Tame Genus Package

Joshua Maglione

Univeristät Bielefeld
jmaglione@math.uni-bielefeld.de

Peter A. Brooksbank

Bucknell University
pbrooksb@bucknell.edu

James B. Wilson

Colorado State University
James.Wilson@ColoState.Edu

Version 2.0
August 20, 2020

Contents

Chapter 1. Introduction 1
1.1. Citing TameGenus 1
1.2. Verbose Printing and Version 1

Chapter 2. Constructors 3
2.1. Direct indecomposability 5
2.2. Genus 6

Chapter 3. Isomorphism 11

Chapter 4. Automorphism groups 15

Chapter 5. Canonical Labeling 19

Bibliography 23

Intrinsics 25

CHAPTER 1

Introduction

The goal of this package is to provide MAGMA [**BCP**] with high performance isomorphism tests for groups with tame genus. A group has tame genus if its genus is less than 3. We include automorphism group constructors, along with canonical signatures. For definitions, theorems, and desriptions of algorithms, the reader should consult [**BMW**]. When appropriate, intrinsics for groups also work for tensors (`TenSpcElt`). This package includes a number of intrinsics for automorphism and isomorphism computations, canonical labeling, and constructing random groups of genus 1 or 2 of exponent $\leq p^2$. The index at the end of the documentation records all intrinsics provided by TameGenus.

The latest versions of `TensorSpace` [**MW2**], `StarAlge` [**BW2**], and `Sylver` [**MW1**] are required to use this package. The URLs are found in the bibliography or in the package `README.md`. Linux users can use the `install.sh` or `update.sh` bash files to install or update dependencies. Note that the `install.sh` will check for the `.magmarc` file in the home directory.

1.1. Citing TameGenus

To cite the TameGenus package, please use the following

Joshua Maglione, Peter A. Brooksbank, and James B. Wilson, *TameGenus*, version 2.0, GitHub, 2019. (https://github.com/thetensor-space/TameGenus).

For AMSRefs:

```
\bib{TameGenusPkg}{misc}{
   author={Maglione, Joshua},
   author={Brooksbank, Peter A.},
   author={Wilson, James B.},
   title={TameGenus},
   publisher={GitHub},
   year={2019},
   edition={version 2.0},
   note={\texttt{https://github.com/thetensor-space/TameGenus}},
}
```

1.2. Verbose Printing and Version

We have included intrinsics to allow for verbose printing.

`SetVerbose(MonStgElt, RngIntElt) : ->`

`SetVerbose` is a built in Magma function, but with this package, this intrinsic accepts the string `"TameGenus"` and an integer in $\{0, 1, 2\}$. Verbose printing level 0 turns off all printing. Level 1 will print out what MAGMA is doing in the TameGenus algorithms, and level 2 will print timings and extra structural information.

Example 1.1. `VerbosePrinting`

We demonstrate the verbose printing by constructing a random genus 2 group of order 3^{20+2}.

```
> G := RandomGenus2Group(3, [4, 6, 10]);
> #G eq 3^(4 + 6 + 10 + 2);
```

```
true
> Genus(G);
2
```

With the verbose printing, we can see how long each part of the algorithm takes and the information it obtains about the input.

```
> SetVerbose("TameGenus", 1);
> A := TGAutomorphismGroup(G);
Extracting the p-central tensor and computing pseudo-isometries.
Checking the radicals.
        dim(Rad_V) = 0
        dim(Rad_W) = 0
Writing tensor over its centroid.
Tensor has genus 2.
Computing the adjoint algebra.
        dim(Adj) = 40
Decomposing tensor into flat and sloped subtensors.
        Block dims = [ 4, 6, 10 ]
Number of sloped blocks to lift: 3.
Field is small enough, applying Pfaffian method.
Number of flat blocks to lift: 0.
Constructing the isometry group.
   G
   |   Sp ( 2 , 3 ^ 2 )
   *
   |   Sp ( 2 , 3 ^ 3 )
   *
   |   Sp ( 2 , 3 ^ 5 )
   *
   |   3 ^ 0    (unipotent radical)
   1
Constructing automorphism group from pseudo-isometries.
```

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

`TameGenusVersion() : -> MonStgElt`

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

Example 1.2. `Version`

We quickly verify that we have the latest version of TameGenus.

```
> TameGenusVersion();
2.0
```

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.

```
TGRandomGroup(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 $\mathbb{F}_p$, containing $\mathbb{F}_q$. If $\circ : V \times V \rightarrowtail W$ is the commutation tensor of G, then $\dim_{\mathcal{C}(\circ)}(V) \leq n$ and $\dim_{\mathcal{C}(\circ)}(W) \leq g$, where $\mathcal{C}(\circ)$ is the centroid of $\circ$. The centroid may strictly contain $\mathbb{F}_q$, in which case the dimensions of V and W divide n and g, respectively. It is also possible that the returned group is not directly indecomposable; this may cause other intrinsics to reject such output. The algorithm is based on the Universal Coefficients Theorem, see [**LGM**, Chapter 9] for the statement and proof. There is one optional parameter: `Exponentp`.

`Exponentp`**:** The default is `true`. Set to `false` if returned group need not have exponent p.

Example 2.1. `RandomGenusGroups`

We construct a random group using the `TGRandomGroup` constructor. Specifically, we create a group G with exponent 5 such that $\circ_G$ has centroid containing $K = \mathbb{F}_{5^7}$ with genus at most 4 and where $\dim_K(G/\Phi(G)) = 8$. First, we create the group.

