CrystCat

The Crystallographic Groups Catalog

A GAP4 Package

Version 1.1.6

Volkmar Felsch

Lehrstuhl D für Mathematik

RWTH-Aachen, Templergraben 55, D-52056 Aachen

volkmar.felsch@math.rwth-aachen.de

Franz Gähler

Fakultät für Mathematik, Universität Bielefeld

Postfach 10 01 31, D-33501 Bielefeld

gaehler@math.uni-bielefeld.de

May 2012

Contents

1 The Crystallographic Groups Catalog

The package CrystCat provides a catalog of crystallographic groups of dimensions 2, 3, and 4 which covers most of the data contained in the book "Crystallographic groups of four-dimensional space" [BBNWZ78]. It has been brought into GAP format by Volkmar Felsch.

The GAP 4 version of the catalog requires the package Cryst, which is loaded automatically. The benefit of this is that space groups extracted from the catalog now have the rich set of methods provided by Cryst at their disposal, and are no longer dumb lists of generators. Moreover, space groups are now fully supported in both the representation acting from the left and the representation acting from the right.

In 2001, Bernd Souvignier has discovered an error in the above mentioned book: On page 118, in the tabulation of enantiomorphic space-group types, it is wrongly claimed that the (affine) four-dimensional space-group type 08/01/01/002 splits into an enantiomorphic pair of (proper) space-group types. This is indicated by an asterisk preceding the space-group number. This asterisk has to be removed. As a consequence, the number of four-dimensional space-group types splitting into enantiomorphic pairs (given on page 11 and page 52 of the book) reduces from 112 to 111. An erratum has been submitted to *Acta Cryst.*.

The only implication of this correction for the package CrystCat is that the output of the function

```
DisplaySpaceGroupType( 4, 8, 1, 1, 2 );
```

had to be changed from

```
#I    *Space-group type (4,8,1,1,2); orbit size 2; fp-free
```

to

```
#I     Space-group type (4,8,1,1,2); orbit size 2; fp-free
```

This has been done in the release GAP 4.3.

1.1 How to access the data of the book

Among others, the catalog offers functions which provide access to the data listed in the Tables 1, 5, and 6 of [BBNWZ78]:

- The information on the crystal families listed in Table 1 can be reproduced using the `DisplayCrystalFamily` function.
- Similarly, the `DisplayCrystalSystem` function can be used to reproduce the information on the crystal systems provided in Table 1.
- The information given in the $\mathbb{Q}$-class headlines of Table 1 can be displayed by the `DisplayQClass` function, whereas the `FpGroupQClass` function can be used to reproduce the presentations that are listed in Table 1 for the $\mathbb{Q}$-class representatives.
- The information given in the $\mathbb{Z}$-class headlines of Table 1 will be covered by the results of the `DisplayZClass` function, and the matrix generators of the $\mathbb{Z}$-class representatives can be constructed by calling the `MatGroupZClass` function.

- The `DisplaySpaceGroupType` and the `DisplaySpaceGroupGenerators` functions can be used to reproduce all of the information on the space-group types that is provided in Table 1.
- The normalizers listed in Table 5 can be reproduced by calling the `NormalizerZClass` function.
- Finally, the `CharTableQClass` function will compute the character tables listed in Table 6, whereas the isomorphism types given in Table 6 may be obtained by calling the `DisplayQClass` function.

The display functions mentioned in the above list print their output with different indentation. So, calling them in a suitably nested loop, you may produce a listing in which the information about the objects of different type will be properly indented as has been done in Table 1 of [BBNWZ78].

1.2 Representation of space groups

Probably the most important function in the catalog is the `SpaceGroupBBNWZ` function which provides representatives of the affine classes of space groups. A space group of dimension n is represented by an $(n+1)$-dimensional rational matrix group as follows.

If S is an n-dimensional space group, then each element s in S is an affine mapping $s : V \to V$ of an n-dimensional $\mathbb{R}$-vector space V onto itself. Hence s can be written as the product of an appropriate invertible linear mapping $g : V \to V$ and a translation by some translation vector $t \in V$ such that, if we write mappings from the left, we have $s(v) = g(v)+t$ for all $v \in V$.

If we fix a basis of V and then replace each $v \in V$ by the column vector of its coefficients with respect to that basis (and hence V by the isomorphic column vector space $\mathbb{R}^{n\times 1}$), we can describe the linear mapping g involved in s by an $n \times n$ matrix $M_g \in GL_n(\mathbb{R})$ which acts by multiplication from the left on the column vectors in $\mathbb{R}^{n\times 1}$. Hence, if we identify V with $\mathbb{R}^{n\times 1}$, we have $s(v) = M_g v + t$ for all $v \in \mathbb{R}^{n\times 1}$.

Moreover, if we extend each column vector $v \in \mathbb{R}^{n\times 1}$ to a column $[[v],[1]]$ of length $n+1$ by adding an entry 1 in the last position and if we define an $(n+1)\times(n+1)$ matrix $M_s = [[M_g, t],[0,1]]$, we have $[[s(v)],[1]] = M_s[[v],[1]]$ for all $v \in \mathbb{R}^{n\times 1}$. This means that we can represent the space group S by the isomorphic group $M(S) = \{M_s | s \in S\}$. The submatrices M_g occurring in the elements of $M(S)$ form an $n \times n$ matrix group $P(S)$, the "point group" of $M(S)$. In fact, we can choose the basis of $\mathbb{R}^{n\times 1}$ such that $M_g \in GL_n(\mathbb{Z})$ and $t \in \mathbb{Q}^{n\times 1}$ for all $M_s \in M(S)$. In particular, the space group representatives that are normally used by the crystallographers are of this form, and the book [BBNWZ78] uses the same convention.

