INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17" x 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9000683

Crystallographic space groups and algorithms

Cook, Michael, Ph.D.

City University of New York, 1989

Copyright ©1989 by Cook, Michael. All rights reserved.

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

4

Crystallographic Space Groups and Algorithms

by

Michael Cook

A dissertation submitted to the Graduate Faculty in Mathematics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1989

©1989
MICHAEL COOK
All Rights Reserved

This manuscript has been read and accepted for the Graduate Faculty in Mathematics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Date	Chair of Examining Committee
Apr 27, 95	Executive Officer

Supervisory Committee

The City University of New York

Abstract

Crystallographic Space Groups and Algorithms

by

Michael Cook

Adviser: Professor Louis Auslander

A mathematical derivation of the 230 crystallographic space groups is presented, based on their solvability. This approach only works in three dimensions, but the deeper insight provided is useful in structuring the computation of crystallographic Fourier transforms.

Next, attention is focussed on primitive space groups from the triclinic, monoclinic, and orthorhombic classes. Algorithms are given for passing from the space group to a program which computes the Fourier transform on data exhibiting the symmetry of that group. In addition, the program

1. Uses only non-redundant data, that is, data on an asymmetric unit.

2. Reduces to one-dimensional symmetrized transforms on that set.

Thus the computation of the Fourier transform is completely reduced to an asymmetric unit, while retaining the use of one-dimensional FFT's.

An abstract data type, "biased" fundamental domains, is developed for representing convenient asymmetric units for primitive triclinic, monoclinic, and orthorhombic space groups. In terms of this data type two basic algorithms are developed:

1. Computing asymmetric units.

2. Mapping between asymmetric units.

The process of going from a given space group to a program has been embodied in a program generator for this class of groups (written in Fortran), which is thoroughly discussed, and included, together with sample output.

PREFACE

The motivation of this work is the following goal: to create a universal crystallographic Fourier transform package which has 2 main features:

1. For each of the 230 space groups, it can perform a Fourier transform on data which observes the symmetry of that space group, as well as the inverse Fourier transform.

2. It can do this and still retain the $NlogN$ behavior of the Cooley-Tukey type family of algorithms.

This work presents a contribution to a part of this program, namely: for certain classes of space groups, it is shown how, using the group theory, to algorithmically proceed from the description of the space group to the creation of Fourier transforms that use only data from a non-redundant set of the input; and which on this set are as fast as the fastest existing algorithms. (In other words, any future improvement in the efficiency of one-dimensional Fourier transform algorithms is automatically included in this package.)

These algorithms have been embodied in a program generator, which takes as input a space group in the given class (monoclinic, orthorhombic, or triclinic), and produces as output Fortran code which computes the Fourier transform of data on an $N \times N \times N$ array (N a multiple of 8); but only uses data on a fundamental domain for the action of the space group on this array.

This material, and the program generator itself (a Fortran program), as well as sample output, is contained in Chapters 4 and 5.

In summary, the program generator is largely based on the creation of an abstract data type, the "biased" fundamental domain. It is biased towards performing symmetrized fourier transforms along a particular dimension. The exact definitions of asymmetric unit boundaries are calculated and maintained. Precise bookkeeping of isotropy properties is also implemented.

What the program generator does is:

1. Compute biased fundamental domains
2. Generate appropriate calls to one-dimensional symmetrized fourier transform routines
3. Compute the group action which maps from one biased fundamental domain to another.
4. Keep book on the phase relations that are generated as the data is fourier transformed.
5. Generate the symmetry expand routines for input and output fundamental domains

Testing

Consider the diagram:

$$\begin{array}{ccc} FD_1 & \stackrel{expand1}{\longrightarrow} & DATA \\ \downarrow symft & & \downarrow F.T. \\ FD_2 & \stackrel{expand2}{\longrightarrow} & OUTPUT \end{array}$$

where:

expand1 is a routine which symmetry expands the data on an asymmetric unit for the input.
F.T. is a conventional 3 dimensional fourier transform
symft is a symmetrized fourier transform created by the program generator.
expand2 is a routine which symmetry expands the data on an output asymmetric unit. (This also includes multiplying by appropriate phase factors.)

The diagram should commute, for all data on FD_1, but going counterclockwise from the upper left should be faster than going the other way.

Since the finite fourier transform is a linear computation, the programs were only tested on a basis for the complex vector space of all possible inputs. The basis chosen was:

$$FD_1(x,y,z) = \delta_{ijk}(x,y,z)$$

where

$$\delta_{ijk}(x,y,z) = 0 \quad if\ (i,j,k) \neq (x,y,z)$$

$$\delta_{ijk}(x,y,z) = 1 \quad if\ (i,j,k) = (x,y,z)$$

and (x,y,z) ranges over the indices present in the chosen asymmetric unit.

With this scheme, all 40 groups were tested for every possible input, in the cases $N \times N \times N = 8 \times 8 \times 8$, and $N \times N \times N = 16 \times 16 \times 16$.

In addition, $32 \times 32 \times 32$, and $64 \times 64 \times 64$ cases were partially tested. (Run for several days on a MicroVax II).

Timing

In the small cases (8 and 16) the results were what would be expected - the symmetrized routines ran faster than the conventional routines, by a factor of about the order of the group.

As N increases the data movement becomes more significant and weighs down the computation. The symmetrized routines still are faster, however. For instance, the symmetrized routines for a a group of order 8 ran almost 4 times faster than the conventional routine. However, these issues were not addressed in the creation of these programs, and there are some obvious improvements to be made along these lines. In addition, some attention to system tuning (paging parameters, allocation of more memory) should also improve performance.

Equipment

All this work was done on a MicroVax II, under the MicroVMS operating system, using standard VAX Fortran.

Space groups

The first 3 chapters are devoted to a mathematical classification of the 230 space groups. Although it is well known that there are 230 space groups, it is

important from the point of view of algorithms to understand them mathematically, structurally, and not just to have the list. Work remains to be done in this area- namely, understanding the Bravais lattices mathematically; and completing the formulation of the extensions in the language of cohomology. It would be nice if the cohomology theory ultimately structured some of the programs.

Let $R(3)$ be the set of rigid motions of R^3, Euclidean 3-space. We may represent $R(3)$ as a set of matrices:

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix}$$

where $A \in O(3,R)$, the orthogonal group, and $X \in R^3$. If $T =$ the set of pure translations of R^3, then $R(3)$ is a semi-direct product:

$$R(3) \cong T \rtimes O(3,R)$$

Definition 1 *A crystallographic group is a subgroup $\Gamma \subset R(3)$ such that $\Gamma \cap T$ is isomorphic to the lattice Z^3, and $\Gamma/(\Gamma \cap T)$ is a finite group.*

Thus we have the exact sequence:

$$1 \longrightarrow Z^3 \longrightarrow \Gamma \longrightarrow G \longrightarrow 1$$

where $G \subset O(3,R)$, and the projection from Γ to G takes $\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix}$ to A.

OBSERVATION: Let I be the 3×3 identity matrix, let $N \in Z^3$. Represent a lattice element as

$$\begin{pmatrix} I & N \\ 0 & 1 \end{pmatrix}$$

Then conjugation by an element of Γ can be written:

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I & N \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A^{-1} & -A^{-1}X \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I & AN \\ 0 & 1 \end{pmatrix}$$

From this we see that the lattice is maximal abelian - any element that centralizes it must be a pure translation, since $AN = N$ for all $N \Longrightarrow A = I$.

This is equivalent to the statement that G, the quotient, acts as a group of automorphisms of Z^3; that is G may be thought of as a subgroup of $GL(3, Z)$, with an appropriate choice of basis.

Definition 2 *2 crystallographic groups, Γ and Γ', are equivalent if, as subsets of the above form, they are related by a change of basis which carries the lattice for Γ onto the lattice for Γ'.*

Since

$$\begin{pmatrix} A & Y \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Gamma & X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A^{-1} & -A^{-1}Y \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A\Gamma A^{-1} & -(\Gamma - I)Y + AX \\ 0 & 1 \end{pmatrix}$$

we have:

$$\begin{pmatrix} A & Y \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I & X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A^{-1} & -A^{-1}Y \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I & AX \\ 0 & 1 \end{pmatrix}$$

and AX must be a lattice point for Γ' whenever X is a lattice point for Γ. It follows that A and A^{-1} are integral matrices, and that Γ and Γ' are integrally equivalent.

The above discussion structures the approach to classifying the crystallographic space groups: first we classify the finite subgroups of $GL(3, Z)$ up to integer equivalence; this is done (after some introductory material in Chapter 1) in Chapter 2. What this gives us is a list of possible G's in

$$1 \longrightarrow Z^3 \longrightarrow \Gamma \longrightarrow G \longrightarrow 1.$$

Next, for each G, we need to compute the possible extensions Γ. This amounts to enumerating equivalence classes of liftings (pre-images in Γ) of G.

Chapter 3 is the computation of most of these extensions. The extensions of all the primitive point groups have been computed. The extensions of the

non-primitive groups can be computed based on the computations for the primitive, using the fact that primitive groups are rationally conjugate to their non-primitive forms. This method is shown, and used for a few of the cases. There remain about 40 such extensions left to compute.

A Note on Notation

We have seen that in

$$1 \longrightarrow Z^3 \longrightarrow \Gamma \longrightarrow G \longrightarrow 1$$

we may take G as a finite subgroup of $GL(3, Z)$. We are often interested in pre-images of G in Γ. If $A \in G$ and

$$\gamma = \begin{pmatrix} l(A) & X \\ 0 & 1 \end{pmatrix}$$

is a pre-image, then $l(A)$ must in fact be the matrix A, since conjugation of the lattice by γ is given by A. Also, if

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} A & X' \\ 0 & 1 \end{pmatrix}$$

are two pre-images of the element A, then they differ by a lattice point:

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A^{-1} & -A^{-1}X' \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I & (-X' + X) \\ 0 & 1 \end{pmatrix}$$

and this element must be in the kernel of the projection $\Gamma \mapsto G$.

Therefore, we write a typical element on a pre-image of A as $\begin{pmatrix} A & t_A \\ 0 & 1 \end{pmatrix}$, where t_A is a choice function differing from any other choice function by an element of the lattice.

Contents

Chapter 1

Background Material

1.1 Introduction.

We assemble some basic group theoretic results, with sketch proofs for the simple ones, and references for the others.

Theorem 1 *Let G be a group, A an abelian normal subgroup of G, and let $Aut(A)$ be the group of automorphisms of A. Then there is a homomorphism of groups*

$$\phi : G/A \longrightarrow Aut(A)$$

where $\phi(gA)$ is the homomorphism defined by the formula:

$$\phi(gA)(a) = gag^{-1}, \quad a \in A$$

Proof: That ϕ is well-defined depends on the commutativity of A:

$$\begin{aligned} gA = hA &\implies g^{-1}h \in A && (1.1)\\ &\implies (g^{-1}h)a(h^{-1}g) = a && (1.2)\\ &\implies hah^{-1} = gag^{-1}. && (1.3) \end{aligned}$$

That ϕ is a homomorphism follows from the fact that conjugation by the element gh is the same as first conjugating by h, then by g:

$$\begin{aligned} \phi(gA \cdot hA)(a) &= \phi(ghA)(a) && (1.4)\\ &= (gh)a(gh)^{-1} && (1.5)\\ &= ghah^{-1}g^{-1} && (1.6)\\ &= g\phi(hA)(a)g^{-1} && (1.7)\\ &= \phi(gA) \circ \phi(hA)(a), \text{ for all } a \in A. && (1.8) \end{aligned}$$

•

Definition 3 *A is <u>maximal abelian in G</u> if A is an abelian subgroup of G, and if whenever B is an abelian subgroup of G containing A, then $A = B$.*

Theorem 2 *With the hypotheses of theorem 1, we have:*

$$\phi \text{ is } 1-1 \Longleftrightarrow A \text{ is maximal abelian in } G.$$

Proof: First, suppose A is maximal abelian. If gA is in the kernel of ϕ then, for all $a \in A$

$$\phi(gA)(a) = a \Longleftrightarrow \tag{1.9}$$

$$gag^{-1} = a \Longleftrightarrow \tag{1.10}$$

$$ga = ag. \tag{1.11}$$

Thus g commutes with every element of A. But then $gp(g, A)$ is an abelian subgroup of G, and since A is maximal abelian, $g \in A$.

Conversely, if A is not maximal abelian in G, there is a $g \in G - A$ which commutes with A; in which case the kernel of ϕ contains $gA \neq A$, and so ϕ is not one to one. •

Definition 4 *If G is a group, let $[G, G]$ be the subgroup of G generated by elements which are products of the form $aba^{-1}b^{-1}$, where a and b are in G. $[G, G]$ is called the commutator subgroup of G. It is obvious that G is abelian if and only if the commutator subgroup of G is the identity element. If we let $G_1 = [G, G]$, we may form the higher commutator subgroups of G inductively as follows:*

$$G_2 = [G_1, G_1],\ G_3 = [G_2, G_2], ..., G_{i+1} = [G_i, G_i]$$

The sequence of subgroups

$$G = G_0 \geq G_1 \geq G_2 \geq ... \geq G_i \geq ...$$

is called the derived series for G.

Definition 5 *G is a solvable group if for some i, $[G_i, G_i] = 1$.*

Theorem 3 *(Burnside) Let G have $p^\alpha q^\beta$ elements, where p and q are prime numbers. Then G is a solvable group.*

Proof: see [3].

Theorem 4 *(Sylow) If p divides the order of a finite group G, then there is a subgroup of G of order p^n, where p^n is the highest power of p dividing the order of G.*

Proof: see [8]

Theorem 5 *If G is a solvable group, then G has an abelian normal subgroup.*

Proof: Let $G = G_0 \geq G_1 \geq ... \geq G_n \geq 1$ be the derived series for G. G_n must be abelian, since $[G_n, G_n] = 1$. To show that is also normal in G, we show that any homomorphism $\phi : G \longrightarrow G$ must map each G_i into itself. In particular, conjuagation by any $g \in G$ will map G_n into itself, that is, G_n is normal in G. The proof is by induction. Certainly $\phi(G_0) \subseteq G_0$. Now suppose $\phi(G_{i-1}) \subseteq G_{i-1}$, and let g be a generator of G_i. Then $\phi(g) = \phi(a)\phi(b)\phi(a)^{-1}\phi(b)^{-1}$, which is a commutator of elements in G_{i-1}, hence is in G_i. •

Definition 6 *If G is a group, $C \in G$, the centralizer of C in G is the subgroup of G consisting of all elements of G that commute with C.*

Theorem 6 *If A is an abelian normal subgroup of G, then the centralizer of A is also normal.*

Proof: Let B be the centralizer of A in G. So $B = \{b \in G | ba = ab \forall a \in A\}$. To see that B is normal, we must conjugate an element of B by an arbitrary $g \in G$, and check that the result still commutes with A.

The computation is:

$$\begin{aligned}
(g^{-1}bg)a(g^{-1}b^{-1}g) &= (g^{-1}b)(gag^{-1})(b^{-1}) \\
&= (g^{-1}b)a'(b^{-1}g) \ where \ a' = gag^{-1} \\
&= g^{-1}(ba'b^{-1})g \\
&= g^{-1}a'g
\end{aligned}$$

$$= g^{-1}(gag^{-1})g$$
$$= a.$$

•

1.2 Semi-direct Products and Split Extensions

Definition 7 *A group G is an extension of K by Q if K is a normal subgroup of G, and $G/K \cong Q$.*

If G is an extension of K by Q, then G fits into the short exact sequence of groups:

$$1 \longrightarrow K \xrightarrow{i} G \xrightarrow{\pi} Q \longrightarrow 1 \tag{1.12}$$

Here i is an injection, π is a surjection, and the image of K in G is the kernel of π.

Definition 8 *The sequence (1.1) splits if there is a homomorphism $s : Q \longrightarrow G$ such that $\pi \circ s = 1_Q$. Such an s is called a splitting, and G is called a split extension.*

Theorem 7 *If there is a splitting s, and if H is the image of s in G, then:*

1) $G = HK$

2) $H \cap K = 1$

Proof: To prove 1), let $g \in G$. If $g \in K$ or $g \in H$, there is nothing to prove. So suppose $g \notin K$ and $g \notin H$. Let $\pi(g) = q$, and $s(q) = h \in H$. Then

$$\pi(h) = \pi(s(q)) = (\pi \circ s)(q) = q = \pi(g).$$

Thus $\pi(h^{-1}) = 1$, and so $h^{-1}g \in K$. Therefore $h^{-1}g = k$ for some $k \in K$, and so $g = hk$, with $h \in H$ and $k \in K$.

To prove 2), suppose $g \in H \cap K$. We have:

$$g \in K \implies \pi(g) = 1 \tag{1.13}$$

$$g \in H \implies g = s(q), \text{ for some } q \in Q. \tag{1.14}$$

Now $\pi \circ s = 1_Q \implies \pi \circ s$ is one to one. But $\pi(s(q)) = \pi(g) = 1$, and so $g = 1$. •

Theorem 8 *The converse is also true. That is, if a group G has two subgroups H and K, with K normal in G, and*

1) $G = HK$

2) $H \cap K = 1$

Then there is a splitting s,

$$1 \longrightarrow K \longrightarrow G \underset{\longleftarrow}{\overset{s}{\longrightarrow}} G/K \longrightarrow 1$$

and $s : G/K \longrightarrow H$ isomorphically.

Proof: We define an isomorphism $\phi : H \longrightarrow G/K$. Then s will be the inverse of ϕ. Let $\phi(h) = hK$. ϕ is the restriction of the natural projection map, and so is a homomorphism. We must show that ϕ is one to one and onto.

One to one: $\phi(h) = K \Longrightarrow h \in K \Longrightarrow h \in H \cap K \Longrightarrow h = 1$.

Onto: Let $gK \in G/K$; but $g = hk$, with $h \in H$, $k \in K$. Thus

$gK = hK$, and $\phi(h) = gK$. •

Definition 9 *A group G that satisfies the hypotheses of the previouis theorem is called the semi-direct product of H and K.*

If we have a split extension

$$1 \longrightarrow K \longrightarrow G \underset{\longleftarrow}{\overset{s}{\longrightarrow}} Q \longrightarrow 1$$

we may define a homomorphism $\phi : Q \longrightarrow Aut(K)$ by letting $\phi(q) = c_q|K$, where $c_q(k) = s(q)ks(q)^{-1}$. We now show the converse, namely:

Theorem 9 *If Q and K are groups, and $\psi : Q \longrightarrow Aut(K)$ is a homomorphism, then there is a group based on ψ,*

$$G = K \rtimes_{\psi} Q$$

such that

$$1 \longrightarrow K \longrightarrow G \longrightarrow Q \longrightarrow 1$$

is a split extension.

Proof: For ease of exposition we look at the case where ψ is an injection. So we may as well take Q to be a subgroup of $Aut(K)$. Then we construct a group G as follows. As a set, let $G = K \times Q$. To define the multiplication, we write K additively. For $k \in K$, $\phi \in Q$, define $(k, \phi)(k', \phi') = (k + \phi(k'), \phi\phi')$. Then it is easy to check that

1) this is an associative operation

2) $(0, 1_K)$ is the identity element.

3) $(k, \phi)(\phi^{-1}(-k), \phi^{-1}) = (0, 1_K)$, so every element is invertible.

Thus G is a group. It is also easy to see that:

4) $K \cong \{(k, 1) | k \in K\} \lhd K \rtimes_\psi Q$

5) $Q \cong \{(0, \phi) | \phi \in Q\}$

6) $G/K \cong Q$ (this gives the splitting). •

We close this section with a theorem which will be of major importance in the classification of point groups.

Theorem 10 *If F is a finite subgroup of $SL(3, Z)$, and A is an abelian normal subgroup which is also maximal abelian in $SL(3, Z)$, and if, furthermore,*

$$1 \longrightarrow A \longrightarrow F \underset{s}{\longleftarrow} Aut(A) \longrightarrow 1$$

is a split extension, then any other extension of A in $SL(3, Z)$ also splits.

Proof: In fact we show that any extension of A in $SL(3, Z)$ is a subgroup of F, F_1. Thus, F_1/A maps onto a subgroup of $Aut(A)$, and the restriction of s to this subgroup is the required splitting.

So suppose

$$1 \longrightarrow A \longrightarrow F \underset{s}{\longleftarrow} Aut(A) \longrightarrow 1 \text{ is split, and}$$

$$1 \longrightarrow K \longrightarrow G \longrightarrow Q \longrightarrow 1$$

is another extension of A in $SL(3, Z)$. Let $f_1 \in F_1 - A$. Conjugation by f_1 is an automorphism of A, hence realized, via s, by some $f \in F$. But then

$$f_1 a f_1^{-1} = f a f^{-1} \text{ for all } a \in A \tag{1.15}$$

$$\Longrightarrow a f_1^{-1} f = f_1^{-1} f a. \tag{1.16}$$

Thus a and $f_1^{-1}f$ commute, and since A is maximal abelian, $f_1^{-1}f \in A$. So $f_1^{-1}f = a$, for some $a \in A$, that is, $f_1^{-1} = af^{-1} \in F$. So $f_1 \in F$. •

1.3 Lemmas on the Orthogonal Group

We record some elementary facts about the real orthogonal groups in two and three dimensions.

Definition 10 *An $n \times n$ matrix A with real entries is orthogonal if the rows of A form an orthonormal set.*

It is straightforward to verify that:

1) A is orthogonal $\Longrightarrow AA^T = I \Longrightarrow A^T = A^{-1}$.

2) $det(A) = det(A^T) = \pm 1$.

Let $O(3, R)$ = the set of 3x3 orthogonal matrices. Then $O(3, R)$ is a group, and the determinant maps $O(3, R)$ onto $\{\pm 1\}$.The kernel is denoted by $O^+(3, R)$. $O(2, R)$ and $O^+(2, R)$ are analogously defined in the 2x2 case.

1.3.1 The 2-dimensional Orthogonal Group

Let $O(2, R)$, $O^+(2, R)$ be the 2 dimensional orthogonal group, and the subgroup of orthogonal matrices with positive determinant. Let $O^{-1}(2, R)$ be the subset of matrices in $O(2, R)$ with negative determinant. Then $O(2, R) = O^+(2, R) \cup O^{-1}(2, R)$.

Theorem 11 *$O^+(2, R)$ is abelian. If $A \in O^+(2, R)$ and $A^2 = I$, then $A = \pm I$.*

Proof: Let $A \in O^+(2, R)$, $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then

$$A^{-1} = A^T \Longrightarrow \tag{1.17}$$

$$\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \Longrightarrow a = d, b = -c. \tag{1.18}$$

Thus, $A = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$, with $a^2 + b^2 = 1$. So $O^+(2, R)$ can be identified with the unit circle; moreover, this is an isomorphism of groups. And so $O^+(2, R)$ is abelian, and $A \in O^+(2, R)$ has the form $A = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$, $\theta \in [0, 2\pi)$.

Now if $A^2 = I$, then $A = A^{-1}$, so $\begin{pmatrix} a & b \\ -b & a \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}$, from which $b = 0$, and $a^2 = 1$. Thus $A = \pm I$.

Theorem 12 *Any finite subgroup of $O^+(2, R)$ is cyclic.*

Proof: Finite subgroups of the circle group are cyclic, generated by an element closest to the identity, $(1, 0)$.

Theorem 13 *1. If $A \in O^-(2, R), M \in O^+(2, R)$, then $A^{-1}MA = M^{-1}$.*

2. $A \in O^-(2, R) \Longrightarrow A^2 = I$.

Proof: Note first, any $A \in O^-(2, R)$ can be written $A = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} B$, where $B \in O^+(2, R)$. (Let $B = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} A$). So if $M = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$, then we can compute:

$$\begin{pmatrix} -1 & \\ & 1 \end{pmatrix} \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} = M^{-1}.$$

Thus

$$A^{-1}MA = B^{-1} \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} M \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} B = B^{-1}M^{-1}B.$$

But $BM = MB$, so $A^{-1}MA = M^{-1}$

Proof of 2): If $A \in O^-(2, R)$, then $A = \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} B$, and so

$$A^2 = \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} B \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} B = B^{-1}B = I$$

•

1.3.2 The 3-dimensional Orthogonal Group

Definition 11 *For $v, w \in R^n$, let $< v, w >= \sum_i v_i w_i$, the standard inner producton R^n.*

Lemma 1 *If $A \in O(3, R)$, $< v, Aw >=< A^T v, w >$.*

Proof: Switch the order of summation.

Corollary 1 *1.* $< Av, Aw >=< A^T A, w >=< v, w >$

2. $< Av, Av >=< v, v >$.

Thus A preserves orthogonality, and A preserves lengths.

Theorem 14 *If* $A \in O^+(3,R)$ *then, if* $A \neq I$, A *has exactly one eigenvalue 1, and the other two are complex conjugates of norm 1.*

Proof: $|Av| = |v| \Longrightarrow$ any solutions λ to the equation $Av = \lambda v$ must have $|\lambda = 1$. Eigenvalues of A are roots of $p(t) = \det(tI - A)$, a cubic polynomial with real coefficients, that may be factored over the complex numbers as:

$$p(t) = (t - \lambda_1)(t - \lambda_2)(t - \lambda_3).$$

$p(0) = \det(-A) \Longrightarrow$ The product $\lambda_1\lambda_2\lambda_3 = 1$. If the λ_i are all real, either they are all 1, or one of them is 1 and the other 2 are -1. If they are not all real, two of them must be complex conjugates, and the third must be 1. •

Theorem 15 *If* $A \in O^+(3,R)$, A *is similar to a matrix of the form*

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & a & b \\ 0 & c & d \end{pmatrix}, \text{ with } \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in O^+(2,R).$$

Proof: Since A has an eigenvalue 1 space L, $A(L) \subseteq L$. Let $L^\perp$ be the orthogonal complement to L. Then $A(L^\perp) \subseteq L^\perp$, since $v \perp w \Longrightarrow Av \perp Aw$. Form a basis by choosing $v \in L$, u and w independent vectors in $L^\perp$. We may also choose u and w to be orthonormal. Then:

$$Av = v \tag{1.19}$$

$$Au = au + bw \tag{1.20}$$

$$Aw = cu + dw. \tag{1.21}$$

And since $< u, w >= 0, < Au, Aw >= 0$, which implies

$$\begin{aligned} < au + bw, cu + dw > &= < au, cu > + < bw, cu > + < au, dw > + < bw, dw > \\ &= ac + bd \\ &= 0. \end{aligned}$$

Thus $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is an element of $O^+(2, R)$. And if P is the transition matrix from the standard basis to (v, u, w), then

$$P^{-1}AP = \begin{pmatrix} 1 & 0 & 0 \\ 0 & a & b \\ 0 & c & d \end{pmatrix}$$

as claimed.

Definition 12 *If R^3 can be written as the direct sum of orthogonal subspaces preserved by A, as in the above theorem, then A is called completely reducible.*

Finally, we sketch the proof of a result which is a useful tool:

Theorem 16 *If $F \subset SL(3, Z)$ is a finite subgroup, then F is conjugate to a subgroup of $O^+(3, R)$.*

Proof: This is a special case of the the "Weyl trick", and may be sketched as follows:

$$< u, v >' = \frac{1}{|G|} \sum_{g \in G} < gu, gv >$$

defines a positive definite bilinear form, $< , >'$, on R^3. With respect to this new inner product, $g \in G$ is an isometry, and hence can be represented by an orthogonal matrix.)

•

Of course if two matrices are conjugate, then they have the same invariants - trace, determinant, characteristic polynomial, eigenvalues. More precisely, if $P^{-1}MP = M'$, then

1. $tr(M) = tr(M')$, where tr(M) is the trace of M, the sum of the diagonal elements.
2. $\det M = \det M'$.
3. $\det(M - tI) = \det(M' - tI)$, i.e., the characteristic polynomials are equal. Hence,
4. M and M' have the same eigenvalues.

Chapter 2

Classification of Point Groups

2.1 Classification Up To Isomorphism

Our first step is to determine up to isomorphism what finite groups can occur as subgroups of $SL(3,Z)$.

Theorem 17 *If* $M \in SL(3,Z)$, *and* $M^k = I$, *then* $k = 1, 2, 3, 4,$ *or* 6.

Proof: The group generated by M is finite, and so by the Weyl trick is conjugate to a subgroup of $O^+(3,R)$. Let M' be the matrix in $O^+(3,R)$ corresponding to M under this conjugation. The order of M' is also k. Also, M' has the same trace and eigenvalues as M. But by theorem 13, M' has one eigenvalue 1, the other 2 being complex conjugates of norm 1. The trace of M is the sum of the eigenvalues, but it is also clear that the trace of M is an integer (it is an integer matrix). Therefore,

$$tr(M') = 1 + \theta + \overline{\theta} \in Z$$

where $\theta = \cos\alpha + i\sin\alpha$. It follows that $\theta + \overline{\theta} = 2\cos\alpha \in Z$. Since $|\cos\alpha| \leq 1, \cos\alpha = 0, \pm\frac{1}{2},$ *or* ± 1.

$$\begin{aligned}
\cos\alpha = 0 &\Longrightarrow \alpha = \pm\pi/2 &&\Longrightarrow k = 4\\
\cos\alpha = \tfrac{1}{2} &\Longrightarrow \alpha = \pm\pi/6 &&\Longrightarrow k = 6\\
\cos\alpha = -\tfrac{1}{2} &\Longrightarrow \alpha = \pm\pi/3 &&\Longrightarrow k = 3\\
\cos\alpha = 1 &\Longrightarrow \alpha = 0 &&\Longrightarrow k = 1\\
\cos\alpha = -1 &\Longrightarrow \alpha = \pm\pi &&\Longrightarrow k = 2
\end{aligned}$$

Theorem 18 $1, 2, 3, 4,$ *and* 6 *all occur as the order of an element in* $SL(3,Z)$.

Proof: if $C = \begin{pmatrix} 1 & \\ & 1 \end{pmatrix}, \begin{pmatrix} -1 & \\ & -1 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix},$ then

$$\begin{pmatrix} 1 & 0 \\ 0 & C \end{pmatrix}^k = I$$

for $k = 1, 2, 3, 4,$ *and* 6, respectively. •

Theorem 19 *If $F \subset SL(3,Z)$, and F is a finite group, then $|F| = 2^\alpha 3^\beta$.*

Proof: If p divides the order of F, Sylow's theorem gives the existence of a p-group in F. Any element in this p-group which is not the identity will have order a power of p. But this order must also be in the list $\{2,3,4,6\}$. Therefore, p can only be 2, or 3. •

Theorem 20 *If F is a finite subgroup of $SL(3,Z)$, F is solvable.*

Proof: This is a direct application of Burnside's theorem. •

Theorem 21 *Let A be a finite abelian subgroup of $SL(3,Z)$. Then A is isomorphic to one of the following: $1, Z/2, Z/3, Z/4, Z/6$, or $Z/2 \oplus Z/2$.*

Proof: By the Weyl trick, A is conjugate to a subgroup of $O^+(3,R)$. So without loss of generality, we assume that $A \subset O^+(3,R)$.

So suppose that $M \in A \subset O^+(3,R), M \neq I$. Then M has exactly one eigenvalue 1 space, by theorem 13. Let L be this one-dimensional subspace of R^3, and let $v \in L, v \neq 0$. Claim: all matrices of A map L into itself. For if $N \in A$, then

$$\begin{aligned} & MN = NM \\ \Longrightarrow \quad & M(Nv) = N(Mv) = Nv \\ \Longrightarrow \quad & Nv \in L \\ \Longrightarrow \quad & Nv = \lambda v, \text{ since } v \text{ is a basis for } L. \end{aligned}$$

But N preserves lengths, so $\lambda = \pm 1$. Thus all matrices of A have L as eigenspace, and we may write

$$A = \left\{ \begin{pmatrix} \pm 1 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{pmatrix} \right\}$$

Now by the previous result, M and N have order equal to $1, 2, 3, 4$, or 6.

Case I: M has order 3,4, or 6. We may write:

$$M = \begin{pmatrix} 1 & \\ & M_1 \end{pmatrix} \qquad N = \begin{pmatrix} \pm 1 & \\ & N_1 \end{pmatrix}$$

But if the corner entry of N is -1, $\det(N_1) = -1$, and $MN \neq NM$, by theorem 12. Therefore, $\det(N_1) = 1$, and so both M_1 and $N_1 \in O^+(2, R)$. Therefore, $gp(M_1, N_1)$ is cyclic (theorem 11). This group is isomorphic to $gp(M, N)$; and the generator of this group, by the first theorem of this section, has order 2,3,4, or 6. So in this case, the theorem is true.

Case II: M has order 2. Then, by theorem 10, we may write

$$M = \begin{pmatrix} 1 & \\ & -I \end{pmatrix}$$

$$If\ N = \begin{pmatrix} 1 & \\ & N_1 \end{pmatrix},\ and\ N \neq M$$

then the order of N is 3,4, or 6, and we may apply Case I to N. Therefore

$$N = \begin{pmatrix} -1 & \\ & N_1 \end{pmatrix}$$

But N also has an eigenvalue 1, and we may change basis - without changing M - and write N as:

$$N = \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix}$$

Then $gp(M, N) \cong Z/2 \oplus Z/2$.

Since any other matrix commuting with M having a -1 in the upper corner also has an eigenvalue 1, if it is not equal to N, it must in fact be equal to MN. So in this case, the theorem is true. And this also shows that $gp(M, N)$ is maximal abelian in $SL(3, Z)$. •

Theorem 22 *If F is a subgroup of $SL(3, Z)$ isomorphic to one of $Z/4$, $Z/6$, or $Z/2 \oplus Z/2$, then F is maximal abelian in $SL(3, Z)$.*

Proof: The list of possible finite abelian subgroups makes this clear. •

Now any finite F in $SL(3,Z)$ is solvable, and hence has an abelian normal subgroup (theorem 5). But more is true.

Theorem 23 *If $F \leq SL(3,Z)$, F finite, then F contains an abelian normal subgroup which is also maximal abelian in F.*

Proof: Let A be an abelian normal subgroup of F. If A is isomorphic to $Z/4$, $Z/6$, or $Z/2 \oplus Z/2$, A is maximal abelian in $SL(3,Z)$, hence in F. It remains to discuss the cases $A \cong Z/2$ and $A \cong Z/3$.

We use the Weyl trick as before, and assume F is a subgroup of $O^+(3,R)$. Let B be the centralizer of A in $O^+(3,R)$. Then

1. B is normal. (Theorem 6)
2. B is abelian. (This depends on properties of the orthogonal group.)
3. B is maximal abelian.

Hence, B is the desired maximal abelian, normal subgroup of F.

First, we show that B is abelian. The proof of [this theorem - 2] shows that there is 1 eigenvalue 1 space L, shared by all matrices of A. We may further assume each $a \in A$ has the form:

$$a = \begin{pmatrix} 1 & \\ & A_1 \end{pmatrix}$$

Now $A \lhd F \Longrightarrow F$ leaves L invariant.

(Proof: if $v \in L$,

$$\begin{aligned} f^{-1}af = a' &\Longrightarrow af = fa' \\ &\Longrightarrow (af)v = (fa')v = f(a'v) = fv \\ &\Longrightarrow fv \in L.) \end{aligned}$$

Thus F consists of matrices $f_i = \begin{pmatrix} \pm 1 & \\ & F_i \end{pmatrix}$, where $F_i \in O(2,R)$. If such an f_i commutes with a, then F_i commutes with A_1. But this is only possible,

according to theorem 12, if $F_i \in O^+(2,R)$. Hence, the centralizer of A must be abelian.
Finally, B is maximal abelian in F, since if $gp(g,B)$ is abelian, g would also centralize A. •

The existence of an abelian normal subgroup which is also maximal abelian in F strongly structures F. Let A be this maximal abelian normal subgroup. Then:

$$1 \longrightarrow A \longrightarrow F \longrightarrow F/A \longrightarrow 1$$

is exact, and F/A may be identified with a subgroup of $Aut(A)$ (theorem 2)

We now prove that F is always isomorphic to a semi-direct product, by exhiting specific splittings of specific embeddings of the possible A's in $SL(3,Z)$. Theorem 10 then assures us that any extension of this embedded A is split.

Splittings for Completely Reducible Forms

The groups $Aut(A)$ are easy to compute. We list them:

- $Aut(Z/2) \cong \{1\}$
- $Aut(Z/3) \cong Z/2$.
- $Aut(Z/4) \cong Z/2$
- $Aut(Z/6) \cong Z/2$
- $Aut(Z/2 \oplus Z/2) \cong P(3)$ where $P(3)$ is the permutation group on the non-zero elements of $Z/2 \oplus Z/2$.

Note that when $Aut(A) \cong Z/2$, the non-trivial automorphism of A is given by the map $a \mapsto a^{-1}$.

First, we write down extensions of the completely reducible forms for these abelian groups. We handle the maximal abelian groups first. By theorem 10 any other extension of a maximal abelian A in $SL(3,Z)$ must be a subgroup of the one we write down.

- $A \cong Z/2 \Longrightarrow Aut(A) = 1_A \Longrightarrow F = A$
- $A \cong Z/4$. Let $F = gp < \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & 0 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} >$.

