Oberwolfach
Preprints

OWP 2017 - 22

ALLA DETINKO, DANE FLANNERY AND ALEXANDER HULPKE

GAP Functionality for Zariski Dense Groups

Mathematisches Forschungsinstitut Oberwolfach gGmbH
Oberwolfach Preprints (OWP) ISSN 1864-7596

Imprint:

Mathematisches Forschungsinstitut Oberwolfach gGmbH (MFO)
Schwarzwaldstrasse 9-11
77709 Oberwolfach-Walke
Germany

Tel +49 7834 979 50
Fax +49 7834 979 55
Email admin@mfo.de
URL www.mfo.de

The Oberwolfach Preprints (OWP, ISSN 1864-7596) are published by the MFO.
Copyright of the content is held by the authors.

DOI 10.14760/OWP-2017-22

GAP FUNCTIONALITY FOR ZARISKI DENSE GROUPS

ALLA DETINKO, DANE FLANNERY, AND ALEXANDER HULPKE

In this document we describe the functionality of GAP [4] routines for Zariski dense or arithmetic groups that are developed in [1, 2, 3].

The research underlying the software was supported through the programme "Research in Pairs", at the Mathematisches Forschungsinstitut Oberwolfach, and part of the software was written during a stay in June 2017. The hospitality we received has been greatly appreciated.

Our research was also supported by a Marie Skłodowska-Curie Individual Fellowship grant under Horizon 2020 (EU Framework Programme for Research and Innovation), and a Simons Foundation Collaboration Grant Nr. 244502. All support is acknowledged with gratitude.

1. Background

We consider H to be an integral matrix group, given by generator matrices which lie in the special linear group

$$\mathrm{SL}(n,\mathbb{Z}) = \left\{A \in \mathbb{Z}^{n\times n} \mid \det A = 1\right\}$$

for $n \geq 2$; respectively the symplectic group

$$\mathrm{Sp}(n,\mathbb{Z}) = \left\{A \in \mathrm{SL}(n,\mathbb{Z}) \mid A^T \cdot J \cdot A = J\right\}$$

with

$$J = \begin{pmatrix} 0 & 1_{n/2} \\ -1_{n/2} & 0 \end{pmatrix}$$

for $n \geq 2$ even. (It will be required to specify which case is to be considered.) We shall write $\mathrm{SX}(n,\mathbb{Z})$ from now on, denoting either choice.

If $m > 1$, we can consider the matrices as being written over $\mathbb{Z}/m\mathbb{Z}$ and obtain a subgroup of $\mathrm{SX}(n,\mathbb{Z}/m\mathbb{Z})$. We denote this reduction homomorphism by φ_m.

By [7], a subgroup $H \leq \mathrm{SX}(n,\mathbb{Z})$ is Zariski dense in $\mathrm{SX}(n,\mathbb{R})$, if and only if the image group $\varphi_p(H)$ is equal to $\mathrm{SX}(n,p)$ for almost all primes p. We denote by $\Pi(H)$ the finite set of primes p for which $\varphi_p(H)$ is a proper subgroup of $\mathrm{SX}(n,p)$.

Furthermore, H has finite index in $\mathrm{SX}(n,\mathbb{Z})$ only if H is Zariski dense. In this case, H is called *arithmetic*. If also $n \geq 3$, there will be a smallest integer M, called the *level* of H, such that the kernel of the

Date: Version 1.4, August 2017.

reduction modulo M homomorphism φ_M (as a map on $\mathrm{SX}(n,\mathbb{Z})$) lies in H. We denote by $\pi(M)$ the set of prime divisors of M.

If H is Zariski dense but not of finite index, there will be a unique minimal arithmetic group $\bar{H}$ with $H < \bar{H} \leq \mathrm{SX}(n,\mathbb{Z})$. We call this group $\bar{H}$ the *arithmetic closure* of H. If H is arithmetic, we simply set $\bar{H} = H$.

