AN IMPLEMENTATION OF THE MORPHISMS $\mathrm{SL}_2(\mathbb{F}) \longrightarrow \mathrm{SL}_2(\mathsf{K}) \longrightarrow \mathsf{X}$

ALEXANDRE BOROVIK AND ŞÜKRÜ YALÇINKAYA

Abstract. We briefly explain how to implement the morphisms in our paper [2] and provide some examples.

1. Introduction

In this note, we discuss how to implement our algorithm in GAP constructing morphisms

$$\mathrm{SL}_2(\mathbb{F}) \to \mathrm{SL}_2(\mathsf{K}) \to \mathsf{X}.$$

Here,

- $\mathrm{SL}_2(\mathbb{F})$ is the group of 2×2 matrices of determinant 1 over the field $\mathbb{F}$ where $\mathbb{F}$ is a prime field of odd characteristic;
- X is a black box group encrypting $\mathrm{SL}_2(\mathbb{F})$;
- $\mathrm{SL}_2(\mathsf{K})$ is the group of 2×2 matrices of determinant 1 over a black box field K encrypting $\mathbb{F}$, which is constructed inside X as presented in [1].

Our GAP code is available on

https://github.com/sukru-yalcinkaya/SL2Morphisms.

Our GAP code is designed to work over prime fields. Our primary focus is to implement our algorithm for the black box groups encrypting SL_2 defined over very big fields so, for simplicity in coding, we assume that the size of the underlying field is at least 13.

2. About the construction of the group PGL_2

Let $\mathsf{X} \models \mathrm{SL}_2(\mathbb{F})$ where $\mathbb{F}$ is an unknown finite field of odd characteristic. We assume that a set of generators of X is given. We work inside the group $\bar{\mathsf{X}} \models \mathrm{PSL}_2(\mathbb{F})$ by using the following equivalence of strings in X:

$$\mathsf{x} \equiv \mathsf{y} \iff \mathsf{x} \cdot \mathsf{y}^{-1} \in Z(\mathsf{X}). \quad (1)$$

First, we construct two tori S and R in X where a diagonal automorphism α of $\bar{\mathsf{X}}$ centralizes S and inverts R. To construct a random element in $\bar{\mathsf{X}}$, we consider the elements of the following form:

$$\mathsf{x} = s_1 r_1 \cdots s_k r_k$$

where k is a reasonably sized random natural number and s_i's and r_i's are random elements from S and R, respectively. We can compute the image of the diagonal automorphism α for x:

$$\mathsf{x}^\alpha = s_1 r_1^{-1} \cdots s_k r_k^{-1}.$$

To construct group $\mathrm{PGL}_2(\mathbb{F})$, we first consider the diagonal embedding:

$$\tilde{\mathsf{X}} = (\mathsf{X}, \mathsf{X}) = \{(\mathsf{x}, \mathsf{x}^\alpha) \mid \mathsf{x} \in \mathsf{X}\}.$$

Clearly, the diagonal automorphism α interchanges the components of $\tilde{\mathsf{X}}$, that is, $(\mathsf{x}, \mathsf{x}^\alpha)^\alpha = (\mathsf{x}^\alpha, \mathsf{x})$. Finally, $\mathsf{Y} = \tilde{\mathsf{X}} \rtimes \langle \alpha \rangle \models \mathrm{PGL}_2(\mathbb{F})$ where Equation 1 is used for checking whether a group element represents the identity element or not.

In our GAP code, to construct the black box group encrypting PGL_2 by using the black box group X, we use the following function for the setup:

$$\mathsf{SetUpForPGL2}(\text{“}S\text{”}, \text{“}Eo\text{”})$$

where S is a generating set for X and Eo is the odd part of its exponent. This function outputs

- The list “S”.
- A list of elements from a centralizer of an involution, say i, inverted by a diagonal automorphism.
- A list of semisimple elements generating a torus centralized by the same diagonal automorphism.
- The involution i.

