IRREDSOL

Version 1r0n1

A library of irreducible solvable matrix groups over finite fieldss

Burkhard Höfling

Institut für Geometrie, Algebra und Diskrete Mathematik
Technische Universität
Braunschweig, Germany
b.hoefling@tu-bs.de

Contents

1 Overview

Let n be a positive integer and q a prime power satisfying $q^n \leq 2^{16}-1$. The package IRREDSOL constitutes a library of irreducible solvable subgroups of $GL(n, q)$, such that each irreducible solvable subgroup of $GL(n, q)$ is conjugate to precisely one group in the library.

This data base is intended to replace and extend the data base of irreducible matrix groups over prime fields created by Mark Short [Sho92] which is also part of GAP. There are functions available to translate from Mark Short's numbering of groups to the numbering used in IRREDSOL and back; see 3.3.

It is possible to construct groups in this list one at a time (see 2.1), by supplying four integer parameters identifying the group in question. In addition, there are functions which facilitate searching the library for groups with given properties (see 2.2).

Given an irreducible solvable matrix group G, it is possible to find the group in the library to which G is conjugate, see 3.1 and 3.2.

Moreover,, the IRREDSOL package provides additional functionality for matrix groups, such as the computation of imprimitivity systems.

Finally, the library of irreducible solvable matrix groups over prime fields translates into a library of isomorphism types of primitive solvable groups. There are functions avalable which faciliate this translation; see 4.

The methods used to construct the data base are slight modifications of those described by Bettina Eick and the author in [EH03].

2 Accessing the data library

This chapter describes various ways of accessing groups in the data library. It is possible to access individual groups in the lists provided (see Section 2.1), or to search for groups with certain properties (see Section 2.2).

2.1 Low level access functions

The access functions described in this section allow to check for the availability of data, the number of groups in each list $\mathcal{A}_{n,q}$, individual groups in such a list, and the corresponding irreducible but not absolutely irreducible groups (see Chapter 1).

1 ▶ `IsAvailableIrreducibleSolvableGroupData(`n`, `q`)` F

This function tests whether the irreducible solvable subgroup of $GL(n, q)$ which cannot be written over a proper subfield of $\mathbb{F}_q$ are part of the IRREDSOL library.

2 ▶ `IndicesIrreducibleSolvableMatrixGroups(`n`, `q`, `d`)` F

Let n, d be positive integers and q a prime power. This function returns a set of integers parametrising the groups in the IRREDSOL library which are subgroups of $GL(n, q)$ that cannot be written over a proper subfield of $\mathbb{F}_q$ and have splitting field $\mathbb{F}_{q^d}$. This set is empty unless d divides n. An error is raised if the relevant data is not available (see 2.1.1 for information how to check this first).

3 ▶ `IrreducibleSolvableMatrixGroup(`n`, `q`, `d`, `k`)` F

Let n be a positive integer and q a prime power. This function returns the k-th irreducible solvable subgroup of $GL(n, q)$ which cannot be written over a proper subfield of $\mathbb{F}_q$. An error is raised if the relevant data is not available (see 2.1.1 for information how to check this first), or if k is not in `IndicesIrreducibleSolvableMatrixGroups(`n`, `q`, `d`)` (see 2.1.2). For the groups returned, the attrbutes and properies described in 5 are set to their appropriate values.

4 ▶ `IsAvailableAbsolutelyIrreducibleSolvableGroupData(`n`, `q`)` F

This function tests whether the absolutely irreducible solvable subgroup of $GL(n, q)$ which cannot be written over a proper subfield of $\mathbb{F}_q$ are in the IRREDSOL library.

5 ▶ `IndicesAbsolutelyIrreducibleSolvableMatrixGroups(`n`, `q`)` F

Let n be a positive integer and q a prime power. This function returns a set of integers parametrising the absolutely irreducible groups in the IRREDSOL library which are subgroups of $GL(n, q)$ that cannot be written over a proper subfield of $\mathbb{F}_q$.