This naturally poses the following algorithmic questions for a subgroup $H \leq \mathrm{SX}(n,\mathbb{Z})$, given by generating matrices:

- Test whether H is Zariski dense.
- For a dense subgroup H, determine the set $\Pi(H)$ of relevant primes.
- If H is arithmetic and $n \geq 3$, determine its level and index in $\mathrm{SX}(n,\mathbb{Z})$.
- If H is dense and $n \geq 3$, determine its arithmetic closure $\bar{H}$.

Deterministic algorithms for these questions are described in [1, 2, 3]; this documents describes implementations of these algorithms.

A fundamental result of [2] is that $\pi(M) = \Pi(H)$ with possible exceptions for $n \leq 4$. In that case we will have that $\pi(M) = \Pi(H) \cup \{2\}$ and $\varphi_4(H)$ is a proper subgroup of $\mathrm{SX}(n,\mathbb{Z}/4\mathbb{Z})$.

The actual code consists of a file that can be read in by GAP. It is available at the web address `http://www.math.colostate.edu/~hulpke/arithmetic.g`

It requires GAP in release at least 4.8; the `matgrp` package, as well as the packages (`recog`, `genss`, `orb`, `io`) on which it relies.

Place the file in a directory readable by GAP and read it as

```
gap> Read("arithmetic.g");
```

You should get a printout that the routines were loaded, respectively an error message that a required package could not be loaded.

In the description of the functionality below, the default input to most of the routines will be a group H, generated by a finite number of integral matrices. Functions also take an optional argument *kind*, which currently may take the values `SL` or `1` (either is indicating that the group is to be considered as a subgroup of $\mathrm{SL}(n,\mathbb{Z})$), respectively `SP` or `2` indicating it as a subgroup of Sp. If no kind is indicated, it will be assumed that the kind is SL.

The routines perform minimal validity checks for the input. They will not check for membership of group elements or correctness of the specified kind. If the kind is $\mathrm{Sp}(n,\mathbb{Z})$, the form

$$\begin{pmatrix} 0 & 1_{n/2} \\ -1_{n/2} & 0 \end{pmatrix}$$

is assumed. If the matrices preserve only a different form, incorrect results might arise. Similarly, if the group given is not generated by integer matrices, the behavior of the routines is undefined.

2. Basic Routines

This section describes routines that can be used to generate $\mathrm{SL}(n, \mathbb{Z})$ or $\mathrm{Sp}(n, \mathbb{Z})$ in matrix form, or as a finitely presented group, as well as for computing with congruence images of matrix groups.

▶ `SLNZFP(n)`
constructs an isomorphism from a finitely presented version of $\mathrm{SL}(n, \mathbb{Z})$ to the natural matrix version. Note that factorization of matrices into generators is not yet possible.

```
gap> hom:=SLNZFP(3);
[ t12, t13, t21, t23, t31, t32 ] ->
[ [ [ 1, 1, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],
  [ [ 1, 0, 1 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],
  [ [ 1, 0, 0 ], [ 1, 1, 0 ], [ 0, 0, 1 ] ],
  [ [ 1, 0, 0 ], [ 0, 1, 1 ], [ 0, 0, 1 ] ],
  [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 1, 0, 1 ] ],
  [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 1, 1 ] ] ]
```

▶ `HNFWord(hom,mat)`
For a homomorphism created through `SLNZFP` and a matrix, this routine returns a word in the finitely presented group that maps to the desired image matrix. (This will only work for SL.)

```
gap> HNFWord(hom,[ [ 26, -3, 9 ], [ 0, -1, 6 ], [ -3, 0, 1 ] ]);
t23^-1*t12^-9*(t21^-1*t12^2)^2*t21^3*t31^-3*t32^-1*t23^-34
  *(t32^-1*t23^2)^2*t32^-1*t23^-5*t12^12*t13^-72
gap> Image(hom,last);
[ [ 26, -3, 9 ], [ 0, -1, 6 ], [ -3, 0, 1 ] ]
```

