Repsn

A GAP4 Package
for constructing representations of finite groups

Version 1.1

May 2004

Vahid Dabbaghian-Abdoly

Vahid Dabbaghian-Abdoly

— Email: vdabbagh@math.carleton.ca

— Homepage: http://www.math.carleton.ca/˜vdabbagh

— Address: School of Mathematics and Statistics,
Carleton University,
Ottawa, Ontario,
K1S 5B6 Canada.

Copyright

Acknowledgements

This package was obtained during my Ph.D. studies at Carleton University. I would like to express deep gratitude to my supervisor Professor John D. Dixon whose guidance and support were crucial for the successful completion of this project. I also thank Professor Charles Wright and referees for pointing out some important comments to improve Repsn.

This documentation was prepared with the GAPDoc package by Frank Lübeck and Max Neunhöffer.

Contents

Chapter 1

Introduction

This manual describes the Repsn package for computing matrix representations in characteristic zero of finite groups. Most of the functions in Repsn have been written according to the algorithm described in the author's Ph.D thesis [DA03].

For constructing representations of simple groups and their covers we use the algorithm described in [Dix93]. To use this algorithm for constructing a representation of a group G affording an irreducible character *chi* of G, we need to have a subgroup H of G such that the restriction of *chi* to H has a linear constituent with multiplicity one. In this case we say H is a *character subgroup* relative to *chi* (or a *chi*-subgroup). A *chi*-subgroup for each irreducible character *chi* of degree less than 32 of simple groups and their covers are listed in [DA03].

All Repsn functions are written entirely in the GAP language. It is proved in [DA03] that the algorithm is correct for any group with a character of degree less than 32. Indeed, if the group is solvable, there is no restriction on the character degree. In practice the program is quite fast when the degree is small, but can be very slow when it is necessary to call one of the subprograms which extend irreducible representations. In the latter case the number of element wise operations required to extend a representation of degree d is proportional to d^6.

Repsn is implemented in the GAP language, and runs on any system supporting GAP4. The Repsn package is loaded into the current GAP session with the command

```
gap> LoadPackage( "repsn" );
```

(see section *Loading a GAP Package* in the GAP Reference Manual). One could install the Repsn package on GAP4.3. In this case it is loaded with the command

```
gap> RequirePackage( "repsn" );
```

Repsn has been developed by

Vahid Dabbaghian-Abdoly
School of Mathematics and Statistics
Carleton University
Ottawa, Ontario,
K1S 5B6 Canada.
e-mail: vdabbagh@math.carleton.ca

Please send bug reports, suggestions and other comments to this e-mail address.

Chapter 2

Irreducible Representations

Let G be a finite group and *chi* be an ordinary irreducible character of G. In this chapter we introduce some functions to construct a complex representation R of G affording *chi*. We proceed recursively, reducing the problem to smaller subgroups of G or characters of smaller degree until we obtain a problem which we can deal with directly. Inputs of most of the functions are a given group G, and an irreducible character *chi*. The output is a mapping (representation) which assigns to each generator x of G a matrix $R(x)$. We can use these functions for all groups and all irreducible characters *chi* of degree less than 32 although in principle the same methods can be extended to characters of larger degree. The main methods in these functions which are used to construct representations of finite groups are induction, extension, tensor product and Dixon's method (for constructing representations of simple groups and their covers).

2.1 Constructing Representations

This section introduces the main function to compute a representation of a finite group G affording an irreducible character *chi* of G.

2.1.1 IrreducibleAffordingRepresentation

◊ `IrreducibleAffordingRepresentation( chi )` (function)

called with an irreducible character `chi` of a group G, this function returns a mapping (representation) which maps each generator of G to a $d*d$ matrix, where d is the degree of `chi`. The group generated by these matrices (the image of the map) is a matrix group which is isomorphic to G modulo the kernel of the map. If G is a solvable group then there is no restriction on the degree of `chi`. In the case that G is not solvable and the character `chi` has degree bigger than 31 the output maybe is not correct. In this case sometimes the output mapping does not afford the given character or it does not return any mapping.

2.1.2 IsAffordingRepresentation

◊ `IsAffordingRepresentation( chi, rep )` (function)

If `chi` and `rep` are a character and a representation of a group G, respectively, then `IsAffordingRepresentation` returns `true` if the trace of `rep(x)` equals `chi(x)` for all elements x in G.

