

Small Genus Package

Peter A. Brooksbank

Bucknell University
pbrooksb@bucknell.edu

Joshua Maglione

Colorado State University
josh.maglione@gmail.com

James B. Wilson

Colorado State University
James.Wilson@ColoState.Edu

Version 0.1
May 23, 2017

©2015 Peter A. Brooksbank, Joshua Maglione, James B. Wilson

Contents

Chapter 1. Introduction	1
1. Verbose Printing and Version	1
Chapter 2. Constructors	3
1. Examples	3
Chapter 3. Automorphism groups	5
1. Examples	6
Chapter 4. Isomorphisms	7
1. Examples	8
Chapter 5. Canonical Labeling	9
1. Examples	9
Bibliography	11
Intrinsics	13

CHAPTER 1

Introduction

The goal of this package is to provide MAGMA [BCP] with high performance functionality for groups with small genus. When appropriate, all intrinsics dealing with groups also work for tensors (`TenSpcElt`). This package includes intrinsics for automorphism and isomorphism computations, canonical labeling, and constructing random groups of genus 1 or 2 of exponent $\leq p^2$. Mathematical details of these algorithms can be found in [BMW].

1. Verbose Printing and Version

We have included intrinsics to allow for verbose printing. Currently, there is only one level of printing, so either it is on or off.

`SetVerbose(MonStgElt, RngIntElt) : ->`

`SetVerbose` is a built in Magma function, but with this package, this intrinsic accepts the string "SmallGenus" and an integer in $\{0, 1\}$.

We have included an intrinsic to check which version of SmallGenus you have attached in Magma.

`SmallGenusVersion() : -> MonStgElt`

Returns the version number for the SmallGenus package attached in Magma.

CHAPTER 2

Constructors

We introduce intrinsics to construct nonabelian groups of genus ≤ 2 . We provide some algorithms to construct random groups, and we make no attempt at asserting any analysis of these algorithms. In particular, these random constructors may not select groups in a uniform distribution.

RandomGroupSG(*q*, *n*, *g* : parameters) : RngIntElt, RngIntElt, RngIntElt -> GrpPC
Exponentp : BoolElt : true

Given $q = p^m$, $n > 0$, and $g > 0$, returns a p -group G with genus $\leq g$ and order $q^{n+g} = p^{m(n+g)}$. If q is not a prime, then the centroid of the group will be a proper field extension of \mathbb{F}_p , isomorphic to \mathbb{F}_q . If $\circ : V \times V \rightarrow W$ is the commutation tensor of G , then $\dim_{\mathcal{C}(\circ)}(V) = n$ and $\dim_{\mathcal{C}(\circ)}(W) = g$, where $\mathcal{C}(\circ)$ is the centroid of \circ . The algorithm is based on the Universal Coefficients Theorem, see [LGM, Chapter 9] the statement and proof. There is one optional parameter: **Exponentp**.

Exponentp: The default is set to **true**. Set to false if you want the returned group to have exponent $\leq p^2$.

RandomGenus2Group(*q*, *d* : parameters) : RngIntElt, [RngIntElt] -> GrpPC
Exponentp : BoolElt : true

Given $q = p^m$ and $d = [d_1, \dots, d_k]$, where $d_i > 0$, it returns a genus 2 group G whose commutation tensor $\circ : V \times V \rightarrow W$ has centroid $\mathcal{C}(\circ) \cong \mathbb{F}_q$ and whose \perp -decomposition has blocks of dimensions d_1, \dots, d_k (over $\mathcal{C}(\circ)$). If $d_i = 1$, then this increases the dimension of the radical by 1 (over $\mathcal{C}(\circ)$). There is one optional parameter: **Exponentp**.

Exponentp: The default is set to **true**. Set to false if you want the returned group to have exponent $\leq p^2$.

RandomGenus1Group(*q*, *d*, *r* : parameters) : RngIntElt, RngIntElt, RngIntElt -> GrpPC
Exponentp : BoolElt : true

Given $q = p^m$, $d > 0$, and $r \geq 0$, it returns a group G with genus 1 and of order q^{2d+r+1} . The center of G has order q^{r+1} , and G is dm -generated. There is one optional parameter: **Exponentp**.

Exponentp: The default is set to **true**. Set to false if you want the returned group to have exponent $\leq p^2$.

Genus2Group(*f*) : RngUPolElt -> GrpPC
Genus2Group(*f*) : RngMPolElt -> GrpPC

Given either a univariate or homogeneous multivariate polynomial in x (and y), it returns a group G whose tensor from commutation has a Pfaffian equivalent to f .