▶ `SPNZFP(n)`
constructs an isomorphism from a finitely presented version of $\mathrm{Sp}(n, \mathbb{Z})$ to the natural matrix version. Note that factorization of matrices into generators is not yet possible.

```
gap> hom:=SPNZFP(4);
[ T12, T21, U12, U21, V12, V21 ] ->
[ [ [ 1, 1, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, -1, 1 ] ],
  [ [ 1, 0, 0, 0 ], [ 1, 1, 0, 0 ], [ 0, 0, 1, -1 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 1 ], [ 0, 1, 1, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 1 ], [ 0, 1, 1, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 1, 1, 0 ], [ 1, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 1, 1, 0 ], [ 1, 0, 0, 1 ] ] ]
```

▶ `ModularImageMatrixGroup(H,m)`
returns the matrix group over $\mathbb{Z}/m\mathbb{Z}$ obtained by reducing the coefficients of the matrices in the group H modulo m. If H is already defined over a residue class ring $\mathbb{Z}/k\mathbb{Z}$ and $m \mid k$ this is the obvious reduction; if $m \not| k$ the result is undefined.

```
gap> h:=Group([ [ [ 26, -3, 9 ], [ 0, -1, 6 ], [ -3, 0, 1 ] ],
>   [ [ -1, 0, 0 ], [ -9, 1, -3 ], [ 3, 0, -1 ] ],
>   [ [ 0, 0, 1 ], [ 1, 0, 9 ], [ 0, 1, 0 ] ] ]);
<matrix group with 3 generators>
gap> a:=ModularImageMatrixGroup(h,9*17);
<matrix group with 3 generators>
gap> a.1;
[ [ ZmodnZObj( 26, 153 ), ZmodnZObj( 150, 153 ), ZmodnZObj( 9, 153 ) ],
  [ ZmodnZObj( 0, 153 ), ZmodnZObj( 152, 153 ), ZmodnZObj( 6, 153 ) ],
    [ ZmodnZObj( 150, 153 ), ZmodnZObj( 0, 153 ), ZmodnZObj( 1, 153 ) ] ]
gap> Display(a.1);
matrix over Integers mod 153:
[ [   26,  150,    9 ],
  [    0,  152,    6 ],
gap> b:=ModularImageMatrixGroup(a,3);
<matrix group with 3 generators>
gap> b.1;
[ [ Z(3), 0*Z(3), 0*Z(3) ], [ 0*Z(3), Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3)^0 ] ]
  [  150,    0,    1 ] ]
gap> Display(b.1);
 2 . .
 . 2 .
 . . 1
```

The general functionality for matrix groups over residue class rings is still limited to some extent. When working further with such a group, the first call should be to the (somewhat technical) function `FittingFreeLiftSetup`. This will establish, for example, the order of the group, and set up basic data structures for other calculations.

```
gap> FittingFreeLiftSetup(a);;
gap> Size(a);
2251866396672
```

It is also possible to convert the group to an (isomorphic) permutation representation, using `IsomorphismPermGroup`, but in general these groups will be of inconveniently large degree.

3. Density Testing

This section describes a variety of routines that can be used to test Zariski density of a matrix group.

▶ `IsTransvection(t)`
tests whether t is a transvection; that is, the rank of $t - 1_n$ is 1 and $(t - 1_n)^2 = 0$.

```
gap> t:=[ [ 1, 0, 0 ], [ -9, 1, 0 ], [ -6, 0, 1 ] ];;
gap> IsTransvection(t);
true
```

In all examples below we will assume that

```
gap> h:=Group([ [ [ 26, -3, 9 ], [ 0, -1, 6 ], [ -3, 0, 1 ] ],
>   [ [ -1, 0, 0 ], [ -9, 1, -3 ], [ 3, 0, -1 ] ],
>   [ [ 0, 0, 1 ], [ 1, 0, 9 ], [ 0, 1, 0 ] ] ]);
```

This is a dense subgroup of $\mathrm{SL}(3, \mathbb{Z})$ and contains the transvection t from the previous example. If we take instead the subgroup generated by the first generator of h together with t, we obtain a subgroup that is no longer dense.

