MANUAL OF GALCOHOM - A PROGRAM FOR COMPUTING GALOIS COHOMOLOGY

1. Introduction

The purpose of this document is to provide a manual to the programs implementing the algorithms for computing the Galois cohomology of real algebraic groups, that are developed in [BG23].

The program is mainly written in the language of the computer algebra system GAP4, [GAP]. However, for some computations involving number fields it uses the system SageMath, [S09]. If necessary the latter system is called from GAP. The communication from SageMath to GAP is done via a file that is written by SageMath and read by GAP. The input to SageMath is sent directly by GAP, without writing a file.

In order to describe what our programs do, we first give a very short introduction to Galois cohomology over $\mathbb{R}$.

1.1. **Very short introduction to Galois cohomology over $\mathbb{R}$.**

Here we consider the Galois cohomology relative to the Galois group $\mathrm{Gal}(\mathbb{C}/\mathbb{R})$. For an extensive treatment we refer to [Serre97].

Let $\Gamma = \mathrm{Gal}(\mathbb{C}/\mathbb{R}) = \{1, \gamma\}$, with $\gamma^2 = 1$. Let $\mathcal{G}$ be a group on which Γ acts by automorphisms. Let X be a set on which Γ acts. For $g \in \mathcal{G}$, $x \in X$ we denote their images under γ by ${}^\gamma g$, ${}^\gamma x$. We also suppose that $\mathcal{G}$ acts on X and that this action is Γ-equivariant: ${}^\gamma g \cdot {}^\gamma x = {}^\gamma(g \cdot x)$ for all $g \in \mathcal{G}$, $x \in X$.

An element of $\mathcal{G}$ is called a *cocycle* if $g\,{}^\gamma g = 1$. (This is different from the usual definition, but in this particular case it yields the same concept.) Two cocycles g_1, g_2 are equivalent if there is an $h \in \mathcal{G}$ with $g_1 = h^{-1} g_2\, {}^\gamma h$. The first cohomology set $\mathrm{H}^1\mathcal{G}$ is the set of equivalence classes of cocycles. For a cocycle g we denote its equivalence class by $[g]$ which thus lies in $\mathrm{H}^1\mathcal{G}$.

Let $x_0 \in X$ be such that ${}^\gamma x_0 = x_0$. Let $Y = \mathcal{G} \cdot x_0$ be its orbit. Let $Z_{x_0} = \{g \in \mathcal{G} \mid g \cdot x_0 = x_0\}$ be the stabilizer of x_0. We have a natural map $i_* : \mathrm{H}^1 Z_{x_0} \to \mathrm{H}^1\mathcal{G}$ mapping the class of a cocycle z in Z_{x_0} to its class in $\mathrm{H}^1\mathcal{G}$. The kernel of this map consists of all elements that are sent to the trivial class, so

$$\ker i_* = \{[z] \in \mathrm{H}^1\mathcal{G} \mid z \text{ is equivalent to } 1 \text{ in } \mathcal{G}\}.$$

By $\mathcal{G}^\Gamma$, Y^Γ we denote the elements of $\mathcal{G}$, Y respectively that are fixed under γ. We now state a theorem that is central to the classification of the $\mathcal{G}^\Gamma$-orbits on X^Γ. For a proof see [Serre97, Section I.5.4, Corollary 1 of Proposition 36].

Theorem 1. *The orbits of $\mathcal{G}^\Gamma$ on Y^Γ are in bijection with $\ker i_*$. This bijection is given as follows. Let $[z] \in \ker i_*$, then there is a $g \in \mathcal{G}$ with $z = g^{-1}\,{}^\gamma g$ and $[z]$ corresponds to the orbit of $g \cdot x_0$.*

The program implements two main functions. The first takes as input a linear algebraic group $G \subset \mathrm{GL}(n, \mathbb{C})$, defined over $\mathbb{R}$, and outputs the $\mathrm{H}^1 G$. The latter is a GAP object, in which a list of cocycles whose classes form $\mathrm{H}^1 G$ is stored. The second function takes a cocycle $g \in G$ and the $\mathrm{H}^1 G$ and outputs an element $h \in G$ such that $h^{-1} g\, {}^\gamma h$ lies in the computed list of cocycles. We note that this also provides a method for computing $\ker i_*$ as above, and for a $[z] \in \ker i_*$ to find $g \in G$ with $g^{-1}\,{}^\gamma g = z$.

2. Installation (Linux)

The program comes in a gzipped tar file, `galcohom.tar.gz`. Unpacking it will result in a directory being created (`galcohom`) in which all files are placed. It is of course possible to put them into a different directory. However, it is necessary to keep them all in the same directory.

