) then Add(g0,Pv[i]); Add(g0,Pvm[i]); fi;
    od;
    Add(bases,[g0,g1,s,m]);
od;
return bases;
end;;

Matad:=function(x,baseg) #x is an element of a Lie algebra g, baseg is a
precomputed basis of g.
                         #the function returns the matrix of the
endomorphism ad_x of g.
local y,vy,l;
l:=[];
for y in baseg do
    vy:=x*y;
    Add(l,Coefficients(baseg,vy));
od;
return l;
end;;

kern:=function(g,baseg,M) #g is a Lie algebra, baseg is a precomputed
basis of g, M is the matrix (in baseg) of an endomorphism of g.
                          #The function computes the kernel of the
endomorphism, as a subspace of g.
local N,K,i;
N:=NullspaceMat(M);       #N is a list of list. each sublist consists of
the coordinates of an element of the basis of the (right) kernel.
K:=[];
for i in [1..Length(N)] do
Add(K,LinearCombination(baseg,N[i]));
od;
return(Subspace(g,K));
end;;
```

```
Info_perso:=NewInfoClass("Info_perso");
SetInfoLevel(Info_perso,2);

Stable_rk1_Inner:=function(g)  #g is a simple Lie algebra.
                               #for each labelling of the extended Dynkin
diagram of g, the function returns the pairs (g0,g1) each followed by
additional information where
                               #g0 and g1 are bases of the homogenous
spaces g_0 and g_1 given by the (inner) grading corresponding to the
labelling
                               #If the first additional information is a
1 then the pair correspond to a locally free stable theta-rep of rank 1
                               #If the first additional information is a
0 then no conclusion is made on the pair;
 local bases, g0,g1,Vg1,x,M,Vp,Vt,i,j,Mj,baseg,Vpg1,test;
 bases:=basesg0g1(g);
 baseg:=Basis(g);
 for i in [1..Length(bases)] do
    Info(Info_perso, 1, "#Testing labelling  ",bases[i][3],"  ",
i,"/",Length(bases));
    test:=0;
    while test=0 do;
      g0:=bases[i][1];           #g0 and g1 contains basis of g_0 and g_1
such that dim g_1-dim g_0=1. In particular, the rank of the action is at
least 1 and a generic element in g1 has non-zero semsimple part.
      g1:=bases[i][2];
      Vg1:=Subspace(g,g1);       #Vg1 is the subspace of g generated by
g1
      x:=Random(Vg1);            #x is a (supposedly) generic element of
g1
      M:=Matad(x,baseg);
      Vp:=kern(g,baseg,M);       #Vp is the kernel of ad_x (i.e., the
centralizer of x in g)
      j:=1;
      Mj:=M*M;
      Vt:=kern(g,baseg,Mj);      #Vt is the kernel of (ad_x)^2
      while (Vp<>Vt) do
        j:=j+1;
        Vp:=Vt;
        Mj:=Mj*Mj;
        Vt:=kern(g,baseg,Mj);
      od;                        #in the end of this loop, Vp is the
kernel of (ad_x)^n for assymptotically large n. In other words, Vp is the
centralizer of the semisimple part s of x.
                                 #x is semisimple if and only if j=1
      if Vp<>g then test:=1; fi; #this is the case when the semisimple
part of x is 0. In these cases, x is not as generic as wished.
    od;
    Vpg1:=Intersection(Vp,Vg1);       #stores the centralizer of s in
g_1.
    if (j=1) and (Dimension(Vpg1)=1) then Add(bases[i],"lf stable of rk
1"); #This happens when x is semisimple and its centralizer is 1-
dimensional.
                                                                #This
situation corresponds to locally free stable theta-representations of
rank 1, see Lemma 4.5 of the paper
```

```
                                                                #If x is not
semisimple (j<>1), x has a nilpotent part so the action cannot be stable
of rank 1 as soon as the semisimple part of x is non-zero.
                                                                #If x is
semisimple (j=1) and Dimension(Vpg1)>1, either the rank is greater than 1
or there exists an element x'\neq x in g_1 with semisimple part x and
non-zero nilpotent part.
    else Add(bases[i],"NOT lfs rk1"); fi;
 od;
 return bases;
end;;


