

Twisted Conjugacy

Computation with twisted conjugacy classes

1.1.0

4 August 2020

Sam Tertooy

Sam Tertooy

Email: sam.tertooy@hotmail.com

Homepage: <https://www.kuleuven-kulak.be/~u0092325/>

Address: Wiskunde

KU Leuven Campus Kulak Kortrijk

Etienne Sabbelaan 53

8500 Kortrijk

Belgium

Copyright

© 2020 by Sam Tertooy

The TWISTEDCONJUGACY package is free software, it may be redistributed and/or modified under the terms and conditions of the [GNU Public License Version 2](#) or (at your option) any later version.

Acknowledgements

This documentation was created using the AUTODOC package. The algorithms in this package are based on [\[Fel00\]](#), [\[Rom16\]](#), [\[MW20\]](#), [\[DT20\]](#) and [\[Ter20\]](#).

Contents

1	Twisted Conjugacy	4
1.1	Twisted Conjugation Action	4
1.2	Reidemeister Classes	5
1.3	Reidemeister Spectra	6
1.4	Zeta Functions	7
2	Double Twisted Conjugacy	9
2.1	Double Twisted Conjugation Action	9
2.2	Reidemeister Coincidence Classes	10
3	Miscellaneous	12
3.1	Groups	12
3.2	Morphisms	13
	References	14
	Index	15

Chapter 1

Twisted Conjugacy

Please note that the functions in this chapter are implemented only for endomorphisms of finite groups and pcg-groups.

1.1 Twisted Conjugation Action

Let G be a group and $\varphi : G \rightarrow G$ an endomorphism. Then φ induces a (right) group action on G given by $G \times G \rightarrow G : (g, h) \mapsto g \cdot h = h^{-1}g\varphi(h)$. This group action is called *φ -twisted conjugation*, and induces an equivalence relation on the group. We say that $g_1, g_2 \in G$ are *φ -twisted conjugate*, denoted by $g_1 \sim_{\varphi} g_2$, if and only if there exists some element $h \in G$ such that $g_1 \cdot h = g_2$, or equivalently $g_1 = hg_2\varphi(h)^{-1}$.

1.1.1 TwistedConjugation (for IsGroupHomomorphism and IsEndoGeneralMapping)

▷ `TwistedConjugation(endo)` (operation)

Implements the twisted conjugation (right) group action induced by the endomorphism *endo*. This is the twisted conjugacy analogue of `OnPoints`.

1.1.2 IsTwistedConjugate (for IsGroupHomomorphism and IsEndoGeneralMapping, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse)

▷ `IsTwistedConjugate(endo, g1, g2)` (operation)

Tests whether the elements *g1* and *g2* are twisted conjugate under the twisted conjugacy action of the endomorphism *endo*. This is the twisted conjugacy analogue of `IsConjugate`. For polycyclic groups, this algorithm may fail if the group is not nilpotent-by-finite and the Reidemeister number of *endo* is infinite.

1.1.3 RepresentativeTwistedConjugation (for IsGroupHomomorphism and IsEndoGeneralMapping, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse)

▷ `RepresentativeTwistedConjugation(endo, g1, g2)` (operation)

Computes an element that maps g_1 to g_2 under the twisted conjugacy action of the endomorphism $endo$ and returns fail if no such element exists. This is the twisted conjugacy analogue of RepresentativeAction. For polycyclic groups, this algorithm may fail if the group is not nilpotent-by-finite and the Reidemeister number of $endo$ is infinite.

Example

```
gap> G := Group([ (3,4)(5,6), (1,2,3)(4,5,7) ]);;
gap> phi := GroupHomomorphismByImages( G, G, [ (2,7)(4,6), (1,4,5,6,7,2,3) ],
> [ (2,4)(6,7), (1,2,4,6,5,7,3) ] );;
gap> tc := TwistedConjugation( phi );;
gap> IsTwistedConjugate( phi, G.1, G.1^2 );
false
gap> g := RepresentativeTwistedConjugation( phi, G.1, G.2 );
(1,6,7,5)(3,4)
gap> tc( G.1, g ) = G.2;
true
```

1.2 Reidemeister Classes

The equivalence classes of the equivalence relation \sim_φ are called the *Reidemeister classes of φ* or the *φ -twisted conjugacy classes*. We denote the Reidemeister class of $g \in G$ by $[g]_\varphi$. The number of Reidemeister classes is called the Reidemeister number $R(\varphi)$ and is always a positive integer or infinity.