  If A is the first matrix, and B is the second matrix, then

  1. $gp(A) \cong Z/4$
  2. $gp(B) \cong Z/2$
  3. $gp(A) \lhd F$
  4. $gp(A) \cap gp(B) = I$, the identity matrix.
  5. Conjugation by B is the non-trivial automorphism of A.

  By theorem 7 we see that $F \cong Z/4 \rtimes_{\phi} Z/2$, where ϕ indicates the isomorphism $Z/2 \longrightarrow Aut(Z/4)$.

- $A \cong Z/6$. The exact same considerations show that

$$F = gp < \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} > .$$

  is isomorphic to $Z/6 \rtimes Z/2$.

- $A \cong Z/2 \oplus Z/2$. In this case, $Aut(A) \cong P(3)$. We represent this group in $SL(3, Z)$ as follows:

$$Let\ \sigma = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}\ and\ let\ \tau = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

  Then $\sigma^3 = I$, $\tau\sigma\tau = \sigma^2$, and so $gp(\sigma, \tau) \cong P(3)$. We are taking

$$A = \left\{ \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}, \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix}, \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix}, \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \right\}$$

  Forming the group generated by A and $P(3)$, we check the usual conditions:

1. A is a normal subgroup.
2. $A \cap P(3) = I$
3. conjugation by elements in $P(3)$ acts as automorphisms of A.

Thus we have exhibited a split extension of $Z/2 \oplus Z/2$ by $P(3)$ in $SL(3,Z)$. We may extend $Z/2 \oplus Z/2$ by subgroups of $P(3)$ by restricting this splitting.

Now let us turn our attention to the case $A \cong Z/3$. Since $Z/3$ is not maximal abelian in $SL(3,Z)$, we don't have uniqueness of split extensions. However, $Z/3$ maximal abelian in $F \Longrightarrow$ the exact sequence

$$1 \longrightarrow A \longrightarrow F \longrightarrow F/A \longrightarrow 1$$

holds, and since $Aut(Z/3) \cong Z/2$, we may proceed as before, verifying that the following two groups are semi-direct products:

1. $F = gp < \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} >.$
2. $F = gp < \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 0 \\ & 0 & -1 \\ & -1 & 0 \end{pmatrix} >.$

Direct computation shows that $\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ are not integer equivalent, and so these two groups are not integer equivalent. They are both isomorphic to $Z/3 \rtimes Z/2$, which imbeds in $Z/6 \rtimes Z/2$ in exactly two ways. Since there is only one form for $Z/6 \rtimes Z/2$, there are only these two forms for $Z/3 \rtimes Z/2$.

We have showed that any completely reducible form is a semi-direct product. Suppose F' is not completely reducible? In this case, F' possesses a maximal abelian normal subgroup A'; this subgroup is conjugate to a subgroup

of $O^+(3,R)$, and hence conjugate to a completely reducible A; and the conjugation carries F' into some extension F of A. F, we know, is a semi-direct product; and $F \cong F' \Longrightarrow F'$ is also.

Summarizing this section, we have classified finite subgroups F of $SL(3,Z)$ up to isomorphism. Any such F is isomorphic to one of the following:

1. $Z/2$
2. $Z/3$
3. $Z/4$
4. $Z/6$
5. $Z/2 \oplus Z/2$
6. $Z/3 \rtimes Z/2$
7. $Z/4 \rtimes Z/2$
8. $Z/6 \rtimes Z/2$
9. $(Z/2 \oplus Z/2) \rtimes P(3)$
10. $(Z/2 \oplus Z/2) \rtimes Z/3$
11. $(Z/2 \oplus Z/2) \rtimes Z/2$

Finally, we note that $(Z/2 \oplus Z/2) \rtimes Z/2 \cong Z/4 \rtimes Z/2$. This is easily seen by finding an element in $(Z/2 \oplus Z/2) \rtimes Z/2$ of order 4, which has a non-trivial automorphism.

Concretely, let $(Z/2 \oplus Z/2) \rtimes Z/2 = \{(a,t) | a \in Z/2 \oplus Z/2,\ t \in Z/2\}$. Let $Z/2 \oplus Z/2 = \{1, x, y, xy\}$. Without loss of generality, let the action of t be $t(x) = y$ ($\Longrightarrow t(y) = x$.) Then $(x,t)(x,t) = (xt(x), 1) = (xy, 1)$, and $(xy,1)(xy,1) = (xyt(xy), 1) = (xyyx, 1) = (1,1)$.

2.2 Classification up to Integer Equivalence

Definition 13 *Let F, F' be finite subgroups of $SL(3,Z)$. If there is a $g \in SL(3,Z)$ such that $gFg^{-1} = F'$, then we say that F is integer equivalent to F', and that F and F' have the same integer form.*

The goal of this section is to enumerate the different integer forms for each isomorphism class of groups determined in the preceding section. We will first do this for the finite abelian subgroups of $SL(3,Z)$, then show how to extend these results to the extensions of these groups.

Theorem 24 *Any $M \in SL(3,Z)$ is integer equivalent to a matrix of the form:*

$$\begin{pmatrix} 1 & n_1 & n_2 \\ & M' & \end{pmatrix}$$

with $M' \in SL(2,Z)$.

Proof: We know from theorem 14 that M has an eigenvalue 1. Thus the equation $Mx = x$ has a non-zero solution. We write this as $(M - I)x = 0$, and observe that since $M - I$ is an integer matrix, there is a solution $v = (r_1, r_2, r_3)$ with each r_i a rational number (the theory of linear equations shows that we may find solutions to a system such as this by means of a finite number of rational operations over the coefficient field.)

The space spanned by v is a line which thus contains points with integer coordinates, and therefore there is a non-zero solution with integer entries which is closest to the origin. Let (a,b,c) be such a solution. Then the greatest common divisor, $gcd(a,b,c)$, must be 1, since otherwise there would be an integer solution closer to the origin.

Claim: (a,b,c) can be completed to a basis for the lattice Z^3. Then, if P is the transition matrix to this new basis, $P^{-1}MP$ is a matrix of the required form. So we complete the proof of the theorem by proving the claim.

Proof of claim: We seek (r,s,t), and $(u,v,w) \in Z^3$ such that

$$\begin{aligned} 1 &= \left| \begin{pmatrix} a & b & c \\ r & s & t \\ u & v & w \end{pmatrix} \right| \\ &= a(sw - tv) - b(rw - tu) + c(rv - us). \end{aligned}$$

But we know that there exists integers h, k, l such that $ah+bk+cl = 1$. Therefore we want to find a solution to the three equations:

$$\begin{aligned} h &= sw - tv \\ -k &= rw - tu \\ l &= rv - us \end{aligned}$$

Let $t = (h, k)$, the greatest common divisor. Then setting $v = -h/t$, and $u = k/t$, we have

1. $tv = -h$
2. $tu = k$
3. $(u, v) = 1$, hence there exists r', s' such that $r'u + s'v = 1$, and so $lr'u + ls'v = l$. So we let $r = lr'$ and $s = ls'$.

Then, letting $w = 0$ we see that $(r, s, t), (u, v, w)$, and t satisfy the necessary equations. •

Theorem 25 *If $M \in SL(3, Z)$, and M has order $k \neq 1$, then M is integer equivalent to $\begin{pmatrix} 1 & a & b \\ & C_\alpha & \end{pmatrix}$ where $a, b \in Z$, and C_α is the companion matrix to the characteristic equation of the matrix $R \in O^+(2, R)$,*

$$R = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}, \quad \textit{where } \alpha = 2\pi/k.$$

Proof: By the previous theorem, we have a basis $\{\alpha_1, \alpha_2, \alpha_3\}$ with respect to which $M = \begin{pmatrix} 1 & a & b \\ & A & \end{pmatrix}$. Then $M^k = I \Longrightarrow A^k = I$. We first show that (a, b) can be any pair of integers. We have the factorization:

$$A^k - I = 0 \Longrightarrow (A-I)(A^{k-1} + A^{k-2} + ... + I) = 0.$$

But 1 is not an eigenvalue of A, since A is similar to a rotation by $2\pi/k$ radians. Thus $\det(A-I) \neq 0$, and so $(A-I)$ is invertible. Thus

$$A^{k-1} + A^{k-2} + ... + I = 0 \ , \quad \text{identically}$$

On the other hand, it is easy to verify that

$$\begin{pmatrix} 1 & a & b \\ & A & \end{pmatrix}^k = \begin{pmatrix} 1 & a' & b' \\ & A^k & \end{pmatrix}$$

where $(a', b') = (a, b) + (a, b)A + ... + (a, b)A^{k-1} = 0$.
Thus, there are no restrictions on (a, b), and we may further alter our basis for R^3, without having to observe any conditions on (a, b). We're only interested in the cases $k = 2, 3, 4, 6$.

$$\begin{aligned}
A^2 - I = 0 &\Longrightarrow (A-I)(A+I) = 0 \Longrightarrow A + I = 0 \Longrightarrow A = -I \\
A^3 - I = 0 &\Longrightarrow (A-I)(A^2 + A + I) = 0 \Longrightarrow A^2 + A + I = 0 \\
A^4 - I = 0 &\Longrightarrow (A^2 - I)(A^2 + I) = 0 \Longrightarrow A^2 + I = 0 \\
A^6 - I = 0 &\Longrightarrow (A^3 - I)(A^3 + I) = 0 \\
&\Longrightarrow A^3 + I = 0 \\
&\Longrightarrow (A+I)(A^2 - A + I) = 0 \\
&\Longrightarrow A^2 - A + I = 0.
\end{aligned}$$

Thus for $k = 3, 4, 6$, A satsifies an equation of degree 2, irreducible over the integers. Now $A \in SL(2, Z)$ and A acts on Z^2, giving it the structure of a module over the ring $Z[A] \cong Z[x]/(p(x))$, where $p(x)$ is one of the above polynomials. Since this ring is a principal ideal domain, Z^2 is a direct sum of cyclic submodules (see, for instance, [7]). Hence, in this case, Z^2 is cyclic, that is, there is an element $v \in Z^2$ such that $< v, Av >$ is a basis for Z^2.

If P is the transition matrix to this basis

$$P^{-1}AP = \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}, \text{ for } k = 3$$
$$P^{-1}AP = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \text{ for } k = 4$$
$$P^{-1}AP = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix}, \text{ for } k = 6$$

That is, there is a change of basis P, such that

$$\begin{pmatrix} 1 & 0 & 0 \\ & P & \end{pmatrix}\begin{pmatrix} 1 & a & b \\ & A & \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ & P^{-1} & \end{pmatrix} = \begin{pmatrix} 1 & a' & b' \\ & C_\alpha & \end{pmatrix}$$

which is what we wanted to show. •

We have thus reduced our problem is to determining conditions on (a, b) that tell us when

$$\begin{pmatrix} 1 & a & b \\ & C & \end{pmatrix} \text{ and } \begin{pmatrix} 1 & a' & b' \\ & C & \end{pmatrix}$$

are integer equivalent.

Lemma 2 *If*

$$P^{-1}\begin{pmatrix} 1 & a & b \\ & C & \end{pmatrix}P = \begin{pmatrix} 1 & a' & b' \\ & C & \end{pmatrix}$$

then P may be chosen to be of the form

$$\begin{pmatrix} 1 & m & n \\ & P_1 & \end{pmatrix}$$

Proof: Any change of basis which brings the (a, b) form into the (a', b') form obviously leaves the first basis vector unchanged. •

Now we may factor such a P as follows:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ & P_1 & \end{pmatrix}\begin{pmatrix} 1 & m & n \\ & I & \end{pmatrix}$$

whence

$$P^{-1} = \begin{pmatrix} 1 & -m & -n \\ & I & \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ & P_1^{-1} & \end{pmatrix}$$

Therefore, we must study the following basic computation:

$$P \begin{pmatrix} 1 & a & b \\ & C & \end{pmatrix} P^{-1} = \begin{pmatrix} 1 & a' & b' \\ & C & \end{pmatrix} \Longrightarrow$$

$$\begin{pmatrix} 1 & 0 & 0 \\ & P_1 & \end{pmatrix} \begin{pmatrix} 1 & m & n \\ & I & \end{pmatrix} \begin{pmatrix} 1 & a & b \\ & C & \end{pmatrix} \begin{pmatrix} 1 & -m & -n \\ & I & \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ & P_1^{-1} & \end{pmatrix} = \begin{pmatrix} 1 & a' & b' \\ & C & \end{pmatrix}$$

Thus we may analyze the situation by first looking at the action of

$$\begin{pmatrix} 1 & m & n \\ & I & \end{pmatrix}$$

and then at the action of

$$\begin{pmatrix} 1 & 0 & 0 \\ & P_1 & \end{pmatrix}$$

Compute:

$$\begin{pmatrix} 1 & m & n \\ & I & \end{pmatrix} \begin{pmatrix} 1 & a & b \\ & C & \end{pmatrix} \begin{pmatrix} 1 & -m & -n \\ & I & \end{pmatrix} = \begin{pmatrix} 1 & (a,b) + (C-I)(m,n) \\ & C & \end{pmatrix}$$

Therefore, if $(a', b') = (a, b) + (C - I)(m, n)$ for some $m, n \in Z$, the forms represented by (a', b') and (a, b) are equivalent.

Before proceeding, we recall a special case of the fundamental theorem of finitely generated abelian groups:

Theorem 26 *Consider the following short exact sequence of groups:*

$$1 \longrightarrow Z \oplus Z \xrightarrow{M} Z \oplus Z \xrightarrow{\pi} A \longrightarrow 1$$

where M is an injective homomorphism, and π is the natural projection onto the coset space. Then if $\det(M) \neq 0$, *A is a finite abelian group whose order is equal to the absolute value of* $\det(M)$.

Note: in the case at hand, $M = (C - I)$, and letting $L^2 = Z \oplus Z$, and $L(C) =$ the two dimensional sub-lattice $(C - I)L^2$, A may be identified with the quotient lattice $L^2/L(C)$. That is, we have the short exact sequence:

$$1 \longrightarrow (C - I)L^2 \xrightarrow{M} Z \oplus Z \xrightarrow{\pi} A \longrightarrow 1$$

We have seen that each coset in this quotient lattice represents possibly distinct integer forms for $\begin{pmatrix} 1 & a & b \\ & & \\ & C & \end{pmatrix}$. But we must also account for the action of matrices of the form $\begin{pmatrix} 1 & 0 & 0 \\ & & \\ & P & \end{pmatrix}$. If

$$\begin{pmatrix} 1 & 0 & 0 \\ & & \\ & P & \end{pmatrix}\begin{pmatrix} 1 & x & y \\ & & \\ & C & \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ & & \\ & P^{-1} & \end{pmatrix} = \begin{pmatrix} 1 & & (x,y)P^{-1} \\ & & \\ & PCP^{-1} & \end{pmatrix} = \begin{pmatrix} 1 & a' & b' \\ & & \\ & C & \end{pmatrix}$$

then we must have

1. $PCP^{-1} = C$, i.e. $PC = CP$
2. $(x, y)P^{-1} = (a', b')$

Definition 14 *If G is a group, $C \in G$, the centralizer of C in G is the subgroup of G consisting of all elements of G that commute with C.*

Lemma 3 *If $PC = CP$, then P maps the sublattice $(C - I)L^2$ into itself, and thus induces an automorphism of the quotient lattice.*

Proof:

$$\begin{aligned} (m, n)(C - I)P &= (m, n)(CP - P) \\ &= (m, n)(PC - P) \\ &= (m, n)P(C - I) \in L(C). \end{aligned}$$

We may summarize the above discussion as follow:

Theorem 27

$$\begin{pmatrix} 1 & a & b \\ & & \\ & C & \end{pmatrix} \text{ is integer equivalent to } \begin{pmatrix} 1 & a' & b' \\ & & \\ & C & \end{pmatrix}$$

if and only if for some $P \in SL(2, Z)$ such that $PC = CP$, we have $(a, b)P = (a', b')$ modulo the lattice $(C - I)L^2$

We now carry out the computation for $Z/2$, $Z/3$, $Z/4$, and $Z/6$.

2.2.1 Integer Forms for $Z/2$

1. $C = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$, and so $C - I = \begin{pmatrix} -2 & 0 \\ 0 & -2 \end{pmatrix}$.
2. $L^2/L(C-I) = (Z \oplus Z)/(2Z \oplus 2Z) \cong Z/2 \oplus Z/2$.

   Thus the possibly distinct forms are represented by $(0,0), (0,1), (1,0), (1,1)$, the elements of the quotient lattice.

3. But the centralizer of $C \in SL(2,Z)$ is all of $SL(2,Z)$, and

$$(0,1)\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = (1\ 1)$$

$$(0,1)\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = (1\ 0)$$

4. Thus, there are only two distinct forms. Representatives may be chosen to be:

$$\begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix} \textit{ and } \begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

2.2.2 Integer Forms for $Z/3$

1. $C = \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}$, and so $C - I = \begin{pmatrix} -1 & -1 \\ 1 & -2 \end{pmatrix}$.
2. $\det(C-I) = 3$, and so the quotient lattice is isomorphic to $Z/3$. Since $(m,n)\begin{pmatrix} -1 & -1 \\ 1 & -2 \end{pmatrix} = (0,k)$ has no integer solutions for $k = \pm 1$, there are 3 possibly distinct forms represented by $(0,0), (0,1), (0,-1)$.
3. But $-I$ is in the centralizer of C, and $(0,1)\begin{pmatrix} -1 & \\ & -1 \end{pmatrix} = (0,-1)$
4. Thus, there are only 2 forms, which we may represent by

$$\begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix} \textit{ and } \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}$$

2.2.3 Integer Forms for $Z/4$

1. $C = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, and so $C - I = \begin{pmatrix} -1 & -1 \\ 1 & -1 \end{pmatrix}$.
2. $\det(C - I) = 2$, and so the quotient is of order two. So there are two possibly distinct forms, which we may represent by $(0, 0)$ and $(0, 1)$. Clearly, these are distinct, since no matrix can move $(0, 0)$. So we have:

$$\begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & 0 \end{pmatrix}$$

2.2.4 Integer Forms for $Z/6$

1. $C = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix}$, $C - I = \begin{pmatrix} -1 & -1 \\ 1 & 0 \end{pmatrix}$
2. $\det(C - I) = 1$, so the quotient lattice has one element, and there is only one integer form:

$$\begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & 1 \end{pmatrix}$$

2.3 Integer Forms for $Z/2 \oplus Z/2$

This computation requires a different approach. Because this is a two-generator group, it is more difficult.

Let $1, A, B, AB$, be a representation of $Z/2 \oplus Z/2$ in $SL(3, Z)$. A has an eigenvalue 1 space, L. By the results on $Z/2$, we may choose a basis such that

$$A = \begin{pmatrix} 1 & a & b \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

Then $A^2 = I, \forall a, b \in Z$.

Now for $v \in L$, $Bv = \pm V$. However, if $B \neq I$, $Bv = v$ implies

$$B = \begin{pmatrix} 1 & a' & b' \\ & B_1 & \end{pmatrix}$$

with $\det(B_1) = 1$, and $B_1^2 = I$. This forces

$$B = \begin{pmatrix} 1 & a' & b' \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

Then $AB = BA$ forces $a = a'$, $b = b'$, and $A = B$.
Thus we see that $B = \begin{pmatrix} -1 & m_1 & m_2 \\ & B_1 & \end{pmatrix}$ with $\det(B_1) = -1$.

But B_1 has an eigenvalue 1, so we may change basis to represent B_1 in the form $\begin{pmatrix} 1 & m \\ & -1 \end{pmatrix}$, and we can do this without disturbing the form of A, because $\begin{pmatrix} -1 & \\ & -1 \end{pmatrix}$ is central, i.e., commutes with everything. Summarizing, we may choose a basis so that

$$A = \begin{pmatrix} 1 & a & b \\ & -1 & 0 \\ & & -1 \end{pmatrix}, \text{ and } B = \begin{pmatrix} -1 & m_1 & m_2 \\ & 1 & m_3 \\ & & -1 \end{pmatrix}$$

We now make 3 observations:

Observation 1:

$$\begin{pmatrix} 1 & 0 & t \\ & 1 & 0 \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & a & b \\ & -1 & 0 \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -t \\ & 1 & 0 \\ & & 1 \end{pmatrix} = \begin{pmatrix} 1 & a & -2t+b \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

Observation 2:

$$\begin{pmatrix} 1 & s & t \\ & 1 & 0 \\ & & 1 \end{pmatrix} \begin{pmatrix} -1 & m_1 & m_2 \\ & 1 & m_3 \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & -s & -t \\ & 1 & 0 \\ & & 1 \end{pmatrix} = \begin{pmatrix} -1 & 2s+m_1 & m_3+m_3 s \\ & 1 & 0 \\ & & -1 \end{pmatrix}$$

Observation 3:

$$\begin{pmatrix} 1 & 0 & s \\ & 1 & t \\ & & 1 \end{pmatrix} \begin{pmatrix} -1 & m_1 & m_2 \\ & 1 & m_3 \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -s \\ & 1 & -t \\ & & 1 \end{pmatrix} = \begin{pmatrix} -1 & m_1 & -m_1 t+m_2 \\ & 1 & m_3-2t \\ & & -1 \end{pmatrix}$$

Observation 1 says we can force $b = 0$ or 1 in A. Observation 3 says we can do this without changing B at all. Meanwhile, observation 2 allows us to force $m_1 = 0$ or 1 in B, and observation 3 allows us to independently force $m_3 = 0$ or 1.

The relation $AB = BA$ puts the following conditions on the m_i and a: (it is noteworthy that there are none put on b)

$$\begin{aligned} m_1 &= -a \\ 2m_2 &= -am_3 \end{aligned}$$

And $B^2 = I \Longrightarrow$

$$2m_2 = m_1 m_3$$

This last relation gives us no new information. However, m_1 and m_3 are both either 0 or 1. Hence, $m_2 = 0$, and m_1 and m_2 can't both be non-zero at the same time. Putting this all together, the possibilities for B, specified by the triple (m_1, m_2, m_3), are:

$$\begin{aligned} (m_1, m_2, m_3) &= (0,0,0), \\ &\quad (0,0,1), \textit{ or} \\ &\quad (1,0,0) \end{aligned}$$

and for each of these, A must satisfy $a = -m_1, b = 0$ or 1. It is more convenient in what follows to force m_1 to be -1, which we may do.

We list the 6 possibilities we must consider:

I.

$$\begin{array}{ccc} A & B & C \\ \begin{pmatrix} 1 & 0 & 0 \\ & -1 & 0 \\ & & -1 \end{pmatrix} & \begin{pmatrix} -1 & 0 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix} & \begin{pmatrix} -1 & 0 & 0 \\ & -1 & 0 \\ & & 1 \end{pmatrix} \end{array}$$

II.

$$\begin{array}{ccc} A & B & C \\ \begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix} & \begin{pmatrix} -1 & 0 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix} & \begin{pmatrix} -1 & 0 & -1 \\ & -1 & 0 \\ & & 1 \end{pmatrix} \end{array}$$

III.

$$A = \begin{pmatrix} 1 & 0 & 0 \\ & -1 & 0 \\ & & -1 \end{pmatrix} \quad B = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 1 \\ & & -1 \end{pmatrix} \quad C = \begin{pmatrix} -1 & 0 & 0 \\ & -1 & -1 \\ & & 1 \end{pmatrix}$$

IV.

$$A = \begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix} \quad B = \begin{pmatrix} -1 & 0 & 0 \\ & 1 & 1 \\ & & -1 \end{pmatrix} \quad C = \begin{pmatrix} -1 & 0 & -1 \\ & -1 & -1 \\ & & 1 \end{pmatrix}$$

V.

$$A = \begin{pmatrix} 1 & 1 & 0 \\ & -1 & 0 \\ & & -1 \end{pmatrix} \quad B = \begin{pmatrix} -1 & -1 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix} \quad C = \begin{pmatrix} -1 & 0 & 0 \\ & -1 & 0 \\ & & 1 \end{pmatrix}$$

VI.

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad B = \begin{pmatrix} -1 & -1 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix} \quad C = \begin{pmatrix} -1 & 0 & -1 \\ & -1 & 0 \\ & & 1 \end{pmatrix}$$

Lemma 4 *II, III, and V are integer equivalent.*

Proof: Each one of these forms has exactly one diagonal matrix. It is easily verified that the permutation of coordinates that takes one of these to another in fact carries one entire representation to the other.

So we focus on I, IV, V, and VI, and show that they are inequivalent.

A in I. is not integer equivalent to any matrix in IV. or VI. (else they would share an eigenvalue 1 space). Similarly, C in V, being diagonal, is not integer equivalent to any matrix in IV or VI.

A, B, and C in I. are integer equivalent, but C is not integer equivalent to A or B in V. Hence, I. is a different form than V.

So the last verification is that IV is not integer equivalent to VI. If they were equivalent, any P conjugating IV to VI would carry the eigenvectors of trhe one group to those of the other.

More precisely, compute:

the eigenvectors associated with eigenvalue 1 in $IV.$:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix}$$

the eigenvectors associated with eigenvalue 1 in VI:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix}$$

If P were a similarity matrix, it would map the one set to the other, at least up to sign. For example, if $A\alpha = \alpha$, $B\beta = \beta$, and $P^{-1}AP = B$ where α generates the unique eigenvalue 1 space of A, and β generates the unique eigenvalue 1 space for B.

$$P^{-1}AP\beta = B\beta = \beta$$

$$A(P\beta) = P\beta$$

$$P\beta = \lambda\alpha$$

but $\det P = 1 \Longrightarrow \lambda = \pm 1$

But then, making these eigenvectors the columns of two matrices, say M and N, we would have $PM = \pm N$; and $\det(M) = \pm \det(N)$ would follow. But this is not true.

2.3.1 Integer Forms of Extensions of Maximal Abelian Subgroups

We have seen that a finite subgroup F of $SL(3,Z)$ fits into the following short exact sequence:

$$1 \longrightarrow A \longrightarrow F \longrightarrow F/A \longrightarrow 1 \tag{2.1}$$

where A is maximal abelian, and thus F/A may be identified with a subgroup of $Aut(A)$.

We have exhibited the extensions for the completely reducible forms at the end of section 2.1. Now we must discuss the extensions for the forms of our abelian groups which are not completely reducible. The automorphism group of $Z/2$ is trivial, and $Z/6$ has only one integer form, the completely reducible one. So we need only look at the remaining forms for $Z/3$, $Z/4$, and $Z/2 \oplus Z/2$.

We take care of $Z/3$ first, then the maximal abelian cases.

We have one non-completely reducible form of $Z/3$, which we have represented by

$$A \cong gp < \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix} >$$

To determine forms for $A \rtimes Z/2$, where $Z/2$ is generated by the non-trivial automorphism of $Z/3$, we first determine β satisfying the following conditions:

1. $\beta \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix} \beta^{-1} = \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}^{-1} = \begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2. $\beta \in GL(2,Z)$, that is, β is an integer matrix with $\det\beta = -1$ ($\det\beta = 1 \Longrightarrow \beta = \pm I$, which would have a non-trivial action on A.)
3. $\beta^2 = I$

Condition 3) implies that $\beta = \begin{pmatrix} a & b \\ c & -a \end{pmatrix}$. Condition 2) implies that $a^2 + bc = 1$.

- Case 1. $a = 0 \Longrightarrow bc = 1 \Longrightarrow b = \pm 1,\ c = \pm 1$ Thus we have

$$\beta = \begin{pmatrix} 0 & \pm 1 \\ \pm 1 & 0 \end{pmatrix}$$

- Case 2. $a^2 \neq 0$ and $a^2 = 1 - bc \Longrightarrow b = 0$, or $c = 0$. Thus $a = \pm 1$. Note: we can't have both b and $c = 0$, since condition 1) would then fail to be fulfilled. So we arrive at the four cases:

  1. $a = 1, c = 0$. Imposing condition 1) on $\begin{pmatrix} 1 & b \\ 0 & -1 \end{pmatrix}$ gives $b = -1$. Similarly,
  2. $a = 1,\ b = 0 \Longrightarrow \beta = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$.
  3. $a = -1,\ c = 0 \Longrightarrow \beta = \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}$, and
  4. $a = -1,\ b = 0 \Longrightarrow \beta = \begin{pmatrix} -1 & 0 \\ -1 & 1 \end{pmatrix}$

But $gp < \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} >$ contains $\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$, and $\begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}$.

Similarly, $gp < \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} >$ contains $\begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$, and $\begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}$.

So we need only look at βs from case 1. Thus we consider

$$\beta_1 = \begin{pmatrix} -1 & n_1 & n_2 \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix}, \text{ and } \beta_2 = \begin{pmatrix} -1 & n_1 & n_2 \\ & 0 & -1 \\ & -1 & 0 \end{pmatrix}$$

and ask: can we determine n_1 and n_2 such that one of these conjugates $\begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}$ to its inverse $\begin{pmatrix} 1 & 1 & 0 \\ & -1 & 1 \\ & -1 & 0 \end{pmatrix}$?

Now $\beta_i^2 = I \Longrightarrow n_1 = n_2$ in β_1, and $n_1 = -n_2$ in β_2.

But $\begin{pmatrix} -1 & n & n \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix} \begin{pmatrix} -1 & n & n \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1-3n & 0 \\ & -1 & 1 \\ & -1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ & -1 & 1 \\ & -1 & 0 \end{pmatrix}$

has no integer solution, whereas

$$\begin{pmatrix} -1 & n & -n \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix} \begin{pmatrix} -1 & n & -n \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -n+1 & 2n \\ & -1 & -1 \\ & -1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ & -1 & 1 \\ & -1 & 0 \end{pmatrix}$$

has one integer solution. So there is only one integer form of $Z/3 \rtimes Z/2$ extending the non-completely reducible form of $Z/3$. Since the completely reducible form has two integer forms, altogether there are 3 distinct integer forms of $Z/3 \rtimes Z/2$

Case: Extensions of $Z/4$

We have one non-completely reducible form, with generator $\begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & -1 & 0 \end{pmatrix}$.

It is an easy matter to verify that $\begin{pmatrix} -1 & 0 & 0 \\ & 0 & -1 \\ & -1 & 0 \end{pmatrix}$ conjugates this into its own inverse, and hence represents the non-trivial automorphism of $Z/4$. Since $Z/4$ is maximal abelian in $SL(3,Z)$, there is no other extension of this form. So altogether for $Z/4 \rtimes Z/2$ there are two integer forms.

Case: Extensions of $Z/2 \oplus Z/2$

There are four forms of this group, and since $Z/2 \oplus Z/2$ is maximal abelian in $SL(3,Z)$, we know that once we extend a representation in $SL(3,Z)$ to a representation of $(Z/2 \oplus Z/2) \rtimes P(3)$, the extension will be unique. We list our forms for $Z/2 \oplus Z/2$:

Form A: $\begin{pmatrix} 1 & 0 & 0 \\ & -1 & 0 \\ & & -1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 & 0 \\ & -1 & 0 \\ & & 1 \end{pmatrix}$

Form B: $\begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix}$ $\begin{pmatrix} -1 & 0 & 0 \\ & 1 & 1 \\ & & -1 \end{pmatrix}$ $\begin{pmatrix} -1 & 0 & -1 \\ & -1 & -1 \\ & & 1 \end{pmatrix}$

Form C: $\begin{pmatrix} 1 & 1 & 0 \\ & -1 & 0 \\ & & -1 \end{pmatrix}$ $\begin{pmatrix} -1 & -1 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix}$ $\begin{pmatrix} -1 & 0 & 0 \\ & -1 & 0 \\ & & 1 \end{pmatrix}$

Form D: $\begin{pmatrix} 1 & 1 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$ $\begin{pmatrix} -1 & -1 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix}$ $\begin{pmatrix} -1 & 0 & -1 \\ & -1 & 0 \\ & & 1 \end{pmatrix}$

(These groups are called **P222, I222, C222, F222** respectively

Now in form (C), we have seen that the diagonal matrix is not integer equivalent to the other two, and thus a cylcic group of order 3 cannot act on this representation non-trivially. Hence, form (C) can only extend to $(Z/2 \oplus Z/2) \rtimes Z/2$, a case we have covered since this group is isomorphic to $Z/4 \rtimes Z/2$.

Form (A) we take care of as follows:

$$P(3) \cong gp < \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix} >$$

and it is straightforward to compute that

$$gp < (A, P(3)) > \cong (Z/2 \oplus Z/2) \rtimes P(3)$$

where by A we mean the group indicated in form (A).

It remains to considere forms B and D. We exhibit splittings for these two forms as follows: we conjugate our diagonal representation into each one, then observe that the conjugating matrices, though not in $SL(3, Z)$, take our given representation of $P(3)$ into integer matrices. Thus B and D have integer splittings, which are unique, since $Z/2 \oplus Z/2$ is maximal abelian in $SL(3, Z)$. The computations are:

$$\text{Let } h_1 = \begin{pmatrix} 1 & 0 & -1 \\ & 1 & -1 \\ & & 2 \end{pmatrix}, \text{ then } h_1^{-1} = \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ & 1 & \frac{1}{2} \\ & & \frac{1}{2} \end{pmatrix}$$

and $h_1(A)h_1^{-1} = B$. We also check that $h_1P(3)h_1^{-1}$ is a group of integer matrices:

$$h_1 \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} h_1^{-1} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 2 & 1 \end{pmatrix}$$

$$h_1 \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} h_1^{-1} = \begin{pmatrix} 0 & -1 & -1 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Completely analagous computations using $h_2 = \begin{pmatrix} 1 & -1 & -1 \\ & 2 & \\ & & 2 \end{pmatrix}$ and $h_2^{-1} = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & 0 \\ & & \frac{1}{2} \end{pmatrix}$ show that Form D also extends to an integer representation of $(Z/2 \oplus Z/2) \rtimes (Aut(Z/2 \oplus Z/2)$.

$$h_2 \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} h_2^{-1} = \begin{pmatrix} -1 & -1 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$h_2 \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} h_2^{-1} = \begin{pmatrix} 1 & -1 & 1 \\ -2 & 0 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

Thus we have 3 integer forms for $(Z/2 \oplus Z/2) \rtimes P(3)$ $P(3)$ has only one subgroup of order 3, from which it follows that $(Z/2 \oplus Z/2) \rtimes P(3)$ has a unique subgroup isomorphic to $(Z/2 \oplus Z/2) \rtimes Z/3$. Each form for the first group restricts to a form for the second, so we also have three forms for $(Z/2 \oplus Z/2) \rtimes Z/3$

This completes the list of conjugacy classes of finite subgroups of $SL(3, Z)$. The next order of business is to extend this list to a list of integer forms for all of $GL(3, Z)$. But first we summarize what we have done so far in a table which lists our various forms, together with the names given to them by crystallographers.

PARTIAL TABLE OF POINT GROUPS

Group	$SL(3,Z)$ forms
I	P1
$Z/2$	P2 B2
$Z/3$	P3 R3
$Z/4$	P4 I4
$Z/6$	P6
$Z/2 \oplus Z/2$	P222 C222 F222 I222
$Z/3 \times Z/2$	P312 P321 R32
$Z/4 \times Z/2$	P422 I422
$Z/6 \times Z/2$	P622
$(Z/2 \oplus Z/2) \times Z/3$	P23 F23 I23
$(Z/2 \oplus Z/2) \times P(3)$	P432 F432 I432

2.3.2 Going from $SL(3,Z)$ to $GL(3,Z)$

Construction I:

For $G \subset SL(3,Z)$, let $G^* = gp(G, -I)$. Then G^* is a group with twice as many elements as G, contained in $GL(3,Z)$. It is obviously not already on our list, since it has elements with determinant -1.

Construction II:

Let $Z/2 = \{\pm 1\}$ If $\alpha : G \longrightarrow Z/2$ is a non-trivial homomorphism, then we have the coset decomposition of G, $G = K(\alpha) \cup C(\alpha)$, where $K(\alpha)$ is the kernel of α, and $C(\alpha)$ is the pre-image of -1. We form a new group,

$$\overline{G}(\alpha) = K(\alpha) \cup -C(\alpha)$$

where $-C(\alpha) = \{-Ig | g \in C(\alpha)\}$. We note the following two things about $\overline{G}(\alpha)$:

1. $\overline{G}(\alpha)$ cannot be conjugate to G, because it has elements with negative determinant.
2. $f : G \longrightarrow \overline{G}(\alpha)$ is an isomorphism, where $f(g) = g$ if $g \in K(\alpha)$, $f(g) = -g$ if $g \in C(\alpha)$.

We need to determine when two homomorphism give rise to integer equivalent $\overline{G}$'s under this construction.