ListofOuterThetatoM:=function(a,b,m,d) #a is a type "A", "D" or "E", b is
a rank, m and d are integers
                                        #the function returns a list of
the outer finite order automorphism of a simple Lie algebra of type and
rank a and b, with order at most m, inducing a diagram automotphism of
order d (2 or 3)
    local l,i;
    l:=[];
    for i in [1..m] do
        l:=Concatenation(l,FiniteOrderOuterAutomorphisms(a,b,i,d));
    od;
    return l;
end;;

gooddimOut:=function(l) #l is a list of (possibly outer) finite order
automorphisms of a Lie algebra g
                        #the function returns those which induce a
grading with dim g_1- dim g_0=1.
                        #Due to the slow function "Grading", ListOfTheta
is much more efficient for inner grading.
    local f,res,Gradf;
    res:=[];
    for f in l do
        Gradf:=Grading(f);
        if (Length(Gradf[2])=Length(Gradf[1])+1) then Add(res,f); fi;
    od;
    return res;
end;;



Stable_rk1_Outer:=function(l)    #works globally as the function
"Stable_rk1_Inner" except that l is a list of finite order (possibly
outer) automorphisms instead of just a Lie algebra g
                                 #l will usually be taken as the output
of gooddimOut(ListofOuterThetaoM(...))
local bases,Gradf, g,g0,g1,Vg1,x,M,Vp,Vt,i,j,Mj,baseg,Vpg1,res,test;
res:=[];
for i in [1..Length(l)] do
    g:=Source(l[i]);
    Gradf:=Grading(l[i]);
    baseg:=Basis(g);
    g0:=Gradf[1];
    g1:=Gradf[2];
    Vg1:=Subspace(g,g1);
```

```
    test:=0;
    while test=0 do;
      x:=Random(Vg1);
      M:=Matad(x,baseg);
      Vp:=kern(g,baseg,M);
      j:=1;
      Mj:=M*M;
      Vt:=kern(g,baseg,Mj);
      while (Vp<>Vt) do
        j:=j+1;
        Vp:=Vt;
        Mj:=Mj*Mj;
        Vt:=kern(g,baseg,Mj);
      od;
      if Vp<>g then test:=1; fi;
    od;
    Vpg1:=Intersection(Vp,Vg1);
    if (j=1) and (Dimension(Vpg1)=1) then
Add(res,[l[i],KacDiagram(l[i]).weights,"lf stable of rk 1"]);
    else Add(res,[l[i],KacDiagram(l[i]).weights,"NOT lfs rk1"]); fi;
od;
return res;
end;;



Has_reg:=function(n,f)          #f is a finite order automorphism of a Lie
algebra g, n is a normal sl_2-triple (f,h,e) for the corresponding
grading (e\in g_1, h\in g_0, f\in g_{-1}) .
                                #the function returns a triple (e,p,q)
where p (resp. q) is the dimension of the centralizer of (e,y) in g_0
(resp. in g) for some supposedly generic element y in the centraliser of
e in g_{-1}.
   local g,basegi,m,g0,g1,gm1,gn,gnm1,y,gny,g0ny;
   g:=Source(f);
   basegi:=Grading(f);
   m:=Length(basegi);
   g0:=Subspace(g,basegi[1]);
   g1:=Subspace(g,basegi[2]);
   gm1:=Subspace(g,basegi[m]);
   gn:=LieCentralizer(g,Subspace(g,[n[3]]));
   gnm1:=Intersection(gn,gm1);
   y:=Random(gnm1);
   gny:=LieCentralizer(g,Subspace(g,[y,n[3]]));
   g0ny:=Intersection(gny,g0);
   return([n[3],Dimension(g0ny),Dimension(gny)]);
end;;

Has_regs:=function(f) #f is a finite order automorphism of a simple Lie
algebra g
                      #the function returns a list of triples (e,p,q) as
above where e ranges over a list of representatives of G_0-orbits in g_1.
   local N,v,n;
   N:=NilpotentOrbitsOfThetaRepresentation(f);
   v:=[];
   for n in N do
       Add(v,Has_reg(n,f));
   od;
```

