

Wedderga

Wedderburn Decomposition of Group Algebras

Version 4.12.1

18 August 2026

Gurmeet Kaur Bakshi

Osnel Broche Cristo

Allen Herman

Olexandr Konovalov

Sugandha Maheshwary

Aurora Olivieri

Gabriela Olteanu

Ángel del Río

Inneke Van Gelder

Gurmeet Kaur Bakshi Email: gkbakshi@pu.ac.in
Address: Center for Advanced Study in Mathematics,
Panjab University, Chandigarh, PIN-160014, India

Osnel Broche Cristo Email: osnel@ufla.br
Address: Departamento de Ciências Exatas, Universidade Federal
de Lavras - UFLA, Campus Universitário - Caixa Postal
3037, 37200-000, Lavras - MG, Brazil

Allen Herman Email: aherman@math.uregina.ca
Homepage: <http://www.math.uregina.ca/~aherman/>
Address: Department of Mathematics and Statistics,
University of Regina,
3737 Wascana Parkway,
Regina, SK, S0G 0E0, Canada

Olexandr Kononov Email: obk1@st-andrews.ac.uk
Homepage: <https://olexandr-kononov.github.io/>
Address: School of Computer Science, University of St Andrews
Jack Cole Building, North Haugh,
St Andrews, Fife, KY16 9SX, Scotland

Sugandha Maheshwary Email: msugandha@ma.iitr.ac.in
Address: Department of Mathematics
Indian Institute of Technology Roorkee
Roorkee, Uttarakhand 247667, India

Aurora Olivieri Email: olivieri@usb.ve
Address: Departamento de Matemáticas
Universidad Simón Bolívar
Apartado Postal 89000, Caracas 1080-A, Venezuela

Gabriela Olteanu Email: gabriela.olteanu@econ.ubbcluj.ro
Homepage: <http://math.ubbcluj.ro/~olteanu>
Address: Department of Statistics-Forecasts-Mathematics
Faculty of Economics and Business Administration
Babes-Bolyai University
Str. T. Mihali 58-60, 400591 Cluj-Napoca, Romania

Ángel del Río Email: adelrio@um.es
Homepage: <http://www.um.es/adelrio>
Address: Departamento de Matemáticas, Universidad de Murcia
30100 Murcia, Spain

Inneke Van Gelder Email: ivgelder@vub.ac.be
Homepage: <http://homepages.vub.ac.be/~ivgelder>
Address: Vrije Universiteit Brussel, Departement Wiskunde
Pleinlaan 2
1050 Brussels, Belgium

Abstract

“Wedderga” stands for “WEDDERburn decomposition of Group Algebras”. This is a GAP package to compute the simple components of the Wedderburn decomposition of semisimple group algebras of finite groups over finite fields and over subfields of finite cyclotomic extensions of the rationals. It also contains functions that produce the primitive central idempotents of semisimple group algebras and a complete set of orthogonal primitive idempotents. Other functions of **Wedderga** allow to construct crossed products over a group with coefficients in an associative ring with identity and the multiplication determined by a given action and twisting.

Copyright

© 2006–2026 by Gurmeet Kaur Bakshi, Osnel Broche Cristo, Allen Herman, Olexandr Konovalov, Sugandha Maheshwary, Aurora Olivieri, Gabriela Olteanu, Ángel del Río and Inneke Van Gelder.

Wedderga is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. For details, see the FSF’s own site <https://www.gnu.org/licenses/gpl.html>.

If you obtained **Wedderga**, we would be grateful for a short notification sent to one of the authors. If you publish a result which was partially obtained with the usage of **Wedderga**, please cite it in the following form:

G. K. Bakshi, O. Broche Cristo, A. Herman, O. Konovalov, S. Maheshwary, A. Olivieri, G. Olteanu, Á. del Río and I. Van Gelder. *Wedderga --- Wedderburn Decomposition of Group Algebras, Version 4.12.1*; 2026 (<https://gap-packages.github.io/wedderga/>).

Acknowledgements

We all are very grateful to Steve Linton for communicating the package and to the referee for careful testing **Wedderga** and useful suggestions. Also we acknowledge very much the members of the GAP team: Thomas Breuer, Alexander Hulpke, Frank Lübeck and many other colleagues for helpful comments and advice. We would like also to thank Thomas Breuer for the code of `PrimitiveCentralIdempotentsByCharacterTable` for rational group algebras.

We gratefully acknowledge the support of **Wedderga** development by the following institutions:

- University of Murcia;
- Francqui Stichting grant ADSI107;
- M.E.C. of Romania (CEEX-ET 47/2006);
- D.G.I. of Spain;
- Fundación Séneca of Murcia;
- CAPES and FAPESP of Brazil;
- Research Foundation Flanders (FWO - Vlaanderen);
- CCP CoDiMa (EP/M022641/1);
- Department of Science and Technology (DST), India.

Contents

1	Introduction	6
1.1	General aims of Wedderga package	6
1.2	Installation and system requirements	6
1.3	Main functions of Wedderga package	7
2	Wedderburn decomposition	9
2.1	Wedderburn decomposition of a group algebra	9
2.2	Simple quotients	13
3	Shoda pairs	17
3.1	Computing extremely strong Shoda pairs	17
3.2	Computing strong Shoda pairs	18
3.3	Properties related with Shoda pairs	18
4	Idempotents	21
4.1	Computing idempotents from character table	21
4.2	Testing lists of idempotents for completeness	21
4.3	Idempotents from Shoda pairs	22
4.4	Complete set of orthogonal primitive idempotents from Shoda pairs and cyclotomic classes	25
5	Crossed products and their elements	27
5.1	Construction of crossed products	27
5.2	Crossed product elements and their properties	33
6	Useful properties and functions	35
6.1	Semisimple group algebras of finite groups	35
6.2	Operations with group rings elements	37
6.3	Cyclotomic classes	38
6.4	Other commands	39
7	Functions for calculating Schur indices and identifying division algebras	40
7.1	Main Schur Index and Division Algebra Functions	40
7.2	Cyclotomic Reciprocity Functions	43
7.3	Global Splitting and Character Descent Functions	44
7.4	Local index functions for Cyclic Cyclotomic Algebras	47
7.5	Local index functions for Non-Cyclic Cyclotomic Algebras	48

7.6	Local index functions for Rational Quaternion Algebras	52
7.7	Functions involving Cyclic Algebras	54
8	Applications of the Wedderga package	57
8.1	Coding theory applications	57
9	The basic theory behind Wedderga	59
9.1	Group rings and group algebras	59
9.2	Semisimple group algebras	59
9.3	Wedderburn components	59
9.4	Characters and primitive central idempotents	60
9.5	Central simple algebras and Brauer equivalence	61
9.6	Crossed Products	61
9.7	Cyclic Crossed Products	62
9.8	Abelian Crossed Products	63
9.9	Classical crossed products	63
9.10	Cyclic Algebras	63
9.11	Cyclotomic algebras	64
9.12	Numerical description of cyclotomic algebras	64
9.13	Idempotents given by subgroups	64
9.14	Shoda pairs of a group	65
9.15	Strong Shoda pairs of a group	65
9.16	Extremely strong Shoda pairs of a group	66
9.17	Strongly monomial characters and strongly monomial groups	66
9.18	Normally monomial characters and normally monomial groups	67
9.19	Cyclotomic Classes and Strong Shoda Pairs	68
9.20	Theory for Local Schur Index and Division Algebra Part Calculations	69
9.21	Obtaining Algebras with structure constants as terms of the Wedderburn decomposition	70
9.22	A complete set of orthogonal primitive idempotents	71
9.23	Applications to coding theory	72
	References	74
	Index	75

Chapter 1

Introduction

1.1 General aims of Wedderga package

The title “Wedderga” stands for “WEDDERburn decomposition of Group Algebras”. This is a GAP package to compute the simple components of the Wedderburn decomposition of semisimple group algebras. So the main functions of the package returns a list of simple algebras whose direct sum is isomorphic to the group algebra given as input.

The method implemented by the package produces the Wedderburn decomposition of a group algebra FG provided G is a finite group and F is either a finite field of characteristic coprime to the order of G , or an abelian number field (i.e. a subfield of a finite cyclotomic extension of the rationals).

Other functions of **Wedderga** compute the primitive central idempotents of semisimple group algebras and a complete set of orthogonal primitive idempotents, and calculate Schur indices of simple algebras.

The package also provides functions to construct crossed products over a group with coefficients in an associative ring with identity and the multiplication determined by a given action and twisting.

Furhermore, the package provides functions to create code words from a group ring element.

1.2 Installation and system requirements

Wedderga does not use external binaries and, therefore, works without restrictions on the type of the operating system. It is designed for GAP4.4 and no compatibility with previous releases of GAP4 is guaranteed.

To use the **Wedderga** online help it is necessary to install the GAP4 package **GAP-Doc** by Frank Lübeck and Max Neunhöffer, which is available from the GAP site or from <http://www.math.rwth-aachen.de/~Frank.Luebeck/GAPDoc/>.

Wedderga is distributed in standard formats (tar.gz, tar.bz2, -win.zip) and can be obtained from <https://gap-packages.github.io/wedderga/>. To install **Wedderga**, unpack its archive into the pkg subdirectory of your GAP installation.

When you don't have access to the directory of your main GAP installation, you can also install the package *outside the GAP main directory* by unpacking it inside a directory MYGAPDIR/pkg. Then to be able to load **Wedderga** you need to call GAP with the `-l " ;MYGAPDIR"` option.

Installation using other archive formats is performed in a similar way.

If the package is installed correctly, it should be loaded as follows:

Example

```
gap> LoadPackage("wedderga");
-----
Loading Wedderga 4.10.5 (Wedderga)
by Gurmeet Kaur Bakshi (gkbakshi@pu.ac.in),
   Osnel Broche Cristo (osnel@ufla.br),
   Allen Herman (aherman@math.uregina.ca),
   Olexandr Konovalov (https://olexandr-konovalov.github.io/),
   Sugandha Maheshwary (msugandha@ma.iitr.ac.in),
   Gabriela Olteanu (http://math.ubbcluj.ro/~olteanu),
   Aurora Olivieri (olivieri@usb.ve),
   Angel del Rio (http://www.um.es/adelfrio), and
   Inneke Van Gelder (http://homepages.vub.ac.be/~ivgelder).
Homepage: https://gap-packages.github.io/wedderga
Report issues at https://github.com/gap-packages/wedderga/issues
-----
true
```

1.3 Main functions of Wedderga package

The main functions of Wedderga are `WedderburnDecomposition` (2.1.1) and `WedderburnDecompositionInfo` (2.1.2).

`WedderburnDecomposition` (2.1.1) computes a list of simple algebras such that their direct product is isomorphic to the group algebra FG , given as input. Thus, the direct product of the entries of the output is the *Wedderburn decomposition* (9.3) of FG .

If F is an abelian number field then the entries of the output are given as matrix algebras over cyclotomic algebras (see 9.11), thus, the entries of the output of `WedderburnDecomposition` (2.1.1) are realizations of the *Wedderburn components* (9.3) of FG as algebras which are *Brauer equivalent* (9.5) to *cyclotomic algebras* (9.11). Recall that the Brauer-Witt Theorem ensures that every simple factor of a semisimple group ring FG is Brauer equivalent (that is represents the same class in the Brauer group of its centre) to a cyclotomic algebra ([Yam74]. In this case the algorithm is based on a computational oriented proof of the Brauer-Witt Theorem due to Olteanu [Olt07] which uses previous work by Olivieri, del Río and Simón [OdS04] (see also [Od03]) for rational group algebras of *strongly monomial groups* (9.17). The algorithms are also based upon the work of Bakshi and Maheshwary [BM14] (see also [BM16]) on the rational group algebras of *normally monomial groups* (9.18).

The Wedderburn components of FG are also matrix algebras over division rings which are finite extensions of the field F . If F is finite then by the Wedderburn theorem these division rings are finite fields. In this case the output of `WedderburnDecomposition` (2.1.1) represents the factors of FG as matrix algebras over finite extensions of the field F .

In theory Wedderga could handle the calculation of the Wedderburn decomposition of group algebras of groups of arbitrary size but in practice if the order of the group is greater than 5000 then the program may crash. The way the group is given is relevant for the performance. Usually the program works better for groups given as permutation groups or pc groups.

Example

```

gap> QG := GroupRing( Rationals, SymmetricGroup(4) );
<algebra-with-one over Rationals, with 2 generators>
gap> WedderburnDecomposition(QG);
[ Rationals, Rationals, <crossed product with center Rationals over CF(
  3) of a group of size 2>, ( Rationals^[ 3, 3 ] ), ( Rationals^[ 3, 3 ] ) ]
gap> FG := GroupRing( CF(5), SymmetricGroup(4) );
<algebra-with-one over CF(5), with 2 generators>
gap> WedderburnDecomposition( FG );
[ CF(5), CF(5), <crossed product with center CF(5) over AsField( CF(5), CF(
  15) ) of a group of size 2>, ( CF(5)^[ 3, 3 ] ), ( CF(5)^[ 3, 3 ] ) ]
gap> FG := GroupRing( GF(5), SymmetricGroup(4) );
<algebra-with-one over GF(5), with 2 generators>
gap> WedderburnDecomposition( FG );
[ ( GF(5)^[ 1, 1 ] ), ( GF(5)^[ 1, 1 ] ), ( GF(5)^[ 2, 2 ] ),
  ( GF(5)^[ 3, 3 ] ), ( GF(5)^[ 3, 3 ] ) ]
gap> FG := GroupRing( GF(5), SmallGroup(24,3) );
<algebra-with-one over GF(5), with 4 generators>
gap> WedderburnDecomposition( FG );
[ ( GF(5)^[ 1, 1 ] ), ( GF(5^2)^[ 1, 1 ] ), ( GF(5)^[ 2, 2 ] ),
  ( GF(5^2)^[ 2, 2 ] ), ( GF(5)^[ 3, 3 ] ) ]

```

Instead of `WedderburnDecomposition` (2.1.1), that returns a list of `GAP` objects, `WedderburnDecompositionInfo` (2.1.2) returns the numerical description of these objects. See Section 9.12 for theoretical background.

Chapter 2

Wedderburn decomposition

2.1 Wedderburn decomposition of a group algebra

2.1.1 WedderburnDecomposition

▷ `WedderburnDecomposition(FG)` (attribute)

Returns: A list of simple algebras.

The input FG should be a group algebra of a finite group G over the field F , where F is either an abelian number field (i.e. a subfield of a finite cyclotomic extension of the rationals) or a finite field of characteristic coprime with the order of G .

The function returns the list of all *Wedderburn components* (9.3) of the group algebra FG . If F is an abelian number field then each Wedderburn component is given as a matrix algebra of a *cyclotomic algebra* (9.11). If F is a finite field then the Wedderburn components are given as matrix algebras over finite fields.

Example

```
gap> WedderburnDecomposition( GroupRing( GF(5), DihedralGroup(16) ) );
[ ( GF(5)^[ 1, 1 ] ), ( GF(5)^[ 1, 1 ] ), ( GF(5)^[ 1, 1 ] ),
  ( GF(5)^[ 1, 1 ] ), ( GF(5)^[ 2, 2 ] ), ( GF(5^2)^[ 2, 2 ] ) ]
gap> WedderburnDecomposition( GroupRing( Rationals, DihedralGroup(16) ) );
[ Rationals, Rationals, Rationals, Rationals, ( Rationals^[ 2, 2 ] ),
  <crossed product with center NF(8,[ 1, 7 ]) over AsField( NF(8,
    [ 1, 7 ]), CF(8) ) of a group of size 2> ]
gap> WedderburnDecomposition( GroupRing( CF(5), DihedralGroup(16) ) );
[ CF(5), CF(5), CF(5), CF(5), ( CF(5)^[ 2, 2 ] ),
  <crossed product with center NF(40,[ 1, 31 ]) over AsField( NF(40,
    [ 1, 31 ]), CF(40) ) of a group of size 2> ]
```

The previous examples show that if D_{16} denotes the dihedral group of order 16 then the *Wedderburn decomposition* (9.3) of $\mathbb{F}_5 D_{16}$, $\mathbb{Q} D_{16}$ and $\mathbb{Q}(\xi_5) D_{16}$ are respectively

$$\mathbb{F}_5 D_{16} = 4\mathbb{F}_5 \oplus M_2(\mathbb{F}_5) \oplus M_2(\mathbb{F}_{25}),$$

$$\mathbb{Q} D_{16} = 4\mathbb{Q} \oplus M_2(\mathbb{Q}) \oplus (K(\xi_8)/K, t),$$

and

$$\mathbb{Q}(\xi_5) D_{16} = 4\mathbb{Q}(\xi_5) \oplus M_2(\mathbb{Q}(\xi_5)) \oplus (F(\xi_{40})/F, t),$$

where $(K(\xi_8)/K, t)$ is a *cyclotomic algebra* (9.11) with the centre $K = NF(8, [1, 7]) = \mathbb{Q}(\sqrt{2})$, $(F(\xi_{40})/F, t) = \mathbb{Q}(\sqrt{2}, \xi_5)$ is a cyclotomic algebra with centre $F = NF(40, [1, 31])$ and ξ_n denotes a n -th root of unity.

Two more examples:

Example

```
gap> WedderburnDecomposition( GroupRing( Rationals, SmallGroup(48,15) ) );
[ Rationals, Rationals, Rationals, Rationals,
  <crossed product with center Rationals over CF(3) of a group of size 2>,
  <crossed product with center Rationals over GaussianRationals of a group of \
size 2>, <crossed product with center Rationals over CF(3) of a group of size
  2>, <crossed product with center NF(8,[ 1, 7 ]) over AsField( NF(8,
  [ 1, 7 ]), CF(8) ) of a group of size 2>, ( CF(3)^[ 2, 2 ] ),
  <crossed product with center Rationals over CF(12) of a group of size 4> ]
gap> WedderburnDecomposition( GroupRing( CF(3), SmallGroup(48,15) ) );
[ CF(3), CF(3), CF(3), CF(3), ( CF(3)^[ 2, 2 ] ),
  <crossed product with center CF(3) over AsField( CF(3), CF(
  12) ) of a group of size 2>, ( CF(3)^[ 2, 2 ] ),
  <crossed product with center NF(24,[ 1, 7 ]) over AsField( NF(24,
  [ 1, 7 ]), CF(24) ) of a group of size 2>, ( CF(3)^[ 2, 2 ] ),
  ( CF(3)^[ 2, 2 ] ), ( <crossed product with center CF(3) over AsField( CF(
  3), CF(12) ) of a group of size 2>^[ 2, 2 ] ) ]
```

In some cases, in characteristic zero, some entries of the output of `WedderburnDecomposition` (2.1.1) do not provide full matrix algebras over a *cyclotomic algebra* (9.11), but "fractional matrix algebras". That entry is not an algebra that can be used as a GAP object. Instead it is a pair formed by a rational giving the "size" of the matrices and a crossed product. See 9.3 for a theoretical explanation of this phenomenon. In this case a warning message is displayed.

Example

```
gap> QG:=GroupRing(Rationals,SmallGroup(240,89));
<algebra-with-one over Rationals, with 2 generators>
gap> WedderburnDecomposition(QG);
Wedderga: Warning!!!
Some of the Wedderburn components displayed are FRACTIONAL MATRIX ALGEBRAS!!!

[ Rationals, Rationals, <crossed product with center Rationals over CF(
  5) of a group of size 4>, ( Rationals^[ 4, 4 ] ), ( Rationals^[ 4, 4 ] ),
  ( Rationals^[ 5, 5 ] ), ( Rationals^[ 5, 5 ] ), ( Rationals^[ 6, 6 ] ),
  <crossed product with center NF(12,[ 1, 11 ]) over AsField( NF(12,
  [ 1, 11 ]), NF(60,[ 1, 11 ])) of a group of size 4>,
  [ 3/2, <crossed product with center NF(8,[ 1, 7 ]) over AsField( NF(8,
  [ 1, 7 ]), NF(40,[ 1, 31 ])) of a group of size 4> ] ]
```

2.1.2 WedderburnDecompositionInfo

▷ `WedderburnDecompositionInfo(FG)`

(attribute)

Returns: A list with each entry a numerical description of a *cyclotomic algebra* (9.11).

The input FG should be a group algebra of a finite group G over the field F , where F is either an abelian number field (i.e. a subfield of a finite cyclotomic extension of the rationals) or a finite field of characteristic coprime to the order of G .

This function is a numerical counterpart of `WedderburnDecomposition` (2.1.1).

It returns a list formed by lists of lengths 2, 4 or 5.

The lists of length 2 are of the form

$$[n, F],$$

where n is a positive integer and F is a field. It represents the $n \times n$ matrix algebra $M_n(F)$ over the field F .

The lists of length 4 are of the form

$$[n, F, k, [d, \alpha, \beta]],$$

where F is a field and n, k, d, α, β are non-negative integers, satisfying the conditions mentioned in Section 9.12. It represents the $n \times n$ matrix algebra $M_n(A)$ over the cyclic algebra

$$A = F(\xi_k)[u | \xi_k^u = \xi_k^\alpha, u^d = \xi_k^\beta],$$

where ξ_k is a primitive k -th root of unity.

The lists of length 5 are of the form

$$[n, F, k, [d_i, \alpha_i, \beta_i]_{i=1}^m, [\gamma_{i,j}]_{1 \leq i < j \leq m}],$$

where F is a field and $n, k, d_i, \alpha_i, \beta_i, \gamma_{i,j}$ are non-negative integers. It represents the $n \times n$ matrix algebra $M_n(A)$ over the cyclotomic algebra (9.11)

$$A = F(\xi_k)[g_1, \dots, g_m | \xi_k^{g_i} = \xi_k^{\alpha_i}, g_i^{d_i} = \xi_k^{\beta_i}, g_j g_i = \xi_k^{\gamma_{ij}} g_i g_j],$$

where ξ_k is a primitive k -th root of unity (see 9.12).

Example

```
gap> WedderburnDecompositionInfo( GroupRing( Rationals, DihedralGroup(16) ) );
[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ],
  [ 2, Rationals ], [ 2, NF(8,[ 1, 7 ]) ] ]
gap> WedderburnDecompositionInfo( GroupRing( CF(5), DihedralGroup(16) ) );
[ [ 1, CF(5) ], [ 1, CF(5) ], [ 1, CF(5) ], [ 1, CF(5) ], [ 2, CF(5) ],
  [ 2, NF(40,[ 1, 31 ]) ] ]
```

The interpretation of the previous example gives rise to the following *Wedderburn decompositions* (9.3), where D_{16} is the dihedral group of order 16 and ξ_5 is a primitive 5-th root of unity.

$$\mathbb{Q}D_{16} = 4\mathbb{Q} \oplus M_2(\mathbb{Q}) \oplus M_2(\mathbb{Q}(\sqrt{2})).$$

$$\mathbb{Q}(\xi_5)D_{16} = 4\mathbb{Q}(\xi_5) \oplus M_2(\mathbb{Q}(\xi_5)) \oplus M_2(\mathbb{Q}(\xi_5, \sqrt{2})).$$

Example

```
gap> F:=FreeGroup("a","b");;a:=F.1;;b:=F.2;;rel:=[a^8,a^4*b^2,b^-1*a*b*a];;
gap> Q16:=F/rel;; QQ16:=GroupRing( Rationals, Q16 );;
gap> QS4:=GroupRing( Rationals, SymmetricGroup(4) );;
gap> WedderburnDecomposition(QQ16);
[ Rationals, Rationals, Rationals, Rationals, ( Rationals^[ 2, 2 ] ),
  <crossed product with center NF(8,[ 1, 7 ]) over AsField( NF(8,
    [ 1, 7 ]), CF(8) ) of a group of size 2> ]
gap> WedderburnDecomposition( QS4 );
[ Rationals, Rationals, <crossed product with center Rationals over CF(
```

```

3) of a group of size 2>, (Rationals^[ 3, 3 ]), (Rationals^[ 3, 3 ] ) ]
gap> WedderburnDecompositionInfo(QQ16);
[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ],
  [ 2, Rationals ], [ 1, NF(8,[ 1, 7 ]) ], 8, [ 2, 7, 4 ] ] ]
gap> WedderburnDecompositionInfo(QS4);
[ [ 1, Rationals ], [ 1, Rationals ], [ 2, Rationals ], [ 3, Rationals ],
  [ 3, Rationals ] ] ]

```

In the previous example we computed the Wedderburn decomposition of the rational group algebra $\mathbb{Q}Q_{16}$ of the quaternion group of order 16 and the rational group algebra $\mathbb{Q}S_4$ of the symmetric group on four letters. For the two group algebras we used both `WedderburnDecomposition` (2.1.1) and `WedderburnDecompositionInfo` (2.1.2).