We consider the elements of Y as follows:

$$(\mathsf{x}, \mathsf{x}^\alpha, 0) \text{ or } (\mathsf{x}, \mathsf{x}^\alpha, 1)$$

where the elements of the form $(\mathsf{x}, \mathsf{x}^\alpha, 0)$ belong to the coset $\tilde{\mathsf{X}}$ and the elements of the form $(\mathsf{x}, \mathsf{x}^\alpha, 1)$ belong to the coset $\tilde{\mathsf{X}}\alpha$. The group multiplication in Y is the usual multiplication in semidirect product of two groups and it is as follows:

- $(\mathsf{x}, \mathsf{x}^\alpha, 0) \circ (\mathsf{y}, \mathsf{y}^\alpha, 0) = (\mathsf{xy}, \mathsf{x}^\alpha \mathsf{y}^\alpha, 0)$.
- $(\mathsf{x}, \mathsf{x}^\alpha, 0) \circ (\mathsf{y}, \mathsf{y}^\alpha, 1) = (\mathsf{xy}, \mathsf{x}^\alpha \mathsf{y}^\alpha, 1)$.
- $(\mathsf{x}, \mathsf{x}^\alpha, 1) \circ (\mathsf{y}, \mathsf{y}^\alpha, 0) = (\mathsf{x}\mathsf{y}^\alpha, \mathsf{x}^\alpha \mathsf{y}, 1)$.
- $(\mathsf{x}, \mathsf{x}^\alpha, 1) \circ (\mathsf{y}, \mathsf{y}^\alpha, 1) = (\mathsf{x}\mathsf{y}^\alpha, \mathsf{x}^\alpha \mathsf{y}, 0)$.

3. Before running the main algorithm

We need to perform two preprocessing steps before we run our main algorithm.

The first one is ToolBoxSL2. The function returns all the necessary tools to work within the black box group X. It can be run as

$$\mathsf{ToolBoxSL2}(\text{“}S\text{”}, \text{“}E\text{”})$$

where S is a generating set for X and E is an any exponent. Its output is the following list.

- The output of the function SetUpForPGL2.
- A list of elements considered to be a generating set for the semidirect product isomorphic to PGL_2.
- Three commuting involutions forming the vertices of the projective plane.
- An element of order 3 permuting the three commuting involutions.
- A unity element on the corresponding coordinate axes.
- A generating set for the centralizer of a fixed involution (item 3) which is a vertex determining the projective plane — the black box field is constructed on the corresponding axis.
- The projective point (an involution) which serves as 0 in the black box field.
- The projective point (an involution) corresponding to the homogenous coordinate (1,1,1).
- Odd part of the exponent of the group generated by the list S.
- Binary representation of the odd part of the exponent.
- An element of order 4 whose square is the fixed involution.
- Identity of the group generated by the list S.

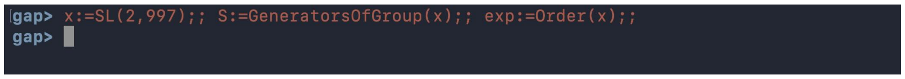
Secondly, we construct the change of basis matrix which is used to transform the elements of $\mathrm{SO}_3^\sharp$ to the elements of $\mathrm{SO}_3^\flat$, see [2] for the definitions of $\mathrm{SO}_3^\sharp$ and $\mathrm{SO}_3^\flat$. This function is

$$\mathsf{SharpVsFlat}(\text{“}TB\text{”})$$

where “TB” is an output of the function ToolBoxSL2.

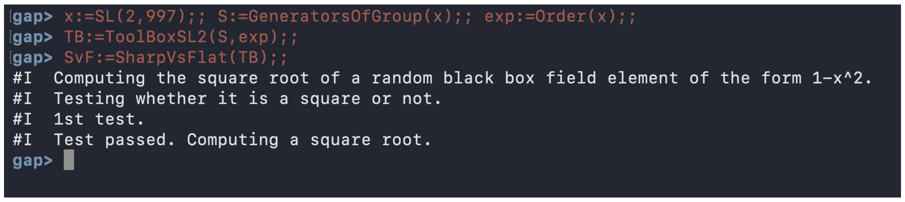
4. How to run our GAP code

We show over an example how our code should be run in GAP. Consider a black box group $\mathsf{X} \vDash \mathrm{SL}_2(997)$. Since the groups $\mathrm{SL}_2(997)$ exist in GAP Library, we can take its generators and its exponent as follows.



```
gap> x:=SL(2,997);; S:=GeneratorsOfGroup(x);; exp:=Order(x);;
gap> 
```