Example

```
gap> G := GL(2,7);:
gap> chi := Irr(G)[ 29 ];;
gap> rep := IrreducibleAffordingRepresentation( chi );
CompositionMapping( [(8,15,22,29,36,43)(9,16,23,30,37,44)
(10,17,24,31,38,45)(11,18,25,32,39,46)(12,19,26,33,40,47)
(13,20,27,34,41,48)(14,21,28,35,42,49), (2,29,12)(3,36,20)
(4,43,28)(5,8,30)(6,15,38)(7,22,46)(9,44,14)(10,16,17)
(11,37,27)(13,23,39)(18,24,25)(19,45,35)(21,31,47)
(26,32,33)(34,40,41)(42,48,49) ] ->
[ [ [ 0, 0, 0, -1, 0, 0, 0 ],
    [ 1, 0, -1, -1, 1, 0, -1 ]
    [ 2, -1, -2, -2, 1, 2, -1 ],
    [ 0, 0, -1, 0, 0, 0, 0 ],
    [ 1, 0, -2, 0, 0, 1, -1 ],
    [ 1, 0, -2, -1, 1, 1, -1 ],
    [ -2, 1, 1, 1, -1, -1, 0 ] ],
  [ [ 1, -1, -1, -1, 0, 2, -1 ],
    [ 0, 0, 1, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, 0, 1, 0 ],
    [ 0, 1, -1, 0, 0, 0, -1 ],
    [ 0, 1, 0, 1, 0, -1, 0 ],
    [ 0, 1, 0, 0, 0, 0, 0 ],
    [ 0, 0, 0, 0, -1, 0, 0 ] ] ], (action isomorphism) )
gap> IsAffordingRepresentation( chi, rep );
true
```

We can obtain the size of the image of this representation by `Size(Image(rep))` and compute the value for an arbitrary element x in G by `x^rep`.

2.2 Induction

2.2.1 InducedSubgroupRepresentation

◊ `InducedSubgroupRepresentation( G, rep )` (function)

computes a representation of `G` induced from the representation `rep` of a subgroup H of `G`. If `rep` has degree d then the degree of the output representation is $d * |G : H|$.

Example

```
gap> G := SymmetricGroup( 6 );;
gap> H := AlternatingGroup( 6 );;
gap> chi := Irr( H )[ 2 ];;
gap> rep := IrreducibleAffordingRepresentation( chi );;
gap> InducedSubgroupRepresentation( G, rep );
[ (1,2,3,4,5,6), (1,2) ] ->
[ [ [ 0, 0, 0, 0, 0, 1, 1, -1, -1, -1 ],
    [ 0, 0, 0, 0, 0, 1, 0, -1, 0, -1 ],
```

```
      [ 0, 0, 0, 0, 0, 1, 0, 0, -1, -1 ],
      [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0 ],
      [ 0, 0, 0, 0, 0, 0, 1, -1, 0, -1 ],
      [ 1, 1, -1, -1, -1, 0, 0, 0, 0, 0 ],
      [ 1, 0, 0, -1, -1, 0, 0, 0, 0, 0 ],
      [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
      [ 1, 0, -1, 0, -1, 0, 0, 0, 0, 0 ],
      [ 0, 1, 0, -1, -1, 0, 0, 0, 0, 0 ] ],
    [ [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0 ],
      [ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
      [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0 ],
      [ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 ],
      [ 0, 0, 0, 0, 0, 1, 1, -1, -1, -1 ],
      [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
      [ 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 ],
      [ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0 ],
      [ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 ],
      [ 1, 1, -1, -1, -1, 0, 0, 0, 0, 0 ] ] ]
```

2.3 Extension

In this section we introduce some functions for extending a representation of a subgroup to the whole group.

2.3.1 ExtendedRepresentation

◊ `ExtendedRepresentation( chi, rep )` (function)

Suppose H is a subgroup of a group G and `chi` is an irreducible character of G such that the restriction of `chi` to H, *phi* say, is irreducible. If `rep` is an irreducible representation of H affording *phi* then `ExtendedRepresentation` extends the representation `rep` of H to a representation of G affording `chi`. This function call can be quite expensive when the representation `rep` has a large degree.

Example

```
gap> G := AlternatingGroup( 6 );;
gap> H := Group([ (1,2,3,4,6), (1,4)(5,6) ]);;
gap> chi := Irr( G )[ 2 ];;
gap> phi := RestrictedClassFunction( chi, H );;
gap> IsIrreducibleCharacter( phi );
true
gap> rep := IrreducibleAffordingRepresentation( phi );;
gap> ext := ExtendedRepresentation( chi, rep );
#I  Need to extend a representation of degree 5. This may take a while.
[ (1,2,3,4,5), (4,5,6) ] -> [
[ [ 0, 1, 0, -1, -1 ],
  [ 0, 0, 0, 1, 0 ],
  [ -1, -1, -1, 0, 0 ],
  [ 0, 0, 0, 0, -1 ],
  [ 0, 0, 1, 1, 1 ] ],
```

```
[ [ 1, 0, 1, 0, 1 ],
  [ 0, 1, 0, 0, 0 ],
  [ -1, -1, 0, 1, 0 ],
  [ 1, 1, 1, 0, 0 ],
  [ 0, 0, -1, 0, 0 ] ] ]
gap> IsAffordingRepresentation( chi, ext );
true
```