The output of `WedderburnDecomposition` (2.1.1) shows that

$$\mathbb{Q}Q_{16} = 4\mathbb{Q} \oplus M_2(\mathbb{Q}) \oplus A,$$

$$\mathbb{Q}S_4 = 2\mathbb{Q} \oplus 2M_3(\mathbb{Q}) \oplus B,$$

where A and B are *crossed products* (9.6) with coefficients in the cyclotomic fields $\mathbb{Q}(\xi_8)$ and $\mathbb{Q}(\xi_3)$ respectively. This output can be used as a GAP object, but it does not give clear information on the structure of the algebras A and B .

The numerical information displayed by `WedderburnDecompositionInfo` (2.1.2) means that

$$A = \mathbb{Q}(\xi | \xi^8 = 1)[g | \xi^g = \xi^7 = \xi^{-1}, g^2 = \xi^4 = -1],$$

$$B = \mathbb{Q}(\xi | \xi^3 = 1)[g | \xi^g = \xi^2 = \xi^{-1}, g^2 = 1].$$

Both A and B are quaternion algebras over its centre which is $\mathbb{Q}(\xi + \xi^{-1})$ and the former is equal to $\mathbb{Q}(\sqrt{2})$ and $\mathbb{Q}$ respectively.

In B , one has $(g+1)(g-1) = 0$, while g is neither 1 nor -1 . This shows that $B = M_2(\mathbb{Q})$. However the relation $g^2 = -1$ in A shows that

$$A = \mathbb{Q}(\sqrt{2})[i, g | i^2 = g^2 = -1, ig = -gi]$$

and so A is a division algebra with centre $\mathbb{Q}(\sqrt{2})$, which is a subalgebra of the algebra of Hamiltonian quaternions. This could be deduced also using well known methods on cyclic algebras (see e.g. [Rei03]).

The next example shows the output of `WedderburnDecompositionInfo` for $\mathbb{Q}G$ and $\mathbb{Q}(\xi_3)G$, where $G = \text{SmallGroup}(48,15)$. The user can compare it with the output of `WedderburnDecomposition` (2.1.1) for the same group in the previous section. Notice that the last entry of the *Wedderburn decomposition* (9.3) of $\mathbb{Q}G$ is not given as a matrix algebra of a cyclic algebra. However, the corresponding entry of $\mathbb{Q}(\xi_3)G$ is a matrix algebra of a cyclic algebra.

Example

```

gap> WedderburnDecompositionInfo( GroupRing( Rationals, SmallGroup(48,15) ) );
[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ],
  [ 2, Rationals ], [ 2, Rationals ], [ 2, Rationals ], [ 2, NF(8,[ 1, 7 ]) ],
  [ 2, CF(3) ], [ 1, Rationals, 12, [ [ 2, 5, 3 ], [ 2, 7, 0 ] ], [ [ 3 ] ] ] ]
gap> WedderburnDecompositionInfo( GroupRing( CF(3), SmallGroup(48,15) ) );
[ [ 1, CF(3) ], [ 1, CF(3) ], [ 1, CF(3) ], [ 1, CF(3) ], [ 2, CF(3) ],
  [ 2, CF(3) ], [ 2, CF(3) ], [ 2, NF(24,[ 1, 7 ]) ], [ 2, CF(3) ],
  [ 2, CF(3) ], [ 4, CF(3) ] ] ]

```

In some cases some of the first entries of the output of `WedderburnDecompositionInfo` (2.1.2) are not integers and so the corresponding *Wedderburn components* (9.3) are given as "fractional matrix algebras" of *cyclotomic algebras* (9.11). See 9.3 for a theoretical explanation of this phenomenon. In that case a warning message will be displayed during the first call of `WedderburnDecompositionInfo`.

Example

```
gap> QG:=GroupRing(Rationals,SmallGroup(240,89));
<algebra-with-one over Rationals, with 2 generators>
gap> WedderburnDecompositionInfo(QG);
Wedderga: Warning!!!
Some of the Wedderburn components displayed are FRACTIONAL MATRIX ALGEBRAS!!!

[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals, 10, [ 4, 3, 5 ] ],
  [ 4, Rationals ], [ 4, Rationals ], [ 5, Rationals ], [ 5, Rationals ],
  [ 6, Rationals ], [ 1, NF(12,[ 1, 11 ]), 10, [ 4, 3, 5 ] ],
  [ 3/2, NF(8,[ 1, 7 ]), 10, [ 4, 3, 5 ] ] ]
```

The interpretation of the output in the previous example gives rise to the following *Wedderburn decomposition* (9.3) of $\mathbb{Q}G$ for G the small group [240, 89]:

$$\mathbb{Q}G = 2\mathbb{Q} \oplus 2M_4(\mathbb{Q}) \oplus 2M_5(\mathbb{Q}) \oplus M_6(\mathbb{Q}) \oplus A \oplus B \oplus C$$

where

$$A = \mathbb{Q}(\xi_{10})[u | \xi_{10}^u = \xi_{10}^3, u^4 = -1],$$

B is an algebra of degree $(4*2)/2 = 4$ which is *Brauer equivalent* (9.5) to

$$B_1 = \mathbb{Q}(\xi_{60})[u, v | \xi_{60}^u = \xi_{60}^{13}, u^4 = \xi_{60}^5, \xi_{60}^v = \xi_{60}^{11}, v^2 = 1, vu = uv],$$

and C is an algebra of degree $(4*2)*3/4 = 6$ which is *Brauer equivalent* (9.5) to

$$C_1 = \mathbb{Q}(\xi_{60})[u, v | \xi_{60}^u = \xi_{60}^7, u^4 = \xi_{60}^5, \xi_{60}^v = \xi_{60}^{31}, v^2 = 1, vu = uv].$$

The precise description of B and C requires the usage of "ad hoc" arguments.

2.2 Simple quotients

2.2.1 SimpleAlgebraByCharacter

▷ `SimpleAlgebraByCharacter(FG, chi)` (operation)

Returns: A simple algebra.

The first input FG should be a *semisimple group algebra* (9.2) over a finite group G and the second input should be an irreducible character of G .

The output is a matrix algebra of a *cyclotomic algebras* (9.11) which is isomorphic to the unique *Wedderburn component* (9.3) A of FG such that $\chi(A) \neq 0$.

Example

```
gap> A5 := AlternatingGroup(5);
Alt( [ 1 .. 5 ] )
gap> chi := First(Irr( A5 ), chi -> Degree(chi) = 3);;
gap> SimpleAlgebraByCharacter( GroupRing( Rationals, A5 ), chi );
( NF(5,[ 1, 4 ])^[ 3, 3 ] )
gap> SimpleAlgebraByCharacter( GroupRing( GF(7), A5 ), chi );
```

```

( GF(7^2)^[ 3, 3 ] )
gap> G:=SmallGroup(128,100);
<pc group of size 128 with 7 generators>
gap> chi4:=Filtered(Irr(G),x->Degree(x)=4);;
gap> List(chi4,x->SimpleAlgebraByCharacter(GroupRing(Rationals,G),x));
[ ( <crossed product with center NF(8,[ 1, 3 ]) over AsField( NF(8,
[ 1, 3 ]), CF(8) ) of a group of size 2^[ 2, 2 ] ),
  ( <crossed product with center NF(8,[ 1, 3 ]) over AsField( NF(8,
[ 1, 3 ]), CF(8) ) of a group of size 2^[ 2, 2 ] ),
  ( <crossed product with center NF(8,[ 1, 3 ]) over AsField( NF(8,
[ 1, 3 ]), CF(8) ) of a group of size 2^[ 2, 2 ] ),
  ( <crossed product with center NF(8,[ 1, 3 ]) over AsField( NF(8,
[ 1, 3 ]), CF(8) ) of a group of size 2^[ 2, 2 ] ) ]

```

2.2.2 SimpleAlgebraByCharacterInfo

▷ SimpleAlgebraByCharacterInfo(FG , χ) (operation)

Returns: The numerical description of the output of SimpleAlgebraByCharacter (2.2.1).

The first input FG is a *semisimple group algebra* (9.2) over a finite group G and the second input is an irreducible character of G .

The output is the numerical description 9.12 of the *cyclotomic algebra* (9.11) which is isomorphic to the unique *Wedderburn component* (9.3) A of FG such that $\chi(A) \neq 0$.

See 9.12 for the interpretation of the numerical information given by the output.

Example

```

gap> G:=SmallGroup(128,100);
<pc group of size 128 with 7 generators>
gap> QG:=GroupRing(Rationals,G);
<algebra-with-one over Rationals, with 7 generators>
gap> chi4:=Filtered(Irr(G),x->Degree(x)=4);;
gap> List(chi4,x->SimpleAlgebraByCharacterInfo(QG,x));
[ [ 4, NF(8,[ 1, 3 ]) ], [ 4, NF(8,[ 1, 3 ]) ], [ 4, NF(8,[ 1, 3 ]) ],
  [ 4, NF(8,[ 1, 3 ]) ] ]

```

2.2.3 SimpleAlgebraByStrongSP (for rational group algebra)

- ▷ SimpleAlgebraByStrongSP(QG, K, H) (operation)
- ▷ SimpleAlgebraByStrongSPNC(QG, K, H) (operation)
- ▷ SimpleAlgebraByStrongSP(FG, K, H, C) (operation)
- ▷ SimpleAlgebraByStrongSPNC(FG, K, H, C) (operation)

Returns: A simple algebra.

In the three-argument version the input must be formed by a *semisimple rational group algebra* QG (see 9.2) and two subgroups K and H of G which form a *strong Shoda pair* (9.15) of G .

The three-argument version returns the Wedderburn component (9.3) of the rational group algebra QG realized by the strong Shoda pair (K,H) .

In the four-argument version the first argument is a semisimple finite group algebra FG , (K,H) is a strong Shoda pair of G and the fourth input data is either a generating q -cyclotomic class modulo the index of H in K or a representative of a generating q -cyclotomic class modulo the index of H in K (see 9.19).

The four-argument version returns the Wedderburn component (9.3) of the finite group algebra FG realized by the strong Shoda pair (K,H) and the cyclotomic class C (or the cyclotomic class containing C).

The versions ending in NC do not check if (K,H) is a strong Shoda pair of G . In the four-argument version it is also not checked whether C is either a generating q -cyclotomic class modulo the index of H in K or an integer coprime to the index of H in K .

Example

```

gap> F:=FreeGroup("a","b");; a:=F.1;; b:=F.2;;
gap> G:=F/[ a^16, b^2*a^8, b^-1*a*b*a^9 ];; a:=G.1;; b:=G.2;;
gap> K:=Subgroup(G,[a]);; H:=Subgroup(G,[]);;
gap> QG:=GroupRing( Rationals, G );;
gap> FG:=GroupRing( GF(7), G );;
gap> SimpleAlgebraByStrongSP( QG, K, H );
<crossed product over CF(16) of a group of size 2>
gap> SimpleAlgebraByStrongSP( FG, K, H, [1,7] );
( GF(7)^[ 2, 2 ] )
gap> SimpleAlgebraByStrongSP( FG, K, H, 1 );
( GF(7)^[ 2, 2 ] )

```

2.2.4 SimpleAlgebraByStrongSPInfo (for rational group algebra)

- ▷ SimpleAlgebraByStrongSPInfo(QG, K, H) (operation)
- ▷ SimpleAlgebraByStrongSPInfoNC(QG, K, H) (operation)

- ▷ `SimpleAlgebraByStrongSPInfo(FG, K, H, C)` (operation)
- ▷ `SimpleAlgebraByStrongSPInfoNC(FG, K, H, C)` (operation)

Returns: A numerical description of one simple algebra.

In the three-argument version the input must be formed by a *semisimple rational group algebra* (9.2) QG and two subgroups K and H of G which form a *strong Shoda pair* (9.15) of G . It returns the numerical information describing the Wedderburn component (9.12) of the rational group algebra QG realized by a the strong Shoda pair (K, H) .

In the four-argument version the first input is a semisimple finite group algebra FG , (K, H) is a strong Shoda pair of G and the fourth input data is either a generating q -cyclotomic class modulo the index of H in K or a representative of a generating q -cyclotomic class modulo the index of H in K (9.19). It returns a pair of positive integers $[n, r]$ which represent the $n \times n$ matrix algebra over the field of order r which is isomorphic to the Wedderburn component of FG realized by a the strong Shoda pair (K, H) and the cyclotomic class C (or the cyclotomic class containing the integer C).

The versions ending in NC do not check if (K, H) is a strong Shoda pair of G . In the four-argument version it is also not checked whether C is either a generating q -cyclotomic class modulo the index of H in K or an integer coprime with the index of H in K .

Example

```
gap> F:=FreeGroup("a","b");; a:=F.1;; b:=F.2;;
gap> G:=F/[ a^16, b^2*a^8, b^-1*a*b*a^9 ];; a:=G.1;; b:=G.2;;
gap> K:=Subgroup(G,[a]);; H:=Subgroup(G,[]);;
gap> QG:=GroupRing( Rationals, G );;
gap> FG:=GroupRing( GF(7), G );;
gap> SimpleAlgebraByStrongSP( QG, K, H );
<crossed product over CF(16) of a group of size 2>
gap> SimpleAlgebraByStrongSPInfo( QG, K, H );
[ 1, NF(16,[ 1, 7 ]), 16, [ [ 2, 7, 8 ] ], [ ] ]
gap> SimpleAlgebraByStrongSPInfo( FG, K, H, [1,7] );
[ 2, 7 ]
gap> SimpleAlgebraByStrongSPInfo( FG, K, H, 1 );
[ 2, 7 ]
```

Chapter 3

Shoda pairs

3.1 Computing extremely strong Shoda pairs

3.1.1 ExtremelyStrongShodaPairs

▷ `ExtremelyStrongShodaPairs( $G$ )` (attribute)

Returns: A list of pairs of subgroups of the input group.

The input should be a finite group G .

Computes a list of representatives of the equivalence classes of *extremely strong Shoda pairs* (9.16) of a finite group G .

Example

```
gap> ExtremelyStrongShodaPairs(DihedralGroup(32));
[ [ <pc group of size 32 with 5 generators>,
  <pc group of size 32 with 5 generators> ],
  [ <pc group of size 32 with 5 generators>,
    Group([ f1*f2*f3*f4*f5, f3, f4, f5 ]) ],
  [ <pc group of size 32 with 5 generators>, Group([ f2, f3, f4, f5 ]) ],
  [ <pc group of size 32 with 5 generators>, Group([ f1, f3, f4, f5 ]) ],
  [ Group([ f1*f2*f3*f4*f5, f3, f4, f5 ]), Group([ f1*f2*f4*f5, f4, f5 ]) ],
  [ Group([ f2, f3, f4, f5 ]), Group([ f5 ]) ],
  [ Group([ f2, f3, f4, f5 ]), Group([ ]) ] ]
gap> ExtremelyStrongShodaPairs(SL(2,3));
[ [ SL(2,3), SL(2,3) ],
  [ SL(2,3),
    Group([ [ [ Z(3)^0, Z(3)^0 ], [ Z(3)^0, Z(3) ] ],
            [ [ Z(3), Z(3)^0 ], [ Z(3)^0, Z(3)^0 ] ],
            [ [ Z(3), 0*Z(3) ], [ 0*Z(3), Z(3) ] ] ]),
  [
    Group([ [ [ Z(3)^0, Z(3)^0 ], [ Z(3)^0, Z(3) ] ],
            [ [ Z(3), Z(3)^0 ], [ Z(3)^0, Z(3)^0 ] ],
            [ [ Z(3), 0*Z(3) ], [ 0*Z(3), Z(3) ] ] ]),
    Group([ [ [ 0*Z(3), Z(3) ], [ Z(3)^0, 0*Z(3) ] ],
            [ [ Z(3), 0*Z(3) ], [ 0*Z(3), Z(3) ] ] ] ) ] ]
gap> ExtremelyStrongShodaPairs(SymmetricGroup(5));
[ [ Sym([ 1 .. 5 ]), Sym([ 1 .. 5 ]) ],
  [ Sym([ 1 .. 5 ]), Alt([ 1 .. 5 ]) ] ]
```

3.2 Computing strong Shoda pairs

3.2.1 StrongShodaPairs

▷ StrongShodaPairs(G)

(attribute)

Returns: A list of pairs of subgroups of the input group.

The input should be a finite group G .

Computes a list of representatives of the equivalence classes of *strong Shoda pairs* (9.15) of a finite group G .

Example

```
gap> ssp:=StrongShodaPairs( SymmetricGroup(4) );;
gap> Length(ssp);
5
gap> List(ssp,x->List(x,StructureDescription));
[ [ "S4", "S4" ], [ "S4", "A4" ], [ "A4", "C2 x C2" ], [ "D8", "C2 x C2" ],
  [ "D8", "C4" ] ]
gap> ssp:=StrongShodaPairs( DihedralGroup(64) );;
gap> Length(ssp);
8
gap> List(ssp,x->List(x,StructureDescription));
[ [ "D64", "D64" ], [ "D64", "D32" ], [ "D64", "C32" ], [ "D64", "D32" ],
  [ "D32", "D16" ], [ "C32", "C4" ], [ "C32", "C2" ], [ "C32", "1" ] ]
```

3.3 Properties related with Shoda pairs

3.3.1 IsExtremelyStrongShodaPair

▷ IsExtremelyStrongShodaPair(G, K, H)

(operation)

The first argument should be a finite group G , the second one a normal subgroup K of G and the third one a subgroup of K .

Returns true if (K, H) is an *extremely strong Shoda pair* (9.16) of G , and false otherwise.

Example

```
gap> G:=SymmetricGroup(4);; K:=Group( (1,3,2,4), (3,4) );;
gap> H1:=Group( (2,4,3), (1,4)(2,3), (1,3)(2,4) );;
gap> H2:=Group( (3,4), (1,2)(3,4) );;
gap> IsExtremelyStrongShodaPair( G, G, H1 );
true
gap> IsExtremelyStrongShodaPair( G, K, H2 );
false
gap> IsExtremelyStrongShodaPair( G, G, H2 );
false
gap> IsExtremelyStrongShodaPair( G, G, K );
false
```

3.3.2 IsStrongShodaPair

▷ IsStrongShodaPair(G, K, H)

(operation)

The first argument should be a finite group G , the second one a subgroup K of G and the third one a subgroup of K .

Returns true if (K, H) is a *strong Shoda pair* (9.15) of G , and false otherwise.

Note that every extremely strong Shoda pair is a strong Shoda pair, but the converse is not true.

Example

```
gap> G:=SymmetricGroup(4);; K:=Group( (1,3,2,4), (3,4) );;
gap> H1:=Group( (2,4,3), (1,4)(2,3), (1,3)(2,4) );;
gap> H2:=Group( (3,4), (1,2)(3,4) );;
gap> IsStrongShodaPair( G, G, H1 );
true
gap> IsExtremelyStrongShodaPair( G, K, H2 );
false
gap> IsStrongShodaPair( G, K, H2 );
true
gap> IsStrongShodaPair( G, G, K );
false
```

3.3.3 IsShodaPair

▷ IsShodaPair(G , K , H)

(operation)

The first argument should be a finite group G , the second a subgroup K of G and the third one a subgroup of K .

Returns true if (K, H) is a *Shoda pair* (9.14) of G .

Note that every strong Shoda pair is a Shoda pair, but the converse is not true.

Example

```
gap> G:=AlternatingGroup(5);;
gap> K:=AlternatingGroup(4);;
gap> H := Group( (1,2)(3,4), (1,3)(2,4) );;
gap> IsStrongShodaPair( G, K, H );
false
gap> IsShodaPair( G, K, H );
true
```

3.3.4 IsStronglyMonomial

▷ `IsStronglyMonomial( $G$ )` (operation)

The input G should be a finite group.

Returns true if G is a *strongly monomial* (9.17) finite group.

Example

```
gap> S4:=SymmetricGroup(4);;
gap> IsStronglyMonomial(S4);
true
gap> G:=SmallGroup(24,3);;
gap> IsStronglyMonomial(G);
false
gap> IsMonomial(G);
false
gap> G:=SmallGroup(1000,86);;
gap> IsMonomial(G);
true
gap> IsStronglyMonomial(G);
false
```

3.3.5 IsNormallyMonomial

▷ `IsNormallyMonomial( $G$ )` (operation)

The input G should be a finite group.

Returns true if G is a finite *normally monomial* (9.18) group.

Example

```
gap> D24:=DihedralGroup(24);
<pc group of size 24 with 4 generators>
gap> IsNormallyMonomial(D24);
true
gap> G:=SmallGroup(192,1023);
<pc group of size 192 with 7 generators>
gap> IsNormallyMonomial(G);
true
gap> G:=SmallGroup(1029,12);
<pc group of size 1029 with 4 generators>
gap> IsNormallyMonomial(G);
false
gap> IsStronglyMonomial(G);
true
gap> G:=SL(2,3);
SL(2,3)
gap> IsNormallyMonomial(G);
false
gap> IsStronglyMonomial(G);
false
```

Chapter 4

Idempotents

4.1 Computing idempotents from character table

4.1.1 PrimitiveCentralIdempotentsByCharacterTable

▷ `PrimitiveCentralIdempotentsByCharacterTable(FG)` (operation)

Returns: A list of group algebra elements.

The input *FG* should be a semisimple group algebra.

Returns the list of primitive central idempotents of *FG* using the character table of *G* (9.4).

Example

```
gap> QS3 := GroupRing( Rationals, SymmetricGroup(3) );;
gap> PrimitiveCentralIdempotentsByCharacterTable( QS3 );
[ (1/6)*() + (-1/6)*(2,3) + (-1/6)*(1,2) + (1/6)*(1,2,3) + (1/6)*(1,3,2) + (-1/6)*(1,3),
  (2/3)*() + (-1/3)*(1,2,3) + (-1/3)*(1,3,2), (1/6)*() + (1/6)*(2,3) + (1/6)*(1,2) + (1/
  6)*(1,2,3) + (1/6)*(1,3,2) + (1/6)*(1,3) ]
gap> QG:=GroupRing( Rationals , SmallGroup(24,3) );
<algebra-with-one over Rationals, with 4 generators>
gap> FG:=GroupRing( CF(3) , SmallGroup(24,3) );
<algebra-with-one over CF(3), with 4 generators>
gap> pciQG := PrimitiveCentralIdempotentsByCharacterTable(QG);;
gap> pciFG := PrimitiveCentralIdempotentsByCharacterTable(FG);;
gap> Length(pciQG);
5
gap> Length(pciFG);
7
```

4.2 Testing lists of idempotents for completeness

4.2.1 IsCompleteSetOfOrthogonalIdempotents

▷ `IsCompleteSetOfOrthogonalIdempotents(R, list)` (operation)

The input should be formed by a unital ring *R* and a list *list* of elements of *R*.

Returns true if the list *list* is a complete list of orthogonal idempotents of *R*. That is, the output is true provided the following conditions are satisfied:

- The sum of the elements of *list* is the identity of *R*,

- $e^2 = e$, for every e in *list* and
- $e * f = 0$, if e and f are elements in different positions of *list*.

No claim is made on the idempotents being central or primitive.

Note that the if a non-zero element t of R appears in two different positions of *list* then the output is false, and that the list *list* must not contain zeroes.

Example

```
gap> QS5 := GroupRing( Rationals, SymmetricGroup(5) );;
gap> idemp := PrimitiveCentralIdempotentsByCharacterTable( QS5 );;
gap> IsCompleteSetOfOrthogonalIdempotents( QS5, idemp );
true
gap> IsCompleteSetOfOrthogonalIdempotents( QS5, [ One( QS5 ) ] );
true
gap> IsCompleteSetOfOrthogonalIdempotents( QS5, [ One( QS5 ), One( QS5 ) ] );
false
```

4.3 Idempotents from Shoda pairs

4.3.1 PrimitiveCentralIdempotentsByESSP

▷ PrimitiveCentralIdempotentsByESSP(QG)

(attribute)

Returns: A list of group algebra elements.

The input QG should be a semisimple rational group algebra of a finite group G .

The output is the list of primitive central idempotents of the group algebra QG realizable by extremely strong Shoda pairs (9.16) of G .

If the list of primitive central idempotents given by the output is not complete (i.e. if the group G is not *normally monomial* (9.18)) then a warning is displayed.

