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 21, 2017

Contents

CHAPTER 1

Introduction

The goal of this package is to provide MAGMA [**BCP**] with high performance functionality for groups with small genus. The uniqueness of organizing groups by genus allows for a seemless transition to systems of forms. When appropriate, all intrinsics dealing with groups also work for systems of forms. This package includes intrinsics for automorphism and isomorphism computations as well as constructing random groups of genus 1 or 2 and exponent p. Details of these algorithms can be found in [**BMW**].

There are a few intrinsics which are not included in MAGMA; specifically, those dealing with systems of forms. We have attempted to make these intrinsics applicable to general systems of forms. However, if the genus is larger than 2, then it resorts to a (basic) brute force algorithm.

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 of 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

Automorphism groups

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

Given a p-group G, returns $\mathrm{Aut}(G)$. If G has genus 1 or 2 and exponent p, then it takes advantage of this structure to compute the automorphism group with much more efficient algorithms. This intrinsic provides three optional parameters: `Method`, `Print`, and `Order`, which are only used when the genus of G is either 1 or 2.

`Method`**:** The default is 0, and it accepts any input from $\{0, 1, 2\}$. For a genus 2 group, this will set the method for handling the sloped part of the bimap. 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.

`Print`**:** The default is set to `false`. To get a print out of the process, set `Print` to `true`.

`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. Using the LMG functions, we are able to do this at the end of the computation. The default is set to `false`. Warning: if set to `true` this step will likely take the majority of the computation time.

```
PseudoIsomtryGroup( F ) : SeqEnum[AlgMatElt] -> GrpMat
```

Given a system of forms F over $\mathbb{F}_q$, returns the pseudo-isometry group $\Psi\mathrm{Isom}(F)$. This intrinsic provides two optional parameters: `Method` and `Print`. For systems of 3 or more forms, for nonalternating systems, or when the characteristic is 2, the algorithm uses brute force.

`Method`**:** The default is 0, and it accepts any input from $\{0, 1, 2\}$. For a pair of alternating forms, this will set the method for handling the sloped part of the bimap. 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.

`Print`**:** The default is set to `false`. To get a print out of the process, set `Print` to `true`.

1. Examples

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

```
> G;
GrpPC : G of order 2187 = 3^7
PC-Relations:
    G.2^G.1 = G.2 * G.6,
    G.3^G.1 = G.3 * G.7,
    G.4^G.2 = G.4 * G.6^2
> A := SGAutomorphismGroup(G,3);
> A.6;
Automorphism of GrpPC : G which maps:
```

```
    G.1 |--> G.1 * G.3^2
    G.2 |--> G.2
    G.3 |--> G.3
    G.4 |--> G.4
    G.5 |--> G.5
    G.6 |--> G.6
    G.7 |--> G.7
```

```
> G;
GrpPC : G of order 531441 = 3^12
PC-Relations:
    G.2^G.1 = G.2 * G.11,
    G.3^G.2 = G.3 * G.12,
    G.4^G.1 = G.4 * G.12,
    G.4^G.2 = G.4 * G.11,
    G.7^G.6 = G.7 * G.11^2,
    G.9^G.5 = G.9 * G.12,
    G.10^G.8 = G.10 * G.11^2
> A := SGAutomorphismGroup(G,3 : Print:=true);
Computing the adjoint algebra... 0.020 seconds.
Computing perp-decomposition... 0.000 seconds.
4 sloped blocks and 0 flat blocks.
Using the Pfaffian method... 0.010 seconds.
Constructing the isometries... 0.000 seconds.
>
> A := SGAutomorphismGroup(G,3 :
       Method:=1,Print:=true);
Computing the adjoint algebra... 0.020 seconds.
Computing perp-decomposition... 0.010 seconds.
4 sloped blocks and 0 flat blocks.
Using the adjoint tensor method... 0.060 seconds.
Constructing the isometries... 0.010 seconds.
>
> A := SGAutomorphismGroup(G,3 :
       Print:=true,Order:=true);
Computing the adjoint algebra... 0.020 seconds.
Computing perp-decomposition... 0.010 seconds.
4 sloped blocks and 0 flat blocks.
Using the Pfaffian method... 0.010 seconds.
Constructing the isometries... 0.010 seconds.
Computing the order... 0.030 seconds.
> #A;
910796483224456888320
```

```
> F;
[
    [0 2 3 1 4 3 4 2]
    [3 0 2 0 4 0 1 4]
    [2 3 0 1 3 2 1 3]
    [4 0 4 0 4 4 0 4]
```

```
    [1 1 2 1 0 0 3 2]
    [2 0 3 1 0 0 1 1]
    [1 4 4 0 2 4 0 1]
    [3 1 2 1 3 4 4 0],

    [0 1 1 0 4 4 2 0]
    [4 0 2 1 2 3 2 3]
    [4 3 0 2 4 1 4 2]
    [0 4 3 0 0 1 2 4]
    [1 3 1 0 0 0 2 1]
    [1 2 4 4 0 0 1 1]
    [3 3 1 3 3 4 0 1]
    [0 2 3 1 4 4 4 0]
]
> PI := PseudoIsometryGroup(F);
> Random(PI);
[3 3 0 0 0 2 1 4 0 0]
[2 0 2 0 0 0 4 2 0 0]
[1 2 2 3 3 3 1 2 0 0]
[0 4 1 1 0 0 4 1 0 0]
[3 3 3 0 4 1 3 0 0 0]
[3 3 1 4 2 1 3 0 0 0]
[0 4 3 2 0 1 2 1 0 0]
[1 2 1 1 0 4 1 2 0 0]
[0 0 0 0 0 0 0 0 1 2]
[0 0 0 0 0 0 0 0 0 4]
> PI := PseudoIsometryGroup(F : Print:=true);
Computing the adjoint algebra... 0.010 seconds.
Computing perp-decomposition... 0.000 seconds.
3 sloped blocks and 0 flat blocks.
Using the Pfaffian method... 0.010 seconds.
Constructing the isometries... 0.010 seconds.
```