The first step is to edit the files `sqrt.gi` and `sagefct.sage`. In the first file the paths on lines 14 and 21 should be changed. Line 14 should have the command that starts Sage. On line 21 should appear the path to the directory where all the files are. More precisely, if the files are in `/home/gauss/compfiles/galcohom`, then line 21 of the file should be

```
WriteLine( sage_proc, "load(\'/home/gauss/compfiles/galcohom/sagefct.sage\')" );
```

In the file `sagefct.sage` the lines 8 and 45 should contain the path to the directory where all files are. More precisely, if the files are in `/home/gauss/compfiles/galcohom`, then the lines 8 and 45 of `sagefct.sage` should be

```
file = open('/home/gauss/compfiles/galcohom/tmp.g','w')
```

The program is started by issuing from the GAP command line

```
Read("galcohom.g");
```

A good test to see whether the communication with SageMath works is to do

```
gap> a:= Sqroot(2);
Sqroot( 2 )
gap> b:= Sqroot( E(4)*One( SqrtField ) );
(-1/2-1/2*E(4))*Sqroot( 2 )
```

3. The field `SqrtField`

For a few steps in our algorithms it is necessary to work with n-th roots of field elements. For this reason we work with what can be called a "dynamic" field, which gets extended every time we need an n-th root not already lying in the field. For this we have copied the implementation of the field `SqrtField` from the CoReLG package of GAP. This field is an extension of $\mathbb{Q}$ containing the square root of every integer. However, as said above, in our implementation it is a dynamic field. At the start it is equal to $\mathbb{Q}(i)$, and it gets extended every time we need an n-th root that does not already lie in the field. In other words, our field is an ever growing radical tower. For deciding whether an n-th root lies in the field (and finding one if it does), we use SageMath.

The main function for creating new n-th roots is

```
Sqroot( n, a )
```

Here $n \geq 2$ is an integer, and `a` is an element of `SqrtField` (or a rational number). This function returns an element of `SqrtField` whose n-th power is `a`.

Elements of `SqrtField` are represented as sums of products of roots. In the printed version of an element, `Sqroot( a )` means the square root of the element `a`, whereas `pRoot( n, a )` means the `n`-th root of the element `a`. Example:

```
gap> a:= Sqroot( 2 );
Sqroot( 2 )
gap> b:= Sqroot( 3, 1+a );
pRoot( 3,1 + Sqroot( 2 ) )
gap> b^3;
1 + Sqroot( 2 )
gap> b^2;
pRoot( 3,1 + Sqroot( 2 ) )^2
```

Fr conctructing some primitive roots of unity in `SqrtField` we have the function

```
PRU( n )
```

which works for n in $\{1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 20, 24, 40, 60, 120\}$. It returns an element of SqrtField that is a primitive n-th root of unity. Example:

```
gap> z:= PRU(40);
(1/8+1/8*E(4))*Sqroot( 2 ) + (1/8+1/8*E(4))*Sqroot( 10 ) + (1/8+1/8*E(4))*Sqroot( 2 )*\
pRoot( 2,-10 + 2*Sqroot( 5 ) )
gap> z^40;
1
gap> z^10;
E(4)
```

4. Computing Galois cohomology

The main purpose of the program is the compute the first Galois cohomology $\mathrm{H}^1(G, \mathbb{R})$ of a linear algebraic group $G \subset \mathrm{GL}(n, \mathbb{C})$ defined over $\mathbb{R}$. Currently we only deal with the complex conjugation that is given by the complex conjugation of the matrix entries.

We note that the identity component G° of G is completely determined by the Lie algebra $\mathrm{Lie}(G) \subset \mathfrak{gl}(n, \mathbb{C})$. In our program the group G is defined by a list of $n \times n$-matrices that form a basis of the Lie algebra of $\mathrm{Lie}(G)$, along with a list of representatives of the elements of the component group (so if the list is $g_1, \dots, g_s$ then $G = g_1 G^\circ \cup \cdots \cup g_s G^\circ$).

The program has two main functions. The first is

```
FirstGaloisCohomology( mats )
FirstGaloisCohomology( mats, reps )
```

Here mats is a list of matrices forming a basis of the Lie algebra of G, and reps is a list of $n \times n$-matrices with reresentatives of the component group. The output is an object that in its printed form gives some information on the structure of the group, and whose only attribute is Cocycles, which gives a list of representatives of the classes in $\mathrm{H}^1(G, \mathbb{R})$.

During its execution the program prints several messages to the screen. These are probably of interest only to the programmer. In order to suppress these messages it is possible to issue the command

```
printstuff:= false;;
```

Example:

```
gap> L:= SimpleLieAlgebra("A",1,Rationals);;
gap> mats:= List( Basis(L), x -> AdjointMatrix( Basis(L), x ) );;
gap> h1:= FirstGaloisCohomology( mats );
H1connected: start computing subgroup of Weyl group...
H1connected: done subgroup...
H1( A_1, connected; 2 cocycles )
gap> Display( Cocycles( h1 )[1] );
[ [  1,  0,  0 ],
  [  0,  1,  0 ],
  [  0,  0,  1 ] ]
gap> Display( Cocycles( h1 )[2] );
[ [     0,  -1/4,     0 ],
  [    -4,     0,     0 ],
  [     0,     0,    -1 ] ]
```

For an example with a non-connected group we consider the Lie algebra spanned by

$$\begin{pmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0\\ 0&-1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{pmatrix}, \quad \begin{pmatrix} 0&0&1&0&0&0&0&0&0\\ 0&0&0&-1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{pmatrix},$$

along with

$\mathrm{diag}(2\ 2\ -1\ -1\ -1\ -1\ -1\ -1\ 2)$, $\mathrm{diag}(-1\ -1\ 1\ 1\ 0\ 0\ 0\ 0\ 0)$, $\mathrm{diag}(0\ -1\ 1\ 0\ 1\ 0\ -1\ 0\ 0)$.

