1.3. Computer checking of the subgroup data

By Franz Gähler

1.3.1. Introduction

Most of the data in Part 2 of this volume have been checked by a computer-algebra program. This program is built upon the package *Cryst* (Eick *et al.*, 2001), formerly known as *CrystGAP* (Eick *et al.*, 1997), which in turn is an extension of the computer-algebra program *GAP* (The *GAP* Group, 2002). *GAP* is a program system for computations involving general algebraic structures, in particular groups. Since space groups are infinite groups, the generic algorithms for finite groups often cannot be used for space groups. The *Cryst* package provides the special methods and algorithms needed for working with space groups, thereby extending the field of applicability of *GAP*. The algorithms used in *Cryst* have been described by Eick *et al.* (1997).

1.3.2. Basic capabilities of the *Cryst* package

Before we describe in more detail the checks that have been performed, we briefly summarize the capabilities of the *Cryst* package and how space groups are handled by *Cryst*.

In *Cryst*, a space group is represented as a fixed group of augmented matrices $\mathbb{W}_i$, *cf.* Section 1.2.2.4. The group multiplication therefore coincides with matrix multiplication. The space group is defined by an arbitrary (finite) set of generating elements. The space group consists of the set of all augmented matrices that can be obtained as products of the generating matrices. It is not important which generating elements are chosen. Different sets of generators can define the same space group.

The representation of a space group as a group of augmented matrices implicitly requires the choice of an origin and a vector basis of Euclidean space. The *Cryst* package does not require any particular choice of origin or vector basis. In particular, it is not necessary to work with a lattice basis; in fact, any basis can be used. This has the advantage that one can continue to use the basis chosen for the parent group when passing to a subgroup.

Space groups are conjugate by some affine coordinate transformation if they differ only in their settings, *i.e.* by the choice of their origins and/or vector bases. Such an affine coordinate transformation is also represented by an augmented matrix. Given a coordinate transformation $\mathbb{V}$ and a space group $\mathcal{G}$, the transformed space group $\mathcal{G}'$ is simply generated by the transformed generators of $\mathcal{G}$.

If a space group $\mathcal{G}$ is given by a set of generators, *Cryst* first needs to compute the point group $\mathcal{P}$ of $\mathcal{G}$, and the translation subgroup $\mathcal{T}$, represented by a (canonical) basis of the translation lattice (the basis of the translation lattice chosen by *Cryst* is canonical in the sense that the same basis is always chosen for a given $\mathcal{T}$). The point group $\mathcal{P}$ is simply generated by the linear parts of the generators of $\mathcal{G}$. The mapping H, which sends each element of $\mathcal{G}$ to its linear part, is in fact a group homomorphism $\mathsf{H}\colon \mathcal{G}\longrightarrow\mathcal{P}$. The point group $\mathcal{P}$ is finite and each point-group element p can easily be expressed as a product of generators of $\mathcal{P}$, which are images of generators of $\mathcal{G}$ under H. This provides a way to compute, for any $p\in\mathcal{P}$, some representative pre-image g of p under H, *i.e.* some element $g\in\mathcal{G}$ which is mapped onto p by H. Having expressed the element p as a product of generators, one simply replaces the factors in this product by their representative pre-images under H, which are known by construction, to obtain the representative pre-image g of p.

To compute a generating set of the translation subgroup $\mathcal{T}\leq\mathcal{G}$, one first computes a set of defining relations of the point group $\mathcal{P}$ for the given generators. For a finite group this is a standard task in *GAP*. These defining relations are a set of inequivalent ways to express the identity element of $\mathcal{P}$ by the generators of $\mathcal{P}$. Replacing the factors in such a defining relation by their representative pre-images in $\mathcal{G}$ yields a pre-image of the identity of $\mathcal{P}$, *i.e.* an element of $\mathcal{T}$. The translations so obtained generate, together with the pure translation generators of $\mathcal{G}$, the entire translation subgroup $\mathcal{T}\leq\mathcal{G}$.

We now have all the necessary information to test whether a given augmented matrix $\mathbb{A}$ is an element of a space group $\mathcal{G}$. One first tests whether the linear part $\boldsymbol{M}$ of $\mathbb{A}$ is an element of the point group $\mathcal{P}$ of $\mathcal{G}$. If $\boldsymbol{M}\notin\mathcal{P}$, $\mathbb{A}$ is not an element of $\mathcal{G}$. Otherwise, some pre-image $\mathbb{G}$ of $\boldsymbol{M}$ in $\mathcal{G}$ is computed. $\mathbb{A}$ is then an element of $\mathcal{G}$ if and only if $\mathbb{A}$ and $\mathbb{G}$ differ in their translation part by an element of the translation group $\mathcal{T}$. With this membership test, we can determine whether two space groups are equal, or whether one is a subgroup of the other: a space group $\mathcal{H}$ is a subgroup of a space group $\mathcal{G}$ if all the generators of $\mathcal{H}$ are elements of $\mathcal{G}$. Two space groups are equal if either of them is a subgroup of index 1 of the other.