The representation described above is the one usually used by crystallographers. There is, however, an alternative to the representation of the space group elements by matrices of the form $[[M_g, t],[0,1]]$ as described above. Instead of considering the coefficient vectors as columns we may consider them as rows. Then we can associate to each affine mapping $s \in S$ an $(n+1)\times(n+1)$ matrix $M'_s = [[M'_{g'}, 0],[t',1]]$ with $M'_{g'} \in GL_n(\mathbb{R})$ and $t' \in \mathbb{R}^{1\times n}$ such that $[s(v'),1] = [v',1]M'_s$ for all $v' \in \mathbb{R}^{1\times n}$, and we may represent S by the matrix group $M'(S) = \{M'_s | s \in S\}$. Again, we can choose the basis of $\mathbb{R}^{1\times n}$ such that $M'_{g'} \in GL_n(\mathbb{Z})$ and $t' \in \mathbb{Q}^{1\times n}$ for all $M'_s \in M'(S)$.

From the mathematical point of view, both approaches are equivalent. In particular, $M(S)$ and $M'(S)$ are isomorphic, for instance via the isomorphism τ mapping $M_s \in M(S)$ to $(M_s^{tr})^{-1}$. Unfortunately, however, neither of the two is a good choice for our GAP catalog.

The first convention, using matrices which act on column vectors from the left, is not consistent with the fact that actions in GAP are usually from the right.

On the other hand, if we choose the second convention, we run into a problem with the names of the space groups as introduced in [BBNWZ78]. Any such name does not just describe the abstract isomorphism type of the respective space group S, but reflects properties of the matrix group $M(S)$. In particular, it contains as a leading part the name of the $\mathbb{Z}$-class of the associated point group $P(S)$. Since the classification of space groups by affine equivalence is tantamount to their classification by abstract isomorphism, $M'(S)$ lies in the same affine class as $M(S)$ and hence should get the same name as $M(S)$. But the point group $P(S)$ that occurs in that name is not always $\mathbb{Z}$-equivalent to the point group $P'(S)$ of $M'(S)$. For example, the isomorphism $\tau : M(S) \to M'(S)$ defined above maps the $\mathbb{Z}$-class representative with the parameters $[3,7,3,2]$ (in the notation described below) to the $\mathbb{Z}$-class representative with the

parameters $[3, 7, 3, 3]$. In other words: The space group names introduced for the groups $M(S)$ in [BBNWZ78] lead to confusing inconsistencies if assigned to the groups $M'(S)$.

In order to avoid this confusion we decided that the first convention is the lesser evil, and so the GAP catalog follows the book. In particular, all functions listed in section 1.1 use the convention of the book. The space groups, however, can be constructed in both representations, so that the user can choose the one that seems more appropriate in the particular situation. The function `SpaceGroupOnLeftBBNWZ` constructs a space group in the "crystallographic" representation acting on the left, whereas `SpaceGroupOnRightBBNWZ` constructs a space group in the representation acting on the right, as preferred by GAP. In order to avoid long function names (and in order to avoid mixing groups in different representations), one can set one's own default with the function `SetCrystGroupDefaultAction` (see 44.7.2), which takes as argument either `LeftAction` of `RightAction`. `SpaceGroupBBNWZ` then constructs a space group in this default representation. Initially, the default is `RightAction`.

The space groups constructed from the catalog are matrix groups, which in addition have the property `IsAffineCrystGroupOnLeft` (or `IsAffineCrystGroupOnRight`, respectively). The package Cryst provides methods to compute with such groups. Cryst is necessary for any serious computation with space groups, because the support of plain GAP for infinite matrix groups (such as space groups) is very limited.

Before we describe all available catalog functions in detail, we have to add two remarks.

Remark 1: The concepts used in this section are defined in chapter 1 (Basic definitions) of [BBNWZ78]. However, note that the definition of the concept of a crystal system given on page 16 of that book relies on the following statement about $\mathbb{Q}$-classes:

> For a $\mathbb{Q}$-class C there is a unique holohedry H such that each f.u. group in C is a subgroup of some f.u. group in H, but is not a subgroup of any f.u. group belonging to a holohedry of smaller order.

This statement is correct for dimensions 1, 2, 3, and 4, and hence the definition of "crystal system" given on page 16 of [BBNWZ78] is known to be unambiguous for these dimensions. However, there is a counterexample to this statement in seven-dimensional space so that the definition breaks down for some higher dimensions.

Therefore, the authors of the book have since proposed to replace this definition of "crystal system" by the following much simpler one, which has been discussed in more detail in [NPW81]. To formulate it, we use the intersections of $\mathbb{Q}$-classes and Bravais flocks as introduced on page 17 of [BBNWZ78], and we define the classification of the set of all $\mathbb{Z}$-classes into crystal systems as follows:

> **Definition**: A crystal system (introduced as an equivalence class of $\mathbb{Z}$-classes) consists of full geometric crystal classes. The $\mathbb{Z}$-classes of two (geometric) crystal classes belong to the same crystal system if and only if these geometric crystal classes intersect the same set of Bravais flocks of $\mathbb{Z}$-classes.

From this definition of a crystal system of $\mathbb{Z}$-classes one then obtains crystal systems of f.u. groups, of space-group types, and of space groups in the same manner as with the preceding definitions in the book.

