

# The ModularGroup Package

Finite-index subgroups of  $(P)SL_2(\mathbb{Z})$

Version 1.0.0

08 November 2018

**Luca L. Junk**

**Luca L. Junk**

Email: [junk@math.uni-sb.de](mailto:junk@math.uni-sb.de)

Homepage: <http://www.math.uni-sb.de/ag/weitze/>

Address: AG Weitze-Schmithüsen

FR 6.1 Mathematik

Universität des Saarlandes

D-66041 Saarbrücken

## **Copyright**

© 2018 by Luca L. Junk

## **Acknowledgements**

Supported by Project I.8 of SFB-TRR 195 'Symbolic Tools in Mathematics and their Application' of the German Research Foundation (DFG).

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	General aims of the ModularGroup package . . . . .	4
1.2	Technicalities . . . . .	4
<b>2</b>	<b>Subgroups of <math>SL_2(\mathbb{Z})</math></b>	<b>5</b>
2.1	Construction of modular subgroups . . . . .	5
2.2	Computing with modular subgroups . . . . .	7
2.3	Miscellaneous . . . . .	13
<b>3</b>	<b>Subgroups of <math>PSL_2(\mathbb{Z})</math></b>	<b>16</b>
3.1	Construction of projective modular subgroups . . . . .	16
3.2	Computing with projective modular subgroups . . . . .	17
	<b>References</b>	<b>23</b>
	<b>Index</b>	<b>24</b>

# Chapter 1

## Introduction

### 1.1 General aims of the **ModularGroup** package

This **GAP** package provides methods for computing with finite-index subgroups of the modular groups  $SL_2(\mathbb{Z})$  and  $PSL_2(\mathbb{Z})$ . This includes, but is not limited to, computation of the generalized level, index or cusp widths. It also implements algorithms described in [Hsu96] and [HL14] for testing if a given group is a congruence subgroup. Hence it differs from the **Congruence** package, which can be used - among other things - to construct canonical congruence subgroups of  $SL_2(\mathbb{Z})$ .

### 1.2 Technicalities

A convenient way to represent finite-index subgroups of  $SL_2(\mathbb{Z})$  is by specifying the action of generator matrices of  $SL_2(\mathbb{Z})$  on the right cosets by right multiplication. For example, one could choose the generators

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

and represent a subgroup as a tuple of transitive permutations  $(\sigma_S, \sigma_T)$  describing the action of  $S$  and  $T$ . This is exactly the way this package internally treats such subgroups. We use the convention that 1 corresponds to the coset of the identity matrix.

## Chapter 2

# Subgroups of $SL_2(\mathbb{Z})$

For representing finite-index subgroups of  $SL_2(\mathbb{Z})$ , this package introduces the new object `ModularSubgroup`. As stated in the introduction, a `ModularSubgroup` essentially consists of the two permutations  $\sigma_S$  and  $\sigma_T$  describing the coset graph with respect to the generators  $S$  and  $T$  (with the convention that 1 corresponds to the identity coset). So explicitly specifying these permutations is the canonical way to construct a `ModularSubgroup`.

Though you might not always have a coset graph of your subgroup at hand, but rather a list of generating matrices. Therefore we implement multiple constructors for `ModularSubgroup`: three that take as input two permutations describing the coset graph with respect to different pairs of generators of  $SL_2(\mathbb{Z})$ , and one that takes a list of  $SL_2(\mathbb{Z})$  matrices as generators.

## 2.1 Construction of modular subgroups

### 2.1.1 Constructors

▷ `ModularSubgroup(s, t)` (operation)

**Returns:** A modular subgroup.

Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of  $SL_2(\mathbb{Z})$  described by the permutations  $s$  and  $t$ .