1.2.1 ReidemeisterClass (for IsGroupHomomorphism and IsEndoGeneralMapping, IsMultiplicativeElementWithInverse)

▷ ReidemeisterClass($endo$, g) (operation)
 ▷ TwistedConjugacyClass($endo$, g) (operation)

Creates the Reidemeister class of an endomorphism $endo$ of a group G with representative g . The following attributes and operations are available:

- Representative, which returns g ,
- GroupHomomorphismsOfReidemeisterClass, which returns a list containing $endo$ and the identity map on G (to be compatible with double twisted conjugacy classes),
- ActingDomain, which returns the group G ,
- FunctionAction, which returns the twisted conjugacy action of $endo$ on G ,
- Random, which returns a random element belonging to the Reidemeister class,
- \in, which can be used to test if an element belongs to the Reidemeister class - only guaranteed to work if the Reidemeister number of $endo$ is finite,
- AsList, which lists all elements in the Reidemeister class - only works for finite groups.
- Size, which gives the number of elements in the Reidemeister class - only works for finite groups.

This is the twisted conjugacy analogue of ConjugacyClass.

1.2.2 ReidemeisterClasses (for IsGroupHomomorphism and IsEndoGeneralMapping)

- ▷ `ReidemeisterClasses(endo)` (operation)
- ▷ `TwistedConjugacyClasses(endo)` (operation)

Returns a list containing the Reidemeister classes of *endo* if the Reidemeister number of *endo* is finite, and returns `fail` otherwise. It is guaranteed that the Reidemeister class of the identity is in the first position. This is the twisted conjugacy analogue of `ConjugacyClasses`.

1.2.3 ReidemeisterNumber (for IsGroupHomomorphism and IsEndoGeneralMapping)

- ▷ `ReidemeisterNumber(endo)` (operation)
- ▷ `NrTwistedConjugacyClasses(endo)` (operation)

Returns the Reidemeister number of *endo*, i.e. the number of Reidemeister classes. This is the twisted conjugacy analogue of `NrConjugacyClasses`.

Example

```
gap> tcc := ReidemeisterClass( phi, G.1 );
(3,4)(5,6)^G
gap> Representative( tcc );
(3,4)(5,6)
gap> GroupHomomorphismsOfReidemeisterClass( tcc );
[ [ (2,7)(4,6), (1,4,5,6,7,2,3) ] -> [ (2,4)(6,7), (1,2,4,6,5,7,3) ],
  IdentityMapping( Group([ (3,4)(5,6), (1,2,3)(4,5,7) ]) ) ]
gap> ActingDomain( tcc ) = G;
true
gap> FunctionAction( tcc )( G.1, g );
(1,2,3)(4,5,7)
gap> Random( tcc ) in tcc;
true
gap> List( tcc );
[ (3,4)(5,6), (1,3)(2,6), (1,6,7)(2,4,3), ... ]
gap> Size( tcc );
42
gap> ReidemeisterClasses( phi );
[ ()^G, (3,4)(5,6)^G, (3,6)(4,5)^G, (2,3,6)(4,7,5)^G ]
gap> NrTwistedConjugacyClasses( phi );
4
```

1.3 Reidemeister Spectra

The set of all Reidemeister numbers of automorphisms is called the *Reidemeister spectrum* and is denoted by $\text{Spec}_R(G)$, i.e.

$$\text{Spec}_R(G) := \{R(\varphi) \mid \varphi \in \text{Aut}(G)\}.$$

The set of all Reidemeister numbers of endomorphisms is called the *extended Reidemeister spectrum* and is denoted by $\text{ESpec}_R(G)$, i.e.

$$\text{ESpec}_R(G) := \{R(\varphi) \mid \varphi \in \text{End}(G)\}.$$

1.3.1 ReidemeisterSpectrum (for IsGroup)

▷ `ReidemeisterSpectrum(G)` (attribute)

Returns the Reidemeister spectrum of *G*.