```
   return v;
end;;

Has_regs_check:=function(f) #f is a finite order automorphism of a simple
Lie algebra g
                            #the function returns the maximum dimension
of the centralizer of (e,y) in g_0 for e ranging in a list of
representatives of the G_0-orbits in g_1
                            #and y being a supposedly generic element in
the centralizer of e in g_{-1}.
                            #if the function returns 0, then the theta-
rep satisfies property p
                            #if it does not returns 0, no conclusion can
be made since y might not be as generic as wished.
   local centmax,Hr,x,num;
   Hr:=Has_regs(f);
   centmax:=0;
   for x in Hr do
       centmax:=Maximum(centmax,x[2]);
   od;
   return centmax;
end;;

Non_regs:=function(f) #f is a finite order automorphism of a simple Lie
algebra g
                      #the function returns a list of triples (e,p,q) as
in Has_regs but selects only the cases where p is non-zero.
   local N,v,n,q;
   N:=NilpotentOrbitsOfThetaRepresentation(f);
   v:=[];
   for n in N do
       q:=Has_reg(n,f);
       if (q[2]<>0) then Add(v,q); fi;
   od;
   return v;
end;;

Really_Non_reg:=function(l,f)      #f is a finite order automorphism of a
simple Lie algebra g, l should be the list given by the function
"Non_regs"
                                   #for each nilpotent representative e
appearing in l, the centralizer gxm1 of e in g_{-1} is computed. Then the
simultaneous centralizer g0xygen in g_0 of x and gxm1 is computed
                                   #The function returns the list of
(e,m,p) where (e,p,q) appears in l and m is the dimension of g0xygen.
                                   #If there exists e with a non-zero m
then the corresponding theta-rep fails to satify property (P), see proof
of Lemma 4.11
                                   #(and an equality between m and p
indicates that the supposedly generic centralizer computed in Has_reg was
indeed generic)
   local Gradf, g, g0,g1,gm1,v,x,gx,gxm1, gygen, gxygen, g0xygen,m;
   Gradf:=Grading(f);
   g:=Source(f);
   g0:=Subspace(g, Gradf[1]);
   g1:=Subspace(g, Gradf[2]);
   m:=Length(Gradf);
   gm1:=Subspace(g, Gradf[m]);
```

```
   v:=[];
   for x in l do
       gx:=LieCentralizer(g, Subspace(g,[x[1]]));
       gxm1:=Intersection(gx,gm1);
       gygen:=LieCentralizer(g,gxm1);
       gxygen:=Intersection(gx,gygen);
       g0xygen:=Intersection(g0,gxygen);
       Add(v,[g0xygen,Dimension(g0xygen),x[2]]);
   od;
   return v;
end;;


Inner_Automorphism_By_Kac_diagram:=function(s,type,rank) #An automorphism
of a Lie algebra of type "type" and of rank "rank" over some cyclotomic
field is constructed.
                                                          #This
automorphism has a Kac diagram given by the labelling s (which is assumed
to have length of rank+1).
    local R,n,Pr,d,hir,m,P,Pn,B,w,ImB,f,cc,lindep,g;
    R:=RootSystem(type,rank);
    n:=Length(SimpleSystem(R));
    Pr:=PositiveRootsNF(R);
    d:=Length(Pr);
    hir:=ShallowCopy(Pr[d]);
    lindep:=[1];
    Append(lindep,hir);
    m:=prodscal(lindep,s);                                #m is the order
    g:=SimpleLieAlgebra(type,rank,CF(m));                 #The Lie algebra
g is constructed over a cyclotomic field of order m
    w:=E(m);
    R:=RootSystem(g);
    P:=PositiveRootVectors(R);
    Pn:=NegativeRootVectors(R);
    B:=[Pn[d]];
    Append(B,P{[1..n]});                                  #The system of
simple roots in the affine Dynkin diagram starts with the extended root.
    ImB:=List( [1..n+1], i -> w^s[i]*B[i] );
    f:=AlgebraHomomorphismByImagesNC(g,g,B,ImB);           #Construction
of the automorphism
    SetOrder(f,m);
    cc:= ExtendedCartanMatrix( R );
    SetKacDiagram( f, rec( CM:= cc.ECM, labels:= cc.labels, weights:=
s));  #additional informations attached to f so that the SLA-routines can
work.
    return f;
end;;