Definition 15 *If $G \subset SL(3,Z)$, and $\alpha, \beta : G \longrightarrow Z/2$, we say $\alpha \sim \beta$ if $\alpha(g) = \beta(hgh^{-1})$ for some $h \in SL(3,Z)$, and for all $g \in G$.*

Theorem 28 *With the notation of Construction II, we have*

$$\overline{G}(\alpha) \sim \overline{G}(\beta) \Longleftrightarrow \alpha \sim \beta$$

Proof: Suppose $\overline{G}(\alpha) = h\overline{G}(\beta)h^{-1}$. Then we may write:

$$\begin{aligned} K(\alpha) \cup -C(\alpha) &= h\,(K(\beta) \cup -C(\beta))\,h^{-1} \\ &= hK(\beta)h^{-1} \cup h(-C(\beta))h^{-1} \end{aligned}$$

Since both unions are disjoint, and since h preserves determinants, $K(\alpha) = hK(\beta)h^{-1}$, and $C(\alpha) = hC(\beta)h^{-1}$. Now if $g \in K(\alpha)$, then $\alpha(g) = 1$. Also $h^{-1}gh \in K(\beta)$, and so $\beta(h^{-1}gh) = 1$. Similarly for elements of $C(\alpha)$, and thus $\alpha \sim \beta$.

Conversely, if $\alpha(g) = \beta(hgh^{-1})$ for some $h \in SL(3,Z)$, then $K(\alpha) = h^{-1}K(\beta)h$, $C(\alpha) = h^{-1}C(\beta)h^{-1}$, and $\overline{G}(\alpha)$ is integer equivalent to $\overline{G}(\beta)$. •

Thus, we must classify, for each $G \subset SL(3,Z)$, inequivalent homomorphisms onto $Z/2$. (Note: the trivial map into $Z/2$ simply recaptures G itself under the above construction.)

Claim: once we have done this, our classification of finite subgroups of $GL(3,Z)$ up to integer equivalence will be complete. For if G is a subgroup of $GL(3,Z)$, $G \not\subset SL(3,Z)$, then $\det : G \longrightarrow Z/2$ is onto. If $-I \in G$, then $G = gp(K, -I)$, where K is the kernel of det. And if $-I \notin G$, then forming $G^+ = K \cup -H$, where K is the kernel, and H is the other coset, we see that $G^+ \subset SL(3,Z)$, and $G = \overline{G^+}(\alpha)$, where $\alpha : G \longrightarrow Z/2$ is obtained from the determinant map in the obvious way.

2.3.3 Classification of Homs

In this section we determine the non-trivial inequivalent homomorphisms from each integer form in $SL(3,Z)$ to $Z/2$.

I. $Z/2$

$Z/2$ admits one non-trivial map into itself, the identity map.

II. $Z/3$

$Z/3$ admits no non-trivial homomorphism to $Z/2$.

III. $Z/4$

$Z/4$ admits one non-trivial homomorphism to $Z/2$.

IV. $Z/6$

$Z/6$ admits one non-trivial homomorphism to $Z/2$.

V. $Z/2 \oplus Z/2$

$Z/2 \oplus Z/2$ requires more work. Let $Z/2 \oplus Z/2 = \{1, a, b, c\}$. Abstractly, there are 3 non-trivial maps to $Z/2$, with kernel consisting of $gp(a)$, $gp(b)$, and $gp(c)$ respectively. Two such maps are equivalent if and only if both the kernel and the other coset are integer equivalent.

We must look at each form separately. We refer to the list of the four inequivalent forms found in the section on extensions of $Z/2 \oplus Z/2$. We have seen, in that section, that forms (A),(B), and (D) all admit extensions to $(Z/2 \oplus Z/2) \rtimes Aut(Z/2 \oplus Z/2)$. This means that for these forms, the full group of automorphisms can be represented as a subgroup of $SL(3, Z)$. In other words, if g_1, g_2 are any two elements in one of these forms, there is a matrix in $SL(3, Z)$ which conjugates one to the other; thus any two homomorphisms onto $Z/2$ are equivalent, and these 3 forms each admits 1 equivalence of such maps.

Form (C), on the other hand, only admits one automorphism, which is of order 2, and which can be represented by an integer matrix. This means that two of the elements of this form are equivalent, and so there are 2 classes of homomorphisms in this case.

Summary:

Form A admits 1 equivalence class of $Z/2$ homs;

Form B admits 1 equivalence class of $Z/2$ homs;

Form C admits 2 equivalence class of $Z/2$ homs;

Form D admits 1 equivalence class of $Z/2$ homs.

Now we continue on to discuss the extensions.

VI. Extensions of $Z/6$ and $Z/4$

There are 3 inequivalent homomorphisms of $Z/n \rtimes Z/2$ onto $Z/2$, where $n = 4$ or 6. To prove this, we need some preliminary lemmas. We shall discuss the case $n = 6$, but the discussion is identical for $n = 4$.

Lemma 5 *If $G = Z/6 \rtimes Z/2$, then $[G, G] \cong Z/3$, and $gp(a^3) \cong Z/2$ is central.*

Proof: G may be presented as $< a, b | a^6 = 1, b^2 = 1, bab = a^{-1} >$. Then $b = b^{-1}$, so $aba^{-1}b^{-1} = aba^{-1}b = aa = a^2$. Thus the group generated by the commutators, $[G, G] = gp(a^2 | a \in Z/6) \cong Z/3$.

Next, $ba^3b^{-1} = a^{-3} = a^3$, and so b and a^3 commute. So a^3 commutes with everything, i.e., is central. •

Lemma 6 *If $G \subset SL(3, Z)$, and $G \cong Z/6 \rtimes Z/2$, and if $h \in SL(3, Z)$ is such that $h^{-1}Gh = G$, then $h \in G$.*

Proof: We use the same presentation for G as in the previous lemma.

$Z/6 \cong Z/3 \oplus Z/2$, and we have seen that $Z/2$ is central, and $Z/3$ is $[G, G]$, hence preserved by any automorphism of $Z/6$. Thus h determines an automorphism of $Z/6$.

If $h^{-1}ah = a$, then since $Z/6$ is maximal abelian in $SL(3, Z)$, $h \in Z/6$.

If the action of h is not trivial, it must map $a \mapsto a^{-1}$. The matrix hb acts by conjugation:

$$\begin{aligned} (hb)a(hb)^{-1} &= h(bab^{-1})h^{-1} \\ &= ha^{-1}h^{-1} \\ &= a. \end{aligned}$$

Thus, hb and a commute, and since $Z/6$ is maximal abelian in $SL(3, Z)$ (again!), $hb \in Z/6$, which means that $h \in G$.

Theorem 29 *There are 3 inequivalent homomorphisms of $Z/6 \rtimes Z/2$ onto $Z/2$.*

Proof: Any homomorphism into an abelian group factors through $G/[G,G]$ as follows:

$$\begin{array}{ccc} G & \longrightarrow & Z/2 \\ \downarrow & \nearrow & \\ G/[G,G] & & \end{array}$$

And any homomorphism from $G/[G,G]$ determines one from G, by composition with the canonical projection. Now $G =< a, b | a^6 = 1, b^2 = 1, bab = a^{-1} >$. $G/[G,G]$ will have the additional relation $ab = ba$, hence

$$a = b^2 a = b(ba) = bab = a^{-1}.$$

Thus, in $G/[G,G]$ $a^2 = 1$, and $G/[G,G] \cong Z/2 \oplus Z/2$.

Now there are three homomorphisms from $Z/2 \oplus Z/2$ onto $Z/2$. Suppose $\alpha, \beta : G \longrightarrow Z/2$ are induced maps corresponding to any two of these.

These maps are equivalent if and only if $\alpha(g) = \beta(h^{-1}gh)$ for some $h \in SL(3,Z)$. But the second lemma says that if $h^{-1}Gh = G$, $h \in G$, so β is actually defined on the element h. But then we have:

$$\begin{aligned} \alpha(g) &= \beta(h^{-1}gh) \\ &= \beta(h)^{-1}\beta(g)\beta(h) \\ &= \beta(g) \end{aligned}$$

since these values are in an abelian group. Thus $\alpha \sim \beta \Longrightarrow \alpha = \beta$, and the three distinct homomorphism are in fact inequivalent.

VII. Extensions of $Z/3$

$Z/3 \rtimes Z/2$ admits 1 non-trivial homomorphism onto $Z/2$. The Sylow theorem implies that $Z/3$ has exactly one subgroup of order 3, and since it is of index two it is normal. The quotient map by this subgroup is the map onto $Z/2$.

VII. Extensions of $Z/2 \oplus Z/2$

Theorem 30 *If $G = (Z/2 \oplus Z/2) \rtimes Z/3$, then $[G,G] = Z/2 \oplus Z/2$, and $G/[G,G] = Z/3$.*

Proof: Let $Z/2 \oplus Z/2 = \{1, a, b, ab\}$ and let $Z/3 = \{1, t, t^2\}$. Let the action of t be described as follows:

$$tat^2 = b, tbt^2 = ab$$

Then the various commutators are:

$$[t,a] = tat^2a = ba$$

$$[t,b] = tbt^2b = abb = a$$

$$[t,ab] = tabt^2ab = tat^2tbt^2ab = baab = 1$$

Thus, $[G,G] = gp(a,b) = Z/2 \oplus Z/2$.

$G/[G,G] \cong gp(t) \cong Z/3$. Since there are no non-trivial maps from $Z/3$ to $Z/2$, the same is true of G. •

Theorem 31 *If $G = (Z/2 \oplus Z/2) \rtimes P(3)$, then there is exactly one non-trivial map from G to $Z/2$.*

Proof: We show that the abelianization of G is $Z/2$. Let $G = gp(a,b,s,t)$, where $gp(s,t) = P(3)$. Let t be the automorphism of order 3, and s be the automorphism of order 2. Since $P(3)$ is isomorphic to $Z/3 \rtimes Z/2$, we have the relation $sts = t^2$. If we add the relation $st = ts$, we get $t = t^2$, or $t = 1$. Without loss of generality, $tat^{-1} = b$ and $tbt^{-1} = ab$; hence in $G/[G,G]$, $a = b = 1$. Thus $G/[G,G] \cong gp(s) \cong Z/2$.

Group	$SL(3,Z)$ forms	# of $Z/2$ homs	forms induced by homs
I	1	0	0
$Z/2$	2	1	2
$Z/3$	2	0	0
$Z/4$	2	1	2
$Z/6$	1	1	1
$Z/2 \oplus Z/2$	Form A	1	1
	Form B	1	1
	Form C	2	2
	Form D	1	1
$Z/3 \rtimes Z/2$	3	1	3
$Z/4 \rtimes Z/2$	2	3	6
$Z/6 \rtimes Z/2$	1	3	3
$(Z/2 \oplus Z/2) \rtimes Z/3$	3	0	0
$(Z/2 \oplus Z/2) \rtimes P(3)$	3	1	3
Total:	24		25

We summarize the information developed so far in the above table.

For each integer form in $G \in SL(3,Z)$ we may form a new space group by adjoining $-I$ to get $gp < G, -I >$. Since groups of the form $gp(G,-I)$ are:

1. not in $SL(3,Z)$
2. not groups which arise from the hom construction (such groups do not contain $-I$)

we can list 73 point groups as follows: 24 $SL(3,Z)$ forms, 24 forms $gp(G,-I)$, and 25 forms from the hom construction.

For convenience we organize a table now with all the crystallographic names as they appear in the X-Ray Tables. The discussion needed to establish this correspondence is given in the following section.

TABLE OF POINT GROUPS

Group	$SL(3,Z)$ forms	adjoin $-I$	Hom constr.
I	P1	P$\bar{1}$	
$Z/2$	P2	P2/m	Pm
	B2	B2/m	Bm
$Z/3$	P3	P$\bar{3}$	
	R3	R$\bar{3}$	
$Z/4$	P4	P4/m	P$\bar{4}$
	I4	I4/m	I$\bar{4}$
$Z/6$	P6	P6/m	P$\bar{6}$
$Z/2 \oplus Z/2$	P222	Pmmm	Pmm2
	C222	Cmmm	Amm2,Cmm2
	F222	Fmmm	Fmm2
	I222	Immm	Imm2
$Z/3 \rtimes Z/2$	P312	P$\bar{3}$1m	P3m1
	P321	P$\bar{3}$m1	P31m
	R32	R$\bar{3}$m	R3m
$Z/4 \rtimes Z/2$	P422	P4/mmm	P4mm,P$\bar{4}$2m,P$\bar{4}$m2
	I422	I4/mmm	I4mm,I$\bar{4}$2m,I$\bar{4}$m2
$Z/6 \rtimes Z/2$	P622	P6/mmm	P6mm,P$\bar{6}$2m,P$\bar{6}$m2
$(Z/2 \oplus Z/2) \rtimes Z/3$	P23	Pm3	
	F23	Fm3	
	I23	Im3	
$(Z/2 \oplus Z/2) \rtimes P(3)$	P432	Pm3m	P$\bar{4}$3m
	F432	Fm3m	F$\bar{4}$3m
	I432	Im3m	I$\bar{4}$3m

2.4 Comparison with The International Tables for X-Ray Crystallography

We want to correlate our list of 73 point groups with those in the above mentioned Tables, Vol. I, ([6]). This is a non-trivial matter, because the notation is different enough to require some computations to establish the correspondence. This section should be read with a copy of the Tables for reference. Let me also say at the outset that this is an ad hoc process, necessary to correlate what's been done with crystallographic naming conventions. More will be said later.

As an example of what we must do, we discuss the forms for the group $Z/2$ in some detail. Recall our two forms for this group in $SL(3,Z)$:

Form I. The group generated by $\begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}$

Form II. The group generated by $\begin{pmatrix} 1 & & 1 \\ & -1 & \\ & & -1 \end{pmatrix}$

The third entry in the tables is $P2$. The symmetries for $P2$ are written as follows:

$$x, y, z;\ \overline{x}, \overline{y}, z.$$

The bar over the x and y denotes a negative sign. This notation represents general coordinates of the unit cell under the action of the group. Clearly, form number 1 represents this action, if we switch the roles of the x and z coordinates. Thus $P2$ corresponds to form 1 for $Z/2$.

But there is also $B2$, number 4 in the Tables. (Note: there are 2 equivalent listings for certain groups. They are given the same space group number and label, and differ only in the choice of the z-axis.) $B2$ lists the same symmetry as $P2$, but also has an added condition, which is stated: "Co-ordinates of equivalent positions." This coordinates in this case are: $0,0,0;\ \frac{1}{2},0,\frac{1}{2}$. This means that the point $\frac{1}{2},0,\frac{1}{2}$ is to be added to the lattice.

The interpretation of this is that the unit cell chosen for the lattice is not primitive, that is, it contains points not on the corners of the unit cell. Before we continue, let us make a few definitions to formalize these notions.

Definition 16 *A primitive unit cell for a 3-dimensional lattice is a parallelopiped such that every lattice point is an integral linear combination of 3 of the edges.*

For every lattice, there is a primitive unit cell, in fact infinitely many. However, crystallographers often choose unit cells which are not primitive.

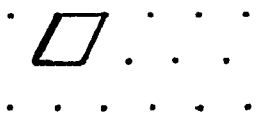
Definition 17 *A non-primitive unit cell for a 3 dimensional lattice is a parallelopiped which contains lattice points other than at the corners, such that the lattice is the set of all integral linear combinations of all these points, both corner and interior.*

Such unit cells are labelled by crystallographers according to the following system: (for more information and pictures, see [2].)

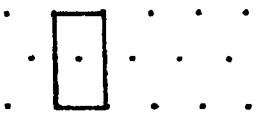
- A,B, or C - with one pair of opposite faces centered, i.e. with a lattice point at the center of the faces.
- F - all faces centered
- I - body centered, i.e. with a lattice point at the center of the unit cell.
- P - primitive
- R - either a primitive cell in the shape of a rhombohedron, or a cell with 2 internal points spaced 1/3 and 2/3 along the long body diagonal.

The reason for non-primitive lattices is that they are more revealing of symmetries; for example, in the two dimensional case compare:

PRIMITIVE CELL



CENTERED UNIT CELL



Note: it is clear that if we take the edges of a non-primitive cell as the basis for our lattice, the centered points will require fractional coordinates. But if we fill out the centered points to a basis, then the symmetries of the lattice only require integer coordinates. This motivates our approach.

We will now show that $B2$ corresponds to our second form for the group $Z/2$.

We complete $(\frac{1}{2}, 0, \frac{1}{2})$ to a basis as follows:

$$a = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad b = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

Every lattice point is, by definition of the centered lattice, an integral linear combination of this basis. If A is the symmetry element of rotation by 180 degrees around the x axis, then with respect to the standard basis,

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}$$

Representing A in terms of (a, b, c) we have: $Aa = a$; $Ab = -b$; $Ac = a - c$; and so

$$A \sim \begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

More concretely, writing the change of basis explicitly:

$$A = \begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ & 1 & \\ & & 2 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ & 1 & 0 \\ & & \frac{1}{2} \end{pmatrix}$$

The moral of this is that our $SL(3, Z)$ representations correspond to primitive and non-primitive unit cells. Specifically, we have that completely reducible representations correspond to primitive space groups, and non completely reducible representations correspond to non-primitive space groups.

Continuing with the example $G = Z/2$, we next want to find $gp(G, -I)$. This is a simple matter - $P2/m$, number 10 in the tables, is easily seen to correspond to:

$$\begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}, \begin{pmatrix} -1 & & \\ & -1 & \\ & & -1 \end{pmatrix}, \begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix}, \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$$

again, with a trivial relabeling of coordinates.

And $B2/m$, number 12 in the tables, is the non-primitive version of the same construction.

Finally, we must apply our homomorphism construction. There is one non-trivial homomorphism mapping $Z/2$ onto $Z/2$, the identity. The pre-image of 1 is $M = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}$. Since $-I(M) = \begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$, we must find a group which in crystallographic notation looks like: $x, y, z;\ \overline{x}, y, z$. But number 6, Pm, has this form, again with a relabeling of coordinates. And finally, Bm, number 8, is the non-primitive version of this construction. So summarizing what we have done, we have the following list for our various representations of $Z/2$:

Space Group Number	Symbol	Construction
3.	P2	$Z/2$, first form
5.	B2	$Z/2$, second form
6.	Pm	P2, hom construction
8.	Bm	B2, hom construction
10.	P2/m	$gp(P2, -I)$
12.	B2/m	$gp(B2, -I)$

Now we will go through the tables, and correlate the point groups with our constructed list. Since there are 13 completely reducible forms in $SL(3, Z)$ according to our list, we must make 13 computations similar to the example given. After that, the only work to be done is when there is more than one inequivalent

homomorphism. Otherwise, correlating the table entries with our list is very straightforward. So first we will go through the tables sequentially, and write down, for each space group, which construction on our list it corresponds to. We put an asterisk by the ones which take a computation. These computations are gathered together at the end.

Before proceeding, however, let us point out that we are forging this correspondence in an ad hoc manner. In other words, what we have done is, by trial and error, determined bases in terms of which the given group is represented by our forms. Is there some mathematical, intrinsic explanation for this choice, or is it truly arbitrary? I do not know. It would certainly be more satisfying if we could show some intrinsic way of making the identification between the irreducible forms in our list, and the non-primitive crystallographic groups. On this matter we are, for the moment, still in the dark.

Space Group Number	Symbol	Construction
1.	P1	I, the identity group
2.	$P\bar{1}$	$gp(I, -I)$
3.	P2	$Z/2$, first form
4.	B2	$Z/2$, second form
6.	Pm	P2, hom constr.
8.	Bm	B2, hom constr.
10.	P2/m	$gp(P2, -I)$
12.	B2/m	$gp(B2, -I)$
16.	P222	$Z/2 \oplus Z/2$, form A
20.*	C222	$Z/2 \oplus Z/2$, form C
22.*	F222	$Z/2 \oplus Z/2$, form D
23.*	I222	$Z/2 \oplus Z/2$, form B
25.	Pmm2	P222, hom constr.
35.	Cmm2	C222, hom constr.
38.	Amm2	C222, hom constr.
42.	Fmm2	F222, hom constr.
44.	Imm2	I222, hom constr.
47.	Pmmm	$gp(P222, -I)$
65.	Cmmm	$gp(C222, -I)$
69.	Fmmm	$gp(F222, -I)$
71.	Immm	$gp(I222, -I)$
75.	P4	$Z/4$, reducible form
79.*	I4	$Z/4$, second form
81.	$P\bar{4}$	P4, hom constr.
82.	$I\bar{4}$	I4, hom constr.
83.	P4/m	$gp(P4, -I)$
87.	I4/m	$gp(I4, -I)$
89.	P422	$Z/4 \rtimes Z/2$, reducible form

97.*	I422	$Z/4 \rtimes Z/2$, non reducible form
99.*	P4mm	P422, hom constr.
107.	I4mm	I422, hom constr.
111.*	P$\bar{4}$2m	P422, hom constr.
115.*	P$\bar{4}$m2	P422, hom constr.
119.	I$\bar{4}$m2	I422, hom constr.
121.	I$\bar{4}$2m	I422, hom constr.
123.	P4/mmm	$gp(P422, -I)$
139.	I4/mmm	$gp(I422, -I)$
143.	P3	$Z/3$, reducible form
146.*	R3	$Z/3$, non-reducible form
147.	P$\bar{3}$	$gp(P3, -I)$
148.	R$\bar{3}$	$gp(R3, -I)$
149.	P312	$Z/3 \rtimes Z/2$, reducible form, using $\begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$
150.	P321	$Z/3 \rtimes Z/2$, reducible form, using $\begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix}$
155.	R32	$Z/3 \rtimes Z/2$, irreducible form
156.	P3m1	P321, hom constr.
157.	P31m	P312, hom constr.
160.	R3m	R32, hom constr.
162.	P$\bar{3}$1m	$gp(P312, -I)$
164.	P$\bar{3}$m1	$gp(P321, -I)$
166.	R$\bar{3}$m	$gp(R32, -I)$
168.	P6	$Z/6$, the only form
174.	P$\bar{6}$	P6, hom constr.

175.	P6/m	$gp(P6, -I)$
177.	P622	$Z/6 \rtimes Z/2$, only one form
183.	P6mm	P622, hom constr.
187.	P$\bar{6}$m2	P622, hom constr.
189.	P$\bar{6}$2m	P622, hom constr.
191.	P6/mmm	$gp(p622, -I)$

195.	P23	$(Z/2 \oplus Z/2) \rtimes Z/3$, reducible form
196.	F23	$(Z/2 \oplus Z/2) \rtimes Z/3$, second form
197.	I23	$(Z/2 \oplus Z/2) \rtimes Z/3$, third form
200.	Pm3	$gp(P23, -I)$
202.	Fm3	$gp(F23, -I)$
204.	Im3	$gp(I23, -I)$
207.	P432	$(Z/2 \oplus Z/2) \rtimes P(3)$, reducible
209.	F432	$(Z/2 \oplus Z/2) \rtimes P(3)$, second form
210.	I432	$(Z/2 \oplus Z/2) \rtimes P(3)$, third form
215.	P$\bar{4}$3m	P432, hom constr.
216.	F$\bar{4}$3m	F432, hom constr.
217.	I$\bar{4}$3m	I432, hom constr.
221.	Pm3m	$gp(P432, -I)$
225.	Fm3m	$gp(F432, -I)$
229.	Im3m	$gp(I432, -I)$

2.4.1 Computations

Now we include the matrix computations necessary to establish the correspondence between the crystallography tables and the mathematical classification of the point groups. We have handled the monoclinic groups, in our example, so we begin with the orthorhombic class.

Centered Unit Cells in the Orthorhombic Class

Body-Centered Cells. One point is added to the lattice, $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$, and all its translates. We complete this vector to a basis as follows:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{2} \end{pmatrix}$$

Then the change of basis computation is:

$$\begin{pmatrix} 1 & 0 & -1 \\ & 1 & -1 \\ & & 2 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ & 1 & \frac{1}{3} \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 0 & -1 \\ & 1 & -1 \\ & & 2 \end{pmatrix} \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ & 1 & \frac{1}{3} \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ & 1 & 1 \\ & & -1 \end{pmatrix}$$

Thus what we have called Form B corresponds to I222.

Side-Centered Cells. In this case, the point added to the unit cell is $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$, and we complete to a basis as follows:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

and we change to this basis as follows:

$$\begin{pmatrix} 1 & -1 & 0 \\ & 2 & 0 \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & 0 \\ & \frac{1}{2} & 0 \\ & & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & -1 & 0 \\ & 2 & 0 \\ & & 1 \end{pmatrix} \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & 0 \\ & \frac{1}{2} & 0 \\ & & 1 \end{pmatrix} = \begin{pmatrix} -1 & -1 & 0 \\ & 1 & 0 \\ & & -1 \end{pmatrix}$$

And so our Form C corresponds to C222.

Face-Centered Cells. In this case, points are added to the center of each face; thus the "coordinates of equivalent points " are:

$$\begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

We map this basis for a rational lattice to one which is integer equivalent:

$$\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Then with respect to this basis, we compute:

$$\begin{pmatrix} 1 & -1 & -1 \\ & 2 & 0 \\ & & 2 \end{pmatrix} \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & 0 \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -1 & 0 & -1 \\ & -1 & 0 \\ & & 1 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & -1 & -1 \\ & 2 & 0 \\ & & 2 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & 0 \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix}$$

Thus Form D corresponds to F222.

The body-centered cells and the face-centered cells come up again, in the cubic class; only body centered cells exists in the tetragonal class; the side-centered cell exists only in the orthorhombic class.

In these other cases we use the same change of basis as we did here. The cubic groups are extensions of orthorhombic groups, so all we need check as that under the above conjugations, the automorphisms go into integer matrices.

Body Centering and Face Centering in the Cubic Class
Body-Centering:

$$\begin{pmatrix} 1 & 0 & -1 \\ & 1 & -1 \\ & & 2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ & 1 & \frac{1}{2} \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 2 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & -1 \\ & 1 & -1 \\ & & 2 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ & 1 & \frac{1}{2} \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 0 & -1 & -1 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Face-Centering:

$$\begin{pmatrix} 1 & -1 & -1 \\ & 2 & \\ & & 2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & 0 \\ & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -1 & -1 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -1 & -1 \\ & 2 & \\ & & 2 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & 0 \\ & 0 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 \\ -2 & 0 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

Body Centering in the Tetragonal Class

$$\begin{pmatrix} 1 & -1 & -1 \\ & 2 & \\ & & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & 0 \end{pmatrix}$$

To handle the one extension of this form, we again check that this change of basis takes the automorphism to a matrix in $SL(3, Z)$. In fact, it takes the automorphism to itself:

$$\begin{pmatrix} 1 & -1 & -1 \\ & 2 & \\ & & 2 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ & 0 & -1 \\ & -1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & \\ & & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ & 0 & -1 \\ & -1 & 0 \end{pmatrix}$$

The Non-primitive Lattice in the Trigonal Class

$$\begin{pmatrix} -1 & 0 & 0 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Thus we see that our non-primitive form for $Z/3$ is integer equivalent to the representation given in the tables for the group R3: $x, y, z;\ z, x, y;\ y, z, x$. These are the rhombohedral coordinates. The unit cell for this space group is primitive, but three times the volume as the unit cell for the completely reducible form.

The hexagonal axes for the trigonal group have coordinates of equivalent positions listed as:

$$\begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ \frac{2}{3} \end{pmatrix} ; \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix}$$

The computation:

$$\begin{pmatrix} -1 & -1 & 0 \\ & 2 & -1 \\ & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix} \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} \\ & \frac{1}{3} & \frac{1}{3} \\ & -\frac{1}{3} & \frac{2}{3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}$$

is a change of basis from primitive to hexagonal coordinates which also preserves the integrality of the automorphism $\begin{pmatrix} -1 & & \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix}$. Explicitly:

$$\begin{pmatrix} -1 & -1 & 0 \\ & 2 & -1 \\ & 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} \\ & \frac{1}{3} & \frac{1}{3} \\ & -\frac{1}{3} & \frac{2}{3} \end{pmatrix} = \begin{pmatrix} -1 & 0 & -1 \\ & -1 & 1 \\ & 0 & 1 \end{pmatrix}$$

For completeness, we include the change of basis from primitive to rhombohedral coordinates:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix} \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & -\frac{2}{3} & \frac{1}{3} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & -\frac{2}{3} & \frac{1}{3} \end{pmatrix} = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$$

Since there are no non-primitive forms in the hexagonal class, we are done.

Chapter 3

Translations

3.1 Introduction

In this section, we determine how to attach translations to our point groups. First we recall some material from the introduction.

Let $R(3)$ be the group of rigid motions of Euclidean 3 space. Let T be the subgroup of pure translations. Let $O(3)$ be the group of rigid motions which fix a point. Then

$$1 \longrightarrow T \longrightarrow R(3) \longrightarrow O(3) \longrightarrow 1$$

is an exact sequence which splits. Thus $R(3) \cong T \rtimes O(3)$. We may represent elements of $R(3)$ by

$$M = \begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix}$$

where $A \in O(3)$, and $X \in R^3$.

A crystallographic group is a group $\Gamma \subset R(3)$ such that

1. $\Gamma \cap T$ is a lattice.

2. $\Gamma/(\Gamma \cap T)$ is finite.

Condition 1) $\Longrightarrow$ that the quotient can be imbedded into $GL(3, Z)$, and condition 2) $\Longrightarrow$ that, with a change of basis, the translational components are rational numbers; that is, if Γ is a crystallographic group, we may represent Γ as a set

$$\Gamma = \{ \begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix} | A \in GL(3, Z), X \in Q^3 \}$$

We say that two such representations are equivalent if they are related by a change of basis which carries the lattice for one onto the lattice for the other. Thus,

$$\Gamma \sim \Gamma'$$

if

$$\begin{pmatrix} B & Y \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B^{-1} & -B^{-1}Y \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A' & X' \\ 0 & 1 \end{pmatrix}$$

where $B \in SL(3, Z)$. From this equation it follows:

Condition I $BAB^{-1} = A'$

Condition II $-(BAB^{-1} - I)Y + BX = X'$

We may break this computation up into two stages, as follows:

$$\begin{pmatrix} I & Y \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B^{-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I & -Y \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} I & Y \\ 0 & 1 \end{pmatrix} \begin{pmatrix} BAB^{-1} & BX \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I & -Y \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} A' & X' \\ 0 & 1 \end{pmatrix}$$

Thus, we may first classify the A's, which is what we have done in the point group classification, and then add in the translation pieces.

Proceeding this way, we must limit ourselves to B's that commute with A, once we have settled on a form for a particular point group. This simplifies condition II above to read:

$$-(A - I)Y + BX = X'$$

If $B = I$, this tells us that we may always replace

$$X \mapsto -(A - I)Y + X$$

where Y is any vector in Q^3. We will want to kill translations modulo the lattice, that is, see if we can map replace X with the 0 vector. This amounts to solving the system of equations

$$-(A-I)Y = -X$$

In particular, if $A-I$ is non-singular, this may be done. In general, the rank of A is the number of components in X which we may kill. When we speak of conjugating by Y, what we mean is replacing X with $-(A-I)Y+X$.

What we do with each form is

1. Try to kill translations by conjugating
2. Limit the remaining translations by using the fact that some finite power of each element in Γ is 0 modulo the lattice.
3. Impose the conditions arising out of the group multiplication between elements. For instance, if the point group in question has elements A, and B, and $AB = BA$, then

$$\begin{pmatrix} A & t_A \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B & t_B \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} B & t_B \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A & t_A \\ 0 & 1 \end{pmatrix}$$

$\Longrightarrow$

$$At_B + t_A = Bt_A + t_B$$

This procedure results in a list of possibilities. We then usually use B's that commute with A to collapse some of these, or, if there is no such B, note that we are done.

This is the general schema. Our program for the rest of this section is to apply this schema to, first, the primitive (completely reducible) forms in $SL(3,Z)$. After that's done, we'll go through forms arising out of the hom construction. Next, we'll attack the forms which come from adjoining $-I$. Finally, we'll turn our attention to the non-primitive forms.

3.2 Primitive forms in $SL(3, Z)$

3.2.1 $P2$

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \; t_A = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

We want to determine t_A.

STEP I

Replace $\begin{pmatrix} a \\ b \\ c \end{pmatrix} \mapsto -(A-I)Y + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$

Since

$$A - I = \begin{pmatrix} 0 & & \\ & -2 & \\ & & -2 \end{pmatrix}$$

$$(A-I)Y = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

becomes

$$\begin{pmatrix} 0 \\ -2y_2 \\ -2y_3 \end{pmatrix} = \begin{pmatrix} 0 \\ b \\ c \end{pmatrix}$$

which can be solved for y_2 and y_3. So I may replace $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ with $\begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix}$.

STEP II Let L denote the lattice.

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A^2 & AX + X \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} \text{ modulo L}$$

$$\Longrightarrow AX = -X \Longrightarrow \begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -a \\ 0 \\ 0 \end{pmatrix}$$

$$\Longrightarrow 2a = 0 \text{ mod L} \Longrightarrow t_A = \begin{pmatrix} a/2 \\ 0 \\ 0 \end{pmatrix}$$

where $a = 0, 1$. $a = 0$ gives $P2$. So we have one form with non-primitive translations: $P2_1$

$$\begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix}$$

3.2.2 $P3$

$$A = \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}; t_A = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

STEP I

$$A - I = \begin{pmatrix} 0 & 0 & 0 \\ & -1 & -1 \\ & 1 & -2 \end{pmatrix}$$

so I can kill b and c by conjugation.

STEP II Impose finite order.

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix}^3 = \begin{pmatrix} A^3 & A^2 + AX + X \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} \text{ modulo L}$$

$$\Longrightarrow (A^2 + A + I) \begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Longrightarrow 3a = 0 \text{ mod L} \Longrightarrow t_A = \begin{pmatrix} a/3 \\ 0 \\ 0 \end{pmatrix}$$

where $a = 0, 1, 2$. $a = 0$ gives $P3$. So we have two forms with non-primitive translations: $P3_1$, and $P3_2$.

$$\begin{pmatrix} 1 & & \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}; \begin{pmatrix} \frac{1}{3} \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 & & \\ & 0 & -1 \\ & 1 & -1 \end{pmatrix}; \begin{pmatrix} \frac{2}{3} \\ 0 \\ 0 \end{pmatrix}$$

3.2.3 $P4$

$$A = \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & 0 \end{pmatrix}; \; t_A = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

STEP I

$$A - I = \begin{pmatrix} 0 & 0 & 0 \\ & -1 & -1 \\ & 1 & -1 \end{pmatrix}$$

so I can kill b and c by conjugation.

STEP II Impose finite order.

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix}^4 = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} \text{ modulo L}$$

$$\Longrightarrow (A^3 + A^2 + A + I) \begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 4a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Longrightarrow 4a = 0 \text{ mod L} \Longrightarrow t_A = \begin{pmatrix} a/4 \\ 0 \\ 0 \end{pmatrix}$$

where $a = 0, 1, 2, 3$. So we get three forms with non-primitive translations: $P4_1$, $P4_2$, and $P4_3$.

3.2.4 $P6$

$$A = \begin{pmatrix} 1 & 0 & 0 \\ & 0 & -1 \\ & 1 & 1 \end{pmatrix}; \; t_A = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

STEP I

$$A - I = \begin{pmatrix} 0 & 0 & 0 \\ & -1 & -1 \\ & 1 & 2 \end{pmatrix}$$

so I can kill b and c by conjugation.

STEP II Impose finite order.

$$\begin{pmatrix} A & X \\ 0 & 1 \end{pmatrix}^6 = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} \text{ modulo L}$$

$$\Longrightarrow 6a = 0 \text{ mod L} \Longrightarrow t_A = \begin{pmatrix} a/6 \\ 0 \\ 0 \end{pmatrix}$$

where $a = 0, 1, 2, 3, 4, 5$. So we get five forms with non-primitive translations: $P6_1$, $P6_2$, $P6_3$, $P6_4$, $P6_5$.

3.2.5 $P222$

Things get a little more interesting when we leave the realm of cyclic groups.

Let $A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}$; $B = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix}$ $C = \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix}$

and let $t_A = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$. $A - I = \begin{pmatrix} 0 & & \\ & -2 & \\ & & -2 \end{pmatrix} \Longrightarrow$ we can kill b and c,

and let $t_A = \begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix}$. Also

$$A^2 = I \Longrightarrow 2a = 0 \text{ mod L}$$

So we have $t_A = \begin{pmatrix} a/2 \\ 0 \\ 0 \end{pmatrix}$ where $a = 0, 1$.

Let $t_B = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$

Now in conjugating t_A to this form, we used a vector $Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$, but we only specified y_2 and y_3. Since y_1 is still free, we may further conjugate in order to kill b_1.

In addition, $AB = BA = C$ and so

$$At_B + t_A = Bt_a + t_B$$

$$\Longrightarrow \begin{pmatrix} 0 \\ -b_2 \\ -b_3 \end{pmatrix} + \begin{pmatrix} a/2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a/2 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ b_2 \\ b_3 \end{pmatrix} \text{ mod L}$$

$$\Longrightarrow 2b_2 = 0,\ 2b_3 = 0 \text{ mod L.}$$

So we may replace $t_B \mapsto \begin{pmatrix} 0 \\ b_2/2 \\ b_3/2 \end{pmatrix}$, with $b_2, b_3 = 0, 1$.