Example

```
gap> QG:=GroupRing( Rationals, DihedralGroup(16) );;
gap> PrimitiveCentralIdempotentsByESSP( QG );
[ (1/16)*<identity> of ...+(1/16)*f1+(1/16)*f2+(1/16)*f3+(1/16)*f4+(1/
16)*f1*f2+(1/16)*f1*f3+(1/16)*f1*f4+(1/16)*f2*f3+(1/16)*f2*f4+(1/
16)*f3*f4+(1/16)*f1*f2*f3+(1/16)*f1*f2*f4+(1/16)*f1*f3*f4+(1/
16)*f2*f3*f4+(1/16)*f1*f2*f3*f4, (1/16)*<identity> of ...+(-1/16)*f1+(-1/
16)*f2+(1/16)*f3+(1/16)*f4+(1/16)*f1*f2+(-1/16)*f1*f3+(-1/16)*f1*f4+(-1/
16)*f2*f3+(-1/16)*f2*f4+(1/16)*f3*f4+(1/16)*f1*f2*f3+(1/16)*f1*f2*f4+(-1/
16)*f1*f3*f4+(-1/16)*f2*f3*f4+(1/16)*f1*f2*f3*f4,
(1/16)*<identity> of ...+(-1/16)*f1+(1/16)*f2+(1/16)*f3+(1/16)*f4+(-1/
16)*f1*f2+(-1/16)*f1*f3+(-1/16)*f1*f4+(1/16)*f2*f3+(1/16)*f2*f4+(1/
16)*f3*f4+(-1/16)*f1*f2*f3+(-1/16)*f1*f2*f4+(-1/16)*f1*f3*f4+(1/
16)*f2*f3*f4+(-1/16)*f1*f2*f3*f4, (1/16)*<identity> of ...+(1/16)*f1+(-1/
16)*f2+(1/16)*f3+(1/16)*f4+(-1/16)*f1*f2+(1/16)*f1*f3+(1/16)*f1*f4+(-1/
16)*f2*f3+(-1/16)*f2*f4+(1/16)*f3*f4+(-1/16)*f1*f2*f3+(-1/16)*f1*f2*f4+(1/
16)*f1*f3*f4+(-1/16)*f2*f3*f4+(-1/16)*f1*f2*f3*f4,
(1/4)*<identity> of ...+(-1/4)*f3+(1/4)*f4+(-1/4)*f3*f4,
(1/2)*<identity> of ...+(-1/2)*f4 ]
gap> QG := GroupRing( Rationals, SmallGroup(24,12) );;
gap> PrimitiveCentralIdempotentsByESSP( QG );
Wedderga: Warning!!!
The output is a NON-COMPLETE list of prim. central idemp.s of the input!
[ (1/24)*<identity> of ...+(1/24)*f1+(1/24)*f2+(1/24)*f3+(1/24)*f4+(1/
```

$$\begin{aligned}
& 24) * f_1 * f_2 + (1/24) * f_1 * f_3 + (1/24) * f_1 * f_4 + (1/24) * f_2^2 + (1/24) * f_2 * f_3 + (1/ \\
& 24) * f_2 * f_4 + (1/24) * f_3 * f_4 + (1/24) * f_1 * f_2^2 + (1/24) * f_1 * f_2 * f_3 + (1/24) * f_1 * f_2 * f_4 + (1/ \\
& 24) * f_1 * f_3 * f_4 + (1/24) * f_2^2 * f_3 + (1/24) * f_2^2 * f_4 + (1/24) * f_2 * f_3 * f_4 + (1/24) * f_1 * f_2^2 * \\
& 2 * f_3 + (1/24) * f_1 * f_2^2 * f_4 + (1/24) * f_1 * f_2 * f_3 * f_4 + (1/24) * f_2^2 * f_3 * f_4 + (1/24) * f_1 * f_2^2 * \\
& 2 * f_3 * f_4, (1/24) * \langle \text{identity} \rangle \text{ of } \dots + (-1/24) * f_1 + (1/24) * f_2 + (1/24) * f_3 + (1/ \\
& 24) * f_4 + (-1/24) * f_1 * f_2 + (-1/24) * f_1 * f_3 + (-1/24) * f_1 * f_4 + (1/24) * f_2^2 + (1/ \\
& 24) * f_2 * f_3 + (1/24) * f_2 * f_4 + (1/24) * f_3 * f_4 + (-1/24) * f_1 * f_2^2 + (-1/24) * f_1 * f_2 * f_3 + (-1/ \\
& 24) * f_1 * f_2 * f_4 + (-1/24) * f_1 * f_3 * f_4 + (1/24) * f_2^2 * f_3 + (1/24) * f_2^2 * f_4 + (1/ \\
& 24) * f_2 * f_3 * f_4 + (-1/24) * f_1 * f_2^2 * f_3 + (-1/24) * f_1 * f_2^2 * f_4 + (-1/24) * f_1 * f_2 * f_3 * f_4 + (1/ \\
& 24) * f_2^2 * f_3 * f_4 + (-1/24) * f_1 * f_2^2 * f_3 * f_4, (1/6) * \langle \text{identity} \rangle \text{ of } \dots + (-1/12) * f_2 + (\\
& 1/6) * f_3 + (1/6) * f_4 + (-1/12) * f_2^2 + (-1/12) * f_2 * f_3 + (-1/12) * f_2 * f_4 + (1/6) * f_3 * f_4 + (-1/ \\
& 12) * f_2^2 * f_3 + (-1/12) * f_2^2 * f_4 + (-1/12) * f_2 * f_3 * f_4 + (-1/12) * f_2^2 * f_3 * f_4]
\end{aligned}$$

4.3.2 PrimitiveCentralIdempotentsByStrongSP

▷ PrimitiveCentralIdempotentsByStrongSP(*FG*)

(attribute)

Returns: A list of group algebra elements.

The input *FG* should be a semisimple group algebra of a finite group *G* whose coefficient field *F* is either a finite field or the field $\mathbb{Q}$ of rationals.

If $F = \mathbb{Q}$ then the output is the list of primitive central idempotents of the group algebra *FG* realizable by strong Shoda pairs (9.15) of *G*.

If *F* is a finite field then the output is the list of primitive central idempotents of *FG* realizable by strong Shoda pairs (*K, H*) of *G* and *q*-cyclotomic classes modulo the index of *H* in *K* (9.19).

If the list of primitive central idempotents given by the output is not complete (i.e. if the group *G* is not *strongly monomial* (9.17)) then a warning is displayed.

Example

```

gap> QG:=GroupRing( Rationals, AlternatingGroup(4) );;
gap> PrimitiveCentralIdempotentsByStrongSP( QG );
[ (1/12)*()+(1/12)*(2,3,4)+(1/12)*(2,4,3)+(1/12)*(1,2)(3,4)+(1/12)*(1,2,3)+(1/
12)*(1,2,4)+(1/12)*(1,3,2)+(1/12)*(1,3,4)+(1/12)*(1,3)(2,4)+(1/12)*
(1,4,2)+(1/12)*(1,4,3)+(1/12)*(1,4)(2,3),
(1/6)*()+(1/12)*(2,3,4)+(1/12)*(2,4,3)+(1/6)*(1,2)(3,4)+(1/12)*(1,2,3)+(
1/12)*(1,2,4)+(1/12)*(1,3,2)+(1/12)*(1,3,4)+(1/6)*(1,3)(2,4)+(1/12)*
(1,4,2)+(1/12)*(1,4,3)+(1/6)*(1,4)(2,3),
(3/4)*()+(1/4)*(1,2)(3,4)+(1/4)*(1,3)(2,4)+(1/4)*(1,4)(2,3) ]
gap> QG := GroupRing( Rationals, SmallGroup(24,3) );;
gap> PrimitiveCentralIdempotentsByStrongSP( QG );;
Wedderga: Warning!!!
The output is a NON-COMPLETE list of prim. central idemp.s of the input!
gap> FG := GroupRing( GF(2), Group((1,2,3)) );;
gap> PrimitiveCentralIdempotentsByStrongSP( FG );
[ (Z(2)^0)*()+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2),
(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2) ]
gap> FG := GroupRing( GF(5), SmallGroup(24,3) );;
gap> PrimitiveCentralIdempotentsByStrongSP( FG );;
Wedderga: Warning!!!
The output is a NON-COMPLETE list of prim. central idemp.s of the input!

```


4.3.3 PrimitiveCentralIdempotentsBySP

▷ PrimitiveCentralIdempotentsBySP(QG)

(function)

Returns: A list of group algebra elements.

The input should be a rational group algebra of a finite group G .

Returns a list containing all the primitive central idempotents e of the rational group algebra QG such that $\chi(e) \neq 0$ for some irreducible monomial character χ of G .

The output is the list of all primitive central idempotents of QG if and only if G is monomial, otherwise a warning message is displayed.

Example

```
gap> QG := GroupRing( Rationals, SymmetricGroup(4) );
<algebra-with-one over Rationals, with 2 generators>
gap> pci:=PrimitiveCentralIdempotentsBySP( QG );
[ (1/24)*()+ (1/24)*(3,4)+ (1/24)*(2,3)+ (1/24)*(2,3,4)+ (1/24)*(2,4,3)+ (1/24)*
(2,4)+ (1/24)*(1,2)+ (1/24)*(1,2)(3,4)+ (1/24)*(1,2,3)+ (1/24)*(1,2,3,4)+ (1/
24)*(1,2,4,3)+ (1/24)*(1,2,4)+ (1/24)*(1,3,2)+ (1/24)*(1,3,4,2)+ (1/24)*
(1,3)+ (1/24)*(1,3,4)+ (1/24)*(1,3)(2,4)+ (1/24)*(1,3,2,4)+ (1/24)*(1,4,3,2)+ (
1/24)*(1,4,2)+ (1/24)*(1,4,3)+ (1/24)*(1,4)+ (1/24)*(1,4,2,3)+ (1/24)*(1,4)
(2,3), (1/24)*()+ (-1/24)*(3,4)+ (-1/24)*(2,3)+ (1/24)*(2,3,4)+ (1/24)*
(2,4,3)+ (-1/24)*(2,4)+ (-1/24)*(1,2)+ (1/24)*(1,2)(3,4)+ (1/24)*(1,2,3)+ (-1/
24)*(1,2,3,4)+ (-1/24)*(1,2,4,3)+ (1/24)*(1,2,4)+ (1/24)*(1,3,2)+ (-1/24)*
(1,3,4,2)+ (-1/24)*(1,3)+ (1/24)*(1,3,4)+ (1/24)*(1,3)(2,4)+ (-1/24)*
(1,3,2,4)+ (-1/24)*(1,4,3,2)+ (1/24)*(1,4,2)+ (1/24)*(1,4,3)+ (-1/24)*(1,4)+ (
-1/24)*(1,4,2,3)+ (1/24)*(1,4)(2,3), (3/8)*()+ (-1/8)*(3,4)+ (-1/8)*(2,3)+ (
-1/8)*(2,4)+ (-1/8)*(1,2)+ (-1/8)*(1,2)(3,4)+ (1/8)*(1,2,3,4)+ (1/8)*
(1,2,4,3)+ (1/8)*(1,3,4,2)+ (-1/8)*(1,3)+ (-1/8)*(1,3)(2,4)+ (1/8)*(1,3,2,4)+ (
1/8)*(1,4,3,2)+ (-1/8)*(1,4)+ (1/8)*(1,4,2,3)+ (-1/8)*(1,4)(2,3),
(3/8)*()+ (1/8)*(3,4)+ (1/8)*(2,3)+ (1/8)*(2,4)+ (1/8)*(1,2)+ (-1/8)*(1,2)(3,4)+ (
-1/8)*(1,2,3,4)+ (-1/8)*(1,2,4,3)+ (-1/8)*(1,3,4,2)+ (1/8)*(1,3)+ (-1/8)*(1,3)
(2,4)+ (-1/8)*(1,3,2,4)+ (-1/8)*(1,4,3,2)+ (1/8)*(1,4)+ (-1/8)*(1,4,2,3)+ (-1/
8)*(1,4)(2,3), (1/6)*()+ (-1/12)*(2,3,4)+ (-1/12)*(2,4,3)+ (1/6)*(1,2)(3,4)+ (
-1/12)*(1,2,3)+ (-1/12)*(1,2,4)+ (-1/12)*(1,3,2)+ (-1/12)*(1,3,4)+ (1/6)*(1,3)
(2,4)+ (-1/12)*(1,4,2)+ (-1/12)*(1,4,3)+ (1/6)*(1,4)(2,3) ]
gap> IsCompleteSetOfPCIs(QG,pci);
true
gap> QS5 := GroupRing( Rationals, SymmetricGroup(5) );
gap> pci:=PrimitiveCentralIdempotentsBySP( QS5 );
Wedderga: Warning!!
The output is a NON-COMPLETE list of prim. central idemp.s of the input!
gap> IsCompleteSetOfPCIs( QS5 , pci );
false
```

The output of PrimitiveCentralIdempotentsBySP contains the output of PrimitiveCentralIdempotentsByStrongSP (4.3.2), possibly properly.

Example

```
gap> QG := GroupRing( Rationals, SmallGroup(48,28) );
gap> pci:=PrimitiveCentralIdempotentsBySP( QG );
Wedderga: Warning!!
The output is a NON-COMPLETE list of prim. central idemp.s of the input!
gap> Length(pci);
6
gap> spci:=PrimitiveCentralIdempotentsByStrongSP( QG );
```

```

Wedderga: Warning!!!
The output is a NON-COMPLETE list of prim. central idemp.s of the input!
gap> Length(spci);
5
gap> IsSubset(pci,spci);
true
gap> QG:=GroupRing(Rationals,SmallGroup(1000,86));
<algebra-with-one over Rationals, with 6 generators>
gap> IsCompleteSetOfPCIs( QG , PrimitiveCentralIdempotentsBySP(QG) );
true
gap> IsCompleteSetOfPCIs( QG , PrimitiveCentralIdempotentsByStrongSP(QG) );
Wedderga: Warning!!!
The output is a NON-COMPLETE list of prim. central idemp.s of the input!
false

```

4.4 Complete set of orthogonal primitive idempotents from Shoda pairs and cyclotomic classes

4.4.1 PrimitiveIdempotentsNilpotent

▷ `PrimitiveIdempotentsNilpotent( $FG$ ,  $H$ ,  $K$ ,  $C$ ,  $args$ )` (operation)

Returns: A list of orthogonal primitive idempotents.

The input FG should be a semisimple group algebra of a finite nilpotent group G whose coefficient field F is a finite field. H and K should form a strong Shoda pair (H, K) of G . $args$ is a list containing an epimorphism map epi from $N_G(K)$ to $N_G(K)/K$ and a generator gq of H/K . C is the $|F|$ -cyclotomic class modulo $[H : K]$ (w.r.t. the generator gq of H/K)

The output is a complete set of orthogonal primitive idempotents of the simple algebra $FGe_C(G, H, K)$ (9.22).

Example

```

gap> G:=DihedralGroup(8);;
gap> F:=GF(3);;
gap> FG:=GroupRing(F,G);;
gap> H:=StrongShodaPairs(G)[5][1];
Group([ f1*f2*f3, f3 ])
gap> K:=StrongShodaPairs(G)[5][2];
Group([ f1*f2 ])
gap> N:=Normalizer(G,K);
Group([ f1*f2*f3, f3 ])
gap> epi:=NaturalHomomorphismByNormalSubgroup(N,K);
[ f1*f2*f3, f3 ] -> [ f1, f1 ]
gap> QHK:=Image(epi,H);
Group([ f1, f1 ])
gap> gq:=MinimalGeneratingSet(QHK)[1];
f1
gap> C:=CyclotomicClasses(Size(F),Index(H,K))[2];
[ 1 ]
gap> PrimitiveIdempotentsNilpotent(FG,H,K,C,[epi,gq]);
[ (Z(3)^0)*<identity> of ...+(Z(3))*f3+(Z(3)^0)*f1*f2+(Z(3))*f1*f2*f3,
  (Z(3)^0)*<identity> of ...+(Z(3))*f3+(Z(3))*f1*f2+(Z(3)^0)*f1*f2*f3 ]

```

4.4.2 PrimitiveIdempotentsTrivialTwisting

▷ `PrimitiveIdempotentsTrivialTwisting(FG, H, K, C, args)` (operation)

Returns: A list of orthogonal primitive idempotents.

The input FG should be a semisimple group algebra of a finite group G whose coefficient field F is a finite field. H and K should form a strong Shoda pair (H, K) of G . $args$ is a list containing an epimorphism map epi from $N_G(K)$ to $N_G(K)/K$ and a generator gq of H/K . C is the $|F|$ -cyclotomic class modulo $[H : K]$ (w.r.t. the generator gq of H/K). The input parameters should be such that the simple component $FGe_C(G, H, K)$ has a trivial twisting.

The output is a complete set of orthogonal primitive idempotents of the simple algebra $FGe_C(G, H, K)$ (9.22).

Example

```
gap> G:=DihedralGroup(8);;
gap> F:=GF(3);;
gap> FG:=GroupRing(F,G);;
gap> H:=StrongShodaPairs(G)[5][1];
Group([ f1*f2*f3, f3 ])
gap> K:=StrongShodaPairs(G)[5][2];
Group([ f1*f2 ])
gap> N:=Normalizer(G,K);
Group([ f1*f2*f3, f3 ])
gap> epi:=NaturalHomomorphismByNormalSubgroup(N,K);
[ f1*f2*f3, f3 ] -> [ f1, f1 ]
gap> QHK:=Image(epi,H);
Group([ f1, f1 ])
gap> gq:=MinimalGeneratingSet(QHK)[1];
f1
gap> C:=CyclotomicClasses(Size(F),Index(H,K))[2];
[ 1 ]
gap> PrimitiveIdempotentsTrivialTwisting(FG,H,K,C,[epi,gq]);
[ (Z(3)^0)*<identity> of ...+(Z(3))*f3+(Z(3)^0)*f1*f2+(Z(3))*f1*f2*f3,
  (Z(3)^0)*<identity> of ...+(Z(3))*f3+(Z(3))*f1*f2+(Z(3)^0)*f1*f2*f3 ]
```

Chapter 5

Crossed products and their elements

The package `Wedderga` provides functions to construct crossed products over a group with coefficients in an associative ring with identity, and with the multiplication determined by a given action and twisting (see 9.6 for definitions). This can be done using the function `CrossedProduct` (5.1.1).

Note that this function does not check the associativity conditions, so in fact it is the NC-version of itself, and its output will be always assumed to be associative. For all crossed products that appear in `Wedderga` algorithms the associativity follows from theoretical arguments, so the usage of the NC-method in the package is safe. If the user will try to construct a crossed product with his own action and twisting, he/she should check the associativity conditions himself/herself to make sure that the result is correct.

5.1 Construction of crossed products

5.1.1 CrossedProduct

▷ `CrossedProduct( $R$ ,  $G$ ,  $act$ ,  $twist$ )` (attribute)

Returns: Ring in the category `IsCrossedProduct`.

The input should be formed by:

- an associative ring R ,
- a group G ,
- a function $act(RG, g)$ of two arguments: the crossed product RG and an element g in G . It must return a mapping from R to R which can be applied via the " $\wedge$ " operation, and
- a function $twist(RG, g, h)$ of three arguments: the crossed product RG and a pair of elements of G . It must return an invertible element of R .

Returns the crossed product of G over the ring R with action act and twisting $twist$.

The resulting crossed product belongs to the category `IsCrossedProduct`, which is defined as a subcategory of `IsFLMLORWithOne`.

An example of the trivial action:

Example

```
act := function(RG,a)
  return IdentityMapping( LeftActingDomain( RG ) );
end;
```

and the trivial twisting:

Example

```
twist := function( RG , g, h )
  return One( LeftActingDomain( RG ) );
end;
```

Let n be a positive integer and ξ_n an n -th complex primitive root of unity. The natural action of the group of units of $\mathbb{Z}_n$, the ring of integers modulo n , on $\mathbb{Q}(\xi_n)$ can be defined as follows:

Example

```
act := function(RG,a)
  return ANFAutomorphism( LeftActingDomain( RG ) , Int( a ) );
end;
```

In the following example one constructs the Hamiltonian quaternion algebra over the rationals as a crossed product of the group of units of the cyclic group of order 2 over $\mathbb{Q}(i) = \text{GaussianRationals}$. One realizes the cyclic group of order 2 as the group of units of $\mathbb{Z}/4\mathbb{Z}$ and one uses the natural isomorphism $\mathbb{Z}/4\mathbb{Z} \rightarrow \text{Gal}(\mathbb{Q}(i)/\mathbb{Q})$ to describe the action.

Example

```
gap> R := GaussianRationals;
GaussianRationals
gap> G := Units( ZmodnZ(4) );; Size( G );
2
gap> act := function(RG,g)
> return ANFAutomorphism( LeftActingDomain(RG), Int(g) );
> end;
function( RG, g ) ... end
gap> twist1 := function( RG, g, h )
> if IsOne(g) or IsOne(h) then
>   return One(LeftActingDomain(RG));
> else
>   return -One(LeftActingDomain(RG));
> fi;
> end;
function( RG, g, h ) ... end
gap> RG := CrossedProduct( R, G, act, twist1 );
<crossed product over GaussianRationals of a group of size 2>
gap> i := E(4) * One(G)^Embedding(G,RG);
(ZmodnZObj( 1, 4 ))*(E(4))
gap> j := ZmodnZObj(3,4)^Embedding(G,RG);
(ZmodnZObj( 3, 4 ))*(1)
gap> i^2;
(ZmodnZObj( 1, 4 ))*(-1)
gap> j^2;
(ZmodnZObj( 1, 4 ))*(-1)
gap> i*j+j*i;
<zero> of ...
```

One can construct the following generalized quaternion algebra with the same action and a different twisting

$$\mathbb{Q}(i, j | i^2 = -1, j^2 = -3, ji = -ij)$$

Example

```

gap> twist2:=function(RG,g,h)
> if IsOne(g) or IsOne(h) then
>   return One(LeftActingDomain( RG ));
> else
>   return -3*One(LeftActingDomain( RG ));
> fi;
> end;
function( RG, g, h ) ... end
gap> RG := CrossedProduct( R, G, act, twist2 );
<crossed product over GaussianRationals of a group of size 2>
gap> i := E(4) * One(G)^Embedding(G,RG);
(ZmodnZObj( 1, 4 ))*(E(4))
gap> j := ZmodnZObj(3,4)^Embedding(G,RG);
(ZmodnZObj( 3, 4 ))*(1)
gap> i^2;
(ZmodnZObj( 1, 4 ))*(-1)
gap> j^2;
(ZmodnZObj( 1, 4 ))*(-3)
gap> i*j+j*i;
<zero> of ...

```

The following example shows how to construct the Hamiltonian quaternion algebra over the rationals using the rationals as coefficient ring and the Klein group as the underlying group.

Example

```

gap> C2 := CyclicGroup(2);;
gap> G := DirectProduct(C2,C2);
<pc group of size 4 with 2 generators>
gap> act := function(RG,a)
>   return IdentityMapping( LeftActingDomain(RG));
> end;
function( RG, a ) ... end
gap> twist := function( RG, g , h )
> local one,g1,g2,h1,h2,G;
> G := UnderlyingMagma( RG );
> one := One( C2 );
> g1 := Image( Projection(G,1), g );
> g2 := Image( Projection(G,2), g );
> h1 := Image( Projection(G,1), h );
> h2 := Image( Projection(G,2), h );
> if g = One( G ) or h = One( G ) then return 1;
>   elif IsOne(g1) and not IsOne(g2) and not IsOne(h1) and not IsOne(h2)
>     then return 1;
>   elif not IsOne(g1) and IsOne(g2) and IsOne(h1) and not IsOne(h2)
>     then return 1;
>   elif not IsOne(g1) and not IsOne(g2) and not IsOne(h1) and IsOne(h2)
>     then return 1;
>   else return -1;
> fi;
> end;
function( RG, g, h ) ... end
gap> HQ := CrossedProduct( Rationals, G, act, twist );

```

```
<crossed product over Rationals of a group of size 4>
```

Changing the rationals by the integers as coefficient ring one can construct the Hamiltonian quaternion ring.

Example

```
gap> HZ := CrossedProduct( Integers, G, act, twist );
<crossed product over Integers of a group of size 4>
gap> i := GeneratorsOfGroup(G)[1]^Embedding(G,HZ);
(f1)*(1)
gap> j := GeneratorsOfGroup(G)[2]^Embedding(G,HZ);
(f2)*(1)
gap> i^2;
(<identity> of ...)*(-1)
gap> j^2;
(<identity> of ...)*(-1)
gap> i*j+j*i;
<zero> of ...
```

One can extract the arguments used for the construction of the crossed product using the following attributes:

- * LeftActingDomain for the coefficient ring.
- * UnderlyingMagma for the underlying group.
- * ActionForCrossedProduct for the action.
- * TwistingForCrossedProduct for the twisting.