ListOfpossLocfreec:=function(R,c)          #R is a root system and c is
an integer
                                           #The function returns the
labellings giving rise to a grading for which dim g_1-dim g_0=c. The
order of the grading is attached with the labelling.
                                           #it works similarly as
ListOfTheta
local S,P,n,l,s,i,g0,g1,k,m,wp,p,hir,res,lindep;
S:=SimpleSystem(R);
```

```
P:=PositiveRootsNF(R);
hir:=ShallowCopy(P[Length(P)]);
lindep:=[1];
Append(lindep,hir);
n:=Length(S);
l:=deuxn(n);
res:=[];
for s in l do
    g1:=0;
    g0:=n;
    m:=prodscal(lindep,s);
    for p in P do
      wp:=(prodscal(s{[2..(n+1)]},p) mod m);
         if wp=1 then g1:=g1+1; fi;
         if wp=m-1 then g1:=g1+1; fi;
      if wp=0 then g0:=g0+2; fi;
    od;
    if (g1-g0=c) then Add(res,[s,m]); fi;
od;
return(res);
end;;

#-------------------------------------------
# B. Computations
#-------------------------------------------

Q:=Rationals

###### TYPE G2 #######

g:=SimpleLieAlgebra("G",2,Q);
Stable_rk1_Inner(g);
#[ [ [ v.13, v.14, v.1, v.7 ], [ v.2, v.3, v.4, v.5, v.12 ], [ 1, 0, 1 ],
3,
#      "lf stable of rk 1" ],
#  [ [ v.13, v.14 ], [ v.1, v.2, v.12 ], [ 1, 1, 1 ], 6, "lf stable of rk
1" ]
# ]

#In other words, there are two labelling of the extended Dynkin diagram
([1,0,1] and [1,1,1], see the article, Table 2 for the numbering of
simple roots) such that the corresponding grading satisfies dim g_1-
dimg_0=1.
#Both correspond to locally free stable theta-representation of rank 1.
#The order of the first grading is 3, with dim g_0=4 and dim g_1=5.
#The last labelling [1,1,1], correspond to G_2^{(1)}(\underline{1}) in
the notation of Lemma 4.9 in the paper. We don't need to make any
computation in such case. Moreover some routine in SLA do not work when
the semisimple part of g_0 is trivial.

f101:=Inner_Automorphism_By_Kac_diagram([1,0,1],"G",2);
#[ v.12, v.1, v.2 ] -> [ (E(3))*v.12, v.1, (E(3))*v.2 ]
Grading(f101);
#[ [ v.1, v.7, v.13, v.14 ], [ v.2, v.3, v.4, v.5, v.12 ],
#  [ v.6, v.8, v.9, v.10, v.11 ] ]

#We recover the corresponding automorphism (and check that the grading on
g is the same)
```

```
Has_regs(f101);
#[ [ v.12, 0, 4 ], [ v.5+v.12, 0, 3 ], [ v.5, 0, 2 ], [ v.4+v.12, 0, 2 ],
[ v.4, 0, 3 ], [ v.3+v.5, 0, 4 ] ]

#In other words, there are 6 non-zero G_0 orbits in g_1.
#For each, one computation of a centralizer in g_0 of a pair (e,y) with e
in the orbit and y in the centralizer in g_{-1} of e, turned out to be
zero
#In particular the corresponding theta-representation satisies property
(P)
#Also, we see that the centralizer in g ranges from 2 (the rank in the
minimal possible centralizer of a commuting pair in g) to 4