This constructor tests if the given permutations actually describe the coset action of the matrices

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

by checking that they act transitively and satisfy the relations

$$s^4 = (s^3t)^3 = s^2ts^{-2}t^{-1} = 1$$

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing `GAP` methods easier. (The fact that 1 corresponds to the identity coset is not changed by this)

Example

```
gap> G := ModularSubgroup(
> (1,2)(3,4)(5,6)(7,8)(9,10),
> (1,4)(2,5,9,10,8)(3,7,6));
<modular subgroup of index 10>
```

▷ `ModularSubgroupST(s, t)` (operation)

**Returns:** A modular subgroup.

Synonymous for `ModularSubgroup` (see above). ▷ `ModularSubgroupRT(r, t)` (operation)

**Returns:** A modular subgroup.

Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of  $SL_2(\mathbb{Z})$  determined by the permutations  $r$  and  $t$  which describe the action of the matrices

$$R = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

on the right cosets.

A check is performed if the permutations actually describe such an action on the cosets of some subgroup.

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing **GAP** methods easier. (The fact that 1 corresponds to the identity coset is not changed by this)

Example

```
gap> G := ModularSubgroupRT(
> (1,9,8,10,7)(2,6)(3,4,5),
> (1,4)(2,5,9,10,8)(3,7,6));
<modular subgroup of index 10>
```

▷ `ModularSubgroupSJ(s, j)` (operation)

**Returns:** A modular subgroup.

Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of  $SL_2(\mathbb{Z})$  determined by the permutations  $s$  and  $j$  which describe the action of the matrices

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad J = \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}$$

on the right cosets.

A check is performed if the permutations actually describe such an action on the cosets of some subgroup.

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing **GAP** methods easier. (The fact that 1 corresponds to the identity coset is not changed by this)

Example

```
gap> G := ModularSubgroupSJ(
> (1,2)(3,6)(4,7)(5,9)(8,10),
> (1,5,6)(2,3,7)(4,9,10));
<modular subgroup of index 10>
```

▷ `ModularSubgroup(gens)` (operation)

**Returns:** A modular subgroup.

Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of  $SL_2(\mathbb{Z})$  generated by the matrices in `gens`.

No test is performed to check if the generated subgroup actually has finite index!

This constructor implicitly computes a coset table of the subgroup. Hence it might be slow for very large index subgroups.

Example

```
gap> G := ModularSubgroup([
> [[1,2], [0,1]],
> [[1,0], [2,1]],
> [[-1,0], [0,-1]]
> ]);
<modular subgroup of index 6>
```

## 2.1.2 Getters for the coset action

▷ `SAction(G)` (operation)

**Returns:** A permutation.

Returns the permutation  $\sigma_S$  describing the action of the matrix  $S$  on the cosets of  $G$ .

▷ `TAction(G)`  
(operation)

**Returns:** A permutation.

Returns the permutation  $\sigma_T$  describing the action of the matrix  $T$  on the cosets of  $G$ .

▷ `RAction(G)` (operation)

**Returns:** A permutation.

Returns the permutation  $\sigma_R$  describing the action of the matrix  $R$  on the cosets of  $G$ .

▷ `JAction(G)` (operation)

**Returns:** A permutation.

Returns the permutation  $\sigma_J$  describing the action of the matrix  $J$  on the cosets of  $G$ .

▷ `CosetActionOf(A, G)` (operation)

**Returns:** A permutation.

Returns the permutation  $\sigma_A$  describing the action of the matrix  $A \in SL_2(\mathbb{Z})$  on the cosets of  $G$ .

## 2.2 Computing with modular subgroups

### 2.2.1 Index (IndexSL2Z)

▷ `Index(G)` (attribute)

**Returns:** A natural number.

For a given modular subgroup  $G$  this method returns its index in  $SL_2(\mathbb{Z})$ . As  $G$  is internally stored as permutations  $(s, t)$  this is just

`LargestMovedPoint(s, t)`

(or 1 if the permutations are trivial).

## Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> Index(G);
6
```

### 2.2.2 GeneralizedLevel (GeneralizedLevelSL2Z)

▷ `GeneralizedLevel(G)` (attribute)

**Returns:** A natural number.

This method calculates the general Wohlfahrt level (i.e. the lowest common multiple of all cusp widths) of  $G$  as defined in [Woh64].

## Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> GeneralizedLevel(G);
2
```

### 2.2.3 RightCosetRepresentatives (RightCosetRepresentativesSL2Z)

▷ `RightCosetRepresentatives(G)` (attribute)

**Returns:** A list of words.

This function returns a list of representatives of the (right) cosets of  $G$  as words in  $S$  and  $T$ .

