

Aclib

A GAP4 Package

**Computations with
Almost Crystallographic Groups**

by

**Karel Dekimpe (KU Leuven Kulak)
and
Bettina Eick (Braunschweig)**

Contents

1	The Almost Crystallographic Groups Package	3
1.1	More about almost crystallographic groups	3
2	Algorithms for almost crystallographic groups	5
2.1	Properties of almost crystallographic groups	5
2.2	Betti numbers	5
2.3	Determination of certain extensions	6
3	The catalog of almost crystallographic groups	7
3.1	Rational matrix groups	7
3.2	Polycyclically presented groups	8
3.3	More about the type and the defining parameters	9
3.4	The electronic versus the printed library	10
4	Example computations with almost crystallographic groups	12
4.1	Example computations I	12
4.2	Example computations II	13
4.3	Example computations III	13
	Bibliography	16
	Index	17

1

The Almost Crystallographic Groups Package

A group is called **almost crystallographic** if it is a finitely generated nilpotent-by-finite group without non-trivial finite normal subgroups. An important special case of almost crystallographic groups are the **almost Bieberbach groups**: these are almost crystallographic and torsion free.

By its definition, an almost crystallographic group G has a finitely generated nilpotent normal subgroup N of finite index. Clearly, N is polycyclic and thus has a polycyclic series. The number of infinite cyclic factors in such a series for N is an invariant of G : the **Hirsch length** of G .

For each almost crystallographic group of Hirsch length 3 and 4 there exists a representation as a rational matrix group in dimension 4 or 5, respectively. These representations can be considered as affine representations of dimension 3 or 4. Via these representations, the almost crystallographic groups act (properly discontinuously) on $\mathbb{R}^3$ or $\mathbb{R}^4$. That is one reason to define the **dimension** of an almost crystallographic group as its Hirsch length.

The 3-dimensional and a part of the 4-dimensional almost crystallographic groups have been classified by K. Dekimpe in [Dek96]. This classification includes all almost Bieberbach groups in dimension 3 and 4. It is the first central aim of this package to give access to the resulting library of groups. The groups in this electronic catalog are available in two different representations: as rational matrix groups and as polycyclically presented groups. While the first representation is the more natural one, the latter description facilitates effective computations with the considered groups using the methods of the Polycyclic package.

The second aim of this package is to introduce a variety of algorithms for computations with polycyclically presented almost crystallographic groups. These algorithms supplement the methods available in the Polycyclic package and give access to some methods which are interesting specifically for almost crystallographic groups. In particular, we present methods to compute Betti numbers and to construct or check the existence of certain extensions of almost crystallographic groups. We note that these methods have been applied in [DE01b] and [DE01a] for computations with almost crystallographic groups.

Finally, we remark that almost crystallographic groups can be seen as natural generalizations of crystallographic groups. A library of crystallographic groups and algorithms to compute with crystallographic groups are available in the GAP packages `cryst`, `carat` and `crystcat`.

1.1 More about almost crystallographic groups

Almost crystallographic groups were first discussed in the theory of actions on Lie groups. We recall the original definition here briefly and we refer to [Aus60], [Dek96] and [Lee88] for more details.

Let L be a connected and simply connected nilpotent Lie group. For example, the 3-dimensional Heisenberg group, consisting of all upper unitriangular 3×3 -matrices with real entries is of this type. Then $L \rtimes \text{Aut}(L)$ acts affinely (on the left) on L via

$$\forall l, l' \in L, \forall \alpha \in \text{Aut}(L) : {}^{(l, \alpha)}l' = l \alpha(l').$$

Let C be a maximal compact subgroup of $\text{Aut}(L)$. Then a subgroup G of $L \rtimes C$ is said to be an almost crystallographic group if and only if the action of G on L , induced by the action of $L \rtimes \text{Aut}(L)$, is properly discontinuous and the quotient space $G \backslash L$ is compact. One recovers the situation of the ordinary crystallographic groups by taking $L = \mathbb{R}^n$, for some n , and $C = O(n)$, the orthogonal group.

More generally, we say that an abstract group is an almost crystallographic group if it can be realized as a genuine almost crystallographic subgroup of some $L \rtimes C$. In the following theorem we outline some algebraic characterizations of almost crystallographic groups; see Theorem 3.1.3 of [Dek96]. Recall that the **Fitting subgroup** $\text{Fitt}(G)$ of a polycyclic-by-finite group G is its unique maximal normal nilpotent subgroup.

