The InduceReduce Package

Version 1.5

26 November 2026

Jonathan Gruber

Jonathan Gruber Email: jonathan.gruber@fau.de

Copyright

© 2018 by Jonathan Gruber

The InduceReduce package 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 3 of the License, or (at your option) any later version.

Acknowledgements

I would like to thank Gunter Malle both for teaching me the theory behind the algorithm which is implemented in the program and for suggesting to make it a GAP package.

Contents

1 The InduceReduce package **4**
1.1 Theory 4
1.2 Program 4
1.3 Options 5

2 Installing and Loading the InduceReduce Package **9**
2.1 Installing the package 9
2.2 Loading the package 9

References **10**

Index **11**

Chapter 1

The InduceReduce package

1.1 Theory

The InduceReduce provides a function for computing the table of ordinary irreducible characters of a finite group G using an algorithm based on Brauer's theorem on induced characters (see the Wikipedia article), which has been described by W. Unger in [Ung06]. By Brauer's theorem, the ring $\mathbb{Z}\mathrm{Irr}(G)$ of generalized characters of G is generated by the induced characters $\mathrm{Ind}_H^G(\lambda)$, where H runs over all elementary subgroups of G and λ runs over all irreducible characters of H. The algorithm runs over suitably chosen elementary subgroups of G, computes their irreducible characters and induces them to G. In the resulting lattice of generalized characters, it searches for an orthonormal basis by means of LLL lattice reduction. Once an orthonormal basis of size equal to the number of conjugacy classes of G is found, it is clear from character theory that the elements of this basis are up to sign the irreducible characters of G.

1.2 Program

Unger's Algorithm is implemented in the function `CharacterTableUnger` (1.2.1).

1.2.1 CharacterTableUnger

▷ `CharacterTableUnger(`*`G[, Opt]`*`)` (function)

This function computes the character table of a finite group using Unger's algorithm. The argument *G* must be a finite group, the argument *Opt* must be a record which sets some options for the algorithm (more details in Section 1.3 below). Without *Opt*, the irreducible characters are computed once and stored in *G*; with *Opt*, every call computes them anew.

Example

```
gap> G:=AlternatingGroup(6);;
gap> CharacterTableUnger(G);
CharacterTable( Alt( [ 1 .. 6 ] ) )
```

1.2.2 InfoCTUnger

▷ `InfoCTUnger` (info class)

The info class `InfoCTUnger` makes the function `CharacterTableUnger` (1.2.1) display information about the current state of the computation, which may be useful for computations with large groups. If `InfoCTUnger` is set to `1` then the function first displays the number of conjugacy classes of *G* and then shows for any elementary subgroup $E = ZP$ whose characters are induced to *G* the order of Z, the order of P and the number of conjugacy classes of E. If `InfoCTUnger` is set to `2` then it additionally displays the orders of the conjugacy. Moreover, it shows after each LLL reduction the total number of irreducible characters found so far, the dimension of the character lattice and the determinant of the Gram matrix.

Example

```
gap> G:=AlternatingGroup(6);;
gap> SetInfoLevel(InfoCTUnger,1);
gap> CharacterTableUnger(G);
#I  Induce/Restrict: group with 7 conjugacy classes.
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 5, 1, 5 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 4, 1, 4 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
CharacterTable( Alt( [ 1 .. 6 ] ) )

gap> G:=AlternatingGroup(6);;
gap> SetInfoLevel(InfoCTUnger,2);
gap> CharacterTableUnger(G);
#I  Induce/Restrict: group with 7 conjugacy classes.
#I  Induce/Restrict: orders of class reps: [ 1, 2, 3, 3, 4, 5, 5 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 5, 1, 5 ]
#I  Reduce: |Irr| = 1, dim = 4, det(G) = 43
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 4, 1, 4 ]
#I  Reduce: |Irr| = 1, dim = 6, det(G) = 8
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
#I  Reduce: |Irr| = 2, dim = 7, det(G) = 4
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
#I  Reduce: |Irr| = 7
CharacterTable( Alt( [ 1 .. 6 ] ) )
```