And $AB = C$, $t_C = At_B + t_A$, so

$$t_C = \begin{pmatrix} a/2 \\ b_2/2 \\ b_3/2 \end{pmatrix} \text{ mod L}$$

Two possibilities for t_A, 4 possibilities for t_B, give 8 possibilities, one of which is $P222$. Only four of these are in fact inequivalent; we write them all down, then relate them.

Possible translations consistent with $P222$.

		t_A	t_B	t_C	
1.	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$P222$
2.	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}0\\0\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}0\\0\\\frac{1}{2}\end{pmatrix}$	$P222_1$
3.	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\0\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\0\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\0\end{pmatrix}$	$P222_1$
4.	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$	$P2_12_12$
5.	$\begin{pmatrix}\frac{1}{2}\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\0\\0\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\0\\0\end{pmatrix}$	$P222_1$
6.	$\begin{pmatrix}\frac{1}{2}\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\0\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\0\\\frac{1}{2}\end{pmatrix}$	$P2_12_12_1$
7.	$\begin{pmatrix}\frac{1}{2}\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\0\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\\frac{1}{2}\\0\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\\frac{1}{2}\\0\end{pmatrix}$	$P2_12_12$
8.	$\begin{pmatrix}\frac{1}{2}\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$	$\begin{pmatrix}\frac{1}{2}\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$	$P2_12_12_1$

The first 5 are clearly equivalent to the forms named. In 6., conjugation by $Y = \begin{pmatrix}0\\\frac{1}{4}\\0\end{pmatrix}$ maps the given set into $\begin{pmatrix}\frac{1}{2}\\\frac{1}{2}\\0\end{pmatrix}$, $\begin{pmatrix}0\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$ $\begin{pmatrix}\frac{1}{2}\\0\\\frac{1}{2}\end{pmatrix}$, modulo the lattice, which is the way $P2_12_12_1$ appears in the tables.

In 7. conjugation by the same vector results in the set $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$.

Finally, in 8., conjugation by $Y = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \\ 0 \end{pmatrix}$ gives the set $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$.

3.2.6 $P321$ and $P312$

We have showed how to attach translations to $P3$, and we got

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & -1 & \\ & & 1 \end{pmatrix}, \; t_A = \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix}$$

where $a = 1, 2$. Now we want to look at forms for $Z/3 \rtimes Z/2$, i.e., add the automorphism together with its translation vector; (we look at $P321$ first)

$$P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix}; \; t_P = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

First, note that we may replace $\begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \mapsto \begin{pmatrix} t_1 \\ t_2 \\ 0 \end{pmatrix}$ using $(P - I)Y$ to kill t_3. The reason is that in the conjugating we did to get A into its form we only used y_1, and y_2.

Next, $P^2 = I \Longrightarrow Pt_P + t_P = 0 \Longrightarrow t_2 + t_1 = 0$ And so we have

$$t_P = \begin{pmatrix} t \\ -t \\ 0 \end{pmatrix}$$

Next,

$$P^{-1}AP = A^{-1} \Longrightarrow AP = PA^{-1}$$

$$\Longrightarrow At_P + t_A = Pt_{A^{-1}} + t_P$$

$$\text{but } t_{A^{-1}} = -A^{-1}t_A = \begin{pmatrix} 0 \\ 0 \\ -a/3 \end{pmatrix}$$

so we get

$$\begin{pmatrix} 0 & -1 & \\ 1 & -1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} t \\ -t \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -a/3 \end{pmatrix} + \begin{pmatrix} t \\ -t \\ 0 \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} t \\ 2t \\ \frac{1}{3} \end{pmatrix} = \begin{pmatrix} t \\ -t \\ \frac{1}{3} \end{pmatrix} \Longrightarrow 3t = 0$$

$$\Longrightarrow t = \tfrac{1}{3},\ \tfrac{2}{3}$$

But $-I$ commutes with everything, so $\begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix} \sim \begin{pmatrix} -\frac{1}{3} \\ -\frac{2}{3} \\ 0 \end{pmatrix} \sim \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ 0 \end{pmatrix}$ modulo the lattice.

But $t_P = \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix}$ can be moved to $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ by conjugating by $\begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ 0 \end{pmatrix}$; and this change of basis does not change t_A, modulo the lattice.

So we have 3 forms corresponding to

$$t_A = \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix},\ a = 0, 1, 2 \text{ and } t_P = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

$P321$, $P3_121$, $P3_221$ (correspond to $t_P = 0$.)

NEXT: $P312$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & -1 & \\ & & 1 \end{pmatrix};\ t_A = \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix}\ P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix};\ t_P = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

$$Pt_P + t_P = \begin{pmatrix} -t_2 \\ -t_1 \\ -t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0$$

$$\Longrightarrow t_1 = t_2$$

Again, we may kill t_3 by conjugating, without affecting t_A.

$$\text{So we may write : } t_P = \begin{pmatrix} t \\ t \\ 0 \end{pmatrix}$$

$$P^{-1}AP = A^{-1} \Longrightarrow$$

$$AP = PA^{-1} \Longrightarrow At_P + t_A = Pt_{A^{-1}} + t_P$$

$$\Longrightarrow \begin{pmatrix} 0 & -1 & \\ 1 & -1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} t \\ t \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -a/3 \end{pmatrix} + \begin{pmatrix} t \\ t \\ 0 \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} -t \\ 0 \\ a/3 \end{pmatrix} = \begin{pmatrix} t \\ t \\ a/3 \end{pmatrix}$$

Thus, $t = 0$. So the only translation vector we may add to the automorphism is the zero vector. So in addition to $P312$, there are $P3_112, P3_212$.

3.2.7 $P422$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix}; t_A = \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix} \quad P = \begin{pmatrix} 0 \\ -1 \end{pmatrix} -10-1; t_P = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

I. Kill t_3, as usual.

II. $P^2 = I \Longrightarrow Pt_P + t_P = 0$

$$\Longrightarrow \begin{pmatrix} -t_2 \\ -t_1 \\ 0 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ 0 \end{pmatrix} = 0$$

$$\Longrightarrow t_1 = t_2$$

So $t_P = \begin{pmatrix} t \\ t \\ 0 \end{pmatrix}$

III $AP = PA^{-1} \Longrightarrow At_P + t_A = Pt_{A^{-1}} + t_P$

$$\Longrightarrow \begin{pmatrix} -t \\ t \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix} + \begin{pmatrix} t \\ t \\ 0 \end{pmatrix}$$

$$\Longrightarrow t = -t \Longrightarrow 2t = 0 \Longrightarrow t = \tfrac{1}{2}.$$

Thus we arrive at

$$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & \\ & & 1 \end{pmatrix}; \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}; \begin{pmatrix} b/2 \\ b/2 \\ 0 \end{pmatrix}$$

with $a = 0, 1, 2, 3$ and $b = 0, 1$.

And these 8 forms correspond to the following entries in the table:

1. $P422$ (a = 0; b = 0)
2. $P4_122$ (a = 1; b = 0)
3. $P4_222$ (a = 2; b = 0)
4. $P4_322$ (a = 3; b = 0)
5. $P42_12$ (a = 0; b = 1)
6. $P4_12_12$ (a = 1; b = 1)
7. $P4_32_12$ (a = 2; b = 1)
8. $P4_32_12$ (a = 3; b = 1)

However, there are some differences in the appearance of the entries in the table, due to different choices of the origin. For example

- In number 2, conjugating by $Y = \begin{pmatrix} 0 \\ 0 \\ 1/8 \end{pmatrix}$ will shift the z axis in P, without affecting A.
- In number 3, similarly, conjugating by $Y = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{4} \end{pmatrix}$ will make our form look like the tables.

- In number 4, conjugate by $Y = \begin{pmatrix} 0 \\ 0 \\ 3/8 \end{pmatrix}$.
- in number 5, conjugating by $Y = \begin{pmatrix} \frac{1}{3} \\ 0 \\ 0 \end{pmatrix}$ kills the translational part of the automorphism, and puts it into the A matrix.

Similar remarks apply to numbers 6,7, and 8.

3.2.8 *P*622

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 1 & \\ & & 1 \end{pmatrix}; \; t_A = \begin{pmatrix} 0 \\ 0 \\ a/6 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix}; \; \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

I. Kill t_3 by conjugating.

II. $Pt_P + t_P = 0 \Longrightarrow \begin{pmatrix} t_2 \\ t_1 \\ 0 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ 0 \end{pmatrix} = 0$

$$\Longrightarrow t_1 = -t_2$$

$$\text{so } t_P = \begin{pmatrix} t \\ -t \\ 0 \end{pmatrix}$$

III $AP = PA^{-1} \Longrightarrow At_P + t_A = Pt_{A^{-1}} + t_P$

$$\text{but } t_{A^{-1}} = -A^{-1}t_A = \begin{pmatrix} 0 \\ 0 \\ -a/6 \end{pmatrix}$$

$$\text{Thus} \begin{pmatrix} 0 & -1 & \\ 1 & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} t \\ -t \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/6 \end{pmatrix} = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -a/6 \end{pmatrix} + \begin{pmatrix} t \\ -t \\ 0 \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} t \\ 0 \\ a/6 \end{pmatrix} = \begin{pmatrix} t \\ -t \\ a/6 \end{pmatrix}$$

And thus, $t = 0$. So in all cases, the only consistent way to add a translation to the automorphism, is if it is zero. So we have the forms: $P6_122, P6_222, P6_322, P6_422, P6_522$.

Again, the tables move the origin to other places than where we have them; the details are omitted.

3.2.9 $P23$

This is the primitive form of $Z/2 \oplus Z/2 \rtimes Z/3$. What we must look at is

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}, t_A; \; B = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix}, t_B; \; C = \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix}, t_C$$

where t_A, t_B, t_C run through the possibilities listed in the section on $P222$; and

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \; t_P = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

We will first examine the consistency relations stemming from $PA = BP$, $PB = CP$, $PC = AP$. The following relations follow:

1. $Pt_A + t_P = Bt_P + t_B$
2. $Pt_B + t_P = Ct_P + t_C$
3. $Pt_C + t_P = At_P + t_A$

In the cases where $t_A = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, we have from relation (1):

$$t_P = Bt_P + t_B \Longrightarrow$$

$$\begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} -t_1 \\ -t_2 \\ t_3 \end{pmatrix} + t_B$$

If t_B has a $\frac{1}{2}$ in the 3d coordinate, we get an inconsistency: $\frac{1}{2} = 0$ modulo L.

In the other case where $t_A = 0$, we have $t_B = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$, and $t_C = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$.

Then relation (2) $\Longrightarrow$

$$\begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} -t_1 \\ t_2 \\ -t_3 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$$

which is again inconsistent, in the second coordinate.

So far we have ruled out adding our automorphism to $P222_1$ and $P2_12_12$. The only case left is $P2_12_12_1$. We take

$$t_A = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix} ; \; t_B = \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} ; \; t_C = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

We now show that we may add P consistently, but only if $\begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

We again apply the consistency relations (1),(2), and (3).

$$(1) \Longrightarrow \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} -t_1 \\ -t_2 \\ t_3 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

$$\Longrightarrow 2t_1 = 0 \text{ and } 2t_2 = 0 \text{ mod L}$$

From condition (2) it also follows that $2t_3 = 0$.

But

$$\begin{pmatrix} P & t_P \\ 0 & 1 \end{pmatrix}^3 = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} \text{ mod L} \Longrightarrow$$

$$(P^2 + P + I)t_P = 0 \Longrightarrow t_1 + t_2 + t_3 = 0$$

$\Longrightarrow$ either all the t_i are 0, or exactly 2 are equal to $\frac{1}{2}$. Thus the possibilities are:

$$t_P = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix} \text{ or } \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \text{ or } \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

But $P - I$ has rank 2, so I can kill these by conjugating by vectors of the form $Y = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix}$; e.g.

$$-(P-I)\begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix} =$$

$$\begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ mod L}$$

And it is readily verified that conjugating by such Y leaves t_A, t_B, t_C invariant modulo the lattice. Therefore there is only one way to add translations to $P23$, and this gives us the form $P2_13$.

3.2.10 $P423$

This is the case of $Z/2 \oplus Z/2 \rtimes P(3)$. So we're looking at

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \; B = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix}; \; C = \begin{pmatrix} -1 & & \\ & 1 & \\ & & -1 \end{pmatrix}$$

with the associated t_A, t_B, t_C, as above; and

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}; \; Q = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$$

with their associated t_P, t_Q.

Case I The discussion for $P23$ shows that if any of t_A, t_B, t_C are non-zero, then we must have $P2_12_12_1$, and $t_P = 0$. So we determine conditions on $t_Q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$

$$I.\ Q^3 = I \Longrightarrow t_Q = \begin{pmatrix} q \\ q \\ q_3 \end{pmatrix}$$

$$II.\ QA = AQ \Longrightarrow Qt_A + t_Q = At_Q + t_A$$

$$QB = CQ \Longrightarrow Qt_B + t_Q = Ct_Q + t_C$$

$$QC = BQ \Longrightarrow Qt_C + t_Q = Bt_Q + t_B$$

From these we derive that $2q = 0$, and $2q_3 = 0$, modulo the lattice.

$$III. PQ = QP^{-1} \Longrightarrow Pt_Q + t_P = Qt_{P^{-1}} + t_Q$$

But $t_P = 0$, so

$$Pt_Q = t_Q \Longrightarrow \begin{pmatrix} q_3 \\ q \\ q \end{pmatrix} = \begin{pmatrix} q \\ q \\ q_3 \end{pmatrix} \Longrightarrow t_Q = \begin{pmatrix} q \\ q \\ q \end{pmatrix}$$

Thus, $t_Q = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ or $t_Q = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$

Case II We can also have $P222$, with $t_P = 0$, and $t_Q = \begin{pmatrix} q \\ q \\ q_3 \end{pmatrix}$. The consistency relations with A, B, C, and P again imply that t_Q must be of the form $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$.

So we may summarize the 432 case as follows: we have the following possibilities for adding non-primitive translations to the point group 432.

1. $P222, t_P = 0, t_Q = 0$
2. $P222, t_P = 0, t_Q = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$
3. $P2_12_12_1, t_P = 0, t_Q = 0$
4. $P2_12_12_1, t_P = 0, t_Q = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$

3.3 Groups Arising From the Hom Construction, and Non-primitive Translations

The aim of this section is to determine how one can attach translations to those point groups constructed by means of the hom construction. Recall that the basic idea is to look at a subgroup of $G \subset SL(3, Z$ and a subgroup of index two in $H \subset G$; and to form a new group consisting of H together with $(-I)(G-H)$ (set-theoretic difference).

For each such group, we want to see how many different inequivalent groups there are with non-primitive translations which have the given group as point group. (See the Table of Point Groups).

If we organize the results according to isomorphism class, we get the following:

Isomorphism Class	Point Group	Extensions
$Z/2$	Pm	Pb
$Z/3$	no homs	none
$Z/4$	$P\bar{4}$	none
$Z/6$	$P\bar{6}$	none
$Z/2 \oplus Z/2$	$Pmm2$	$Pmc2_1, Pcc2, Pma2, Pca2_1, Pnc2,$ $Pmn2_1, Pba2, Pna2_1, Pnn2$
$Z/3 \rtimes Z/2$	$P3m1$	$P3c1$
	$P31m$	$P31c$
$Z/4 \rtimes Z/2$	$P4mm$	$P4bm, P4_2cm, P4_2nm, P4cc, P4nc, P4_2mc, P4_2bc$
	$P\bar{4}2m$	$P\bar{4}2c, P\bar{4}2_1m, P\bar{4}2_1c$
	$P\bar{4}m2$	$P\bar{4}c2, P\bar{4}b2, P\bar{4}n2$
$Z/6 \rtimes Z/2$	$P6mm$	$P6cc, P6_3cm, P6_3mc$
	$P\bar{6}2m$	$P\bar{6}2c$
	$P\bar{6}m2$	$P\bar{6}c2$
$Z/2 \oplus Z/2 \rtimes Z/3$	no homs	none

$Z/2 \oplus Z/2 \times P(3)$ $P\bar{4}3m$ $P\bar{4}3n$

We go through the point groups in question, and show that for each form in the second column, there are the required number of extensions. The details of correlating with the names in the tables is omitted. It involves changing the origin, as has been illustrated in the previous sections.

3.3.1 $Z/2$

$P2 = gp(A)$, where $A = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix}$. The hom construction applied to $P2$ gives the point group Pm. Let $N = -A = \begin{pmatrix} 1 & & \\ & 1 & \\ & & -1 \end{pmatrix}$, and let $t_N = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$ Then

$$-A - I = \begin{pmatrix} 0 & & \\ & 0 & \\ & & -2 \end{pmatrix} \Longrightarrow$$

we can kill t_3. Also $N^2 = I \Longrightarrow$

$$Nt_N + t_N = 0 \text{ mod L}, \Longrightarrow \begin{pmatrix} t_1 \\ t_2 \\ 0 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Thus $2t_1 = 2t_2 = 0$, and $t_i = 0$ or $\frac{1}{2}$. So the possible non-primitive translations are:

$$\begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}; \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix}; \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$$

But $t_N \sim Bt_N$, where $B \in SL(3, Z)$ commutes with A. Since

$$\begin{pmatrix} 1 & 1 & \\ 0 & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix} \text{ mod L}$$

$$\text{and} \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$$

We have only one non-primitive translation, which gives us the form Pb.

3.3.2 $Z/4$

$P4 = gp(A)$, where $A = \begin{pmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix}$ Let $N = -A$. Since

$$N - I = \begin{pmatrix} -1 & 1 & \\ -1 & -1 & \\ & & -2 \end{pmatrix}$$

is non-singular, we can kill all the translations. So we can add no non-primitive translations to the point group $P\bar{4}$.

3.3.3 $Z/6$

The identical argument works for this case, since the generating matrix for $P\bar{6}$ is $\begin{pmatrix} 0 & 1 & \\ -1 & -1 & \\ & & -1 \end{pmatrix}$.

3.3.4 $Z/2 \oplus Z/2$

The hom construction applied to $P222$ yields $Pmm2$:

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; B = \begin{pmatrix} 1 & & \\ & -1 & \\ & & 1 \end{pmatrix}; C = \begin{pmatrix} 1 & & \\ & 1 & \\ & & -1 \end{pmatrix}$$

We have to determine 9 non split extensions of this group.

Case I: $t_A = 0$, $t_B = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$

$$B^2 = I \Longrightarrow t_B = \begin{pmatrix} a/2 \\ b \\ c/2 \end{pmatrix}$$

$$AB = BA \Longrightarrow A \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Longrightarrow -b = b$$

Thus we have $t_B = t_C = \begin{pmatrix} a/2 \\ b/2 \\ c/2 \end{pmatrix}$ Now $\begin{pmatrix} -1 & & \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix}$ is an automorphism of $Pmm2$, switching B and C; and carries

$$\begin{pmatrix} a/2 \\ \frac{1}{2} \\ 0 \end{pmatrix} \mapsto \begin{pmatrix} a/2 \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

Therefore, for $a = 0$ we have 3 possibilities: $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$; $\begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$ $\begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$; and for $a = \frac{1}{2}$, the same cases for b and c. Thus we have 6 possibilities here, but one of them is $Pmm2$.

Case II: A has translations, $t_A = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$. Since $A^2 = I$, $a = \frac{1}{2}$. And we may kill the b and c, since $A - I = \begin{pmatrix} 0 & & \\ & -2 & \\ & & -2 \end{pmatrix}$. Thus

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \; t_A = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix}$$

We also get:

$$B = \begin{pmatrix} 1 & & \\ & -1 & \\ & & 1 \end{pmatrix}; \; t_B = \begin{pmatrix} a/2 \\ b/2 \\ c/2 \end{pmatrix}$$

where $B^2 = I$ restricts the a and c; and $AB = BA$ forces the condition on b.

$$AB = C \Longrightarrow t_C = At_B + t_A, \text{ so}$$

$$t_B = \begin{pmatrix} 0 \\ b/2 \\ c/2 \end{pmatrix} \Longrightarrow t_C = \begin{pmatrix} \frac{1}{2} \\ b/2 \\ c/2 \end{pmatrix}$$

$$t_B = \begin{pmatrix} \frac{1}{2} \\ b/2 \\ c/2 \end{pmatrix} \Longrightarrow t_C = \begin{pmatrix} 0 \\ b/2 \\ c/2 \end{pmatrix}$$

Thus, switching B and C by means of the group action listed above preserves the bottom two components, and lets us pick one of these two.

So we have 4 cases, corresponding to $b, c = 0, 1$.

3.3.5 $Z/3 \rtimes Z/2$

There are two cases to consider:

Case I

$$P312 = gp < A = \begin{pmatrix} 0 & -1 & \\ 1 & -1 & \\ & & 1 \end{pmatrix}, P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix} >$$

Then $P31m = gp(A, -P)$

$$A - I = \begin{pmatrix} -1 & -1 & \\ 1 & -2 & \\ & & 0 \end{pmatrix} \Longrightarrow t_A = \begin{pmatrix} 0 \\ 0 \\ a \end{pmatrix}$$

by conjugation. And $A^3 = I \Longrightarrow t_A = \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix}$, with $a = 0, 1$, or 2. Next:

$$t_Q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} . \; Q^2 = I \Longrightarrow Qt_Q + t_Q = 0$$

$$\Longrightarrow t_1 = -t_2,\ 2t_3 = 0;\ \text{ so } t_Q = \begin{pmatrix} t \\ -t \\ t_3/2 \end{pmatrix},\ t_3 = 0, 1.$$

Next:

$$AQ = QA^{-1} \Longrightarrow At_Q + t_A = Qt_{A^{-1}} + t_Q$$

$$\Longrightarrow \begin{pmatrix} t \\ 2t \\ t_3/2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -a/3 \end{pmatrix} + \begin{pmatrix} t \\ -t \\ t_3/2 \end{pmatrix}$$

$$\Longrightarrow 3t = 0;\ 2a/3 = 0 \bmod \mathrm{L} \Longrightarrow a \text{ is a multiple of 3.}$$

Thus $t_A = 0$, and $t = 0, \frac{1}{3}, \frac{2}{3}$.

So far we have two possibilities: $t_Q = \begin{pmatrix} \frac{1}{3} \\ -\frac{1}{3} \\ t_3/2 \end{pmatrix}$ or $\begin{pmatrix} \frac{2}{3} \\ -\frac{2}{3} \\ t_3/2 \end{pmatrix}$.

Since $\frac{1}{3} \sim -\frac{2}{3} \bmod \mathrm{L}$, and $-I$ is central, these two possibilities are in fact equivalent. But more: conjugating by $Y = \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix}$ leaves t_A alone:

$$(A - I)Y = \begin{pmatrix} -1 & -1 & \\ 1 & -2 & \\ & & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}$$

and kills the first two components of t_Q:

$$-(Q - I)Y + t_Q = \begin{pmatrix} 1 & -1 & \\ -1 & 1 & \\ & & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{1}{3} \\ \frac{1}{3} \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 0 \end{pmatrix}.$$

Thus there is exactly one non-trivial extension of $P31m$; given by $t_Q = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$.

Case II

$$P321 = gp < A, P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix} >$$

The hom construction gives:

$$P3m1 = gp < A, -P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & 1 \end{pmatrix} >$$

Let $Q = -P$.

I. As above $t_A = \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix}$, with $a = 0, 1, 2$.

II. $t_Q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$; $Q^2 = 0 \Longrightarrow$

$$Qt_Q + t_Q = 0 \Longrightarrow \begin{pmatrix} -t_2 \\ -t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0$$

$\Longrightarrow t_1 = t_2$, and $2t_3 = 0$. Thus $t_Q = \begin{pmatrix} t \\ t \\ t_3 \end{pmatrix}$. Next,

$$AQ = QA^{-1} \Longrightarrow At_Q + t_A = Qt_{A^{-1}} + t_Q$$

$$\Longrightarrow \begin{pmatrix} -t \\ 0 \\ t_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -a/3 \end{pmatrix} + \begin{pmatrix} t \\ t \\ t_3 \end{pmatrix}.$$

From this we conlude, as in the preceding case, that $t_A = 0$; but also, that $t = 0$. Therefore, the only non-trivial non-primitive translation is given by $t_P = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$.

3.3.6 $Z/4 \rtimes Z/2$

$P422$ is the group generated by

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix} ; \; P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$$

There are 3 subgroups of index 2:

1. I, A, A^2, A^3, with complementary set P, AP, A^2P, A^3P
2. I, A^2, P, A^2P, with complementary set A, A^3, AP, A^3P
3. I, AP, A^2, A^3P, with complementary set A, P, A^2P, A^3

By inspection of the tables, it is evident that

- $P4mm = gp < A, -P >$
- $P\bar{4}m2 = gp < -A, P >$
- $P\bar{4}2m = gp < -A, -P >$

We analyze these three cases in the usual way. First, $P4mm$.

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix} ; \; t_A = \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix}$$

with $a = 0, 1, 2, 3$ as calculated in the previous section.

$$Q = -P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix} ; \; t_P = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

I. $Q^2 = I \Longrightarrow Qt_Q + t_Q = 0 \Longrightarrow$

$$\begin{pmatrix} t_2 \\ t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \Longrightarrow$$

$$t_1 = -t_2 \text{ and } 2t_3 = 0 \text{ mod L}$$

II. $AQ = QA^{-1} \Longrightarrow At_Q + t_A = Qt_{A^{-1}} + t_P$

$$\Longrightarrow \begin{pmatrix} t \\ t \\ t_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -a/4 \end{pmatrix} + \begin{pmatrix} t \\ -t \\ 0 \end{pmatrix}$$

$$\Longrightarrow 2t = 0;\ 2a/4 = 0 \text{ so } a = 0, 2.$$

So we have 8 possibilities here, one of them being the point group itself, corresponding to:

$a = 0, 2$

$t = 0, \frac{1}{2}$

$t/3 = 0, \frac{1}{2}$.

NEXT CASE: $P\bar{4}m2$ Here we must look at

$$N = -A = \begin{pmatrix} 0 & 1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix};\quad P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$$

I. $(N - I)$ is non-singular, so we may change basis so that $t_N = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

II. $P^2 = I \Longrightarrow Pt_P + t_P = 0 \Longrightarrow$

$$\begin{pmatrix} -t_2 \\ -t_1 \\ -t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \Longrightarrow t_1 = t_2.$$

So we have $t_P = \begin{pmatrix} t \\ t \\ t_3 \end{pmatrix}$

III. $P^{-1}NP = N^{-1} \Longrightarrow NP = PN^{-1}$

$$\Longrightarrow Nt_P + t_N = Pt_{N-1} + t_P. \text{ But } t_N = 0$$

$$Nt_P = t_P \Longrightarrow \begin{pmatrix} t \\ -t \\ -t_3 \end{pmatrix} = \begin{pmatrix} t \\ t \\ t_3 \end{pmatrix} \Longrightarrow$$

$2t = 2t_3 = 0 \bmod L$. So we have 3 non-primitive groups, given by $t_P = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$, $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$, and $\begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$.

NEXT: $P\bar{4}m2$. In this case

$$N = -A;\ Q = -P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix}$$

In this case again, $(N - I)$ is non-singular, so we force $t_N = 0$.

$$Q^2 = I \Longrightarrow Qt_Q + t_Q = 0 \Longrightarrow$$

$$\begin{pmatrix} t_2 \\ t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \Longrightarrow$$

$$t_1 = -t_2;\ \ 2t_3 = 0$$

So $t_Q = \begin{pmatrix} t \\ -t \\ t_3/2 \end{pmatrix}$, which amounts to 3 non-primitive groups corresponding to:

$$t_Q = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix};\ \text{or}\ \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ 0 \end{pmatrix};\ \text{or}\ \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

3.3.7 $Z/6\times Z/2$

We'll take $P622$ as the group generated by

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 1 & \\ & & 1 \end{pmatrix}; \quad P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix}$$

(Note: in the tables, the transpose of A is used.)

In $P622$ there are 3 subgroups of index 2, as follows.

1. $gp(A)$
2. $gp(A^2, P)$
3. $gp(A^2, AP)$

By inspection of the tables, it is evident that

- $P6mm = gp < A, -P >$
- $P\bar{6}2m = gp < -A, P >$
- $P\bar{4}m2 = gp < -A, -P >$

Case: $P6mm$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 1 & \\ & & 1 \end{pmatrix}; \quad Q = -P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & 1 \end{pmatrix}.$$

By our previous computations, $t_A = \begin{pmatrix} 0 \\ 0 \\ a/6 \end{pmatrix}$, where $a \in \{0, ..., 5\}$. Let

$t_P = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$.

I. $Q^2 = I \Longrightarrow Qt_Q + t_Q = 0 \Longrightarrow$

$$\begin{pmatrix} -t_2 \\ -t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \Longrightarrow t_1 = t_2.$$

so $t_Q = \begin{pmatrix} t \\ t \\ t_3/2 \end{pmatrix}$, $t_3 = 0, 1.$

II. $AQ = QA^{-1} \Longrightarrow At_Q + t_A = Qt_{A^{-1}} + t_Q$

$$\Longrightarrow \begin{pmatrix} -t \\ 2t \\ t_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -a/6 \end{pmatrix} + \begin{pmatrix} t \\ t \\ t_3 \end{pmatrix}.$$

$$\Longrightarrow t = 0, \text{ and } 2a/6 = 0 \bmod \mathrm{L}$$

or $a = 0, 3$. So we have $t_A = \begin{pmatrix} 0 \\ 0 \\ a/6 \end{pmatrix}$, $a = 0, 3$, $t_Q = \begin{pmatrix} 0 \\ 0 \\ t/2 \end{pmatrix}$, $t = 0, 1.$
This gives 3 extensions of $P6mm$ with non-primitive translations.

Case: $P\bar{6}2m$

Here we must examine

$$N = -A = \begin{pmatrix} 0 & 1 & \\ -1 & -1 & \\ & & -1 \end{pmatrix}; \; P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix}.$$

$(N - I) = \begin{pmatrix} -1 & 1 & \\ -1 & -2 & \\ & & -2 \end{pmatrix}$, which is non-singular, so we may force $t_N = 0$. On the other hand, $P^2 = I \Longrightarrow$

$$Pt_P + t_P = 0 \Longrightarrow \begin{pmatrix} t_2 \\ t_1 \\ -t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0$$

$$\Longrightarrow t_1 = -t_2. \text{ So } t_P = \begin{pmatrix} t \\ -t \\ t_3 \end{pmatrix}$$

$NP = PN^{-1} \Longrightarrow Nt_P + t_N = Pt_{N^{-1}} + t_P$; but $t_N = 0$, so

$$Nt_P = t_P \Longrightarrow \begin{pmatrix} -t \\ 0 \\ -t_3 \end{pmatrix} = \begin{pmatrix} t \\ -t \\ t_3 \end{pmatrix}$$

$$\Longrightarrow t = 0;\ 2t_3 = 0$$

So we have one non-trivial non-primitive form, corresponding to $t_P = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$.

Case: $P\bar{6}m2$

Again, we have $N = -A$, and we can change basis so that $t_N = 0$. This time, though, we have $Q = -P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & 1 \end{pmatrix}$.

$$Qt_q + t_q = 0 \Longrightarrow \begin{pmatrix} -t_2 \\ -t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0$$

$$\Longrightarrow t_1 = t_2;\ 2t_3 = 0;$$

so $t_Q = \begin{pmatrix} t \\ t \\ t_3/2 \end{pmatrix}$, with $t_3 = 0, 1$.

But $NQ = QN^{-1}$, and $t_N = 0, \Longrightarrow$

$$Nt_Q = t_Q \Longrightarrow \begin{pmatrix} t \\ -2t \\ -t_3 \end{pmatrix} = \begin{pmatrix} t \\ t \\ t_3 \end{pmatrix}$$

From this we see that $t_Q = \begin{pmatrix} a/3 \\ a/3 \\ b/2 \end{pmatrix}$, with $a = 0, 1, 2$, and $b = 0, 1$. But I can kill off the $a/3$ part of the translation by conjugating by the vector $Y = \begin{pmatrix} a/3 \\ a/3 \\ 0 \end{pmatrix}$, without changing t_N, since:

$$(N - I)Y = \begin{pmatrix} -1 & 1 & \\ -1 & -2 & \\ & & -2 \end{pmatrix} \begin{pmatrix} a/3 \\ a/3 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -a \\ 0 \end{pmatrix} \in L$$

$$\text{and } (Q - I)Y = \begin{pmatrix} -1 & -1 & \\ -1 & -1 & \\ & & 0 \end{pmatrix} \begin{pmatrix} a/3 \\ a/3 \\ 0 \end{pmatrix} = \begin{pmatrix} -a/3 \\ -a/3 \\ 0 \end{pmatrix}.$$

So in this case, there is only one real possibility.

3.3.8 $Z/2 \oplus Z/2 \rtimes P(3)$

The last case to look at is $P432$, which we think of as being built out of:

$Z/2 \oplus Z/2 \rtimes Z/3 \cong P23 = gp(A, B, Q)$, where

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \; B = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix}; \; Q = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

together with $P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$.

The only non-trivial homomorphism from $P432$ to $Z/2$ sends everything but P to the identity element, and the hom construction for $P432$ thus produces one group:

$$P\bar{4}3m = gp(P23, -P)$$

We have seen that there are only two ways to add non-primitive translations to $P23$. Therefore, we have to look at two cases:

1. $P23 \cong gp(P222, Q)$

2. $P2_13 \cong gp(P2_12_12_1, Q)$

In both cases, $t_Q = 0$ Let $R = -P$, and compute:

$$Rt_R + t_R = \begin{pmatrix} t_2 \\ t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0 \Longrightarrow$$

$$t_1 = -t_2 \text{ and } 2t_3 = 0.$$

$RQ = QR \Longrightarrow$

$$Rt_Q + t_R = Qt_R + t_A \Longrightarrow t_R = Qt_R \Longrightarrow$$

$$\begin{pmatrix} t \\ -t \\ t_3 \end{pmatrix} = \begin{pmatrix} t_3 \\ t \\ -t \end{pmatrix} \Longrightarrow t = t_3;\ 2t = 0.$$

Thus, $t_R = \begin{pmatrix} t/2 \\ -t/2 \\ t/2 \end{pmatrix}$, $t = 0, 1$.

Now in the case (2), t_A, t_B, t_C are all the zero vector; and $t_R = \begin{pmatrix} t/2 \\ -t/2 \\ t/2 \end{pmatrix}$ is consistent with the relations $RA = CR \Longrightarrow Rt_A + t_R = Ct_R + t_C$, etc.

However, in the $P2_12_12_1$ case, we are taking $t_A = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$, $t_B = \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$; and $t_C = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$.

But $Rt_B + t_R = Bt_R + t_B \Longrightarrow$

$$\begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix} + \begin{pmatrix} t/2 \\ -t/2 \\ t/2 \end{pmatrix} = \begin{pmatrix} t/2 \\ t/2 \\ -t/2 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

which is inconsistent.

Summary: $P432$ **has only one extension with non-primitive translations.**

TABLE OF CENTROSYMMETRIC FORMS WITH NON-PRIMITIVE TRANSLATIONS

Point Group G	$gp < G, -I >$	Extensions
$P2$	$P2/m$	$P2_1/m$, $P2/b$, $P2_1/c$
$P3$	$P\bar{3}$	none
$P4$	$P4/m$	$P4_2/m$, $P4/n$, $P4_2/n$
$P6$	$P6/m$	$P6_3/m$
$P222$	$Pmmm$	$Pnnn$, $Pccm$, $Pban$, $Pmma$, $Pnna$, $Pmna$, $Pcca$, $Pbam$, $Pccn$, $Pbcm$, $Pnnm$, $Pmmn$, $Pbcn$, $Pbca$, $Pnma$
$P312$	$P\bar{3}1m$	$P\bar{3}1c$
$P321$	$P\bar{3}m1$	$P\bar{3}c1$
$P422$	$P4/mmm$	$P4/mcc$, $P4/nbm$, $P4/nnc$, $P4/mbm$, $P4/mnc$, $P4/nmm$, $P4/ncc$, $P4_2/mmc$, $P4_2/mcm$, $P4_2/nbc$, $P4_2/nnm$, $P4_2/mbc$, $P4_2/mnm$, $P4_2/nmc$, $P4_2/ncm$
$P622$	$P6/mmm$	$P6/mcc$, $P6_3/mcm$, $P6_3/mmc$
$P23$	$Pm3$	$Pn3$, $Pa3$
$P423$	$Pm3m$	$Pn3n$, $Pm3n$, $Pn3m$

3.4 Attaching Translations to Centro-Symmetric Point Groups

In this section we look at groups of the form $gp < G, -I >$, where G is a point group, and $-I = \begin{pmatrix} -1 & & \\ & -1 & \\ & & -1 \end{pmatrix}$. We want to establish the information in the table.

There are two approaches we could take. On the one hand, using the substitution $t_{-I} \mapsto -(-I-I)Y + t_{-I}$ lets us choose Y so that $t_{-I} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$. We will sometimes use this fact.

On the other hand, we want to build on our previous work and use our knowledge of forms containing non-primitive translations. If we kill t_{-I}, we

move the translational components of the other elements, and have to re-analyze them. So our other approach will be to take a non-split extension, adjoin $-I$, and then determine conditions on t_{-I} that follow from its centrality.