Theorem. The following are equivalent for a polycyclic-by-finite group G :

- (1) G is an almost crystallographic group.
- (2) $\text{Fitt}(G)$ is torsion free and of finite index in G .
- (3) G contains a torsion free nilpotent normal subgroup N of finite index in G with $C_G(N)$ torsion free.
- (4) G has a nilpotent subgroup of finite index and there are no non-trivial finite normal subgroups in G .

In particular, if G is almost crystallographic, then $G/\text{Fitt}(G)$ is finite. This factor is called the **holonomy group** of G .

The dimension of an almost crystallographic group equals the dimension of the Lie group L above which coincides also with the Hirsch length of the polycyclic-by-finite group. This library therefore contains families of virtually nilpotent groups of Hirsch length 3 and 4.

2 Algorithms for almost crystallographic groups

This chapter presents a variety of algorithms for almost crystallographic groups. In most cases, they assume a polycyclically presented group as input; in particular, the input groups must be polycyclic in this case. The methods described here supplement the methods of the `Polycyclic` package for polycyclically presented groups. Many of the functions in this chapter are based on methods of the `Polycyclic` package and thus this package must be installed to use the functions introduced here. We refer to the `Polycyclic` package for further information on polycyclic presentations.

2.1 Properties of almost crystallographic groups

1 ► `IsAlmostCrystallographic( G )` P

This function checks if a polycyclically presented group G is almost crystallographic; that is, it checks if G is nilpotent-by-finite and has no non-trivial finite normal subgroup.

2 ► `IsAlmostBieberbachGroup( G )` P

This function checks if a polycyclically presented group G is almost Bieberbach; that is, it checks if G is nilpotent-by-finite and torsion free.

2.2 Betti numbers

Let G be a polycyclically presented and torsion free group of Hirsch length n . Then we can compute the Betti numbers $\beta_i(G)$ for $i \in \{0, 1, 2, n-2, n-1, n\}$. If $n \leq 6$, then we can compute all Betti numbers $\beta_i(G)$ for $0 \leq i \leq 6$ of G . We introduce the following functions for this purpose and we refer to [\[Bro82\]](#) for the details on the orientation module and the Betti numbers.

1 ► `OrientationModule( G )` F

This function determines the orientation module of the polycyclically presented group G ; that is, it returns a list of matrices $m_1, \dots, m_n \leq GL(1, \mathbb{Z})$ which are the images of the 'Igs(G)' in their action on the orientation module.

2 ► `BettiNumber( G, m )` F

This function returns the m th Betti number of the polycyclically presented torsion free group G if $m \in \{0, 1, 2, n-2, n-1, n\}$, where n is the Hirsch length of G .

3 ► `BettiNumbers( G )` A

This function returns the Betti number of the polycyclically presented torsion free group G if the Hirsch length of G is smaller than 7.

2.3 Determination of certain extensions

Let G be a polycyclically presented almost crystallographic group. We want to check the existence of certain extensions of G .

First, it is well-known that the equivalence classes of extensions of G correspond to the second cohomology group of G . This cohomology group can be computed using the methods of the **Polycyclic** package for any explicitly given module of G . Further, we can construct a polycyclic presentation for each cocycle of the second cohomology group. We give an example for such a computation below.

However, we may be interested in certain extensions only; for example, the torsion free extensions are often of particular interest. If the second cohomology group is finite, then we can compute a polycyclic presentation for each element of this group and check the resulting group for torsion freeness. But if the second cohomology group is infinite, then this approach is not available. Hence we introduce the following special method to cover this and related applications.

1 ► `HasExtensionOfType(  $G$ , torsionfree, minimalcentre )` F

Suppose that G is a polycyclically presented almost crystallographic group with Fitting subgroup N . This function checks if there is a G -module $M \cong \mathbb{Z}$ which is centralized by N such that there exists a torsion free extension of M by G (if the flag *torsionfree* is true) or an extension E with $Z(\text{Fitt}(E)) = M$ (if the flag *minimalcentre* is true) or an extension which satisfies both conditions (if both flags are true).

We note that the existence of such extensions is of interest in the determination of extensions which are almost Bieberbach groups. We refer to [DE01b] for a more detailed account of this application and for further results of a similar nature.