```
> G := TGRandomGroup(5^7, 8, 4);
> #G eq 5^(7*(8 + 4));
true
```

Now we check properties of the group we created. We expect its genus to be at most 4.

```
> Genus(G);
4
```

We verify that the commutator tensor (coming from the p-central series) has a centroid containing the field of order 5^7.

```
> t := pCentralTensor(G);
> t;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 56 over GF(5)
U1 : Full Vector space of degree 56 over GF(5)
U0 : Full Vector space of degree 28 over GF(5)
> C := Centroid(t);
> Dimension(C);
7
> IsSimple(C);
true
```

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

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

`Exponentp`**:** The default is `true`. Set to `false` if returned group need not have exponent p.

Example 2.2. `PrescribedBlocks`

Unlike the `TGRandomGroup` constructor, if we want to construct a genus 2 group with prescribed block dimensions, we can do this with `RandomGenus2Group`. If we want an elementary abelian direct factor, we can append a sequence of 1s whose sum is the rank of such a factor. We construct a genus 2 group G with an indecomposable blocks of dimensions 10, 7, 4, 4, and 1. Therefore, $G \cong H \times \mathbb{Z}/(p)$, where H is direct indecomposable and of order 7^{25+2}.

```
> G := RandomGenus2Group(7, [10, 7, 4, 4, 1]);
> #G eq 7^(10 + 7 + 4 + 4 + 1 + 2);
true
```

We can see these blocks with the genus 2 signature of the group.

```
> S := TGSignature(G);
> S;
[* <7, 25, 2>, <1, 0>,
    [ 7 ],
    [*
        [ 1, 5, 5 ],
        [ 1, 4, 6 ],
        [ 1, 5, 1, 3, 6, 6 ]
    *]
*]
```

To learn how to read the above data, see the documentation for `TGSignature`. This shows that $\circ_G$ has a 1-dimensional radical and is full. Moreover, it is $\perp$-decomposable as a flat 7-dimensional block with three sloped blocks. The corresponding Pfaffians (unique up to $\mathrm{PGL}(2,7)$) are

$$x^2 + 5xy + 5y^2, \qquad x^2 + 4xy + 6y^2, \qquad x^5 + 5x^4y + x^3y^2 + 3x^2y^3 + 6xy^4 + 6y^5.$$

`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 $m(2d + r)$-generated. There is one optional parameter: `Exponentp`.

`Exponentp`**:** The default is `true`. Set to `false` if returned group need not have exponent p.

Example 2.3. `Heisenbergs`

The classic Heisenberg group over a field is an example of a genus 1 group, and one can generalize this construction in a number of ways. We will create the following matrix group, as a polycyclic group, over $\mathbb{F}_{81}$:

$$G = \left\{ \begin{bmatrix} 1 & * & * & * \\ & 1 & 0 & * \\ & & 1 & * \\ & & & 1 \end{bmatrix} \right\}$$

```
> G := RandomGenus1Group(81, 2, 0);
> #G eq 81^(2*2 + 1);
true
```

We will verify that that G is genus 1, but first note that the derived subgroup is isomorphic to $(\mathbb{Z}/(3))^4$.

```
> D := DerivedSubgroup(G);
> IsElementaryAbelian(D);
true
> #D eq 3^4;
true
```

Although the underlying field $\mathbb{F}_{81}$ is hidden within the relations of the polycyclic presentation, we can still determine it and compute the genus over this larger field.

```
> Genus(G);
1
```

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

Given either a univariate or homogeneous multivariate polynomial in x (and y) over $\mathbb{F}_q$, it returns a group G whose commutator $\mathbb{F}_q$-tensor has a Pfaffian equivalent to f.

Example 2.4. `Pfaffians`

We take the constructor `RandomGenus2Group` and specialize to sloped genus 2 groups. We will construct the genus 2 group G whose (univariate) Pfaffian is equivalent to $f(x) = x^3(x-1)^2(x-2)$ in $\mathbb{F}_9[x]$. Therefore, we expect $|G| = 9^{2\cdot 6+2}$.