We shall go through the point groups in question, and show that for each form in the second column, there are the required number of extensions. As in the previous section the details of correlating with the names in the tables is largely omitted.

We set $t_{-I} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ once and for all.

3.4.1 $P2, -I$

Form: $P2$

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix} . \, t_A = 0.$$

I. Consistency relations. $A(-I) = (-I)A \Longrightarrow$

$$A \begin{pmatrix} a \\ b \\ c \end{pmatrix} + t_A = -t_A + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} a \\ -b \\ -c \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\Longrightarrow 2b = 2c = 0$$

II. I can kill a by conjugating, since this does not disturb A.

III. So we get $\begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$; $\begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$; $\begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$; $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

But

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix} \sim \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}.$$

$$\text{and } \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$$

IV. So there only two inequivalent cases, given by $t_{-I} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$ or $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

Of course we don't count the latter, since this would give us $P2/m$ back.

This group is $P2/c$

Form: $P2_1$

$t_A = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix}$.

I.

$$A \begin{pmatrix} a \\ b \\ c \end{pmatrix} + t_A = -t_A + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} a \\ -b \\ -c \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\Longrightarrow 2b = 2c = 0$$

II. Kill a, by conjugating.

III. As above, get cases $t_{-I} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$; $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ (which we must count in this case).

These two forms are $P2_1/c$ and $P2_1/m$.

3.4.2 $P3, -I$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & -1 & \\ & & 1 \end{pmatrix}; t_A = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

We want to adjoin $-I, t_{-I}$. We approach this as follows:

I. $gp < P3, -I >= gp < A(-I) >= \{-A, A^2, -I, A, -A^2, I\} \cong Z/6$. So consider

$$-A = \begin{pmatrix} 0 & 1 & \\ -1 & 1 & \\ & & -1 \end{pmatrix}; \; t_{-A} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Since $(-A - I) = \begin{pmatrix} -1 & 1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$ is non-singular, we may kill t_{-A} by conjugation.

This shows that there are no non-split extensions of $P\bar{3}$.

3.4.3 $P4, -I$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix}$$

$P4$

I. $t_A = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

$$A \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Longrightarrow a = -b = b \Longrightarrow t_{-I} = \begin{pmatrix} a/2 \\ a/2 \\ c \end{pmatrix}$$

II. Kill c by conjugation.

III. So we have $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$; $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$

which corresponds to $P4/m$ and $P4/n$.

Form $P4_1$

$t_A = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{4} \end{pmatrix}$

$$A\begin{pmatrix} a \\ b \\ c \end{pmatrix} + t_A = -t_A + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} -b \\ a \\ c \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{4} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\frac{1}{4} \end{pmatrix} + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

This is inconsistent. A similar argument rules out any non-split extensions for $P4_3$, since $t_A = \begin{pmatrix} 0 \\ 0 \\ 3/4 \end{pmatrix}$.

Form $P4_2$

$t_A = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$

We get the same conditions on a, b as above; and kill c as usual by conjugation; getting:

$P4_2/m$; $(t_{-I} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

$P4_2/n$; $(t_{-I} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$

3.4.4 $P6, -I$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 1 & \\ & & 1 \end{pmatrix}$$

I. $A\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Longrightarrow \begin{pmatrix} -b \\ a+b \\ c \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Longrightarrow a = 0 = b.$

We can kill c by conjugating, so this case is settled - there are no non-split extensions, only $P6/m$.

$P6_1, P6_2, P6_3, P6_4, P6_5$

Let $t_A = \begin{pmatrix} 0 \\ 0 \\ a/6 \end{pmatrix}$. Then

$$A \begin{pmatrix} a \\ b \\ c \end{pmatrix} + t_A = -t_A + \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Longrightarrow$$

$$\begin{pmatrix} -b \\ a+b \\ c+a/6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -a/6 \end{pmatrix} + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

This is only consistent when $a = 3$, giving $t_A = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$. $a = b = 0$, and we can kill c as above; so we get the form

$P6_3/m$ corresponding to $t_{-I} = 0$.

3.4.5 $P222, -I$

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \; B = \begin{pmatrix} 1 & & \\ & -1 & \\ & & 1 \end{pmatrix}; \; C = \begin{pmatrix} 1 & & \\ & 1 & \\ & & -1 \end{pmatrix}$$

In this case, whatever t_A, t_B, t_C are, we may easily derive from the various consistency relations that $t_{-I} = \begin{pmatrix} a/2 \\ b/2 \\ c/2 \end{pmatrix}$ where $a, b, c = 0, 1$.

Thus there are 8 possibilities for t_{-I}, and we must examine these in conjunction with 4 forms for $P222$. This turns out to be a bit of a chore, which will occupy the next few pages.

COMPUTING EQUIVALENT EXTENSIONS OF PMMM. There are 32 possibilities, 4 extensions of P222, 8 choices for t_{-I} for each. Each possibility is represented by t_A, t_B, t_{-I}.

Our strategy is to conjugate $t_{-I} \mapsto \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, and see what this conjugation does to t_A, t_B. This provides a representation for each extension of $Pmmm$ in which all the non-primitive translations are determined by t_A, t_B. We do this for $P222$, $P222_1$, $P2_12_12_1$, and $P2_12_12_1$. Then we can see which of these 32 forms are equivalent via the automorphism group $P(3)$ of $P222$.

In the following table, The t_A and t_B vectors have been combined to compactify the information into the table; the vectors along the top are the 8 possibilities for t_{-I}; and the entries in the table are the t_A and t_B that result when a change of basis is performed which kills the t_{-I} vector. (Note: I can never change the 1st coordinate of t_A by conjugating, nor can I change the last coordinate of t_B. In the other cases, killing a $\frac{1}{2}$ in t_{-I} forces a $\frac{1}{2}$ to be added in the corresponding coordinate of t_A, t_B.

As an example, the 5th entry in the $P222$ row is $\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix}$; this are the t_A, t_B vectors that results when $t_{-I} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$ is killed by conjugating by $\begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \\ 0 \end{pmatrix}$.

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \quad B = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix}$$

	t_{-I}						
	$\begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$
t_A t_B							
$P222$							
$\begin{pmatrix} 00 \\ 00 \\ 00 \end{pmatrix}$	$:\begin{pmatrix} 0\frac{1}{2} \\ 00 \\ 00 \end{pmatrix}$	$\begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix}$	$\begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}0 \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ 00 \\ \frac{1}{2}0 \end{pmatrix}$	$\begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix}$
$P222_1$							
$\begin{pmatrix} 00 \\ 00 \\ 0\frac{1}{2} \end{pmatrix}$	$:\begin{pmatrix} 0\frac{1}{2} \\ 00 \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$
$P2_12_12$							
$\begin{pmatrix} 00 \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix}$	$:\begin{pmatrix} 0\frac{1}{2} \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 00 \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$
$P2_12_12_1$							
$\begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix}$	$:\begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$

When are two forms in this table equivalent under an automorphism of the group? Unfortunately it is not visible to the naked eye. If conjugating by P is an automorphism of $\{I, A, B, C\}$, then P must carry t_A to $t_{P^{-1}MP}$. That is, P must permute the associated translations in the same way as it does the matrices.

$$\begin{pmatrix} P & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} M & t_M \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P^{-1} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} PMP^{-1} & Pt_M \\ 0 & 1 \end{pmatrix}$$

Example: to see if $t_A, t_B, t_C \sim t_{A'}, t_{B'}, t_{C'}$ under the automorphism

$P : A \to B \to C$, we

1. permute t_A, t_B, t_C according to P^{-1}, getting t_B, t_C, t_A, then
2. check if $Pt_B = t_{A'}$, $Pt_C = t_{B'}$, and $Pt_A = t_{C'}$

We go through the table, compute the orbit of each form under the action of $P(3)$, keeping only one representative from each orbit.

We use the following permutation matrices:

$$\begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix}; \quad A \leftrightarrow C \begin{pmatrix} 1 & & \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix}; \quad B \leftrightarrow C$$

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad A \leftrightarrow B \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad A \to C \to B$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad A \to B \to C$$

The computation follows; the result is that the following list of 16 forms are the inequivalent extensions of $Pmmm$ with non-primitive translations (the first form is $Pmmm$). We number the forms in the table, in row order, 1 through 32. These are the numbers used below. We also include the crystallographic name, indicating the permutation needed to relate our form to the form in the tables.

1. $\left(\begin{array}{c} 00 \\ 00 \\ 00 \end{array}\right)$ $Pmmm$

2. $\left(\begin{array}{c} 0\frac{1}{2} \\ 00 \\ 00 \end{array}\right)$ $Pccm$

5. $\left(\begin{array}{c} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{array}\right)$ $Pban$

8. $\left(\begin{array}{c} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{array}\right)$ $Pnnn$

9. $\left(\begin{array}{c} 00 \\ 00 \\ 0\frac{1}{2} \end{array}\right)$ $Pmma$, via a 3 cyle permutation

10. $\left(\begin{array}{c} 0\frac{1}{2} \\ 00 \\ 0\frac{1}{2} \end{array}\right)$ $Pmna$

11. $\left(\begin{array}{c} 00 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{array}\right)$ $Pcca$, via a 3 cycle

13. $\left(\begin{array}{c} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{array}\right)$ $Pnna$, via a 3 cycle

17. $\left(\begin{array}{c} 00 \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{array}\right)$ $Pbam$, via a 3-cycle

18. $\left(\begin{array}{c} 0\frac{1}{2} \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{array}\right)$ $Pnnm$, via $A \leftrightarrow B$

19. $\left(\begin{array}{c} 00 \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{array}\right)$ $Pbcm$

21. $\left(\begin{array}{c} 0\frac{1}{2} \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{array}\right)$ $Pbcn$, via a 3 cycle

23. $\begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$ *Pmmn*, via $A \leftrightarrow B$

24. $\begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$ *Pccn* via $A \leftrightarrow B$

25. $\begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix}$ *Pnma*, via $A \leftrightarrow C$

28. $\begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix}$ *Pbca*, via $A \leftrightarrow B$

THE COMPUTATIONS

In each case is shown, first the result of permuting t_A, t_B, t_C according to P (the first two vectors), then the result of applying P^{-1} to the coordinates of each vector. The composite is an equivalent t_A, t_B under the automorphism induced by P.

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ 000 \\ 000 \end{pmatrix} = \#2.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \quad \#4.$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ 00 \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 00 \\ 00 \end{pmatrix} \quad \#2.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix} \quad \#4.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \quad \#4.$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix} \quad \#3.$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2}0 \\ 000 \end{pmatrix} = \#5.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad \#7.$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \quad \#6.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix} \quad \#5.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \quad \#6.$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ 00 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad \#7.$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2}0 \\ \frac{1}{2}0\frac{1}{2} \end{pmatrix} = \#8.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad \#8.$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad \#8.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad \#8.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad \#8.$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad \#8.$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 000 \\ 000 \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#9.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ 00 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ 00 \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#12.$$

$$P : A\,B\,C \rightarrow \begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ 00 \end{pmatrix} \quad -$$

$$P : A\,C\,B \rightarrow \begin{pmatrix} 00 \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ 00 \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ 000 \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#10.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ 00 \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 0\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#15.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 000 \\ \frac{1}{2}\frac{1}{2}0 \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#11.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#14.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ 00 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2}0 \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#13.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#16.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 000 \\ 0\frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#17.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ 00 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad -$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \quad \#17.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{3}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{3}0 \\ \frac{1}{2}0 \\ 00 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#18.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \quad \#18.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad -$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 000 \\ \frac{1}{2}0\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#19.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#20.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad -$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ 00 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 00 \\ 0\frac{1}{2} \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#21.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#22.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad -$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 000 \\ \frac{1}{2}0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2}0 \end{pmatrix} = \#23.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#23.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ 0\frac{1}{2} \end{pmatrix} \quad -$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} 00 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ 00 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} 00 \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ 00 \\ 0\frac{1}{2} \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} 0\frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2}0 \end{pmatrix} = \#24.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#24.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \quad -$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \quad -$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \quad -$$

$$(t_A, t_B, t_C) = \begin{pmatrix} \frac{1}{2}0\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \end{pmatrix} = \#25.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix} \quad \#27.$$

$$P : B \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ 0\frac{1}{2} \end{pmatrix} \quad \#26.$$

$$P : A \leftrightarrow C \rightarrow \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#32.$$

$$P : A\ B\ C \rightarrow \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}0 \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}0 \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#31.$$

$$P : A\ C\ B \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ 0\frac{1}{2} \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \quad \#30.$$

$$(t_A, t_B, t_C) = \begin{pmatrix} \frac{1}{2}0\frac{1}{2} \\ 0\frac{1}{2}\frac{1}{2} \\ \frac{1}{2}\frac{1}{2}0 \end{pmatrix} = \#28.$$

$$P : A \leftrightarrow B \rightarrow \begin{pmatrix} 0\frac{1}{2} \\ \frac{1}{2}0 \\ \frac{1}{2}\frac{1}{2} \end{pmatrix} \overset{P^{-1}}{\rightarrow} \begin{pmatrix} \frac{1}{2}\frac{1}{2} \\ \frac{1}{2}0 \\ 0\frac{1}{2} \end{pmatrix} \quad \#29.$$

And we are done - no more forms left in the table.

3.4.6 $Z/3\times Z/2, -I$

There are two cases to consider:

$P312$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & -1 & \\ & & 1 \end{pmatrix}, ; t_A = \begin{pmatrix} 0 \\ 0 \\ s/3 \end{pmatrix} ; P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix} ; \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

To this set up, we want to add $-I$, $t_{-I} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ We argue as in the discussion of $P\bar{3} = gp(P3, -I)$.

$gp < A, P, -I > \cong gp < -A, P >$. Since $(-A - I)$ is invertible, we may kill t_{-A} by conjugation, and this also kills $t_{-I} = t_{-A^2}$. Thus we only have to worry about

$$t_P = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

I. $P(-I) = (-I)P, \Longrightarrow Pt_{-I} + t_P = -t_P + t_{-I}$

$\Longrightarrow t_P = \begin{pmatrix} t_1/2 \\ t_2/2 \\ t_3/2 \end{pmatrix}$ (since $t_{-I} = 0$)

II. $P^2 = I \Longrightarrow Pt_P + t_P = 0 \Longrightarrow \begin{pmatrix} t_2 \\ t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = 0$

$\Longrightarrow t_1 = -t_2$. So far we have

$$t_P = \begin{pmatrix} t \\ -t \\ t_3 \end{pmatrix} \text{ where each entry is 0 or } \frac{1}{2}$$

III. $P^{-1}AP = A^{-1} \Longrightarrow$

$$AP = PA^{-1} \Longrightarrow At_P + t_A = Pt_{A^{-1}} + t_P$$

$$\begin{pmatrix} -t \\ 2t \\ t_3 \end{pmatrix} + 0 = 0 + \begin{pmatrix} t \\ -t \\ t_3 \end{pmatrix}$$

$$\Longrightarrow t = -t = 2t \Longrightarrow t = 0$$

Thus there is one non-split extension in this case, $P\bar{3}c1$, corresponding to $t_P = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$.

$P312$

$P\bar{3}c1$ is derived from $P312$ in exactly the same way.

3.4.7 $P422, -I$

We must look at

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{pmatrix}; t_A = \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix}; P = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}; t_P = \begin{pmatrix} b/2 \\ b/2 \\ 0 \end{pmatrix}$$

Let $t_{-I} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$

I. $A(-I) = (-I)A \Longrightarrow At_{-I} + t_A = -t_A + t_{-I}$

$$\Longrightarrow \begin{pmatrix} -t_2 \\ t_1 \\ t_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ a/4 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

$$\Longrightarrow 2a/4 = 0 \Longrightarrow a/2 = 0 \Longrightarrow a = 2, 0$$

$$\text{and } t_1 = -t_2 = t_2 \text{ mod L} \Longrightarrow 2t_2 = 0.$$

From $P(-I) = (-I)P$ we also get $2t_3 = 0$. So we have the following cases:

$$t_A = \begin{pmatrix} 0 \\ 0 \\ a/2 \end{pmatrix} \text{ where } a = 0, 1;$$

$$t_P = \begin{pmatrix} b/2 \\ b/2 \\ 0 \end{pmatrix} \text{ where } b = 0, 1;$$

$$t_{-I} = \begin{pmatrix} -t/2 \\ t/2 \\ t_3/2 \end{pmatrix} \text{ where } t = 0, 1;\ t_3 = 0, 1$$

This gives 16 possibilities, one of which is $P422$ itself. All of these occur.

3.4.8 $P622$

$$A = \begin{pmatrix} 0 & -1 & \\ 1 & 1 & \\ & & 1 \end{pmatrix};\ t_A = \begin{pmatrix} 0 \\ 0 \\ s/6 \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 1 & \\ 1 & 0 & \\ & & -1 \end{pmatrix};\ t_P = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Let $t_{-I} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$.

I. $A(-I) = -I(A) \Longrightarrow A \begin{pmatrix} a \\ b \\ c \end{pmatrix} + t_A = -t_A + t_{-I}$

$$\Longrightarrow \begin{pmatrix} b \\ a+b \\ c \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ s/6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -s/6 \end{pmatrix} + \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

Thus $2s/6 = 0 \bmod L \Longrightarrow s = 3$ or 0. Also $a = -b$, but $a + b = b \Longrightarrow a = 0$.

So far we have determined $t_{-I} = \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix}$, $t_A = \begin{pmatrix} 0 \\ 0 \\ s/2 \end{pmatrix}$, $s = 0, 1$.

II. But $P(-I) = (-I)P \Longrightarrow P \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix} \Longrightarrow 2c = 0$. So we get

$$t_A = \begin{pmatrix} 0 \\ 0 \\ s/2 \end{pmatrix};\ s = 0, 1;\ t_{-I} = \begin{pmatrix} 0 \\ 0 \\ c/2 \end{pmatrix};\ c = 0, 1$$

which gives 3 non-split extensions; $P6/mcc, P6_3/mcm, P6_3/mmc$.

3.4.9 $P2_13, -I$

$P2_13 = gp(A, B, Q)$, where

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix};\ t_A = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix};\ B = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix};\ t_B = \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

$$Q = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix};\ t_Q = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

As usual, let $t_{-I} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$

I. $t_{-I} = \begin{pmatrix} a/2 \\ b/2 \\ c/2 \end{pmatrix}$ follows from the fact that $-I$ commutes with A, B, and C.

II. $-I$ commutes with $Q \Longrightarrow Q \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Longrightarrow a = b = c$.

Thus, $t_{-I} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, which is $Pn3$; or

$t_{-I} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$, which is $Pa3$.

3.4.10 $P432, -I$

We must examine

$$A = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}; \; B = \begin{pmatrix} -1 & & \\ & -1 & \\ & & 1 \end{pmatrix};$$

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}; \; Q = \begin{pmatrix} 0 & -1 & \\ -1 & 0 & \\ & & -1 \end{pmatrix}$$

$$-I = \begin{pmatrix} -1 & & \\ & -1 & \\ & & -1 \end{pmatrix}$$

$Pm3m = gp < A, B, P, Q, I >$ and we want to find the non-split extensions of this group. Suppose $t_A, t_B, t_C, t_P, t_Q, t_{-I}$ are all given.

We can kill t_{-I} by conjugation. Then we observe the following facts:

I. $(-I)P = P(-I) \Longrightarrow -t_P = t_P \Longrightarrow t_P = \begin{pmatrix} t_1/2 \\ t_2/2 \\ t_3/2 \end{pmatrix}$.

But $P^3 = I \Longrightarrow t_1 + t_2 + t_3 = 0 \Longrightarrow$ either all three entries are 0, or exactly 2 are $\frac{1}{2}$.

II. $Q(-I) = -I(Q) \Longrightarrow Qt_{-I} + t_Q = -t_Q + t_{-I} \Longrightarrow$

$t_Q = \begin{pmatrix} q_1/2 \\ q_2/2 \\ q_3/2 \end{pmatrix}$.

But $Q^2 = I \Longrightarrow \begin{pmatrix} -q_2 \\ -q_1 \\ -q_3 \end{pmatrix} + \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = 0$

$\Longrightarrow q_1 = q_2$. Thus $t_Q = \begin{pmatrix} q/2 \\ q/2 \\ q_3/2 \end{pmatrix}$.

III. $Q^{-1}PQ = P^2 \Longrightarrow PQ = QP^2 \Longrightarrow$

$Pt_Q + t_P = Qt_{P^2} + t_Q \Longrightarrow$

$$\begin{pmatrix} q_3/2 \\ q/2 \\ q/2 \end{pmatrix} + \begin{pmatrix} t_1/2 \\ t_2/2 \\ t_3/2 \end{pmatrix} = \begin{pmatrix} -t_1/2 \\ -t_3/2 \\ -t_2/2 \end{pmatrix} + \begin{pmatrix} q/2 \\ q/2 \\ q_3/2 \end{pmatrix} \Longrightarrow$$

$q_3/2 = q/2$ and also $t_2 = t_3$.

Thus we have $t_P = \begin{pmatrix} 0 \\ t/2 \\ t/2 \end{pmatrix}$ and $t_Q = \begin{pmatrix} q/2 \\ q/2 \\ q/2 \end{pmatrix}$

IV. Next we examine the action of P and Q on A, B, and C. We have

$P^{-1}AP = B$

$P^{-1}BP = C$

$P^{-1}CP = A$

$Q^{-1}AQ = A$

$Q^{-1}BQ = C$

$Q^{-1}CQ = B$

Now $AP = PB \Longrightarrow At_P + t_A = Pt_B + t_P$ But $At_P = t_P$ and so

$$t_A = Pt_B \Longrightarrow \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} b_3 \\ b_1 \\ b_2 \end{pmatrix}$$

Similarly, we get

$$t_B = Pt_C \Longrightarrow \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} c_3 \\ c_1 \\ c_2 \end{pmatrix}$$

And thus we may write:

$$t_A = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}; \; t_B = \begin{pmatrix} a_2 \\ a_3 \\ a_1 \end{pmatrix}; \; t_C = \begin{pmatrix} a_3 \\ a_1 \\ a_2 \end{pmatrix}.$$

V. Finally, $AQ = QA \Longrightarrow At_Q + t_A = Qt_A + t_Q$. But $At_Q = t_Q$, so $t_A = Qt_A$. Similarly, $t_B = Qt_B$ and $t_C = Qt_C$. From this we see

$a_1 = -a_3$; $a_2 = -a_3$; $a_1 = a_2 = -a_3$. Since each $a_i = 0$ or $\frac{1}{2}$, this means they must all be zero, since, for example:

$At_B + t_A = t_C \Longrightarrow$

$$\begin{pmatrix} a_2 \\ -a_3 \\ -a_1 \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} a_3 \\ a_1 \\ a_2 \end{pmatrix}$$

$\Longrightarrow a_3 = 0.$

3.5 Extensions of Non-primitive Forms

The only task remaining is the computation of extensions of non-primitive forms. We can bootstrap some of our previous work by means of the following theorem.

Theorem 32 *Let P be a primitive point group, B be a non-primitive point group, and suppose $WPW^{-1} = B$, where $W \in GL(3,Q)$. Let $\begin{pmatrix} B & t_B \\ 0 & 1 \end{pmatrix}$ be an extension of B in $R(3)$. Then:*

1. *there is an extension $\begin{pmatrix} P & t_P \\ 0 & 1 \end{pmatrix}$ in $R(3)$ such that*

$$\begin{pmatrix} W & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P & t_P \\ 0 & 1 \end{pmatrix} \begin{pmatrix} W^{-1} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} B & t_B \\ 0 & 1 \end{pmatrix}$$

2. *if*

$$\begin{pmatrix} I & Y \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P & t_P \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I & -Y \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} P & t'_P \\ 0 & 1 \end{pmatrix}$$

then

$$\begin{pmatrix} I & WY \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B & Wt_P \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I & -WY \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} B & Wt'_P \\ 0 & 1 \end{pmatrix}$$

Proof: for 1), take $t_P = W^{-1}t_B$. For 2), note first that $t'_P = -(P-I)Y + t_P$. But conjugating $\begin{pmatrix} B & Wt_P \\ 0 & 1 \end{pmatrix}$ by $\begin{pmatrix} I & WY \\ 0 & 1 \end{pmatrix}$ maps

$$Wt_P \mapsto -(B-I)WY + Wt_P = -W(P-I)Y + Wt_P = Wt'_P.$$

•

Since I have a list of W's which conjugate the primitive forms into non-primitive forms, the theorem tells me that in order to enumerate extensions of B, I need only conjugate the extensions already computed for P, then work with that list.

Since the W's developed in chapter 2 are all integral, and

$$\begin{pmatrix} P & t_P \\ 0 & 1 \end{pmatrix} \mapsto \begin{pmatrix} B & Wt_P \\ 0 & 1 \end{pmatrix}$$

under conjugation by W, we see that this map may actually transform non-primitive translations of P into primitive translations for B.

(Remark: the above theorem would be much easier to state in the language of cohomology: conjugation by W essentially defines a map

$$H^2(P;Z^3) \longrightarrow H^2(B;Z^3) \longrightarrow 0$$

It is the onto-ness of this map which we need.)

NOTE: This theorem does not say that if

$$\begin{pmatrix} C & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P & t_P \\ 0 & 1 \end{pmatrix} \begin{pmatrix} C^{-1} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} P & Ct_P \\ 0 & 1 \end{pmatrix}$$

then

$$\begin{pmatrix} B & Wt_P \\ 0 & 1 \end{pmatrix} \sim_Z \begin{pmatrix} B & WCt_P \\ 0 & 1 \end{pmatrix}$$

Indeed,

$$\begin{pmatrix} WCW^{-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B & Wt_P \\ 0 & 1 \end{pmatrix} \begin{pmatrix} WC^{-1}W^{-1} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} B & WCt_P \\ 0 & 1 \end{pmatrix}$$

But D is not integral, since W^{-1} is not integral.

This means that the list of P's we start with must have separate entries for those groups which are equivalent because of a centralizing matrix. We look carefully at an example to clarify what we mean.

First, we outline the basic computation we must perform.

Outline of Computation

Given: a primitive group $\{ \begin{pmatrix} P_i & t_{P_i} \\ 0 & 1 \end{pmatrix} \}$; and W such that $WP_iW^{-1} = B_i$, B_i a non-primitive form of the point group;

We seek $t_W \in Q^3$ such that

$$\begin{pmatrix} W & t_W \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P & t_P \\ 0 & 1 \end{pmatrix} \begin{pmatrix} W^{-1} & -W^{-1}t_W \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} B & t_B \\ 0 & 1 \end{pmatrix}$$

We have seen that:

$$t_B = -(WPW^{-1} - I)t_W + Wt_P$$

This is the computation; can we simultaneously kill all t_{B_i} by judicious choice of t_W?

The Example of $B2/m$

We have $P2/b$ given by the following data:

$$P_1 = \begin{pmatrix} 1 & & \\ & -1 & \\ & & -1 \end{pmatrix}\ ;\ \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}\ ;\ P_2 = \begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix}\ ;\ \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}\ ;\ P_3 = -$$

$$B_1 = \begin{pmatrix} 1 & 0 & 1 \\ & -1 & 0 \\ & & -1 \end{pmatrix}\ ;\ B_2 = -B_1 = \begin{pmatrix} -1 & 0 & -1 \\ & 1 & 0 \\ & & 1 \end{pmatrix}\ ;\ B_3 = -I$$

and we have

$$W = \begin{pmatrix} 1 & 0 & -1 \\ & 1 & \\ & & 2 \end{pmatrix}\ ;\ WP_iW^{-1} = B_i$$

We want to determine t_{B_i}; we try to kill them, as usual. Now

$$t_{B_1} = -(B_1 - I)t_W + Wt_{P_1} = -\begin{pmatrix} 0 & 0 & 1 \\ & -2 & \\ & & -2 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} + \begin{pmatrix} -\frac{1}{2} \\ 0 \\ 1 \end{pmatrix}$$

$$t_{B_2} = -(B_2 - I)t_W + Wt_{P_2} = -\begin{pmatrix} -2 & 0 & -1 \\ & 0 & \\ & & 0 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} + \begin{pmatrix} -\frac{1}{2} \\ 0 \\ 1 \end{pmatrix}$$

and setting $w_1 = w_2 = 0, w_3 = \frac{1}{2}$ kills both t_{B_i} modulo the lattice. All well and good;

HOWEVER; if $C = \begin{pmatrix} 1 & & \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix}$, then $CP_i = P_iC$ and the t_{P_i} may be replaced by $Ct_{P_i} = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$; this is an equivalent form.

But performing the computations for t_{B_i} again, we get

$$t_{B_2} = -\begin{pmatrix} -2 & 0 & -1 \\ & 0 & \\ & & 0 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$$

and the $\frac{1}{2}$ can obviously not be killed.

Thus we have equivalent forms P, P' such that WPW^{-1}, $WP'W^{-1}$ are <u>not</u> equivalent (over the integers).

Similar considerations apply to cases where translations were made equivalent via <u>automorphisms</u>. That is, if

$$MP_1M^{-1} = P_2; \text{ and } Mt_{P_1} = t_{P_2}$$

where $P_i \in P$, a primitive group; and $WP_iW^{-1} = B_i$; then WMW^{-1} is an automorphism of B_i; but it is not integral, since W^{-1} is not.

We omit the remaining computations. There are 43 more extensions to compute, using the above basic computation as a model. The extensions are listed in the following table.

<u>Non-Primitive Point Group</u>	<u>Extensions</u>
$B2$	none
Bm	Bb
$B2/m$	$B2/b$
$C222$	$C222_1$
$I222$	$I2_12_12_1$
$F222$	none
$Cmm2$	$Cmc2_1, Ccc2$
$Amm2$	$Abm2, Ama2, Aba2$
$Fmm2$	$Fdd2$
$Imm2$	$Iba2, Ima2$

$Cmmm$	$Cmcm, Cmca, Ccem, Cmma, Ccca$
$Fmmm$	$Fddd$
$Immm$	$Ibam, Ibca, Imma$
$I4$	$I4_1$
$I\bar{4}$	none
$I4/m$	$I4_1/a$
$I422$	$I4_122$
$I4mm$	$I4cm, I4_1md, I4_1cd$
$I\bar{4}m2$	$I\bar{4}c2$
$I\bar{4}2m$	$I\bar{4}2d$
$I4/mmm$	$I4/mcm, I4_1/amd, I4_1/acd$
$R3$	none
$R32$	none
$R3m$	R3c
$R\bar{3}m$	$R\bar{3}c$
$I23$	$I2_13$
$F23$	none
$Fm3$	$Fd3$
$Im3$	$Ia3$
$F423$	$F4_132$
$I423$	$I4_132$
$F\bar{4}m3$	$F\bar{4}3c$
$I\bar{4}m3$	$I\bar{4}3d$
$Fm3m$	$Fm3c, Fd3m, Fd3c$
$Im3m$	$Ia3d$

Chapter 4

Automatic Generation of Programs

4.1 Introduction

This chapter is a description of a program generator which takes as input a primitive triclinic, monoclinic, or orthorhombic crystallographic space group, and produces as output Fortran programs which compute the Fourier transform of data exhibiting the given symmetry.

A modest understanding of the crystallographic space groups is clearly necessary - this is discussed in the second section. (Also, see [1].) We then focus attention on the three above mentioned classes, which have the useful property that the three-dimensional fourier transform can be reduced to a sequence of three one-dimensional fourier transforms, by means of the orbit exchange method developed by Myoung Shenefelt in her doctoral dissertation [10]. In the third section we classify the one - dimensional symmetries than can arise. In the fourth section we describe the orbit exchange algorithm.

Once we have these tools in hand, we can describe in detail the program generator itself, which we do in a "step-wise refinement" fashion: first, and most abstractly, purely in terms of the mathematics involved; second, a more detailed specification with algorithms concretely spelled out in pseudo-code; and finally, we include the actual program generator itself, with sample output.

4.2 On Crystallography Groups

We take as our starting point the 230 space groups as they are described in The International Tables of X-ray Crytstallography [6]. Here they are presented as finite groups which act on a unit cell, i.e. a parallelipiped in Euclidean 3-space. Three independent edges of this unit cell span a lattice in R^3, and all arithmetic is done modulo this lattice. As an example, the space group *Pbam* is represented in the tables as follows:

$x, y, z;\ \bar{x}, \bar{y}, z;\ \frac{1}{2}+x, \frac{1}{2}-y, \bar{z};\ \frac{1}{2}-x, \frac{1}{2}+y, \bar{z};$

$\bar{x}, \bar{y}, \bar{z};\ x, y, \bar{z};\ \frac{1}{2}-x, \frac{1}{2}+y, z;\ \frac{1}{2}+x, \frac{1}{2}-y, z;$

The existence of the 230 space groups, though in large part worked out empirically by experimenters over a period of years, is a purely mathematical fact. A purely mathematical classification, based on their solvability, is the subject of the first part of this thesis. It should prove useful in the ongoing project of optimizing various computations that arise in crystallography, notably the Fourier transform.

A basic fact about crystallographic symmetry is that if a rotational symmetry is present, then it may only be rotation by 60 degrees, 90 degrees, 120 degrees, or 180 degrees. If we restrict attention to only those groups with either no rotational symmetry, or at most a two-fold axis of rotation, we get the triclinic, monoclinic, and orthorhombic classes. An examination of the Tables reveals that all these groups act diagonally on the coordinates of the unit cell. That is, if g is an element of one of these groups,

$$g(x, y, z) = (g_1 x, g_2 y, g_3 z) \tag{4.1}$$

where g_i is the action of g restricted to the ith coordinate. This is the key feature of these groups which we exploit in the construction of fourier transform algorithms which exploit crystallographic symmetry.

To be precise, we state the following definitions and theorem.

Definition 18 *Let $G \times X \longrightarrow X$ be a group action. A fundamental domain*

for the action of G on X is a subset $A \subseteq X$ such that:

- $\bigcup_{g \in G} gA = X$
- *if $g_i A \neq g_j A$ then $g_i A \cap g_j A = \phi$*

Notation: Let $FD^G X$ denote a fundamental domain for the action of G on X. (Such a creature is not unique.)

Definition 19 *If a group G acts on a set X, and $x \in X$, then by $Iso(x)$ we mean the isotropy group of x in G, that is, the set, $\{g \in G \mid gx = x\}$.*

Theorem 33 *If G is a group that acts diagonally on a cartesian product $X \times Y$, then we can construct a fundamental domain for this action as follows:*

$$FD^G(X \times Y) = \bigcup_{y \in FD^G Y} FD^{Iso(y)} X \times \{y\} \tag{4.2}$$

We call this an "X-biased" fundamental domain, since it is biased towards using symmetrized fourier transforms along the x coordinate. Note that we may apply this theorem recursively to define an X-biased fundamental domain for the action of a group on $X \times Y \times Z$.

Proof: First we show that this set fills out all of $X \times Y$ under the action of G. To that end, suppose we have (x, y) in $X \times Y$, but not in $FD^G(X \times Y)$. We want to find $(x', y') \in FD^G(X \times Y)$ such that $g(x', y') = (x, y)$.
Case 1: $y \in FD^G Y$.
Then $FD^{Iso(y)} X \times \{y\}$ is a component of $FD^G(X \times Y)$. Also $x \notin FD^{Iso(y)} X$, so $\exists\ x' \in FD^{Iso(y)} X$, and $g \in Iso(y)$, with $gx' = x$. But then $g(x', y) = (gx', gy) = (x, y)$.

Case 2: $y \notin FD^G Y$.
Then $\exists\ g$ such that $y = gy'$, for some $y' \in FD^G Y$. g moves $y' \Rightarrow g \notin Iso(y') \Rightarrow g$ fixes $FD^{Iso(y')} X$. Hence, if $x \in FD^{Iso(y')} X$, $g(x, y') = (x, y)$ will do; but if $y' \in FD^G Y$, and $x \notin FD^{Iso(y')} X$, we are back in Case 1.

Finally, it is easy to see that if $g(x,y) \neq (x,y)$, and $(x,y) \in FD^G(X \times Y)$, then $g(x,y)$ is not in $FD^G(X \times Y)$. •

To discuss the significance of these fundamental domain decompositions, we need the following notion:

Definition 20 *Suppose $f : X \longrightarrow \mathbf{C}$, where $\mathbf{C}$ is the complex numbers. Further suppose that $G \times X \longrightarrow X$ is a group action. By a* symmetrized fourier transform *we mean an algorithm for computing the fourier transform of f which requires as input only the values of f on a fundamental domain for the action of G on X.*

Note: A simple minded symmetrized fourier transform always exists, namely, given the data on a fundamental domain, use the group to fill out all of X, then perform the usual fourier transform. However, we can sometimes do better. For instance, if $X = \mathbf{Z}/\mathbf{N}$, where N is even, and if $f(x) = f(-x)$, then there exist $N/2$ point FFT's which reduce the computation to the points $0, 1, 2, ..., N/2$.

It is precisely the existence of such fast symmetrized one-dimensional algorithms which make our biased fundamental domains useful.