6 ▶ `AbsolutelyIrreducibleSolvableMatrixGroup(`n`, `q`, `k`)` F

Let n be a positive integer and q a prime power. This function returns the k-th absolutely irreducible solvable subgroup of $GL(n, q)$ which cannot be written over a proper subfield of $\mathbb{F}_q$. An error is raised if the relevant data is not available (see 2.1.4 for information how to check this first), or if k does not lie

in `IndicesAbsolutelyIrreducibleSolvableMatrixGroups`(n, q) (see 2.1.5). For the groups returned, the attrbutes and properies described in 5 are set to their appropriate values.

7 ► `IndicesMaximalAbsolutelyIrreducibleSolvableMatrixGroups`(*n*, *q*) F

Let n be a positive integer and q a prime power. This function returns a set of integers parametrising the absolutely irreducible groups in the IRREDSOL library which are subgroups of $GL(n, q)$ that cannot be written over a proper subfield of $\mathbb{F}_q$and which is maximal with respect to being solvable. An error is raised if the relevant data is not available (see 2.1.4 for information how to check this first).

```
gap> Reread("/Macintosh HD/Applications (Mac OS 9)/gap/4.0/pkg/irredsol/lib/loading.gi");
gap> inds := IndicesMaximalAbsolutelyIrreducibleSolvableMatrixGroups (2,3);
[ 2 ] # there is only one maximal solvable subgroup of GL(2,3)
gap> max := AbsolutelyIrreducibleSolvableMatrixGroup (2,3,2); # construct it
Group([ [ [ Z(3), 0*Z(3) ], [ 0*Z(3), Z(3)^0 ] ], [ [ Z(3)^0, Z(3) ], [ 0*Z(3), Z(3)^0 ] ],
  [ [ Z(3), Z(3)^0 ], [ Z(3)^0, Z(3)^0 ] ], [ [ 0*Z(3), Z(3)^0 ], [ Z(3), 0*Z(3) ] ],
  [ [ Z(3), 0*Z(3) ], [ 0*Z(3), Z(3) ] ] ])
gap> max = GL(2,3); # it is the whole GL
true
```

2.2 Finding matrix groups with given properties

This section describes three functions (`AllIrreducibleSolvableMatrixGroups`, `OneIrreducibleSolvableMatrixGroup`, `IteratorIrreducibleSolvableMatrixGroups`) which allow you to find matrix groups with special properties. These functions are usually more efficient than to construct each group in the library using the functions in Section 2.1.

1 ► `AllIrreducibleSolvableMatrixGroups`(*func_1*, *arg_1*, *func_2*, *arg_2*, ...) F

This function returns a list of all irreducible solvable matrix groups G in the IRREDSOL library for which the return value of *func_i*(G) lies in *arg_i*. The arguments *func_1*, *func_2*, ..., must be GAP functions which take matrix group as their only argument and return a value, and *arg_1*, *arg_2*, ..., must be lists. If *arg_i* is not a list, *arg_i* is replaced by the list `[`*arg_i*`]`. One of the functions must be `DegreeOfMatrixGroup` (or one of its equivalents, see below), and one function must be `FieldOfMatrixGroup` (or `Field`). All groups G in the data library have the property that they cannot be written over a subfield of `FieldOfMatrixGroup`(G); see 6.1 for details. For the groups returned, the attrbutes and properies described in 5 are set to their appropriate values.

Note that there is also a function `IteratorIrreducibleSolvableMatrixGroups` (see 2.2.3) which allows to run through the list produced by `AllIrreducibleSolvableMatrixGroups` without having to store all of the groups simultaneously.

The following functions *func_i* are handled particularly efficiently. For the definitions of most of these functions, see Chapter 5.

- `DegreeOfMatrixGroup` (or `Degree`, `Dimension`, `DimensionOfMatrixGroup`),
- `CharacteristicOfField` (or `Characteristic`)
- `FieldOfMatrixGroup` (or `Field`)
- `Order` (or `Size`)
- `IsMaximalAbsolutelyIrreducibleSolvableMatrixGroup`
- `IsAbsolutelyIrreducibleMatrixGroup` (or `IsAbsolutelyIrreducible`)
- `MinimalBlockDimensionOfMatrixGroup` (or `MinimalBlockDimension`)
- `IsPrimitiveMatrixGroup` (or `IsPrimitive`, `IsLinearlyPrimitive`)