```
> P<x> := PolynomialRing(GF(9));
> f := x^3*(x-1)^2*(x-2);
> G := Genus2Group(f);
> #G eq 9^(2*6 + 2);
true
```

The genus 2 signature will reveal the correct block structure. All six $\perp$-indecomposable subspaces will be 2-dimensional, but they will partition similar to the splitting behavoir of the polynomial f.

```
> S := TGSignature(G);
> S[4];
[*
    [ 1, 1 ],
    [ 1, 0, 0 ],
    [ 0, 0, 0, 1 ]
*]
```

This shows that homogenization of f is equivalent to the polynomial $g(x, y) = x^2y^3(x + y)$.

2.1. Direct indecomposability

The definition of genus requires knowing the (direct) indecomposable factors of a group G. Although we do not have intrinsics to produce a list of subgroups that are indecomposable, we have an efficient test to decide if the given nilpotent group is indecomposable. There is an analog for tensors as well. These are based off of algorithms from [**W**].

`IsIndecomposable(G) : GrpPC -> BoolElt`
`IsIndecomposable(t) : TenSpcElt -> BoolElt`

Given either a group G or a tensor t, decide if the given object is indecomposable.

Warning: We make no claim that the intrinsics in this package will accept input that return `false` from `IsIndecomposable`. There are indeed cases where the group or tensor is not indecomposable (e.g. nontrivial radical) where all the intrinsics will work fine.

Example 2.5. `Decomposable`

We can create a decomposable genus 2 group from the above constructors. For example, the group G with the Pfaffian xy will be decomposable.

```
> P<x, y> := PolynomialRing(GF(7), 2);
> f := x*y;
> G := Genus2Group(f);
> G;
GrpPC : G of order 117649 = 7^6
PC-Relations:
G.3^G.1 = G.3 * G.5,
G.4^G.2 = G.4 * G.6
> IsIndecomposable(G);
false
```

This example also shows that `TGRandomGroup` may return a group which is directly decomposable.

Example 2.6. `DirectNotCentral`

Note that the commutator tensor of a group can be directly indecomposable while still having a nontrivial $\perp$-decomposition. We create the directly indecomposable group G which is a central product of two genus 2 groups.

```
> G := RandomGenus2Group(5, [3, 4]);
> #G eq 5^(3 + 4 + 2);
true
> IsIndecomposable(G);
true
```

We can observe the central factors from the genus 2 signature of the group.

```
> TGSignature(G);
[* <5, 7, 2>, <0, 0>,
    [ 3 ],
    [*
        [ 1, 0, 2 ]
    *]
*]
```

2.2. Genus

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

Given an indecomposable p-group G, it returns the genus of G. That is, the rank of $[G, G]$ over the centroid. For an indecomposable tensor $t : V \times V \rightarrowtail W$, it returns the rank of W over its centroid.

Example 2.7. `Genus`

We will determine the genus of some groups in the Small Group library. First, let G be the group of order 3^6 with ID number 440. Its derived subgroup is isomorphic to $(\mathbb{Z}/(3))^2$, and G is genus 2.

```
> G := SmallGroup(3^6, 440);
> #DerivedSubgroup(G) eq 3^2;
true
> IsElementaryAbelian(DerivedSubgroup(G));
true
```

```
> Genus(G);
2
```

Let H be the group of order 3^6 with ID number 469. Its derived subgroup is isomorphic to $(\mathbb{Z}/(3))^2$ also, but H is genus 1.

```
> H := SmallGroup(3^6, 469);
> #DerivedSubgroup(H) eq 3^2;
true
> IsElementaryAbelian(DerivedSubgroup(H));
true
> Genus(H);
1
```

We can see this difference in multiple ways. If we extract the commutator tensor from both groups, one will be bilinear over a quadratic field extension where the other is not. The algebra that records this is the centroid of a bilinear map.

```
> s := pCentralTensor(G);
> t := pCentralTensor(H);
> C_G := Centroid(s);
> C_H := Centroid(t);
> C_G;
Matrix Algebra of degree 6 with 2 generators over GF(3)
> C_H;
Matrix Algebra of degree 6 with 2 generators over GF(3)
```

The centroid associated to the tensor s is 2-dimensional but with a nontrivial Jacobson radical. The centroid associated to the tensor t is 2-dimensional and a simple $\mathbb{F}_3$-algebra.

```
> Dimension(C_G);
2
> Dimension(C_H);
2
> WedderburnDecomposition(C_G);
Matrix Algebra of degree 6 with 1 generator over GF(3)
Matrix Algebra of degree 6 with 1 generator over GF(3)
> IsSimple(C_H);
true
```

`IsTameGenusGroup(G) : Group -> BoolElt`

Decides if the functions in the TameGenus package can be applied to the given group G.

`IsTameGenusTensor(t) : TenSpcElt -> BoolElt`

Decides if the functions in the TameGenus package can be applied to the given tensor t.

Example 2.8. `NonExample`

We create a group that is not covered by the TameGenus package. From the Genus example, see Example 2.7, the small groups of order 3^6 with IDs 440 and 469 both have tame genus: 2 and 1 respectively. The direct product of the two groups has genus 2, but because the group is not directly decomposable, the current implementation of TameGenus cannot handle it.