1.2.3 CTUngerDefaultOptions

▷ `CTUngerDefaultOptions` (global variable)

The global variable `CTUngerDefaultOptions` contains a record with four components `DoCyclicFirst`, `DoCyclicLast`, `LLLOffset` and `Delta`, which specify the default options for the function `CharacterTableUnger` (1.2.1). The options can be changed manually by the user, for more details see Subsection 1.3.4 below. The initial values for the record components are as follows:

Example

```
gap> CTUngerDefaultOptions;
rec( Delta := 3/4, DoCyclicFirst := false, DoCyclicLast := false, LLLOffset := 0 )
```

1.3 Options

In this section, we explain the options for `CharacterTableUnger` (1.2.1) using the optional second argument *Opt*, which should be a record with components `DoCyclicFirst`, `DoCyclicLast`,

LLLOffset and Delta. You only need to specify those components of the record that you want to manually set to a certain value, the others will be assigned their default value by the program. Additional components of the record will be ignored. The intended use of the different components of *Opt* is as follows:

1.3.1 DoCyclicFirst and DocyclicLast

The component Opt.DoCyclicFirst should be a boolean which is by default set to false. It tells the function CharacterTableUnger (1.2.1) to first induce all irreducible characters of the cyclic subgroups and then proceed to do the same for the non-cyclic elementary subgroups. The option Opt.DoCyclicLast does the opposite, that is, it tells CharacterTableUnger (1.2.1) to first induce the characters of the non-cyclic elementary subgroups and then proceed to induce the characters of the cyclic subgroups. The default value is also false Note that even when Opt.DoCyclicLast is true, it may happen that some cyclic group are used by the algorithm earlier than some non-cyclic elementary subgroups, but only when the group P in $E = ZP$ is cyclic. When both Opt.DoCyclicFirst and Opt.DoCyclicLast are set to be true then the program still induces from cyclic groups first.

Example

```
gap> CharacterTableUnger(G,rec( DoCyclicFirst:=true ));
#I  Induce/Restrict: group with 7 conjugacy classes.
#I  Induce/Restrict: orders of class reps: [ 1, 2, 3, 3, 4, 5, 5 ]
#I  Induce: from cyclic subgroups
#I  Reduce: |Irr| = 7
CharacterTable( Alt( [ 1 .. 6 ] ) )

gap> CharacterTableUnger(G,rec( DoCyclicLast:=true ));
#I  Induce/Restrict: group with 7 conjugacy classes.
#I  Induce/Restrict: orders of class reps: [ 1, 2, 3, 3, 4, 5, 5 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 1, 5, 5 ]
#I  Reduce: |Irr| = 1, dim = 4, det(G) = 43
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 1, 8, 5 ]
#I  Reduce: |Irr| = 5, dim = 6, det(G) = 2
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 1, 9, 9 ]
#I  Reduce: |Irr| = 7
CharacterTable( Alt( [ 1 .. 6 ] ) )
```

1.3.2 LLLOffset

The component Opt.LLLOffset should be an integer telling the function CharacterTableUnger (1.2.1) to not apply LLL reduction to the Gram matrix of the character lattice each time after the characters of some elementary subgroup have been induced. More precisely, the first LLL reduction will be carried out after the characters of the first Opt.LLLOffsetelementary subgroups have been induced. When Opt.DoCyclicFirst is true then the first LLL reduction will be carried out after the characters of the cyclic groups and the first Opt.LLLOffset non-cyclic elementary groups have been induced. The default value for Opt.LLLOffset is 0.