The new definition is unambiguous for all dimensions. Moreover, it can be checked from the tables in the book that it defines the same classification as the old one for dimensions 1, 2, 3, and 4.

It should be noted that the concept of crystal family is well-defined independently of the dimension if one uses the "more natural" second definition of it at the end of page 17. Moreover, the first definition of crystal family on page 17 defines the same concept as the second one if the now proposed definition of crystal system is used.

Remark 2: The second remark just concerns a different terminology in the tables of [BBNWZ78] and in the current catalog. In group theory, the number of elements of a finite group usually is called the "order" of the group. This notation has been used throughout in the book. Here, however, we will follow the GAP conventions and use the term "size" instead.

1.3 Crystal Families

1 ► `NrCrystalFamilies( dim )`

returns the number of crystal families in case of dimension *dim*. It can be used to formulate loops over the crystal families.

There are 4, 6, and 23 crystal families of dimension 2, 3, and 4, respectively.

```
gap> n := NrCrystalFamilies( 4 );
23
```

2 ► `DisplayCrystalFamily( dim, family )`

displays for the specified crystal family essentially the same information as is provided for that family in Table 1 of [BBNWZ78], namely

- the family name,
- the number of parameters,
- the common rational decomposition pattern,
- the common real decomposition pattern,
- the number of crystal systems in the family, and
- the number of Bravais flocks in the family.

For details see [BBNWZ78].

```
gap> DisplayCrystalFamily( 4, 17 );
#I Family XVII: cubic orthogonal; 2 free parameters;
#I  Q-decomposition pattern 1+3; R-decomposition pattern 1+3;
#I  2 crystal systems; 6 Bravais flocks
gap> DisplayCrystalFamily( 4, 18 );
#I Family XVIII: octagonal; 2 free parameters;
#I  Q-irreducible; R-decomposition pattern 2+2;
#I  1 crystal system; 1 Bravais flock
gap> DisplayCrystalFamily( 4, 21 );
#I Family XXI: di-isohexagonal orthogonal; 1 free parameter;
#I  R-irreducible; 2 crystal systems; 2 Bravais flocks
```

1.4 Crystal Systems

1 ► `NrCrystalSystems( dim )`

returns the number of crystal systems in case of dimension *dim*. It can be used to formulate loops over the crystal systems.

There are 4, 7, and 33 crystal systems of dimension 2, 3, and 4, respectively.

```
gap> n := NrCrystalSystems( 2 );
4
```

The following two functions are functions of crystal systems.

Each crystal system is characterized by a pair (*dim*, *system*) where *dim* is the associated dimension, and *system* is the number of the crystal system.

2 ► `DisplayCrystalSystem( dim, system )`

displays for the specified crystal system essentially the same information as is provided for that system in Table 1 of [BBNWZ78], namely

- the number of $\mathbb{Q}$-classes in the crystal system and
- the identification number, i. e., the triple (*dim*, *system*, *q-class*) described below, of the $\mathbb{Q}$-class that is the holohedry of the crystal system.

For details see [BBNWZ78].

```
gap> for sys in [ 1 .. 4 ] do  DisplayCrystalSystem( 2, sys );  od;
#I  Crystal system 1: 2 Q-classes; holohedry (2,1,2)
#I  Crystal system 2: 2 Q-classes; holohedry (2,2,2)
#I  Crystal system 3: 2 Q-classes; holohedry (2,3,2)
#I  Crystal system 4: 4 Q-classes; holohedry (2,4,4)
```

1.5 Q-Classes

1 ► `NrQClassesCrystalSystem( dim, system )`

returns the number of $\mathbb{Q}$-classes within the given crystal system. It can be used to formulate loops over the $\mathbb{Q}$-classes.

The following five functions are functions of $\mathbb{Q}$-classes.

In general, the parameters characterizing a $\mathbb{Q}$-class will form a triple (*dim*, *system*, *q-class*) where *dim* is the associated dimension, *system* is the number of the associated crystal system, and *q-class* is the number of the $\mathbb{Q}$-class within the crystal system. However, in case of dimensions 2 or 3, a $\mathbb{Q}$-class may also be characterized by a pair (*dim*, *IT-number*) where *IT-number* is the number in the International Tables for Crystallography [Hah95] of any space-group type lying in (a $\mathbb{Z}$-class of) that $\mathbb{Q}$-class, or just by the Hermann-Mauguin symbol of any space-group type lying in (a $\mathbb{Z}$-class of) that $\mathbb{Q}$-class.

The Hermann-Mauguin symbols which we use in GAP are the short Hermann-Mauguin symbols defined in the 1983 edition of the International Tables [Hah95], but any occurring indices are expressed by ordinary integers, and bars are replaced by minus signs. For example, the Hermann-Mauguin symbol $P\bar{4}2_1m$ will be represented by the string `"P-421m"`.

2 ► `DisplayQClass( dim, system, q-class )`
► `DisplayQClass( dim, IT-number )`
► `DisplayQClass( Hermann-Mauguin-symbol )`

displays for the specified $\mathbb{Q}$-class essentially the same information as is provided for that $\mathbb{Q}$-class in Table 1 of [BBNWZ78] (except for the defining relations given there), namely

- the size of the groups in the $\mathbb{Q}$-class,
- the isomorphism type of the groups in the $\mathbb{Q}$-class,
- the Hurley pattern,
- the rational constituents,
- the number of $\mathbb{Z}$-classes in the $\mathbb{Q}$-class, and
- the number of space-group types in the $\mathbb{Q}$-class.