1. Examples

CHAPTER 3

Automorphism groups

We have two intrinsics for constructing automorphism groups: one with group inputs and the other with tensor inputs.

```
AutomorphismGroupSG(G : parameters) : GrpPC -> GrpAuto
  Cent : BoolElt : true
  Method : RngIntElt : 0
  Order : BoolElt : false
```

Given a p -group G , of class 2, exponent p , and genus ≤ 2 , this returns $\text{Aut}(G)$. This currently does not work with $p = 2$. This intrinsic provides three optional parameters: **Cent**, **Method**, and **Order**.

Cent: This parameter is a flag for the algorithm to rewrite the tensor over its centroid. In order to rewrite over the centroid, we assume the centroid is a local ring, see [MW] for details. The default is set to **true**, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If you know the centroid is trivial, set **Cent** to **false** to save time. WARNING: if the centroid is a proper field extension of \mathbb{F}_p , then the full automorphism group may not be constructed (this is a bug we will fix in the near future).

Method: The default is 0, and it accepts any input from $\{0, 1, 2\}$. This will set the method for handling the sloped part of the tensor. If you want to use the adjoint-tensor method, set **Method** to 1, and if you want to use the Pfaffian method, set **Method** to 2. The default will try to find the optimal method based on the input; this is not optimally tuned.

Order: Finally, **Order** is included so you have the order of the returned automorphism group. The data structure of **GrpAuto** makes it hard to compute the order of the group. However, when computing $\text{Aut}(G)$, we construct a (linear) representation of the group. We can take advantage of the LMG functions [HOM] in Magma to compute the order of the representation. The default is set to **false**. WARNING: if set to **true** this step could take the majority of the computation time.

```
PseudoIsometryGroupSG(T : parameters) : TenSpcElt -> GrpMat
  Cent : BoolElt : true
  Method : RngIntElt : 0
```

Given an alternating tensor $T : V \times V \rightarrow W$, where V and W are \mathbb{F}_q -vector spaces, returns the pseudo-isometry group $\Psi\text{Isom}(T) \leq \text{GL}(V) \times \text{GL}(W)$. This currently does not work for even q . This intrinsic provides two optional parameters: **Cent** and **Method**.

Cent: This parameter is a flag for the algorithm to rewrite the tensor over its centroid. In order to rewrite over the centroid, we assume the centroid is a local ring, see [MW] for details. The default is set to **true**, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If you know the centroid is trivial, set **Cent** to **false** to save time. WARNING: if the centroid is a proper field extension of \mathbb{F}_q , then the full pseudo-isometry group may not be constructed (this is a bug we will fix in the near future).

Method: The default is 0, and it accepts any input from $\{0, 1, 2\}$. This will set the method for handling the sloped part of the tensor. If you want to use the adjoint-tensor method, set **Method** to 1, and if you want to use the Pfaffian method, set **Method** to 2. The default will try to find the optimal method based on the input; this is not optimally tuned.

1. Examples

We list some examples to illustrate the functionality of the automorphism invariants.

CHAPTER 4

Isomorphisms

```

IsIsomorphicSG(G, H : parameters) : GrpPC, GrpPC -> BoolElt
  Cent : BoolElt : true
  Constructive : BoolElt : true
  Method : RngIntElt : 0

```

Given class 2, exponent p , p -groups G and H of genus ≤ 2 , decides if $G \cong H$. If $G \cong H$, then an isomorphism is provided unless specified otherwise. Currently this does not work for $p = 2$. There are three optional parameters: **Cent**, **Constructive**, and **Method**.

Cent: This parameter is a flag for the algorithm to rewrite the tensors over their centroids. In order to rewrite over the centroid, we assume the centroid is a local ring, see [MW] for details. The default is set to **true**, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If you know the centroid is trivial, set **Cent** to **false** to save time. WARNING: if the centroid is a proper field extension of \mathbb{F}_p , then the full coset of (potential) isomorphisms are not fully exhausted. That is, it is possible the algorithm concludes **false** when $G \cong H$ (this is a bug we will fix in the near future).

Constructive: The default is **true**. Set to **false** if you do not want the algorithm to explicitly construct an isomorphism.