Example

```
gap> LeftActingDomain(HZ);
Integers
gap> G:=UnderlyingMagma(HZ);
<pc group of size 4 with 2 generators>
gap> ac := ActionForCrossedProduct(HZ);
function( RG, a ) ... end
gap> List( G , x -> ac( HZ, x ) );
[ IdentityMapping( Integers ), IdentityMapping( Integers ),
  IdentityMapping( Integers ), IdentityMapping( Integers ) ]
gap> tw := TwistingForCrossedProduct( HZ );
function( RG, g, h ) ... end
gap> List( G , x -> List( G , y -> tw( HZ, x, y ) ) );
[ [ 1, 1, 1, 1 ], [ 1, -1, -1, 1 ], [ 1, 1, -1, -1 ], [ 1, -1, 1, -1 ] ]
```

Some more examples of crossed products arise from the *Wedderburn decomposition* (9.3) of group algebras.

Example

```
gap> G := SmallGroup(32,50);
<pc group of size 32 with 5 generators>
gap> A := SimpleAlgebraByCharacter( GroupRing(Rationals,G), Irr(G)[17] );
( <crossed product with center Rationals over GaussianRationals of a group of \
size 2>^[ 2, 2 ] )
gap> SimpleAlgebraByCharacterInfo( GroupRing(Rationals,G), Irr(G)[17] );
[ 2, Rationals, 4, [ 2, 3, 2 ] ]
gap> B := LeftActingDomain(A);
<crossed product with center Rationals over GaussianRationals of a group of si\
```

```

ze 2>
gap> L := LeftActingDomain(B);
GaussianRationals
gap> H := UnderlyingMagma( B );
<group of size 2 with 2 generators>
gap> Elements(H);
[ ZmodnZObj( 1, 4 ), ZmodnZObj( 3, 4 ) ]
gap> i := E(4) * One(H)^Embedding(H,B);
(ZmodnZObj( 1, 4 ))*(E(4))
gap> j := ZmodnZObj(3,4)^Embedding(H,B);
(ZmodnZObj( 3, 4 ))*(1)
gap> i^2;
(ZmodnZObj( 1, 4 ))*(-1)
gap> j^2;
(ZmodnZObj( 1, 4 ))*(-1)
gap> i*j+j*i;
<zero> of ...
gap> ac := ActionForCrossedProduct( B );
function( RG, a ) ... end
gap> tw := TwistingForCrossedProduct( B );
function( RG, a, b ) ... end
gap> List( H , x -> ac( B, x ) );
[ IdentityMapping( GaussianRationals ), ANFAutomorphism( GaussianRationals,
  3 ) ]
gap> List( H , x -> List( H , y -> tw( B, x, y ) ) );
[ [ 1, 1 ], [ 1, -1 ] ]

```

Example

```

gap> QG:=GroupRing( Rationals, SmallGroup(24,3) );;
gap> WedderburnDecomposition(QG);
[ Rationals, CF(3), ( Rationals^[ 3, 3 ] ),
  <crossed product with center Rationals over GaussianRationals of a group of \
size 2>, <crossed product with center CF(3) over AsField( CF(3), CF(
  12) ) of a group of size 2> ]
gap> R:=WedderburnDecomposition( QG )[4];
<crossed product with center Rationals over GaussianRationals of a group of si\
ze 2>
gap> IsCrossedProduct(R);
true
gap> IsAlgebra(R);
true
gap> IsRing(R);
true
gap> LeftActingDomain( R );
GaussianRationals
gap> AsList( UnderlyingMagma( R ) );
[ ZmodnZObj( 1, 4 ), ZmodnZObj( 3, 4 ) ]
gap> Print( ActionForCrossedProduct( R ) ); Print("\n");
function ( RG, a )
  local cond, redu;
  cond := OperationRecord( RG ).cond;
  redu := OperationRecord( RG ).redu;
  return

```



```

    ANFAutomorphism( CF( cond ), Int( PreImagesRepresentativeNC( redu, a ) ) );
end
gap> Print( TwistingForCrossedProduct( R ) ); Print("\n");
function ( RG, a, b )
  local orderroot, cocycle;
  orderroot := OperationRecord( RG ).orderroot;
  cocycle := OperationRecord( RG ).cocycle;
  return E( orderroot ) ^ Int( cocycle( a, b ) );
end
gap> IsAssociative(R);
true
gap> IsFinite(R);
false
gap> IsFiniteDimensional(R);
true
gap> AsList(Basis(R));
[ (ZmodnZObj( 1, 4 ))*(1), (ZmodnZObj( 3, 4 ))*(1) ]
gap> GeneratorsOfLeftOperatorRingWithOne(R);
[ (ZmodnZObj( 1, 4 ))*(1), (ZmodnZObj( 3, 4 ))*(1) ]
gap> One(R);
(ZmodnZObj( 1, 4 ))*(1)
gap> Zero(R);
<zero> of ...
gap> Characteristic(R);
0
gap> CenterOfCrossedProduct(R);
Rationals

```

The next example shows how one can use `CrossedProduct` to produce generalized quaternion algebras. Note that one can construct quaternion algebras using the GAP function `QuaternionAlgebra`.

Example

```

gap> Quat := function(R,a,b)
> local G,act,twist;
> if not(a in R and b in R and a <> Zero(R) and b <> Zero(R) ) then
> Error("<a> and <b> must be non zero elements of <R>!!!");
> fi;
> G := SmallGroup(4,2);
> act := function(RG,a)
>   return IdentityMapping( LeftActingDomain(RG));
> end;
> twist := function( RG, g , h )
> local one,g1,g2;
> one := One(G);
> g1 := G.1;
> g2 := G.2;
> if g = one or h = one then
>   return One(R);
> elif g = g1 then
>   if h = g2 then
>     return One(R);
>   else

```

```

>     return a;
>   fi;
> elif g = g2 then
>   if h = g1 then
>     return -One(R);
>   elif h=g2 then
>     return b;
>   else
>     return -b;
>   fi;
> else
>   if h = g1 then
>     return -b;
>   elif h=g2 then
>     return b;
>   else
>     return -a*b;
>   fi;
> fi;
> end;
> return CrossedProduct(R,G,act,twist);
> end;
function( R, a, b ) ... end
gap> HQ := Quat(Rationals,2,3);
<crossed product over Rationals of a group of size 4>
gap> G := UnderlyingMagma(HQ);
<pc group of size 4 with 2 generators>
gap> tw := TwistingForCrossedProduct( HQ );
function( RG, g, h ) ... end
gap> List( G, x -> List( G, y -> tw( HQ, x, y ) ) );
[ [ 1, 1, 1, 1 ], [ 1, 3, -1, -3 ], [ 1, 1, 2, 2 ], [ 1, 3, -3, -6 ] ]

```

5.2 Crossed product elements and their properties

5.2.1 ElementOfCrossedProduct

▷ `ElementOfCrossedProduct(Fam, zerocoeff, coeffs, elts)`

(property)

Returns the element $m_1 * c_1 + \dots + m_n * c_n$ of a crossed product, where $elts = [m_1, m_2, \dots, m_n]$ is a list of magma elements, $coeffs = [c_1, c_2, \dots, c_n]$ is a list of coefficients. The output belongs to the crossed product whose elements lie in the family *Fam*. The second argument *zerocoeff* must be the zero element of the coefficient ring containing coefficients c_i , and will be stored in the attribute `ZeroCoefficient` of the crossed product element.

The output will be in the category `IsElementOfCrossedProduct`, which is a subcategory of `IsRingElementWithInverse`. It will have the presentation `IsCrossedProductObjDefaultRep`.

Similarly to magma rings, one can obtain the list of coefficients and elements with `CoefficientsAndMagmaElements`.

Also note from the example below and several other examples in this chapter that instead of `ElementOfCrossedProduct` one can use `Embedding` to embed elements of the coefficient ring and

of the underlying magma into the crossed product.

Example

```
gap> QG := GroupRing( Rationals, SmallGroup(24,3) );
<algebra-with-one over Rationals, with 4 generators>
gap> R := WedderburnDecomposition( QG )[4];
<crossed product with center Rationals over GaussianRationals of a group of si\
ze 2>
gap> H := UnderlyingMagma( R );
gap> fam := ElementsFamily( FamilyObj( R ) );
gap> g := ElementOfCrossedProduct( fam, 0, [ 1, E(4) ], AsList(H) );
(ZmodnZObj( 1, 4 ))*(1)+(ZmodnZObj( 3, 4 ))*(E(4))
gap> CoefficientsAndMagmaElements( g );
[ ZmodnZObj( 1, 4 ), 1, ZmodnZObj( 3, 4 ), E(4) ]
gap> t := List( H, x -> x^Embedding( H, R ) );
[ (ZmodnZObj( 1, 4 ))*(1), (ZmodnZObj( 3, 4 ))*(1) ]
gap> t[1] + t[2]*E(4);
(ZmodnZObj( 1, 4 ))*(1)+(ZmodnZObj( 3, 4 ))*(E(4))
gap> g = t[1] + E(4)*t[2];
false
gap> g = t[1] + t[2]*E(4);
true
gap> h := ElementOfCrossedProduct( fam, 0, [ E(4), 1 ], AsList(H) );
(ZmodnZObj( 1, 4 ))*(E(4))+(ZmodnZObj( 3, 4 ))*(1)
gap> g+h;
(ZmodnZObj( 1, 4 ))*(1+E(4))+(ZmodnZObj( 3, 4 ))*(1+E(4))
gap> g*E(4);
(ZmodnZObj( 1, 4 ))*(E(4))+(ZmodnZObj( 3, 4 ))*(-1)
gap> E(4)*g;
(ZmodnZObj( 1, 4 ))*(E(4))+(ZmodnZObj( 3, 4 ))*(1)
gap> g*h;
(ZmodnZObj( 1, 4 ))*(2*E(4))
```

Chapter 6

Useful properties and functions

6.1 Semisimple group algebras of finite groups

6.1.1 IsSemisimpleZeroCharacteristicGroupAlgebra

▷ IsSemisimpleZeroCharacteristicGroupAlgebra(KG) (property)

The input must be a group ring.

Returns true if the input KG is a *semisimple group algebra* (9.2) over a field of characteristic zero (that is if G is finite), and false otherwise.

Example

```
gap> CG:=GroupRing( GaussianRationals, DihedralGroup(16) );;
gap> IsSemisimpleZeroCharacteristicGroupAlgebra( CG );
true
gap> FG:=GroupRing( GF(2), SymmetricGroup(3) );;
gap> IsSemisimpleZeroCharacteristicGroupAlgebra( FG );
false
gap> f := FreeGroup("a");
<free group on the generators [ a ]>
gap> Qf:=GroupRing(Rationals,f);
<algebra-with-one over Rationals, with 2 generators>
gap> IsSemisimpleZeroCharacteristicGroupAlgebra(Qf);
false
```

6.1.2 IsSemisimpleRationalGroupAlgebra

▷ IsSemisimpleRationalGroupAlgebra(KG) (property)

The input must be a group ring.

Returns true if KG is a *semisimple rational group algebra* (9.2) and false otherwise.

Example

```
gap> QG:=GroupRing( Rationals, SymmetricGroup(4) );;
gap> IsSemisimpleRationalGroupAlgebra( QG );
true
gap> CG:=GroupRing( GaussianRationals, DihedralGroup(16) );;
gap> IsSemisimpleRationalGroupAlgebra( CG );
false
```

```
gap> FG:=GroupRing( GF(2), SymmetricGroup(3) );
gap> IsSemisimpleRationalGroupAlgebra( FG );
false
```

6.1.3 IsSemisimpleANFGroupAlgebra

▷ IsSemisimpleANFGroupAlgebra(KG)

(property)

The input must be a group ring.

Returns true if KG is the group algebra of a finite group over a subfield of a cyclotomic extension of the rationals and false otherwise.

Example

```
gap> IsSemisimpleANFGroupAlgebra( GroupRing( NF(5,[4]) , CyclicGroup(28) ) );
true
gap> IsSemisimpleANFGroupAlgebra( GroupRing( GF(11) , CyclicGroup(28) ) );
false
```

6.1.4 IsSemisimpleFiniteGroupAlgebra

▷ IsSemisimpleFiniteGroupAlgebra(KG)

(property)

The input must be a group ring.

Returns true if KG is a *semisimple finite group algebra* (9.2), that is a group algebra of a finite group G over a field K of order coprime to the order of G , and false otherwise.

Example

```
gap> FG:=GroupRing( GF(5), SymmetricGroup(3) );
gap> IsSemisimpleFiniteGroupAlgebra( FG );
true
gap> KG:=GroupRing( GF(2), SymmetricGroup(3) );
gap> IsSemisimpleFiniteGroupAlgebra( KG );
false
gap> QG:=GroupRing( Rationals, SymmetricGroup(4) );
gap> IsSemisimpleFiniteGroupAlgebra( QG );
false
```

6.1.5 IsTwistingTrivial

▷ IsTwistingTrivial(G, H, K)

(property)

The input must be a group and a strong Shoda pair of the group.

Returns true if the simple algebra $\mathbb{Q}Ge(G, H, K)$ has a *trivial twisting* (9.15), and false otherwise.

Example

```
gap> G:=DihedralGroup(8);
gap> H:=StrongShodaPairs(G)[5][1];
Group([ f1*f2*f3, f3 ])
gap> K:=StrongShodaPairs(G)[5][2];
Group([ f1*f2 ])
gap> IsTwistingTrivial(G,H,K);
true
```

6.2 Operations with group rings elements

6.2.1 Centralizer

▷ `Centralizer( $G$ ,  $x$ )`

(operation)

Returns: A subgroup of a group G .

The input should be formed by a finite group G and an element x of a group ring FH whose underlying group H contains G as a subgroup.

Returns the centralizer of x in G .

This operation adds a new method to the operation that already exists in GAP.

Example

```
gap> D16 := DihedralGroup(16);
<pc group of size 16 with 4 generators>
gap> QD16 := GroupRing( Rationals, D16 );
<algebra-with-one over Rationals, with 4 generators>
gap> a:=QD16.1;b:=QD16.2;
(1)*f1
(1)*f2
gap> e := PrimitiveCentralIdempotentsByStrongSP( QD16)[3];;
gap> Centralizer( D16, a);
Group([ f1, f4 ])
gap> Centralizer( D16, b);
Group([ f2 ])
gap> Centralizer( D16, a+b);
Group([ f4 ])
gap> Centralizer( D16, e);
Group([ f1, f2 ])
```

6.2.2 OnPoints

▷ `OnPoints( $x$ ,  $g$ )`

(operation)

▷ `\^( $x$ ,  $g$ )`

(operation)

Returns: An element of a group ring.

The input should be formed by an element x of a group ring FG and an element g in the underlying group G of FG .

Returns the conjugate $x^g = g^{-1}xg$ of x by g . Usage of `x^g` produces the same output.

This operation adds a new method to the operation that already exists in GAP.

The following example is a continuation of the example from the description of `Centralizer` (6.2.1).

Example

```
gap> ForAll(D16,x->a^x=a);
false
gap> ForAll(D16,x->e^x=e);
true
```

6.2.3 AverageSum

▷ `AverageSum( $RG$ ,  $X$ )`

(operation)

Returns: An element of a group ring.

The input must be composed of a group ring RG and a finite subset X of the underlying group G of RG . The order of X must be invertible in the coefficient ring R of RG .

Returns the element of the group ring RG that is equal to the sum of all elements of X divided by the order of X .

If X is a subgroup of G then the output is an idempotent of RG which is central if and only if X is normal in G .

Example

```
gap> G:=DihedralGroup(16);;
gap> QG:=GroupRing( Rationals, G );;
gap> FG:=GroupRing( GF(5), G );;
gap> e:=AverageSum( QG, DerivedSubgroup(G) );
(1/4)*<identity> of ...+(1/4)*f3+(1/4)*f4+(1/4)*f3*f4
gap> f:=AverageSum( FG, DerivedSubgroup(G) );
(Z(5)^2)*<identity> of ...+(Z(5)^2)*f3+(Z(5)^2)*f4+(Z(5)^2)*f3*f4
gap> G=Centralizer(G,e);
true
gap> H:=Subgroup(G,[G.1]);
Group([ f1 ])
gap> e:=AverageSum( QG, H );
(1/2)*<identity> of ...+(1/2)*f1
gap> G=Centralizer(G,e);
false
gap> IsNormal(G,H);
false
```

6.3 Cyclotomic classes

6.3.1 CyclotomicClasses

▷ `CyclotomicClasses( $q, n$ )`

(operation)

Returns: A partition of $[0..n]$.

The input should be formed by two relatively prime positive integers.

Returns the list q -cyclotomic classes (9.19) modulo n .

Example

```
gap> CyclotomicClasses( 2, 21 );
[ [ 0 ], [ 1, 2, 4, 8, 16, 11 ], [ 3, 6, 12 ], [ 5, 10, 20, 19, 17, 13 ],
  [ 7, 14 ], [ 9, 18, 15 ] ]
gap> CyclotomicClasses( 10, 21 );
[ [ 0 ], [ 1, 10, 16, 13, 4, 19 ], [ 2, 20, 11, 5, 8, 17 ],
  [ 3, 9, 6, 18, 12, 15 ], [ 7 ], [ 14 ] ]
```

6.3.2 IsCyclotomicClass

▷ `IsCyclotomicClass( $q, n, C$ )`

(operation)

The input should be formed by two relatively prime positive integers q and n and a sublist C of $[0..n]$.

Returns true if C is a q -cyclotomic class (9.19) modulo n and false otherwise.

Example

```
gap> IsCyclotomicClass( 2, 7, [1,2,4] );
true
gap> IsCyclotomicClass( 2, 21, [1,2,4] );
false
gap> IsCyclotomicClass( 2, 21, [3,6,12] );
true
```

6.4 Other commands

6.4.1 InfoWedderga

▷ InfoWedderga

(info class)

InfoWedderga is a special Info class for Wedderga algorithms. It has 3 levels: 0, 1 (default) and 2. To change the info level to k , use the command `SetInfoLevel(InfoWedderga, k)`.

In the example below we use this mechanism to see more details about the Wedderburn components each time when we call `WedderburnDecomposition`.

Example

```
gap> SetInfoLevel(InfoWedderga, 2);
gap> WedderburnDecomposition( GroupRing( CF(5), DihedralGroup( 16 ) ) );
#I Info version : [ [ 1, CF(5) ], [ 1, CF(5) ], [ 1, CF(5) ],
[ 1, CF(5) ], [ 2, CF(5) ], [ 2, NF(40,[ 1, 31 ]) ] ]
[ CF(5), CF(5), CF(5), CF(5), ( CF(5)^[ 2, 2 ] ),
<crossed product with center NF(40,[ 1, 31 ]) over AsField( NF(40,
[ 1, 31 ]), CF(40) ) of a group of size 2> ]
```


Chapter 7

Functions for calculating Schur indices and identifying division algebras

7.1 Main Schur Index and Division Algebra Functions

7.1.1 WedderburnDecompositionWithDivAlgParts

▷ `WedderburnDecompositionWithDivAlgParts(A)` (property)

Returns: A list of lists $[r, D]$, each representing a ring of $r \times r$ matrices over a field or division algebra D .

The input A should be a group ring of a finite group over an abelian number field. The function will give the same result as `WedderburnDecompositionInfo` (2.1.2) if the field of coefficients for the group ring is finite. The output is a list of pairs $[r, D]$, each of which indicates a simple component isomorphic to the ring of $r \times r$ matrices over a division algebra described using the information in the record D . This record contains information on the center, Schur index, and local indices of the division algebra.

`LocalIndices` is a list of pairs $[p, m]$, where p is a rational prime (possibly 'infinity') and m is the local index of the division algebra at the prime p .

Example

```
gap> G:=SmallGroup(48,15);
<pc group of size 48 with 5 generators>
gap> R:=GroupRing(Rationals,G);
<algebra-with-one over Rationals, with 5 generators>
gap> WedderburnDecompositionInfo(R);
[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ],
  [ 2, Rationals ], [ 2, Rationals ], [ 2, Rationals ], [ 2, NF(8,[ 1, 7 ]) ],
  [ 2, CF(3) ], [ 1, Rationals, 12, [ [ 2, 5, 3 ], [ 2, 7, 0 ] ], [ [ 3 ] ] ]
]
gap> WedderburnDecompositionWithDivAlgParts(R);
[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals ],
  [ 2, Rationals ], [ 2, Rationals ], [ 2, Rationals ], [ 2, NF(8,[ 1, 7 ]) ],
  [ 2, CF(3) ],
  [ 2,
    rec( Center := Rationals, DivAlg := true,
          LocalIndices := [ [ 2, 2 ], [ 3, 2 ] ], SchurIndex := 2 ) ] ]
```

7.1.2 CyclotomicAlgebraWithDivAlgPart

▷ `CyclotomicAlgebraWithDivAlgPart(A)` (property)

Returns: A list of length two indicating a matrix ring of a given size over a field or a noncommutative division algebra.

The input A should be a cyclotomic algebra; i.e. a crossed product in the same form as in the output of `WedderburnDecompositionInfo` (2.1.2). The output is in the form $[r, D]$, which indicates an $r \times r$ matrix ring over the division algebra described by D . D is either a field or a noncommutative division algebra described using a record giving information on the center, Schur index, and local indices of the division algebra.

Example

```
gap> G:=SmallGroup(240,89);
<permutation group of size 240 with 2 generators>
gap> R:=GroupRing(Rationals,G);
<algebra-with-one over Rationals, with 2 generators>
gap> W:=WedderburnDecompositionInfo(R);
Wedderga: Warning!!!
Some of the Wedderburn components displayed are FRACTIONAL MATRIX ALGEBRAS!!!

[ [ 1, Rationals ], [ 1, Rationals ], [ 1, Rationals, 10, [ 4, 3, 5 ] ],
  [ 4, Rationals ], [ 4, Rationals ], [ 5, Rationals ], [ 5, Rationals ],
  [ 6, Rationals ], [ 1, NF(12,[ 1, 11 ]), 10, [ 4, 3, 5 ] ],
  [ 3/2, NF(8,[ 1, 7 ]), 10, [ 4, 3, 5 ] ] ]
gap> CyclotomicAlgebraWithDivAlgPart(W[3]);
[ 2, rec( Center := Rationals, DivAlg := true,
  LocalIndices := [ [ 5, 2 ], [ infinity, 2 ] ], SchurIndex := 2 ) ]
gap> CyclotomicAlgebraWithDivAlgPart(W[9]);
[ 2, rec( Center := NF(12,[ 1, 11 ]), DivAlg := true,
  LocalIndices := [ [ infinity, 2 ] ], SchurIndex := 2 ) ]
gap> CyclotomicAlgebraWithDivAlgPart(W[10]);
[ 3, rec( Center := NF(8,[ 1, 7 ]), DivAlg := true,
  LocalIndices := [ [ infinity, 2 ] ], SchurIndex := 2 ) ]
```

7.1.3 SchurIndex

▷ `SchurIndex(A)` (property)

▷ `SchurIndexByCharacter(F, G, n)` (operation)

Returns: The first of these returns the Schur index of the simple algebra A . The second returns the Schur index of the simple component of the group ring FG corresponding to the irreducible character $\text{Irr}(G)[n]$ of G .

These are the main functions for computing Schur indices. The first can be used to find the rational Schur index of a simple component of the group ring of a finite group over an abelian number field, or a quaternion algebra in GAP (see `QuaternionAlgebra` (**Reference: QuaternionAlgebra**)) whose center is the field of rational numbers. If A is a quaternion algebra over a number field other than the Rationals, fail is returned. In these cases, the quaternion algebra can be converted to a cyclic algebra and the Schur index of the cyclic algebra can be determined through the solution of norm equations. Currently this functionality is not implemented in GAP, but available in number theory packages such as PARI/GP.

The second function computes the Schur index of the cyclotomic algebra that would occur as the simple component of the group ring FG that corresponds to the irreducible character $\text{Irr}(G)[n]$.

The function uses `SimpleComponentByCharacterDescent` (7.3.2), which uses the character descent algorithm to reduce to as small a group as possible. For larger groups this is preferable as it is less demanding on memory. The Schur index of the resulting cyclotomic algebra is then computed with the `SchurIndex` function.