1.3.2 ExtendedReidemeisterSpectrum (for IsGroup)

▷ `ExtendedReidemeisterSpectrum(G)` (attribute)

Returns the extended Reidemeister spectrum of *G*.

Example

```
gap> ReidemeisterSpectrum( G );
[ 4, 6 ]
gap> ExtendedReidemeisterSpectrum( G );
[ 1, 4, 6 ]
```

1.4 Zeta Functions

Let $\varphi : G \rightarrow G$ be an endomorphism such that $R(\varphi^n) < \infty$ for all $n \in \mathbb{N}$. Then the Reidemeister zeta function $R_\varphi(z)$ of φ is defined as

$$R_\varphi(z) := \exp \sum_{n=1}^{\infty} R(\varphi^n) \frac{z^n}{n}.$$

Please note that the functions below are only implemented for endomorphisms of finite groups.

1.4.1 ReidemeisterZeta (for IsGroupHomomorphism and IsEndoGeneralMapping)

▷ `ReidemeisterZeta(endo)` (operation)

Returns the Reidemeister zeta function of *endo*.

1.4.2 PrintReidemeisterZeta (for IsGroupHomomorphism and IsEndoGeneralMapping)

▷ `PrintReidemeisterZeta(endo)` (operation)

Returns a string describing the Reidemeister zeta function of *endo*.

1.4.3 ReidemeisterZetaCoefficients (for IsGroupHomomorphism and IsEndoGeneralMapping)

▷ `ReidemeisterZetaCoefficients(endo)` (attribute)

For a finite group, the sequence of Reidemeister numbers of the iterates of *endo*, i.e. the sequence $R(\text{endo})$, $R(\text{endo}^2)$, ..., is periodic (see [Fel00, Theorem 16]). This function returns a list containing the first period of this sequence.

Example

```
gap> zeta1 := ReidemeisterZeta( phi );;  
gap> zeta1( 10/3 );  
-729/218491  
gap> PrintReidemeisterZeta( phi );  
"( 1-z^1 )^-4 * ( 1-z^2 )^-1"  
gap> ReidemeisterZetaCoefficients( phi );  
[ 4, 6 ]
```


Chapter 2

Double Twisted Conjugacy

Please note that the functions in this chapter are implemented only for homomorphisms between finite groups or between pcg-groups.

2.1 Double Twisted Conjugation Action

Let G, H be groups and $\varphi, \psi : H \rightarrow G$ group homomorphisms. Then the pair (φ, ψ) induces a (right) group action on G given by $G \times H \rightarrow G : (g, h) \mapsto g \cdot h = \psi(h)^{-1} g \varphi(h)$. This group action is called (φ, ψ) -twisted conjugation, and induces an equivalence relation on the group. We say that $g_1, g_2 \in G$ are (φ, ψ) -twisted conjugate, denoted by $g_1 \sim_{\varphi, \psi} g_2$, if and only if there exists some element $h \in H$ such that $g_1 \cdot h = g_2$, or equivalently $g_1 = \psi(h) g_2 \varphi(h)^{-1}$.

2.1.1 TwistedConjugation (for IsGroupHomomorphism, IsGroupHomomorphism)

▷ `TwistedConjugation(hom1, hom2)` (operation)

Implements the twisted conjugation (right) group action induced by the pair of homomorphisms (`hom1, hom2`).

2.1.2 IsTwistedConjugate (for IsGroupHomomorphism, IsGroupHomomorphism, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse)

▷ `IsTwistedConjugate(hom1, hom2, g1, g2)` (operation)

Tests whether the elements $g1$ and $g2$ are double twisted conjugate under the twisted conjugacy action of the pair of homomorphisms (`hom1, hom2`). For polycyclic groups, this algorithm may fail if the range is not nilpotent-by-finite and the Reidemeister number of `hom1` and `hom2` is infinite.