Except for groups which are irreducible but not absolutely irreducible and functions `MinimalBlockDimensionOfMatrixGroup` and `IsPrimitiveMatrixGroup` (or their equivalents), the return values of the above functions can be read off the IRREDSOL library without actually constructing the relevant matrix group. But even in this last case, some groups can be ruled out before constructing them.

```
# get just those groups which cannot be written over GF(3)
gap> l := AllIrreducibleSolvableMatrixGroups (Degree, 1, Field, GF(9));;
gap> List (l, Order);
[ 4, 8 ]
# get all irreducible subgroups
gap> l := AllIrreducibleSolvableMatrixGroups (Degree, 1, Field, Subfields (GF(9)));;
gap> List (l, Order);
[ 1, 2, 4, 8 ]
# get only maximal absolutely irreducible ones
gap> l := AllIrreducibleSolvableMatrixGroups (Degree, 4, Field, GF(3),
>             IsMaximalAbsolutelyIrreducibleSolvableMatrixGroup, true);
gap> SortedList (List (l, Order));
[ 320, 640, 2304, 4608 ]
gap> l := AllIrreducibleSolvableMatrixGroups (Degree, 4, Field, GF(3),
> IsAbsolutelyIrreducibleMatrixGroup, true);;
gap> Collected (List (l, Order));
[ [ 20, 1 ], [ 32, 7 ], [ 40, 2 ], [ 64, 10 ], [ 80, 2 ], [ 96, 6 ],
  [ 128, 9 ], [ 160, 3 ], [ 192, 9 ], [ 256, 6 ], [ 288, 1 ], [ 320, 2 ],
  [ 384, 4 ], [ 512, 1 ], [ 576, 3 ], [ 640, 1 ], [ 768, 1 ], [ 1152, 4 ],
  [ 2304, 3 ], [ 4608, 1 ] ]
```

2 ▶ `OneIrreducibleSolvableMatrixGroup`(*func_1*, *arg_1*, *func_2*, *arg_2*, ...) F

This function returns a matrix group G from the IRREDSOL library such that *func_i*(G) lies in *arg_i*, or `fail` if no such group exists. The arguments *func_1*, *func_2*, ..., must be GAP functions taking one argument and returning a value, and *arg_1*, *arg_2*, ..., must be lists. If *arg_i* is not a list, *arg_i* is replaced by the list [*arg_i*]. One of the functions must be `DegreeOfMatrixGroup` (or one of its equivalents, see below), and one function must be `FieldOfMatrixGroup` (or `Field`). All groups G in the data library have the property that they cannot be written over a subfield of `FieldOfMatrixGroup`(G); see see 6.1 for details. For the group returned, the attrbutes and properies described in 5 are set to their appropriate values.

For a list of functions which are handled particularly efficiently, see `AllIrreducibleSolvableMatrixGroups` (2.2.1).

3 ▶ `IteratorIrreducibleSolvableMatrixGroups`(*func_1*, *arg_1*, *func_2*, *arg_2*, ...) F

This function returns an iterator which runs through the list of all matrix groups G in the IRREDSOL library such that *func_i*(G) lies in *arg_i*. The arguments *func_1*, *func_2*, ..., must be GAP functions taking one argument and returning a value, and *arg_1*, *arg_2*, ..., must be lists. If *arg_i* is not a list, *arg_i* is replaced by the list [*arg_i*]. One of the functions must be `DegreeOfMatrixGroup` (or one of its equivalents, see below), and one function must be `FieldOfMatrixGroup` (or `Field`).

For a list of functions which are handled particularly efficiently, see `AllIrreducibleSolvableMatrixGroups` (2.2.1).

Using

`IteratorIrreducibleSolvableMatrixGroups`(*func_1*, *arg_1*, *func_2*, *arg_2*, ...))

is functionally equivalent to

`Iterator`(`AllIrreducibleSolvableMatrixGroups`(*func_1*, *arg_1*, *func_2*, *arg_2*, ...))

(see 28.7 for details) but does not compute all relevant matrix groups at the same time. This may save some memory. For the groups returned, the attrbutes and properies described in 5 are set to their appropriate values.

3 Recognition of matrix groups

This chapter describes some functions which, given an irreducible matrix group, identify a group in the IRREDSOL library which is conjugate to that group.