Example

```
gap> G:=SmallGroup(63,1);
<pc group of size 63 with 3 generators>
gap> R:=GroupRing(Rationals,G);
<algebra-with-one over Rationals, with 3 generators>
gap> W:=WedderburnDecompositionInfo(R);
[ [ 1, Rationals ], [ 1, CF(3) ], [ 1, CF(9) ], [ 3, NF(7,[ 1, 2, 4 ]) ],
  [ 1, NF(21,[ 1, 4, 16 ]) ], 21, [ 3, 4, 7 ] ] ]
gap> SchurIndex(W[5]);
3
gap> G:=SmallGroup(40,3);
<pc group of size 40 with 4 generators>
gap> i:=First([1..Length(Irr(G)),i->Size(KernelOfCharacter(Irr(G)[i]))=1]);
gap> SchurIndexByCharacter(GaussianRationals,G,Irr(G)[i]);
2
gap> SchurIndexByCharacter(CF(3),G,i);
1
```

Example

```
gap> G:=AtlasGroup("J2");
<permutation group of size 604800 with 2 generators>
gap> Irr(G)[20];
gap> SchurIndexByCharacter(Rationals,G,Irr(G)[20]);
1
```

7.1.4 WedderburnDecompositionAsSCAlgebras

- ▷ `WedderburnDecompositionAsSCAlgebras(R)` (operation)
- ▷ `CyclotomicAlgebraAsSCAlgebra(A)` (operation)
- ▷ `SimpleComponentByCharacterAsSCAlgebra(F, G, n)` (operation)

Returns: The first of these returns the Wedderburn decomposition of the group ring R with each simple component presented as an algebra with structure constants in **GAP** (see **(Reference: Constructing Algebras by Structure Constants)** in the main **GAP** manual). The second converts a list A that is output from `WedderburnDecompositionInfo` (2.1.2) into an algebra with structure constants in **GAP**. The third determines an algebra with structure constants that is isomorphic to the simple component of the group ring of the finite group G over the field F that corresponds to the irreducible character $\text{Irr}(G)[n]$.

These functions are an option for obtaining a Wedderburn decomposition or simple component of the group ring FG in which the output is in the form of an algebra with structure constants, which is more compatible with **GAP**'s built-in operations for finite-dimensional algebras.

Example

```
gap> G:=SmallGroup(63,1);
<pc group of size 63 with 3 generators>
gap> R:=GroupRing(Rationals,G);
<algebra-with-one over Rationals, with 3 generators>
gap> W:=WedderburnDecompositionInfo(R);
[ [ 1, Rationals ], [ 1, CF(3) ], [ 1, CF(9) ], [ 3, NF(7,[ 1, 2, 4 ]) ],
```

```

[ 1, NF(21,[ 1, 4, 16 ]), 21, [ 3, 4, 7 ] ] ]
gap> WedderburnDecompositionWithDivAlgParts(R);
[ [ 1, Rationals ], [ 1, CF(3) ], [ 1, CF(9) ], [ 3, NF(7,[ 1, 2, 4 ]) ],
  [ 1,
    rec( Center := NF(21,[ 1, 4, 16 ]), DivAlg := true,
        LocalIndices := [ [ 7, 3 ] ], SchurIndex := 3 ) ] ]
gap> WedderburnDecompositionAsSCAlgebras(R);
[ Rationals, CF(3), CF(9), ( NF(7,[ 1, 2, 4 ])^[ 3, 3 ] ),
  <algebra of dimension 9 over NF(21,[ 1, 4, 16 ])> ]
gap> CyclotomicAlgebraAsSCAlgebra(W[5]);
<algebra of dimension 9 over NF(21,[ 1, 4, 16 ])>
gap> Size(Irr(G));
15
gap> SimpleComponentByCharacterAsSCAlgebra(Rationals,G,15);
<algebra of dimension 9 over NF(21,[ 1, 4, 16 ])>

```

7.2 Cyclotomic Reciprocity Functions

7.2.1 PPartOfN

- ▷ PPartOfN(n , p) (operation)
 ▷ PDashPartOfN(n , p) (operation)

These are standard arithmetic functions required by several subroutines for the cyclotomic reciprocity and Schur index functions in Wedderga.

Example

```

gap> PPartOfN(2275,5);
25
gap> PDashPartOfN(2275,5);
91

```

7.2.2 PSplitSubextension

- ▷ PSplitSubextension(F , n , p) (operation)

Returns: The maximal subextension K of the cyclotomic extension $F(E(n))/F$ for which K/F splits completely at the prime p .

This function finds the maximal subextension K of the cyclotomic extension $F(E(n))$ of an abelian number field F for which both the ramification index and residue degree of K/F over any prime lying over p are 1. To do this, it finds the field fixed by an appropriate power of the field automorphism inducing the local Frobenius automorphism.

Example

```

gap> PSplitSubextension(Rationals,60,5);
GaussianRationals
gap> PSplitSubextension(NF(5,[1,4]),70,2);
NF(35,[ 1, 4, 9, 11, 16, 29 ])
gap> PSplitSubextension(NF(40,[1,9,11,19]),20,2);
NF(40,[ 1, 9, 11, 19])

```

7.2.3 SplittingDegreeAtP

- ▷ SplittingDegreeAtP(F, n, p) (operation)
- ▷ ResidueDegreeAtP(F, n, p) (operation)
- ▷ RamificationIndexAtP(F, n, p) (operation)

Returns: The splitting degree, residue degree, and ramification index of the extension $F(E(n))/F$ at the prime p .

These functions calculate the cyclotomic reciprocity parameters g , f , and e for the extension $F(E(n))/F$ at the prime p for an abelian number field F . To do this, it finds the p -split subextension K and the p -dash part n' of n , then calculates $g = [K:F]$, $f = [K(E(n')):K]$, and $e = [K(E(n)):K(E(n'))]$. These functions enable the user to calculate cyclotomic reciprocity parameters for any extension of abelian number fields, as the example illustrates.

Example

```
gap> F:=CF(12);
CF(12)
gap> K:=NF(120,[1,49]) # Note that F is a subfield of K, with index 4.
> ; # Then we can find e, f, and g for the extension K/F at the prime 5.
NF(120,[ 1, 49 ])
gap> RamificationIndexAtP(F,120,5); RamificationIndexAtP(K,120,5); last2/last;
4
2
2
gap> ResidueDegreeAtP(F,120,5); ResidueDegreeAtP(K,120,5); last2/last;
1
1
1
gap> SplittingDegreeAtP(F,120,5); SplittingDegreeAtP(K,120,5); last2/last;
2
1
2
```

7.3 Global Splitting and Character Descent Functions

7.3.1 GlobalSplittingOfCyclotomicAlgebra

- ▷ GlobalSplittingOfCyclotomicAlgebra(A) (operation)
- ▷ KillingCocycle(A) (operation)
- ▷ AntiSymMatUpMat(x) (operation)
- ▷ CyclotomicExtensionGenerator(K, F) (operation)
- ▷ ReducingCyclotomicAlgebra(A) (operation)

Returns: If the input cyclotomic crossed product algebra A admits a rescaling of basis elements by roots of unity that makes it equivalent to one with a trivial factor set, the first function here returns a full matrix algebra over the same center with the same dimension as A . In general the function will output a cyclotomic algebra equivalent to A whose factor set does not admit a partial splitting by a simple rescaling of basis elements by roots of unity. KillingCocycle is a

subroutine used by `GlobalSplittingOfCyclotomicAlgebra` function to produce equivalent factor sets that have more zeroes, `AntiSymMatUpMat` is a subroutine it uses to manipulate the factor set information in anti-symmetric matrix form. `ReducingCyclotomicAlgebra` is another subroutine used by the `GlobalSplittingOfCyclotomicAlgebra` that either returns an isomorphic cyclotomic algebra whose associated Galois group has fewer generators than the input, or returns fail. `CyclotomicExtensionGenerator` is a subroutine called by `ReducingCyclotomicAlgebra` which returns a positive integer m if the field K is the cyclotomic extension of F obtained by adjoining a primitive m -th root of 1, and returns fail otherwise.

The input A must be a list representing a cyclotomic algebra in the same form as in the output of `WedderburnDecompositionInfo` (2.1.2) or `SimpleAlgebraByCharacterInfo` (2.2.2). If the output has length 5, the function will first search using `KillingCocycle` and `AntiSymMatUpMat` for a rescaling of basis elements that makes the factor set trivial, and in general finds one that produces as many zeroes as possible in the factor set information. If there is a generator that produces a split tensor factor with a cyclotomic algebra B defined over a root of unity of smaller order, then `ReducingCyclotomicAlgebra` reduces to B . These steps are repeated until such tensor factorizations are no longer available.

Example

```
gap> A := [ 1, Rationals, 20, [ [ 2, 11, 0 ], [ 4, 3, 0 ] ], [ [ 0 ] ] ];
gap> GlobalSplittingOfCyclotomicAlgebra(A);
[ 8, Rationals ]
gap> A := [ 1, Rationals, 20, [ [ 2, 11, 0 ], [ 4, 3, 10 ] ], [ [ 0 ] ] ];
gap> GlobalSplittingOfCyclotomicAlgebra(A);
[ 2, Rationals, 10, [ 4, 3, 5 ] ]
```

7.3.2 CharacterDescent

- ▷ `CharacterDescent( $F, G, n, e, H$ )` (operation)
- ▷ `GlobalCharacterDescent( $F, G, n$ )` (operation)
- ▷ `SimpleComponentByCharacterDescent( $F, G, n$ )` (operation)

Returns: The first function returns $[r, F_1, H, k]$ where r is a positive rational, F_1 is an extension field of F , H is the input subgroup H , and k is the position in $\text{Irr}(H)$ of the irreducible character that admits a Global Character Descent reduction; i.e. an irreducible character of H that has the same local indices as $\text{Irr}(G)[n]$. The second function returns the limit of the Character Descent reductions, it returns the last `CharacterDescent` output obtained before reaching a state where no maximal subgroup of G admits a further reduction by `CharacterDescent`. The third function returns the simple component of FG resulting from `GlobalCharacterDescent` that is produced by applying `SimpleComponentOfGroupRingByCharacter` (7.5.3) and adjusting the matrix degree.

The `CharacterDescent` function tries to find an irreducible character $\psi = \text{Irr}(H)[k]$ of H for which $[F(\chi, \psi), F(\chi)]$ and (χ_H, ψ) are both coprime to $|G|$. This character ψ will have the same local indices over F as χ by a theorem of Yamada. The `GlobalCharacterDescent` function iterates a search for these global character descent reductions over irreducible characters of maximal subgroups of G , and when it finds one it replaces $G, F(\chi)$, and χ by $H, F(\chi, \psi)$, and ψ , and then begins a search over maximal subgroups of H . It terminates when no maximal subgroup admits a global reduction, and returns the relevant matrix degree factor along with the last H and ψ it found. `SimpleComponentByCharacterDescent` implements an algorithm that returns the simple component of FG associated with $\text{Irr}(G)[n]$ obtained using the global character descent algorithm.

Example

```
gap> G:=PSU(3,3);
<permutation group of size 6048 with 2 generators>
gap> sc := SimpleComponentByCharacterDescent(Rationals,G,8);;
gap> sc{[1..3]}; # the 4th entry is [ 2, 5, 3 ] or [ 2, 5, 9 ]
[ 21/2, GaussianRationals, 12 ]
gap> SchurIndex(sc);
1
```

7.3.3 GaloisRepsOfCharacters

▷ `GaloisRepsOfCharacters(F, G)` (operation)

Returns: A list of positive integers n that indicates the positions in $\text{Irr}(G)$ of the Galois conjugacy classes of irreducible characters of G over the field F .

`GaloisRepsOfCharacters` finds a list of representatives of the distinct Galois conjugacy classes of irreducible characters of G over F . It runs through the irreducible characters and determines if a given irreducible is Galois conjugate over F to any of the previous ones, and if not it adds the position of that character to the list.

Example

```
gap> G:=SmallGroup(63,1);
<pc group of size 63 with 3 generators>
gap> Length(GaloisRepsOfCharacters(Rationals,G));
5
gap> Length(GaloisRepsOfCharacters(CF(9),G));
12
gap> Length(GaloisRepsOfCharacters(NF(7,[1,2,4]),G));
7
gap> Length(GaloisRepsOfCharacters(CF(63),G));
15
```

7.3.4 WedderburnDecompositionByCharacterDescent

▷ `WedderburnDecompositionByCharacterDescent(F, G)` (operation)

Returns: A Wedderburn decomposition of FG that is obtained by applying the global character descent algorithm to representatives of the Galois conjugacy classes of irreducible characters of G over F .

`WedderburnDecompositionByCharacterDescent` is available as an option to `WedderburnDecompositionInfo` (2.1.2) that can be more effective for larger groups, especially ones with relatively few irreducible characters. As long as GAP is able to compute maximal subgroups and restrict irreducible characters of G to them, this function should eventually return a Wedderburn decomposition. As it operates on one irreducible character at a time, it is less effective on Abelian groups, which have far too many irreducible characters, and on most groups in GAP's Small Groups Library, where memory limitations are not a factor and it is possible for the Shoda pair algorithms can deal with many irreducible characters at the same time.

Example

```
gap> G:=GL(3,3);
GL(3,3)
gap> Size(G);
11232
```

```
gap> WedderburnDecompositionByCharacterDescent(Rationals,G);
[ [ 1, Rationals ], [ 1, Rationals ], [ 12, Rationals ], [ 12, Rationals ],
[ 13, Rationals ], [ 13, Rationals ], [ 16, NF(13,[ 1, 3, 9 ]) ],
[ 16, NF(13,[ 1, 3, 9 ]) ], [ 26, Rationals ], [ 26, Rationals ],
[ 26, NF(8,[ 1, 3 ]) ], [ 26, NF(8,[ 1, 3 ]) ], [ 27, Rationals ],
[ 27, Rationals ], [ 39, Rationals ], [ 39, Rationals ] ]
```

7.4 Local index functions for Cyclic Cyclotomic Algebras

7.4.1 LocalIndicesOfCyclicCyclotomicAlgebra

▷ LocalIndicesOfCyclicCyclotomicAlgebra(A) (operation)

Returns: A list of the pairs $[p, m]$ indicating the nontrivial local indices m at the primes p of the cyclic cyclotomic algebra indicated by A .

The input A must be a list representing a cyclic cyclotomic algebra in the same form as in the output of WedderburnDecompositionInfo (2.1.2) or SimpleAlgebraByCharacterInfo (2.2.2). This function computes the local Schur indices at rational primes p using the specialized functions for cyclic cyclotomic algebras described in this section.

Example

```
gap> A:=[1,Rationals,6,[2,5,3]];
[ 1, Rationals, 6, [ 2, 5, 3 ] ]
gap> LocalIndicesOfCyclicCyclotomicAlgebra(A);
[ [ 3, 2 ], [ infinity, 2 ] ]
```

7.4.2 LocalIndexAtInfnty

▷ LocalIndexAtInfnty(A) (operation)

▷ LocalIndexAtTwo(A) (operation)

▷ LocalIndexAtOddP(A, p) (operation)

Returns: These return the local index of the cyclic cyclotomic algebra A at the indicated rational prime.

The input A must be a cyclic cyclotomic algebra; that is, a list of the form $[r, F, n, [a, b, c]]$ that indicates a cyclic cyclotomic crossed product algebra. This is a special case of the output of wedderga's WedderburnDecompositionInfo (2.1.2) or SimpleAlgebraByCharacterInfo (2.2.2). For the LocalIndexAtOddP function, p must be an odd prime. The functions PPartOfN (7.2.1) and PDashPartOfN (7.2.1) are standard (and self-explanatory) arithmetic functions for a positive integer n and prime p .

These functions determine the local index of a cyclic cyclotomic algebra at the rational primes 'infinity', 2, or odd primes p , respectively. The first two functions check for a relationship of A to a nonsplit real or 2-adic quaternion algebra. LocalIndexAtOddP calculates the local index at p by counting the number of roots of unity coprime to p found in the p -adic completion, and using a formula due to Janusz.

Example

```
gap> A:=[1,CF(4),20,[4,13,15]];
[ 1, GaussianRationals, 20, [ 4, 13, 15 ] ]
gap> LocalIndexAtOddP(A,5);
4
gap> A:=[1,NF(8,[1,7]),8,[2,7,4]];
[ 1, NF(8,[ 1, 7 ]), 8, [ 2, 7, 4 ] ]
```



```

[ 1, NF(8,[ 1, 7 ]), 8, [ 2, 7, 4 ] ]
gap> LocalIndexAtInfty(A);
2
gap> A:=[1,CF(7),28,[2,15,14]];
[ 1, CF(7), 28, [ 2, 15, 14 ] ]
gap> LocalIndexAtTwo(A);
2

```

7.5 Local index functions for Non-Cyclic Cyclotomic Algebras

7.5.1 LocalIndicesOfCyclotomicAlgebra

▷ LocalIndicesOfCyclotomicAlgebra(A) (operation)

Returns: A list of pairs $[p, m]$ indicating the nontrivial local indices m at the primes p of the cyclic cyclotomic algebra indicated by A .

The input A should be a cyclotomic algebra; i.e. a list of length 2, 4, or 5 in the form of the output by Wedderga's “-Info” functions. If the cyclotomic algebra A is represented by a list of length 2, the local indices are all 1, so the function will return an empty list. If the cyclotomic algebra A is given by a list of length 4, then it represents a cyclic cyclotomic algebra, so the function LocalIndicesOfCyclicCyclotomicAlgebra (7.4.1) is utilized. If the cyclotomic algebra A is presented as a list of length 5 or more, the function first applies GlobalSplittingOfCyclotomicAlgebra (7.3.1) to reduce the length as much as possible. If this does not reduce the length to 4 or less, it applies the character descent algorithm to try to reduce it again with Clifford theory: it determines the group and character χ that faithfully represent the algebra using DefiningGroupOfCyclotomicAlgebra (7.5.3) and DefiningCharacterOfCyclotomicAlgebra (7.5.3), then applies SimpleComponentByCharacterDescent (7.3.2). It repeats this until it cannot reduce the length of cyclotomic algebra any longer. If the length is 4 it will apply the local index functions for cyclic cyclotomic algebras to compute the local indices at each prime dividing the order of the group. If the length is 5 or more, it applies the character-theoretic local Schur index functions to the output $[G, \chi]$ of DefiningGroupAndCharacterOfCyclotAlg (7.5.3). It uses the Frobenius-Schur indicator of χ to determine the local index at infinity (see LocalIndexAtInftyByCharacter (7.5.4)). For local indices at odd primes and sometimes for the prime 2, the defect group of the block containing χ will be cyclic, so the local index can be found using the values of a Brauer character by a theorem of Benard (see LocalIndexAtPByBrauerCharacter (7.5.6).) Sometimes for the prime 2 the defect group is not necessarily cyclic, so in these cases we appeal to the classification of dyadic Schur groups by Schmid and Riese (see LocalIndexAtTwoByCharacter (7.5.7)).

Example

```

gap> G:=SmallGroup(480,600);
<pc group of size 480 with 7 generators>
gap> W:=WedderburnDecompositionInfo(GroupRing(Rationals,G));;
gap> Size(W);
27
gap> W[27];
[ 1, NF(5,[ 1, 4 ]), 60, [ [ 2, 11, 0 ], [ 2, 19, 30 ], [ 2, 31, 30 ] ],
  [ [ 0, 45 ], [ 15 ] ] ]

```

```
gap> LocalIndicesOfCyclotomicAlgebra(W[27]);
[ [ infinity, 2 ] ]
```

Example

```
gap> G:=SmallGroup(160,82);
<pc group of size 160 with 6 generators>
gap> W:=WedderburnDecompositionInfo(GroupRing(Rationals,G));;
gap> Size(W);
14
gap> W[14];
[ 1, Rationals, 20, [ [ 2, 11, 0 ], [ 4, 3, 0 ] ], [ [ 5 ] ] ]
gap> LocalIndicesOfCyclotomicAlgebra(W[14]);
[ [ 2, 2 ], [ 5, 2 ] ]
```

7.5.2 RootOfDimensionOfCyclotomicAlgebra

▷ RootOfDimensionOfCyclotomicAlgebra(A) (operation)

Returns: A positive integer representing the square root of the dimension of the cyclotomic algebra over its center.

Example

```
gap> A:=[3,Rationals,12,[[2,5,3],[2,7,0]],[[3]]];
[ 3, Rationals, 12, [ [ 2, 5, 3 ], [ 2, 7, 0 ] ], [ [ 3 ] ] ]
gap> RootOfDimensionOfCyclotomicAlgebra(A);
12
```

7.5.3 DefiningGroupOfCyclotomicAlgebra

▷ DefiningGroupOfCyclotomicAlgebra(A) (operation)

▷ DefiningCharacterOfCyclotomicAlgebra(A) (operation)

▷ DefiningGroupAndCharacterOfCyclotAlg(A) (operation)

Returns: These functions return a finite group G and a positive integer n (or, in the case to the third function, an irreducible character χ of G for which the simple component of a group algebra over G over the center of the cyclotomic algebra A corresponding to the character $\text{Irr}(G)[n]$ (resp. χ) will be (rationally) isomorphic to A .

▷ SimpleComponentOfGroupRingByCharacter(F, G, χ) (operation)

Returns: A list that describes the algebraic structure of the simple component of the group algebra FG which corresponds to the irreducible character $\text{Irr}(G)[n]$ (or χ).

This function is an alternative to `SimpleAlgebraByCharacterInfo(GroupRing(F,G), Irr(G)[n])`; . It is used in subroutines of local index functions when we need to work over a field larger than the field of character values.

Example

```
gap> G:=SmallGroup(48,15);
<pc group of size 48 with 5 generators>
gap> R:=GroupRing(Rationals,G);
<algebra-with-one over Rationals, with 5 generators>
gap> W:=WedderburnDecompositionInfo(R);;
gap> A:=W[10];
[ 1, Rationals, 12, [ [ 2, 5, 3 ], [ 2, 7, 0 ] ], [ [ 3 ] ] ]
gap> g:=DefiningGroupOfCyclotomicAlgebra(A);;
gap> IdSmallGroup(g);
```

```
[ 48, 15 ]
gap> n:=DefiningCharacterOfCyclotomicAlgebra(A);
12
gap> SimpleComponentOfGroupRingByCharacter(Rationals,G,n)
> ;#Note:this cyclotomic algebra is isomorphic to the other by a change of basis.
[ 1, Rationals, 12, [ [ 2, 5, 3 ], [ 2, 7, 0 ] ], [ [ 3 ] ] ]
```

7.5.4 LocalIndexAtInfntyByCharacter

▷ `LocalIndexAtInfntyByCharacter(F, G, n)` (operation)

Returns: The local index at an infinite prime of the field F of the irreducible character $\text{Irr}(G)[n]$ (or n) of the finite group G .

This function computes the Frobenius-Schur indicator of the irreducible character $\text{Irr}(G)[n]$, and uses it to calculate the local index at infinity of the corresponding simple component of FG .

Example

```
gap> G:=SmallGroup(48,16);
<pc group of size 48 with 5 generators>
gap> i:=First([1..Length(Irr(G)),i->Size(KernelOfCharacter(Irr(G)[i]))=1]);
gap> LocalIndexAtInfntyByCharacter(Rationals,G,i);
2
gap> LocalIndexAtInfntyByCharacter(CF(3),G,Irr(G)[i]);
1
```

7.5.5 DefectGroupOfConjugacyClassAtP

▷ `DefectGroupOfConjugacyClassAtP(G, c, p)` (operation)

▷ `DefectGroupsOfPBlock(G, n, p)` (operation)

▷ `DefectOfCharacterAtP(G, n, p)` (operation)

Returns: The first of these functions returns a defect group of the c -th conjugacy class of the finite group G at the prime p . The second returns the conjugacy class of p -subgroups of G that consists of defect groups for the p -block containing the ordinary irreducible character $\text{Irr}(G)[n]$. The last of these functions returns the nonnegative integer d for which p^d is the order of a p -defect group for $\text{Irr}(G)[n]$.

The p -defect group of a given conjugacy class of G is a p -Sylow subgroup of the centralizer in G of any representative of the class. A defect group for a p -block of G is a minimal p -subgroup that is a defect group for a defect class of the block. By Brauer's Min-Max theorem, this will occur for at least one p -regular class of G . The function `DefectGroupsOfPBlock` identifies the defect classes for the block containing $\text{Irr}(G)[n]$, finds the one whose defect group has minimal order, and returns the conjugacy class of the defect group of this class. The function `DefectOfCharacterAtP` gives the logarithm base p of the order of a defect group of the p -block containing the character $\text{Irr}(G)[n]$.