We treat the group X as a black box group. We construct our tool box for X and the change of basis matrix from $\mathrm{SO}_3^\sharp$ to $\mathrm{SO}_3^\flat$ as follows. The construction of the change of basis matrix involves a computation of a square root of a black box field element and its running time may get lengthy due to some unlucky choices of random elements. Therefore, we provide some information on the screen as the algorithm runs.



```
gap> x:=SL(2,997);; S:=GeneratorsOfGroup(x);; exp:=Order(x);;
gap> TB:=ToolBoxSL2(S,exp);;
gap> SvF:=SharpVsFlat(TB);;
#I  Computing the square root of a random black box field element of the form 1-x^2.
#I  Testing whether it is a square or not.
#I  1st test.
#I  Test passed. Computing a square root.
gap> 
```

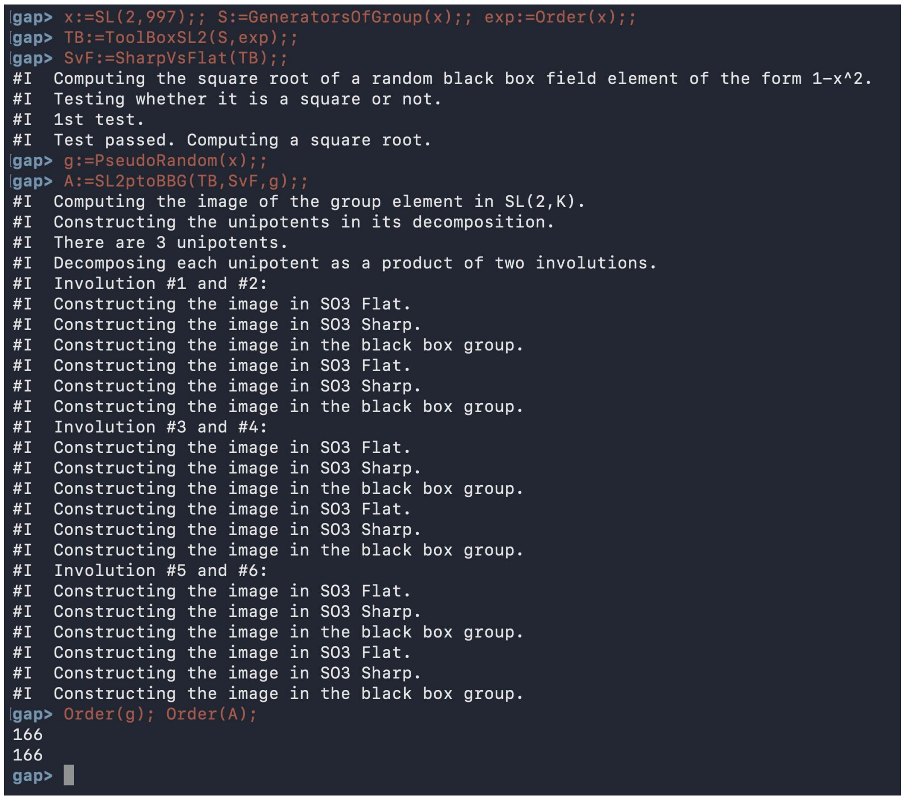
Now, we show an example of how an image of random element is found and some simple checks showing that the corresponding images have same orders.

The input group element is an ordinary element of SL_2 in GAP format. Then, we first represent the 2×2 matrix in GAP as a 2×2 matrix over our black box field by finding the images of the entries of the input matrix in the black box field. Then, we decompose the 2×2 matrix over the black box field K as a product of unipotent elements as explained in our paper [2]. After that, we write each unipotent as a product of two involutions and find the images of these involutions by constructing their images in each of the groups below.

$$\mathrm{SL}_2(\mathsf{K}) \to \mathrm{SO}_3^\flat(\mathsf{K}) \to \mathrm{SO}_3^\sharp(\mathsf{K}) \to \mathsf{X}$$