For details see [BBNWZ78].

```
gap> DisplayQClass( "p2" );
#I   Q-class H (2,1,2): size 2; isomorphism type 2.1 = C2;
#I    Q-constituents 2*(2,1,2); cc; 1 Z-class; 1 space group
gap> DisplayQClass( "R-3" );
#I   Q-class (3,5,2): size 6; isomorphism type 6.1 = C6;
#I    Q-constituents (3,1,2)+(3,4,3); ncc; 2 Z-classes; 2 space grps
gap> DisplayQClass( 3, 195 );
#I   Q-class (3,7,1): size 12; isomorphism type 12.5 = A4;
#I    C-irreducible; 3 Z-classes; 5 space grps
gap> DisplayQClass( 4, 27, 4 );
#I   Q-class H (4,27,4): size 20; isomorphism type 20.3 = D10xC2;
#I    Q-irreducible; 1 Z-class; 1 space group
gap> DisplayQClass( 4, 29, 1 );
#I  *Q-class (4,29,1): size 18; isomorphism type 18.3 = D6xC3;
#I    R-irreducible; 3 Z-classes; 5 space grps
```

Note in the preceding examples that, as pointed out above, the term "size" denotes the order of a representative group of the specified $\mathbb{Q}$-class and, of course, not the (infinite) class length.

3 ► `FpGroupQClass(` *dim*, *system*, *q-class* `)`
► `FpGroupQClass(` *dim*, *IT-number* `)`
► `FpGroupQClass(` *Hermann-Mauguin-symbol* `)`

returns a finitely presented group F, say, which is isomorphic to the groups in the specified $\mathbb{Q}$-class.

The presentation of that group is the same as the corresponding presentation given in Table 1 of [BBNWZ78] except for the fact that its generators are listed in reverse order. The reason for this change is that, whenever the group in question is solvable, the resulting generators form a pcgs (as defined in section 45 in the reference manual of GAP) if they are numbered "from the top to the bottom", and the presentation is a power-commutator presentation. The `PcGroupQClass` function described next will make use of this fact in order to construct a pc group isomorphic to F.

Note that, for any $\mathbb{Z}$-class in the specified $\mathbb{Q}$-class, the matrix group returned by the `MatGroupZClass` function (see below) not only is isomorphic to F, but also its generators satisfy the defining relators of F.

Besides of the usual components, F will have an attribute `CrystCatRecord`, which is a record with component `parameters`, which keeps a list of the parameters that specify the given $\mathbb{Q}$-class.

```
gap> F := FpGroupQClass( 4, 20, 3 );
FpGroupQClass( 4, 20, 3 )
gap> GeneratorsOfGroup( F );
[ f1, f2 ]
gap> RelatorsOfFpGroup( F );
[ f1^2*f2^-3, f2^6, f2^-1*f1^-1*f2*f1*f2^-4 ]
gap> Size( F );
12
gap> CrystCatRecord( F ).parameters;
[ 4, 20, 3 ]
```

4 ► `PcGroupQClass(` *dim*, *system*, *q-class* `)`
► `PcGroupQClass(` *dim*, *IT-number* `)`
► `PcGroupQClass(` *Hermann-Mauguin-symbol* `)`

returns a pc group P, say, isomorphic to the groups in the specified $\mathbb{Q}$-class, if these groups are solvable, or the value `fail` (together with an appropriate warning), otherwise.

P is constructed by first establishing a finitely presented group (as it would be returned by the `FpGroupQClass` function described above) and then constructing from it an isomorphic pc group. If the underlying pcgs is not a prime orders pcgs (see section 45), then it will be refined appropriately (and a warning will be displayed).

Besides of the usual components, *P* will have an attribute `CrystCatRecord`, which is a record with component `parameters`, which saves a list of the parameters that specify the given $\mathbb{Q}$-class.

```
gap> P := PcGroupQClass( 4, 31, 3 );
#I  Warning: a non-solvable group can't be represented as a pc group
fail
gap> P := PcGroupQClass( 4, 20, 3 );
#I  Warning: the presentation has been extended to get a prime order pcgs
PcGroupQClass( 4, 20, 3 )
gap> GeneratorsOfGroup( P );
[ f1, f2, f3 ]
gap> Size( P );
12
gap> CrystCatRecord( P ).parameters;
[ 4, 20, 3 ]
```

5 ► `CharTableQClass(` *dim*, *system*, *q-class* `)`
► `CharTableQClass(` *dim*, *IT-number* `)`
► `CharTableQClass(` *Hermann-Mauguin-symbol* `)`

returns the character table *T*, say, of a representative group of (a $\mathbb{Z}$-class of) the specified $\mathbb{Q}$-class.

Although the set of characters can be considered as an invariant of the specified $\mathbb{Q}$-class, the resulting table will depend on the order in which GAP sorts the conjugacy classes of elements and the irreducible characters and hence, in general, will not coincide with the corresponding table presented in [BBNWZ78].

`CharTableQClass` proceeds as follows. If the groups in the given $\mathbb{Q}$-class are solvable, then it first calls the `PcGroupQClass` and `RefinedPcGroup` functions to get a suitable isomorphic pc group, and then it calls the `CharacterTable` function to compute the character table of that pc group. In the case of the five $\mathbb{Q}$-classes of dimension 4 whose groups are not solvable, it first calls the `FpGroupQClass` function to get an isomorphic finitely presented group, then it constructs a specially chosen faithful permutation representation of low degree for that group, and finally it determines the character table of the resulting permutation group again by calling the `CharacterTable` function.