```
> H1 := SmallGroup(3^6, 440);
> H2 := SmallGroup(3^6, 469);
> G := DirectProduct(H1, H2);
> G;
GrpPC : G of order 531441 = 3^12
```

```
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.8^G.7 = G.8 * G.11,
    G.9^G.7 = G.9 * G.12,
    G.9^G.8 = G.9 * G.11^2 * G.12,
    G.10^G.7 = G.10 * G.11 * G.12,
    G.10^G.8 = G.10 * G.11
> IsTameGenusGroup(G);
false
```

Example 2.9. `AllTheSmallGroups`

We can use the `IsTameGenusGroup` and `IsTameGenusTensor` to filter large databases down to the ones covered by TameGenus. We do that here with the Small Group library for groups of order 3^6, and we find there are seven such groups.

```
> G2_SG := [*G : G in SmallGroups(3^6) | IsTameGenusGroup(G)*];
> #G2_SG;
7
```

We further filter these groups based on their genus, and we find there are three groups of positive genus (and one abelian group).

```
> G1 := [*G : G in TG_SG | Genus(G) eq 1*];
> G2 := [*G : G in TG_SG | Genus(G) eq 2*];
> #G1;
3
> #G2;
3
```

Now we determine the tame genus signatures to show that all these groups yield different signatures. We first consider the genus 1 groups.

```
> for G in G1 do
for>     TGSignature(G);
for> end for;
[* <9, 2, 1>, <0, 0>, [* *], [* *] *]
[* <3, 2, 1>, <3, 0>, [* *], [* *] *]
[* <3, 4, 1>, <1, 0>, [* *], [* *] *]
```

The genus 2 groups have the following signatures.

```
> for G in G2 do
for>     TGSignature(G);
for> end for;
[* <3, 3, 2>, <1, 0>,
    [ 3 ],
    [* *]
*]
[* <3, 4, 2>, <0, 0>,
    [],
    [*
        [ 0, 0, 1 ]
    *]
```

```
*]
[* <3, 4, 2>, <0, 0>,
    [],
    [*
        [ 1, 0 ],
        [ 0, 1 ]
    *]
*]
```

CHAPTER 3

Isomorphism

We have two intrinsics for deciding isomorphism: one for groups and one for tensors.

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

Given class ≤ 2, exponent p, indecomposable 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 $p = 2$. When the groups are decomposable, the intrinsic might decide isomorphism (e.g. when the centers properly contain the derived group), or it might return an error. In particular, an incorrect decision should not occur. 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 [**MW2**] for details. The default is `true`, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If this step should be avoided, set to `false` to save some time.

`Constructive`: The default is `true`. Set to `false` if an explicit isomorphism is not needed.

`Method`: The default is 0, and any input from $\{0, 1, 2\}$ is acceptable. This will set the method for handling the sloped part of the tensor. If the adjoint-tensor method should be used, set to 1. If the Pfaffian method should be used, set to 2. The default will try to find the optimal method based on the input; this is not yet optimally tuned but is generally good.

Example 3.1. `IsomorphismTesting`

We will create an isomorphic pair of groups in two different ways, and then we will construct an isomorphism between the two. This example can be seen as verifying Theorem 1.1 of [**LM**]. We construct groups G and H from two random irreducible cubic polynomials in $\mathbb{F}_{67}[x]$.

```
> f := RandomIrreduciblePolynomial(GF(67), 3);
> g := RandomIrreduciblePolynomial(GF(67), 3);
> f, g;
$.1^3 + 36*$.1^2 + 26*$.1 + 33
$.1^3 + 57*$.1^2 + 24*$.1 + 65
> G := Genus2Group(f);
> H := Genus2Group(g);
```

The default method, in this case, will run the adjoint-tensor method. We can see it running if we turn on verbose printing.

```
> SetVerbose("TameGenus", 1);
> isomorphic, phi := TGIsIsomorphic(G, H);
Extracting p-central tensors and deciding pseudo-isometry.
Checking the radicals.
        dim(Rad_V) = 0
        dim(Rad_W) = 0
Checking the radicals.
        dim(Rad_V) = 0
        dim(Rad_W) = 0
```

```
Writing tensor over its centroid.

Writing tensor over its centroid.

Computing the adjoint algebra.
        dim(Adj_s) = 12
        dim(Adj_t) = 12

Computing the adjoint algebra.

Genus 2 case.