CHAPTER 3

Isomorphisms

SGIsIsomorphic(G, H, p) : GrpPC, GrpPC, RngIntElt -> BoolElt

Given p-groups G and H, decides if $G \cong H$. This intrinsic provides one optional parameters: `Constructive`. If G and H have genus 1 or 2 and exponent p, then it takes advantage of this structure to decide isomorphism with much more efficient algorithms. For this case, the algorithm defaults to the Pfaffian method at the moment. Eventually, the adjoint-tensor method will be incorporated. For groups of genus greater than 2 or exponent greater than p, the algorithm defaults to the standard algorithm in MAGMA.

Constructive: The default is `false`. When set to `true` and $G \cong H$, the algorithm returns an isomorphism from G to H.

IsPseudoIsometric(F, G) : SeqEnum[AlgMatElt], SeqEnum[AlgMatElt] -> BoolElt

Given a system of forms F and G over $\mathbb{F}_q$, decides if F is pseudo-isometric to G. This intrinsic provides one optional parameters: `Constructive`. If F and G are genus 1 or 2 over their centroids, then the algorithm uses this structure to decide pseudo-isometry with much more efficient algorithms. For this case, the algorithm defaults to the Pfaffian method at the moment. Eventually, the adjoint-tensor method will be incorporated. For systems forms with genus greater than 2, for nonalternating systems, or when the characteristic is 2, the algorithm uses brute force.

Constructive: The default is `false`. When set to `true` and F is pseudo-isometric to G, the algorithm returns a pseudo-isometry from F to G.