3 The catalog of almost crystallographic groups

This chapter introduces the access functions to the catalog of 3- and 4-dimensional crystallographic groups. This catalog is an electronic version of the classification obtained in [Dek96].

3.1 Rational matrix groups

The following three main functions are available to access the library of almost crystallographic groups as rational matrix groups.

- 1 ► `AlmostCrystallographicGroup( dim, type, parameters )`
- `AlmostCrystallographicDim3( type, parameters )`
- `AlmostCrystallographicDim4( type, parameters )`

dim is the dimension of the required group. Thus *dim* must be either 3 or 4. The inputs *type* and *parameters* are used to define the desired group as described in [Dek96]. We outline the possible choices for *type* and *parameters* here briefly. A more extended description is given later in Section 1.1 or can be obtained from [Dek96].

type specifies the type of the required group. There are 17 types in dimension 3 and 95 types in dimension 4. The input *type* can either be an integer defining the position of the desired type among all types; that is, in this case *type* is a number in [1..17] in dimension 3 or a number in [1..95] in dimension 4. Alternatively, *type* can be a string defining the desired type. In dimension 3 the possible strings are "01", "02", ..., "17". In dimension 4 the possible strings are listed in the list `ACDim4Types` and thus can be accessed from `GAP`.

parameters is a list of integers. Its length depends on the type of the chosen group. The lists `ACDim3Param` and `ACDim4Param` contain at position *i* the length of the parameter list for the type number *i*. Every list of integers of this length is a valid *parameter* input. Alternatively, one can input `false` instead of a parameter list. Then `GAP` will chose a random parameter list of suitable length.

```
gap> G := AlmostCrystallographicGroup( 4, 50, [ 1, -4, 1, 2 ] );
<matrix group of size infinity with 5 generators>
gap> DimensionOfMatrixGroup( G );
5
gap> FieldOfMatrixGroup( G );
Rationals
gap> GeneratorsOfGroup( G );
[ [ [ 1, 0, -1/2, 0, 0 ], [ 0, 1, 0, 0, 1 ], [ 0, 0, 1, 0, 0 ],
    [ 0, 0, 0, 1, 0 ], [ 0, 0, 0, 0, 1 ] ],
  [ [ 1, 1/2, 0, 0, 0 ], [ 0, 1, 0, 0, 0 ], [ 0, 0, 1, 0, 1 ],
    [ 0, 0, 0, 1, 0 ], [ 0, 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0, 0 ], [ 0, 1, 0, 0, 0 ], [ 0, 0, 1, 0, 0 ],
    [ 0, 0, 0, 1, 1 ], [ 0, 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0, 1 ], [ 0, 1, 0, 0, 0 ], [ 0, 0, 1, 0, 0 ],
```

```

      [ 0, 0, 0, 1, 0 ], [ 0, 0, 0, 0, 1 ] ],
    [ [ 1, -4, 1, 0, 1/2 ], [ 0, 0, -1, 0, 0 ], [ 0, 1, 0, 0, 0 ],
      [ 0, 0, 0, 1, 1/4 ], [ 0, 0, 0, 0, 1 ] ] ]
gap> G.1;
[ [ 1, 0, -1/2, 0, 0 ], [ 0, 1, 0, 0, 1 ], [ 0, 0, 1, 0, 0 ],
  [ 0, 0, 0, 1, 0 ], [ 0, 0, 0, 0, 1 ] ]
gap> ACDim4Types[50];
"076"
gap> ACDim4Param[50];
4

```

3.2 Polycyclically presented groups

All the almost crystallographic groups considered in this package are polycyclic. Hence they have a polycyclic presentation and this can be used to facilitate efficient computations with the groups. To obtain the polycyclic presentation of an almost crystallographic group we supply the following functions. Note that the share package **Polycyclic** must be installed to use these functions.

- 1 ► `AlmostCrystallographicPcpGroup( dim, type, parameters )`
- `AlmostCrystallographicPcpDim3( type, parameters )`
- `AlmostCrystallographicPcpDim4( type, parameters )`

The input is the same as for the corresponding matrix group functions. The output is a pcp group isomorphic to the corresponding matrix group. An explicit isomorphism from an almost crystallographic matrix group to the corresponding pcp group can be obtained by the following function.