Decomposing tensors into flat and sloped subtensors.
        Block dims = [ 6 ]

PGammaL is larger than symmetric group, applying adjoint-tensor method.
```

We see that the two groups are isomorphic, and ϕ is such an isomorphism.

```
> isomorphic;
true
> phi;
Homomorphism of GrpPC : G into GrpPC : H induced by
    G.1 |--> H.1 * H.2^41 * H.3^30 * H.4^58 * H.5^43 * H.6^30
    G.2 |--> H.1^53 * H.2^4 * H.3^34 * H.4^54 * H.5^6 * H.6^25
    G.3 |--> H.1^5 * H.2^7 * H.3^49 * H.4^52 * H.5^21 * H.6^7
    G.4 |--> H.1^30 * H.2^4 * H.3^17 * H.4^53 * H.5^40 * H.6^9
    G.5 |--> H.1^14 * H.2^21 * H.3^30 * H.4^24 * H.5^38 * H.6^34
    G.6 |--> H.1^47 * H.2^46 * H.3^66 * H.4^65 * H.5^30 * H.6^30
    G.7 |--> H.7^10 * H.8^18
    G.8 |--> H.7^54 * H.8^12
```

If we want to run the Pfaffian test, we set the `Method` to 2. This method is considerably slower for these groups.

```
> isomorphic, phi2 := TGIsIsomorphic(G, H : Method := 2);
Extracting p-central tensors and deciding pseudo-isometry.
Checking the radicals.
        dim(Rad_V) = 0
        dim(Rad_W) = 0
Checking the radicals.
        dim(Rad_V) = 0
        dim(Rad_W) = 0

Writing tensor over its centroid.

Writing tensor over its centroid.

Computing the adjoint algebra.
        dim(Adj_s) = 12
        dim(Adj_t) = 12

Computing the adjoint algebra.

Genus 2 case.

Decomposing tensors into flat and sloped subtensors.
```

```
        Block dims = [ 6 ]

Method set to Pfaffian.
> phi2;
Homomorphism of GrpPC : G into GrpPC : H induced by
    G.1 |--> H.1^47 * H.2^54 * H.3^54
    G.2 |--> H.1^63 * H.2^33 * H.3^15
    G.3 |--> H.1^26 * H.2^3 * H.3^60
    G.4 |--> H.4^44 * H.5^61 * H.6^16
    G.5 |--> H.4^66 * H.5^26 * H.6^36
    G.6 |--> H.4^57 * H.5^6 * H.6^30
    G.7 |--> H.7^62 * H.8^42
    G.8 |--> H.7^18 * H.8^58
```

```
TGIsPseudoIsometric(s, t : parameters) : TenSpcElt, TenSpcElt -> BoolElt, Hmtp
    Cent : BoolElt : true
    Constructive : BoolElt : true
    Method : RngIntElt : 0
```

Given indecomposable, alternating tensors $s, t : V \times V \rightarrowtail W$, where V and W are $\mathbb{F}_q$-vector spaces, decide if s is pseudo-isometric to t. If so, a pseudo-isometry is also returned as a homotopism. Currently this does not work for even q. When the tensors are decomposable, the intrinsic might decide isomorphism (e.g. when the tensors have nontrivial radical), or it might return an error. In particular, an incorrect decision should not occur. 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 [**MW2**] for details. The default is `true`, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If this step should be avoided, set to `false` to save some time.

`Constructive`**:** The default is `true`. Set to `false` if an explicit isomorphism is not needed.

`Method`**:** The default is 0, and any input from $\{0, 1, 2\}$ is acceptable. This will set the method for handling the sloped part of the tensor. If the adjoint-tensor method should be used, set to 1. If the Pfaffian method should be used, set to 2. The default will try to find the optimal method based on the input; this is not yet optimally tuned but is generally good.

Example 3.2. `PseudoIsometries`

A pair of alternating matrices determines a tensor of tame genus. We will construct a pair of pseudo-isometric tensors by applying a change of basis isotopism. This is essentially how Experiment A from [**BMW**] was conducted. Generating random 100×100 matrices over $\mathbb{F}_3$, we build two tensors $s, t : \mathbb{F}_3^{50} \times \mathbb{F}_3^{50} \rightarrowtail \mathbb{F}_3^2$.

```
> M := RandomMatrix(GF(3), 50, 50);
> N := RandomMatrix(GF(3), 50, 50);
> Forms1 := [M - Transpose(M), N - Transpose(N)];
> s := Tensor(Forms1, 2, 1);
> s;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 50 over GF(3)
U1 : Full Vector space of degree 50 over GF(3)
U0 : Full Vector space of degree 2 over GF(3)
> IsAlternating(s);
true
```

Now we construct a random invertible co-homotopism φ such that $\varphi_2 \in \mathrm{GL}(50, 3)$ and $\varphi_0 \in \mathrm{GL}(2, 3)$. We define a new and pseudo-isometric tensor $t = s^\varphi$ in this way.