3.1 Identification of irreducible groups

1 ▶ `IdIrreducibleSolvableMatrixGroupAvailable(`G`)` F

This function returns true if `IdIrreducibleSolvableMatrixGroup` (see 3.1.2 will work for the group G.

2 ▶ `IdIrreducibleSolvableMatrixGroup(`G`)` A

If the matrix group G is solvable and irreducible over F = `FieldOfMatrixGroup`(G), (see 42.1.3), and a conjugate in $GL(n, F)$ of G belongs to the database of irreducible solvable groups in IRREDSOL, this function returns a list `[`n`, `q`, `d`, `k`]` such that G is conjugate to `IrreducibleSolvableMatrixGroup`(n, q, d, k) (see 2.1.3).

3.2 Identification of absolutely irreducible groups

1 ▶ `IdAbsolutelyIrreducibleSolvableMatrixGroupAvailable(`G`)` F

This function returns true if `IdAbsolutelyIrreducibleSolvableMatrixGroup` (see 3.2.2 will work for the group G.

2 ▶ `IdAbsolutelyIrreducibleSolvableMatrixGroup(`G`)` A

If the matrix group G is solvable and absolutely irreducible, and if a conjugate in $GL(n, F)$ of G belongs to the database of irreducible solvable groups in IRREDSOL, this function returns a list `[`n`, `q`, `k`]` such that G is conjugate to `AbsolutelyIrreducibleSolvableMatrixGroup`(n, q, k) (see 2.1.6).

3 ▶ `RecognitionAbsolutelyIrreducibleSolvableMatrixGroup(`G`, `*wantmat*`, `*wantgroup*`)` F
▶ `RecognitionAbsolutelyIrreducibleSolvableMatrixGroupNC(`G`, `*wantmat*`,`*wantgroup*`)` F

Let G be an absolutely irreducible solvable matrix group over a finite field. Theses functions identify a conjugate H of G group in the library. They return a record which has the following entries:

`id`
: contains the id of H (and thus of G); cf. `IdAbsolutelyIrreducibleSolvableMatrixGroup` (3.2.2

`mat` (optional)
: a matrix x such that $G^x = H$

`group` (optional)
: the group H

The entries `mat` and `group` are only present if the booleans *wantmat* and/or *wantgroup* are true, respectively. Note that in most cases, the function may be much slower if wantmat is set to true.

The `NC` version does not check its arguments. It returns `fail` if the group G is beyond the scope of the IRREDSOL library; see `IdAbsolutelyIrreducibleSolvableMatrixGroupAvailable` (3.2.1), while the ordinary version raises an error in this case.

3.3 Compatibility with other data libraries

A library of irreducible solvable subgroups of $GL(n,p)$, where p is a prime and $p^n leq 255$ already exists in GAP. The following functions allow to translate between these libraries.

1 ▶ `IdIrreducibleSolvableMatrixGroupIndexMS(`n`, `p`, `k`)` F

This function returns the id (see 3.1.2) of G, where G is `IrreducibleSolvableGroupMS`(n, p, k) (see 48.11.1).

2 ▶ `IndexMSIdIrreducibleSolvableMatrixGroup(`n`, `q`, `d`, `k`)` F

This function returns a triple $[n, p, l]$ such that `IrreducibleSolvableGroupMS`(n, p, l)' (see 48.11.1) is conjugate to `IrreducibleSolvableMatrixGroup`(n, q, d, k) (see 2.1.3).

4 Primitive solvable groups

A finite group G is primitive if it has a maximal subgroup M with trivial core; the group acts primitively on the cosets of such a maximal subgroup. If G is solvable, there is a unique conjugacy class of such maximal subgroups; the index of M in G is called the degree of G. The degree of G is always a prime power, p^n, say. There exists a well known bijection between the isomorphpism types of primitive solvable groups of degree p^n and the conjugacy classes of irreducible subgrups of $GL(n, p)$.

The IRREDSOL package provides functions for performing these translations, described in 4.1. Moreover, there are functions for finding primitive solvable groups with given properties, see 4.2.