In general, the above strategy will be much more efficient than the alternative possibilities of calling the `CharacterTable` function for a finitely presented group provided by the `FpGroupQClass` function or for a matrix group provided by the `MatGroupZClass` function.

```
gap> T := CharTableQClass( 4, 20, 3 );;
gap> Display( T );
CharTableQClass( 4, 20, 3 )

     2  2  2  1  1  2  2
     3  1  .  1  1  .  1

       1a 4a 6a 3a 4b 2a
    2P 1a 2a 3a 3a 2a 1a
    3P 1a 4b 2a 1a 4a 2a
    5P 1a 4a 6a 3a 4b 2a
```

```
X.1     1  1  1  1  1  1
X.2     1 -1  1  1 -1  1
X.3     1  A -1  1 -A -1
X.4     1 -A -1  1  A -1
X.5     2  .  1 -1  . -2
X.6     2  . -1 -1  .  2

A = E(4)
  = ER(-1) = i
```

1.6 Z-Classes

1 ▶ `NrZClassesQClass(` *dim*, *system*, *q-class* `)`
▶ `NrZClassesQClass(` *dim*, *IT-number* `)`
▶ `NrZClassesQClass(` *Hermann-Mauguin-symbol* `)`

returns the number of $\mathbb{Z}$-classes within the given $\mathbb{Q}$-class. It can be used to formulate loops over the $\mathbb{Z}$-classes.

The following functions are functions of $\mathbb{Z}$-classes.

In general, the parameters characterizing a $\mathbb{Z}$-class will form a quadruple (*dim*, *system*, *q-class*, *z-class*) where *dim* is the associated dimension, *system* is the number of the associated crystal system, *q-class* is the number of the associated $\mathbb{Q}$-class within the crystal system, and *z-class* is the number of the $\mathbb{Z}$-class within the $\mathbb{Q}$-class. However, in case of dimensions 2 or 3, a $\mathbb{Z}$-class may also be characterized by a pair (*dim*, *IT-number*) where *IT-number* is the number in the International Tables [Hah95] of any space-group type lying in that $\mathbb{Z}$-class, or just by the Hermann-Mauguin symbol of any space-group type lying in that $\mathbb{Z}$-class.

2 ▶ `DisplayZClass(` *dim*, *system*, *q-class*, *z-class* `)`
▶ `DisplayZClass(` *dim*, *IT-number* `)`
▶ `DisplayZClass(` *Hermann-Mauguin-symbol* `)`

displays for the specified $\mathbb{Z}$-class essentially the same information as is provided for that $\mathbb{Z}$-class in Table 1 of [BBNWZ78] (except for the generating matrices of a class representative group given there), namely

- for dimensions 2 and 3, the Hermann-Mauguin symbol of a representative space-group type which belongs to that $\mathbb{Z}$-class,
- the Bravais type,
- some decomposability information,
- the number of space-group types belonging to the $\mathbb{Z}$-class,
- the size of the associated cohomology group.

For details see [BBNWZ78].

```
gap> DisplayZClass( 2, 3 );
#I    Z-class (2,2,1,1) = Z(pm): Bravais type II/I; fully Z-reducible;
#I     2 space groups; cohomology group size 2
gap> DisplayZClass( "F-43m" );
#I    Z-class (3,7,4,2) = Z(F-43m): Bravais type VI/II; Z-irreducible;
#I     2 space groups; cohomology group size 2
gap> DisplayZClass( 4, 2, 3, 2 );
#I    Z-class B (4,2,3,2): Bravais type II/II; Z-decomposable;
#I     2 space groups; cohomology group size 4
gap> DisplayZClass( 4, 21, 3, 1 );
#I   *Z-class (4,21,3,1): Bravais type XVI/I; Z-reducible;
#I     1 space group; cohomology group size 1
```

3 ► `MatGroupZClass(` *dim*, *system*, *q-class*, *z-class* `)`
► `MatGroupZClass(` *dim*, *IT-number* `)`
► `MatGroupZClass(` *Hermann-Mauguin-symbol* `)`

returns a *dim* × *dim* matrix group *M*, say, which is a representative of the specified $\mathbb{Z}$-class. Its generators satisfy the defining relators of the finitely presented group which may be computed by calling the `FpGroupQClass` function (see above) for the $\mathbb{Q}$-class which contains the given $\mathbb{Z}$-class.

The generators of *M* are the same matrices as those given in Table 1 of [BBNWZ78]. Note, however, that they will be listed in reverse order to keep them in parallel to the abstract generators provided by the `FpGroupQClass` function (see above).

Besides of the usual components, *M* will have an attribute `CrystCatRecord`, which is a record with two components. The first component is `parameters`, which saves a list of the parameters that specify the given $\mathbb{Z}$-class. The second component is `conjugator`, whose value is the identity element of *M*. Its purpose is to make the resulting record consistent with those returned by the `NormalizerZClass` or `ZClassRepsDadeGroup` functions described below.

```
gap> M := MatGroupZClass( 4, 20, 3, 1 );
MatGroupZClass( 4, 20, 3, 1 )
gap> for g in GeneratorsOfGroup( M ) do
>  Print( "\n" ); PrintArray( g ); od; Print( "\n" );

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

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

gap> Size( M );
12
gap> CrystCatRecord( M ).parameters;
[ 4, 20, 3, 1 ]
```

4 ► `NormalizerZClass(` *dim*, *system*, *q-class*, *z-class* `)`
► `NormalizerZClass(` *dim*, *IT-number* `)`
► `NormalizerZClass(` *Hermann-Mauguin-symbol* `)`

returns the normalizer *N*, say, in $GL(dim, \mathbb{Z})$ of the representative *dim* × *dim* matrix group which is constructed by the `MatGroupZClass` function (see above).