2.1.3 RepresentativeTwistedConjugation (for IsGroupHomomorphism, IsGroupHomomorphism, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse)

▷ `RepresentativeTwistedConjugation(hom1, hom2, g1, g2)` (operation)

Computes an element that maps g_1 to g_2 under the twisted conjugacy action of the pair of homomorphisms $(hom1, hom2)$ and returns fail if no such element exists. For polycyclic groups, this algorithm may fail if the range is not nilpotent-by-finite and the Reidemeister number of $hom1$ and $hom2$ is infinite.

Example

```
gap> G := AlternatingGroup( 6 );;
gap> H := SymmetricGroup( 5 );;
gap> phi := GroupHomomorphismByImages( H, G, [ (1,2)(3,5,4), (2,3)(4,5) ],
> [ (1,2)(3,4), () ] );;
gap> psi := GroupHomomorphismByImages( H, G, [ (1,2)(3,5,4), (2,3)(4,5) ],
> [ (1,4)(3,6), () ] );;
gap> tc := TwistedConjugation( phi, psi );;
gap> g1 := (4,6,5);;
gap> g2 := (1,6,4,2)(3,5);;
gap> IsTwistedConjugate( psi, phi, g1, g2 );
false
gap> h := RepresentativeTwistedConjugation( phi, psi, g1, g2 );
(1,2)
gap> tc( g1, h ) = g2;
true
```

2.2 Reidemeister Coincidence Classes

The equivalence classes of the equivalence relation $\sim_{\varphi, \psi}$ are called the *Reidemeister coincidence classes of (φ, ψ)* or the *(φ, ψ) -twisted conjugacy classes*. We denote the Reidemeister class of $g \in G$ by $[g]_{\varphi, \psi}$. The number of Reidemeister coincidence classes is called the Reidemeister coincidence number $R(\varphi, \psi)$ and is always a positive integer or infinity.

2.2.1 ReidemeisterClass (for IsGroupHomomorphism, IsGroupHomomorphism, Is-MultiplicativeElementWithInverse)

▷ `ReidemeisterClass(hom1, hom2, g)` (operation)
 ▷ `TwistedConjugacyClass(hom1, hom2, g)` (operation)

Creates the Reidemeister coincidence class of the pair of homomorphisms $(hom1, hom2) : H \rightarrow G$ with representative g . The following attributes and operations are available:

- `Representative`, which returns g ,
- `GroupHomomorphismsOfReidemeisterClass`, which returns $[hom1, hom2]$,
- `ActingDomain`, which returns the group H ,
- `FunctionAction`, which returns the twisted conjugacy action on G ,
- `Random`, which returns a random element belonging to the Reidemeister class,
- `\in`, which can be used to test if an element belongs to the Reidemeister class - only guaranteed to work if the Reidemeister number $R(hom1, hom2)$ is finite,

- `AsList`, which lists all elements in the Reidemeister class - only works for finite groups.
- `Size`, which gives the number of elements in the Reidemeister class - only works for finite groups.

2.2.2 ReidemeisterClasses (for IsGroupHomomorphism, IsGroupHomomorphism)

- ▷ `ReidemeisterClasses(hom1, hom2)` (operation)
 ▷ `TwistedConjugacyClasses(hom1, hom2)` (operation)

Returns a list containing the Reidemeister coincidence classes of $(hom1, hom2)$ if the Reidemeister number $R(hom1, hom2)$ is finite, and returns `fail` otherwise. It is guaranteed that the Reidemeister class of the identity is in the first position.

2.2.3 ReidemeisterNumber (for IsGroupHomomorphism, IsGroupHomomorphism)

- ▷ `ReidemeisterNumber(hom1, hom2)` (operation)
 ▷ `NrTwistedConjugacyClasses(hom1, hom2)` (operation)

Returns the Reidemeister number of $(hom1, hom2)$, i.e. the number of Reidemeister classes.