1. Examples

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

```
> G,H;
GrpPC : G of order 729 = 3^6
PC-Relations:
    G.2^G.1 = G.2 * G.5,
    G.3^G.1 = G.3 * G.6,
    G.3^G.2 = G.3 * G.5,
    G.4^G.1 = G.4 * G.5 * G.6,
    G.4^G.2 = G.4 * G.6^2,
    G.4^G.3 = G.4 * G.5^2 * G.6

GrpPC : H of order 729 = 3^6
PC-Relations:
    H.2^H.1 = H.2 * H.5,
    H.3^H.1 = H.3 * H.5,
    H.3^H.2 = H.3 * H.6,
    H.4^H.1 = H.4 * H.5^2,
    H.4^H.2 = H.4 * H.5 * H.6
> SGIsIsomorphic(G,H,3);
```

```
true
> SGIsIsomorphic(G,H,3 : Constructive:=true);
true Homomorphism of GrpPC : G into GrpPC : H induced by
    G.1 |--> H.2^2 * H.3 * H.4^2
    G.2 |--> H.1 * H.2 * H.3
    G.3 |--> H.1^2 * H.2^2 * H.3^2 * H.4^2
    G.4 |--> H.1 * H.2 * H.4^2
    G.5 |--> H.6^2
    G.6 |--> H.5 * H.6^2
```

```
> F,G;
[
    [0 4 2 1 1 3]
    [1 0 3 3 3 1]
    [3 2 0 2 2 0]
    [4 2 3 0 4 1]
    [4 2 3 1 0 3]
    [2 4 0 4 2 0],

    [0 3 0 1 0 2]
    [2 0 2 3 4 1]
    [0 3 0 0 0 2]
    [4 2 0 0 3 4]
    [0 1 0 2 0 0]
    [3 4 3 1 0 0]
]
[
    [0 0 3 4 3 2]
    [0 0 2 0 4 4]
    [2 3 0 2 0 3]
    [1 0 3 0 3 1]
    [2 1 0 2 0 0]
    [3 1 2 4 0 0],

    [0 1 0 3 3 3]
    [4 0 4 0 3 0]
    [0 1 0 1 2 4]
    [2 0 4 0 2 1]
    [2 2 3 3 0 2]
    [2 0 1 4 3 0]
]
> IsPseudoIsometric(F,G);
true
> IsPseudoIsometric(F,G : Constructive:=true);
true
[3 2 0 1 4 4 0 0]
[0 0 2 4 2 1 0 0]
[3 3 2 2 3 3 0 0]
[0 4 2 4 2 3 0 0]
[1 4 0 0 2 0 0 0]
```

```
[0 3 1 3 0 1 0 0]
[0 0 0 0 0 0 2 2]
[0 0 0 0 0 0 1 0]
```

CHAPTER 4

Miscellaneous

`Genus( G, p ) : GrpPC, RngIntElt -> RngIntElt`

Given a p-group G, returns the genus of G over $\mathbb{Z}_p$.

`SGRandomGroup( p, n, g ) : RngIntElt, RngIntElt, RngIntElt -> GrpPC`

Given a prime p and natural numbers $n, g \geq 1$, return a random p-group G with $|G| = p^{n+g}$ with genus g over $\mathbb{Z}_p$ and exponent p.

`Genus2RandomGroupWithBlocks( p, d ) : RngIntElt, SeqEnum[RngIntElt] -> GrpPC`

Given a prime p and a sequence d of integers ≥ 1, return a random genus 2 p-group G with block structure given by d. An entry of 1 will add a dimension to the radical (ie center).

`Genus2Signature( G, p ) : GrpPC, RngIntElt -> Rec`
`Genus2Signature( F ) : SeqEnum[AlgMatElt] -> Rec`

Given either a genus 2 p-group G with $p > 2$ or a pair of forms over $\mathbb{F}_q$ (with odd characteristic), returns the genus 2 signature.

`EquivalentSignatures( S1, S2 ) : Rec, Rec -> BoolElt`

Given a pair of genus 2 signatures $S1$ and $S2$, decide if they are equivalent.