Example

```
gap> CharacterTableUnger(G,rec( LLLOffset:=3 ));
#I  Induce/Restrict: group with 7 conjugacy classes.
#I  Induce/Restrict: orders of class reps: [ 1, 2, 3, 3, 4, 5, 5 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 5, 1, 5 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 4, 1, 4 ]
```

```
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
#I  Reduce: |Irr| = 2, dim = 7, det(G) = 4
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
#I  Reduce: |Irr| = 7
CharacterTable( Alt( [ 1 .. 6 ] ) )
```

1.3.3 Delta

The component `Opt.Delta` can be used to specify the parameter δ for the LLL reduction, where $1/4 < \delta \leq 1$. The default value for `Opt.Delta` is $3/4$ and `Opt.Delta` is ignored if it is not a rational with $1/4 <$ `Opt.Delta` and `Opt.Delta` ≤ 1.

Example

```
gap> SetInfoLevel(InfoCTUnger,2);
gap> G:=AlternatingGroup(6);;
gap> CharacterTableUnger(G,rec( Delta:=3/10 ));
#I  Induce/Restrict: group with 7 conjugacy classes.
#I  Induce/Restrict: orders of class reps: [ 1, 2, 3, 3, 4, 5, 5 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 5, 1, 5 ]
#I  Reduce: |Irr| = 1, dim = 4, det(G) = 43
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 4, 1, 4 ]
#I  Reduce: |Irr| = 1, dim = 6, det(G) = 8
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
#I  Reduce: |Irr| = 2, dim = 7, det(G) = 4
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 3, 1, 3 ]
#I  Reduce: |Irr| = 5, dim = 7, det(G) = 1
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 1, 9, 9 ]
#I  Reduce: |Irr| = 7
CharacterTable( Alt( [ 1 .. 6 ] ) )

gap> CharacterTableUnger(G,rec( Delta:=3/10, DoCyclicLast:=true ));
#I  Induce/Restrict: group with 7 conjugacy classes.
#I  Induce/Restrict: orders of class reps: [ 1, 2, 3, 3, 4, 5, 5 ]
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 1, 5, 5 ]
#I  Reduce: |Irr| = 1, dim = 4, det(G) = 43
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 1, 9, 9 ]
#I  Reduce: |Irr| = 2, dim = 6, det(G) = 3
#I  Induce/Restrict: Trying [|Z|, |P|, k(E)] = [ 1, 8, 5 ]
#I  Reduce: |Irr| = 5, dim = 7, det(G) = 1
#I  Reduce: |Irr| = 7
CharacterTable( Alt( [ 1 .. 6 ] ) )
```

1.3.4 Changing the default options

In cases where the character tables of several groups need to be computed using the same options, it may be useful to change the default options, which are specified in the record `CTUngerDefaultOptions` (1.2.3). This can be done by simply overwriting the the components of that record.

Example

```
gap> CTUngerDefaultOptions;
rec( Delta := 3/4, DoCyclicFirst := false, DoCyclicLast := false, LLLOffset := 0 )
gap> CTUngerDefaultOptions.Delta:=1;;
```

```
gap> CTUngerDefaultOptions.DoCyclicLast:=true;;
gap> CTUngerDefaultOptions;
rec( Delta := 1, DoCyclicFirst := false, DoCyclicLast := true, LLLOffset := 0 )
```

Chapter 2

Installing and Loading the InduceReduce Package

2.1 Installing the package

InduceReduce does not use external binaries and, therefore, works without restrictions on the type of the operating system.
There are two ways of installing a GAP package. If you have permission to add files to the installation of GAP on your system you may install InduceReduce into the `pkg` subdirectory of the GAP installation tree. Otherwise you may install InduceReduce in a private `pkg` directory (for details see Installing a GAP Package and GAP Root Directories in the GAP Reference Manual).

2.2 Loading the package

Once you have installed the package as described in the previous section, you can load it in a GAP session using the command `LoadPackage("InduceReduce")`.

References

[Ung06] W. R. Unger. Computing the character table of a finite group. *J. Symbolic Comput.*, 41(8):847–862, 2006. 4

Index

`CharacterTableUnger`, 4
`CTUngerDefaultOptions`, 5

`InfoCTUnger`, 4

License, 2