Example

```
gap> tcc := ReidemeisterClass( phi, psi, g1 );
(4,6,5)^G
gap> Representative( tcc );
(4,6,5)
gap> GroupHomomorphismsOfReidemeisterClass( tcc );
[ [ (1,2)(3,5,4), (2,3)(4,5) ] -> [ (1,2)(3,4), () ],
  [ (1,2)(3,5,4), (2,3)(4,5) ] -> [ (1,4)(3,6), () ] ]
gap> ActingDomain( tcc ) = H;
true
gap> FunctionAction( tcc )( g1, h );
(1,6,4,2)(3,5)
gap> Random( tcc ) in tcc;
true
gap> List( tcc );
[ (4,6,5), (1,6,4,2)(3,5) ]
gap> Size( tcc );
2
gap> ReidemeisterClasses( phi, psi );
[ ()^G, (4,5,6)^G, (4,6,5)^G, ... ]
gap> NrTwistedConjugacyClasses( phi, psi );
184
```

Chapter 3

Miscellaneous

3.1 Groups

3.1.1 FixedPointGroup (for IsGroupHomomorphism and IsEndoGeneralMapping)

▷ FixedPointGroup(*endo*) (operation)

Let *endo* be an endomorphism of a group *G*. This command returns the subgroup of *G* consisting of the elements fixed under the endomorphism *endo*. This command is implemented only for endomorphisms of finite groups and nilpotent groups.

3.1.2 CoincidenceGroup (for IsGroupHomomorphism, IsGroupHomomorphism)

▷ CoincidenceGroup(*hom1*, *hom2*) (operation)

Let *hom1*, *hom2* be group homomorphisms from *H* to *G*. This command returns the subgroup of *H* consisting of the elements *h* for which $h^{\sim hom1} = h^{\sim hom2}$. This command is implemented only for homomorphisms where either *H* is finite, or *H* is polycyclic and *G* is nilpotent.

Example

```
gap> G := AlternatingGroup( 6 );;
gap> phi := GroupHomomorphismByImages( G, G, [ (1,2,3,4,5), (4,5,6) ],
> [ (1,2,6,3,5), (1,4,5) ] );;
gap> FixedPointGroup( phi );
Group([ (), (1,6,5,2,4), (1,5,4,6,2), (1,2,6,4,5), (1,4,2,5,6) ])
gap> H := SymmetricGroup( 5 );;
gap> khi := GroupHomomorphismByImages( H, G, [ (1,2)(3,5,4), (2,3)(4,5) ],
> [ (1,2)(3,4), () ] );;
gap> psi := GroupHomomorphismByImages( H, G, [ (1,2)(3,5,4), (2,3)(4,5) ],
> [ (1,4)(3,6), () ] );;
gap> CoincidenceGroup( khi, psi );
<permutation group with 60 generators>
```

3.2 Morphisms

3.2.1 InducedHomomorphism

▷ `InducedHomomorphism(epi1, epi2, hom)` (function)

Let *hom* be a group homomorphism from *H* to *G*, let *epi1* be an epimorphism from *H* to a group *Q* and *epi2* be an epimorphism from *G* to a group *P* such that the kernel of *epi1* is mapped into the kernel of *epi2* by *hom*. This command returns the homomorphism from *Q* to *P* induced by *hom* via *epi1* and *epi2*, that is, the homomorphism from *Q* to *P* which maps h^{epi1} to $(h^{\text{hom}})^{\text{epi2}}$, for any element *h* of *H*. This generalises `InducedAutomorphism` to homomorphisms.

3.2.2 RestrictedHomomorphism

▷ `RestrictedHomomorphism(hom, N, M)` (function)

Let *hom* be a group homomorphism from *H* to *G*, and let *N* be subgroup of *H* such that its image under *hom* is a subgroup of *M*. This command returns the homomorphism from *N* to *M* induced by *hom*. This is similar to `RestrictedMapping`, but the range is explicitly set to *M*.