2.3.2 ExtendedRepresentationNormal

◊ ExtendedRepresentationNormal(chi, rep) (function)

Suppose H is a normal subgroup of a group G and `chi` is an irreducible character of G such that the restriction of `chi` to H, *phi* say, is irreducible. If `rep` is an irreducible representation of H affording *phi* then `ExtendedRepresentationNormal` extends the representation `rep` of H to a representation of G affording `chi`. This function is more efficient than `ExtendedRepresentation`.

Example

```
gap> G := GL(2,7);;
gap> chi := Irr( G )[ 29 ];;
gap> H := SL(2,7);;
gap> phi := RestrictedClassFunction( chi, H );;
gap> IsIrreducibleCharacter( phi );
true
gap> rep := IrreducibleAffordingRepresentation( phi );;
gap> ext := ExtendedRepresentationNormal( chi, rep );
#I  Need to extend a representation of degree 7. This may take a while.
CompositionMapping( [(8,15,22,29,36,43)(9,16,23,30,37,44)
 (10,17,24,31,38,45)(11,18,25,32,39,46)(12,19,26,33,40,47)
 (13,20,27,34,41,48)(14,21,28,35,42,49),(2,29,12)(3,36,20)
 (4,43,28)(5,8,30)(6,15,38)(7,22,46)(9,44,14)(10,16,17)
 (11,37,27)(13,23,39)(18,24,25)(19,45,35)(21,31,47)
 (26,32,33)(34,40,41)(42,48,49) ] ->
[ [ [ -1, 0, 0, 1, 0, -1, 0 ], [ -1, 0, 0, 0, 0, 0, 0 ],
  [ -1, 1, 0, 0, -1, 0, 0 ], [ 0, -1, 0, 0, 0, 0, 0 ],
  [ -1, -1, 1, 0, 1, -1, 0 ], [ 0, 0, 0, -1, 0, 0, 0 ],
  [ -1, 0, 1, -1, 1, 0, -1 ] ],
  [ [ 1, -1, 0, 1, 0, -1, 1 ], [ 1, 0, -1, 1, -1, 0, 1 ],
  [ 1, -1, 0, 1, -1, 0, 1 ], [ 0, 0, -1, 0, 0, 0, 0 ],
  [ -1, 0, 0, 1, 0, -1, 0 ], [ -1, 0, 0, 0, 0, 0, 0 ],
  [ -1, 1, 0, 0, -1, 0, 0 ] ] ], (action isomorphism) )
gap> IsAffordingRepresentation( chi, ext );
true
```

2.4 Character Subgroups

If *chi* is an irreducible character of a group G and H is a subgroup of G such that the restriction of *chi* to H has a linear constituent with multiplicity one, then we call H a character subgroup relative to *chi* or a *chi*-subgroup.

2.4.1 CharacterSubgroupRepresentation

◊ `CharacterSubgroupRepresentation( G, chi )` (function)
◊ `CharacterSubgroupRepresentation( G, chi, H )` (function)

returns a representation of `G` affording `chi` by finding a `chi`-subgroup and using the method described in [Dix93]. If the third argument is a `chi`-subgroup then it returns a representation of `G` affording `chi` without searching for a `chi`-subgroup. In this case an error is signalled if no `chi`-subgroup exists.

2.4.2 IsCharacterSubgroup

◊ `IsCharacterSubgroup( chi, H )` (function)

is `true` if `H` is a `chi`-subgroup and `false` otherwise.