#Another way to check it:
Has_regs_check(f101);
#0

###### TYPE F4 #######

g:=SimpleLieAlgebra("F",4,Q);
Stable_rk1_Inner(g);
#[ [ [ v.49, v.50, v.51, v.52, v.3, v.27 ], [ v.1, v.2, v.4, v.5, v.7,
v.10, v.48 ], [ 1, 1, 1, 0, 1 ], 8, "lf stable of rk 1" ],
#  [ [ v.49, v.50, v.51, v.52 ], [ v.1, v.2, v.3, v.4, v.48 ], [ 1, 1, 1,
1, 1 ], 12, "lf stable of rk 1" ] ]

#Again 2 grading giving rise to stable locally free representations of
rank 1. The first one satisfies dim g_0=6, dim g_1=7, and the order is 8.

f11101:=Inner_Automorphism_By_Kac_diagram([1,1,1,0,1],"F",4);
Has_regs_check(f11101);
#0

#So, the theta-respresentaiton satisfies property (P) again in this case.

###### TYPE E6 INNER #######

g:=SimpleLieAlgebra("E",6,Q);
Stable_rk1_Inner(g);
#[ [ [ v.73, v.74, v.75, v.76, v.77, v.78, v.4, v.40 ], [ v.1, v.2, v.3,
v.5, v.6, v.8, v.9, v.10, v.72 ], [ 1, 1, 1, 1, 0, 1, 1 ], 9, "lf stable
of rk 1" ],
#  [ [ v.73, v.74, v.75, v.76, v.77, v.78 ], [ v.1, v.2, v.3, v.4, v.5,
v.6, v.72 ], [ 1, 1, 1, 1, 1, 1, 1 ], 12, "lf stable of rk 1" ] ]

#Again 2 grading giving rise to stable locally free representations of
rank 1.

f1111011:=Inner_Automorphism_By_Kac_diagram([1,1,1,1,0,1,1],"E",6);
Has_regs_check(f1111011);
#1
#So, the theta-respresentation may not satisfy property (P) in this case.

l:=Non_regs(f1111011);
```

```
#[ [ v.1+v.6+v.8+v.9+v.10, 1, 12 ], [ v.1+v.6+v.8+v.9+v.10+v.72, 1, 16 ],
[ v.1+v.8+v.9+v.10, 1, 12 ], [ v.1+v.8+v.9+v.10+v.72, 1, 12 ], [
v.6+v.8+v.9+v.10, 1, 12 ], [ v.6+v.8+v.9+v.10+v.72, 1, 12 ],
#  [ v.8+v.9+v.10, 1, 16 ], [ v.8+v.9+v.10+v.72, 1, 12 ] ]

#In other words there are 8 orbits likely to contradict property (P)

l1:=Really_Non_reg(l,f1111011);
#[ [ <vector space of dimension 1 over CF(9)>, 1, 1 ], [ <vector space of
dimension 1 over CF(9)>, 1, 1 ], [ <vector space of dimension 1 over
CF(9)>, 1, 1 ],
#  [ <vector space of dimension 1 over CF(9)>, 1, 1 ], [ <vector space of
dimension 1 over CF(9)>, 1, 1 ], [ <vector space of dimension 1 over
CF(9)>, 1, 1 ],
#  [ <vector space of dimension 1 over CF(9)>, 1, 1 ], [ <vector space of
dimension 1 over CF(9)>, 1, 1 ] ]

#So it turns that in each of the 8 cases, the 1-dimensional centralizer
found was indeed minimal amond pairs (e,y) with e in each selected orbit.
[l[7][1],GeneratorsOfLeftModule(l1[7][1])];
#[ v.8+v.9+v.10, [ v.4 ] ]

#v.8+v.9+v.10 lies in a nilpotent orbit and the considered 1-dim
centralizer is generated by v.4. This is the example given in Table 1 of
the paper.