```
> X := Random(GL(50, 3));
> Y := Random(GL(2, 3));
> Maps := [*X, X, Y*];
> H := Homotopism(Maps, CohomotopismCategory(3));
> t := s @ H;
> t;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 50 over GF(3)
U1 : Full Vector space of degree 50 over GF(3)
U0 : Full Vector space of degree 2 over GF(3)
> s eq t;
false
```

We verify that both s and t are pseudo-isometric, and since we do not need to write these tensors over their centroid, we avoid that step by setting `Cent` to `false`.

```
> pisometric, Phi := TGIsPseudoIsometric(s, t : Cent := false);
> pisometric;
true
> Phi.0;
[0 1]
[2 2]
```

CHAPTER 4

Automorphism groups

Like with isomorphisms, we have two distinct automorphism intrinsics.

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

Given an indecomposable p-group G, of class ≤ 2, exponent p, and genus ≤ 2, return $\mathrm{Aut}(G)$. This currently does not work with $p = 2$. When G is decomposable, the intrinsic might construct $\mathrm{Aut}(G)$ (e.g. when the center properly contains the derived group), or it might return an error. In particular, the wrong (or incomplete) automorphism group should not be returned. This intrinsic provides three optional parameters: `Cent`, `Method`, and `Mat`.

`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 [**MW2**] for details. The default is `true`, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If this step should be avoided, set to `false` to save some time.

`Method`: The default is 0, and any input from $\{0, 1, 2\}$ is acceptable. This will set the method for handling the sloped part of the tensor. If the adjoint-tensor method should be used, set to 1. If the Pfaffian method should be used, set to 2. The default will try to find the optimal method based on the input; this is not yet optimally tuned but is generally good.

`Mat`: The default is `false`. When `true`, the output will be a linear representation of the automorphism group. The row vectors of the matrix correspond to the polycyclic vectors from the polycyclic presentation.

Example 4.1. `FlatIndecomposable`

We build a flat indecomposable genus 2 p-group G and construct its automorphism group. We know that $\mathrm{Aut}(G)$ maps surjectivity onto $\mathrm{GL}(2, p)$, so we will verify that $|\mathrm{Aut}(G)|$ is divisible by $(p^2 - 1)(p^2 - p)$. This corresponds to the quotient of the pseudo-isometry group by the isometry group.

```
> p := 541;
> G := RandomGenus2Group(p, [29]);
> #G eq p^(29 + 2);
true
> A := TGAutomorphismGroup(G);
> IsDivisibleBy(#A, #GL(2, p));
true
```

The isometry group of the corresponding commutator tensor has an elementary abelian unipotent radical of rank 28, and so is isomorphic to $(\mathbb{Z}/(p))^{28} \rtimes \mathrm{GL}(1, p)$, as seen in [**BW1**]. Therefore, we can verify that we have constructed all automorphisms as

$$|\mathrm{Aut}(G)| = (p-1)p^{28} \cdot (p^2-1)(p^2-p) \cdot p^{29\cdot 2}.$$

```
> #A eq (p - 1)*p^28 * (p^2 - 1)*(p^2 - p) * p^(29*2);
true
```

```
TGPseudoIsometryGroup(t : parameters) : TenSpcElt -> GrpMat
    Cent : BoolElt : true
    Method : RngIntElt : 0
```

Given an indecomposable, alternating tensor $t : V \times V \rightarrowtail W$, where V and W are $\mathbb{F}_q$-vector spaces, return the pseudo-isometry group $\Psi\mathrm{Isom}(t) \leq \mathrm{GL}(V) \times \mathrm{GL}(W)$. This currently does not work for even q. When t is decomposable, the intrinsic might construct $\Psi\mathrm{Isom}(t)$ (e.g. when t has a nontrivial radical), or it might return an error. In particular, the wrong (or incomplete) pseudo-isometry group should not be returned. This intrinsic provides two optional parameters: `Cent` 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 [**MW2**] for details. The default is `true`, so by default, the algorithm computes the centroid and rewrites the tensor over the residue field. If this step should be avoided, set to `false` to save some time.

`Method`: The default is 0, and any input from $\{0, 1, 2\}$ is acceptable. This will set the method for handling the sloped part of the tensor. If the adjoint-tensor method should be used, set to 1. If the Pfaffian method should be used, set to 2. The default will try to find the optimal method based on the input; this is not yet optimally tuned but is generally good.

Example 4.2. Extensions

For K vector spaces V and W, we create a genus 2 alternating tensor $t : V \times V \rightarrowtail W$ such that t is bilinear over an extension field E/K and $\dim_E(W) = 2$. One way to do this is via the constructor `RandomGenus2Group`.