Example

```
gap> G:=SmallGroup(72,21);
<pc group of size 72 with 5 generators>
gap> i:=First([1..Length(Irr(G)),i->Size(KernelOfCharacter(Irr(G)[i]))=1]);
gap> D:=DefectGroupOfConjugacyClassAtP(G,i,3);
Group([ f4, f5 ])
gap> IsCyclic(last);
false
```

```

gap> D:=DefectGroupsOfPBlock(G,Irr(G)[i],3);
Group( [ f4, f5 ] )^G
gap> IsCyclic(Representative(D));
false
gap> DefectOfCharacterAtP(G,Irr(G)[i],3);
2

```

7.5.6 LocalIndexAtPByBrauerCharacter

▷ `LocalIndexAtPByBrauerCharacter(F, G, n, p)` (operation)

▷ `FinFieldExt(F, G, p, n, m)` (operation)

Returns: The first returns the local index at the rational prime p of the simple component of the group ring FG that corresponds to $\text{Irr}(G)[n]$. The second returns the degree of a certain extension of finite fields of p -power order.

The input of `LocalIndexAtPByBrauerCharacter` must be an abelian number field F , a finite group G , and the number n of an ordinary irreducible character $\text{Irr}(G)[n]$, and p a prime divisor of the order of G . Since this function is intended to be used for faithful characters of groups that are the defining groups of non-cyclic cyclotomic algebras that result from Wedderga's Info functions, it is expected that G is a non-nilpotent cyclic-by-abelian group, and $\text{Irr}(G)[n]$ is a faithful character. The Brauer character table records of such groups can be accessed in GAP (provided G is sufficiently small).

The local index calculation uses Benard's theorem, which shows that the local index at p of the simple component of the rational group algebra QG corresponding to the character $\text{Irr}(G)[n]$ is the degree of the extension of the residue field of the center given by adjoining an irreducible p -Brauer character $\text{IBr}(G, p)[m]$ lying in the same block, provided the defect group of the block is cyclic. If the defect group of the block is not cyclic, the resulting calculation is unreliable, and the function will output a list whose second term is the warning label "DGnotCyclic". The degree of this finite field extension is calculated by `FinFieldExt`. It determines the local index relative to the field F by dividing the local index at p over the rationals by a constant determined using a theorem of Yamada.

Example

```

gap> G:=SmallGroup(80,28);
<pc group of size 80 with 5 generators>
gap> T:=CharacterTable(G);;
gap> i:=PositionProperty(Irr(T),chi->Size(KernelOfCharacter(chi))=1);;
gap> S:=T mod 5;
BrauerTable( <pc group of size 80 with 5 generators>, 5 )
gap> LocalIndexAtPByBrauerCharacter(Rationals,G,i,5);
2
gap> j:=PositionProperty(Irr(S),phi->Conductor(phi)=8);;
gap> FinFieldExt(Rationals,G,5,i,j);
2

```

Example

```

gap> G:=SmallGroup(72,20);
<pc group of size 72 with 5 generators>
gap> i:=First([1..Length(Irr(G))],i->Size(KernelOfCharacter(Irr(G)[i]))=1);;
gap> LocalIndexAtPByBrauerCharacter(Rationals,G,Irr(G)[i],3);
[ 2, "DGnotCyclic" ]
gap> LocalIndexAtPByBrauerCharacter(Rationals,G,i,2);
1

```

7.5.7 LocalIndexAtOddPByCharacter

- ▷ LocalIndexAtOddPByCharacter(F, G, n, p) (operation)
- ▷ LocalIndexAtTwoByCharacter(F, G, n) (operation)
- ▷ IsDyadicSchurGroup(G) (operation)

Returns: The first two function determines the local index at the given prime p of the simple component of FG corresponding to the irreducible character $\text{Irr}(G)[n]$. The third one returns 'true' if G is a dyadic Schur group, and otherwise 'false'.

LocalIndexAtOddPByCharacter and LocalIndexAtTwoByCharacter first determine a cyclotomic algebra representing the simple component of FG corresponding to the character $\text{Irr}(G)[n]$. They then extend the field F to K , where K is the maximal p -split subextension of $F(E(n))/F$, and recalculates the simple component of KG corresponding to $\text{Irr}(G)[n]$. It then uses the DefiningGroup... functions to reduce to a faithful character of a possibly smaller cyclic-by-abelian group. If the simple component for this character is given in Wedderga as a list of length 2 or 4, they make use of LocalIndexAtOddP (7.4.2) or LocalIndexAtTwo (7.4.2) as appropriate. If the simple component over F has length 5, it checks if the defect group of the p -block containing $\text{Irr}(G)[n]$ is cyclic. If this is definitely so, they use LocalIndexAtPByBrauerCharacter (7.5.6) to calculate the p -local index. Exceptions can occur when p is 2. When the defect group is not necessarily cyclic, LocalIndexAtTwoByCharacter makes use of IsDyadicSchurGroup, which checks if a quasi-elementary group has a faithful irreducible character 2-local index 2, then verifies that K does not split the simple component generated by this character.

These functions are designed for faithful characters of groups that faithfully represent cyclotomic algebras, and so should be used with caution in other situations.

Example

```
gap> G:=SmallGroup(48,15);
<pc group of size 48 with 5 generators>
gap> i:=First([1..Length(Irr(G))],i->Size(KernelOfCharacter(Irr(G)[i]))=1);
gap> LocalIndexAtOddPByCharacter(Rationals,G,Irr(G)[i],3);
2
gap> LocalIndexAtTwoByCharacter(Rationals,G,Irr(G)[i]);
2
gap> LocalIndexAtTwoByCharacter(CF(3),G,Irr(G)[i]);
1
```

7.6 Local index functions for Rational Quaternion Algebras

7.6.1 LocalIndicesOfRationalQuaternionAlgebra

- ▷ LocalIndicesOfRationalQuaternionAlgebra(A) (operation)
- ▷ LocalIndicesOfRationalSymbolAlgebra(a, b) (operation)
- ▷ LocalIndicesOfTensorProductOfQuadraticAlgs(L, M) (operation)
- ▷ GlobalSchurIndexFromLocalIndices(L) (operation)

Returns: The first of these functions return a list of pairs $[p,m]$ indicating that m is the local index at the prime p for the given quaternion algebra. The second does the same for QuaternionAlgebra(Rationals, a, b). The third returns a list of local indices computed from two given lists of local indices, and the fourth returns the least common multiple of the local indices in the given list of local indices.

For the first function, the input must be a quaternion algebra over the rationals, output from `QuaternionAlgebra(Rationals,a,b)`. For the first function, a and b can be any pair of integers, and for the second rational symbol algebra version, a and b are restricted to be either -1 or positive prime integers, otherwise it will return `fail`. The input of the third function is a pair of lists of p -local indices in which the maximum local index at any prime is at most 2. The input of the fourth function is a list of pairs $[p,m]$ in which each prime that appears only appears in one of the pairs, and the m 's that appear are all positive integers.

`LocalIndicesOfRationalQuaternionAlgebra` first factors the algebra as a tensor product of rational quaternion algebras, obtaining suitable pairs a and b to which `LocalIndicesOfRationalSymbolAlgebra` can be applied. The local indices are calculated using well-known formulas involving the Legendre Symbol. The local indices of the original algebra are then determined using `LocalIndicesOfTensorProductOfQuadraticAlgs`, which takes a pair of lists of local indices of quadratic algebras - for which the maximum local index at any prime p is 2, and finds the list of local indices of the tensor product of two algebras with these local indices.

`GlobalSchurIndexFromLocalIndices` simply computes the least common multiple of the local indices at each prime that occurs in the list.

Example

```
gap> LocalIndicesOfRationalSymbolAlgebra(-1,-1);
[ [ infinity, 2 ], [ 2, 2 ] ]
gap> LocalIndicesOfRationalSymbolAlgebra(3,-1);
[ [ 2, 2 ], [ 3, 2 ] ]
gap> LocalIndicesOfRationalSymbolAlgebra(-3,2);
fail
gap> LocalIndicesOfRationalSymbolAlgebra(3,7);
[ [ 2, 2 ], [ 7, 2 ] ]
gap> A:=QuaternionAlgebra(Rationals,-30,-15);
<algebra-with-one of dimension 4 over Rationals>
gap> LocalIndicesOfRationalQuaternionAlgebra(A);
[ [ 5, 2 ], [ infinity, 2 ] ]
gap> A:=QuaternionAlgebra(CF(5),3,-2);
<algebra-with-one of dimension 4 over CF(5)>
gap> LocalIndicesOfRationalQuaternionAlgebra(A);
fail
```

7.6.2 IsRationalQuaternionAlgebraADivisionRing

▷ `IsRationalQuaternionAlgebraADivisionRing(A)` (operation)

Returns: If the rational quaternion algebra is a noncommutative division ring, `true` is returned, and if otherwise, `false`.

The input A must be a quaternion algebra over the rationals, as output from `QuaternionAlgebra(Rationals,a,b)`. a and b must be rational integers. When applied to other algebras, it returns `fail`.

The function calculates the rational Schur index of the algebra using `LocalIndicesOfRationalQuaternionAlgebra` (7.6.1), and returns `true` if the rational Schur index of the algebra is 2, and `false` if the rational Schur index is 1.

This function should be preferred over GAP's `IsDivisionRing` (**Reference:** `IsDivisionRing`) when dealing with rational quaternion algebras, since the result of latter function only depends on the local index at infinity for quaternion algebras, and makes no use of the local indices at the finite primes.

Example

```

gap> A:=QuaternionAlgebra(Rationals,-30,-15);
<algebra-with-one of dimension 4 over Rationals>
gap> IsRationalQuaternionAlgebraADivisionRing(A);
true
gap> LocalIndicesOfRationalQuaternionAlgebra(A);
[ [ 5, 2 ], [ infinity, 2 ] ]
gap> A:=QuaternionAlgebra(Rationals,3,-2);
<algebra-with-one of dimension 4 over Rationals>
gap> IsRationalQuaternionAlgebraADivisionRing(A);
false
gap> LocalIndicesOfRationalQuaternionAlgebra(A);
[ ]

```

7.7 Functions involving Cyclic Algebras

Cyclic algebras are represented in *Wedderga* as lists of length 3, in the form $[F, K, [c]]$, which stands for a cyclic crossed product algebra of the form $(K/F, c)$, with K/F a cyclic galois extension of abelian number fields, and c an element of F determining the factor set. Schur indices of cyclic algebras can be determined through the solution of inverse norm equations in general. Though currently algorithms for this are not available in *GAP*, algorithms have been implemented in some computational number theory software systems such as *PARI/GP*.

The functions in this section allow one to convert cyclotomic algebras into cyclic algebras (or possibly as tensor products of two cyclic algebras), to convert generalized quaternion algebras into quadratic algebras (i.e. cyclic algebras for a Galois extension of degree 2), to convert quadratic algebras into generalized quaternion algebras, and to convert cyclic algebras into cyclic cyclotomic algebras, whenever possible.

7.7.1 DecomposeCyclotomicAlgebra

▷ `DecomposeCyclotomicAlgebra(A)` (operation)

Returns: Two lists, each representing a cyclic algebra over the center of A , whose tensor product is isomorphic to the cyclotomic algebra described by A .

The input must be list representing a cyclotomic algebra of length 5 whose Galois group has 2 generators. This is represented in *Wedderga* as a list of the form $[r, F, n, [[m_1, k_1, l_1], [m_2, k_2, l_2]], [[d]]]$. (Longer presentations of cyclotomic algebras do occur in *Wedderga* output. Currently we do not have a general decomposition algorithm for them.)

For these algebras, the extension $F(E(n))/F$ is the tensor product of two disjoint extensions K_1 and K_2 of F , and the program adjusts one of the factor sets (corresponding to l_1 or l_2) so that d becomes 0. After this adjustment, the algebra is then the tensor product of cyclic algebras of the form $[F, K_1, [c_1]]$ and $[F, K_2, [c_2]]$ provided c_1 and c_2 lie in F . If the latter condition is not satisfied, the string “fails” is appended to the output. (We have not encountered this problem among the group algebras of small groups we have tested so far.)

Example

```

gap> G:=SmallGroup(96,35);
<pc group of size 96 with 6 generators>
gap> W:=WedderburnDecompositionInfo(GroupRing(Rationals,G));;
gap> Size(W);

```

```

12
gap> A:=W[12];
[ 1, NF(8,[ 1, 7 ]), 24, [ [ 2, 7, 12 ], [ 2, 17, 9 ] ], [ [ 3 ] ] ]
gap> DecomposeCyclotomicAlgebra(A);
[ [ NF(8,[ 1, 7 ]), CF(8), [ -1 ] ],
  [ NF(8,[ 1, 7 ]), NF(24,[ 1, 7 ]), [ E(8)+2*E(8)^2+E(8)^3 ] ] ]

```

7.7.2 ConvertCyclicAlgToCyclicCyclotomicAlg

▷ ConvertCyclicAlgToCyclicCyclotomicAlg(A) (operation)

Returns: A list of the form $[1, F, n, [a, b, c]]$ which represents a cyclic cyclotomic algebra.

This function converts a cyclic algebra given by a list $[F, F(E(n)), [E(n)^c]]$ to an isomorphic cyclic cyclotomic algebra represented as the list $[1, F, n, [a, b, c]]$.

▷ ConvertQuadraticAlgToQuaternionAlg(A) (operation)

Returns: A generalized quaternion algebra.

The input should be a list of the form $[F, K, [c]]$ where the field K must be obtained by adjoining the square root of a nonsquare element d of F . The function then returns the quaternion algebra given in GAP by QuaternionAlgebra(F, d, c);.

Example

```

gap> A:=[NF(24,[1,11]),CF(24),[-1]];
[ NF(24,[ 1, 11 ]), CF(24), [ -1 ] ]
gap> ConvertCyclicAlgToCyclicCyclotomicAlg(A);
[ 1, NF(24,[ 1, 11 ]), 24, [ 2, 11, 12 ] ]
gap> LocalIndicesOfCyclicCyclotomicAlgebra(last);
[ ]
gap> ConvertQuadraticAlgToQuaternionAlg(A);
<algebra-with-one of dimension 4 over NF(24,[ 1, 11 ])>
gap> b:=Basis(last);; b[1]^2; b[2]^2; b[3]^2; b[4]^2;
e
(-1)*e
(-1)*e
(-1)*e
gap> b[2]*b[3]+b[3]*b[2];
0*e

```

7.7.3 ConvertQuaternionAlgToQuadraticAlg

▷ ConvertQuaternionAlgToQuadraticAlg(A) (operation)

Returns: A list of the form $[F, K, [c]]$ representing a cyclic algebra for which the degree of the extension K/F is 2.

The input must be a quaternion algebra whose center is an abelian number field F , presented as in the output from QuaternionAlgebra(F, a, b), with a, b in F . It returns a list $[F, F(ER(a)), [b]]$ representing the cyclic algebra isomorphic to A .

▷ ConvertCyclicCyclotomicAlgToCyclicAlg(A) (operation)

Returns: A list of the form $[F, K, [c]]$.

The input should be a list $[r, F, n, [a, b, c]]$ representing a matrix ring over a cyclic cyclotomic algebra. The function returns the list $[F, F(E(n)), [E(n)^c]]$, which represents a cyclic algebra that is Morita equivalent to the given cyclic cyclotomic algebra.

Example

```

gap> A:=QuaternionAlgebra(CF(5),-3,-1);
<algebra-with-one of dimension 4 over CF(5)>
gap> ConvertQuaternionAlgToQuadraticAlg(A);
[ CF(5), CF(15), [ -1 ] ]
gap> ConvertCyclicAlgToCyclicCyclotomicAlg(last);
[ 1, CF(5), 30, [ 2, 11, 15 ] ]
gap> SchurIndex(last);
1
gap> ConvertCyclicCyclotomicAlgToCyclicAlg(last2);
[ 1, [ CF(5), CF(15), [ -1 ] ] ]
gap> ConvertQuadraticAlgToQuaternionAlg(last[2]);
<algebra-with-one of dimension 4 over CF(5)>
gap> b:=Basis(last);; b[1]^2; b[2]^2; b[3]^2; b[4]^2;
e
(-3)*e
(-1)*e
(-3)*e

```

Chapter 8

Applications of the Wedderga package

8.1 Coding theory applications

8.1.1 CodeWordByGroupRingElement

▷ `CodeWordByGroupRingElement(F, S, a)` (operation)

Returns: The code word of length the length of S associated to the group ring element a .

The input F should be a finite field. The input S is a fixed ordering of a group G and a is an element in the group algebra FG .

Each element c in FG is of the form $c = \sum_{i=1}^n f_i g_i$, where we fix an ordering $\{g_1, g_2, \dots, g_n\}$ of the group elements of G and $f_i \in F$. If we look at c as a codeword, we will write $[f_1 f_2 \dots f_n]$. (9.23).

Example

```
gap> G:=DihedralGroup(8);;
gap> F:=GF(3);;
gap> FG:=GroupRing(F,G);;
gap> a:=AsList(FG)[27];
(Z(3)^0)*<identity> of ...+(Z(3)^0)*f1+(Z(3)^0)*f2+(Z(3)^0)*f3+(Z(3)^
0)*f1*f2+(Z(3)^0)*f2*f3+(Z(3))*f1*f2*f3
gap> S:=AsSet(G);
[ <identity> of ..., f1, f2, f3, f1*f2, f1*f3, f2*f3, f1*f2*f3 ]
gap> CodeWordByGroupRingElement(F,S,a);
[ Z(3)^0, Z(3)^0, Z(3)^0, Z(3)^0, Z(3)^0, 0*Z(3), Z(3)^0, Z(3) ]
```

8.1.2 CodeByLeftIdeal

▷ `CodeByLeftIdeal(F, G, S, I)` (operation)

Returns: All code words of length the length of S associated to the group ring elements in the ideal I of FG .

The input F should be a finite field. The input S is a fixed ordering of a group G and I is a left ideal of the group algebra FG .

Each element c in FG is of the form $c = \sum_{i=1}^n f_i g_i$, where we fix an ordering $\{g_1, g_2, \dots, g_n\}$ of the group elements of G and $f_i \in F$. If we look at c as a codeword, we will write $[f_1 f_2 \dots f_n]$. (9.23).

Example

```
gap> G:=DihedralGroup(8);;
gap> F:=GF(3);;
gap> FG:=GroupRing(F,G);;
```

```

gap> S:=AsSet(G);
[ <identity> of ..., f1, f2, f3, f1*f2, f1*f3, f2*f3, f1*f2*f3 ]
gap> H:=StrongShodaPairs(G)[5][1];
Group([ f1*f2*f3, f3 ])
gap> K:=StrongShodaPairs(G)[5][2];
Group([ f1*f2 ])
gap> N:=Normalizer(G,K);
Group([ f1*f2*f3, f3 ])
gap> epi:=NaturalHomomorphismByNormalSubgroup(N,K);
[ f1*f2*f3, f3 ] -> [ f1, f1 ]
gap> QHK:=Image(epi,H);
Group([ f1, f1 ])
gap> gq:=MinimalGeneratingSet(QHK)[1];
f1
gap> C:=CyclotomicClasses(Size(F),Index(H,K))[2];
[ 1 ]
gap> e:=PrimitiveIdempotentsNilpotent(FG,H,K,C,[epi,gq]);
[ (Z(3)^0)*<identity> of ...+(Z(3))*f3+(Z(3)^0)*f1*f2+(Z(3))*f1*f2*f3,
  (Z(3)^0)*<identity> of ...+(Z(3))*f3+(Z(3))*f1*f2+(Z(3)^0)*f1*f2*f3 ]
gap> FGe := LeftIdealByGenerators(FG,[e[1]]);;
gap> V := VectorSpace(F,CodeByLeftIdeal(F,G,S,FGe));;
gap> B := Basis(V);;
gap> LoadPackage("guava");;
gap> code := GeneratorMatCode(B,F);
a linear [8,2,1..4]4..5 code defined by generator matrix over GF(3)
gap> MinimumDistance(code);
4

```

Chapter 9

The basic theory behind Wedderga

In this chapter we describe the theory that is behind the algorithms used by Wedderga.

All the rings considered in this chapter are associative and have an identity.

We use the following notation: $\mathbb{Q}$ denotes the field of rationals and $\mathbb{F}_q$ the finite field of order q . For every positive integer k , we denote a complex k -th primitive root of unity by ξ_k and so $\mathbb{Q}(\xi_k)$ is the k -th cyclotomic extension of $\mathbb{Q}$.

9.1 Group rings and group algebras

Given a group G and a ring R , the *group ring* RG over the group G with coefficients in R is the ring whose underlying additive group is a right R -module with basis G such that the product is defined by the following rule

$$(gr)(hs) = (gh)(rs)$$

for $r, s \in R$ and $g, h \in G$, and extended to RG by linearity.

A *group algebra* is a group ring in which the coefficient ring is a field.

9.2 Semisimple group algebras

We say that a ring R is semisimple if it is a direct sum of simple left (alternatively right) ideals or equivalently if R is isomorphic to a direct product of simple algebras each one isomorphic to a matrix ring over a division ring.

By Maschke's Theorem, if G is a finite group then the group algebra FG is semisimple if and only if the characteristic of the coefficient field F does not divide the order of G .

In fact, an arbitrary group ring RG is semisimple if and only if the coefficient ring R is semisimple, the group G is finite and the order of G is invertible in R .

Some authors use the notion semisimple ring for rings with zero Jacobson radical. To avoid confusion we usually refer to semisimple rings as semisimple artinian rings.

9.3 Wedderburn components

If R is a *semisimple ring* (9.2) then the *Wedderburn decomposition* of R is the decomposition of R as a direct product of simple algebras. The factors of this Wedderburn decomposition are called

Wedderburn components of R . Each Wedderburn component of R is of the form Re for e a *primitive central idempotent* (9.4) of R .

Let FG be a *semisimple group algebra* (9.2). If F has positive characteristic, then the Wedderburn components of FG are matrix algebras over finite extensions of F . If F has zero characteristic then by the *Brauer-Witt Theorem* [Yam74], the *Wedderburn components* of FG are *Brauer equivalent* (9.5) to *cyclotomic algebras* (9.11).

The main functions of **Wedderga** compute the Wedderburn components of a semisimple group algebra FG , such that the coefficient field is either an abelian number field (i.e. a subfield of a finite cyclotomic extension of the rationals) or a finite field. In the finite case, the Wedderburn components are matrix algebras over finite fields and so can be described by the size of the matrices and the size of the finite field.

In the zero characteristic case each Wedderburn component A is *Brauer equivalent* (9.5) to a *cyclotomic algebra* (9.11) and therefore A is a (possibly fractional) matrix algebra over *cyclotomic algebra* and can be described numerically in one of the following three forms:

$$[n, K],$$

$$[n, K, k, [d, \alpha, \beta]],$$

$$[n, K, k, [d_i, \alpha_i, \beta_i]_{i=1}^m, [\gamma_{i,j}]_{1 \leq i < j \leq n}],$$

where n is the matrix size, K is the centre of A (a finite field extension of F) and the remaining data are integers whose interpretation is explained in 9.12.

In some cases (for the zero characteristic coefficient field) the size n of the matrix algebras is not a positive integer but a positive rational number. This is a consequence of the fact that the *Brauer-Witt Theorem* [Yam74] only ensures that each *Wedderburn component* (9.3) of a semisimple group algebra is Brauer equivalent (9.5) to a *cyclotomic algebra* (9.11), but not necessarily isomorphic to a full matrix algebra of a cyclotomic algebra. For example, a Wedderburn component D of a group algebra can be a division algebra but not a cyclotomic algebra. In this case $M_n(D)$ is a cyclotomic algebra C for some n and therefore D can be described as $M_{1/n}(C)$ (see last Example in *WedderburnDecomposition* (2.1.1)).

The main algorithm of **Wedderga** is based on a computational oriented proof of the Brauer-Witt Theorem due to Olteanu [Olt07] which uses previous work by Olivieri, del Río and Simón [OdS04] (see also [Od03]) for rational group algebras of *strongly monomial groups* (9.17). The algorithms are also based upon the work of Bakshi and Maheshwary [BM14] (see also [BM16]) on the rational group algebras of *normally monomial groups* (9.18).

9.4 Characters and primitive central idempotents

A *primitive central idempotent* of a ring R is a non-zero central idempotent e which cannot be written as the sum of two non-zero central idempotents of Re , or equivalently, such that Re is indecomposable as a direct product of two non-trivial two-sided ideals.

The *Wedderburn components* (9.3) of a semisimple ring R are the rings of the form Re for e running over the set of primitive central idempotents of R .