1.3.3. Computing maximal subgroups

The *Cryst* package has built-in facilities for computing the maximal subgroups of a given index for any space group $\mathcal{G}$. More precisely, given a prime number p, *Cryst* can compute conjugacy-class representatives of those maximal subgroups of $\mathcal{G}$ whose index in $\mathcal{G}$ is a power of p. The algorithms used for this task are described in Eick *et al.* (1997). Essentially, one determines the maximal subgroups of the (finite) factor group $\mathcal{G}/\mathcal{T}_p$, where $\mathcal{T}_p$ is the subgroup of those translations of $\mathcal{G}$ which are a p-fold multiple of an element of the full translation group $\mathcal{T}$ of $\mathcal{G}$. After the maximal subgroups are obtained, the translations in $\mathcal{T}_p$ are added back to the subgroups.

From a representative $\mathcal{H}$ of a conjugacy class of subgroups, the list of all subgroups in the same conjugacy class is obtained by repeatedly conjugating $\mathcal{H}$ (and the subgroups obtained from it by conjugation) with generators of $\mathcal{G}$, until no new conjugate subgroups are obtained. This is also the way of determining whether two subgroups are conjugate: one enumerates the groups in the conjugacy class of one of them, and checks whether the other is among them.

The index of a subgroup $\mathcal{H}$ of $\mathcal{G}$ is easily computed as the product of the index of the point group of $\mathcal{H}$ in the point group of $\mathcal{G}$ and the index of the translation group of $\mathcal{H}$ in the translation group of $\mathcal{G}$. For maximal subgroups, only one of these factors is different from 1. For *klassengleiche* subgroups, the two point groups are equal, whereas for *translationengleiche* subgroups the two translation subgroups are equal. Therefore, a maximal

subgroup is easily identified either as a *klassengleiche* or a *translationengleiche* subgroup.

1.3.4. Description of the checks

In order to avoid new errors being introduced in the typesetting process after the data have been checked and corrected, we carried out the computer checks directly on the LaTeX 2_ε sources used for the production of this volume. The tables have been typeset with specially designed LaTeX 2_ε macros, the primary purpose of which was to guarantee a homogeneous layout throughout the book. As a side effect, it was relatively easy to write a *GAP* program which parses the LaTeX 2_ε sources, recognizes the macros, extracts the data (the arguments of the macros), brings the data into a form suitable for further processing with routines from the *Cryst* package and finally performs the various checks. In this way, the checks could be done fully automatically, and could be repeated after every modification of the tabulated data.

In the following, we describe the checks that have been carried out. We first applied a number of tests individually to each of the tabulated maximal subgroups of low index: whether it is a subgroup, whether the index given is correct, whether the listed coordinate transformation maps the subgroup to the preferred setting of the given space-group type (and thus, whether the space-group type of the subgroup is correct) and whether the listed coordinate transformation maps the given generators of the subgroup to the standard generators of its space-group type, in the same order. Here, the preferred setting is the setting of the parent group, where applicable, and otherwise the preferred setting of the space-group type of the subgroup, if there is more than one setting in the tables.

In a second step, a complete set of maximal subgroups of low index (2, 3 or 4) is computed afresh with the routines from the *Cryst* package. These subgroups are then divided into conjugacy classes, and classified as *klassengleiche* or *translationengleiche* subgroups. This list is then compared with the tabulated list of maximal subgroups. It was verified that each maximal subgroup of a given index was listed exactly once, that the classification into conjugacy classes of subgroups was correct and that the subgroups were correctly identified as *klassengleiche* or as *translationengleiche* subgroups.

All the tests described above concern the maximal subgroups of low index. Unfortunately, similar automatic tests could not be performed on the series of isomorphic subgroups. The subgroups in these series contain variable parameters, and *Cryst* can only deal with fixed, concrete space groups without free parameters.

References

Eick, B., Gähler, F. & Nickel, W. (1997). *Computing maximal subgroups and Wyckoff positions of space groups. Acta Cryst.* A**53**, 467–474.

Eick, B., Gähler, F. & Nickel, W. (2001). *The 'Cryst' Package.* Version 4.1. http://www.itap.physik.uni-stuttgart.de/~gaehler/gap/packages.html.

The *GAP* Group (2002). *GAP – Groups, Algorithms and Programming*. Version 4.3. http://www.gap-system.org.