```
> G := RandomGenus2Group(3^2, [6]);
> t := pCentralTensor(G);
> t;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 12 over GF(3)
U1 : Full Vector space of degree 12 over GF(3)
U0 : Full Vector space of degree 4 over GF(3)
> Genus(t);
2
```

The Pfaffian of t is an irreducible cubic over $\mathbb{F}_9$, so computing its pseudo-isometry group may contain elements of $\mathrm{Gal}(\mathbb{F}_9)$.

```
> PI := TGPseudoIsometryGroup(t);
> Random(PI);
[0 1 0 0 0 2 1 1 0 2 1 0 0 0 0 0]
[2 0 0 0 1 0 0 2 1 0 1 2 0 0 0 0]
[1 1 0 0 0 1 1 1 0 0 2 1 0 0 0 0]
[0 2 0 0 2 0 0 2 0 0 1 1 0 0 0 0]
[1 2 2 0 0 2 2 2 1 0 0 0 0 0 0 0]
[2 2 2 1 1 0 0 1 1 2 0 0 0 0 0 0]
[2 2 2 0 1 0 1 0 2 2 1 2 0 0 0 0]
[0 1 2 1 1 2 1 2 0 1 2 2 0 0 0 0]
[2 0 1 0 2 2 2 1 1 1 0 0 0 0 0 0]
[2 1 1 2 0 1 1 1 0 2 0 0 0 0 0 0]
[0 2 0 1 2 2 1 1 0 1 1 0 0 0 0 0]
[1 0 2 0 0 1 0 2 2 0 1 2 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0]
[0 0 0 0 0 0 0 0 0 0 0 0 2 0 1 2]
[0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0]
> Factorization(#PI);
[ <2, 8>, <3, 7>, <5, 1>, <7, 1>, <13, 1>, <73, 1> ]
```

Indeed, the pseudo-isometry group may not be linear over the centroid but semi-linear. To see the discrepancy, we rewrite t over its centroid and determine its pseudo-isometry group. We see that $\Psi\mathrm{Isom}(t)$ is strictly semi-linear over $\mathbb{F}_9$, as there exists a Galois involution.

```
> s := TensorOverCentroid(t);
> s;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 6 over GF(3^2)
U1 : Full Vector space of degree 6 over GF(3^2)
U0 : Full Vector space of degree 2 over GF(3^2)
> PI_C := TGPseudoIsometryGroup(s);
> Random(PI_C);
[    1 $.1^7     1 $.1^2 $.1^2 $.1^5     0     0]
[  $.1     2 $.1^7 $.1^3 $.1^6 $.1^5     0     0]
[$.1^7 $.1^2 $.1^7     1 $.1^6 $.1^2     0     0]
[$.1^7 $.1^7 $.1^7 $.1^3 $.1^7     1     0     0]
[    2 $.1^6 $.1^3 $.1^2 $.1^3     0     0     0]
[$.1^7   $.1 $.1^7 $.1^6 $.1^6 $.1^7     0     0]
[    0     0     0     0     0     0     2 $.1^7]
[    0     0     0     0     0     0 $.1^3 $.1^5]
> Factorization(#PI_C);
[ <2, 7>, <3, 7>, <5, 1>, <7, 1>, <13, 1>, <73, 1> ]
```

CHAPTER 5

Canonical Labeling

TGSignature(G) : GrpPC -> List
TGSignature(t) : TenSpcElt -> List

Given an indecomposable group or tensor of tame genus, returns the canonical tame genus signature as a list. The list has four entries. For inputs with genus ≤ 1, the last two entries will be empty lists.

(1) The first is a triple of three integers (q, d, e) such that associated fully nondegenerate tensor of the (commutator) tensor has the form $\mathbb{F}_q^d \times \mathbb{F}_q^d \rightarrowtail \mathbb{F}_q^e$. Here, $\mathbb{F}_q$ is the Galois field of the residue field of the centroid, which is a local ring. Therefore, $\mathbb{F}_q = \mathrm{GF}(q)$.
(2) The second entry is a pair of integers (r, c), where r is the dimension of the radical of the (commutator) tensor and c is the codimension of the image in the codomain.
(3) The third entry is a sequence of odd integers corresponding to the dimensions of the flat indecomposable spaces.
(4) The fourth and last entry is a list sequences in $\mathbb{F}_q$ of possibly different lengths. A sequence $[c_1, \ldots, c_{d+1}]$ in the second entry corresponds to the coefficients of the homogeneous polynomial in x and y in degree d:

$$c_1 x^d + c_2 x^{d-1} y + c_3 x^{d-2} y^2 + \cdots + c_{d+1} y^d.$$

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. Therefore, isoclinism or pseudo-isometry is determined by (honest) equality of lists.

When the input is decomposable, the intrinsic might construct the signature (e.g. when the tensor has a nontrivial radical), or it might return an error. In particular, the wrong signature should not be returned.

Example 5.1. ManyBlocks

We construct a genus 2 group G with many sloped and flat blocks.