Let FG be a *semisimple group algebra* (9.2) and χ an irreducible character of G (in an algebraic closure of F). Then there is a unique Wedderburn component $A = A_F(\chi)$ of FG such that $\chi(A) \neq 0$.

Let $e_F(\chi)$ denote the unique primitive central idempotent of FG in $A_F(\chi)$, that is the identity of $A_F(\chi)$, i.e.

$$A_F(\chi) = FG e_F(\chi).$$

The centre of $A_F(\chi)$ is $F(\chi) = F(\chi(g) : g \in G)$, the *field of character values* of χ over F .

The map $\chi \mapsto A_F(\chi)$ defines a surjective map from the set of irreducible characters of G (in an algebraic closure of F) onto the set of Wedderburn components of FG .

Equivalently, the map $\chi \mapsto e_F(\chi)$ defines a surjective map from the set of irreducible characters of G (in an algebraic closure of F) onto the set of primitive central idempotents of FG .

If the irreducible character χ of G takes values in F then

$$e_F(\chi) = e(\chi) = \frac{\chi(1)}{|G|} \sum_{g \in G} \chi(g^{-1})g.$$

In general one has

$$e_F(\chi) = \sum_{\sigma \in \text{Gal}(F(\chi)/F)} e(\sigma \circ \chi).$$

9.5 Central simple algebras and Brauer equivalence

Let K be a field. A *central simple K -algebra* is a finite dimensional K -algebra with center K which has no non-trivial proper ideals. Every central simple K -algebra is isomorphic to a matrix algebra $M_n(D)$ where D is a division algebra (which is finite-dimensional over K and has centre K). The division algebra D is unique up to K -isomorphisms.

Two central simple K -algebras A and B are said to be *Brauer equivalent*, or simply *equivalent*, if there is a division algebra D and two positive integers m and n such that A is isomorphic to $M_m(D)$ and B is isomorphic to $M_n(D)$.

9.6 Crossed Products

Let R be a ring and G a group.

INTRINSIC DEFINITION. A *crossed product* [Pas89] of G over R (or with coefficients in R) is a ring $R * G$ with a decomposition into a direct sum of additive subgroups

$$R * G = \bigoplus_{g \in G} A_g$$

such that for each g, h in G one has:

- * $A_1 = R$ (here 1 denotes the identity of G),
- * $A_g A_h = A_{gh}$ and
- * A_g has a unit of $R * G$.

EXTRINSIC DEFINITION. Let $\text{Aut}(R)$ denote the group of automorphisms of R and let R^* denote the group of units of R .

Let $a : G \rightarrow \text{Aut}(R)$ and $t : G \times G \rightarrow R^*$ be mappings satisfying the following conditions for every g, h and k in G :

(1) $a(gh)^{-1}a(g)a(h)$ is the inner automorphism of R induced by $t(g, h)$ (i.e. the automorphism $x \mapsto t(g, h)^{-1}xt(g, h)$) and

(2) $t(gh, k)t(g, h)^k = t(g, hk)t(h, k)$, where for $g \in G$ and $x \in R$ we denote $a(g)(x)$ by x^g .

The *crossed product* [Pas89] of G over R (or with coefficients in R), action a and twisting t is the ring

$$R *_a^t G = \bigoplus_{g \in G} u_g R$$

where $\{u_g : g \in G\}$ is a set of symbols in one-to-one correspondence with G , with addition and multiplication defined by

$$(u_g r) + (u_g s) = u_g(r + s), \quad (u_g r)(u_h s) = u_{gh} t(g, h) r^h s$$

for $g, h \in G$ and $r, s \in R$, and extended to $R *_a^t G$ by linearity.

The associativity of the product defined is a consequence of conditions (1) and (2) [Pas89].

EQUIVALENCE OF THE TWO DEFINITIONS. Obviously the crossed product of G over R defined using the extrinsic definition is a crossed product of G over $u_1 R$ in the sense of the first definition. Moreover, there is r_0 in R^* such that $u_1 r_0$ is the identity of $R *_a^t G$ and the map $r \mapsto u_1 r_0 r$ is a ring isomorphism $R \rightarrow u_1 R$.

Conversely, let $R * G = \bigoplus_{g \in G} A_g$ be an (intrinsic) crossed product and select for each $g \in G$ a unit $u_g \in A_g$ of $R * G$. This is called a *basis of units for the crossed product* $R * G$. Then the maps $a : G \rightarrow \text{Aut}(R)$ and $t : G \times G \rightarrow R^*$ given by

$$r^g = u_g^{-1} r u_g, \quad t(g, h) = u_{gh}^{-1} u_g u_h \quad (g, h \in G, r \in R)$$

satisfy conditions (1) and (2) and $R * G = R *_a^t G$.

The choice of a basis of units $u_g \in A_g$ determines the action a and twisting t . If $\{u_g \in A_g : g \in G\}$ and $\{v_g \in A_g : g \in G\}$ are two sets of units of $R * G$ then $v_g = u_g r_g$ for some units r_g of R . Changing the basis of units results in a change of the action and the twisting and so changes the extrinsic definition of the crossed product but it does not change the intrinsic crossed product.

It is customary to select $u_1 = 1$. In that case $a(1)$ is the identity map of R and $t(1, g) = t(g, 1) = 1$ for each g in G .

9.7 Cyclic Crossed Products

Let $R * G = \bigoplus_{g \in G} A_g$ be a *crossed product* (9.6) and assume that $G = \langle g \rangle$ is cyclic. Then the crossed product can be given using a particularly nice description.

Select a unit u in A_g , and let a be the automorphism of R given by $r^a = u^{-1} r u$.

If G is infinite then set $u_{g^k} = u^k$ for every integer k . Then

$$R * G = R[u | ru = ur^a],$$

a skew polynomial ring. Therefore in this case $R * G$ is determined by

$$[R, a].$$

If G is finite of order d then set $u_{g^k} = u^k$ for $0 \leq k < d$. Then $b = u^d \in R$ and

$$R * G = R[u | ru = ur^a, u^d = b]$$

Therefore, $R * G$ is completely determined by the following data:

$$[R, [d, a, b]]$$

9.8 Abelian Crossed Products

Let $R * G = \bigoplus_{g \in G} A_g$ be a *crossed product* (9.6) and assume that G is abelian. Then the crossed product can be given using a simple description.

Express G as a direct sum of cyclic groups:

$$G = \langle g_1 \rangle \times \cdots \times \langle g_n \rangle$$

and for each $i = 1, \dots, n$ select a unit u_i in A_{g_i} .

Each element g of G has a unique expression

$$g = g_1^{k_1} \cdots g_n^{k_n},$$

where k_i is an arbitrary integer, if g_i has infinite order, and $0 \leq k_i < d_i$, if g_i has finite order d_i . Then one selects a basis for the crossed product by taking

$$u_g = u_{g_1^{k_1} \cdots g_n^{k_n}} = u_1^{k_1} \cdots u_n^{k_n}.$$

* For each $i = 1, \dots, n$, let a_i be the automorphism of R given by $r^{a_i} = u_i^{-1} r u_i$.

* For each $1 \leq i < j \leq n$, let $t_{i,j} = u_j^{-1} u_i^{-1} u_j u_i \in R$.

* If g_i has finite order d_i , let $b_i = u_i^{d_i} \in R$.

Then

$$R * G = R[u_1, \dots, u_n | ru_i = u_i r^{a_i}, u_j u_i = t_{i,j} u_i u_j, u_i^{d_i} = b_i (1 \leq i < j \leq n)],$$

where the last relation vanishes if g_i has infinite order.

Therefore $R * G$ is completely determined by the following data:

$$[R, [d_i, a_i, b_i]_{i=1}^n, [t_{i,j}]_{1 \leq i < j \leq n}].$$

9.9 Classical crossed products

A *classical crossed product* is a crossed product $L *_a^t G$, where L/K is a finite Galois extension, $G = \text{Gal}(L/K)$ is the Galois group of L/K and a is the natural action of G on L . Then t is a 2-cocycle and the *crossed product* (9.6) $L *_a^t G$ is denoted by $(L/K, t)$. The crossed product $(L/K, t)$ is known to be a central simple K -algebra [Rei03].

9.10 Cyclic Algebras

A *cyclic algebra* is a *classical crossed product* (9.9) $(L/K, t)$ where L/K is a finite cyclic field extension. The cyclic algebras have a very simple form.

Assume that $\text{Gal}(L/K)$ is generated by g and has order d . Let $u = u_g$ be the basis unit (9.6) of the crossed product corresponding to g and take the remaining basis units for the crossed product by setting $u_{g^i} = u^i$, ($i = 0, 1, \dots, d-1$). Then $a = u^n \in K$. The cyclic algebra is usually denoted by $(L/K, a)$ and one has the following description of $(L/K, t)$

$$(L/K, t) = (L/K, a) = L[u | ru = ur^g, u^d = a].$$

9.11 Cyclotomic algebras

A *cyclotomic algebra* over F is a *classical crossed product* (9.9) $(F(\xi)/F, t)$, where F is a field, ξ is a root of unity in an extension of F and $t(g, h)$ is a root of unity for every g and h in $\text{Gal}(F(\xi)/F)$.

The *Brauer-Witt Theorem* [Yam74] asserts that every *Wedderburn component* (9.3) of a group algebra is *Brauer equivalent* (9.5) (over its centre) to a cyclotomic algebra.

9.12 Numerical description of cyclotomic algebras

Let $A = (F(\xi)/F, t)$ be a *cyclotomic algebra* (9.11), where $\xi = \xi_k$ is a k -th root of unity. Then the Galois group $G = \text{Gal}(F(\xi)/F)$ is abelian and therefore one can obtain a simplified form for the description of cyclotomic algebras as for any *abelian crossed product* (9.8).

Then the $n \times n$ matrix algebra $M_n(A)$ can be described numerically in one of the following forms:

* If $F(\xi) = F$, (i.e. $G = 1$) then $A = M_n(F)$ and thus the only data needed to describe A are the matrix size n and the field F :

$$[n, F]$$

* If G is cyclic (but not trivial) of order d then A is a cyclic cyclotomic algebra

$$A = F(\xi)[u | \xi u = u \xi^\alpha, u^d = \xi^\beta]$$

and so $M_n(A)$ can be described with the following data

$$[n, F, k, [d, \alpha, \beta]],$$

where the integers k, d, α and β satisfy the following conditions:

$$\alpha^d \equiv 1 \pmod{k}, \quad \beta(\alpha - 1) \equiv 0 \pmod{k}.$$

* If G is abelian but not cyclic then $M_n(A)$ can be described with the following data (see 9.8):

$$[n, F, k, [d_i, \alpha_i, \beta_i]_{i=1}^m, [\gamma_{i,j}]_{1 \leq i < j \leq m}]$$

representing the $n \times n$ matrix ring over the following algebra:

$$A = F(\xi)[u_1, \dots, u_m | \xi u_i = u_i \xi^{\alpha_i}, \quad u_i^{d_i} = \xi^{\beta_i}, \quad u_s u_r = \xi^{\gamma_{rs}} u_r u_s, \quad i = 1, \dots, m, \quad 0 \leq r < s \leq m]$$

where

- * $\{g_1, \dots, g_m\}$ is an independent set of generators of G ,
- * d_i is the order of g_i ,
- * α_i, β_i and γ_{rs} are integers, and

$$\xi^{g_i} = \xi^{\alpha_i}.$$

9.13 Idempotents given by subgroups

Let G be a finite group and F a field whose characteristic does not divide the order of G . If H is a subgroup of G then set

$$\hat{H} = |H|^{-1} \sum_{x \in H} x.$$

The element $\widehat{H}$ is an idempotent of FG which is central in FG if and only if H is normal in G .

If H is a proper normal subgroup of a subgroup K of G then set

$$\varepsilon(K, H) = \prod_L (\widehat{N} - \widehat{L})$$

where L runs on the normal subgroups of K which are minimal among the normal subgroups of K containing N properly. By convention, $\varepsilon(K, K) = \widehat{K}$. The element $\varepsilon(K, H)$ is an idempotent of FG .

If H and K are subgroups of G such that H is normal in K then $e(G, K, H)$ denotes the sum of all different G -conjugates of $\varepsilon(K, H)$. The element $e(G, K, H)$ is central in FG . In general it is not an idempotent but if the different conjugates of $\varepsilon(K, H)$ are orthogonal then $e(G, K, H)$ is a central idempotent of FG .

If (K, H) is a Shoda Pair (9.14) of G then there is a non-zero rational number a such that $ae(G, K, H)$ is a *primitive central idempotent* (9.4) of the rational group algebra $\mathbb{Q}G$. If (K, H) is a strong Shoda pair (9.15) of G then $e(G, K, H)$ is a primitive central idempotent of $\mathbb{Q}G$.

Assume now that F is a finite field of order q , (K, H) is a strong Shoda pair of G and C is a cyclotomic class of K/H containing a generator of K/H . Then $e_C(G, K, H)$ is a primitive central idempotent of FG (see 9.19).

9.14 Shoda pairs of a group

Let G be a finite group. A *Shoda pair* of G is a pair (K, H) of subgroups of G for which there is a linear character χ of K with kernel H such that the induced character χ^G in G is irreducible. By [Sho33] or [OdS04], (K, H) is a Shoda pair if and only if the following conditions hold:

- * H is normal in K ,
- * K/H is cyclic and
- * if $K^g \cap K \subseteq H$ for some $g \in G$ then $g \in K$.

If (K, H) is a Shoda pair and χ is a linear character of $K \leq G$ with kernel H then the *primitive central idempotent* (9.4) of $\mathbb{Q}G$ associated to the irreducible character χ^G is of the form $e = e_{\mathbb{Q}}(\chi^G) = ae(G, K, H)$ for some $a \in \mathbb{Q}$ [OdS04] (see 9.13 for the definition of $e(G, K, H)$). In that case we say that e is the *primitive central idempotent realized by the Shoda pair* (K, H) of G .

A group G is monomial, that is every irreducible character of G is monomial, if and only if every primitive central idempotent of $\mathbb{Q}G$ is realizable by a Shoda pair of G .

9.15 Strong Shoda pairs of a group

A *strong Shoda pair* of G is a pair (K, H) of subgroups of G satisfying the following conditions:

- * H is normal in K and K is normal in the normalizer N of H in G ,
- * K/H is cyclic and a maximal abelian subgroup of N/H and
- * for every $g \in G \setminus N$, $\varepsilon(K, H)\varepsilon(K, H)^g = 0$. (See 9.13 for the definition of $\varepsilon(K, H)$).

Let (K, H) be a strong Shoda pair of G . Then (K, H) is a Shoda pair (9.14) of G . Thus there is a linear character θ of K with kernel H such that the induced character $\chi = \chi(G, K, H) = \theta^G$ is irreducible. Moreover the *primitive central idempotent* (9.4) $e_{\mathbb{Q}}(\chi)$ of $\mathbb{Q}G$ realized by (K, H) is $e(G, K, H)$, see [OdS04].

Two *strong Shoda pairs* (9.15) (K_1, H_1) and (K_2, H_2) of G are said to be *equivalent* if the characters $\chi(G, K_1, H_1)$ and $\chi(G, K_2, H_2)$ are Galois conjugate, or equivalently if $e(G, K_1, H_1) = e(G, K_2, H_2)$. A

set of representatives of strong Shoda pairs of G is termed as a complete irredundant set of strong Shoda pairs of G .

The advantage of strong Shoda pairs over Shoda pairs is that one can describe the simple algebra $FGe_F(\chi)$ as a matrix algebra of a *cyclotomic algebra* (9.11, see [OdS04] for $F = \mathbb{Q}$ and [Olt07] for the general case).

More precisely, $\mathbb{Q}Ge(G, K, H)$ is isomorphic to $M_n(\mathbb{Q}(\xi) *_a^t N/K)$, where ξ is a $[K : H]$ -th root of unity, N is the normalizer of H in G , $n = [G : N]$ and $\mathbb{Q}(\xi) *_a^t N/K$ is a *crossed product* (see 9.6) with action a and twisting t given as follows:

Let x be a fixed generator of K/H and $\varphi : N/K \rightarrow N/H$ a fixed left inverse of the canonical projection $N/H \rightarrow N/K$. Then

$$\xi^{a(r)} = \xi^i, \text{ if } x^{\varphi(r)} = x^i$$

and

$$t(r, s) = \xi^j, \text{ if } \varphi(rs)^{-1} \varphi(r) \varphi(s) = x^j,$$

for $r, s \in N/K$ and integers i and j , see [OdS04]. Notice that the cocycle is the one given by the natural extension

$$1 \rightarrow K/H \rightarrow N/H \rightarrow N/K \rightarrow 1$$

where K/H is identified with the multiplicative group generated by ξ . Furthermore the centre of the algebra is $\mathbb{Q}(\chi)$, the field of character values over $\mathbb{Q}$, and N/K is isomorphic to $Gal(\mathbb{Q}(\xi)/\mathbb{Q}(\chi))$.

If the rational field is changed to an arbitrary ring F of characteristic 0 then the Wedderburn component $A_F(\chi)$, where $\chi = \chi(G, K, H)$ is isomorphic to $F(\chi) \otimes_{\mathbb{Q}(\chi)} A_{\mathbb{Q}}(\chi)$. Using the description given above of $A_{\mathbb{Q}}(\chi) = \mathbb{Q}Ge(G, K, H)$ one can easily describe $A_F(\chi)$ as $M_{nd}(F(\xi)/F(\chi), t')$, where $d = [\mathbb{Q}(\xi) : \mathbb{Q}(\chi)]/[F(\xi) : F(\chi)]$ and t' is the restriction to $Gal(F(\xi)/F(\chi))$ of t (a cocycle of $N/K = Gal(\mathbb{Q}(\xi)/\mathbb{Q}(\chi))$).

9.16 Extremely strong Shoda pairs of a group

An *extremely strong Shoda pair* of G is a pair (K, H) of subgroups of G satisfying the following conditions:

- * K is normal in G ,
- * K/H is cyclic and a maximal abelian subgroup of N/H , where N is the normalizer of H in G .

Let (K, H) be an extremely strong Shoda pair of G . Then (K, H) is a strong Shoda pair (9.15) of G , with K normal in G [BM14], so that there is a linear character θ of K with kernel H such that the induced character $\chi = \chi(G, K, H) = \theta^G$ is irreducible. Moreover, the *primitive central idempotent* $e_{\mathbb{Q}}(\chi)$ of $\mathbb{Q}G$ realized by (K, H) is $e(G, K, H)$ (9.4) and one can describe the associated simple algebra (9.15). Two *extremely strong Shoda pairs* of G are said to be *equivalent* if they are equivalent as strong Shoda pairs (9.15). A set of representatives of extremely strong Shoda pairs of G is called a *complete irredundant set* of extremely strong Shoda pairs of G [BM14].

If G is a normally monomial group (9.18), then the set of primitive central idempotents of the rational group algebra realized by strong Shoda pairs of G is same as the one realized by extremely strong Shoda pairs of G [BM14]. The algorithm to compute a complete irredundant set of extremely strong Shoda pairs of G has been explained in [BM16].

9.17 Strongly monomial characters and strongly monomial groups

Let G be a finite group and χ an irreducible character of G .

One says that χ is *strongly monomial* if there is a *strong Shoda pair* (9.15) (K, H) of G and a linear character θ of K of G with kernel H such that $\chi = \theta^G$.

The group G is *strongly monomial* if every irreducible character of G is strongly monomial.

Strong Shoda pairs were firstly introduced by Olivieri, del Río and Simón who proved that every abelian-by-supersolvable group is strongly monomial [OdS04]. The algorithm to compute the Wedderburn decomposition of rational group algebras for strongly monomial groups was explained in [Od03]. This method was extended for semisimple finite group algebras by Broche Cristo and del Río in [Bd07] (see Section 9.19). Finally, Olteanu [Olt07] shows how to compute the *Wedderburn decomposition* (9.3) of an arbitrary semisimple group ring by making use of not only the strong Shoda pairs of G but also the strong Shoda pairs of the subgroups of G .

9.18 Normally monomial characters and normally monomial groups

Let G be a finite group and χ be an irreducible character of G .

One says that χ is *normally monomial* if there is a normal subgroup K of G such that χ is induced from a linear character of K .

The group G is *normally monomial* if every irreducible character of G is normally monomial. Bakshi and Maheshwary proved that if G is a normally monomial group, then for every irreducible character χ of G , there exists an extremely strong Shoda pair (K, H) of G (9.16) such that $\chi = \theta^G$, where θ is a linear character of K with kernel H [BM14].

9.19 Cyclotomic Classes and Strong Shoda Pairs

Let G be a finite group and F a finite field of order q , coprime to the order of G .

Given a positive integer n , coprime to q , the q -cyclotomic classes modulo n are the set of residue classes modulo n of the form

$$\{i, iq, iq^2, iq^3, \dots\}$$

The q -cyclotomic classes modulo n form a partition of the set of residue classes modulo n .

A *generating cyclotomic class* modulo n is a cyclotomic class containing a generator of the additive group of residue classes modulo n , or equivalently formed by integers coprime to n .

Let (K, H) be a strong Shoda pair (9.15) of G and set $n = [K : H]$. Fix a primitive n -th root of unity ξ in some extension of F and an element g of K such that gH is a generator of K/H . Let C be a generating q -cyclotomic class modulo n . Then set

$$\varepsilon_C(K, H) = [K : H]^{-1} \hat{H} \sum_{i=0}^{n-1} \text{tr}(\xi^{-ci}) g^i,$$

where c is an arbitrary element of C and tr is the trace map of the field extension $F(\xi)/F$. Then $\varepsilon_C(K, H)$ does not depend on the choice of $c \in C$ and is a *primitive central idempotent* (9.4) of FK .

Finally, let $e_C(G, K, H)$ denote the sum of the different G -conjugates of $\varepsilon_C(K, H)$. Then $e_C(G, K, H)$ is a *primitive central idempotent* (9.4) of FG [Bd07]. We say that $e_C(G, K, H)$ is the primitive central idempotent realized by the strong Shoda pair (K, H) of the group G and the cyclotomic class C .

If G is *strongly monomial* (9.17) then every primitive central idempotent of FG is realizable by some *strong Shoda pair* (9.15) of G and some cyclotomic class C [Bd07]. As in the zero characteristic case, this explain how to compute the *Wedderburn decomposition* (9.3) of FG for a finite semisimple algebra of a strongly monomial group (see [Bd07] for details). For non strongly monomial groups the algorithm to compute the Wedderburn decomposition just uses the Brauer characters.

9.20 Theory for Local Schur Index and Division Algebra Part Calculations

(By Allen Herman, May 2013. Updated October 2014.)

The division algebra parts of simple algebras in the Wedderburn Decomposition of the group algebra of a finite group over an abelian number field F correspond to elements of the Schur Subgroup $S(F)$ of the Brauer group of F . Like all classes in the Brauer group of an algebraic number field F , the division algebra part of a representative of a given Brauer class is determined up to F -algebra isomorphism by its list of local Hasse invariants at all primes (i.e. places) of F . The local invariant at a prime P of F is a lowest terms fraction r/m_P whose denominator is the local Schur index m_P of the simple algebra at the prime q (see [Rei03]). For division algebras whose Brauer class lies in the Schur Subgroup of an abelian number field F , the local indices at any of the primes P lying over the same rational prime p are equal to the same positive integer m_p , and the numerator of the local invariants among these primes are uniformly distributed among the integers r coprime to m_p [BS72].

The local Schur index functions in `wedderga` produce a list of the nontrivial local indices of the division algebra part of the simple algebra at all rational primes. The Schur index of the simple algebra over F is the least common multiple m of these local indices, and the dimension of the division algebra part of the simple algebra over F is m^2 . While not sufficient to identify these division algebras up to ring isomorphism in general, this list of local indices does identify the division algebra up to ring isomorphism whenever there is no pair of local indices at odd primes that are greater than 2. (This is at least the case for groups of order less than $3^2 \cdot 7 \cdot 13$.) So it gives the information desired in most basic situations, and allows one to distinguish almost all pairs of simple components of group algebras.