4.1 Translating irreducible solvable matrix groups into primitive solvable groups and back

1 ▶ `PrimitivePcGroupIrreducibleMatrixGroup(`G`)` F
▶ `PrimitivePcGroupIrreducibleMatrixGroupNC(`G`)` F

For a given irreducible solvable matrix group G over a prime field, this function returns a primitive pc group which is the split extension of G with its natural underlying vector space. The NC version does not check whether G is over a prime field, or whether G is irreducible.

2 ▶ `IrreducibleMatrixGroupPrimitivePcGroup(`G`)` F
▶ `IrreducibleMatrixGroupPrimitivePcGroupNC(`G`)` F

For a given primitive solvable group G, this function returns the corresponding irreducible matrix group, that is, the matrix group obtained from the conjugation action of G on its unique minimal normal subgroup N, regarded as a vector space over $GF(p)$, where p is the exponent of N.

4.2 Finding primitive solvable groups with given properties

1 ▶ `AllPrimitivePcGroups(`*func_1*`,` *arg_1*`,` *func_2*`,` *arg_2*`, ...)` F

This function returns a list of all primitive solvable groups G in the IRREDSOL librar for which the return value of *func_i*(G) lies in *arg_i*. The arguments *func_1*, *func_2*, ..., must be GAP functions which take pc group as their only argument and return a value, and *arg_1*, *arg_2*, ..., must be lists. If *arg_i* is not a list, *arg_i* is replaced by the list `[`*arg_i*`]`. One of the functions must be `Degree` (or one of its equivalents, see below).

Note that there is also a function `IteratorPrimitivePcGroups` (see 4.2.3) which allows to run through the list produced by `AllPrimitivePcGroups` without having to store all of the groups simultaneously.

The following functions *func_i* are handled particularly efficiently.

- `Degree`, `NrMovedPoints`, `LargestMovedPoint`
- `Order`, `Size`

2 ► `OnePrimitivePcGroup(`*func_1*`,` *arg_1*`,` *func_2*`,` *arg_2*`, ...)` F

This function returns one primitive solvable groups G in the IRREDSOL librarfor which the return value of *func_i*(G) lies in *arg_i*, or `fail` if no such group exists. The arguments *func_1*, *func_2*, ..., must be GAP functions which take pc group as their only argument and return a value, and *arg_1*, *arg_2*, ..., must be lists. If *arg_i* is not a list, *arg_i* is replaced by the list `[`*arg_i*`]`. One of the functions must be `Degree`.

For a list of functions which are handled particularly efficiently, see `AllPrimitivePcGroups` (4.2.1).

3 ► `IteratorPrimitivePcGroups(`*func_1*`,` *arg_1*`,` *func_2*`,` *arg_2*`, ...)` F

This function returns an iterator which runs through the list of all primitive solvable groups G in the IRREDSOL library such that *func_i*(G) lies in *arg_i*. The arguments *func_1*, *func_2*, ..., must be GAP functions taking one argument and returning a value, and *arg_1*, *arg_2*, ..., must be lists. If *arg_i* is not a list, *arg_i* is replaced by the list `[`*arg_i*`]`. One of the functions must be `Degree`. For a list of functions which are handled particularly efficiently, see `AllPrimitivePcGroups` (4.2.1).

Using

`IteratorPrimitivePcGroups(`*func_1*, *arg_1*, *func_2*, *arg_2*, ...`)`

is functionally equivalent to

`Iterator(AllPrimitivePcGroups(`*func_1*, *arg_1*, *func_2*, *arg_2*, ...`))`

(see 28.7 for details) but does not compute all relevant matrix groups at the same time. This may save some memory.

5 Additional functionality for matrix groups

This chapter explains some attributes, properties and operations which may be useful for working with matrix groups. Some of these are part of the GAP library and have been included for the sake of completeness, and some are provided by the package IRREDSOL. Note that groups constructed by functions in IRREDSOL already have the appropriate properties and attributes.

5.1 Basic attributes for matrix groups

1 ▶ `DegreeOfMatrixGroup(`G`)` F
▶ `Degree(`G`)` O
▶ `DimensionOfMatrixGroup(`G`)` A
▶ `Dimension(`G`)` O

This is the degree of the matrix group or, equivalently, the dimension of the natural underlying vector space. See also 42.1.1.

2 ▶ `FieldOfMatrixGroup(`G`)` A

This is the field generated by the matrix entries of the elements of G. See also 42.1.3.