If the size of *N* is finite, then *N* again lies in some $\mathbb{Z}$-class. In this case, *N* will have an attribute `CrystCatRecord`, which is a record with two components, `parameters` and `conjugator`. These contain, respectively, the list of parameters of that $\mathbb{Z}$-class, and a matrix $g \in GL(dim, \mathbb{Z})$, such that $N = g^{-1}Rg$, where *R* is the representative group of that $\mathbb{Z}$-class.

```
gap> N := NormalizerZClass( 4, 20, 3, 1 );
NormalizerZClass( 4, 20, 3, 1 )
gap> for g in GeneratorsOfGroup( N ) do
>  Print( "\n" ); PrintArray( g ); od; Print( "\n" );
```

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

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

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

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

gap> Size( N );
96
gap> CrystCatRecord( N ).parameters;
[ 4, 20, 22, 1 ]
gap> CrystCatRecord( N ).conjugator = One( N );
true
gap> L := NormalizerZClass( 3, 42 );
NormalizerZClass( 3, 3, 2, 4 )
gap> Size( L );
16
gap> CrystCatRecord( L ).parameters;
[ 3, 4, 7, 2 ]
gap> CrystCatRecord( L ).conjugator;
[ [ 0, 0, -1 ], [ 1, 0, 0 ], [ 0, -1, -1 ] ]
gap> M := NormalizerZClass( "C2/m" );
<matrix group of size infinity with 5 generators>
gap> Size( M );
infinity
gap> HasCrystCatRecord( M );
false
```

1.7 Dade groups

Some of the $\mathbb{Z}$-classes of dimension d, say, are "maximal" in the sense that the groups in these classes are maximal finite subgroups of $GL(d,\mathbb{Z})$. Generalizing a term which is being used for dimension 4, we call the representatives of these maximal $\mathbb{Z}$-classes the **Dade groups** of dimension d.

1 ► `NrDadeGroups(` *dim* `)`

returns the number of Dade groups of dimension *dim*. It can be used to formulate loops over the Dade groups.

There are 2, 4, and 9 Dade groups of dimension 2, 3, and 4, respectively.

```
gap> NrDadeGroups( 4 );
9
```

2 ► `DadeGroup(` *dim*, *n* `)`

returns the *n*th Dade group of dimension *dim*.

```
gap> D := DadeGroup( 4, 7 );
MatGroupZClass( 4, 31, 7, 2 )
```

3 ► `DadeGroupNumbersZClass(` *dim*, *system*, *q-class*, *z-class* `)`
► `DadeGroupNumbersZClass(` *dim*, *IT-number* `)`
► `DadeGroupNumbersZClass(` *Hermann-Mauguin-symbol* `)`

returns the set of all those integers n_i for which the n_ith Dade group of dimension *dim* contains a subgroup which, in $GL(dim, \mathbb{Z})$, is conjugate to the representative group of the given $\mathbb{Z}$-class.

```
gap> dadeNums := DadeGroupNumbersZClass( 4, 4, 1, 2 );
[ 1, 5, 8 ]
gap> for d in dadeNums do
>     D := DadeGroup( 4, d );
>     Print( D, " of size ", Size( D ), "\n" );
> od;
MatGroupZClass( 4, 20, 22, 1 ) of size 96
MatGroupZClass( 4, 30, 13, 1 ) of size 288
MatGroupZClass( 4, 32, 21, 1 ) of size 384
```

4 ► `ZClassRepsDadeGroup(` *dim*, *system*, *q-class*, *z-class*, *n* `)`
► `ZClassRepsDadeGroup(` *dim*, *IT-number*, *n* `)`
► `ZClassRepsDadeGroup(` *Hermann-Mauguin-symbol*, *n* `)`

determines in the *n*th Dade group of dimension *dim* all those conjugacy classes whose groups are, in $GL(dim, \mathbb{Z})$, conjugate to the $\mathbb{Z}$-class representative group *R*, say, of the given $\mathbb{Z}$-class. It returns a list of representative groups of these conjugacy classes.

Let *M* be any group in the resulting list. *M* then has an attribute `CrystCatRecord`, which is a record with two components. The component `parameters` is the list of parameters of the $\mathbb{Z}$-class of *R*, and `conjugator` is a suitable matrix *g* from $GL(dim, \mathbb{Z})$, respectively, such that *M* equals $g^{-1}Rg$.

```
gap> DadeGroupNumbersZClass( 2, 2, 1, 2 );
[ 1, 2 ]
gap> ZClassRepsDadeGroup( 2, 2, 1, 2, 1 );
[ MatGroupZClass( 2, 2, 1, 2 )^[ [ 0, 1 ], [ -1, 0 ] ] ]
gap> ZClassRepsDadeGroup( 2, 2, 1, 2, 2 );
[ MatGroupZClass( 2, 2, 1, 2 )^[ [ 1, -1 ], [ 0, -1 ] ],
  MatGroupZClass( 2, 2, 1, 2 )^[ [ 1, 0 ], [ -1, 1 ] ] ]
gap> R := last[2];;
gap> CrystCatRecord( R ).parameters;
[ 2, 2, 1, 2 ]
gap> CrystCatRecord( R ).conjugator;
[ [ 1, 0 ], [ -1, 1 ] ]
```

1.8 Space groups and space group types

1 ▶ `NrSpaceGroupTypesZClass( dim, system, q-class, z-class )`
▶ `NrSpaceGroupTypesZClass( dim, IT-number )`
▶ `NrSpaceGroupTypesZClass( Hermann-Mauguin-symbol )`

returns the number of space-group types within the given $\mathbb{Z}$-class. It can be used to formulate loops over the space-group types.