## Example

```
gap> G := ModularSubgroup((1,2),(2,3));
<modular subgroup of index 3>
gap> RightCosetRepresentatives(G);
[ <identity ...>, S, S*T ]
```

### 2.2.4 GeneratorsOfGroup (GeneratorsOfGroupSL2Z)

▷ `GeneratorsOfGroup(G)` (attribute)

**Returns:** A list of words.

Calculates a list of generators (as words in  $S$  and  $T$ ) of  $G$ . This list might include redundant generators (or even duplicates).

## Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> GeneratorsOfGroup(G);
[ S^-2, T^-2, S*T^-2*S^-1 ]
```



## 2.2.5 MatrixGeneratorsOfGroup

▷ `MatrixGeneratorsOfGroup( $G$ )` (attribute)

**Returns:** A list of matrices.

Calculates a list of generator matrices of  $G$ . This list might include redundant generators (or even duplicates).

Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> MatrixGeneratorsOfGroup(G);
[ [ [ -1, 0 ], [ 0, -1 ] ], [ [ 1, -2 ], [ 0, 1 ] ], [ [ 1, 0 ], [ 2, 1 ] ] ]
```

## 2.2.6 IsCongruence (IsCongruenceSL2Z)

▷ `IsCongruence( $G$ )` (attribute)

**Returns:** True or false.

This method test whether a given modular subgroup  $G$  is a congruence subgroup. It is essentially an implementation of an algorithm described in [HL14].

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> IsCongruence(G);
true
```

## 2.2.7 Cusps (CuspsSL2Z)

▷ `Cusps( $G$ )` (attribute)

**Returns:** A list of rational numbers and infinity.

This method computes a list of inequivalent cusp representatives with respect to  $G$ .

Example

```
gap> G := ModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<modular subgroup of index 24>
gap> Cusps(G);
[ infinity, 0, 1, 2, 3/2, 5/3 ]
```

### 2.2.8 CuspWidth (CuspWidthSL2Z)

▷ CuspWidth( $c$ ,  $G$ ) (operation)

**Returns:** A natural number.

This method takes as input a cusp  $c$  (a rational number or infinity) and a modular group  $G$  and calculates the width of this cusp with respect to  $G$ .

Example

```
gap> G := ModularSubgroup(
> (1,2,6,3)(4,11,15,12)(5,13,16,14)(7,17,9,18)(8,19,10,20)(21,24,22,23),
> (1,4,5)(2,7,8)(3,9,10)(6,15,16)(11,20,21)(12,19,22)(13,23,17)(14,24,18)
> );
<modular subgroup of index 24>
gap> CuspWidth(-1, G);
3
gap> CuspWidth(infinity, G);
3
```

### 2.2.9 CuspsEquivalent (CuspsEquivalentSL2Z)

▷ CuspsEquivalent( $p$ ,  $q$ ,  $G$ ) (operation)

**Returns:** True or false.

Takes two cusps  $p$  and  $q$  and a modular subgroup  $G$  and checks if they are equivalent modulo  $G$ , i.e. if there exists a matrix  $A \in G$  with  $Ap = q$ .

Example

```
gap> G := ModularSubgroup(
> (1,2,6,3)(4,11,15,12)(5,13,16,14)(7,17,9,18)(8,19,10,20)(21,24,22,23),
> (1,4,5)(2,7,8)(3,9,10)(6,15,16)(11,20,21)(12,19,22)(13,23,17)(14,24,18)
> );
<modular subgroup of index 24>
gap> CuspsEquivalent(infinity, 1, G);
false
gap> CuspsEquivalent(-1, 1/2, G);
true
```

### 2.2.10 IndexModN (IndexModNSL2Z)

▷ IndexModN( $G$ ,  $N$ ) (operation)

**Returns:** A natural number.

For a modular subgroup  $G$  and a natural number  $N$  this method calculates the index of the projection  $\bar{G}$  of  $G$  in  $SL_2(\mathbb{Z}/N\mathbb{Z})$ .

Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> IndexModN(G, 2);
```

### 2.2.11 Deficiency (DeficiencySL2Z)

▷ Deficiency( $G$ ,  $N$ ) (operation)

**Returns:** A natural number.

For a modular subgroup  $G$  and a natural number  $N$  this method calculates the so-called *deficiency* of  $G$  from being a congruence subgroup of level  $N$ .