###### TYPE E7 #######

g:=SimpleLieAlgebra("E",7,Q);
Stable_rk1_Inner(g);
#[ [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133, v.1, v.64, v.2,
v.65, v.3, v.66, v.5, v.68, v.8, v.71 ], [ v.4, v.6, v.7, v.9, v.10,
v.11, v.12, v.14, v.15, v.16, v.17, v.20, v.21, v.22, v.26,
#          v.124, v.125, v.126 ], [ 1, 0, 0, 0, 1, 0, 1, 1 ], 8, "NOT lfs
rk1" ], [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133, v.1, v.64,
v.4, v.67, v.6, v.69 ],
#       [ v.2, v.3, v.5, v.7, v.8, v.9, v.10, v.11, v.12, v.13, v.14,
v.18, v.125, v.126 ], [ 1, 0, 1, 1, 0, 1, 0, 1 ], 10, "NOT lfs rk1" ],
#  [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133, v.1, v.64, v.4,
v.67 ], [ v.2, v.3, v.5, v.6, v.7, v.8, v.9, v.10, v.11, v.14, v.125,
v.126 ], [ 1, 0, 1, 1, 0, 1, 1, 1 ], 12, "NOT lfs rk1" ],
#  [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133, v.2, v.65, v.3,
v.66, v.5, v.68, v.6, v.69, v.12, v.75 ], [ v.1, v.4, v.7, v.8, v.9,
v.10, v.11, v.13, v.15, v.16, v.17, v.18, v.19, v.22, v.23,
#          v.24, v.29, v.126 ], [ 1, 1, 0, 0, 1, 0, 0, 1 ], 8, "NOT lfs
rk1" ], [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133, v.2, v.65,
v.3, v.66, v.5, v.68 ],
#       [ v.1, v.4, v.6, v.7, v.8, v.9, v.10, v.11, v.12, v.15, v.16,
v.17, v.22, v.126 ], [ 1, 1, 0, 0, 1, 0, 1, 1 ], 10, "NOT lfs rk1" ],
#  [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133, v.4, v.67, v.6,
v.69 ], [ v.1, v.2, v.3, v.5, v.7, v.9, v.10, v.11, v.12, v.13, v.18,
v.126 ], [ 1, 1, 1, 1, 0, 1, 0, 1 ], 12, "NOT lfs rk1" ],
#  [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133, v.4, v.67 ], [
v.1, v.2, v.3, v.5, v.6, v.7, v.9, v.10, v.11, v.126 ], [ 1, 1, 1, 1, 0,
1, 1, 1 ], 14, "lf stable of rk 1" ],
```

```
#  [ [ v.127, v.128, v.129, v.130, v.131, v.132, v.133 ], [ v.1, v.2,
v.3, v.4, v.5, v.6, v.7, v.126 ], [ 1, 1, 1, 1, 1, 1, 1, 1 ], 18, "lf
stable of rk 1" ] ]

#Among the 8 grading satisfying dim g_1-dim g_0=1, only 2 correspond to
stable locally free representations of rank 1. [1,1,1,1,0,1,1,1] of order
14 and [1,1,1,1,1,1,1,1].

f11110111:=Inner_Automorphism_By_Kac_diagram([1,1,1,1,0,1,1,1],"E",7);
Has_regs_check(f11110111);
#1
#So, the theta-respresentation may not satisfy property (P) in this case.

l:=Non_regs(f11110111);
l1:=Really_Non_reg(l,f11110111);
#64 orbits yield a centralizer of a generic pair of dimension 1. This
contradicts property (P) in this case.

[l[15][1],GeneratorsOfLeftModule(l1[15][1])];
[ v.9+v.10+v.11, [ v.4 ] ]
#Hence v.9+v.10+v.11 lies in a nilpotent orbit and the considered 1-dim
centralizer is generated by v.4. This is the example given in Table 1 of
the paper.


###### TYPE E8 #######

g:=SimpleLieAlgebra("E",8,Q);
Stable_rk1_Inner(g);
# [  [  [ 1, 0, 0, 0, 1, 0, 0, 0, 1 ], 9, "NOT lfs rk1" ], [  [ 1, 0, 0,
0, 1, 0, 0, 1, 1 ], 12, "NOT lfs rk1" ],  [  [ 1, 1, 0, 0, 1, 0, 0, 1, 1
], 14, "NOT lfs rk1" ],
# [  [ 1, 1, 0, 0, 1, 0, 1, 0, 1 ], 15, "lf stable of rk 1" ], [  [ 1, 1,
0, 0, 1, 0, 1, 1, 1 ], 18, "NOT lfs rk1" ],  [  [ 1, 1, 1, 1, 0, 1, 0, 1,
0 ], 18,  "NOT lfs rk1" ],
# [  [ 1, 1, 1, 1, 0, 1, 0, 1, 1 ], 20, "lf stable of rk 1" ], [  [ 1, 1,
1, 1, 0, 1, 1, 1, 1 ], 24, "lf stable of rk 1" ],  [  [ 1, 1, 1, 1, 1, 1,
1, 1, 1 ], 30, "lf stable of rk 1" ] ]