```
gap> N := NrSpaceGroupTypesZClass( 4, 4, 1, 1 );
13
```

The following functions are functions of space-group types.

In general, the parameters characterizing a space-group type will form a quintuple (*dim*, *system*, *q-class*, *z-class*, *sg-type*) where *dim* is the associated dimension, *system* is the number of the associated crystal system, *q-class* is the number of the associated $\mathbb{Q}$-class within the crystal system, *z-class* is the number of the $\mathbb{Z}$-class within the $\mathbb{Q}$-class, and *sg-type* is the space-group type within the $\mathbb{Z}$-class. However, in case of dimensions 2 or 3, you may instead specify a $\mathbb{Z}$-class by a pair (*dim*, *IT-number*) or by its Hermann-Mauguin symbol (as described above). Then the function will handle the first space-group type within that $\mathbb{Z}$-class, i.e., *sg-type* = 1, that is, the corresponding symmorphic space group (split extension).

2 ▶ `DisplaySpaceGroupType( dim, system, q-class, z-class, sg-type )`
▶ `DisplaySpaceGroupType( dim, IT-number )`
▶ `DisplaySpaceGroupType( Hermann-Mauguin-symbol )`

displays for the specified space-group type some of the information which is provided for that space-group type in Table 1 of [BBNWZ78], namely

- the orbit size associated with that space-group type and,
- for dimensions 2 and 3, the *IT-number* and the Hermann-Mauguin symbol.

For details see [BBNWZ78].

```
gap> DisplaySpaceGroupType( 2, 17 );
#I     Space-group type (2,4,4,1,1); IT(17) = p6mm; orbit size 1
gap> DisplaySpaceGroupType( "Pm-3" );
#I     Space-group type (3,7,2,1,1); IT(200) = Pm-3; orbit size 1
gap> DisplaySpaceGroupType( 4, 32, 10, 2, 4 );
#I    *Space-group type (4,32,10,2,4); orbit size 18
gap> DisplaySpaceGroupType( 3, 6, 1, 1, 4 );
#I    *Space-group type (3,6,1,1,4); IT(169) = P61, IT(170) = P65;
#I      orbit size 2; fp-free
```

3 ▶ `DisplaySpaceGroupGenerators( dim, system, q-class, z-class, sg-type)`
▶ `DisplaySpaceGroupGenerators( dim, IT-number )`
▶ `DisplaySpaceGroupGenerators( Hermann-Mauguin-symbol )`

displays the non-translation generators of a representative space group of the specified space-group type without actually constructing that matrix group. The generators are given in the representation acting from the left on column vectors.

In more details: Let n = *dim* be the given dimension, and let $M_1, \ldots, M_r$ be the generators of the representative $n \times n$ matrix group of the given $\mathbb{Z}$-class (this is the group which you will get if you call the `MatGroupZClass` function (see above) for that $\mathbb{Z}$-class). Then, for the given space-group type, the `SpaceGroupOnLeftBBNWZ` function described below will construct as representative of that space-group type an $(n+1) \times (n+1)$ matrix group which is generated by the n translations which are induced by the (standard) basis vectors of the n-dimensional Euclidian space, and r

additional matrices $S_1, \ldots, S_r$ of the form $S_i = \begin{bmatrix} M_i & t_i \\ 0 & 1 \end{bmatrix}$, where the $n \times n$ submatrices M_i are as defined above, and the t_i are n-columns with rational entries. The `DisplaySpaceGroupGenerators` function saves time by not constructing the group, but just displaying the r matrices $S_1, \ldots, S_r$.

```
gap> DisplaySpaceGroupGenerators( "P61" );
#I  Non-translation generators of SpaceGroupOnLeftBBNWZ( 3, 6, 1, 1, 4 )

[ [   -1,    0,    0,    0 ],
  [    0,   -1,    0,    0 ],
  [    0,    0,    1,  1/2 ],
  [    0,    0,    0,    1 ] ]

[ [    0,   -1,    0,    0 ],
  [    1,   -1,    0,    0 ],
  [    0,    0,    1,  1/3 ],
  [    0,    0,    0,    1 ] ]
```

4 ► `SpaceGroupOnLeftBBNWZ(` *dim*, *system*, *q-class*, *z-class*, *sg-type* `)`
► `SpaceGroupOnLeftBBNWZ(` *dim*, *IT-number* `)`
► `SpaceGroupOnLeftBBNWZ(` *Hermann-Mauguin-symbol* `)`

returns a representative, S, of the space group type specified by the arguments. S is returned in the form of an `AffineCrystGroupOnLeft`, which acts from the left on column vectors (see also the description of the `DisplaySpaceGroupGenerators` function above). The package Cryst provides methods for the computation with space groups.