The deficiency of a finite-index subgroup  $\Gamma$  of  $SL_2(\mathbb{Z})$  was introduced in [WS15]. It is defined as the index  $[\Gamma(N) : \Gamma(N) \cap \Gamma]$  where  $\Gamma(N)$  is the principal congruence subgroup of level  $N$ .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> Deficiency(G, 2);
2
gap> Deficiency(G, 4);
1
```

### 2.2.12 Projection

▷ Projection( $G$ ) (operation)

**Returns:** A projective modular subgroup.

For a given modular subgroup  $G$  this function calculates its image  $\bar{G}$  under the projection  $\pi: SL_2(\mathbb{Z}) \rightarrow PSL_2(\mathbb{Z})$ .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> Projection(G);
<projective modular subgroup of index 6>
```

### 2.2.13 NormalCore (NormalCoreSL2Z)

▷ NormalCore( $G$ ) (attribute)

**Returns:** A modular subgroup.

Calculates the normal core of  $G$  in  $SL_2(\mathbb{Z})$ , i.e. the maximal subgroup of  $G$  that is normal in  $SL_2(\mathbb{Z})$ .

## Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> NormalCore(G);
<modular subgroup of index 48>
```

### 2.2.14 QuotientByNormalCore (QuotientByNormalCoreSL2Z)

▷ QuotientByNormalCore( $G$ )

(attribute)

**Returns:** A finite group.

Calculates the quotient of  $SL_2(\mathbb{Z})$  by the normal core of  $G$ .

## Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> QuotientByNormalCore(G);
<permutation group with 2 generators>
```

### 2.2.15 AssociatedCharacterTable (AssociatedCharacterTableSL2Z)

▷ AssociatedCharacterTable( $G$ )

(attribute)

**Returns:** A character table.

Returns the character table of  $SL_2(\mathbb{Z})/N$  where  $N$  is the normal core of  $G$ .

## Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> AssociatedCharacterTable(G);
CharacterTable( <permutation group of size 48 with 2 generators> )
```

### 2.2.16 IsElementOf

▷ IsElementOf( $A$ ,  $G$ )

(operation)

**Returns:** True or false.

This function checks if a given matrix  $A$  is an element of the modular subgroup  $G$ .

## Example

```

gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> IsElementOf([[-1,0],[0,-1]], G);
false
gap> IsElementOf([[1,4],[0,1]], G);
true

```

## 2.3 Miscellaneous

The following functions are mostly helper functions used internally and are only documented for sake of completeness.

### 2.3.1 DefinesCosetActionST

▷ DefinesCosetActionST( $s$ ,  $t$ ) (operation)

**Returns:** True or false.

Checks if two given permutations  $s$  and  $t$  describe the action of the generator matrices  $S$  and  $T$  on the cosets of some subgroup. This is the case if they satisfy the relations

$$s^4 = (s^3t)^3 = s^2ts^{-2}t^{-1} = 1$$

and act transitively.

## Example

```

gap> s := (1,2)(3,4)(5,6)(7,8)(9,10);;
gap> t := (1,4)(2,5,9,10,8)(3,7,6);;
gap> DefinesCosetActionST(s,t);
true

```

### 2.3.2 DefinesCosetActionRT

▷ DefinesCosetActionRT( $r$ ,  $t$ ) (operation)

**Returns:** True or false.

Checks if two given permutations  $r$  and  $t$  describe the action of the generator matrices  $R$  and  $T$  on the cosets of some subgroup. This is the case if they satisfy the relations

$$(rt^{-1}r)^4 = ((rt^{-1}r)^3t)^3 = (rt^{-1}r)^2t(rt^{-1}r)^{-2}t^{-1} = 1$$

and act transitively.

## Example