▶ `IsDenseDFH(H[,kind],t)`
implements the density test of [2]. Here t must be a transvection in H, given as a matrix. (Membership of t in H is assumed and not tested.)

```
gap> IsDenseDFH(h,SL,t);
true
gap> IsDenseDFH(Group(h.1,t),SL,t);
false
```

▶ `IsDenseIR1(H[,kind])`
implements the Monte-Carlo algorithm [8, Algorithm 1] to test density. It returns `true` if the group is dense, and `false` if it is not dense or the test failed.

This routine is by far the fastest of the density tests; however, if it returns `false`, this might indicate either that the group is not dense, or that a random search failed to find a suitable element.

```
gap> IsDenseIR1(h,SL);
true
gap> IsDenseIR1(Group(h.1,t));
false
```

▶ `IsDenseIR2(H[,kind])`
implements the deterministic algorithm [8, p.23] to test density by verifying absolute irreducibility of the adjoint representation. It is the

slowest of the density test routines. Due to the current implementation, it requires H to consist of integral matrices (though the algorithm itself would also work over the rationals or number fields).

```
gap> IsDenseIR2(h,SL);
true
gap> IsDenseIR2(Group(h.1,t));
false
```

4. Primes and Level

Next, we describe functions that determine the set of primes $\pi(M)$ and $\Pi(H)$ associated to H, as well as the level M of the arithmetic closure of H. All functions in this section assume that H is dense and might not terminate, or produce meaningless results, if it is not.

▶ `PrimesNonSurjective(H[,kind])`
takes an integral matrix group H and returns the set $\pi(M)$ of primes that divide the level M of the arithmetic closure $\bar{H}$ of H.

At the moment this function is only implemented for SL in prime degree; future releases will cover further cases.

(We continue with the same group H in the examples.)

```
gap> PrimesNonSurjective(h,SL);
[ 3, 73 ]
```

We can translate between $\Pi(H)$ and $\pi(M)$ (which is only required in dimension ≤ 4) by considering the images modulo 2 and modulo 4. This is illustrated in the following example:

```
gap> gp:=Group([[[0,0,1],[1,0,0],[0,1,0]],[[1,2,4],[0,-1,-1],[0,1,0]]]);;
gap> PrimesNonSurjective(gp);
[ 2, 3, 5, 19 ]
gap> gp2:=ModularImageMatrixGroup(gp,2);
Group([ <an immutable 3x3 matrix over GF2>,
  <an immutable 3x3 matrix over GF2> ])
gap> FittingFreeLiftSetup(gp2);;Size(gp2);
168
gap> gp4:=ModularImageMatrixGroup(gp,4);;
gap> FittingFreeLiftSetup(gp4);;Size(gp4);
168
```

We note that the image $\varphi_2(H)$ modulo 2 has full order $|\mathrm{SL}(3,2)|$; thus $2 \notin \Pi(H)$. On the other hand, the image $\varphi_4(H)$ modulo 4 is a proper subgroup of $\mathrm{SL}(3,\mathbb{Z}/4\mathbb{Z})$, since $|\varphi_4(H)| < |\mathrm{SL}(3,\mathbb{Z}/4\mathbb{Z})|$. Thus $2 \in \pi(M)$.

▶ `PrimesForDense(H,t[,kind])`
takes an integral matrix group H and returns $\Pi(H)$. The element `t` must be a transvection in H.

```
gap> PrimesForDense(h,t);
[ 3, 73 ]
```

▶ `MaxPCSPrimes(H,primes,[,kind])`
takes an integral matrix group H of dimension ≥ 3 and the set $\pi(M)$ of primes dividing the level M of $\bar{H}$ and returns a list of length 2, containing the level and index of (the arithmetic closure of) H.

```
gap> MaxPCSPrimes(h,[3,73]);
[ 1971, 33180341688 ]
```

If only a subset of $\pi(M)$ is given, only these primes are considered, leading to a divisor of level and index.

```
gap> MaxPCSPrimes(gp,[2,3,5,19]);
[ 5700, 242646091084800000 ]
gap> MaxPCSPrimes(gp,[3,5,19]);
[ 1425, 947836293300000 ]
```