1. Examples

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

```
> G;
GrpPC : G of order 729 = 3^6
PC-Relations:
    G.2^G.1 = G.2 * G.5,
    G.3^G.1 = G.3 * G.6,
    G.3^G.2 = G.3 * G.5^2,
    G.4^G.1 = G.4 * G.6^2,
    G.4^G.2 = G.4 * G.5^2,
    G.4^G.3 = G.4 * G.5 * G.6^2
> Genus(G,3);
2
> H;
GrpPC : H of order 2187 = 3^7
PC-Relations:
    H.2^H.1 = H.2 * H.5,
    H.3^H.1 = H.3 * H.5,
    H.3^H.2 = H.3 * H.6,
    H.4^H.3 = H.4 * H.7
> Genus(H,3);
3
```

```
> SGRandomGroup(5,4,2);
GrpPC of order 15625 = 5^6
PC-Relations:
    $.2^$.1 = $.2 * $.5,
    $.3^$.1 = $.3 * $.6,
    $.3^$.2 = $.3 * $.6^4,
    $.4^$.1 = $.4 * $.5^2 * $.6,
    $.4^$.2 = $.4 * $.5^2 * $.6,
    $.4^$.3 = $.4 * $.5^4 * $.6^3
> SGRandomGroup(5,8,1);
GrpPC of order 1953125 = 5^9
PC-Relations:
    $.2^$.1 = $.2 * $.9,
    $.3^$.1 = $.3 * $.9^2,
    $.3^$.2 = $.3 * $.9^2,
    $.4^$.1 = $.4 * $.9^3,
    $.5^$.2 = $.5 * $.9^2,
    $.5^$.3 = $.5 * $.9^4,
    $.5^$.4 = $.5 * $.9^3,
    $.6^$.1 = $.6 * $.9,
    $.6^$.2 = $.6 * $.9^2,
    $.6^$.3 = $.6 * $.9^4,
    $.6^$.4 = $.6 * $.9^3,
    $.6^$.5 = $.6 * $.9^3,
    $.7^$.1 = $.7 * $.9,
    $.7^$.2 = $.7 * $.9,
    $.7^$.3 = $.7 * $.9^2,
    $.7^$.4 = $.7 * $.9^3,
    $.8^$.1 = $.8 * $.9,
    $.8^$.2 = $.8 * $.9^3,
    $.8^$.3 = $.8 * $.9^4,
    $.8^$.4 = $.8 * $.9^2,
    $.8^$.5 = $.8 * $.9,
    $.8^$.6 = $.8 * $.9
```

```
> G := Genus2RandomGroupWithBlocks(3,[2,4]);
> G;
GrpPC : G of order 6561 = 3^8
PC-Relations:
    G.2^G.1 = G.2 * G.7,
    G.3^G.1 = G.3 * G.8,
    G.3^G.2 = G.3 * G.7^2 * G.8,
    G.4^G.1 = G.4 * G.8^2,
    G.4^G.2 = G.4 * G.8,
    G.4^G.3 = G.4 * G.7^2,
    G.5^G.1 = G.5 * G.7 * G.8^2,
    G.5^G.2 = G.5 * G.7,
    G.5^G.3 = G.5 * G.7^2,
    G.5^G.4 = G.5 * G.7^2,
```

```
    G.6^G.1 = G.6 * G.8^2,
    G.6^G.3 = G.6 * G.8,
    G.6^G.4 = G.6 * G.7^2,
    G.6^G.5 = G.6 * G.7 * G.8^2
> sig := Genus2Signature(G,3);
> sig`flatdims;
[]
> sig`pfaffians;
[
    $.2,
    $.1^2 + 2*$.1*$.2 + 2*$.2^2
]
```

```
> G := Genus2RandomGroupWithBlocks(3,[1,1,3,3]);
> G;
GrpPC : G of order 59049 = 3^10
PC-Relations:
    G.2^G.1 = G.2 * G.9,
    G.3^G.2 = G.3 * G.10,
    G.4^G.2 = G.4 * G.9^2,
    G.4^G.3 = G.4 * G.10^2,
    G.5^G.1 = G.5 * G.10,
    G.5^G.2 = G.5 * G.10,
    G.5^G.3 = G.5 * G.9^2 * G.10^2,
    G.5^G.4 = G.5 * G.10^2,
    G.6^G.1 = G.6 * G.9^2,
    G.6^G.2 = G.6 * G.10^2,
    G.6^G.3 = G.6 * G.9,
    G.6^G.4 = G.6 * G.9^2 * G.10^2,
    G.6^G.5 = G.6 * G.10^2,
    G.7^G.1 = G.7 * G.10^2,
    G.7^G.2 = G.7 * G.10^2,
    G.7^G.3 = G.7 * G.9^2 * G.10^2,
    G.7^G.4 = G.7 * G.10,
    G.7^G.5 = G.7 * G.9 * G.10,
    G.7^G.6 = G.7 * G.9^2 * G.10,
    G.8^G.1 = G.8 * G.9,
    G.8^G.2 = G.8 * G.9^2,
    G.8^G.3 = G.8 * G.10,
    G.8^G.4 = G.8 * G.9,
    G.8^G.6 = G.8 * G.9 * G.10^2
> sig := Genus2Signature(G,3);
> sig`flatdims;
[ 3, 3 ]
> sig`pfaffians;
[]
> Center(G);
GrpPC of order 81 = 3^4
PC-Relations:
    $.1^3 = Id($),
```

```
    $.2^3 = Id($),
    $.3^3 = Id($),
    $.4^3 = Id($)
```

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, DOI 10.1006/jsco.1996.0125. 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