```

gap> r := (1,9,8,10,7)(2,6)(3,4,5);;

```

```
gap> t := (1,4)(2,5,9,10,8)(3,7,6);;
gap> DefinesCosetActionRT(r,t);
true
```

### 2.3.3 DefinesCosetActionSJ

▷ `DefinesCosetActionSJ(s, j)` (operation)

**Returns:** True or false.

Checks if two given permutations  $s$  and  $j$  describe the action of the generator matrices  $S$  and  $J$  on the cosets of some subgroup. This is the case if they satisfy the relations

$$s^4 = (s^3 j^{-1} s^{-1})^3 = s^2 j^{-1} s^{-2} j = 1$$

and act transitively.

Example

```
gap> s := (1,2)(3,4)(5,6)(7,8)(9,10);;
gap> j := (1,5,6)(2,3,7)(4,9,10);;
gap> DefinesCosetActionSJ(s,j);
true
```

### 2.3.4 CosetActionFromGenerators

▷ `CosetActionFromGenerators(gens)` (operation)

**Returns:** A tuple of permutations.

Takes a list of generator matrices and calculates the coset graph (as two permutations  $\sigma_S$  and  $\sigma_T$ ) of the generated subgroup of  $SL_2(\mathbb{Z})$ .

Example

```
gap> CosetActionFromGenerators([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
[ (1,2,5,3)(4,8,10,9)(6,11,7,12), (1,4)(2,6)(3,7)(5,10)(8,12,9,11) ]
```

### 2.3.5 STDecomposition

▷ `STDecomposition(A)` (operation)

**Returns:** A word in  $S$  and  $T$ .

Takes a matrix  $A \in SL_2(\mathbb{Z})$  and decomposes it into a word in the generator matrices  $S$  and  $T$ .

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> STDecomposition(M);
S^2*T^-1*S^-1*T^2*S^-1*T^-1*S^-1
```

### 2.3.6 RTDecomposition

▷ `RTDecomposition(A)` (operation)

**Returns:** A word in  $R$  and  $T$ .

Takes a matrix  $A \in SL_2(\mathbb{Z})$  and decomposes it into a word in the generator matrices  $R$  and  $T$ .

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> RTDecomposition(M);
(R*T^-1*R)^2*T^-1*R^-1*(T*R^-1*T)^2*R^-1*T^-1*R^-1*T*R^-1
```

### 2.3.7 SJDecomposition

▷ `SJDecomposition(A)` (operation)

**Returns:** A word in  $S$  and  $J$ .

Takes a matrix  $A \in SL_2(\mathbb{Z})$  and decomposes it into a word in the generator matrices  $S$  and  $J$ .

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> SJDecomposition(M);
S^3*J*(S^-1*J^-1)^2*S^-1*J*S^-1
```

### 2.3.8 STDecompositionAsList

▷ `STDecompositionAsList(A)` (operation)

**Returns:** A list representing a word in  $S$  and  $T$ .

Takes a matrix  $A \in SL_2(\mathbb{Z})$  and decomposes it into a word in the generator matrices  $S$  and  $T$ . The word is represented as a list in the format `[[generator, exponent], ...]`

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> STDecompositionAsList(M);
[ [ "S", 2 ], [ "T", -1 ], [ "S", -1 ], [ "T", 2 ], [ "S", -1 ], [ "T", -1 ],
  [ "S", -1 ], [ "T", 0 ] ]
```

## Chapter 3

# Subgroups of $PSL_2(\mathbb{Z})$

Analogous to finite-index subgroups of  $SL_2(\mathbb{Z})$ , we define a new type `ProjectiveModularSubgroup` for representing subgroups of  $PSL_2(\mathbb{Z})$ . It consists essentially of two permutations  $\sigma_{\bar{S}}$  and  $\sigma_{\bar{T}}$  describing the action of  $\bar{S}$  and  $\bar{T}$  on the cosets of the given subgroup, where  $\bar{S}$  and  $\bar{T}$  are the images of  $S$  and  $T$  in  $PSL_2(\mathbb{Z})$ .

The methods implemented for  $PSL_2(\mathbb{Z})$  subgroups are mostly the same as for  $SL_2(\mathbb{Z})$  subgroups and behave more or less identically. Nevertheless we list them here.

### 3.1 Construction of projective modular subgroups

#### 3.1.1 Constructors

▷ `ProjectiveModularSubgroup(s, t)` (operation)

**Returns:** A projective modular subgroup.

Constructs a `ProjectiveModularSubgroup` object corresponding to the finite-index subgroup of  $PSL_2(\mathbb{Z})$  described by the permutations  $s$  and  $t$ .

This constructor tests if the given permutations actually describe the coset action of  $\bar{S}$  and  $\bar{T}$  on some subgroup by checking that they act transitively and satisfy the relations

$$s^2 = (st)^3 = 1$$

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing `GAP` methods easier.

Example