There are two components. A representative of the non-identity component is

$$Q = \begin{pmatrix} 0&-1&0&0&0&0&0&0&0\\ -1&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&-1 \end{pmatrix}.$$

We now do the following computation.

```
gap> L:= [ [ [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 1, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, -1, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ] ],
>   [ [ 0, 0, 1, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, -1, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ] ],
>   [ [ 2, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 2, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, -1, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, -1, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, -1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, -1, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, -1, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, -1, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 2 ] ],
>   [ [ -1, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, -1, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 1, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 1, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ] ],
>   [ [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, -1, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 1, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, -1, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
>        [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ] ] ];;
gap> Q:= [[0 , -1 , 0 , 0 , 0 , 0 , 0 , 0 , 0],
>                 [-1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0],
>                 [0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0],
>                 [0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0],
>                 [0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0],
>                 [0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0],
>                 [0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0],
>                 [0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0],
>                 [0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1]];;
gap> printstuff:= false;;
gap> FirstGaloisCohomology( L, [ Q^0, Q ] );
H1( A_1.T_2, 2 components; 3 cocycles )
gap> Cocycles( h1 )[3];
[ [ 0, 0, 0, -E(4), 0, 0, 0, 0, 0 ], [ 0, 0, -E(4), 0, 0, 0, 0, 0, 0 ],
  [ 0, -E(4), 0, 0, 0, 0, 0, 0, 0 ], [ -E(4), 0, 0, 0, 0, 0, 0, 0, 0 ],
  [ 0, 0, 0, 0, 0, 0, -1, 0, 0 ], [ 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
  [ 0, 0, 0, 0, -1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 1, 0 ],
  [ 0, 0, 0, 0, 0, 0, 0, 0, -1 ] ]
```

The second main function is

```
NormalizeCocycle( H1, c )
```

Here H1 is a Galois cohomology set of a linear algebraic group G computed with FirstGaloisCohomology, and c is a cocycle in G. This function returns an $h \in G$ such that $h^{-1}c^{\gamma}h$ lies in the list of cocycles of H1. In the following example we compute the first Galois cohomology of $\mathrm{SL}(9, \mathbb{C})$ (which is known to be trivial) and then we find an $h \in \mathrm{SL}(9, \mathbb{C})$ such that $h^{-1}c^{\gamma}h = 1$, where c is the cocycle that we computed in the previous example.

```
gap> K:= SimpleLieAlgebra("A",8,Rationals);;
gap> V:= HighestWeightModule(K,[1,0,0,0,0,0,0,0]);;
gap> mats:= List( Basis(K), x -> MatrixOfAction(Basis(V),x) );;
gap> h1sl9:= FirstGaloisCohomology( mats );
H1( A_8, connected; 1 cocycles )
gap> c:= Cocycles( h1 )[3];;
gap> h:= NormalizeCocycle( h1sl9, c );
[ [ E(4), 0, 0, -1, 0, 0, 0, 0, 0 ],
  [ 0, 5/8+3/8*E(4), -3/8-5/8*E(4), 0, 0, 0, 0, 0, 0 ],
  [ 0, -3/8-5/8*E(4), 5/8+3/8*E(4), 0, 0, 0, 0, 0, 0 ],
  [ -1, 0, 0, E(4), 0, 0, 0, 0, 0 ],
  [ 0, 0, 0, 0, 1/2+1/2*E(4), 0, -1/2+1/2*E(4), 0, 0 ],
  [ 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
  [ 0, 0, 0, 0, -1/2+1/2*E(4), 0, 1/2+1/2*E(4), 0, 0 ],
  [ 0, 0, 0, 0, 0, 0, 0, 1, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, E(4) ] ]
gap> h^-1*c*ComplexConjugate(h);
[ [ 1, 0, 0, 0, 0, 0, 0, 0, 0 ], [ 0, 1, 0, 0, 0, 0, 0, 0, 0 ],
  [ 0, 0, 1, 0, 0, 0, 0, 0, 0 ], [ 0, 0, 0, 1, 0, 0, 0, 0, 0 ],
  [ 0, 0, 0, 0, 1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
  [ 0, 0, 0, 0, 0, 0, 1, 0, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 1, 0 ],
  [ 0, 0, 0, 0, 0, 0, 0, 0, 1 ] ]
```

References

[BG23] M. Borovoi and W. A. de Graaf *Computing Galois cohomology of a real linear algebraic group*, 2023.

[GAP] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.12.2*; 2022, https://www.gap-system.org.

[S09] W. A. Stein et al., *Sage Mathematics Software (Version 9.2)*, The Sage Development Team, 2020, https://www.sagemath.org.

[Serre97] J.-P. Serre. *Galois cohomology*, Springer-Verlag, Berlin, 1997.