#Among the 9 grading satisfying dim g_1-dim g_0=1, only 4 correspond to
stable locally free representations of rank 1:
#[1,1,0,0,1,0,1,0,1] of order 15; [1,1,1,1,0,1,0,1,1] of order 20;
[1,1,1,1,0,1,1,1,1] of order 24 and [1,1,1,1,1,1,1,1,1] of order 30.

f110010101:=Inner_Automorphism_By_Kac_diagram([1,1,0,0,1,0,1,0,1],"E",8);
f111101011:=Inner_Automorphism_By_Kac_diagram([1,1,1,1,0,1,0,1,1],"E",8);
f111101111:=Inner_Automorphism_By_Kac_diagram([1,1,1,1,0,1,1,1,1],"E",8);

Has_regs_check(f110010101);
#2
Has_regs_check(f111101011);
#1
Has_regs_check(f111101111);
#1
#So, the theta-respresentation may not satisfy property (P) in this case.

l:=Non_regs(f110010101);
```

```
l1:=Really_Non_reg(l,f110010101);
#84 orbits yield a centralizer of a generic pair of dimension at least 1.
This contradicts property (P) in this case.
#6 orbits even yield a centralizer of dimension 2.

[l[17][1],GeneratorsOfLeftModule(l1[17][1])];
#[ v.15+v.17+v.18+v.21, [ v.2, v.7 ] ]
#Hence v.15+v.17+v.18+v.21 lies in a nilpotent orbit and the considered
2-dim centralizer is generated by v.2 and v.7. This is the example given
in Table 1 of the paper.



l:=Non_regs(f111101011);
l1:=Really_Non_reg(l,f111101011);
#64 orbits yield a centralizer of a generic pair of dimension 1. This
contradicts property (P) in this case.

[l[55][1],GeneratorsOfLeftModule(l1[55][1])];
#[ v.14+v.20, [ v.6 ] ]
#Hence v.14+v.20 lies in a nilpotent orbit and the considered 1-dim
centralizer is generated by v.6. This is the example given in Table 1 of
the paper.



l:=Non_regs(f111101111);
l1:=Really_Non_reg(l,f111101111);
#32 orbits yield a centralizer of a generic pair of dimension 1. This
contradicts property (P) in this case.

[l[31][1],GeneratorsOfLeftModule(l1[31][1])];
#[ v.10+v.11+v.12, [ v.4 ] ]
#Hence v.10+v.11+v.12 lies in a nilpotent orbit and the considered 1-dim
centralizer is generated by v.4. This is the example given in Table 1 of
the paper.

###### TYPE D4 Outer #######

Aut3_D4:=ListofOuterThetatoM("D",4,12,3);
#This is the list of finite order outer automorphisms in the case D4 with
involution of Dynkin diagram of order 3.
#Given labellings of Kac diagrams in {0,1}, we reach a maximum possible
order of automorphism of 12.
#We consider here ALL outer automorphisms of order at most 12.

gooddimD4:=gooddimOut(Aut3_D4);
#[ [ v.17+(-E(3)^2)*v.18+(-E(3))*v.19, v.23, v.8+v.9+(-1)*v.10 ] -> [ (-
E(12)^7)*v.17+(E(4))*v.18+(E(12)^11)*v.19, (-E(12)^7)*v.23, (-
E(12)^7)*v.8+(-E(12)^7)*v.9+(E(12)^7)*v.10 ] ]
#So we get only one automorphism giving rise to a grading with dim g_1-
dim g_0=1.