```
gap> G := ProjectiveModularSubgroup(  
> (1,2)(3,4)(5,6)(7,8)(9,10),  
> (1,4)(2,5,9,10,8)(3,7,6));  
<projective modular subgroup of index 10>
```

If you want to construct a `ProjectiveModularSubgroup` from a list of generators, you can lift each generator to a matrix in  $SL_2(\mathbb{Z})$ , construct from these a `ModularSubgroup`, and then project it to  $PSL_2(\mathbb{Z})$  via `Projection`.



### 3.1.2 Getters for the coset action

- ▷ `SAction( $G$ )` (operation)  
**Returns:** A permutation.  
Returns the permutation  $\sigma_{\bar{S}}$  describing the action of  $\bar{S}$  on the cosets of  $G$ .
- ▷ `TAction( $G$ )` (operation)  
**Returns:** A permutation.  
Returns the permutation  $\sigma_{\bar{T}}$  describing the action of  $\bar{T}$  on the cosets of  $G$ .

## 3.2 Computing with projective modular subgroups

### 3.2.1 Index (IndexPSL2Z)

- ▷ `Index( $G$ )` (attribute)  
**Returns:** A natural number.  
For a given projective modular subgroup  $G$  this method returns its index in  $PSL_2(\mathbb{Z})$ . As  $G$  is internally stored as permutations  $(s, t)$  this is just

`LargestMovedPoint(s, t)`

(or 1 if the permutations are trivial).

Example

```
gap> G := ProjectiveModularSubgroup((1,2),(2,3));
<projective modular subgroup of index 3>
gap> Index(G);
3
```

### 3.2.2 GeneralizedLevel (GeneralizedLevelPSL2Z)

- ▷ `GeneralizedLevel( $G$ )` (attribute)  
**Returns:** A natural number.  
This method calculates the general Wohlfahrt level (i.e. the lowest common multiple of all cusp widths) of  $G$  as defined in [Woh64].

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> GeneralizedLevel(G);
12
```

### 3.2.3 RightCosetRepresentatives (RightCosetRepresentativesPSL2Z)

▷ `RightCosetRepresentatives( $G$ )` (attribute)

**Returns:** A list of words.

This function returns a list of representatives of the (right) cosets of  $G$  as words in  $\bar{S}$  and  $\bar{T}$ .

Example

```
gap> G := ProjectiveModularSubgroup((1,2),(2,3));
<projective modular subgroup of index 3>
gap> RightCosetRepresentatives(G);
[ <identity ...>, S, S*T ]
```

### 3.2.4 GeneratorsOfGroup (GeneratorsOfGroupPSL2Z)

▷ `GeneratorsOfGroup( $G$ )` (attribute)

**Returns:** A list of words.

Calculates a list of generators (as words in  $\bar{S}$  and  $\bar{T}$ ) of  $G$ . This list might include redundant generators.

Example

```
gap> G := ProjectiveModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<projective modular subgroup of index 6>
gap> GeneratorsOfGroup(G);
[ T^-2, S*T^-2*S^-1 ]
```

### 3.2.5 IsCongruence (IsCongruencePSL2Z)

▷ `IsCongruence( $G$ )` (attribute)

**Returns:** True or false.

This method test whether a given modular subgroup  $G$  is a congruence subgroup. It is essentially an implementation of an algorithm described in [Hsu96].