Method: The default is 0, and it accepts any input from $\{0, 1, 2\}$. This will set the method for handling the sloped part of the tensor. If you want to use the adjoint-tensor method, set **Method** to 1, and if you want to use the Pfaffian method, set **Method** to 2. The default will try to find the optimal method based on the input; this is not optimally tuned.

```

IsPseudoIsometricSG(T, S : parameters) : TenSpcElt, TenSpcElt -> BoolElt
  Cent : BoolElt : true
  Constructive : BoolElt : true
  Method : RngIntElt : 0

```

Given alternating tensors $T, S : V \times V \rightarrow W$, where V and W are \mathbb{F}_q -vector spaces, it decides if T is pseudo-isometric to S . If T is pseudo-isometric to S , then a pseudo-isometry is provided (as an element of $\text{GL}(V) \times \text{GL}(W)$) unless specified otherwise. Currently this does not work for even q . There are three optional parameters: **Cent**, **Constructive**, and **Method**.

Cent: This parameter is a flag for the algorithm to rewrite the tensors over their centroids. In order to rewrite over the centroid, we assume the centroid is a local ring, see [MW] for details. The default is set to **true**, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If you know the centroid is trivial, set **Cent** to **false** to save time. WARNING: if the centroid is a proper field extension of \mathbb{F}_q , then the full coset of (potential) pseudo-isometries are not fully exhausted. That is, it is possible the algorithm concludes **false** when T is pseudo-isometric to S (this is a bug we will fix in the near future).

Constructive: The default is **true**. Set to **false** if you do not want the algorithm to explicitly construct an isomorphism.

Method: The default is 0, and it accepts any input from $\{0, 1, 2\}$. This will set the method for handling the sloped part of the tensor. If you want to use the adjoint-tensor method,

set **Method** to 1, and if you want to use the Pfaffian method, set **Method** to 2. The default will try to find the optimal method based on the input; this is not optimally tuned.

1. Examples

We list some examples to illustrate the functionality of the isomorphism invariants.

CHAPTER 5

Canonical Labeling

`Genus(G) : GrpPC -> RngIntElt`
`Genus(T) : TenSpcElt -> RngIntElt`

Given a p -group G , returns the genus of G . That is, the dimension of $[G, G]$ over its centroid. For a tensor T , it returns the dimension of its image over its centroid.

`Genus2Signature(G) : GrpPC -> List`
`Genus2Signature(T) : TenSpcElt -> List`

Given a genus 2 group or tensor, returns the canonical genus 2 signature. The list has two parts. The first entry is a sequence of odd integers corresponding to the dimensions of the flat indecomposable spaces. The second entry is a list of sequences in \mathbb{F}_q , the field the given data is defined over. The sequences in the second entry are the coefficients of the homogeneous polynomial in x and y in degree d :

$$c_1x^d + c_2x^{d-1}y + c_3x^{d-2}y^2 + \cdots + c_{d+1}y^d.$$

Because this is a canonical label, two groups (or tensors) are isoclinic (or pseudo-isometric) if, and only if, their genus 2 signatures are equal.

Currently, this does not work for even q . WARNING: if the centroid is a proper field extension of \mathbb{F}_q , the field the data is defined over, then this is not a canonical label. That is, two groups can be isomorphic but have two different genus 2 signatures. This is a bug we will fix in the near future.

1. Examples

We list some examples to illustrate the functionality of these intrinsics.

Bibliography

- [BCP] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265. Computational algebra and number theory (London, 1993). MR1484478
- [BMW] Peter A. Brooksbank, Joshua Maglione, and James B. Wilson, *A fast isomorphism test for groups whose Lie algebra has genus 2*, J. Algebra **473** (2017), 545–590. MR3591162
- [HOM] Derek F. Holt, E. A. O’Brien, and MORE???, *The Large Matrix Group Package*, <http://magma.maths.usyd.edu.au/magma/handbook/text/680>.
- [LGM] C. R. Leedham-Green and S. McKay, *The structure of groups of prime power order*, London Mathematical Society Monographs. New Series, vol. 27, Oxford University Press, Oxford, 2002. Oxford Science Publications. MR1918951
- [MW] Joshua Maglione and James B. Wilson, *The eMAGma Package*, <https://github.com/algeboy/eMAGma>.

Intrinsics

AutomorphismGroupSG, 5

Genus

groups, 9

tensors, 9

Genus2Group, 3

Genus2Signature

groups, 9

tensors, 9

IsIsomorphicSG, 7

IsPseudoIsometricSG, 7

PseudoIsometryGroupSG, 5

RandomGenus1Group, 3

RandomGenus2Group, 3

RandomGroupSG, 3

SetVerbose, 1

SmallGenusVersion, 1