16.68 SYMMETRY: Operations on symmetric matrices

This package computes symmetry-adapted bases and block diagonal forms of matrices which have the symmetry of a group. The package is the implementation of the theory of linear representations for small finite groups such as the dihedral groups.
Author: Karin Gatermann.

16.68.1 Introduction

The exploitation of symmetry is a very important principle in mathematics, physics and engineering sciences. The aim of the SYMMETRY package is to give an easy access to the underlying theory of linear representations for small groups. For example the dihedral groups D_3, D_4, D_5, D_6 are included. For an introduction to the theory see SERRE [3] or STIEFEL and FÄSSLER [4]. For a given orthogonal (or unitarian) linear representation

$$\vartheta : G \longrightarrow GL(K^n), \qquad K = R, C.$$

the character $\psi \rightarrow K$, the canonical decomposition or the bases of the isotypic components are computed. A matrix A having the symmetry of a linear representation,e.g.

$$\vartheta_t A = A\vartheta_t \quad \forall\, t \in G,$$

is transformed to block diagonal form by a coordinate transformation. The dependence of the algorithm on the field of real or complex numbers is controled by the switch `complex`. An example for this is given in the testfile *symmetry.tst*.
As the algorithm needs information concerning the irreducible representations this information is stored for some groups (see the operators in Section 3). It is assumed that only orthogonal (unitar) representations are given.
The package is loaded by

```
load symmetry;
```

16.68.2 Operators for linear representations

First the data structure for a linear representation has to be explained. *representation* is a list consisting of the group identifier and equations which assign matrices to the generators of the group.
Example:

```
rr:=mat((0,1,0,0),
        (0,0,1,0),
        (0,0,0,1),
```

```
          (1,0,0,0));

sp:=mat((0,1,0,0),
          (1,0,0,0),
          (0,0,0,1),
          (0,0,1,0));

representation:={D4,rD4=rr,sD4=sp};
```

For orthogonal (unitarian) representations the following operators are available.

`canonicaldecomposition(representation);`
returns an equation giving the canonical decomposition of the linear representation.

`character(representation);`
computes the character of the linear representation. The result is a list of the group identifier and of lists consisting of a list of group elements in one equivalence class and a real or complex number.

`symmetrybasis(representation,nr);`
computes the basis of the isotypic component corresponding to the irreducible representation of type nr. If the nr-th irreducible representation is multidimensional, the basis is symmetry adapted. The output is a matrix.

`symmetrybasispart(representation,nr);`
is similar as `symmetrybasis`, but for multidimensional irreducible representations only the first part of the symmetry adapted basis is computed.

`allsymmetrybases(representation);`
is similar as `symmetrybasis` and `symmetrybasispart`, but the bases of all isotypic components are computed and thus a complete coordinate transformation is returned.

`diagonalize(matrix,representation);`
returns the block diagonal form of matrix which has the symmetry of the given linear representation. Otherwise an error message occurs.

`on complex;`
Of course the property of irreducibility depends on the field K of real or complex numbers. This is why the algorithm depends on K. The type of computation is set by the switch *complex*.

16.68.3 Display Operators

In this section the operators are described which give access to the stored information for a group. First the operators for the abstract groups are given. Then it is described how to get the irreducible representations for a group.

`availablegroups();`

returns the list of all groups for which the information such as irreducible representations is stored. In the following `group` is always one of these group identifiers.
`printgroup(group);`
returns the list of all group elements;
`generators(group);`
returns a list of group elements which generates the group. For the definition of a linear representation matrices for these generators have to be defined.
`charactertable(group);`
returns a list of the characters corresponding to the irreducible representations of this group.
`charactern(group,nr);`
returns the character corresponding to the nr-th irreducible representation of this group as a list (see also `character`).
`irreducibleреptable(group);`
returns the list of irreducible representations of the group.
`irreduciblerepnr(group,nr);`
returns an irreducible representation of the group. The output is a list of the group identifier and equations assigning the representation matrices to group elements.

16.68.4 Storing a new group

If the user wants to do computations for a group for which information is not predefined, the package SYMMETRY offers the possibility to supply information for this group.
For this the following data structures are used.
elemlist = list of identifiers.
relationlist = list of equations with identifiers and operators @ and $**$.
grouptable = matrix with the (1,1)-entry grouptable.
filename = "myfilename.new".

The following operators have to be used in this order.
`setgenerators(group,elemlist,relationlist);`
Example:

```
setgenerators(K4,{s1K4,s2K4},
  {s1K4^2=id,s2K4^2=id,s1K4@s2K4=s2K4@s1K4});
```

setelements(group,relationlist);

The group elements except the neutral element are given as product of the defined generators. The neutral element is always called `id`.
Example:

```
setelements(K4,
     {s1K4=s1K4,s2K4=s2K4,rK4=s1K4@s2K4});
```

setgrouptable(group,grouptable);
installs the group table.
Example:

```
tab:=
 mat((grouptable,     id,    s1K4, s2K4, rK4),
     (id        ,     id,    s1K4, s2K4, rK4),
     (s1K4      ,    s1K4,     id,  rK4,s2K4),
     (s2K4      ,    s2K4,    rK4,   id,s1K4),
     (rK4       ,     rK4,   s2K4, s1K4,  id));

setgrouptable(K4,tab);
```

Rsetrepresentation(representation,type);
is used to define the real irreducible representations of the group. The variable `type` is either *realtype* or *complextype* which indicates the type of the real irreducible representation.
Example:

```
eins:=mat((1));
mineins:=mat((-1));
rep3:={K4,s1K4=eins,s2K4=mineins};
Rsetrepresentation(rep3,realtype);
```

Csetrepresentation(representation);
This defines the complex irreducible representations.
setavailable(group);
terminates the installation of the group203. It checks some properties of the irreducible representations and makes the group available for the operators in Sections 2 and 3.
storegroup(group,filename);
writes the information concerning the group to the file with name *filename*.
loadgroups(filename);
loads a user defined group from the file *filename* into the system.

Bibliography

[1] G. James, A. Kerber: *Representation Theory of the Symmetric Group.* Addison, Wesley (1981).

[2] W. Ludwig, C. Falter: *Symmetries in Physics.* Springer, Berlin, Heidelberg, New York (1988).

[3] J.–P. Serre, *Linear Representations of Finite Groups*. Springer, New York (1977).

[4] E. Stiefel, A. Fässler, *Gruppentheoretische Methoden und ihre Anwendung*. Teubner, Stuttgart (1979). (English translation to appear by Birkhäuser (1992)).