- 2 ► `IsomorphismPcpGroup( G )`

We can use the polycyclic presentations of almost crystallographic groups to exhibit structure information on these groups. For example, we can determine their Fitting subgroup and ask group-theoretic questions about this nilpotent group. The factor $G/\text{Fit}(G)$ of an almost crystallographic group G is called **holonomy group**. We provide access to this factor of a pcp group via the following functions. Let G be an almost crystallographic pcp group.

- 3 ► `HolonomyGroup( G )`
- `NaturalHomomorphismOnHolonomyGroup( G )`

The following example shows applications of these functions.

```

gap> G := AlmostCrystallographicPcpGroup( 4, 50, [ 1, -4, 1, 2 ] );
Pcp-group with orders [ 4, 0, 0, 0, 0 ]
gap> Cgs(G);
[ g1, g2, g3, g4, g5 ]

gap> F := FittingSubgroup( G );
Pcp-group with orders [ 0, 0, 0, 0 ]
gap> Centre(F);
Pcp-group with orders [ 0, 0 ]
gap> LowerCentralSeries(F);
[ Pcp-group with orders [ 0, 0, 0, 0 ], Pcp-group with orders [ 0 ],
  Pcp-group with orders [ ] ]
gap> UpperCentralSeries(F);
[ Pcp-group with orders [ 0, 0, 0, 0 ], Pcp-group with orders [ 0, 0 ],
  Pcp-group with orders [ ] ]
gap> MinimalGeneratingSet(F);
[ g2, g3, g4 ]

```

```

gap> H := HolonomyGroup( G );
Pcp-group with orders [ 4 ]
gap> hom := NaturalHomomorphismOnHolonomyGroup( G );
[ g1, g2, g3, g4, g5 ] -> [ g1, identity, identity, identity, identity ]
gap> U := Subgroup( H, [Pcp(H)[1]^2] );
Pcp-group with orders [ 2 ]
gap> PreImage( hom, U );
Pcp-group with orders [ 2, 0, 0, 0, 0 ]

```

3.3 More about the type and the defining parameters

Each group from this library knows that it is almost crystallographic and, additionally, it knows its type and defining parameters.

1 ► `AlmostCrystallographicInfo( G )`

This attribute is set for groups from the library only. It is not possible at current to determine the type and the defining parameters for an arbitrary almost crystallographic groups which is not defined by the library access functions.

```

gap> G := AlmostCrystallographicGroup( 4, 70, false );
<matrix group of size infinity with 5 generators>
gap> IsAlmostCrystallographic(G);
true
gap> AlmostCrystallographicInfo(G);
rec( dim := 4, type := 70, param := [ 1, -4, 1, 2, -3 ] )

gap> G := AlmostCrystallographicPcpGroup( 4, 70, false );
Pcp-group with orders [ 6, 0, 0, 0, 0 ]
gap> IsAlmostCrystallographic(G);
true
gap> AlmostCrystallographicInfo(G);
rec( dim := 4, type := 70, param := [ -3, 2, 5, 1, 0 ] )

```

We consider the types of almost crystallographic groups in more detail. The almost crystallographic groups in dimensions 3 and 4 fall into three families

- (1) 3-dimensional almost crystallographic groups.
- (2) 4-dimensional almost crystallographic groups with a Fitting subgroup of class 2.
- (3) 4-dimensional almost crystallographic groups with a Fitting subgroup of class 3.

These families are split up further into subfamilies in [Dek96] and to each subfamily is assigned a type; that is, a string which is used to identify the subfamily. As mentioned above, for the 3-dimensional almost crystallographic groups the type is a string representing the numbers from 1 to 17, i.e. the available types are "01", "02", ..., "17".

For the 4-dimensional almost crystallographic groups with a Fitting subgroup of class 2 the type is a string of 3 or 4 characters. In general, a string of 3 characters representing the number of the table entry in [Dek96] is used. So possible types are "001", "002", The reader is warned however that not all possible numbers are used, e.g. there are no groups of type "016". Also, the types do not appear in their natural order in [Dek96]. Moreover, for certain numbers there is more than one family of groups listed in [Dek96]. For example, the 3 families of groups corresponding to number 19 on pages 179-180 of [Dek96] have types "019", "019b" and "019c" (the order is the one given in [Dek96]).