Example

```
gap> G := AlternatingGroup( 8 );;
gap> chi := Irr( G )[ 2 ];;
gap> H := AlternatingGroup( 3 );;
gap> IsCharacterSubgroup( chi, H );
true
gap> rep := CharacterSubgroupRepresentation( G, chi, H );
[ (1,2,3,4,5,6,7), (6,7,8) ] -> [ [ [
 1/3*E(3)+2/3*E(3)^2, 0, 0, -E(3), 0, -1/3*E(3)-2/3*E(3)^2, 1 ],
   [ 2/3*E(3)+4/3*E(3)^2, 0, 1, 0, 0, 1/3*E(3)-1/3*E(3)^2, 0 ],
   [ 2/3*E(3)+4/3*E(3)^2, 0, 0, 1, 0, 1/3*E(3)-1/3*E(3)^2, 0 ],
   [ E(3)^2, 0, 0, 0, 0, 0, 0 ],
   [ 2/3*E(3)+4/3*E(3)^2, 0, 0, 0, 1, 1/3*E(3)-1/3*E(3)^2, 0 ],
   [ -2/3*E(3)-1/3*E(3)^2, 0, 0, -1, 0, 2/3*E(3)+1/3*E(3)^2, E(3)^2 ],
   [ 0, 1, 0, 0, 0, 0, 0 ] ],
 [ [ 1, 0, 0, 0, 0, 0, 0 ], [ 0, 1, 0, 0, 0, 0, 0 ],
   [ 0, 0, 0, 1, 0, 0, 0 ], [ 0, 0, 0, 0, 1, 0, 0 ],
   [ 0, 0, 1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 0, 1, 0 ],
   [ 0, 0, 0, -E(3), E(3), 0, 1 ] ] ]
```

2.4.3 AllCharacterPSubgroups

◊ `AllCharacterPSubgroups( G, chi )` (function)

returns a list of all *p*-subgroups of `G` which are `chi`-subgroups. The subgroups are chosen up to conjugacy in `G`.

2.4.4 AllCharacterSubgroups

◊ `AllCharacterSubgroups( G, chi )` (function)

returns a list of all `chi`-subgroups of `G` among the lattice of subgroups. This function call can be quite expensive for larger groups. The call is expensive in particular if the lattice of subgroups of the given group is not yet known.

2.5 Equivalent Representation

2.5.1 EquivalentRepresentation

◊ `EquivalentRepresentation( rep )` (function)

computes an equivalent representation to an irreducible representation `rep` by transforming `rep` to a new basis by spinning up one vector (i.e. getting the other basis vectors as images under the first one under words in the generators). If the input representation, `rep`, is not irreducible then `EquivalentRepresentation` does not return any mapping.

Example

```
gap> G := SymmetricGroup( 7 );;
gap> chi := Irr( G )[ 2 ];;
gap> rep := CharacterSubgroupRepresentation( G, chi );;
gap> equ := EquivalentRepresentation( rep );
[ (1,2,3,4,5,6,7), (1,2) ] ->
[ [ [ 0, 0, 0, E(5)+E(5)^2+E(5)^3+2*E(5)^4, -1, -E(5)-E(5)^2-E(5)^3-2*E(5)^4 ],
    [ E(5)^3-E(5)^4, E(5)^2+E(5)^3+E(5)^4, E(5)+E(5)^3-E(5)^4, -E(5)+E(5)^2
           -3*E(5)^3-E(5)^4, -E(5)-E(5)^3+E(5)^4, 2*E(5)-2*E(5)^2+2*E(5)^3 ]
     , [ 0, 0, 0, 1, 0, 0 ],
    [ 0, 4/5*E(5)+3/5*E(5)^2+2/5*E(5)^3+1/5*E(5)^4, E(5), 1, -E(5),
        6/5*E(5)+2/5*E(5)^2+3/5*E(5)^3+4/5*E(5)^4 ], [ 0, 1, 0, 0, 0, 0 ],
    [ 0, 0, E(5), 1, -E(5), 2*E(5)+E(5)^2+E(5)^3+E(5)^4 ] ],
  [ [ -1, 0, E(5)+E(5)^2+E(5)^3+2*E(5)^4, -E(5)-E(5)^2-3*E(5)^4,
     -E(5)-E(5)^2-E(5)^3-2*E(5)^4, E(5)+E(5)^2+3*E(5)^4 ],
   [ 0, -1, 0, 0, 0, 0 ],
   [ 0, 0, 0, E(5)+E(5)^2+E(5)^3+2*E(5)^4, -1, -E(5)-E(5)^2-E(5)^3-2*E(5)^4
      ], [ 0, 0, -1, -E(5)^4, 1, E(5)+E(5)^2+E(5)^3+2*E(5)^4 ],
   [ 0, 0, -E(5)^4, -E(5)^3+E(5)^4, E(5)+E(5)^2+E(5)^3+2*E(5)^4,
       E(5)^3-E(5)^4 ], [ 0, 0, 0, 0, 0, -1 ] ] ]
gap> IsAffordingRepresentation( chi, equ );
true
```

References

[DA03] Vahid Dabbaghian-Abdoly. *An algorithm to construct representations of finite groups*. Ph.D. thesis, Dept. Mathematics, Univ. Carleton, 2003. 4

[Dix93] John D. Dixon. Constructing representations of finite groups. In *Groups and Computation*, volume 11 of *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*, pages 105–112, 1993. 4, 9

Index