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,5)(4,6),
> (1,3)(2,4)(5,6)
> );
<projective modular subgroup of index 6>
gap> IsCongruence(G);
true
```

### 3.2.6 Cusps (CuspsPSL2Z)

▷ `Cusps( $G$ )` (attribute)

**Returns:** A list of rational numbers and infinity.

This method computes a list of inequivalent cusp representatives with respect to  $G$ .

## Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> Cusps(G);
[ infinity, 0, 1, 2, 3/2, 5/3 ]
```

### 3.2.7 CuspWidth (CuspWidthPSL2Z)

▷ CuspWidth( $c$ ,  $G$ ) (operation)

**Returns:** A natural number.

This method takes as input a cusp  $c$  (a rational number or infinity) and a modular group  $G$  and calculates the width of this cusp with respect to  $G$ .

## Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> CuspWidth(-1, G);
3
gap> CuspWidth(infinity, G);
3
```

### 3.2.8 CuspsEquivalent (CuspsEquivalentPSL2Z)

▷ CuspsEquivalent( $p$ ,  $q$ ,  $G$ ) (operation)

**Returns:** True or false.

Takes two cusps  $p$  and  $q$  and a projective modular subgroup  $G$  and checks if they are equivalent modulo  $G$ , i.e. if there exists  $A \in G$  with  $Ap = q$ .

## Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> CuspsEquivalent(infinity, 1, G);
false
gap> CuspsEquivalent(-1, 1/2, G);
true
```

### 3.2.9 LiftToSL2ZEven

▷ `LiftToSL2ZEven( $G$ )` (operation)

**Returns:** A modular subgroup.

Lifts a given subgroup  $G$  of  $PSL_2(\mathbb{Z})$  to an even subgroup of  $SL_2(\mathbb{Z})$ , i.e. a group that contains  $-1$  and whose projection to  $PSL_2(\mathbb{Z})$  is  $G$ .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> LiftToSL2Z0dd(G);
<modular subgroup of index 12>
```

### 3.2.10 LiftToSL2ZOdd

▷ `LiftToSL2Z0dd( $G$ )` (operation)

**Returns:** A modular subgroup.

Lifts a given subgroup  $G$  of  $PSL_2(\mathbb{Z})$  to an odd subgroup of  $SL_2(\mathbb{Z})$ , i.e. a group that does not contain  $-1$  and whose projection to  $PSL_2(\mathbb{Z})$  is  $G$ .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> LiftToSL2Z0dd(G);
<modular subgroup of index 24>
```

### 3.2.11 IndexModN (IndexModNPSL2Z)

▷ `IndexModN( $G$ ,  $N$ )` (operation)

**Returns:** A natural number.

For a projective modular subgroup  $G$  and a natural number  $N$  this method calculates the index of the projection  $\bar{G}$  of  $G$  in  $PSL_2(\mathbb{Z}/N\mathbb{Z})$ .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> IndexModN(G, 2);
6
```

### 3.2.12 Deficiency (DeficiencyPSL2Z)

▷ Deficiency( $G$ ,  $N$ ) (operation)

**Returns:** A natural number.

For a projective modular subgroup  $G$  and a natural number  $N$  this method calculates the so-called *deficiency* of  $G$  from being a congruence subgroup of level  $N$ .

The deficiency of a finite-index subgroup  $\Gamma$  of  $PSL_2(\mathbb{Z})$  was introduced in [WS15]. It is defined as the index  $[\Gamma(N) : \Gamma(N) \cap \Gamma]$  where  $\Gamma(N)$  is the principal congruence subgroup of level  $N$ .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> Deficiency(G, 2);
4
```

### 3.2.13 NormalCore (NormalCorePSL2Z)

▷ NormalCore( $G$ ) (attribute)

**Returns:** A projective modular subgroup.

Calculates the normal core of  $G$  in  $PSL_2(\mathbb{Z})$ , i.e. the maximal subgroup of  $G$  that is normal in  $PSL_2(\mathbb{Z})$ .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> NormalCore(G);
<projective modular subgroup of index 3456>
```

### 3.2.14 QuotientByNormalCore (QuotientByNormalCorePSL2Z)

▷ QuotientByNormalCore( $G$ ) (attribute)

**Returns:** A finite group.

Calculates the quotient of  $PSL_2(\mathbb{Z})$  by the normal core of  $G$ .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> QuotientByNormalCore(G);
```

```
<permutation group with 2 generators>
```