For the last category of groups, the 4-dimensional almost crystallographic groups with a Fitting subgroup of class 3, the type is a string of 2 or 3 characters, where the first character is always the letter "B". This "B" is followed by the number of the table entry as found in [Dek96], eventually followed by a "b" or "c" as in the previous case.

For each type of almost crystallographic group contained in the library there exists a function taking a parameter list as input and returning the desired matrix or pcg group. These functions can be accessed from **GAP** using the lists `ACDim3Funcs`, `ACDim4Funcs`, `ACPcpDim3Funcs` and `ACPcpDim4Funcs` which consist of the corresponding functions.

Although we include these direct access functions here for completeness, we note that the user should in general use the higher-level functions introduced above to obtain almost crystallographic groups from the library. In particular, these low-level access functions return matrix or pcg groups, but the almost crystallographic info flags will not be attached to them.

```
gap> ACDim3Funcs[15];
function( k1, k2, k3, k4 ) ... end
gap> ACDim3Funcs[15](1,1,1,1);
<matrix group with 5 generators>
gap> ACPcpDim3Funcs[1](1);
Pcp-group with orders [ 0, 0, 0 ]
```

3.4 The electronic versus the printed library

The package `aclib` can be considered as the electronic version of Chapter 7 of [Dek96]. In this section we outline the relationship between the library presented in this manual and the printed version in [Dek96]. First we consider an example. At page 175 of [Dek96], we find the following groups in the table starting with entry “13”.

13. $Q = P2/c$

$$E : \langle a, b, c, d, \alpha, \beta \mid \begin{array}{ll} [b, a] = 1 & [d, a] = 1 \\ [c, a] = d^{2k_1} & [d, b] = 1 \\ [c, b] = 1 & [d, c] = 1 \\ \alpha a = a^{-1} \alpha d^{k_2} & \alpha^2 = d^{k_3} \\ \alpha b = b \alpha & \alpha d = d \alpha \\ \alpha c = c^{-1} \alpha d^{-2k_6} & \\ \beta a = a^{-1} \beta d^{k_1+k_2} & \beta^2 = d^{k_5} \\ \beta b = b^{-1} \beta d^{k_4} & \beta d = d \beta \\ \beta c = c^{-1} \beta d^{-2k_6} & \alpha \beta = c \beta \alpha d^{k_6} \end{array} \rangle$$

$$\lambda(\alpha) = \begin{pmatrix} 1 & \frac{k_1}{2} + k_2 & 0 & -2k_6 & \frac{k_3}{2} + \frac{k_6}{2} \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \lambda(\beta) = \begin{pmatrix} 1 & k_1 + k_2 & k_4 & -2k_6 & \frac{k_5}{2} \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H^2(Q, \mathbb{Z}) = \mathbb{Z} \oplus (\mathbb{Z}_2)^4 = \mathbb{Z}^6 / A,$$

$$A = \{(k_1, \dots, k_6) \mid k_1 = 0, k_2, \dots, k_5 \in 2\mathbb{Z}, k_6 \in \mathbb{Z}\}$$

AB-groups:

$$\forall k > 0, k \equiv 0 \pmod{2}, (k, 0, 1, 0, 1, 0)$$

The number “13” at the beginning of this entry is the type of the almost crystallographic group in this library. This family of groups with type 13 depends on 6 parameters $k_1, k_2, \dots, k_6$ and these are the *parameters* list in this library. The rational matrix representation in **GAP** corresponds exactly to the printed version in [Dek96] where it is named λ . In the example below, we consider the group with parameters $(k_1, k_2, k_3, k_4, k_5, k_6) = (8, 0, 1, 0, 1, 0)$.

```

gap> G:=AlmostCrystallographicDim4("013",[8,0,1,0,1,0]);
<matrix group with 6 generators>
gap> G.5;
[ [ 1, 4, 0, 0, 1/2 ], [ 0, -1, 0, 0, 0 ], [ 0, 0, 1, 0, 0 ],
  [ 0, 0, 0, -1, 1/2 ], [ 0, 0, 0, 0, 1 ] ]
gap> G.6;
[ [ 1, 8, 0, 0, 1/2 ], [ 0, -1, 0, 0, 0 ], [ 0, 0, -1, 0, 0 ],
  [ 0, 0, 0, -1, 0 ], [ 0, 0, 0, 0, 1 ] ]

```

For a 4-dimensional almost crystallographic group the matrix group is built up such that $\{a, b, c, d, \alpha, \beta, \gamma\}$ as described in [Dek96] forms the defining generating set of G . For certain types the elements α , β or γ may not be present. Similarly, for a 3-dimensional group we have the generating set $\{a, b, c, \alpha, \beta\}$ and α and β may be absent.