If the option `quotient` is given, the function returns a list of length three with the extra third entry being the congruence quotient modulo the level M.

```
gap> q:=MaxPCSPrimes(h,[3,73]:quotient);
[ 1971, 33180341688, <matrix group of size 5874712004650752 with
    3 generators> ]
gap> Size(SL(3,Integers mod 1971))/Size(q[3]);
33180341688
```

5. Further Examples

▶ `BetaT(t)`
returns the group $\beta_T(\Gamma)$ of [6].

```
gap> h:=BetaT(1);
gap> PrimesNonSurjective(h);
[5]
gap> MaxPCSPrimes(h,[5],SL);
[ 5, 31 ]
```

As the index is small we can verify arithmeticity by coset enumeration:

```
gap> hom:=SLNZFP(3);;
gap> w:=List(GeneratorsOfGroup(h),x->HNFWord(hom,x));
[ t13^-1*t31*(t12*t21^-1*t12)^2*t12*t32*t23^-2,
  t21*t31^-1*(t21^-1*t12^2)^2*t23^-1*(t32^-1*t23^2)^2,
  t12^-1*t21*t23^-1*t32*t23^-1*t13^2 ]
gap> u:=Subgroup(Source(hom),w);
Group([ t13^-1*t31*(t12*t21^-1*t12)^2*t12*t32*t23^-2,
  t21*t31^-1*(t21^-1*t12^2)^2*t23^-1*(t32^-1*t23^2)^2,
  t12^-1*t21*t23^-1*t32*t23^-1*t13^2 ])
```

```
gap> Index(Source(hom),u);
31
```

Such a calculation will be increasingly difficult as the index grows; for example determining the index 3670016 of $\beta_{-2}(\Gamma)$ takes a couple of minutes and any attempt for $\beta_7(\Gamma)$ will be hopeless because of the large index.

▶ `RhoK(k)`
returns the group $\rho_k(\Gamma)$ of [6].

▶ `HofmannStraatenExample(d,k)`
returns the group $G(d,k)$ of [5].

```
gap> h:=HofmannStraatenExample(12,7);;IsTransvection(h.2);
true
gap> PrimesForDense(h,h.2,SP);
[ 2, 3 ]
gap> MaxPCSPrimes(h,[2,3],SP);
[ 288, 2388787200 ]
```

References

1. A. S. Detinko, D. L. Flannery, and A. Hulpke, *Algorithms for arithmetic groups with the congruence subgroup property*, J. Algebra **421** (2015), 234–259.
2. A. S. Detinko, D. L. Flannery, and A. Hulpke, *Zariski density and computing in arithmetic groups*, Math. Comp., `https://doi.org/10.1090/mcom/3236`.
3. A. S. Detinko, D. L. Flannery, and A. Hulpke, *Experimenting with Zariski dense subgroups*, in preparation.
4. The GAP Group, GAP – Groups, Algorithms, and Programming, `http://www.gap-system.org`
5. J. Hofmann and D. van Straten, *Some monodromy groups of finite index in* $\mathrm{Sp}_4(\mathbb{Z})$, `http://arxiv.org/abs/1312.3063v1`.
6. D. D. Long, A. W. Reid, *Small subgroups of* $\mathrm{SL}(3,\mathbb{Z})$, Exper. Math. **20** (2011), no. 4, 412–425.
7. C. Matthews, L. N. Vaserstein, and B. Weisfeiler, *Congruence properties of Zariski-dense subgroups. I*, Proc. London Math. Soc. (3) **48** (1984), no. 3, 514–532.
8. I. Rivin, *Large Galois groups with applications to Zariski density*, `http://arxiv.org/abs/1312.3009v4`

School of Computer Science, University of St Andrews, UK

School of Mathematics, Statistics and Applied Mathematics, National University of Ireland, Galway, Ireland

Department of Mathematics, Colorado State University, Fort Collins, CO 80523-1874, USA