3 ▶ `CharacteristicOfField(`G`)` A
▶ `Characteristic(`G`)` O

This is the characteristic of `FieldOfMatrixGroup` (see 5.1.2).

4 ▶ `RepresentationIsomorphism(`G`)` A

This attribute stores an isomorphism $H \to G$, where H is a group in which computations can be carried out more efficiently than in G, and the isomorphism can be evalueated easily. It is usually advantageous to carry out computations in H, and to translate them to G via `RepresentationIsomorphism`(G). (Note that the inverse mapping of `RepresentationIsomorphism` is not required to be efficient.)

5.2 Irreducibility and maximality of matrix groups

1 ▶ `IsIrreducibleMatrixGroup(`G`)` A
▶ `IsIrreducibleMatrixGroup(`G`, `F`)` O
▶ `IsIrreducible(`G` [, `F`])` O

The matrix group G of degree d is irreducible over the field F if no subspace of F^d is invariant under the action of G. If F is not specified, $F =$ `FieldOfMatrixGroup`(G) is assumed.

2 ▶ `IsAbsolutelyIrreducibleMatrixGroup(`G`)` A

If present, this attribute is true if G is absolutely irreducible, i. e., irreducible over any extension field of `FieldOfMatrixGroup`(G).

3 ▶ `IsMaximalAbsolutelyIrreducibleSolvableMatrixGroup(`G`)` A

This attribute, if present, is true if, and only if, G is absolutely irreducible and maximal among the soluble subgroups of $GL(d, F)$, where $d =$ `DegreeOfMatrixGroup`(G) and $F =$ `FieldOfMatrixGroup`(G).

5.3 Primitivity of matrix groups

1 ▶ `MinimalBlockDimensionOfMatrixGroup(`G`)` A
▶ `MinimalBlockDimensionOfMatrixGroup(`G`, `F`)` O
▶ `MinimalBlockDimension(`G` [, `F`])` O

Let G be a matrix group of degree d over the field F. A decomposition $V_1 \oplus \cdots \oplus V_k$ of F^d into F-subspaces V_i is a block system of G if the V_i are permuted by the natural action of G. Obviously, all V_i have the same dimension; this is the dimension of the block system $V_1 \oplus \cdots \oplus V_k$. The function `MinimalBlockDimensionOfMatrixGroup` returns the minimum of the dimensions of all block systems of G. If F is not specified, $F =$ `FieldOfMatrixGroup`(G) is assumed.

2 ▶ `IsPrimitiveMatrixGroup(`G`)` A
▶ `IsPrimitiveMatrixGroup(`G`, `F`)` O
▶ `IsLinearlyPrimitive(`G` [, `F`])` F
▶ `IsPrimitive(`G` [, `F`])` O

The matrix group G of degree d is primitive over the field F if it only has the trivial block system F^d or, equivalently, if `MinimalBlockDimensionOfMatrixGroup`$(G, F) = d$. If F is not specified, $F =$ `FieldOfMatrixGroup`(G) is assumed.

5.4 Conjugating matrix groups into smaller fields

1 ▶ `TraceField(`G`)` A

This is the field generated by the traces of the elements of G. If G is a matrix group over a finite field, then By Brauer's theorem, G has a conjugate which is a matrix group over `TraceField`(G).

2 ▶ `ConjugatingMatTraceField(`G`)` A

If bound, this is a matrix x over `FieldOfMatrixGroup`(G) such that G^x is a matrix group over `TraceField`(G). Currently, there are only methods available for absolutely irreducible groups G (described in [GH97]) and certain trivial cases.

6 Advanced information

This chapter gives some additional information about the IRREDSOL library. It describes details about the library design (see 6.1) and some advanced functions for managing data in the GAP workspace, which may be useful if you run out of memory. See 6.2 and 6.3.

6.1 Design of the group library

To avoid redundancy, the package IRREDSOL does not store lists of irreducible subgroups of $GL(n, q)$ but only has lists $\mathcal{A}_{n,q}$ of subgroups of $GL(n, q)$ such that

- each group in $\mathcal{A}_{n,q}$ is absolutely irreducible and solvable
- $\mathcal{A}_{n,q}$ contains a conjugate of each absolutely irreducible solvable subgroup of
- no two groups in $\mathcal{A}_{n,q}$ are conjugate
- no conjugate of G is a subgroup of $GL(n, q_0)$, where $GF(q_0)$ is a proper subfield of $GF(q)$.