KacDiagram(gooddimD4[1]).weights;
#[ 1, 1, 1 ]
#This automorphism arises from the grading with only 1s. Hence we do not
have to consider it (see Lemma 4.9 in the paper).
```

```
###### TYPE E6 Outer #######

Aut2_E6:=ListofOuterThetatoM("E",6,18,2);
#This is the list of finite order outer automorphisms in the case E6 with
involution of Dynkin diagram of order 2.
#Given labellings of Kac diagrams in {0,1}, we reach a maximum possible
order of automorphism of 18.
#We consider here ALL outer automorphisms of order at most 18.

gooddimE6:=gooddimOut(Aut2_E6);
l:=Stable_rk1_Outer(gooddimE6);
#[ [ [ v.49+v.50, v.40, v.70, v.37+v.42, v.32+(-1)*v.33 ] -> [ (-
E(5)^3)*v.49+(-E(5)^3)*v.50, v.40, (-E(5)^3)*v.70, (-E(5)^3)*v.37+(-
E(5)^3)*v.42, v.32+(-1)*v.33 ], [ 1, 0, 1, 1, 0 ], "NOT lfs rk1" ],
#   [ [ v.49+v.50, v.40, v.70, v.37+v.42, v.32+(-1)*v.33 ] -> [ (-
E(12)^7)*v.49+(-E(12)^7)*v.50, (-E(12)^7)*v.40, (-E(12)^7)*v.70, (-
E(12)^7)*v.37+(-E(12)^7)*v.42, v.32+(-1)*v.33 ], [ 1, 1, 1, 1, 0 ],
#       "lf stable of rk 1" ], [ [ v.49+v.50, v.40, v.70, v.37+v.42,
v.32+(-1)*v.33 ] -> [ (-E(9)^5)*v.49+(-E(9)^5)*v.50, (-E(9)^5)*v.40, (-
E(9)^5)*v.70, (-E(9)^5)*v.37+(-E(9)^5)*v.42,
#           (-E(9)^5)*v.32+(E(9)^5)*v.33 ], [ 1, 1, 1, 1, 1 ], "lf stable
of rk 1" ] ]

#So we have a total of 3 automorphisms giving rise to a grading with dim
g_1-dim g_0=1.
#The last 2 correspond to stable locally free theta-representations of
rank 1

f11110:=l[2][1]
#We consider the one with Kac diagram labelled by [1,1,1,1,0]

Has_regs_check(f11110);
#0

#So, the theta-respresentaiton satisfies property (P) in this case.

###### Levi subalgebras #######

R:=RootSystem("E",7);
ListOfpossLocfreec(R,2);
#[ [ [ 0, 0, 0, 0, 0, 1, 0, 0 ], 3 ], [ [ 0, 0, 0, 1, 0, 0, 0, 0 ], 3 ],
[ [ 0, 0, 0, 1, 0, 0, 0, 1 ], 4 ], [ [ 1, 0, 0, 0, 0, 1, 0, 0 ], 4 ] ]

#Any locally free stable theta-representation of rank 2 of type E7 must
appear in the above list
#But the order of the gradings are 3 or 4, hence not a multiple of 9
#As a consequence, in the above cases, no Levi arising from g_1 of rank 1
can coincide with the E6 type with Kac diagram [1,1,1,1,0,1,1]


R:=RootSystem("E",8);
ListOfpossLocfreec(R,2);
#[ [ [ 0, 0, 0, 0, 0, 1, 0, 0, 0 ], 5 ], [ [ 0, 0, 0, 0, 1, 0, 0, 0, 1 ],
8 ], [ [ 1, 0, 0, 0, 1, 0, 0, 1, 0 ], 10 ], [ [ 1, 1, 0, 0, 1, 0, 0, 1, 0
], 12 ] ]
```

```
#Any locally free stable theta-representation of rank 2 of type E8 must
appear in the above list
#But the order of the gradings are not multiple of 9 or 14.
#As a consequence, in the above cases, no Levi arising from g_1 of rank 1
can coincide with the E6 type with Kac diagram [1,1,1,1,0,1,1] or the E7
type [1,1,1,1,0,1,1,1]

ListOfpossLocfreec(R,3);
#[ ]
#There are no locally free stable theta-rep of rank 3 of type E8.
```