```
gap> S := SpaceGroupOnLeftBBNWZ( "P61" );
SpaceGroupOnLeftBBNWZ( 3, 6, 1, 1, 4 )
gap> for s in GeneratorsOfGroup( S ) do
>  Print( "\n" ); PrintArray( s ); od; Print( "\n" );

[ [   -1,    0,    0,    0 ],
  [    0,   -1,    0,    0 ],
  [    0,    0,    1,  1/2 ],
  [    0,    0,    0,    1 ] ]

[ [    0,   -1,    0,    0 ],
  [    1,   -1,    0,    0 ],
  [    0,    0,    1,  1/3 ],
  [    0,    0,    0,    1 ] ]

[ [  1,  0,  0,  1 ],
  [  0,  1,  0,  0 ],
  [  0,  0,  1,  0 ],
  [  0,  0,  0,  1 ] ]

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

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

gap> CrystCatRecord( S ).parameters;
[ 3, 6, 1, 1, 4 ]
```

The resulting group has an attribute `CrystCatRecord`, whose component `parameters` specifies the given space-group type.

- ▶ `SpaceGroupOnRightBBNWZ(` *dim*, *system*, *q-class*, *z-class*, *sg-type* `)`
- ▶ `SpaceGroupOnRightBBNWZ(` *dim*, *IT-number* `)`
- ▶ `SpaceGroupOnRightBBNWZ(` *Hermann-Mauguin-symbol* `)`
- ▶ `SpaceGroupOnRightBBNWZ(` *S* `)`

returns a representative, *T*, of the space group type specified by the arguments. *T* is returned in the form of an `AffineCrystGroupOnRight`, which acts from the right on row vectors. The generators of *T* are the transposed generators (in the same order) of the corresponding `SpaceGroupOnLeftBBNWZ`, *S*, specified by the same arguments. The space group *S* is also accepted as argument. The package Cryst provides methods for the computation with space groups.

```
gap> T := SpaceGroupOnRightBBNWZ( S );
SpaceGroupOnRightBBNWZ( 3, 6, 1, 1, 4 )
gap> for m in GeneratorsOfGroup( T ) do
>  Print( "\n" ); PrintArray( m ); od; Print( "\n" );

[ [   -1,    0,    0,    0 ],
  [    0,   -1,    0,    0 ],
  [    0,    0,    1,    0 ],
  [    0,    0,  1/2,    1 ] ]

[ [    0,    1,    0,    0 ],
  [   -1,   -1,    0,    0 ],
  [    0,    0,    1,    0 ],
  [    0,    0,  1/3,    1 ] ]

[ [  1,  0,  0,  0 ],
  [  0,  1,  0,  0 ],
  [  0,  0,  1,  0 ],
  [  1,  0,  0,  1 ] ]

[ [  1,  0,  0,  0 ],
  [  0,  1,  0,  0 ],
  [  0,  0,  1,  0 ],
  [  0,  1,  0,  1 ] ]

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

6 ► `SpaceGroupBBNWZ(` *dim*, *system*, *q-class*, *z-class*, *sg-type* `)`
► `SpaceGroupBBNWZ(` *dim*, *IT-number* `)`
► `SpaceGroupBBNWZ(` *Hermann-Mauguin-symbol* `)`

calls either `SpaceGroupOnLeftBBNWZ` or `SpaceGroupOnRightBBNWZ` with the same arguments, depending on the value of the variable `CrystGroupDefaultAction`.

7 ► `FpGroupSpaceGroupBBNWZ(` *S* `)`

returns a finitely presented group *G*, say, which is isomorphic to *S*, where *S* is expected to be a space group from the BBNWZ catalog (acting from the left or from the right). It is chosen such that there is an isomorphism from *G* to *S* which maps each generator of *G* onto the corresponding generator of *S*. This means, in particular, that the matrix generators of *S* satisfy the relators of *G*. If the factor group of *S* by its translation normal subgroup is solvable, then the presentation returned is a polycyclic power commutator presentation.

```
gap> G := FpGroupSpaceGroupBBNWZ( S );
FpGroupSpaceGroupOnLeftBBNWZ( 3, 6, 1, 1, 4 )
gap> for rel in RelatorsOfFpGroup( G ) do Print( rel, "\n" ); od;
g1^2*g5^-1
g2^3*g5^-1
g2^-1*g1^-1*g2*g1
g3^-1*g1^-1*g3*g1*g3^2
g3^-1*g2^-1*g3*g2*g4*g3^2
g4^-1*g1^-1*g4*g1*g4^2
g4^-1*g2^-1*g4*g2*g4*g3^-1
g4^-1*g3^-1*g4*g3
g5^-1*g1^-1*g5*g1
g5^-1*g2^-1*g5*g2
g5^-1*g3^-1*g5*g3
g5^-1*g4^-1*g5*g4
gap> # Verify that the matrix generators of S satisfy the relators of G.
gap> ForAll( RelatorsOfFpGroup( G ), rel -> One(S) =
>  MappedWord( rel, FreeGeneratorsOfFpGroup(G), GeneratorsOfGroup(S) ) );
true
```

Bibliography

[BBNWZ78] H. Brown, R. Bülow, J. Neubüser, H. Wondratschek, and H. Zassenhaus. *Crystallographic Groups of Four-Dimensional Space*. John Wiley, New York, 1978.

[Hah95] T. Hahn, editor. *International Tables for Crystallography, Volume A, Space-group Symmetry, 4th Edition*. Kluwer, Dordrecht, 1995.

[NPW81] J. Neubüser, W. Plesken, and H. Wondratschek. An emendatory discursion on defining crystal systems. *Match*, 10:77–96, 1981.

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".