To obtain a polycyclic generating sequence from the defining generators of the matrix group we have to order the elements in the generating set suitably. For this purpose we take the subsequence of $(\gamma, \beta, \alpha, a, b, c, d)$ of those generators which are present in the defining generating set of the matrix group. This new ordering of the generators is then used to define a polycyclic presentation of the given almost crystallographic group.

4

Example computations with almost crystallographic groups

4.1 Example computations I

Using the functions available for pcg groups in the share package `polycyclic` it is now easy to redo some of the calculations of [Dek96]. As a first example we check whether the groups indicated as torsion free in [Dek96] are also determined as torsion free ones by GAP. In [Dek96] these almost Bieberbach groups are listed as “AB-groups”. So for type “013” these are the groups with parameters $(k, 0, 1, 0, 1, 0)$ where k is an even integer. Let’s look at some examples in GAP:

```
gap> G:=AlmostCrystallographicPcgDim4("013",[8,0,1,0,1,0]);
Pcg-group with orders [ 2, 2, 0, 0, 0, 0 ]
gap> IsTorsionFree(G);
true
gap> G:=AlmostCrystallographicPcgDim4("013",[9,0,1,0,1,0]);
Pcg-group with orders [ 2, 2, 0, 0, 0, 0 ]
gap> IsTorsionFree(G);
false
```

Further, there is also some cohomology information in the tables of [Dek96]. In fact, the groups in this library were obtained as extensions E of the form

$$1 \rightarrow \mathbb{Z} \rightarrow E \rightarrow Q \rightarrow 1$$

where, in the 4-dimensional case $Q = E/\langle d \rangle$. The cohomology information for the particular example above shows that the groups determined by a parameter set $(k_1, k_2, k_3, k_4, k_5, k_6)$ are equivalent as extensions to the groups determined by the parameters $(k_1, k_2 \bmod 2, k_3 \bmod 2, k_4 \bmod 2, k_5 \bmod 2, 0)$. This is also visible in finding torsion:

```

gap> G:=AlmostCrystallographicPcpDim4("013",[10,0,2,0,1,0]);
Pcp-group with orders [ 2, 2, 0, 0, 0, 0 ]
gap> IsTorsionFree(G);
false
gap> G:=AlmostCrystallographicPcpDim4("013",[10,0,3,0,1,9]);
Pcp-group with orders [ 2, 2, 0, 0, 0, 0 ]
gap> IsTorsionFree(G);
true

```

4.2 Example computations II

The computation of cohomology groups played an important role in the classification of the almost Bieberbach groups in [Dek96]. Using GAP, it is now possible to check these computations. As an example we consider the 4-dimensional almost crystallographic groups of type 85 on page 202 of [Dek96]. This group E has 6 generators. In the table, one also finds the information

$$H^2(Q, \mathbb{Z}) = \mathbb{Z} \oplus (\mathbb{Z}_2)^2 \oplus \mathbb{Z}_4$$

for $Q = E/\langle d \rangle$ as above. Moreover, the Q -module $\mathbb{Z}$ is in fact the group $\langle d \rangle$, where the Q -action comes from conjugation inside E . In the case of groups of type 85, $\mathbb{Z}$ is a trivial Q -module. The following example demonstrates how to (re)compute this two-cohomology group $H^2(Q, \mathbb{Z})$.

```

gap> G:=AlmostCrystallographicPcpGroup(4, "085", false);
Pcp group with orders [ 2, 4, 0, 0, 0, 0 ]
gap> GroupGeneratedByd:=Subgroup(G, [G.6] );
Pcp group with orders [ 0 ]
gap> Q:=G/GroupGeneratedByd;
Pcp group with orders [ 2, 4, 0, 0, 0 ]
gap> action:=List( Pcp(Q), x -> [[1]] );
[[ [ 1 ] ], [ [ 1 ] ], [ [ 1 ] ], [ [ 1 ] ], [ [ 1 ] ] ]
gap> C:=CRRecordByMats( Q, action);;
gap> TwoCohomologyCR( C ).factor.rels;
[ 2, 2, 4, 0 ]

```

This last line gives us the abelian invariants of the second cohomology group $H^2(Q, \mathbb{Z})$. So we should read this line as

$$H^2(Q, \mathbb{Z}) = \mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}$$

which indeed coincides with the information in [Dek96].