`Wedderga`'s functions compute local indices for generalized quaternion algebras defined over the rationals and cyclotomic algebras defined over any abelian number field. Special shortcut functions are available for cyclic cyclotomic algebras. There are also versions of the functions that compute the local and global Schur index of a character of a finite group over a given abelian number field. The steps in the general character-theoretic method involve 1) a Brauer-Witt reduction to a cyclic-by-abelian group, 2) use of the Frobenius-Schur indicator to compute the local index at infinity, 3) computing the p -local index for an ordinary irreducible character χ of a p -solvable group using the values of an irreducible Brauer character in the same p -block in cases where the p -defect group of χ is cyclic, and 4) use of Riese and Schmid's characterization of dyadic Schur groups ([Sch94] and [RS96]) to handle the exceptional cases where step 3) is not available. Our approach to rational quaternion algebras is the standard one given, for example, in [Pie82]. The Legendre symbol operation in GAP is used to determine the local index at odd primes. The local index of the generalized quaternion algebra (a, b) over $\mathbb{Q}$ at the infinite prime will be 2 if both a and b are negative, and otherwise 1. We avoid the complicated case of quadratic reciprocity when working over $\mathbb{Q}$ by using the Hasse-Brauer-Albert-Noether Theorem ([Rei03], pg. 276): since we know the other primes of $\mathbb{Q}$ where the local index is 2, it determines the local index at the prime 2. For generalized quaternion algebras over number fields F other than $\mathbb{Q}$, we have to convert to cyclic or cyclic cyclotomic algebras and use the other local index functions, or appeal to a number theory system outside of GAP that can solve norm equations.

There are three shortcut functions used to compute local indices of cyclic cyclotomic algebras, which `wedderga`'s `-Info` functions produce in the form $[r, F, n, [a, b, c]]$. The local index at infinity is calculated by determining if the real completion of the corresponding algebra will produce a real

quaternion algebra. In order to do this, F must be a real subfield, n must be strictly greater than 2, and $E(n)^c$ (which has to be a root of unity in F) must be -1 . These facts can be checked directly, so this is faster than calculating the character table of the group and checking the value of a Frobenius–Schur indicator. The shortcut to calculate the local index of a cyclic cyclotomic algebra at an odd prime makes direct use of the following lemma of Janusz: If E_p/F_p is a Galois extension of p -local fields with ramification index e , and z is a root of unity with order prime to p , then z is a norm in E_p/F_p if and only if it is the e -th power of a root of unity in F . ([Jan75], pg. 535). It follows that in order to calculate the local index at p of a cyclic cyclotomic algebra $[r, F, n, [a, b, c]]$, we first determine the splitting degree, residue degree, and ramification index e of the extension $F(\zeta_n)/F$ at p . Comparing the behaviour of the Galois automorphism σ_b to the behaviour of the Frobenius automorphism at p allows us to determine the order of the largest root of unity z with order coprime to p in the p -completion F_p . The local index m_p is then the least power of $E(n)^c$ that lies in the group generated by z^e .

Calculation of the local index at the prime 2 makes use of the following consequence of ([Jan75], Theorem 5): A cyclic cyclotomic algebra $[r, F_2, n, [a, b, c]]$ over a 2-local field F_2 that is a subfield of a cyclotomic extension of the rational 2-local field $\mathbb{Q}_2$ has Schur index at most 2. It has Schur index 2 if and only if 4 divides n , $F_2(\zeta_4)$ is totally ramified of degree 2, the Galois automorphism σ_b of $F_2(\zeta_n)/F_2$ inverts all 2-power roots of unity in $F_2(\zeta_n)$, the order of $E(n)^c$ is 2 times an odd number, and $(F_2 : \mathbb{Q}_2)$ is odd. The same approach to cyclotomic reciprocity makes it possible to check all of these conditions in the 2-local situation.

The wedderga function that computes the p -local index of an ordinary irreducible character χ of a finite non-nilpotent cyclic-by-abelian group G is based directly on a theorem of Benard [Ben76] that applies whenever the p -defect group of χ is cyclic. We have to restrict our application of it to groups whose orders are small because the GAP records for irreducible Brauer characters are only available in these cases. In order to use this approach effectively, we developed a function that computes the defect group of the block containing a given ordinary irreducible character χ . This function makes use of the Min half of Brauer’s Min-Max theorem (see Theorem 4.4 of [Nav98]), and thus is able to find the defect group directly from the ordinary character table. It is thus available for nonsolvable groups, even in cases where GAP’s Brauer character records are not available. We are indebted to Michael Geline and Friederich Ladisch for discussions concerning the calculation of defect groups in GAP. The current algorithm we use is based on an approach suggested by Ladisch.

9.21 Obtaining Algebras with structure constants as terms of the Wedderburn decomposition

Some users may find it desirable to have an alternative description for the components of the Wedderburn decomposition of a group ring as algebras with structure constants, because the operations for algebras in GAP are designed for algebras with structure constants. We have provided such an algorithm that converts the output of WedderburnDecompositionInfo (2.1.2) into algebras with structure constants. Matrix rings over fields are converted directly. For components that are cyclotomic algebras, it calculates their defining group and defining character using those Wedderga operations, then uses IrreducibleRepresentationsDixon (Reference: IrreducibleRepresentationsDixon) to obtain matrix generators of an algebra isomorphic to the simple component corresponding to the character over a suitable field. An algebra with structure constants version of this is finally obtained by applying IsomorphismSCAlgebra (Reference: IsomorphismSCAlgebra w.r.t. a given basis) to this algebra.

9.22 A complete set of orthogonal primitive idempotents

When R is a semisimple ring, then every left ideal L of R is of the form $L = Re$, where e is an idempotent of R . Therefore, we can use the idempotents to characterize the decompositions of semisimple rings as a direct sum of minimal left ideals. In particular, let $R = \bigoplus_{i=1}^t L_i$ be a decomposition of a semisimple ring as a direct sum of minimal left ideals. Then, there exists a family $\{e_1, \dots, e_t\}$ of elements of R such that: each $e_i \neq 0$ is an idempotent element, if $i \neq j$, then $e_i e_j = 0$, $1 = e_1 + \dots + e_t$ and each e_i cannot be written as $e_i = e'_i + e''_i$, where e'_i, e''_i are idempotents such that $e'_i, e''_i \neq 0$ and $e'_i e''_i = 0$, $1 \leq i \leq t$. Conversely, if there exists a family of idempotents $\{e_1, \dots, e_t\}$ satisfying the previous conditions, then the left ideals $L_i = Re_i$ are minimal and $R = \bigoplus_{i=1}^t L_i$. Such a set of idempotents is called a *complete set of orthogonal primitive idempotents* of the ring R . Such a set is not uniquely determined.

Let $\mathbb{F}$ be a finite field and G a finite nilpotent group such that $\mathbb{F}G$ is semisimple. Let (H, K) be a strong Shoda pair of G , $C \in \mathcal{C}(H/K)$ and set $e_C = e_C(G, H, K)$, $\varepsilon_C = \varepsilon_C(H, K)$, $H/K = \langle \bar{a} \rangle$, $E = E_G(H/K)$. Let E_2/K and $H_2/K = \langle \bar{a}_2 \rangle$ (respectively $E_{2'}/K$ and $H_{2'}/K = \langle \bar{a}_{2'} \rangle$) denote the 2-parts (respectively 2'-parts) of E/K and H/K respectively. Then $\langle \bar{a}_{2'} \rangle$ has a cyclic complement $\langle \bar{b}_{2'} \rangle$ in $E_{2'}/K$. Using the description of the primitive central idempotents and the Wedderburn components of a semisimple finite group algebra $\mathbb{F}G$ (9.19), a complete set of orthogonal primitive idempotents of $\mathbb{F}Ge_C$ is described (see [OV11]) as the set of conjugates of $\beta_{e_C} = \widetilde{b_{2'}} \beta_2 \varepsilon_C$ by the elements of $T_{e_C} = T_{2'} T_2 T_E$, where $T_{2'} = \{1, a_{2'}, a_{2'}^2, \dots, a_{2'}^{[E_{2'}:H_{2'}]-1}\}$, T_E denotes a right transversal of E in G and β_2 and T_2 are given according to the cases below.

1. If H_2/K has a complement M_2/K in E_2/K then $\beta_2 = \widetilde{M_2}$. Moreover, if M_2/K is cyclic, then there exists $b_2 \in E_2$ such that E_2/K is given by the following presentation

$$\langle \bar{a}_2, \bar{b}_2 \mid \bar{a}_2^{2^n} = \bar{b}_2^{2^k} = 1, \bar{a}_2 \bar{b}_2 = \bar{a}_2^r \rangle,$$

and if M_2/K is not cyclic, then there exist $b_2, c_2 \in E_2$ such that E_2/K is given by the following presentation

$$\langle \bar{a}_2, \bar{b}_2, \bar{c}_2 \mid \bar{a}_2^{2^n} = \bar{b}_2^{2^k} = \bar{c}_2^{2^l} = 1, \bar{a}_2 \bar{b}_2 = \bar{a}_2^r, \bar{a}_2 \bar{c}_2 = \bar{a}_2^{-1}, [\bar{b}_2, \bar{c}_2] = 1 \rangle,$$

with $r \equiv 1 \pmod{4}$ (or equivalently $\bar{a}_2^{2^{n-2}}$ is central in E_2/K). Then

- (a) $T_2 = \{1, a_2, a_2^2, \dots, a_2^{2^k-1}\}$, if $\bar{a}_2^{2^{n-2}}$ is central in E_2/K (unless $n \leq 1$) and M_2/K is cyclic; and
 - (b) $T_2 = \{1, a_2, a_2^2, \dots, a_2^{d/2-1}, a_2^{2^{n-2}}, a_2^{2^{n-2}+1}, \dots, a_2^{2^{n-2}+d/2-1}\}$, where $d = [E_2 : H_2]$, otherwise.
2. If H_2/K has no complement in E_2/K , then there exist $b_2, c_2 \in E_2$ such that E_2/K is given by the following presentation

$$\langle \bar{a}_2, \bar{b}_2, \bar{c}_2 \mid \bar{a}_2^{2^n} = \bar{b}_2^{2^k} = 1, \bar{c}_2^{2^l} = \bar{a}_2^{2^{n-1}}, \bar{a}_2 \bar{b}_2 = \bar{a}_2^r, \bar{a}_2 \bar{c}_2 = \bar{a}_2^{-1}, [\bar{b}_2, \bar{c}_2] = 1 \rangle,$$

with $r \equiv 1 \pmod{4}$. In this case, $\beta_2 = \widetilde{b_2^{1+xa_2^{2^{n-2}}+ya_2^{2^{n-2}}c_2}}$ and

$$T_2 = \{1, a_2, a_2^2, \dots, a_2^{2^k-1}, c_2, c_2 a_2, c_2 a_2^2, \dots, c_2 a_2^{2^k-1}\},$$

with $x, y \in \mathbb{F}$, satisfying $x^2 + y^2 = -1$ and $y \neq 0$.

When G is not nilpotent, we can still use the following description in some specific cases. Let G be a finite group and $\mathbb{F}$ a finite field of order s such that s is coprime to the order of G . Let (H, K) be a strong Shoda pair of G such that $\tau(gH, g'H) = 1$ for all $g, g' \in E = E_G(H/K)$, and let $C \in \mathcal{C}(H/K)$. Let $\varepsilon = \varepsilon_C(H, K)$ and $e = e_C(G, H, K)$ (9.19). Let w be a normal element of $\mathbb{F}_{s^o}/\mathbb{F}_{s^o/[E:H]}$ (with o the multiplicative order of s modulo $[H : K]$) and B the normal basis determined by w . Let ψ be the isomorphism between $\mathbb{F}E\varepsilon$ and the matrix algebra $M_{[E:H]}(\mathbb{F}_{s^o/[E:H]})$ with respect to the basis B as stated in Corollary 29.8 in [Rei03]. Let $P, A \in M_{[E:H]}(\mathbb{F}_{s^o/[E:H]})$ be the matrices

$$P = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & -1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & -1 & 0 \\ 1 & 0 & 0 & \cdots & 0 & -1 \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Then

$$\{x\widehat{T}_1 \varepsilon x^{-1} \mid x \in T_2 \langle x_e \rangle\}$$

is a complete set of orthogonal primitive idempotents of $\mathbb{F}Ge$ where $x_e = \psi^{-1}(PAP^{-1})$, T_1 is a transversal of H in E and T_2 is a right transversal of E in G ([OV15]). By $\widehat{T}_1$ we denote the element $\frac{1}{|T_1|} \sum_{t \in T_1} t$ in $\mathbb{F}G$.

9.23 Applications to coding theory

A *linear code* of length n and rank k is a linear subspace C with dimension k of the vector space $\mathbb{F}_q^n$. The standard basis of $\mathbb{F}_q^n$ is denoted by $E = \{e_1, \dots, e_n\}$. The vectors in C are called codewords, the size of a code is the number of codewords and equals q^k . The distance of a code is the minimum distance between distinct codewords, i.e. the number of elements in which they differ.

For any group G , we denote by $\mathbb{F}_q G$ the group algebra over G with coefficients in $\mathbb{F}_q$. If G is a group of order n and $C \subseteq \mathbb{F}_q^n$ is a linear code, then we say that C is a left G -code (respectively a G -code) if there is a bijection $\phi : E \rightarrow G$ such that the linear extension of ϕ to an isomorphism $\phi : \mathbb{F}_q^n \rightarrow \mathbb{F}_q G$ maps C to a left ideal (respectively a two-sided ideal) of $\mathbb{F}_q G$. A left *group code* (respectively a *group code*) is a linear code which is a left G -code (respectively a G -code) for some group G .

Since left ideals in $\mathbb{F}_q G$ are generated by idempotents, there is a one-one relation between (sums of) primitive idempotents of $\mathbb{F}_q G$ and left G -codes over $\mathbb{F}_q$.

Note that each element c in $\mathbb{F}_q G$ is of the form $c = \sum_{i=1}^n f_i g_i$, where we fix an ordering $\{g_1, g_2, \dots, g_n\}$ of the group elements of G and $f_i \in \mathbb{F}_q$. If one looks at c as a codeword, one writes $[f_1 f_2 \dots f_n]$.

References

- [Bd07] O. Broche and Á. del Río. Wedderburn decomposition of finite group algebras. *Finite Fields Appl.*, 13(1):71–79, 2007. [67](#), [68](#)
- [Ben76] M. Benard. Schur indices and cyclic defect groups. *Ann. of Math. (2)*, 103(2):283–304, 1976. [70](#)
- [BM14] G. K. Bakshi and S. Maheshwary. The rational group algebra of a normally monomial group. *J. Pure Appl. Algebra*, 218(9):1583–1593, 2014. [7](#), [60](#), [66](#), [67](#)
- [BM16] G. K. Bakshi and S. Maheshwary. Extremely strong Shoda pairs with GAP. *J. Symbolic Comput.*, 76(9):97–106, 2016. [7](#), [60](#), [66](#)
- [BS72] M. Benard and M. Schacher. The schur subgroup. ii. *J. Algebra*, 22(1):378–385, 1972. [69](#)
- [Jan75] G. Janusz. Generators for the schur group of local and global number fields. *Pacific J. Math.*, 56(2):525–546, 1975. [70](#)
- [Nav98] G. Navarro. *Characters and Blocks of Finite Groups*, volume 250 of *Lecture Note Series*. London Mathematical Society, Cambridge, UK, 1998. [70](#)
- [Od03] A. Olivieri and Á. del Río. An algorithm to compute the primitive central idempotents and the Wedderburn decomposition of a rational group algebra. *J. Symbolic Comput.*, 35(6):673–687, 2003. [7](#), [60](#), [67](#)
- [OdS04] A. Olivieri, Á. del Río, and J. J. Simón. On monomial characters and central idempotents of rational group algebras. *Comm. Algebra*, 32(4):1531–1550, 2004. [7](#), [60](#), [65](#), [66](#), [67](#)
- [Olt07] G. Olteanu. Computing the Wedderburn decomposition of group algebras by the Brauer-Witt theorem. *Math. Comp.*, 76(258):1073–1087 (electronic), 2007. [7](#), [60](#), [66](#), [67](#)
- [OV11] G. Olteanu and I. Van Gelder. Finite group algebras of nilpotent groups: A complete set of orthogonal primitive idempotents. *Finite Fields Appl.*, 17(2):157–165, 2011. [71](#)
- [OV15] G. Olteanu and I. Van Gelder. Construction of minimal non-abelian left group codes. *Des. Codes Cryptography*, 75(3):359–373, 2015. [72](#)
- [Pas89] D. S. Passman. *Infinite crossed products*, volume 135 of *Pure and Applied Mathematics*. Academic Press Inc., Boston, MA, 1989. [61](#), [62](#)
- [Pie82] R. S. Pierce. *Associative Algebras*, volume 88 of *Graduate Texts in Mathematics*. Springer Verlag, New York - Berlin, 1982. [69](#)

- [Rei03] I. Reiner. *Maximal orders*, volume 28 of *London Mathematical Society Monographs. New Series*. The Clarendon Press Oxford University Press, Oxford, 2003. Corrected reprint of the 1975 original, With a foreword by M. J. Taylor. [12](#), [63](#), [69](#), [72](#)
- [RS96] U. Riese and P. Schmid. Schur indices and schur groups, ii. *J. Algebra*, 182(1):183–200, 1996. [69](#)
- [Sch94] P. Schmid. Schur indices and schur groups. *J. Algebra*, 169(15):226–247, 1994. [69](#)
- [Sho33] K. Shoda. Über die monomialen Darstellungen einer endlichen Gruppe. *Proc. Phys.-Math. Soc. Japan*, III(15):249–257, 1933. [65](#)
- [Yam74] T. Yamada. *The Schur subgroup of the Brauer group*. Springer-Verlag, Berlin, 1974. Lecture Notes in Mathematics, Vol. 397. [7](#), [60](#), [64](#)

Index

- $\varepsilon(K, H)$, [64](#)
- $e(G, K, H)$, [64](#)
- $e_C(G, K, H)$, [64](#)
- $\wedge$, [37](#)
- Abelian Crossed Product, [63](#)
- ActionForCrossedProduct, [30](#)
- AntiSymMatUpMat, [44](#)
- AverageSum, [37](#)
- Basis of units (for crossed product), [62](#)
- (Brauer) equivalence, [61](#)
- central simple algebra, [61](#)
- Centralizer, [37](#)
- CharacterDescent, [45](#)
- Classical Crossed Product, [63](#)
- CodeByLeftIdeal, [57](#)
- CodeWordByGroupRingElement, [57](#)
- CoefficientsAndMagmaElements, [33](#)
- Complete set of orthogonal primitive idempotents, [71](#)
- ConvertCyclicAlgToCyclicCyclotomicAlg, [55](#)
- ConvertCyclicCyclotomicAlgToCyclicAlg, [55](#)
- ConvertQuadraticAlgToQuaternionAlg, [55](#)
- ConvertQuaternionAlgToQuadraticAlg, [55](#)
- Crossed Product, [61](#)
- CrossedProduct, [27](#)
- Cyclic Algebra, [63](#)
- Cyclic Crossed Product, [62](#)
- Cyclotomic algebra, [64](#)
- cyclotomic class, [68](#)
- CyclotomicAlgebraAsSCAlgebra, [42](#)
- CyclotomicAlgebraWithDivAlgPart, [41](#)
- CyclotomicClasses, [38](#)
- CyclotomicExtensionGenerator, [44](#)
- DecomposeCyclotomicAlgebra, [54](#)
- DefectGroupOfConjugacyClassAtP, [50](#)
- DefectGroupsOfPBlock, [50](#)
- DefectOfCharacterAtP, [50](#)
- DefiningCharacterOfCyclotomicAlgebra, [49](#)
- DefiningGroupAndCharacterOfCyclotAlg, [49](#)
- DefiningGroupOfCyclotomicAlgebra, [49](#)
- ElementOfCrossedProduct, [33](#)
- Embedding, [33](#)
- equivalence (Brauer), [61](#)
- equivalent extremely strong Shoda pairs, [66](#)
- equivalent strong Shoda pairs, [65](#)
- extremely strong Shoda pair, [66](#)
- ExtremelyStrongShodaPairs, [17](#)
- field of character values, [60](#)
- FinFieldExt, [51](#)
- GaloisRepsOfCharacters, [46](#)
- generating cyclotomic class, [68](#)
- GlobalCharacterDescent, [45](#)
- GlobalSchurIndexFromLocalIndices, [52](#)
- GlobalSplittingOfCyclotomicAlgebra, [44](#)
- group algebra, [59](#)
- group code, [72](#)
- group ring, [59](#)
- InfoWedderga, [39](#)
- IsCompleteSetOfOrthogonalIdempotents, [21](#)
- IsCrossedProduct, [27](#)
- IsCrossedProductObjDefaultRep, [33](#)
- IsCyclotomicClass, [38](#)
- IsDyadicSchurGroup, [52](#)
- IsElementOfCrossedProduct, [33](#)
- IsExtremelyStrongShodaPair, [18](#)
- IsNormallyMonomial, [20](#)

- IsRationalQuaternionAlgebraADivision-
Ring, [53](#)
- IsSemisimpleANFGroupAlgebra, [36](#)
- IsSemisimpleFiniteGroupAlgebra, [36](#)
- IsSemisimpleRationalGroupAlgebra, [35](#)
- IsSemisimpleZeroCharacteristicGroup-
Algebra, [35](#)
- IsShodaPair, [19](#)
- IsStronglyMonomial, [20](#)
- IsStrongShodaPair, [18](#)
- IsTwistingTrivial, [36](#)
- KillingCocycle, [44](#)
- LeftActingDomain, [30](#)
- linear code, [72](#)
- LocalIndexAtInfty, [47](#)
- LocalIndexAtInftyByCharacter, [50](#)
- LocalIndexAtOddP, [47](#)
- LocalIndexAtOddPByCharacter, [52](#)
- LocalIndexAtPByBrauerCharacter, [51](#)
- LocalIndexAtTwo, [47](#)
- LocalIndexAtTwoByCharacter, [52](#)
- LocalIndicesOfCyclicCyclotomicAlgebra,
[47](#)
- LocalIndicesOfCyclotomicAlgebra, [48](#)
- LocalIndicesOfRationalQuaternion-
Algebra, [52](#)
- LocalIndicesOfRationalSymbolAlgebra, [52](#)
- LocalIndicesOfTensorProductOf-
QuadraticAlgs, [52](#)
- normally monomial character, [67](#)
- normally monomial group, [67](#)
- OnPoints, [37](#)
- PDashPartOfN, [43](#)
- PPartOfN, [43](#)
- primitive central idempotent, [60](#)
- primitive central idempotent realized by a Shoda
pair, [65](#)
- primitive central idempotent realized by a strong
Shoda pair and a cyclotomic class, [68](#)
- PrimitiveCentralIdempotentsBy-
CharacterTable, [21](#)
- PrimitiveCentralIdempotentsByESSP, [22](#)
- PrimitiveCentralIdempotentsBySP, [24](#)
- PrimitiveCentralIdempotentsByStrongSP,
[23](#)
- PrimitiveIdempotentsNilpotent, [25](#)
- PrimitiveIdempotentsTrivialTwisting, [26](#)
- PSplitSubextension, [43](#)
- Quaternion algebra, [32](#)
- RamificationIndexAtP, [44](#)
- ReducingCyclotomicAlgebra, [44](#)
- ResidueDegreeAtP, [44](#)
- RootOfDimensionOfCyclotomicAlgebra, [49](#)
- SchurIndex, [41](#)
- SchurIndexByCharacter, [41](#)
- semisimple ring, [59](#)
- Shoda pair, [65](#)
- SimpleAlgebraByCharacter, [13](#)
- SimpleAlgebraByCharacterInfo, [14](#)
- SimpleAlgebraByStrongSP
for rational group algebra, [15](#)
for semisimple finite group algebra, [15](#)
- SimpleAlgebraByStrongSPInfo
for rational group algebra, [15](#)
for semisimple finite group algebra, [16](#)
- SimpleAlgebraByStrongSPInfoNC
for rational group algebra, [15](#)
for semisimple finite group algebra, [16](#)
- SimpleAlgebraByStrongSPNC
for rational group algebra, [15](#)
for semisimple finite group algebra, [15](#)
- SimpleComponentByCharacterAsSCAlgebra,
[42](#)
- SimpleComponentByCharacterDescent, [45](#)
- SimpleComponentOfGroupRingByCharacter,
[49](#)
- strongly monomial character, [66](#)
- strongly monomial group, [67](#)
- SplittingDegreeAtP, [44](#)
- strong Shoda pair, [65](#)
- StrongShodaPairs, [18](#)
- TwistingForCrossedProduct, [30](#)
- UnderlyingMagma, [30](#)
- Wedderburn components, [59](#)
- Wedderburn decomposition, [59](#)

WedderburnDecomposition, [9](#)
WedderburnDecompositionAsSCAlgebras, [42](#)
WedderburnDecompositionByCharacter-
Descent, [46](#)
WedderburnDecompositionInfo, [10](#)
WedderburnDecompositionWithDivAlg-
Parts, [40](#)
Wedderga package, [2](#)
ZeroCoefficient, [33](#)