### 3.2.15 AssociatedCharacterTable (AssociatedCharacterTablePSL2Z)

▷ AssociatedCharacterTable( $G$ )

(attribute)

**Returns:** A character table.

Returns the character table of  $PSL_2(\mathbb{Z})/N$  where  $N$  is the normal core of  $G$ .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> AssociatedCharacterTable(G);
CharacterTable( <permutation group of size 3456 with 2 generators> )
```

# References

- [HL14] Thomas Hamilton and David Loeffler. Congruence testing for odd subgroups of the modular group. *LMS Journal of Computation and Mathematics*, 17(1):206–208, 2014. [4](#), [9](#)
- [Hsu96] Tim Hsu. Identifying congruence subgroups of the modular group. *Proceedings of the American Mathematical Society*, 124(5), April 1996. [4](#), [18](#)
- [Woh64] Klaus Wohlfahrt. An extension of F. Klein’s level concept. *Illinois J. Math.*, 8(3):529–535, 09 1964. [8](#), [17](#)
- [WS15] Gabriela Weitze-Schmithuesen. The deficiency of being a congruence group for veech groups of origamis. *International Mathematics Research Notices*, 2015(6):1613–1637, 2015. [11](#), [21](#)

# Index

AssociatedCharacterTable  
  AssociatedCharacterTablePSL2Z, [22](#)  
  AssociatedCharacterTableSL2Z, [12](#)

CosetActionFromGenerators, [14](#)  
CosetActionOf, [7](#)  
Cusps  
  CuspsPSL2Z, [18](#)  
  CuspsSL2Z, [9](#)  
CuspsEquivalent  
  CuspsEquivalentPSL2Z, [19](#)  
  CuspsEquivalentSL2Z, [10](#)  
CuspWidth  
  CuspWidthPSL2Z, [19](#)  
  CuspWidthSL2Z, [10](#)

Deficiency  
  DeficiencyPSL2Z, [21](#)  
  DeficiencySL2Z, [11](#)  
DefinesCosetActionRT, [13](#)  
DefinesCosetActionSJ, [14](#)  
DefinesCosetActionST, [13](#)

GeneralizedLevel  
  GeneralizedLevelPSL2Z, [17](#)  
  GeneralizedLevelSL2Z, [8](#)  
GeneratorsOfGroup  
  GeneratorsOfGroupPSL2Z, [18](#)  
  GeneratorsOfGroupSL2Z, [8](#)

Index  
  IndexPSL2Z, [17](#)  
  IndexSL2Z, [7](#)  
IndexModN  
  IndexModNPSL2Z, [20](#)  
  IndexModNSL2Z, [10](#)  
IsCongruence  
  IsCongruencePSL2Z, [18](#)  
  IsCongruenceSL2Z, [9](#)  
IsElementOf, [12](#)

JAction, [7](#)  
LiftToSL2ZEven, [20](#)  
LiftToSL2ZOdd, [20](#)  
MatrixGeneratorsOfGroup, [9](#)  
ModularSubgroup, [5](#)  
  ModularSubgroupGens, [7](#)  
ModularSubgroupRT, [6](#)  
ModularSubgroupST, [6](#)  
ModularSubgroupSJ, [6](#)  
NormalCore  
  NormalCorePSL2Z, [21](#)  
  NormalCoreSL2Z, [11](#)  
Projection, [11](#)  
ProjectiveModularSubgroup, [16](#)  
QuotientByNormalCore  
  QuotientByNormalCorePSL2Z, [21](#)  
  QuotientByNormalCoreSL2Z, [12](#)  
RAction, [7](#)  
RightCosetRepresentatives  
  RightCosetRepresentativesPSL2Z, [18](#)  
  RightCosetRepresentativesSL2Z, [8](#)  
RTDecomposition, [15](#)  
SAction  
  SActionPSL2Z, [17](#)  
  SActionSL2Z, [7](#)  
SJDecomposition, [15](#)  
STDecomposition, [14](#)  
STDecompositionAsList, [15](#)  
TAction  
  TActionPSL2Z, [17](#)  
  TActionSL2Z, [7](#)