4.3 Example computations III

As another application of the capabilities of the combination of `aclib` and `polycyclic` we check some computations of [DM01].

Section 5 of the paper [DM01] is completely devoted to an example of the computation of the P -localization of a virtually nilpotent group, where P is a set of primes. Although it is not our intention to develop the theory of P -localization of groups at this place, let us summarize some of the main results concerning this topic here.

For a set of primes P , we say that $n \in P$ if and only if n is a product of primes in P . A group G is said to be P -local if and only if the map $\mu_n : G \rightarrow G : g \mapsto g^n$ is bijective for all $n \in P'$, where P' is the set of all primes not in P . The P -localization of a group G , is a P -local group G_P together with a morphism $\alpha : G \rightarrow G_P$ which satisfy the

following universal property: For each P -local group L and any morphism $\varphi : G \rightarrow L$, there exists a unique morphism $\psi : G_P \rightarrow L$, such that $\psi \circ \alpha = \varphi$.

This concept of localization is well developed for finite groups and for nilpotent groups. For a finite group G , the P -localization is the largest quotient of G , having no elements with an order belonging to P' (the morphism α , mentioned above is the natural projection).

In [DM01] a contribution is made towards the localization of virtually nilpotent groups. The theory developed in the paper is then illustrated in the last section of the paper by means of the computation of the P -localization of an almost crystallographic group. For their example the authors have chosen an almost crystallographic group G of dimension 3 and type 17. For the set of parameters (k_1, k_2, k_3, k_4) they have considered all cases of the form $(k_1, k_2, k_3, k_4) = (2, 0, 0, k_4)$.

Here we will check their computations in two cases $k_4 = 0$ and $k_4 = 1$ using the set of primes $P = \{2\}$. The holonomy group of these almost crystallographic groups G is the dihedral group $\mathcal{D}_6$ of order 12. Thus there is a short exact sequence of the form

$$1 \rightarrow \text{Fitt}(G) \rightarrow G \rightarrow \mathcal{D}_6 \rightarrow 1.$$

As a first step in their computation, Descheemaeker and Malfait determine the group $I_{P'}\mathcal{D}_6$, which is the unique subgroup of order 3 in $\mathcal{D}_6$. One of the main objects in [DM01] is the group $K = p^{-1}(I_{P'}\mathcal{D}_6)$, where p is the natural projection of G onto its holonomy group. It is known that the P -localization of G coincides with the P -localization of $G/\gamma_3(K)$, where $\gamma_3(K)$ is the third term in the lower central series of K . As $G/\gamma_3(K)$ is finite in this example, we exactly know what this P -localization is. Let us now show, how GAP can be used to compute this P -localization in two cases:

First case: The parameters are $(k_1, k_2, k_3, k_4) = (2, 0, 0, 0)$

```
gap> G := AlmostCrystallographicPcpGroup(3, 17, [2,0,0,0] );
Pcp group with orders [ 2, 6, 0, 0, 0 ]
gap> projection := NaturalHomomorphismOnHolonomyGroup( G );
[ g1, g2, g3, g4, g5 ] -> [ g1, g2, identity, identity, identity ]
gap> F := HolonomyGroup( G );
Pcp group with orders [ 2, 6 ]
gap> IPprimeD6 := Subgroup( F, [F.2^2] );
Pcp group with orders [ 3 ]
gap> K := PreImage( projection, IPprimeD6 );
Pcp group with orders [ 3, 0, 0, 0 ]
gap> PrintPcpPresentation( K );
pcp presentation on generators [ g2^2, g3, g4, g5 ]
g2^2 ^ 3 = identity
g3 ^ g2^2 = g3^-1*g4^-1
g3 ^ g2^2^-1 = g4*g5^-2
g4 ^ g2^2 = g3*g5^2
g4 ^ g2^2^-1 = g3^-1*g4^-1*g5^2
g4 ^ g3 = g4*g5^2
g4 ^ g3^-1 = g4*g5^-2
gap> Gamma3K := CommutatorSubgroup( K, CommutatorSubgroup( K, K ) );
Pcp group with orders [ 0, 0, 0 ]
gap> quotient := G/Gamma3K;
Pcp group with orders [ 2, 6, 3, 3, 2 ]
gap> S := SylowSubgroup( quotient, 3 );
Pcp group with orders [ 3, 3, 3 ]
gap> N := NormalClosure( quotient, S );
Pcp group with orders [ 3, 3, 3 ]
gap> localization := quotient/N;
```

```

Pcp group with orders [ 2, 2, 2 ]
gap> PrintPcpPresentation( localization );
pcp presentation on generators [ g1, g2, g3 ]
g1 ^ 2 = identity
g2 ^ 2 = identity
g3 ^ 2 = identity