Example

```
gap> G := ExamplesOfSomePcpGroups( 5 );;
gap> phi := GroupHomomorphismByImages( G, G, [ G.1, G.2, G.3, G.4 ],
> [ G.1*G.4^-1, G.3, G.2*(G.3*G.4)^2, G.4^-1 ] );;
gap> N := DerivedSubgroup(G);;
gap> p := NaturalHomomorphismByNormalSubgroup( G, N );
[ g1, g2, g3, g4, g2^2, g3^2, g4^2 ] -> [ g1, g2, g3, g4, id, id, id ]
gap> InducedHomomorphism( p, p, phi );
[ g1, g2, g3, g4 ] -> [ g1*g4, g3, g2, g4 ]
gap> RestrictedEndomorphism( phi, N, N );
[ g2^2, g3^2, g4^2 ] -> [ g3^2, g2^2*g3^4*g4^8, g4^-2 ]
```

References

- [DT20] Karel Dekimpe and Sam Tertooy. Algorithms for twisted conjugacy classes of polycyclic-by-finite groups. *To appear in: Topology and its Applications*, 2020. [2](#)
- [Fel00] Alexander Fel'shtyn. Dynamical zeta functions, nielsen theory and reidemeister torsion. *Mem. Amer. Math. Soc.*, 147(699):xii+146, 2000. [2](#), [7](#)
- [MW20] Thaís F. M. Monis and Peter Wong. Computation of nielsen and reidemeister coincidence numbers for multiple maps. *arXiv e-prints*, 2001.06730 [math.AT], 2020. [2](#)
- [Rom16] Vitaly Roman'kov. On solvability of equations with endomorphisms in nilpotent groups. *Sib. Elektron. Mat. Izv.*, 13:716–725, 2016. [2](#)
- [Ter20] Sam Tertooy. Algorithms for twisted conjugacy in nilpotent-by-finite groups. *In preparation*, 2020. [2](#)

Index

- CoincidenceGroup
 - for IsGroupHomomorphism, IsGroupHomomorphism, 12
- ExtendedReidemeisterSpectrum
 - for IsGroup, 7
- FixedPointGroup
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 12
- InducedHomomorphism, 13
- IsTwistedConjugate
 - for IsGroupHomomorphism and IsEndoGeneralMapping, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse, 4
 - for IsGroupHomomorphism, IsGroupHomomorphism, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse, 9
- NrTwistedConjugacyClasses
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 6
 - for IsGroupHomomorphism, IsGroupHomomorphism, 11
- PrintReidemeisterZeta
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 7
- ReidemeisterClass
 - for IsGroupHomomorphism and IsEndoGeneralMapping, IsMultiplicativeElementWithInverse, 5
 - for IsGroupHomomorphism, IsGroupHomomorphism, IsMultiplicativeElementWithInverse, 10
- ReidemeisterClasses
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 6
- ReidemeisterNumber
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 6
 - for IsGroupHomomorphism, IsGroupHomomorphism, 11
- ReidemeisterSpectrum
 - for IsGroup, 7
- ReidemeisterZeta
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 7
- ReidemeisterZetaCoefficients
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 7
- RepresentativeTwistedConjugation
 - for IsGroupHomomorphism and IsEndoGeneralMapping, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse, 4
 - for IsGroupHomomorphism, IsGroupHomomorphism, IsMultiplicativeElementWithInverse, IsMultiplicativeElementWithInverse, 9
- RestrictedHomomorphism, 13
- TwistedConjugacyClass
 - for IsGroupHomomorphism and IsEndoGeneralMapping, IsMultiplicativeElementWithInverse, 5
 - for IsGroupHomomorphism, IsGroupHomomorphism, IsMultiplicativeElementWithInverse, 10
- TwistedConjugacyClasses
 - for IsGroupHomomorphism and IsEndoGeneralMapping, 6

for IsGroupHomomorphism, IsGroupHomomorphism, [11](#)
TwistedConjugation
for IsGroupHomomorphism and IsEndoGeneralMapping, [4](#)
for IsGroupHomomorphism, IsGroupHomomorphism, [9](#)