The form of the X-biased fundamental domain given in the theorem above shows that we can begin a fourier transform algorithm by performing symmetrized one-dimensional fourier transforms along the X-coordinate, for each point y in FD^GY . Then, we can re-arrange the data into a Y-biased fundamental domain, and continue the algorithm by performing symmetrized fourier transforms along the Y-coordinate.

To make these remarks and the general idea as transparent as possible, we illustrate the procedure in a simple, 2-dimensional case.

Example

Let $X \times Y = Z/8 \times Z/8$.

Let $G =$ p2; in crystallographic notation, $x, y; -x, -y$.

First we must choose a one-dimensional fundamental domain for the action of G on a coordinate. Let F_X denote the fixed points of the action of G on $Z/8$,

and let M_X denote the points that are moved by G. Then we have:

$$\begin{aligned} FD^G X &= \{0,4\} \cup \{1,2,3\} \\ &= F_X \cup M_X \end{aligned}$$

Now, $Iso(F_X) = G$ and $Iso(M_X) = I$. So applying our formula:

$$FD^G(X \times Y) = \bigcup_{y \in FD^G Y} FD^{Iso(y)} X \times \{y\} \tag{4.3}$$

and grouping the elements of $FD^G Y$ into the subsets F_Y and M_Y, we get:

$$\begin{aligned} FD^G(X \times Y) &= (FD^G X \times F_Y) \cup (FD^I X \times M_Y) \\ &= ((F_X \cup M_X) \times F_Y) \cup (X \times M_Y) \\ &= (F_X \times F_Y) \cup (M_X \times F_Y) \cup (X \times M_Y) \end{aligned}$$

We outline this fundamental domain in the following table, which represents data satisfying the condition $f(x,y) = f(-x,-y)$. Some of the array indices are indicated.

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0							
2,0							
3,0							
4,0							
5,0							
6,0							
7,0							7,7

Now we want to use a row/column approach to the fourier transform, taking advantage of symmetry. The first column of the array in the figure satisfies:

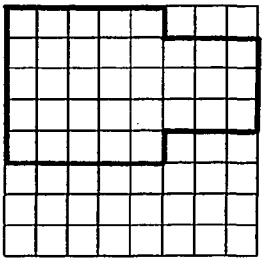
$$f(i,0) = f(-i,0) \tag{4.4}$$

so we use a COSINE transform. Also, the data in the 4th column satisfies:

$$f(i,4) = f(-i,4) \tag{4.5}$$

so again we use a COSINE transform. On columns 1,2,3 we use a 1-dimensional algorithm.

Now we would like to continue our row/column approach by transforming along rows. A row-biased fundamental domain is outlined in the next figure:



Our half-transformed data does not lie on this fundamental domain, and so we must use the group action to permute the data into this new fundamental domain. In this case, the half-transformed data satisfies the same relation, $f(x^*, y) = f(-x^*, y)$, so we may map:

$$(5, 6, 7) \times (5, 6, 7) \longrightarrow (3, 2, 1) \times (3, 2, 1) \qquad (4.6)$$

using the p2 action.

This map from one fundamental domain to another is an "orbit exchange." This example illustrates what is done in all cases:
The Fourier Transform computation is reduced to a fundamental domain for the given space group, and the use of 1-dimensional fourier transform routines is retained.

The general case has some more complications to be treated, including:

1. The fundamental domain decomposition of 3 dimensional sets for more complicated groups.

2. There are two orbit exchange steps, and two intermediate fundamental domains for (possibly) different groups. In other words, the output group is not always identical with the input group.

3. The possible 1-dimensional symmetries that occur are not limited to cases that are treated by existing routines (to my knowledge).

We now proceed to a description of the 10 possible one-dimensional symmetries that do arise in this context.

4.3 One-dimensional symmetries

We introduce some notation now, which will be useful.
We define four basic operators as follows:

$$\begin{aligned} I(x) &= x \\ R(x) &= -x \\ T(x) &= \tfrac{1}{2} + x \\ S(x) &= \tfrac{1}{2} - x \end{aligned}$$

We view these as operating on $\mathbf{Z/N}$, where N is even.

First note that $gp < I, R, S, T > \cong \mathbf{Z/2} \oplus \mathbf{Z/2}$.

The classification of space groups shows that in the case we are interested in, in which the action is diagonal, the action restricted to any coordinate must be a subgroup of this group.

Now let $F = gp < R, S, T >$. Then F and its subgroups account for 5 possible one-dimensional symmetries (including the identity map):

$$\begin{aligned} I: \quad & f(x) = f(x) \\ R: \quad & f(x) = f(-x) \\ S: \quad & f(x) = f(\tfrac{1}{2} - x) \\ T: \quad & f(x) = f(\tfrac{1}{2} + x) \\ F: \quad & f(x) = f(-x) = f(\tfrac{1}{2} - x) = f(\tfrac{1}{2} + x) \end{aligned}$$

But other cases occur, because of the following lemma:

Lemma 7 *Let $\hat{f}$ denote the fourier transform of f. Then:*
If $f(x) = f(\frac{1}{2} + x)$, then $\hat{f}(x) = (-1)^x \hat{f}(x)$.
If $f(x) = f(\frac{1}{2} - x)$, then $\hat{f}(x) = (-1)^x \hat{f}(-x)$.

The factor of (-1) which occurs in the fourier transform of a function with translational symmetry is called a phase factor. It is the existence of the phase relations in partially transformed data that accounts for the other forms of one-dimensional symmetry. The following example should illustrate what happens.

Suppose $f(x, y) = f(\frac{1}{2} + x, -y)$. Then, after we fourier transform along the x coordinate, the transformed data exhibits the following relation:

$$f(x, y) = (-1)^x f(x, -y) \tag{4.7}$$

Thus, for odd values of the index x, when we perform one-dimensional fourier transforms along the y-coordinate, we must use a sine transform.

We summarize the situation in the following lemma.

Lemma 8 *Assume N is divisible by 8. Assume we have data exhibiting the symmetry of one of the primitive triclinic, monoclinic, or orthorhombic crystallographic groups. Then on any biased fundamental domain for partially transformed data, the following ten one-dimensional symmetries may occur:*

$$\begin{aligned}
1. f(x) &= f(-x) \\
2. f(x) &= -f(-x) \\
3. f(x) &= f(\tfrac{1}{2} - x) \\
4. f(x) &= -f(\tfrac{1}{2} - x) \\
5. f(x) &= f(\tfrac{1}{2} + x) \\
6. f(x) &= -f(\tfrac{1}{2} + x) \\
7. f(x) &= f(-x) = f(\tfrac{1}{2} - x) = f(\tfrac{1}{2} + x) \\
8. f(x) &= f(-x) = -f(\tfrac{1}{2} - x) = -f(\tfrac{1}{2} + x) \\
9. f(x) &= -f(-x) = f(\tfrac{1}{2} - x) = -f(\tfrac{1}{2} + x)
\end{aligned}$$

$$10. f(x) \quad = \quad -f(-x) = -f(\tfrac{1}{2} - x) = f(\tfrac{1}{2} + x)$$

We now list the one-dimensional fundamental domains that we have chosen for each of the above symmetries.

For 1., 2.	$\{0, 1, 2, ..., N/2\}$
For 3., 4.	$\{0, 1, 2, ..., N/2 - 1\}$
For 5., 6.	$\{0, 1, 2, ..., N/4\} \cup \{N/2 + 1, ..., 3N/4\}$
For 7. – 10.	$\{0\} \cup \{1, 2, ..., N/4 - 1\} \cup \{N/4\}$

Of the 10 symmetries listed above, I have only been able to find coded versions of 4 - the first, second, fifth, and seventh.

There are two other kinds of symmetry which are important, namely the case where the data is real, and the case where the data exhibits Hermitian symmetry: $f(-x) = \overline{f(x)}$, (the bar denotes complex conjugate). These are related, because if $f(x)$ is real data, $\hat{f}(x)$ exhibits Hermitian symmetry. There do exist optimized Cooley-Tukey type programs which take advantage of these cases. Crystallographic data typically exhibits Hermitian symmetry, because the fourier transform of it is the electron density function, which is real data.

4.4 Orbit Exchange Algorithms

The concept of orbit exchange was developed by Myoung Shenefelt in her doctoral dissertation.

Given a fundamental domain D_1 for a group action on a set X, there is a unique map, $X \longrightarrow D_1$, the "choice" map, which chooses the orbit representative from each orbit. Thus, given any other fundamental domain, D_2 we may compose this choice map with the inclusion map, obtaining:

$$D_2 \longrightarrow X \longrightarrow D_1 \tag{4.8}$$

which is a uniquely defined map between any two fundamental domains.

We may use this fact to permute our data, after performing fourier transforms along one coordinate, from an X-biased fundamental domain to a Y-biased fundamental domain. This is the basic idea of all orbit exchange algorithms.

In the summer of 1988, Dr. Shenefelt and I applied this idea to the class of primitive triclinic, monoclinic, and orthorhombic groups acting diagonally on 3-dimensional space. We were able to completely "solve" all of these groups, in the case of data given on an 8x8x8 array, where by solve is meant complete reduction of the fourier transform to a fundamental domain [9].

Two things became clear in the process of writing these 40 programs.

1. The methods and algorithms would work just as easily, with virtually no change, for an array of data of dimension NxNxN where N is divisible by 8.

2. The passage from crystallographic group to the program seemed as if it should be "automatable".

4.4.1 Motivation to Create the Program Generator

The motivation to create the program generator was fourfold:

1. It would express a uniform control over this class of groups.

2. It would enable me to gain more insight into the concepts involved - a mix of group theory, programming, and crystallography.

3. Once the program generator was finished, it could serve as a prototype model for a more general program generator designed to handle the general orbit exchange situation.

4. There are two main conceptual parts to the program generator. The first part is using the group theory to create the relevant data structures: various fundamental domains, symmetry groups of the data in various stages of transformation, and arrays of phase relations that come into

existence. The second part is the interpretation of these data structures in order to generate the Fortran code which is the output. Once the program generator was written, the second part could be changed to generate code in other high level languages, and even (unoptimized) assembly language.

Thus, the project was undertaken, and successfully completed. It consists of slightly less than 4000 lines of Fortran code. The next section will begin to describe the program generator.

4.5 Description of the Program Generator

The input to the program generator, which we will refer to as *pg*, is one of the primitive triclinic, monoclinic, or orthorhombic groups as listed in the X-ray Tables. The user chooses from the following menu:

1. P1
2. P1bar
3. P2
4. P21
5. Pm
6. Pc
7. P2/m
8. P21/m
9. P2/c
10. P21/c
11. P222
12. P2221
13. P21212
14. P212121
15. Pmm2
16. Pmc21
21. Pmn21
22. Pba2
23. Pna21
24. Pnn2
25. Pmmm
26. Pnnn
27. Pccm
28. Pban
29. Pmma
30. Pnna
31. Pmna
32. Pcca
33. Pbam
34. Pccn
35. Pbcm
36. Pnnm

17. Pcc2	37. Pmmn
18. Pma2	38. Pbcn
19. Pca21	39. Pbca
20. Pnc2	40. Pnma

The selected group is displayed for verification. The group is stored internally in an array whose entries are from the set of operators defined previously: I,R,S, or T. For example, the group Pbam:

$$x, y, z;\ \bar{x}, \bar{y}, z;\ \tfrac{1}{2}+x, \tfrac{1}{2}-y, \bar{z};\ \tfrac{1}{2}-x, \tfrac{1}{2}+y, \bar{z};$$
$$\bar{x}, \bar{y}, \bar{z};\ x, y, \bar{z};\ \tfrac{1}{2}-x, \tfrac{1}{2}+y, z;\ \tfrac{1}{2}+x, \tfrac{1}{2}-y, z;$$

is stored internally as:

I	I	I
R	R	I
T	S	R
S	T	R
R	R	R
I	I	R
S	T	I
T	S	I

We name this group G1, and create three other arrays, G2, G3, and G4 as follows: Let G2 be the array obtained from G1 by replacing any S in column 1 by an R, and any T in column 1 by an I. G3 is obtained from G2 by performing the same operation on column 2 of G2. Finally, G4 is obtained from G3 by performing the same operation on column 3 of G3. These four groups are the symmetry groups of the data at intermediate stages of the fourier transform. Note that they all have the same number of elements. If a subgroup $A \leq G_i$ is represented as a list of rows in the G_i array, then by the *corresponding subgroup* A' *in* G_j we will mean the corresponding list of rows in G_j.

In this context, all the programs to be generated by *pg* have the same basic structure. They all assume that input lies on an X-biased fundamental domain for G1. Then the algorithm may be informally described as follows:

1. Compute fourier transform along x-axis
2. Perform an orbit exchange from the output of step 1 to a Y-biased fundamental domain for G2.
3. Compute fourier transform along y-axis
4. Perform an orbit exchange from the output of step 2 to a z-biased fundamental domain for G3.
5. Compute fourier transform along z-axis
6. Perform an orbit exchange from the output of step 5 to an x-biased fundamental domain for G4.

Remark: the reason for step 6 is that if the initial group has no translations, this step will arrange the data back into the original fundamental domain.

There are two other routines that *pg* must generate, one called expand, the other called expand2. As the names suggest, they take data given on a particular fundamental domain, and use the group to fill out the unit cell appropriately. Expand uses the group G1, and assumes data lies on an X-biased fundamental domain; expand2 uses the group G4, and also assumes data lies on an X-biased fundamental domain.

Thus, the main driving program for all of the groups is exactly the same, and need only be written once. It looks like this:

```
call xformx (x,y)
call oex1 (y,s)
call xformy (s,y)
call oex2 (y,s)
call xformz (s,y)
```

call oex3'(y,t)

Here, x,y,s, and t are NxNxN arrays, and N is a multiple of 8.

Thus, for each group, *pg* must create 8 subroutines: xformx, xformy, xformz, oex1, oex2, oex3, and the two expand routines.

We proceed now to describe the key algorithms and routines of *pg*.

4.6 Algorithm for Fundamental Domain Decomposition

Recall the basic theorem: If G acts diagonally on $X \times Y$, then we may construct an X-biased fundamental domain for this action in terms of fundamental domains for the action of G on the coordinates as follows:

$$FD^G(X \times Y) = \bigcup_{y \in FD^G Y} FD^{Iso(y)} X \times \{y\} \tag{4.9}$$

If G acts on $X \times Y \times Z$, we may write:

$$FD^G(X \times Y \times Z) = FD^{Iso(z)}(X \times Y) \times \{z\} \tag{4.10}$$

and by applying the construction recursively to $FD^{Iso(z)}(X \times Y)$, we get:

$$FD^G(X \times Y \times Z) = \bigcup_{z \in FD^G Z} \bigcup_{y \in FD^{Iso(z)} Y} FD^{Iso(y)} X \times \{y\} \times \{z\} \tag{4.11}$$

From equation 11) we see that we can use a library of 1-dimensional symmetrized fourier transforms along the x coordinate to perform a first round of fourier transforms.

So we now make this fundamental domain decomposition algorithm explicit.

The algorithm may be stated in words as follows:

For each $z \in FD^G Z$

compute Iso(z) $\leq G$

compute $FD^{Iso(z)}Y$

For each $y \in FD^{Iso(z)}Y$

Compute Iso(y) $\leq$ **Iso(z)**

Compute $FD^{Iso(y)}X$

Form $FD^{Iso(y)}X \times \{y\} \times \{z\}$.

To implement this, it is useful to group the elements of one-dimensional fundamental domains into sets of the same isotropy type, and to store this representation in an array. We define the isotropy pieces for the various fundamental domains by the following equations (R,T, and S are the operators as defined above, and F is the group generated by all three of them):

$FD^R X$ = FR $\cup$ MR = {0,N/2} $\cup$ {1,2,...,N/2-1}

$FD^T X$ = MT = {0,1,2,...,N/2-1}

$FD^S X$ = FS $\cup$ MS = {N/4,3N/4} $\cup$ {0,1,...,N/4-1, N/2+1,...,3N/4-1}

$FD^F X$ = M0 $\cup$ M1 $\cup$ M2 = {0} $\cup$ {1,2,...,N/4-1} $\cup$ {N/4}

FR, FS denote the fixed points of R and S respectively. Thus the isotropy group of FR in $gp < R >$ is the entire group. MR denotes the chosen set of orbit representatives which are "moved" by R, i.e. whose isotropy group in $gp < R >$ is the identity. Similarly, the isotropy group of MS in $gp < S >$ is the identity. Finally, Iso(M0) in F is $gp < R >$; Iso(M2) in F is $gp < S >$; and Iso(M1) in F is the identity element. In addition we let NN stand for the one-dimensional fundamental domain for the identity operator.

These symbols are stored in the *pieces* array as follows:

FR	FS	M0	MT	NN
MR	MS	M1	0	0
0	0	M2	0	0

We restate the GETFD algorithm in terms of isotropy pieces:

Determine $FD^G Z$

For each zpiece of $FD^G Z$ **, let A = Iso(zpiece)**

Determine FD^AY

For each ypiece of FD^AY , let B = Iso(ypiece)

Determine FD^BZ

For each zpiece of FD^BZ

Let FD(i,0) = xpiece, FD(i,1) = ypiece, FD(i,2) = zpiece.

The output of this algorithm is an array with 3 columns and a variable number of rows. This is a *fundamental domain* array, one of the key data structures of the program generator. We show a typical example below.

FR	FR	FR
MR	FR	FR
NN	MR	FR
FR	FR	MR
MR	FR	MR
NN	MR	MR

Table 1.

This array is what we need to generate subroutine calls to one-dimensional fourier transform routines. This is explained in the next section. We include here a fragment of *pg* which actually computes a fundamental domain, based on the following data structures: the pieces array, a group array as described above. The group array is scanned, coordinate by coordinate starting with the Z coordinate; the "type" of action of the Z coordinate is determined (I,R,S,T, or F); this type is represented as 0,1,2,3, or 4, and thus can be used as a pointer into the pieces array. The isotropy pieces are then accessed, and their isotropy groups are computed. A subgroup is simply represented as a list of rows in the group array. This subgroup acts on the Y coordinate, and the type of this action is then determined, etc.

4.6.1 Fortran fragment computing a FD array

```
the full GETFD algorithm, a Fortran extract.

The supporting routines are:
```

```
      scan - scans a coordinate of the group array,
             and determines the type of action of the
             given subgroup on that coordinate
             (the type is one of I,R,S,T,or F).
             This type is used as a pointer to the
             pieces array.
      iso  - given a group array, gp, a subgroup,
             and an element in the pieces array
             (pointed to by the type determined by scan),
             and a coordinate, iso determines the isotropy
             subgroup of the given piece. The answer is left
             in the last argument.
      num_pieces - returns the number of pieces in a given
             one-dimensional fundamental domain.

      .
      .
      .
      ztype = scan (gp,ZCOORD,subgroup)
      do 20 i = 0, num_pieces(ztype) - 1
         call iso (gp,pieces(ztype,i),ZCOORD,subgroup,isoz)
         ytype = scan (gp,YCOORD,isoz)
         do 30 j = 0, num_pieces(ytype) - 1
            call iso (gp,pieces(ytype,j),YCOORD,isoz,isoy)
            xtype = scan (gp,XCOORD,isoy)
            do 40 k = 0, num_pieces(xtype) - 1
               output (cptr+k,XCOORD) = pieces(xtype,k)
               output (cptr+k,YCOORD) = pieces(ytype,j)
               output (cptr+k,ZCOORD) = pieces(ztype,i)
40          continue
            cptr = cptr + k
```

```
30        continue
20     continue
       .
       .
       .
```

4.7 Interpreting the FD array

Once we have created a fundamental domain array, it is quite a simple matter to read through it, generating the necessary sequence of subroutine calls to the appropriate one-dimensional symmetrized fourier transform routines. This part of the code generation is controlled from subroutine GENXFORM in *pg.*

For example, the fundamental domain array listed in the previous section, in Table 1., can be written as follows:

$FD^R X$	FR	FR
NN	MR	FR
$FD^R X$	FR	MR
NN	MR	MR

Table 2.

This array may be interpreted row by row as follows: column 1 determines which symmetrized routines to call, and the other two columns are the index sets which are traversed.

For example, since FR = {0,N/2}, the first row may be translated directly into Fortran:

```
      do 10 i = 0,N/2,N/2
      do 10 j = 0,N/2,N/2
         call fteven (x(0,i,j),y(0,i,j))
10    continue.
```

Here, fteven is a name for a cosine fourier transform, and x and y are input and output arrays. At this point it is assumed that the data is stored in an NxNxN array, and data to be transformed is always moved to the first column,

since Fortran stores data column majorant order. This can easily be changed so that the data is stored in a 1-dimensional array, and the accessing of different columns is accomplished by different multi-dimensional stride permutations.

Similarly, since MR = {1,2,...,N/2-1}, the second row may be mapped to the following Fortran code:

```
      do 20 i = 1,N/2-1
      do 20 j = 0,N/2,N/2
         call ft1 (x(0,i,j),y(0,i,j))
20    continue.
```

Here, ft1 is a standard one-dimensional fourier transform.

For the first round of fourier transforms, this is really all there is to the code generation. (There is one programming wrinkle - MS is not a connected set of indices, and thus two sets of doloops are generated for this case. This explains the existence of the doloop labelled "999" in genxform. The same remarks holds for the generation of the expand routines.)

Now the program generator can similarly interpret a Y-biased fundamental domain for the group G2. However, if there is any translational symmetry in the X coordinate of G1 (i.e., a T or an S in column 1 of G1) then the partially transformed data has phase factors of (-1). We must thus apply Lemma 1 to formulate the following check for phase factors:

CHECK FOR PHASE FACTORS:

Let FD be a Y-biased fundamental domain array for G2.
For each row in the FD array
  Let A be the isotropy group of the X and Z pieces
    (A is a subgroup of G2)
  Let A′ be the corresponding subgroup of G1.
  If A′ has any translations in it, then there will be
    phase factors in the data.

Applying Lemma 1, we see that we must generate code of the following general form:

```
      do 10 i = 1,N/2-1
      do 10 j = 0,N/2,N/2
         if (mod(i,2).eq.0)then
            call fteven(x(0,i,j),y(0,i,j))
         else
            call fteven2(x(0,i,j),y(0,i,j))
         endif
10    continue
```

Here, fteven2 is a one-dimensional sine routine.

If we are transforming along the z coordinate, we might have to deal with translational symmetries in X alone, in Y alone, or in both the X and Y coordinates. This results in phase factors of the form $(-1)^{(i)}$, $(-1)^{(j)}$, or $(-1)^{(i+j)}$ respectively.

One further point should be made here about the programming. Generally, FD arrays and group arrays are kept in X-Y-Z order. However, the GENXFORM subroutine permutes the "current" coordinate - the one being transformed - to the first column. This means that there are some supporting routines which perform the necessary re-arrangements, and should help explain some movements of the data that take place.

4.8 Generating the Orbit Exchange Map

The algorithm used by *pg* to generate the orbit exchange map is based on Theorem 2. This theorem guarantees the existence of a map from any fundamental domain to any other, and we know that this map is given by the action of an element in the group. In conjunction with the data structure for a fundamental domain array, we can formulate the following algorithm:

Suppose that FD1 and FD2 are two fundamental domain arrays for the action

of a group stored in a group array G. Then the algorithm for generating the orbit exchange code is:

Algorithm for Orbit Exchange:

```
For each row in FD1
   if the row appears in FD2 then
      next row in FD1 (nothing to be done)
   else
      for each g in G
         if g(row) appears in FD2 then
            use g to implement the orbit exchange
         else
            next g
         endif
      endfor
   endif
endfor
```

The theorem guarantees that this algorithm will succeed, and produce the necessary group element, g.

A brief explanation of the algorithm: rows in FD1 that appear in FD2 do not need to be mapped. If a row in FD1 does not appear in FD2, then we search through the group array until we find an element that maps that row to some row in FD2.

There may be more than one g that does the trick; we do not distinguish between them.

4.9 Summary of the Program Generator

This ends the "tour" of *pg*. In summary, the heart of it is the following three modules, based primarily on the data structure for biased fundamental domains.

- GETFD - calculates fundamental domains
- GENXFORM - reads a FD array, generates xformx, xformy, xformz subroutines.
- GENOEX - generates orbit exchange from one FD array to another.

The rest of the routines are lower level supporting routines, that perform the necessary group functions, and output routines that write lines of fortran code to the output file.

Chapter 5

Program Listings for the Program Generator

TABLE OF CONTENTS OF PROGRAM LISTING

The listing is broken up into 6 modules, as follows:

1) PG

This is the driver module, which gets the choice of group from the user (INPUT), and calls GETFD, GENXFORM, and GENOEX.

2) GETFD

This module handles creating fundamental domains.

3) GENXFORM

This module handles generating fourier transform routines based on a fundamental domain decomposition.

4) GENOEX

This module handles generating orbit exchange subroutines, which permute the data from one fundamental domain into another.

5) INPUT

This module gets the choice of group from the user.

6) PUTS

This module handles the writing of lines of fortran code to the output file.

Each module is prefaced with a list of subroutines and functions that are in it, together with a brief description.

5.1 SAMPLE OUTPUT: PCC2

```
c**********************************
c
c     GROUP is Pcc2
c
c
c
c        x      y      z
c       -x     -y      z
c       -x      y    1/2+z
c        x     -y    1/2+z
c
c**********************************
c          FDX:
c      FR  FR  MT
c      MR  FR  MT
c      NN  MR  MT
      subroutine xformx(x,y)
      implicit none
      include 'dimension.'
      complex x(0:n-1,0:n-1,0:n-1)
      complex y(0:n-1,0:n-1,0:n-1)
      integer i,j,k
      do  10 i = 0,N/2,N/2
      do  10 j = 0,N/2-1
         call fteven (x(0,i,j),y(0,i,j))
 10   continue
      do  30 i = 1,N/2-1
      do  30 j = 0,N/2-1
         call ft1 (n,x(0,i,j),y(0,i,j))
 30   continue
      return
      end
c          FDY:  Y-X-Z order
c      FR  FR  MT
c      MR  FR  MT
c      NN  MR  MT
      subroutine xformy(x,y)
      implicit none
      include 'dimension.'
      complex x(0:n-1,0:n-1,0:n-1)
      complex y(0:n-1,0:n-1,0:n-1)
      integer i,j,k
      do  10 i = 0,N/2,N/2
      do  10 j = 0,N/2-1
         call fteven (x(0,i,j),y(0,i,j))
 10   continue
      do  30 i = 1,N/2-1
      do  30 j = 0,N/2-1
         call ft1 (n,x(0,i,j),y(0,i,j))
 30   continue
```

```
      return
      end
c         FDZ:  Z-X-Y order
c      MT  FR  FR
c      MT  MR  FR
c      MT  FR  MR
c      NN  MR  MR
      subroutine xformz(x,y)
      implicit none
      include 'dimension.'
      complex x(0:n-1,0:n-1,0:n-1)
      complex y(0:n-1,0:n-1,0:n-1)
      integer i,j,k
      do  10 i = 0,N/2,N/2
      do  10 j = 0,N/2,N/2
         call ftahalf (x(0,i,j),y(0,i,j))
10    continue
      do  20 i = 1,N/2-1
      do  20 j = 0,N/2,N/2
         call ftahalf (x(0,i,j),y(0,i,j))
20    continue
      do  30 i = 0,N/2,N/2
      do  30 j = 1,N/2-1
         call ftahalf (x(0,i,j),y(0,i,j))
30    continue
      do  40 i = 1,N/2-1
      do  40 j = 1,N/2-1
         call ft1 (n,x(0,i,j),y(0,i,j))
40    continue
      return
      end
      subroutine oex1(x,y)
      implicit none
      include 'dimension.'
      complex x(0:n-1,0:n-1,0:n-1)
      complex y(0:n-1,0:n-1,0:n-1)
      integer i,j,k
      do  10 i = 0,N/2,N/2
      do  10 j = 0,N/2,N/2
      do  10 k = 0,N/2-1
         y(j,i,k )=x(i,j,k)
10    continue
      do  20 i = 1,N/2-1
      do  20 j = 0,N/2,N/2
      do  20 k = 0,N/2-1
         y(j,i,k )=x(i,j,k)
20    continue
      do  30 i = 0,N/2,N/2
      do  30 j = 1,N/2-1
      do  30 k = 0,N/2-1
         y(j,i,k )=x(i,j,k)
30    continue
```

```
      do  40 i = i,N/2-1
      do  40 j = 1,N/2-1
      do  40 k = 0,N/2-1
         y(j,i,k )=x(i,j,k)
40    continue
      do  50 i = N/2+1,N-1
      do  50 j = 1,N/2-1
      do  50 k = 0,N/2-1
         y(mod(N-j,N),mod(N-i,N),k )=x(i,j,k)
50    continue

      return
      end
      subroutine oex2(x,y)
      implicit none
      include 'dimension.'
      complex x(0:n-1,0:n-1,0:n-1)
      complex y(0:n-1,0:n-1,0:n-1)
      integer i,j,k
      do  10 i = 0,N/2,N/2
      do  10 j = 0,N/2,N/2
      do  10 k = 0,N/2-1
         y(k,i,j ) = x(j,i,k)
10    continue
      do  20 i = 0,N/2,N/2
      do  20 j = 1,N/2-1
      do  20 k = 0,N/2-1
         y(k,i,j ) = x(j,i,k)
20    continue
      do  30 i = 1,N/2-1
      do  30 j = 0,N/2,N/2
      do  30 k = 0,N/2-1
         y(k,i,j ) = x(j,i,k)
30    continue
      do  40 i = 1,N/2-1
      do  40 j = 1,N/2-1
      do  40 k = 0,N/2-1
         y(k,i,j ) = x(j,i,k)
40    continue
      do  50 i = 1,N/2-1
      do  50 j = N/2+1,N-1
      do  50 k = 0,N/2-1
         y(mod(N/2+k,N),i,mod(N-j,N) ) = x(j,i,k)
50    continue
      return
      end
      subroutine oex3(x,y)
      implicit none
      include 'dimension.'

      complex x(0:n-1,0:n-1,0:n-1)
      complex y(0:n-1,0:n-1,0:n-1)
      integer i,j,k
```

```
      do  10 i = 0,N/2,N/2
      do  10 j = 0,N/2,N/2
      do  10 k = 0,N-1
         y(i,j,k ) = x(k,i,j)
10    continue
      do  20 i = 1,N/2-1
      do  20 j = 0,N/2,N/2
      do  20 k = 0,N-1
         y(i,j,k ) = x(k,i,j)
20    continue
      do  30 i = 0,N/2,N/2
      do  30 j = 1,N/2-1
      do  30 k = 0,N-1
         y(i,j,k ) = x(k,i,j)
30    continue
      do  40 i = 1,N/2-1
      do  40 j = 1,N/2-1
      do  40 k = 0,N-1
         y(i,j,k ) = x(k,i,j)
40    continue
      return
      end
      subroutine expand(x)
      implicit none
      include 'dimension.'
      complex x(0:n-1,0:n-1,0:n-1)
      integer i,j,k
      do  10 i = 0,N/2,N/2
      do  10 j = 0,N/2,N/2
      do  10 k = 0,N/2-1
         x(i,j,mod(N/2+k,N)   )=x(i,j,k)
10    continue
      do  20 i = 1,N/2-1
      do  20 j = 0,N/2,N/2
      do  20 k = 0,N/2-1
         x(mod(N-i,N),mod(N-j,N),k )=x(i,j,k)
         x(mod(N-i,N),j,mod(N/2+k,N)   )=x(i,j,k)
         x(i,mod(N-j,N),mod(N/2+k,N)   )=x(i,j,k)
20    continue
      do  30 i = 0,N-1
      do  30 j = 1,N/2-1
      do  30 k = 0,N/2-1
         x(mod(N-i,N),mod(N-j,N),k )=x(i,j,k)
         x(mod(N-i,N),j,mod(N/2+k,N)   )=x(i,j,k)
         x(i,mod(N-j,N),mod(N/2+k,N)   )=x(i,j,k)
30    continue
      return
      end
      subroutine expand2(x)
      implicit none
      include 'dimension.'
      complex x(0:n-1,0:n-1,0:n-1)
```

```
      integer i,j,k
      do  20 i = 1,N/2-1
      do  20 j = 0,N/2,N/2
      do  20 k = 0,N-1
         x(mod(N-i,N),j,k )=x(i,j,k)
20    continue
      do  30 i = 0,N/2,N/2
      do  30 j = 1,N/2-1
      do  30 k = 0,N-1
         x(i,mod(N-j,N),k )=x(i,j,k)
30    continue
      do  40 i = 1,N/2-1
      do  40 j = 1,N/2-1
      do  40 k = 0,N-1
         x(mod(N-i,N),mod(N-j,N),k )=x(i,j,k)
         x(mod(N-i,N),j,k )=x(i,j,k)*(-1)**(k)
         x(i,mod(N-j,N),k )=x(i,j,k)*(-1)**(k)
40    continue
      return
      end
```

5.2 INCLUDE FILES

```
c
c  params.
      integer EOS,YES,NO
      integer II, TT, SS, RR, FF
c
c these numbers are used as indices into the PIECES array;
      parameter (II = 0)   !identity operator
      parameter (RR = 1)   !-x
      parameter (TT = 2)   !1/2 + x
      parameter (SS = 3)   !1/2 - x
      parameter (FF = 4)   !full group = P21
      parameter (EOS = -1)
      parameter (YES = 1)
      parameter (NO =  0)

      integer FR, MR, FS, MS, M0, M1, M2, NN, MT
      parameter (FR = 100)
      parameter (MR = 101)
      parameter (FS = 102)
      parameter (MS = 103)
      parameter (M0 = 104)
      parameter (M1 = 105)
      parameter (M2 = 106)
      parameter (NN = 107)
      parameter (MT = 108)


      integer M3,M4,M5,M6,M7
      parameter (M3=109)
      parameter (M4=110)
      parameter (M5=111)
      parameter (M6=112)
      parameter (M7=113)

      integer SMS,TMT,RMR
      parameter (SMS=114)       !SMS = 1/2 - MS
      parameter (TMT=115)       !TMT = MT + 1/2
      parameter (RMR=116)       ! RMR = -MR

c
      integer FD_RR,FD_FF
      parameter (FD_RR = 109)
      parameter (FD_FF = 110)

c
c file parameters
      integer codefile
      parameter (CODEFILE = 10)
      integer stdout
      parameter (STDOUT = 6)
```

```
c
c  PIECES.
c  the following is in common:
c   the five types of fundamental domains that arise
c    Identity, Rotation, Translation, R*T, P21.

c the file 'params.' must be included before this file.
c
c MAXSIZE governs the size of output array.
      integer MAXSIZE
      parameter (maxsize = 100)

c the pieces array is good for general N
      integer pieces (0:4,0:2)
      data pieces / NN, FR, MT, FS, M0,
     .               0, MR,  0, MS, M1,
     .               0,  0,  0,  0, M2 /

c
      integer xcoord /0/
      integer ycoord /1/
      integer zcoord /2/
c
c
      integer MAX_G
      parameter (MAX_G = 8)
c
c ordg is computed at beginning of program.
      integer ordg
      integer debug

c   ordg will hold the real order of G.
       common pieces,xcoord,ycoord,ordg,debug
```