```

This shows that $G_P \cong \mathbb{Z}_2^3$.

Second case: The parameters are $(k_1, k_2, k_3, k_4) = (2, 0, 0, 1)$

```

gap> G := AlmostCrystallographicPcpGroup(3, 17, [2,0,0,1]);;
gap> projection := NaturalHomomorphismOnHolonomyGroup( G );;
gap> F := HolonomyGroup( G );;
gap> IPprimeD6 := Subgroup( F , [F.2^2] );;
gap> K := PreImage( projection, IPprimeD6 );;
gap> Gamma3K := CommutatorSubgroup( K, CommutatorSubgroup( K, K ));;
gap> quotient := G/Gamma3K;;
gap> S := SylowSubgroup( quotient, 3);;
gap> N := NormalClosure( quotient, S);;
gap> localization := quotient/N;
Pcp group with orders [ 2, 2, 2 ]
gap> PrintPcpPresentation( localization );
pcp presentation on generators [ g1, g2, g3 ]
g1 ^ 2 = identity
g2 ^ 2 = g3
g3 ^ 2 = identity
g2 ^ g1 = g2*g3
g2 ^ g1^-1 = g2*g3

```

In this case, we see that $G_P = \mathcal{D}_4$.

The reader can check that these results coincide with those obtained in [DM01]. Note also that we used a somewhat different scheme to compute this localization than the one used in [DM01]. We invite the reader to check the same computations, tracing exactly the steps made in [DM01].

Bibliography

- [Aus60] Louis Auslander. Bieberbach's theorem on space groups and discrete uniform subgroups of lie groups. *Ann. of Math*, 71(3):579–590, 1960.
- [Bro82] Kenneth S. Brown. *Cohomology of Groups*, volume 87 of *Grad. Texts in Math*. Springer-Verlag, New York, 1982.
- [DE01a] Karel Dekimpe and Bettina Eick. Computational aspects of group extensions and their application in topology. *Submitted*, 2001.
- [DE01b] Karel Dekimpe and Bettina Eick. Computations with almost crystallographic groups. *Submitted*, 2001.
- [Dek96] Karel Dekimpe. *Almost-Bieberbach Groups: Affine and Polynomial Structures*, volume 1639 of *Lecture Notes in Mathematics*. Springer-Verlag, Heidelberg, 1996.
- [DM01] An Descheemaeker and Wim Malfait. P -localization of relative groups. *Journal of Pure and Applied Algebra*, 159(1), 2001.
- [Lee88] Kyung Bai Lee. There are only finitely many infra-nilmanifolds under each nilmanifold. *Quart. J. Math. Oxford*, 39(2):61–66, 1988.

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”.

A

AlmostCrystallographicDim3, 7
AlmostCrystallographicDim4, 7
AlmostCrystallographicGroup, 7
AlmostCrystallographicInfo, 9
AlmostCrystallographicPcpDim3, 8
AlmostCrystallographicPcpDim4, 8
AlmostCrystallographicPcpGroup, 8

B

BettiNumber, 5
Betti numbers, 5
BettiNumbers, 5

D

Determination of certain extensions, 6

E

Example computations I, 12
Example computations II, 13
Example computations III, 13

H

HasExtensionOfType, 6

HolonomyGroup, 8

I

IsAlmostBieberbachGroup, 5
IsAlmostCrystallographic, 5
IsomorphismPcpGroup, 8

M

More about almost crystallographic groups, 3
More about the type and the defining parameters, 9

N

NaturalHomomorphismOnHolonomyGroup, 8

O

OrientationModule, 5

P

Polycyclically presented groups, 8
Properties of almost crystallographic groups, 5

R

Rational matrix groups, 7

T

The electronic versus the printed library, 10