```
> blocks := [1, 1, 1, 2, 2, 3, 3, 5, 6];
> G := RandomGenus2Group(9, blocks);
> #G eq 9^(&+blocks + 2);
true
```

We expect the commutator tensor from G to have a 3-dimensional (over $\mathbb{F}_9$) radical, and to have data corresponding to the blocks above.

```
> TGSignature(G);
[* <9, 21, 2>, <6, 0>,
    [ 3, 3, 5 ],
    [*
        [ 0, 1 ],
        [ 1, 1 ],
        [ 1, 2, 2, 2 ]
    *]
*]
```

The first entry of this list tells us that the associated fully nondegenerate tensor of G has the frame $\mathbb{F}_9^{21} \times \mathbb{F}_9^{21} \rightarrowtail \mathbb{F}_9^2$. The second entry says that the commutator tensor from G has 6-dimensional radical over $\mathbb{F}_3$, the field for which the commutator tensor was initially defined over. In other words, $G \cong H \times \mathbb{F}_9^3$. The third entry

tells us that H is a central product of six indecomposable groups: three flats of orders 9^{3+2}, 9^{3+2}, and 9^{5+2}, and three sloped with Pfaffians in $\mathbb{F}_9[x, y]$ equivalent to

$$y, \qquad x + y, \qquad x^3 + 2x^2y + 2xy^2 + 2y^3.$$

Example 5.2. `MoreSmallGroups`

We run through all the small groups of order 3^7 and determine their tame genus signatures. There are 9310 such groups, and 10 of them are covered by TameGenus.

```
> TG_SG := SmallGroupProcess(3^7, IsTameGenusGroup);
>
> repeat
repeat>     G := Current(TG_SG);
repeat>     _, ID := CurrentLabel(TG_SG);
repeat>     print "Small group ID:", ID;
repeat>     TGSignature(G);
repeat>     Advance(~TG_SG);
repeat> until IsEmpty(TG_SG);
```

The loop yields the following output.

```
Small group ID: 9106
[* <3, 3, 2>, <2, 0>,
    [ 3 ],
    [* *]
*]
Small group ID: 9107
[* <3, 4, 2>, <1, 0>,
    [],
    [*
        [ 1, 0 ],
        [ 0, 1 ]
    *]
*]
Small group ID: 9108
[* <3, 4, 2>, <1, 0>,
    [],
    [*
        [ 0, 0, 1 ]
    *]
*]
Small group ID: 9109
[* <9, 2, 1>, <1, 0>, [* *], [* *] *]
Small group ID: 9110
[* <3, 5, 2>, <0, 0>,
    [ 3 ],
    [*
        [ 0, 1 ]
    *]
*]
Small group ID: 9111
[* <3, 5, 2>, <0, 0>,
    [ 5 ],
    [* *]
*]
```

```
Small group ID: 9302
[* <3, 2, 1>, <4, 0>, [* *], [* *] *]
Small group ID: 9305
[* <3, 4, 1>, <2, 0>, [* *], [* *] *]
Small group ID: 9308
[* <3, 6, 1>, <0, 0>, [* *], [* *] *]
Small group ID: 9310
[* <3, 7, 0>, <7, 0>, [* *], [* *] *]
```

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

[BW1] Peter A. Brooksbank and James B. Wilson, *Computing isometry groups of Hermitian maps*, Trans. Amer. Math. Soc. **364** (2012), no. 4, 1975–1996. MR2869196

[BW2] ———, *StarAlge*, GitHub, 2019. https://github.com/thetensor-space/StarAlge.

[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

[LM] Mark L. Lewis and Joshua Maglione, *Enumerating isoclinism classes of semi-extraspecial groups*. Proc. Edinb. Math. Soc. (2), to appear.

[MW1] Joshua Maglione and James B. Wilson, *Sylver*, GitHub, 2019. https://github.com/thetensor-space/Sylver.

[MW2] ———, *TensorSpace*, GitHub, 2019. Contributions from Peter A. Brooksbank, https://github.com/thetensor-space/TensorSpace.

[W] James B. Wilson, *Existence, algorithms, and asymptotics of direct product decompositions, I*, Groups Complex. Cryptol. **4** (2012), no. 1, 33–72. MR2921155

Intrinsics

Genus
  groups, 6
  tensors, 6
Genus2Group, 5

IsIndecomposable
  groups, 5
  tensors, 5
IsTameGenusGroup, 7
IsTameGenusTensor, 7

RandomGenus1Group, 4
RandomGenus2Group, 3

SetVerbose, 1

TameGenusVersion, 2
TGAutomorphismGroup, 15
TGIsIsomorphic, 11
TGIsPseudoIsometric, 13
TGPseudoIsometryGroup, 15
TGRandomGroup, 3
TGSignature
  groups, 19
  tensors, 19