5.3 MODULE: PG

SUBROUTINES IN PG

```
*******************************
c extinguish: creates a group array with one column of
c      translational symmetries replaced with the
c      symmetry after fourier transform.

       subroutine extinguish (g_in,g_out)
c
c the following routines move columns of arrays
c around, to facilitate bookkeeping

       subroutine y_to_x (g_in,g_out)
       subroutine g1_2 (g_in,g_out)
       subroutine fd1_3 (fd_in,fd_out)
       subroutine gz_to_x (g_in,g_out)
c
c prt_g: prints out a group array

       subroutine prt_g (group,lun)
c
c replace: fdxout is the FD array of fdx after
c   a fourier transform (it might be larger due to
c   translational symmetries being "extinguished"

       subroutine replace (fdx,fdxout)
c
c
c action: gets the quotient of in_g/sub_g
       subroutine action (in_g,sub_g,quo,mod_g)
c
c   strip: a utility routine that strips out duplicate rows;
c       used by subroutine action

       subroutine strip (mod_g,quo)
*******************************
```

```
c*************************************************************
c
c      Program Generator for Orbit Exchange Algorithm;
c
c       The plan of this program is as follows:
c       first generate the data structures needed
c       by the code generator, then generate the code
c
c    1) We need an x-biased fundamental domain to
c       generate the first round of FT's, xformx,
c       as well as the expand routine.
c
c    2) We need the output fundamental domain (x-translations
c       extinguished) to generate the orbit exchange code
c
c    3) We need the Y-biased fundamental domain for
c       G* (G with x-translations extinguished) to
c       generate the second round of FT's; xformy
c
c    4) We need the output fundamental domain (y-translations
c       extinguished) to generate oex2;
c
c    5) We need the Z-biased fundamental domain for G**
c       (G* with y-translations extinguished) to generate
c       the third round of FT's; xformz
c
c    6) We need the output fundamental domain (z-translations
c       extinguished) to generate the final expand routine;
c
c
c*************************************************************
      program pg
      implicit none
      include 'params.'
      include 'pieces.'
c DATA STRUCTURES:  fdx, fdxout, etc.
c x-biased, y-biased, z-biased fundamental domains for
c G, G*, G** respectively;
c and their "fattened" outputs.
c Thus fdxout and fdy are the same up to the orbit exchange.
c These are the arrays that are read by the code generator
c to generate xformx, xformy, xformz, and oex1, oex2, oex3
c
      integer fdx(0:MAXSIZE,0:2), fdxout(0:MAXSIZE,0:2)
      integer fdy(0:MAXSIZE,0:2), fdyout(0:MAXSIZE,0:2)
      integer fdz(0:MAXSIZE,0:2), fdzout(0:MAXSIZE,0:2)
      integer fdout(0:MAXSIZE,0:2)
      integer temp1(0:MAXSIZE,0:2)
c
c G1 is the given group, G2 is G1 with the translations
```

```
c in the X-coord "extinguished"; G3 is G2 with the
c translations in the Y-coord extinguished; G4 has all
c translations extinguished.
c  G1, and fdx generate expandin
c  G4, and fdzout generate expandout (with record of extinctions)
      integer G1(0:MAX_G,0:2)
      integer G2(0:MAX_G,0:2)
      integer G3(0:MAX_G,0:2)
      integer G4(0:MAX_G,0:2)
      integer tempg(0:MAX_G,0:2)
      logical check
      integer sub_g(0:MAX_G)
      integer out_g(0:MAX_G,0:2)
      integer i
      character*20 gpname
      character*20 fname
c
c
c declarations for code generation
      integer gen_xform, num
      integer gen_oex
      integer expandin,expandout
      integer status
      character*1 ok
      integer tflag(0:2)
c
c
c  get the choice of group from the user
      do while (ok.ne.'y'.and.ok.ne.'Y')
         call get_g(G1,gpname)
         call prt_g (G1,STDOUT)
         write (6,1) gpname(1:20)
1        format(1x,'Input group is ',a20,'OK?')
         read (5,5) ok
5        format(a)
      enddo
      write (6,2)
2     format('$enter debug level ==> ')
      read (5,3) debug
3     format(i4)
c
c FIRST: compute number of rows actually in G
      do 10 i = 0,MAX_G
         if (G1(i,0).eq.EOS)then
            go to 20
         endif
10    continue
20    continue
      ordg = i
c*********************************************************
c
c  open up the code file, and the error file, and the log file
```

```
c
c*********************************************************
      write (6,*) 'about to generate xformx'
      read (*,*)
      i = 1
      do while (gpname(i:i).ne.' ')
         i = i+1
      enddo
      fname = gpname(1:i-1)//'.for'

      open (unit=CODEFILE,file=fname,status='new',
     .      carriagecontrol='LIST')

      write(CODEFILE,15)gpname
15    format('c**********************************',/,
     .       'c',/,
     .       'c',5x,'GROUP is ',a20,/,'c',/,'c',/,
     .       'c')
      call prt_g(G1,CODEFILE)
      write(CODEFILE,16)
16    format('c',/,'c**********************************')

c*************************************************************
c
c   GET X-BIASED FD and generate the routine xformx.
c
c*************************************************************
      call getfd (G1,fdx)
      if (.not.check (fdx,G1)) then
         write (6,*) 'fdx doesn''t check'
         call prt_fd (fdx,STDOUT)
         stop
      endif
      write(CODEFILE,150)
150   format('c         FDX:')
      call prt_fd (fdx,CODEFILE)
      status = gen_xform (fdx,1,G1,G1)
c
c  now replace input with output FD;
c  MT --> NN;   FS U MS --> FR U MR;   M0 U M1 U M2 --> FR U MR;
c  after this, X --> FR U MR U -MR.
c  (all this in the X coordinate)
      call replace (fdx,fdxout)
c*************************************************************
c
c   GET Y-BIASED FD and generate the routine xformy
c
c*************************************************************

c first re-arrange G, so that it we have
c the Ycoordinate first,then X, then Z;   XYZ-->YXZ
      call y_to_x(G1,G2)
      call getfd (G2,fdy)
```

```
      if (.not.check (fdy,G2)) then
         write (6,*) 'fdy doesn''t check'
         call prt_fd (fdy,STDOUT)
         stop
      endif
c
c print out fdy as comments
      write(CODEFILE,160)
160   format('c         FDY:  Y-X-Z order')
      call prt_fd (fdy,CODEFILE)
      status = gen_xform (fdy,2,G2,G1)
c
c create output FD - for G*
      call replace (fdy,fdyout)
c straighten out G2,fdy,fdyout
      call fd1_2 (fdyout,temp1)
      call fd_copy (temp1,fdyout)
      call fd1_2 (fdy,temp1)
      call fd_copy (temp1,fdy)
      call g1_2 (G2,tempg)
      call gcopy (tempg,G2)
c*************************************************************
c
c   GET Z-BIASED FD, and generate the routine xformz
c
c*************************************************************
      call gz_to_x(G1,G3)
      call getfd (G3,fdz)
      if (.not.check(fdz,G3)) then
         write (6,*) 'fdz doesn''t check'
         call prt_fd (fdz,STDOUT)
         stop
      endif
c print out fdz as comments
      write(CODEFILE,170)
170   format('c         FDZ:  Z-X-Y order')
      call prt_fd (fdz,CODEFILE)
      status = gen_xform (fdz,3,G3,G1)
c
c create output FD, for G**
      call replace (fdz,fdzout)
c straighten out G3,fdz
      call fd1_3 (fdzout,temp1)      !move 1st col to back
      call fd_copy (temp1,fdzout)
      call fd1_3 (fdz,temp1)
      call fd_copy (temp1,fdz)
      call g1_3 (G3,tempg)
      call gcopy (tempg,G3)
c
c*************************************************************
c
c   GET X-BIASED FD of the OUTPUT.
```

```
c
c*************************************************************
c
c get FD for G***
      call extinguish (G1,G4)
      call getfd (g4,fdout)
c
c
c
      call prt_fd (fdx,STDOUT)
      write (6,*)'fdx'
      read (*,*)

      call prt_fd (fdxout,STDOUT)
      write (6,*)'fdxout'
      read (*,*)

      call prt_fd (fdy,STDOUT)
      write (6,*)'fdy'
      read (*,*)

      call prt_fd (fdyout,STDOUT)
      write (6,*)'fdyout'
      read (*,*)

      call prt_fd (fdz,STDOUT)
      write (6,*)'fdz'
      read (*,*)

      call prt_fd (fdzout,STDOUT)
      write (6,*)'fdzout'
      read (*,*)

c      call prt_g (G1,STDOUT)
c      write (6,*) 'G1'
c      read (*,*)
c      call prt_g (G2,STDOUT)
c      write (6,*) 'G2'
c      read (*,*)
c      call prt_g (G3,STDOUT)
c      write (6,*) 'G3'
c      read (*,*)
c      call prt_g (G4,STDOUT)
c      write (6,*) 'G4'
c      read (*,*)

      write (6,*) 'about to generate oex'
      read (*,*)
c
c the symmetry group on fdxout is on data
c that has been transformed in XCOORD
      status = gen_oex (1,fdxout,fdy,G2,G1)
```

```
      status = gen_oex (2,fdyout,fdz,G3,G1)
      status = gen_oex (3,fdzout,fdout,G4,G1)

      status = expandin (fdx,G1)
c
c it's probably better to compute fd for G4 from scratch -

c less pieces
cc      call getfd (G4,temp1)
c find which coords have translations
      call get_tflag (G1,tflag)

      status = expandout (fdout,G1,G4,tflag)

      close (CODEFILE)



      stop
      end
c*************************************************************
c
c
c       switch  - permute G1 into G2 like this:
c          x,y,z --> y,z,x
c
c               - if the X column has a S, or a T, replace
c                 with R, or I respectively
c
c*************************************************************
      subroutine switch (g_in,g_out)
      implicit none
      include 'params.'
      include 'pieces.'
      integer g_in(0:MAX_G,0:2)
      integer g_out(0:MAX_G,0:2)
      integer i


c
      do 10 i = 0, ordg
         g_out(i,XCOORD) = g_in(i,YCOORD)
         g_out(i,YCOORD) = g_in(i,ZCOORD)
         g_out(i,ZCOORD) = g_in(i,XCOORD)
         if (g_out(i,ZCOORD).eq.SS) then

            g_out(i,ZCOORD) = RR
         else if (g_out(i,ZCOORD).eq.TT) then
            g_out(i,ZCOORD) = II
         endif
10    continue

      return
```

```
      end
c*********************************************************
c
c
c       extinguish (g_in,g_out)
c             replace TT with II, SS with RR in the group
c
c*********************************************************
      subroutine extinguish (g_in,g_out)
      implicit none
      include 'params.'
      include 'pieces.'
      integer g_in(0:MAX_G,0:2)
      integer g_out(0:MAX_G,0:2)
      integer i,j

      do 10 i = 0, ordg
      do 10 j = 0,2
         g_out(i,j) = g_in(i,j)
         if (g_out(i,j).eq.SS) then
            g_out(i,j) = RR
         else if (g_out(i,j).eq.TT) then
            g_out(i,j) = II
         endif
10    continue

      return
      end
c*********************************************************
c

c
c       y_to_x  - permute G1 into G2 like this:
c          x,y,z --> y,x,z
c
c               - if the X column has a S, or a T, replace
c                 with R, or I respectively
c
c       note: could replace this routine with a call to
c             g1_2, followed by call extinguish (out_gp,xcoord)
c
c*********************************************************
      subroutine y_to_x (g_in,g_out)
      implicit none
      include 'params.'
      include 'pieces.'
      integer g_in(0:MAX_G,0:2)
      integer g_out(0:MAX_G,0:2)
      integer i


c
```

```
      do 10 i = 0, ordg
         g_out(i,XCOORD) = g_in(i,YCOORD)
         g_out(i,YCOORD) = g_in(i,XCOORD)
         g_out(i,ZCOORD) = g_in(i,ZCOORD)
         if (g_out(i,YCOORD).eq.SS) then
            g_out(i,YCOORD) = RR
         else if (g_out(i,YCOORD).eq.TT) then
            g_out(i,YCOORD) = II
         endif
10    continue

      return
      end
c*********************************************************
c
c
c       g1_2 - permute G1 into G2 like this:

c          x,y,z --> y,x,z
c
c*********************************************************
      subroutine g1_2 (g_in,g_out)
      implicit none
      include 'params.'
      include 'pieces.'
      integer g_in(0:MAX_G,0:2)
      integer g_out(0:MAX_G,0:2)
      integer i


c
      do 10 i = 0, ordg
         g_out(i,XCOORD) = g_in(i,YCOORD)
         g_out(i,YCOORD) = g_in(i,XCOORD)
         g_out(i,ZCOORD) = g_in(i,ZCOORD)
10    continue
      return

      entry g1_3 (g_in,g_out)
      do 20 i = 0, ordg
         g_out(i,ZCOORD) = g_in(i,XCOORD)
         g_out(i,XCOORD) = g_in(i,YCOORD)
         g_out(i,YCOORD) = g_in(i,ZCOORD)
20    continue

      return

      entry g3_1 (g_in,g_out)
      do 30 i = 0, ordg
         g_out(i,YCOORD) = g_in(i,XCOORD)
         g_out(i,ZCOORD) = g_in(i,YCOORD)
         g_out(i,XCOORD) = g_in(i,ZCOORD)
30    continue
```

```
      return
      end
c*************************************************************
c
c
c       fd1_3  - put fd(X,Y,Z) to fd2(Y,Z,X)
c       fd3_1  - do the inverse
c       fd1_2  - put XYZ to YXZ
c*************************************************************
      subroutine fd1_3 (fd_in,fd_out)
      implicit none
      include 'params.'
      include 'pieces.'
      integer fd_in(0:MAXSIZE,0:2)
      integer fd_out(0:MAXSIZE,0:2)
      integer i


c
      i = 0
      do while (fd_in(i,0).ne.EOS)
         fd_out(i,XCOORD) = fd_in(i,YCOORD)
         fd_out(i,YCOORD) = fd_in(i,ZCOORD)
         fd_out(i,ZCOORD) = fd_in(i,XCOORD)
         i = i+1
      enddo
      fd_out(i,0) = EOS

      return
C
C second entry point
      entry fd3_1 (fd_in,fd_out)
      i = 0
      do while (fd_in(i,0).ne.EOS)
         fd_out(i,XCOORD) = fd_in(i,ZCOORD)
         fd_out(i,YCOORD) = fd_in(i,XCOORD)
         fd_out(i,ZCOORD) = fd_in(i,YCOORD)
         i = i+1
      enddo
      fd_out(i,0) = EOS


      return
c
c next entry   XYZ --> YXZ
      entry fd1_2 (fd_in,fd_out)
      i = 0
      do while (fd_in(i,0).ne.EOS)
         fd_out(i,ZCOORD) = fd_in(i,ZCOORD)
         fd_out(i,YCOORD) = fd_in(i,XCOORD)
         fd_out(i,XCOORD) = fd_in(i,YCOORD)
```

```
         i = i+1
      enddo
      fd_out(i,0) = EOS

      return

c next entry   straight copy
      entry fd_copy (fd_in,fd_out)
      i = 0
      do while (fd_in(i,0).ne.EOS)
         fd_out(i,ZCOORD) = fd_in(i,ZCOORD)
         fd_out(i,XCOORD) = fd_in(i,XCOORD)
         fd_out(i,YCOORD) = fd_in(i,YCOORD)
         i = i+1
      enddo
      fd_out(i,0) = EOS

      return


      end
c*********************************************************
c
c
c       gz_to_x  - permute G1 into G3 like this:
c          x,y,z --> z,x,y
c
c                - if the X column has a S, or a T, replace
c                  with R, or I respectively
c
c*********************************************************
      subroutine gz_to_x (g_in,g_out)
      implicit none
      include 'params.'
      include 'pieces.'
      integer g_in(0:MAX_G,0:2)
      integer g_out(0:MAX_G,0:2)
      integer i


c
      do 10 i = 0, ordg
         g_out(i,XCOORD) = g_in(i,ZCOORD)
         g_out(i,YCOORD) = g_in(i,XCOORD)
         g_out(i,ZCOORD) = g_in(i,YCOORD)
         if (g_out(i,YCOORD).eq.SS) then
            g_out(i,YCOORD) = RR
         else if (g_out(i,YCOORD).eq.TT) then
            g_out(i,YCOORD) = II
         endif

         if (g_out(i,ZCOORD).eq.SS) then
```

```
            g_out(i,ZCOORD) = RR
         else if (g_out(i,ZCOORD).eq.TT) then
            g_out(i,ZCOORD) = II
         endif
10    continue

      return
      end
c*****************************************************************
c
c              prt_g(g) - print out table for G,G*,G**,G***
c
c*****************************************************************
      subroutine prt_g (group,lun)

      implicit none
      include 'params.'
      include 'pieces.'
      integer group(0:MAX_G,0:2)
      integer i,j
      integer lun
      character*1 cchar

      character*7 xsym (0:3) /'   x   ','  -x   ',' 1/2+x ',
     .                        ' 1/2-x '/
      character*7 ysym (0:3) /'   y   ','  -y   ',' 1/2+y ',
     .                        ' 1/2-y '/
      character*7 zsym (0:3) /'   z   ','  -z   ',' 1/2+z ',
     .                        ' 1/2-z '/


      if (lun .eq. STDOUT) then
         cchar = ' '
      else
         cchar = 'c'
      endif
      i = 0
      if(lun.eq.STDOUT)then
         do while (group(i,0).ne.EOS)
            write (lun,200) cchar,xsym(group(i,0)),ysym(group(i,1)),
     .                      zsym(group(i,2))
200         format (1x,a1,5x,3a7)
         i = i+1
         enddo
      else
         do while (group(i,0).ne.EOS)
            write (lun,300) cchar,xsym(group(i,0)),ysym(group(i,1)),
     .                      zsym(group(i,2))
300         format (a1,5x,3a7)
         i = i+1
         enddo
      endif
```

```
c      write (6,*) 'group table: '
c      do 10 i = 0,ordg-1
c         write (6,100) (group(i,j),j=0,2)
c100      format (1x,3i4)
c10    continue
       return
       end
c****************************************
c
c replace: replace MT or NN with NN
c              FS U MS with FR U MR
c              M0,M1,M2 with FR U MR
c
c  this replaces a X column of a FD with
c  what you get when the translations have been extinguished
c
c****************************************
       subroutine replace (fdx,fdxout)
       implicit none
       include 'params.'
       include 'pieces.'
       integer fdx (0:MAXSIZE,0:2)
       integer fdxout (0:MAXSIZE,0:2)
       integer i,j,k


d      write (6,*) 'on entry to replace;'
d      call prt_fd (fdx,STDOUT)

       i = 0
       j = 0
       do while ( fdx(i,XCOORD) .ne. EOS)
          if ( fdx(i,XCOORD) .eq. MT .or. fdx(i,XCOORD) .eq. NN) then
             fdxout(j,XCOORD)   =  NN
             do 10 k = 1,2
                fdxout(j,k) = fdx(i,k)
10           continue
             i = i+1

             j = j+1
          else if ( fdx(i,XCOORD) .eq. FR) then
             do 15 k = 0,2
                fdxout(j,k) = fdx(i,k)
15           continue
             i = i+1
             j = j+1
          else if ( fdx(i,XCOORD) .eq. MR) then
             do 20 k = 0,2
                fdxout(j,k) = fdx(i,k)
20           continue
             i = i+1
             j = j+1
```

```
         else if ( fdx(i,XCOORD) .eq. FS) then
            if ( fdx(i+1,XCOORD) .ne. MS) then
               write (6,*) 'error in replace, MS missing'
               stop
            else
               fdxout(j,XCOORD) = FR
               fdxout(j+1,XCOORD) = MR
               do 30 k = 1,2
                  fdxout(j,k) = fdx(i,k)
                  fdxout(j+1,k) = fdx(i+1,k)
30             continue
               i = i+2
               j = j+2
            endif
c
c note: when replacing M0,M1,M2 with FR,MR,0
c the extra row must have the same
c Y and Z coords as the previous two.
         else if ( fdx(i,XCOORD) .eq. M0) then
            if ( fdx(i+1,XCOORD) .ne. M1 .or.
     .           fdx(i+2,XCOORD) .ne. M2) then
               write (6,*) 'error in replace, M1 missing'
               stop
            else

               fdxout(j,XCOORD)   =  FR
               fdxout(j+1,XCOORD) =  MR
               do 40 k = 1,2
                  fdxout(j,k) = fdx(i,k)
                  fdxout(j+1,k) = fdx(i+1,k)
40             continue
               j = j+2
               i = i+3
            endif

         endif

      enddo
      fdxout(j,XCOORD) = EOS
c
      return
      end
c*************************************************************
c
c   action: computes in_g/sub_g; the 'action group' corresponding
c           to the isotropy group sub_g.
c
c           for each row of in_g, process xcoord,ycoord,zcoord
c              look at each coordinate; put it in a 'canonical
c              form' relative to the subgroup - either
c              identity, or 4+x, or x;
c                 that is:
c
```

```
c                       Z/2 + Z/2 = T,R,RT=S and so
c               Mod T , R,S --> R
c               Mod R , T,S --> T
c               Mod S , T,R --> T
c
c           process: (e.g. xcoord); for i .lt. ordg
c                    cval = in_g(i,xcoord)
c                    could be I,R,S,T
c                        I: mod_g(i,xcoord) = cval
c                        R: if a preceding in_g(i,xcoord) = R
c                              mod_g(i,xcoord) = I
c                           else
c                              mod_g(i,xcoord) = cval
c                        S: if a preceding in_g(i,xcoord) = S
c                              mod_g(i,xcoord) = I
c                           else if a preceding in_g(sub_g(i),0) =
R
c                              mod_g(i,xcoord) = T
c                           else
c                              mod_g(i,xcoord) = cval
c                        T: if a preceding in_g(sub_g(i),xcoord) =
T
c                              mod_g(i,xcoord) = I
c
c                     at this point, mod_g(*,*) contains
c                     the quotient group, with dups;
c                     we must strip out the dups.
      subroutine action (in_g,sub_g,quo,mod_g)
      implicit none
      include 'params.'
      include 'pieces.'
      integer in_g (0:MAX_G,0:2)
      integer sub_g(0:MAX_G)
      integer mod_g(0:MAX_G,0:2)
      integer out_g (0:MAX_G,0:2)
      integer quo (0:MAX_G,0:2)
      integer i,j,k,cval
      integer slen,type,scan


d      write (6,*) 'on entry to action,g is'
d      call prt_g (in_g)

d      write (6,101) (sub_g(i),i = 0,slen(sub_g)-1)
d101   format (1x,'subgroup is:',1x,9i5)

      i = 0
      j = 0
      do while ( i .lt. ordg )
         do 10 j = 0,2

            type = scan (in_g,j,sub_g)
```

```
            mod_g(i,j) = in_g (i,j)
d            write (6,123)i,j,mod_g(i,j)
d123         format(1x,'in action i, j, mod_g(i,j) = ',3i5)
c CASE I:
c CASE R:(i.e. mod_g(i,j) = R
            if (mod_g(i,j) .eq. RR) then
               if (type .eq. FF) then
                  mod_g(i,j) = II
               else if (type .eq. RR) then
                  mod_g(i,j) = II
               else if (type .eq. TT) then
                  mod_g(i,j) = RR
               else if (type .eq. SS) then
                  mod_g(i,j) = TT
               endif
c CASE S:
            else if (mod_g(i,j) .eq. SS) then
               if (type .eq. FF) then
                  mod_g(i,j) = II
               else if (type .eq. SS) then
                  mod_g(i,j) = II
               else if (type .eq. RR) then
                  mod_g(i,j) = TT
               else if (type .eq. TT) then
                  mod_g(i,j) = RR
               endif
c CASE T:
            else if (mod_g(i,j) .eq. TT) then
               if (type .eq. FF) then
                  mod_g(i,j) = II
               else if (type .eq. TT) then
                  mod_g(i,j) = II
               else if (type .eq. RR) then
                  mod_g(i,j) = TT
               else if (type .eq. SS) then
                  mod_g(i,j) = TT
               endif

            endif
10        continue
          i = i+1
      enddo
      mod_g(i,0) = EOS
d      call prt_g(mod_g)
d      write(6,*) 'mod_g'

c       at this point, mod_g(*,*) contains
c       the quotient group, with dups;
c        we must strip out the dups.
      call strip (mod_g,quo)
d      call prt_g(quo)
d      write(6,*) 'quo, the quotient'
c
```

```
c check: is (# row in quo)*slen(sub_g)-1 = ordg?
      return
      end
c*****************************************************************
c
c
c
c     strip ; remove dups from g_array created in action
c
c*****************************************************************
      subroutine strip (mod_g,quo)
      include 'params.'
      include 'pieces.'
      integer mod_g(0:MAX_G,0:2)
      integer quo(0:MAX_G,0:2)
      integer i,j
      logical eqflg

c
c copy first row.
      do 10 n = 0,2
         quo(0,n) = mod_g(0,n)
10    continue

      i = 1
      j = 1
      do while (i .lt. ordg)
         eqflg = .FALSE.
         do 20 k = 0,i-1
            if (mod_g(k,0) .eq. mod_g(i,0) .and.
     .           mod_g(k,1) .eq. mod_g(i,1) .and.
     .           mod_g(k,2) .eq. mod_g(i,2) ) then
               eqflg = .TRUE.
            endif
20       continue
c
c if current row of mod_g was none of the preceding rows
c then copy it; else it has already been copied.
         if (eqflg .eq. .FALSE.) then
            do 30 n = 0,2
               quo(j,n) = mod_g(i,n)
30          continue
            j = j+1
         endif
         i = i+1
      end do
c
d      if (j.lt.ordg-1) then
d         write(6,*)'in strip: action group not whole group'
d         quo(j,0) = EOS
d      endif
      quo(j,0) = EOS
```

```
      return
      end
c*********************************************************
c
c      call get_tflag (G1,tflag)
c         find which coords have translations
c
c*********************************************************

      subroutine get_tflag(G1,tflag)
      implicit none
      include 'params.'
      include 'pieces.'

      integer G1 (0:MAX_G,0:2)


      integer tflag (0:2)
      integer scan
      integer type,i
      integer subgp(0:MAX_G)

      do 5 i = 0,ordg-1
         subgp(i) = i
5     continue
      subgp(i) = EOS
      do 10 i = 0,2

         type = scan(G1,i,subgp)
         tflag(i) = NO
         if (type .eq. FF. or. type .eq. SS .or. type .eq. TT) then
            tflag(i) = YES
         endif

10    continue

      return
      end
```

5.4 MODULE: GETFD

```
                SUBROUTINES AND FUNCTIONS IN GETFD

*******************************
c
c getfd: routine which generates a fundamental domain

      subroutine getfd (g_array,output)
c
c prt_fd: prints a fundamental domain

      subroutine prt_fd (FD,lun)
c
c gcopy: copies one group array to another

      subroutine gcopy (g_in,g_out)
c
c iso: returns an isotropy subgroup

      integer function iso (gp,piece,coord,group,isogroup)
c fixes: answers question does given operation fix a given piece?

      logical function fixes (oper,piece)
c scan: returns the type of action a given subgroup of a given
c       group has along a given coordinate

      integer function scan (gp,coord,subgroup)
c num_pieces: how many pieces in a fundamental domain of given type?
      integer function num_pieces(type)
c sizeof: how many elements in a given row of a FD array?
      integer function sizeof(FD,rownum)
c iso_row: what is the isotropy subgroup of a given row of a
c          FD array

      integer function iso_row (rownum,FD,gp,subgp)
c check: checks that a given fundamental domain has the right number
c  of elements
      logical function check (fd,gp)
*******************************
```

```
c*****************************************************************
c
c     GETFD (g_array,output)
c
c       Dec 3, 1988
c       Get fundamental domain for 3 dimensional space group
c       this routine gets an X-biased FD for the group in
c       G, leaves it in FDX
c       G is in common
c
c*****************************************************************
      subroutine getfd (g_array,output)
      implicit none
      integer i,j,k
c
      include 'params.'
      include 'pieces.'
      integer g_array (0:MAX_G,0:2)
      integer output (0:MAXSIZE,0:2)
      integer cptr      !current pointer to output array
      integer iso
      integer scan

      integer type

      integer xtype,ytype,ztype
      integer isoz(0:MAX_G)
      integer isoy(0:MAX_G)
      integer subgroup(0:MAX_G)
      integer num_pieces
      integer gp(0:MAX_G,0:2)

      call gcopy (g_array,gp)
c let subgroup be G restricted to Z
      do 10 i = 0,ordg-1
         subgroup(i) = i
10    continue
      subgroup(i) = EOS

c initialize pointer to FD array
      cptr = 0

c get type of Group restricted to Z: II,RR,TT,SS, or FF (full)
      ztype = scan (gp,ZCOORD,subgroup)
      do 20 i = 0, num_pieces(ztype) - 1
         call iso (gp,pieces(ztype,i),ZCOORD,subgroup,isoz)
         ytype = scan (gp,YCOORD,isoz)
         do 30 j = 0, num_pieces(ytype) - 1
            call iso (gp,pieces(ytype,j),YCOORD,isoz,isoy)
            xtype = scan (gp,XCOORD,isoy)
            do 40 k = 0, num_pieces(xtype) - 1
               output (cptr+k,XCOORD) = pieces(xtype,k)
```

```
               output (cptr+k,YCOORD) = pieces(ytype,j)
               output (cptr+k,ZCOORD) = pieces(ztype,i)
40          continue
            cptr = cptr + k
30       continue
20    continue

c now put in a terminator.
      output (cptr, XCOORD) = EOS
      return
      end
c*****************************************************
c
c  iso (gp,piece,coord,group,isogroup)
c
c     on the given coordinate the given group acts;
c     for the given piece, this routine
c     determines the isotropy subgroup

c
c     routine scans the given column of gp,
c     only checking those elements in group,
c     and sees which ones fix point.
c
c*****************************************************
      integer function iso (gp,piece,coord,group,isogroup)
      implicit none
      include 'params.'
      include 'pieces.'
      integer i,k
      integer piece
      integer coord
      integer group (0:MAX_G), isogroup (0:MAX_G)
      integer gp(0:MAX_G,0:2)
      logical fixes

d      write (6,321)piece
d321   format (1x,'in iso: piece = ',i4)

      k = 0
      i = 0
      do while (group(i) .ne. EOS .and. i .lt. ordg)
         if (fixes (gp(group(i),coord),piece)) then
            isogroup(k) = group(i)
            k = k+1
         endif
         i = i+1
      end do
      isogroup(k) = EOS
d      write (6,100)group,isogroup
d100   format (1x,'in iso:, group, isogroup are; ',
d     .        9i2,/,31x,9i2)
      return
```

```
      end
c*************************************************************8
c
c

c   logical function fixes (oper,piece)
c
c    routine checks if the given operation fixes the
c       given piece
c
c*************************************************************

      logical function fixes (oper,piece)
      implicit none
      include 'params.'
      integer oper
      integer piece
      fixes = .true.

c IDENTITY
      if (oper .eq. II) return
c TRANSLATION
      if (oper.eq.TT) then
         fixes = .false.
         return
      endif
c ROTATION
      if (oper .eq. RR) then
         if (piece .eq. FR .or. piece .eq. M0) then
            return
         else
            fixes = .false.   !case is -1 * movable piece
         endif
      endif
c TR * ROT is only case left
      if (piece .eq. FS .or. piece .eq. M2) then
         return
      else
         fixes = .false.
         return
      endif

      return

      end
c*************************************************************
c
c
c      integer function scan (gp,coord,subgroup)
c
c            for given coordinate and subgroup, returns
c             type of fundamental domain
c
```

```
c     subgroup: a 0 terminated list of pointers to rows of G
c
c*****************************************************************
      integer function scan (gp,coord,subgroup)
      implicit none
      include 'params.'
      include 'pieces.'
      integer coord,subgroup(0:MAX_G)
      integer type
      integer gp(0:MAX_G,0:2)

      integer i, irot, itrans, itrot

d      write (6,11) (subgroup(i),i=0,ordg),coord
d11    format (1x,'entering scan: subgroup,coord are; '/,
d     . 9(i3),/,i3)


      i = 0
      irot = 0
      itrans = 0
      itrot = 0
      do while ( subgroup(i) .ne. EOS .and. i .lt. ordg)

d         write (6,999) i, g(subgroup(i),coord)
d999      format (1x, 'i = ',i3, 'g(subgroup(i),coord) = ',i3)

         if (gp(subgroup(i),coord) .eq. RR) then
            irot = 1

         else if (gp(subgroup(i),coord) .eq. TT) then
            itrans = 1
         else if (gp(subgroup(i),coord) .eq. SS) then
            itrot = 1
         endif
         i = i+1
      enddo
c
      type = irot + 2*itrans + 4*itrot
c
c  0 = ID; 1 = ROT; 2 = TRANS; 3 = TR*ROT; 4 = P21
c
c  fix up type to be between 0 and 4
      if (type .eq. 4) type = 3
      if (type .eq. 7) type = 4

      scan = type
      return
      end
c*****************************************************************
c
c
c
```

```
c*********************************************************************
      integer function num_pieces(type)
      implicit none
      integer type
      include 'params.'
      include 'pieces.'

      if (type.eq.II.or.type.eq.TT)then
         num_pieces = 1
      else if (type.eq.FF)then
         num_pieces = 3
      else if (type.eq.RR.or.type.eq.SS)then
         num_pieces = 2
      else
         write(6,100)type
100      format (1x,'error in num_pieces, type = ',i3)
      endif

d      write (6,200) num_pieces
d200   format(1x,'in num_pieces,  = ',i3)
      return
      end
c*********************************************************************
c
c     sizeof (row); gets size of row of a FD (assuming
c     unit cell is 8x8x8. This is used in checking the FD
c     so this assumption is OK.
c
c  WARNING: this assumes the FD has no pieces of the form -MR.
c
c*********************************************************************
      integer function sizeof(FD,rownum)
      implicit none
      integer rownum

      include 'params.'
      include 'pieces.'
      integer FD (0:MAXSIZE,0:2)
      integer i

      sizeof = 1
      do 10 i = 0,2
         if (FD (rownum,i).eq.FR.or.FD (rownum,i).eq.FS)then
            sizeof = sizeof * 2
         else if (FD (rownum,i).eq.NN)then
            sizeof = sizeof * 8
         else if (FD (rownum,i).eq.MR.or.FD (rownum,i).eq.MS)then
            sizeof = sizeof * 3
         else if (FD (rownum,i).eq.M0.or.FD (rownum,i).eq.M2)then
            sizeof = sizeof * 1
         else if (FD (rownum,i).eq.M1) then
            sizeof = sizeof * 1
```

```
         else if (FD (rownum,i).eq.MT) then
            sizeof = sizeof * 4
         else
            write(6,100)FD (rownum,i)
100         format (1x,'error in sizeof, FD (rownum,i) = ',i3)
         endif
10     continue

d       write (6,200) sizeof
d200    format(1x,'in sizeof  = ',i3)
       return
       end
c*********************************************************************
c
c
c
c*********************************************************************
       integer function slen (subgp)
       implicit none
       include 'params.'
       integer subgp(0:*)
       integer i
       slen = 1
       i = 1
       do while (subgp(i).ne.EOS)
          slen = slen+1
          i = i+1
       enddo
       return
       end
c*********************************************************************
c
c
c      iso_row: input: rownum,fd,gp,subpg
c
c               looks at a given row of FD;
c               puts isotropy of that row in subgp
c               returns number of elements in isotropy subgp
c

c
c*********************************************************************
       integer function iso_row (rownum,FD,gp,subgp)
       implicit none
       include 'params.'
       include 'pieces.'
       integer FD(0:MAXSIZE,0:2)
       integer rownum
       integer gp(0:MAX_G,0:2)
       integer subgp(0:*)
       integer i,j
       logical fixes
```

```
      iso_row = 0
      j = 0
      do 10 i = 0,ordg-1        !for each element of G

         if (fixes (gp(i,0), FD(rownum,0)) .and.
     .       fixes (gp(i,1), FD(rownum,1)) .and.
     .       fixes (gp(i,2), FD(rownum,2)) ) then

            subgp (j) = i
            j = j + 1
            iso_row = iso_row + 1
         endif

10    continue
      subgp(j) = EOS

d      write (6,100)iso_row
d100   format (1x,'in iso_row, size of iso_row = ',i4)
d      write (6,200)(subgp(i),i=0,j)
d200   format (1x,'in iso_row, subgp =',9i4)
      return
      end
c*****************************************************************
c
c
c    CHECK(FD,gp) : checks that the fd is a fd for the gp
c                   that G/Iso(piece) * size(piece) = 512
c                   when summed over isotropy pieces
c
c
c*****************************************************************
      logical function check (fd,gp)
      implicit none
      include 'params.'
      include 'pieces.'
      integer FD(0:MAXSIZE,0:2)
      integer total, rownum, rowtot, sizeof, iso_row
      integer subgp(0:MAX_G)
      integer gp(0:MAX_G,0:2)
      integer n,m

      check = .true.
      total = 0
      rownum = 0
      do while ( FD(rownum,0) .ne. EOS)
d        write (6,100) rownum, FD(rownum,0)
d100     format (1x,'in check: row#,xvalue ',i2,1x,i4)
         n = iso_row (rownum,FD,gp,subgp)
         m = sizeof (FD, rownum)
         rowtot = m * (ordg/n)
```

```
d          write (6,121)rowtot
d121       format (1x,'rowtot = ',i4)
          total = total + rowtot
d          write (6,122)total
d122       format (1x,'total = ',i4)
          rownum = rownum + 1
       enddo

       if (total .ne. 512) then
          write (6,10)total
10        format (1x, 'error in check, total = ',i3)
          check = .false.

       endif
       return
       end
c*********************************************************************
c
c
c     PRT_FD :
c
c
c*********************************************************************
       subroutine prt_fd (FD,lun)
       implicit none
       include 'params.'
       include 'pieces.'
       integer FD(0:MAXSIZE,0:2)
       integer i,j,kk
       integer row(0:2)
       integer lun
       character*9 char (0:16) /' FR ',' MR ',' FS ',' MS ',
      .                         ' M0 ',' M1 ',' M2 ',
      .                         ' NN ',' MT ',' M3 ',' M4 ',
      .                         ' M5 ',' M6 ',' M7 ','SMS ',
      .                         'TMT ','-MR'/
       character*1 cchar

c
c character version
       if (lun .eq. CODEFILE)then
          cchar = 'c'
       else
          cchar = ' '
       endif

       if (lun.eq.STDOUT)then
          i = 0
          do while (FD(i,0) .ne. EOS)
             do 10 kk = 0,2
                row(kk) = FD(i,kk)

                if (row(kk) .eq. -MR)then
```

```
                  write (6,*)' that pesky -MR'
                  stop
c                  row(kk) = MT + 1      !install at end of char array
               endif
10          continue
            write (lun,999) cchar,(char( row(j)-100 ),j=0,2)
999         format (1x,a1,5x,3a4)
            i = i+1
         enddo
      else
         i = 0
         do while (FD(i,0) .ne. EOS)
            do 20 kk = 0,2
               row(kk) = FD(i,kk)
               if (row(kk) .eq. -MR)then
                  write (6,*)' that pesky -MR'
                  stop
c                  row(kk) = MT + 1      !install at end of char array
               endif
20          continue
            write (lun,1000) cchar,(char( row(j)-100 ),j=0,2)
1000        format (a1,5x,3a4)
            i = i+1
         enddo
      endif


c the following code prints out the array with
c the integer token values
d      i = 0
d      do while (FD(i,0) .ne. EOS)
d         write (6,200) (FD (i,j),j=0,2)
d200      format (1x,3i4)
d         i = i+1
d      enddo


      return
      end

c*********************************************************************
c
c
c    GCOPY - copy one g array to another
c
c
c*********************************************************************
      subroutine gcopy (g_in,g_out)
      implicit none
      include 'params.'
      include 'pieces.'
```

```
      integer g_in (0:MAX_G,0:2)
      integer g_out (0:MAX_G,0:2)
      integer i,j

      i = 0
      do while (g_in(i,0).ne.EOS)
         do 10 j = 0,2
            g_out (i,j) = g_in(i,j)
10       continue
         i = i+1
      enddo
c
c copy terminator row
      do 20 j = 0,2
         g_out (i,j) = g_in(i,j)
20    continue
      return
      end
```