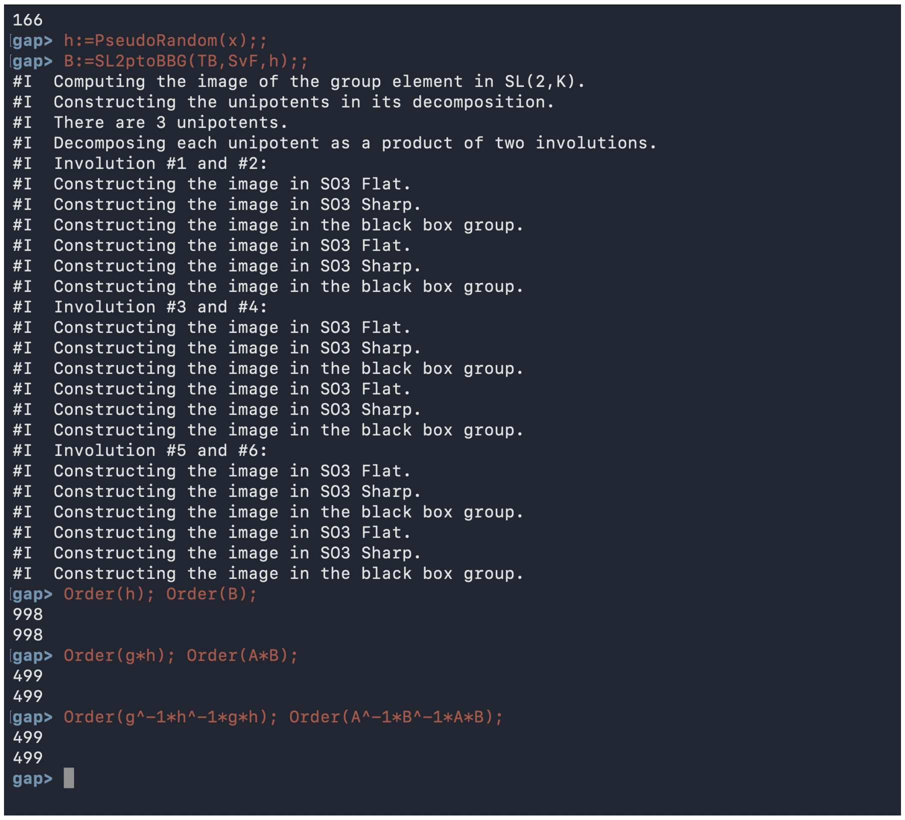
As the algorithm runs, we provide the relevant information on the screen. At the end, we take the corresponding component as an output.

Notice that the orders of the corresponding elements are same.



```
gap> x:=SL(2,997);; S:=GeneratorsOfGroup(x);; exp:=Order(x);;
gap> TB:=ToolBoxSL2(S,exp);;
gap> SvF:=SharpVsFlat(TB);;
#I  Computing the square root of a random black box field element of the form 1-x^2.
#I  Testing whether it is a square or not.
#I  1st test.
#I  Test passed. Computing a square root.
gap> g:=PseudoRandom(x);;
gap> A:=SL2ptoBBG(TB,SvF,g);;
#I  Computing the image of the group element in SL(2,K).
#I  Constructing the unipotents in its decomposition.
#I  There are 3 unipotents.
#I  Decomposing each unipotent as a product of two involutions.
#I  Involution #1 and #2:
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Involution #3 and #4:
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Involution #5 and #6:
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
gap> Order(g); Order(A);
166
166
gap> 
```

For the correctness of the algorithm, it is clear that checking the orders of some random elements and their images is not enough. A more rigorous approach is to examine the Chevalley Commutator Relations for various Chevalley generators. As the constructions of the images of certain Chevalley generators are identical to the example above and the corresponding relations are straightforward to verify, we skip presenting such an example as it would take up too much space in this note. Instead, we present another image of a random element and some checks of the orders of the corresponding elements.



```
166
gap> h:=PseudoRandom(x);;
gap> B:=SL2ptoBBG(TB,SvF,h);;
#I  Computing the image of the group element in SL(2,K).
#I  Constructing the unipotents in its decomposition.
#I  There are 3 unipotents.
#I  Decomposing each unipotent as a product of two involutions.
#I  Involution #1 and #2:
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Involution #3 and #4:
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Involution #5 and #6:
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
#I  Constructing the image in SO3 Flat.
#I  Constructing the image in SO3 Sharp.
#I  Constructing the image in the black box group.
gap> Order(h); Order(B);
998
998
gap> Order(g*h); Order(A*B);
499
499
gap> Order(g^-1*h^-1*g*h); Order(A^-1*B^-1*A*B);
499
499
gap> 
```

In our paper [2], we presented an algorithm constructing the inverse of this morphism. An implementation of this inverse morphism will be published later in the same GitHub repository.

References

[1] Alexandre Borovik and Şükrü Yalçınkaya, *Adjoint representations of black box groups* $\mathrm{PSL}_2(\mathbb{F}_q)$, J. Algebra **506** (2018), 540–591.

[2] Alexandre Borovik and Şükrü Yalçınkaya, *Natural representations of black box groups encrypting* $\mathrm{SL}_2(\mathbb{F})$, arxiv.org/abs/2001.10292.