Here, "conjugate" means "conjugate in $GL(n, q)$". We will briefly say that $\mathcal{A}_{n,q}$ contains, up to conjugacy, all absolutely irreducible solvable subgroups of $GL(n, q)$ which cannot be written over a proper subfield.

These lists are sufficient to reconstruct lists of irreducible soluble subgroups of $GL(n, q)$, since any such subgroup can be obtained from an absolutely irreducible subgroup of $GL(n/d, q^d)$, where d divides n, by regarding the underlying $\mathbb{F}_{q^d}$-vector space as an $\mathbb{F}_q$-vector space. Note that two subgroups of $GL(n, q)$ constructed in that way are conjugate if, and only if, the images of the corresponding subgroups of under Galois automorphisms of $\mathbb{F}_{q^d}/\mathbb{F}_q$ are conjugate in $GL(n/d, q^d)$. This information also forms part of the IRREDSOL library.

Note that by the Deuring-Noether theorem, two subgroups of $GL(n, q)$ are conjugate in $GL(n, q)$ if, and only if, they are conjugate in $GL(n, q^d)$ for some $d > 1$. This ensures that lists of irreducible subgroups obtained from the $\mathcal{A}_{n,q}$ do not contain conjugate subgroups.

6.2 Loading and unloading group data manually

The functions in this section are not required for normal use of the data library. They are only relevant if timing or conservation of memory is an issue.

1 ▶ `LoadAbsolutelyIrreducibleSolvableGroupData(`n`, `q`)` F

This function loads the data for $GL(n, q)$ into memory and does some pre-processing. If the data is already loaded, the function does nothing. This function is called automatically when you access the IRREDSOL library, so most users will not need this function.

2 ▶ `LoadedAbsolutelyIrreducibleSolvableGroupData()` F

This function returns a list. Each entry consists of an integer n and a set l. The set l contains all prime powers q such that the group data for $GL(n, q)$ is currently in memory.

3 ► `UnloadAbsolutelydIrreducibleSolvableGroupData([n [,q]])` F

This function can be used to delete data for irreducible groups from the GAP workspace. If no argument is given, all data will be deleted. If only n is given, all data for degree n (and any q) will be deleted. If n and q are given, only the data for $GL(n, q)$ will be deleted from the GAP workspace. Use this function if you run out of GAP workspace. The data is automatically re-loaded when required.

6.3 Loading and unloading recognition data manually

The functions in this section are not required for normal use of the data library. They are only relevant if timing or conservation of memory is an issue.

1 ► `LoadedAbsolutelyIrreducibleSolvableGroupFingerprints()` F

This function returns a list. Each entry consists of an integer n and a set l. The set l contains all prime powers q such that the recognition data for $GL(n, q)$ is currently in memory.

2 ► `UnloadAbsolutelyIrreducibleSolvableGroupFingerprints([n [,q]])` F

This function can be used to delete recognition for irreducible groups from the GAP workspace. If no argument is given, all data will be deleted. If only n is given, all data for degree n (and any q) will be deleted. If n and q are given, only the data for $GL(n, q)$ will be deleted from the GAP workspace. Use this function if you run out of GAP workspace. The data is automatically re-loaded when required.

Bibliography

[EH03] Bettina Eick and Burkhard Höfling. The solvable primitive permutation groups of degree at most 6560. *LMS J. Comput. Math.*, 6:29–39, 2003.

[GH97] S. P. Glasby and R. B. Howlett. Writing representations over mnimal fields. *Comm. Alg.*, 25:1703–1711, 1997.

[Sho92] Mark W. Short. *The Primitive Soluble Permutation Groups of Degree less than 256*, volume 1519 of *Lecture Notes in Math.* Springer, 1992.

Index

This index covers only this manual. A page number in *italics* refers to a whole section which is devoted to the indexed subject. Keywords are sorted with case and spaces ignored, e.g., "`PermutationCharacter`" comes before "permutation group".

P

R

T

U

W