5.5 MODULE: GENXFORM

```
                SUBROUTINES AND FUNCTIONS IN GENXFORM

*******************************
c
c
c
c The following routines put out lines in the
c various xform routines:

      subroutine put_do (lun,label,indx,piece)
      subroutine put_line (lun,row,phase)
c
c gen_xform is routine which generates xformx,xformy,xformz

      integer function gen_xform (fd,num,gp,G1)
c
c expandin,expandout generate the symmetry expand routines
c for input and output fundamental domains

      integer function expandin (fdx,G1)
      integer function expandout (fd,G1,G4,tflag)
c
c iso_yz counts elements which fix a given row in the
c y and z coordinates; used by chk_for_trans.

      integer function iso_yz (rownum,FD,gp,subgp)
c
c chk_for_trans checks for translations; needed
c to determine which symmetrized routine to call,
c  for example, a cosine (fteven1) vs. a sine (fteven2)

      integer function chk_for_trans (rownum,FD,gp,G1,
*******************************
```

```
c*************************************************************
c
c
c  GENXFORM.FOR
c contains gen_xform which generates
c xformx, xformy, xformz,
c        also contains routines to generate:
c
c  fd is the fundamental domain fdx,fdy, or fdz
c  num is the round number - 1 2 or 3; which corresponds to
c                            x y or z coordinate;
c
c   if it's y or z, we must handle possible extinctions due
c   to translational symmetry in previously transformed
c   coordinates.
c*************************************************************
      integer function gen_xform (fd,num,gp,G1)
      implicit none
      include 'params.'
      include 'pieces.'
      integer fd(0:MAXSIZE,0:2)
      integer gp(0:MAX_G,0:2)
      integer G1(0:MAX_G,0:2)
      integer status,label,i,lun
      integer num
      integer chk_for_trans,xy_flg
      integer mscount,ijk
      character*2 ft_type
      character*30 subs (0:2) /'      subroutine xformx(n,x,y)',
     .                         '      subroutine xformy(n,x,y)',
     .                         '      subroutine xformz(n,x,y)'/

      lun = CODEFILE

      write (lun,100) subs(num-1)
100   format(a30)
      call put_head (lun)
      i = 0
      do while (fd(i,0).ne.EOS)
c
c the first x piece determines the symmetrized ft that we use.
         if (fd(i,XCOORD).eq.MR .or. fd(i,XCOORD) .eq. MS .or.
     .       fd(i,XCOORD).eq.M1 .or. fd(i,XCOORD) .eq. M2) then
            go to 25
         endif
c
c the following construction - the 999 loop -
c is necessary to deal
c with the case of MS (0,1,5). In this case, we execute
c do the put_do's in a loop, fixing up the pieces in
```

```
c an ad hoc fashion.
         if (fd(i,YCOORD).eq.MS.or.fd(i,ZCOORD).eq.MS) then
            if (fd(i,YCOORD).eq.fd(i,ZCOORD))then
               write (*,*) 'something horrible occurred'
               stop
            endif
            mscount = 1
         else
            mscount = 0
         endif

         do 999 ijk = 0,mscount
            label = (i+1)*10
            if (ijk .eq. 1) then
               label = label + 5
               if (fd(i,YCOORD).eq.MS)then
                  fd(i,YCOORD) = -MS
               else
                  fd(i,ZCOORD) = -MS
               endif
            endif

c print out do statements
            call put_do (lun,label,'i',fd(i,YCOORD))
            call put_do (lun,label,'j',fd(i,ZCOORD))

c restore fd(i,1:2)
            if (ijk .eq. 1) then
               if (fd(i,YCOORD).eq.-MS)then
                  fd(i,YCOORD) = MS
               else
                  fd(i,ZCOORD) = MS
               endif
            endif
c print out call statements
            if (chk_for_trans(i,fd,gp,G1,num,ft_type,xy_flg) .eq.
                  NO) then
               call put_call (lun,fd(i,XCOORD))
               call put_cont (lun,label)
            else
               call put_if(lun,num,xy_flg)
               call put_call (lun,fd(i,XCOORD))
               call put_else(lun)
               call put_call2 (lun,fd(i,XCOORD),ft_type)
               call put_endif(lun)
               call put_cont (lun,label)
            endif
999      continue

25       continue
         i = i+1

      enddo
```

```
c print out return/end.
      call put_end(lun)
      return
      end
c*************************************************************
c
c
c   subroutine put_do (lun,label,indx,piece)
c               generate do loops which loop over
c               the indices in piece;
c
c                (also accept TYPES ,e.g. FF,RR,SS,II=0,
c
c           e.g.   do 10 i = 1,3   for label = 10,indx = 'i',
c                                      piece = M
c*************************************************************
      subroutine put_do (lun,label,indx,piece)
      implicit none
      include 'params.'
      include 'pieces.'
      integer label,piece,lun
      character*2 indx
      integer i

      if (piece .eq. FR) then
         call put_f (lun,label,indx)
      else if (piece .eq. MR) then
         call put_m (lun,label,indx)
      else if (piece .eq. RMR) then
         call put_mm (lun,label,indx)
      else if (piece .eq. -MR) then
         write (6,*) 'we have a -MR in put_do'
         stop
      else if (piece .eq. FS) then
         call put_fs (lun,label,indx)
c
      else if (piece .eq. MS) then
         call put_ms1 (lun,label,indx)    !0,1
      else if (piece .eq. -MS) then
         call put_ms2 (lun,label,indx)    !5
c
      else if (piece .eq. M0) then
         call put_m0 (lun,label,indx)
      else if (piece .eq. M1) then
         call put_m1 (lun,label,indx)
      else if (piece .eq. M2) then
         call put_m2 (lun,label,indx)
      else if (piece .eq. NN) then
         call put_xx (lun,label,indx)

      else if (piece .eq. MT) then
         call put_tt (lun,label,indx)
```

```
      else if (piece .eq. TMT) then
         call put_tmt (lun,label,indx)
c
c must put M4, M5, M6, M7 for oex's
      else if (piece .eq. M3) then
         call put_M3 (lun,label,indx)
      else if (piece .eq. M4) then
         call put_M4 (lun,label,indx)
      else if (piece .eq. M5) then
         call put_M5 (lun,label,indx)
      else if (piece .eq. M6) then
         call put_M6 (lun,label,indx)
      else if (piece .eq. M7) then
         call put_M7 (lun,label,indx)

c
c types:
      else if (piece .eq. RR) then
         call put_fd_rr (lun,label,indx)
      else if (piece .eq. FF) then
         call put_fd_ff (lun,label,indx)
      else if (piece .eq. II) then
         call put_xx (lun,label,indx)
      else
         write (6,10)piece
10       format(1x,'error in put_do: unknown piece ',i4)
         stop
      endif

      return
      end
c*************************************************************
c
c     EXPANDIN (FDX,G)
c
c          generates code to expand FDX to a full
c          NxNxN array;
c
c*************************************************************
      integer function expandin (fdx,G1)
      implicit none
      include 'params.'
      include 'pieces.'
      integer fdx(0:MAXSIZE,0:2)
      integer G1(0:MAX_G,0:2)
      integer status,label,lun
      character*28 sub /'      subroutine expand(n,x)'/
      integer i,j,k
      integer quo(0:MAX_G,0:2)
      integer out_g(0:MAX_G,0:2)
      integer subgp(0:MAX_G)
      integer row(0:2)
```

```
      integer phase(0:2)
c*************************************************************
      integer msx_count
      integer msy_count
      integer msz_count
      integer msx_flg(0:1)
      integer msy_flg(0:1)
      integer msz_flg(0:1)
      integer hh,jj,kk
      integer offset

      msx_count = 0
      msx_flg(0) = 1
      msx_flg(1) = 1
      msy_count = 0
      msy_flg(0) = 1
      msy_flg(1) = 1
      msz_count = 0
      msz_flg(0) = 1
      msz_flg(1) = 1



      phase(0) = NO
      phase(1) = NO
      phase(2) = NO
      lun = CODEFILE

      write (lun,100) sub
100   format(a28)
      call put_head2 (lun)

      i = 0
      do while (fdx(i,0).ne.EOS)
         call iso_row (i,fdx,G1,subgp)
         call action (G1,subgp,quo,out_g)    !quo=G/subgp
c print do statements
         if (quo(1,XCOORD).eq.EOS) then
            go to 15   ! group is identity
         endif
c
         label = (i+1)*10
c
c set up ms*_counts and flags
         if (fdx(i,XCOORD).eq.MS) then
            msx_count = 1
            msx_flg(1) = -1
         else
            msx_count = 0
            msx_flg(1) = 1
         endif
         if (fdx(i,YCOORD).eq.MS) then
```

```
            msy_count = 1
            msy_flg(1) = -1
         else
            msy_count = 0
            msy_flg(1) = 1
         endif
         if (fdx(i,ZCOORD).eq.MS) then
            msz_count = 1

            msz_flg(1) = -1
         else
            msz_count = 0
            msz_flg(1) = 1
         endif
         do 999 hh = 0,msx_count
         do 999 jj = 0,msy_count
         do 999 kk = 0,msz_count
            label = (i+1)*10
            offset = hh*4 + jj*2 + kk
            label = label + offset

            fdx(i,XCOORD) = fdx(i,XCOORD)*msx_flg(hh)
            fdx(i,YCOORD) = fdx(i,YCOORD)*msy_flg(jj)
            fdx(i,ZCOORD) = fdx(i,ZCOORD)*msy_flg(kk)

            call put_do (lun,label,'i',fdx(i,XCOORD))
            call put_do (lun,label,'j',fdx(i,YCOORD))
            call put_do (lun,label,'k',fdx(i,ZCOORD))
c put back with
c right sign
            fdx(i,XCOORD) = fdx(i,XCOORD)*msx_flg(hh)
            fdx(i,YCOORD) = fdx(i,YCOORD)*msy_flg(jj)
            fdx(i,ZCOORD) = fdx(i,ZCOORD)*msy_flg(kk)

c
c now use action group to move the data around
c loop over each row in action group, print out stuff like

c     "y(mod(8-i,8),j,4+k) = y(i,j,k)*(-1)**(i+j)"
c
c NOTE! we start with j = 1, because the first
c row is the identity element and results in x(i,j,k) = x(i,j,k).

            j = 1
            do while (quo(j,0).ne.EOS)
               do 10 k = 0,2
                  row(k) = quo(j,k)

10             continue
c print out the group action statement
               call put_line (lun,row,phase)
               j = j+1
            enddo
```

```
            call put_cont (lun,label)
999       continue
15     continue
          i = i+1
       enddo

       call put_end(lun)
       return
       end
c*************************************************************
c
c      EXPANDOUT (FD,G1,G4,tflag)
c
c          generates code to expand FDX to a full
c          NxNxN array;
c
c*************************************************************
       integer function expandout (fd,G1,G4,tflag)
       implicit none
       include 'params.'
       include 'pieces.'
       integer fd(0:MAXSIZE,0:2)
       integer G1(0:MAX_G,0:2)
       integer G4(0:MAX_G,0:2)
       integer status,label,lun
       character*30 sub /'      subroutine expand2(n,x)'/
       integer i,j,k,ll,jj
       integer out_g(0:MAX_G,0:2)
       integer quo(0:MAX_G,0:2)
       integer subgp(0:MAX_G)
       integer row(0:2)
       integer phase(0:2),tflag(0:2)
       logical flag, in_gp
       integer phases (0:MAX_G,0:2)



       lun = CODEFILE
       i = 0
       do while (G1(i,0).ne.EOS)
          do 5 j = 0,2
             if (G1(i,j) .eq. SS .or. G1(i,j) .eq. TT ) then
                phases(i,j) = YES
             else
                phases(i,j) = NO
             endif
5         continue
          i = i+1
       enddo

       write (lun,100) sub
100    format(a30)
```

```
      call put_head2 (lun)

      i = 0
      do while (fd(i,0).ne.EOS)
         call iso_row (i,fd,G4,subgp)
         call action (G4,subgp,quo,out_g)
c quo=G/subgp; out_g = G4 array mod subgp; i.e. with dups
c this enables keeping track of corresponding row in G1
c for phase information.
c print do statements
         if (quo(1,XCOORD).eq.EOS) then
            go to 15   ! action group is identity
         endif
         label = (i+1)*10
         call put_do (lun,label,'i',fd(i,XCOORD))
         call put_do (lun,label,'j',fd(i,YCOORD))
         call put_do (lun,label,'k',fd(i,ZCOORD))
c
c now use action group to move the data around
c loop over each row in action group, print out stuff like

c     "y(mod(8-i,8),j,4+k) = y(i,j,k)*(-1)**(i+j)"
c
c NOTE! we start with j = 1, because the first
c row is the identity element and results in x(i,j,k) = x(i,j,k).

         j = 1
         do while (out_g(j,0).ne.EOS)
c
c skip dups:
            do 30 jj = 0,j-1
               if ( out_g(jj,0) .eq. out_g(j,0) .and.
     .              out_g(jj,1) .eq. out_g(j,1) .and.
     .              out_g(jj,2) .eq. out_g(j,2) ) then
                  go to 99 !next j
               endif
30          continue

            do 10 k = 0,2
               row(k) = out_g(j,k)
10          continue
c print out the group action statement
c first determine phases
            do 20 k = 0,2
               phase(k) = NO
c
c if element of action group(row) is present in the
c original group, there is no phase factor in expand -
c it could be moved by an element in the original group.
               if ( fd(i,k).ne.FR ) then
                  phase(k) = phases(j,k)
               endif
```

```
20          continue
            call put_line (lun,row,phase)
c            write (6,444)(row(ll),ll=0,2),(phase(ll),ll=0,2)
c444         format(1x,'row,phase ',3i4,3i4)
99          continue

            j = j+1
         enddo
         call put_cont (lun,label)

15    continue
         i = i+1
      enddo

      call put_end(lun)
      return
      end
c*************************************************************
c
c            call put_line (lun,row,phase)
c               generates a line in output file based on
c               row of action group:
c                 x(mod(8-i,8),mod(12-i,8),mod(4+i,8) = x(i,j,k)
c
c*************************************************************
      subroutine put_line (lun,row,phase)
      implicit none
      include 'params.'
      include 'pieces.'
      integer row(0:2), lun, phase(0:2)
      integer i
      character*2 indx(0:2) /'i,','j,','k '/
      character*11 modr(0:2)  /'mod(N-i,N),',
     .                         'mod(N-j,N),',
     .                         'mod(N-k,N) '/
      character*13 modt(0:2)  /'mod(N/2+i,N),',
     .                         'mod(N/2+j,N),',
     .                         'mod(N/2+k,N) '/
      character*15 mods(0:2)  /'mod(3*N/2-i,N),',
     .                         'mod(3*N/2-j,N),',
     .                         'mod(3*N/2-k,N) '/
      character*14 factor(0:7) /' ',                    !000
     .                          '*(-1)**(k)',           !001
     .                          '*(-1)**(j)',           !010

     .                          '*(-1)**(j+k)',         !011
     .                          '*(-1)**(i)',           !100
     .                          '*(-1)**(i+k)',         !101
     .                          '*(-1)**(i+j)',         !110
     .                          '*(-1)**(i+j+k)'  /     !111
c
c  fmt(2:4) determined in routine,
c  item contains 'x(', 3 coord actions, '=x(i,j,k)'
```

```
c  and, possibily, multiplication by a phase factor '*(-1)(i+j+k)'
      character*20 item(0:5)
      data item(0) /'x('/
      data item(4) /')=x(i,j,k)'/
c
c      data item(5) /'*(-1)**(i+j+k)'/
      character*7 fmt(0:5)
      data fmt(0) /'(9x,a2,'/    ! for 'x('
      data fmt(4) /'a10,'/       ! for ')=x(i,j,k)'
      data fmt(5) /'a14)'/       !for phase factor - could be blank

c
c
c      write (6,999)(row(i),i=0,2),(phase(i),i=0,2)
c999   format(1x,'entry to put_line:row,phase ',3i4,3i4)

      do 10 i = 0,2
         if (row(i).eq.II)then
            fmt(i+1) = 'a2,'
            item(i+1)(1:2) = indx(i)(1:2)
         else if (row(i).eq.RR) then
            fmt(i+1) = 'a11,'
            item(i+1)(1:11) = modr(i)(1:11)
         else if (row(i).eq.TT) then
            fmt(i+1) = 'a15,'
            item(i+1)(1:15) = modt(i)(1:13)
         else if (row(i).eq.SS) then
            fmt(i+1) = 'a17,'
            item(i+1)(1:17) = mods(i)(1:15)
         else
            write (6,*) 'error in put_line'
            stop
         endif
10    continue
c
c
c  now determine contents of item 5, for phase factor
c
      i = 4*phase(0) + 2*phase(1) + phase(2)
      item(5)(1:20) = factor(i)(1:14)


c      write (6,123)(fmt(i),i=0,4)
c123   format(1x,5a7)
      write (lun,fmt) (item(i),i=0,5)
      return
      end
c*************************************************************88
c
c
c
c       in_gp (G1,row): is row in G1?
```

```
c
c*************************************************************
      logical function in_gp (G1,row)
      implicit none
      include 'params.'
      include 'pieces.'
      integer G1(0:MAX_G,0:2)
      integer i
      integer row(0:2)

c      write (6,100) (row(i),i=0,2)
c100   format(1x,'entering in_gp,row = ',3i4,/,'G1 = ')
c      call prt_g(G1,STDOUT)

      in_gp = .false.
      do 10 i = 0,ordg-1

         if (row(0).eq.G1(i,0) .and.
     .       row(1).eq.G1(i,1) .and.
     .       row(2).eq.G1(i,2) ) then
            in_gp = .true.
            return
         endif
10    continue

c      write (6,200) in_gp
c200   format (1x,'ANSWER IS: ',L1)
      return
      end
c*************************************************************************
c
c
c     iso_xz: input: rownum,fd,gp,subpg
c             (exactly like iso_row!)
c              looks at a given row of FD;
c              puts isotropy of X and Z coordinates of
c                 that row in subgp
c              returns number of elements in isotropy subgp
c
c
c*************************************************************************
      integer function iso_xz (rownum,FD,gp,subgp)
      implicit none
      include 'params.'
      include 'pieces.'
      integer FD(0:MAXSIZE,0:2)
      integer rownum
      integer gp(0:MAX_G,0:2)
      integer subgp(0:*)
      integer i,j
      logical fixes
```

```
      iso_xz = 0
      j = 0

      do 10 i = 0,ordg-1          !for each element of G

         if (fixes (gp(i,XCOORD), FD(rownum,0)) .and.
     .       fixes (gp(i,ZCOORD), FD(rownum,2)) ) then

            subgp (j) = i
            j = j + 1
            iso_xz = iso_xz + 1
         endif

10    continue
      subgp(j) = EOS

d      write (6,100)iso_xz
d100   format (1x,'in iso_xz, size of iso_xz = ',i4)
d      write (6,200)(subgp(i),i=0,j)
d200   format (1x,'in iso_xz, subgp =',9i4)
      return
      end
c*********************************************
c
c
c
c*********************************************
      integer function iso_yz (rownum,FD,gp,subgp)
      implicit none
      include 'params.'
      include 'pieces.'
      integer FD(0:MAXSIZE,0:2)
      integer rownum
      integer gp(0:MAX_G,0:2)
      integer subgp(0:*)
      integer i,j
      logical fixes

      iso_yz = 0
      j = 0
      do 20 i = 0,ordg-1          !for each element of G

         if (fixes (gp(i,YCOORD), FD(rownum,1)) .and.
     .       fixes (gp(i,ZCOORD), FD(rownum,2)) ) then

            subgp (j) = i
            j = j + 1
            iso_yz = iso_yz + 1
         endif

20    continue
      subgp(j) = EOS
```

```
d      write (6,100)iso_yz
d100   format (1x,'in iso_yz, size of iso_yz = ',i4)
d      write (6,200)(subgp(i),i=0,j)
d200   format (1x,'in iso_yz, subgp =',9i4)
      return


      end
c*********************************************************************
c
c
c     chk_for_trans(rownum,fd,gp,G1,num,ft_type)
c
c         this routine determines if xformy (xformz) needs
c         an "after translation" fourier transform or not.
c         extra wrinkle: if the action on Y (Z) is FULL,
c         must determine whether to use ft2135,57,or 37
c
c            rownum: rownum of fd
c                gp: G2 or G3
c               num: 1,2,or 3
c
c                    if num = 1, return NO
c                    else if num = 2
c                      for each elt in iso(x,z)
c                         if corresponding elt in G1 has a translation
c                            record that fact
c
c     NOTE: FD is arranged with the first coordinate being
c           the one about to be transformed;
c           the group is also arranged in this way.
c           the only thing to do - to keep things in synch with
c            G1 for instance, and to keep things straight in
c            my head - is to straighten things out;
c
c*********************************************************************
      integer function chk_for_trans (rownum,FD,gp,G1,
     .num,ft_type,xy_flg)

      implicit none
      include 'params.'
      include 'pieces.'
      integer FD(0:MAXSIZE,0:2)
      integer rownum,num
      integer gp(0:MAX_G,0:2)
      integer G1(0:MAX_G,0:2)
      character*2 ft_type
      integer xy_flg
      integer subgp(0:MAX_G)
      integer i,j,k,count,sum
```

```
      integer xptr (0:MAX_G)
      integer yptr (0:MAX_G)
      integer iso_xz,iso_xy,scan
      integer iso_yz
      chk_for_trans = NO

      if(num.eq.1)then
         return
      endif

      if (fd(rownum,YCOORD).eq.FR .and.
     .    fd(rownum,ZCOORD).eq.FR ) then
         return

      endif

      if (num .eq. 2) then
         if (fd(rownum,YCOORD).eq.FR) then
            return    !(-1)**i = 1 always.
         endif

         count = iso_yz (rownum,FD,gp,subgp)
c
c iso(x,z) is in subgp; get corresponding elements
c in XCOORD; if there are translations, return YES,
c unless the type of action on YCOORD is FULL - then
c we must determine 35,37,or 57. REMEMBER! things are
c  arranged: YXZ  so YCOORD in following really means XCOORD
c   except! G1 is in the right order. Ugh!
            j = 0
            do 10 i = 0,count-1
               if (G1(subgp(i),XCOORD).eq.SS. or.
     .              G1(subgp(i),XCOORD).eq.TT ) then
                  xptr(j) = subgp(i)
                  j = j+1
               endif
10          continue
            xptr(j) = EOS
c
c if type is not FF, and there were translations, return YES
            if (j .eq. 0) then
               return
            endif
c
c
            xy_flg = 1
    chk_for_trans = YES

            if (scan(gp,XCOORD,subgp) .ne. FF) then
               return
            endif
```

```
      write (6,901) j,(subgp(sum),sum=0,MAX_G),
     .              (xptr(sum),sum=0,MAX_G)
901   format (1x,'debug code,2nd round : j = ',i4,'subgp = ',9i4,
     .        /,'xptr = ',9i4)

c
c fall through means we must determine 35,57,37
c we use the following facts: RR + TT = 1 + 2
c                             RR + SS = 1 + 3
c                             SS + TT = 2 + 3
            sum = gp(xptr(0),XCOORD) + gp(xptr(1),XCOORD)
            if (sum.eq.3) then
       ft_type = '57'
            else if (sum.eq.4) then
               ft_type = '37'
            else if (sum.eq.5) then
               ft_type = '35'
            else
               write (6,210) (xptr(i),i=0,MAX_G-1)
210            format(1x,'error in chk_for, xptr = ',8i4)
               call prt_g (gp,STDOUT)
               stop
            endif

      endif
c
c CASE OF NUM = 3
      if (num .eq. 3) then

d         write(6,*)'in chk_trans,num=3'
d         call prt_g (gp,STDOUT)
d         call prt_fd (fd,STDOUT)
d         read(*,*)

         count = iso_yz (rownum,FD,gp,subgp)
            j = 0

            k = 0
            do 20 i = 0,count-1
               if (G1(subgp(i),XCOORD).eq.SS .or.
     .              G1(subgp(i),XCOORD).eq.TT ) then
                  xptr(j) = subgp(i)
                  j = j+1
               endif

               if (G1(subgp(i),YCOORD).eq.SS .or.
     .              G1(subgp(i),YCOORD).eq.TT ) then
                  yptr(k) = subgp(i)
                  k = k+1
               endif
20          continue
```

```
            xptr(j) = EOS
            yptr(k) = EOS

d       write (6,900) j,k,(subgp(sum),sum=0,MAX_G),
d      .              (xptr(sum),sum=0,MAX_G),(yptr(sum),
d      .               sum = 0,MAX_G)
d900    format (1x,'in chk_trans,num=3: j,k = ',2i4,'subgp = ',9i4,
d      .        /,1x,'xptr = ',9i4,/,1x,'yptr = ',9i4)

            if (j .eq. 0 .and. k .eq. 0) then
               return
            endif
c
c set phase indicator flag for first two coordinates
c 1 means use mod(i,2)
c 3 means use mod(i+j,2)
c 2 means use mod(j,2)   in oex.
            if (j.eq.0) then
               xy_flg = 1
            else if (k .eq. 0) then
               xy_flg = 2
            else
               xy_flg = 3
            endif
    chk_for_trans = YES

            if (scan(gp,XCOORD,subgp) .ne. FF) then
               return
            endif
c
c  recall: arrangement is Z X Y, so YCOORD means X;
c          also, yptr is pointing to ycoord of G1;
            if (fd(rownum,YCOORD).eq.FR) then
               if (k.eq.0) then
                  chk_for_trans = NO
                  return
               else
                  sum = gp(yptr(0),XCOORD) + gp(yptr(1),XCOORD)
               endif
            else
               if (j.eq.0) then
                  chk_for_trans = NO
                  return
               else
                  sum = gp(xptr(0),XCOORD) + gp(xptr(1),XCOORD)
               endif
            endif

            if (sum.eq.3) then
       ft_type = '57'
            else if (sum.eq.4) then
               ft_type = '37'
```

```
            else if (sum.eq.5) then
               ft_type = '35'
            else
               write (6,220) (xptr(i),i=0,MAX_G-1)
220            format(1x,'error in chk_for, xptr =,num=3 ',8i4)
               call prt_g(gp,STDOUT)
               stop
            endif

      endif


d      write (6,223) (xptr(i),i=0,MAX_G-1)
d223   format(1x,'exiting chk_for: xptr = ',8i4)


      return
      end
```

5.6 MODULE: GENOEX

SUBROUTINES AND FUNCTIONS IN GENOEX

```
*******************************
c
c the following routines are used in calculating
c the orbit exchange; in conjunction they break
c up a fundamental domain into its most basic isotropy
c pieces;

      subroutine atomize (fdxout,temp2,gp)
      subroutine decompose (fd_in,fd_out,gp)
c
c  the following routines put out a line of an
c  oex routine, of the form:
c
c      y(i,j,k) = x(mod(N-i),j,mod(N/2+k))*(-1)^(i+j)

      subroutine put_oex1_line (lun,in_row,phase)
      subroutine put_oex2_line (lun,in_row,phase)
      subroutine put_oex3_line (lun,row,phase)
c
c gen_oex is the routine which generates the orbit exchange code

      integer function gen_oex (num,fdxout,fdy,in_gp,Gi)
c
c gp_elt finds which group element in gp will map
c current row to the given fd (fundamental domain)

      integer function gp_elt (gp,row,fd)
c
c returns the action of a group element on an isotropy piece;
c it is only called when it is well-defined.
      integer function mult (elt,piece)
*******************************
```

```
c*************************************************************
c
c
c     GEN_OEX (num,FDXOUT,fdy,in_gp)
c
c       generatex oex1,oex2,oex3 depending on num
c
c    fdxout is a fundamental domain to be mapped to
c    the next fundamental domain via an orbit exchange;
c
c    num = 1: fdxout --> fdy;
c    num = 2: fdyout --> fdz;
c    num = 3: fdzout --> fdout (= FD for G***)
c
c*************************************************************
      integer function gen_oex (num,fdxout,fdy,in_gp,G1)
      implicit none
      include 'params.'
      include 'pieces.'
      integer fdxout(0:MAXSIZE,0:2)
      integer fdy(0:MAXSIZE,0:2)
      integer temp1(0:MAXSIZE,0:2)
      integer temp2(0:MAXSIZE,0:2)
      integer temp3(0:MAXSIZE,0:2)
      integer in_gp(0:MAX_G,0:2)
      integer gp(0:MAX_G,0:2)
      integer G1(0:MAX_G,0:2)

      integer subgp(0:MAX_G)
      integer status,label,lun
      integer num
      character*28 subs (0:2) /'      subroutine oex1(n,x,y)',
     .                        '      subroutine oex2(n,x,y)',
     .                        '      subroutine oex3(n,x,y)'/

c
      integer i,j,kk,n,ll,m
      integer row(0:2)
      integer gp_elt
      logical is_fd,stat
      integer phase(0:2)

      lun = CODEFILE

      write (lun,100) subs(num-1)
100   format(a28)
      call put_head (lun)
c
      call fd_copy (fdxout,temp2)
      call fd_copy (fdy,temp3)
      call gcopy (in_gp,gp)
```

```
c
c
c  now temp1 and temp2 are lined up in XYZ order, ready for
c  comparison. And the group is synched up with everything too!
c  first we must decompose each column into the isotropy pieces
c  for the group acting on that column
      call atomize (temp2,temp1,gp)
      call atomize (temp3,temp2,gp)

      if (debug.ge.1)then
         write (6,*)'in gen_oex, comparing FD1:'
         call prt_fd(temp1,STDOUT)
         read(*,*)
         write (6,*)'with FD2:'
         call prt_fd(temp2,STDOUT)
         read(*,*)
      endif

      i = 0
      do while (temp1(i,0).ne.EOS)
         label = (i+1)*10

         call put_do (lun,label,'i',temp1(i,XCOORD))
         call put_do (lun,label,'j',temp1(i,YCOORD))
         call put_do (lun,label,'k',temp1(i,ZCOORD))
         do 10 j = 0,2
            row(j) = temp1(i,j)
10       continue
c
c  now find which gp elt maps current row of temp1 to
c  any row of temp2. Answer is index to gp array.
         n = gp_elt(gp,row,temp2)
c
c now use row to pass the group element to the
c put_oex_line routine.
         do 20 kk = 0,2
            row(kk) = gp(n,kk)
            if (G1(n,kk) .eq. TT .or. Gi(n,kk) .eq. SS) then
               phase(kk) = YES
            else
               phase(kk) = NO
            endif
20       continue



         if (num.eq.1)then
            call put_oex1_line(lun,row,phase)
         else if (num.eq.2)then
            call put_oex2_line(lun,row,phase)
         else if (num.eq.3)then
            call put_oex3_line(lun,row,phase)
         endif
```

```
         call put_cont(lun,label)
         i = i+1
      enddo

      call put_end(lun)


      return
      end
c*******************************************************
c
c     gp_elt(gp,row,fd)
c          finds the group element which
c          maps given row to given fd
c          this is the oex: row is from,e.g.,fdxout,
c          and fd is fdy.
c
c*******************************************************
      integer function gp_elt (gp,row,fd)
      implicit none
      include 'params.'
      include 'pieces.'
      integer gp(0:MAX_G,0:2)
      integer fd(0:MAXSIZE,0:2)
      integer row(0:2)
      integer i,j
      integer mult

c      write (6,99)(row(i),i=0,2)
c99    format(1x,'entering gp_elt, trying to match row: ',3i4)
c      read(*,*)

      i = 0
      do while (fd(i,0).ne.EOS)
         do 10 j = 0,ordg-1
            if ( mult(gp(j,0),row(0)) .eq. fd(i,0) .and.
     .           mult(gp(j,1),row(1)) .eq. fd(i,1) .and.
     .           mult(gp(j,2),row(2)) .eq. fd(i,2) ) then
               go to 100
            endif
10       continue
         i = i+1
      enddo

c

c  go here with j = index in gp table of element that performs oex.
100   continue

      if (fd(i,0).eq.EOS)then
         write (6,110) (row(i),i=0,2)
110      format(1x,'error: no oex found for: ',3i4)
```

```
         call prt_g(gp,STDOUT)
         stop
      endif

      gp_elt = j

c      write (6,123)j
c123   format(1x,'gp_elt = ',i4)

      return
      end

c***********************************************
c
c          mult(elt,piece)
c              returns action of group elt on given piece
c
c     elt = 0,1,2,3 = II,RR,TT,SS
c
c***********************************************
      integer function mult (elt,piece)
      implicit none
      include 'params.'
      include 'pieces.'
      integer elt,piece,indx
      integer table(0:3,0:16)
      integer i,j
c
c   i,j entry equals entry in following table:
c
c                        I    R    T    S
      data table /  FR,  FR,  -1,  -1,
     .              MR, RMR,  -1,  -1,
     .              FS,  -1,  -1,  FS,
     .              MS,  -1,  -1, SMS,
     .              M0,  M0,  M4,  M4,
     .              M1,  M7,  M5,  M3,
     .              M2,  M6,  M6,  M2,
     .              NN,  -1,  -1,  -1,
     .              MT,  -1, TMT,  -1,
     .              M3,  M5,  M7,  M1,
     .              M4,  M4,  M0,  M0,
     .              M5,  M3,  M1,  M7,
     .              M6,  M2,  M2,  M6,
     .              M7,  M1,  M3,  M5,
     .             SMS,  -1,  -1,  MS,
     .             TMT,  -1,  MT,  -1,
     .             RMR,  MR,  -1,  -1 /


      if (piece.eq.-MR) then
         write (6,*) 'how did this! get here?'
```

```
      else
         indx = piece - 100
      endif

      if (table(elt,indx).eq.-1) then
         write (6,1000)elt,piece,indx
1000     format(1x,'bad mult: elt = ',i4,' piece = ',i4,'indx=',i4)
         stop
      endif


      mult = table (elt,indx)

      return
      end
c*************************************************************
c
c        call atomize (fdxout,temp2,gp)
c        this routine decompose each column of fdxout
c        into its isotropy pieces relative to the action of
c        the gp restricted to said column
c        it does this by calling decompose, which does
c        it for the first column, then switching the
c        the columns around and calling decompose again, etc.
c        the output is left in temp2, and fdxout is never
c        disturbed.
c
c*************************************************************8
      subroutine atomize (fdxout,temp2,gp)
      implicit none
      include 'params.'
      include 'pieces.'

      integer fdxout (0:MAXSIZE,0:2)
      integer temp2 (0:MAXSIZE,0:2)
      integer temp1 (0:MAXSIZE,0:2)
      integer gp(0:MAX_G,0:2)
      integer gp2(0:MAX_G,0:2)
      integer gp3(0:MAX_G,0:2)

      integer i
      integer j

      call decompose (fdxout,temp1,gp)

c      call prt_fd(temp1,STDOUT)
c      write (6,*)'decompose of first col'
c      read (*,*)

      call fd1_2(temp1,temp2)
      call g1_2(gp,gp2)
      call decompose (temp2,temp1,gp2)
```

```
c       call prt_fd(temp1,STDOUT)
c       write (6,*)'decompose of second col'
c       read (*,*)


c
c must put temp1 back in XYZ order, then
c bring Z up front
      call fd1_2(temp1,temp2)
      call fd3_1(temp2,temp1)
      call g1_2(gp2,gp3)
      call g3_1(gp3,gp2)

c       write(*,*)'about to decompose '
c       call prt_fd(temp1,STDOUT)
c       call prt_g(gp2,STDOUT)
c       read(*,*)
      call decompose (temp1,temp2,gp2)

c       call prt_fd(temp2,STDOUT)
c       write (6,*)'decompose of third col'
c       read (*,*)

c
c finally put it back
      call fd1_3(temp2,temp1)
      call fd_copy (temp1,temp2)
c
c       write(6,*)'in atomize,input fd, output fd'
c       call prt_fd(fdxout,STDOUT)
c       read(*,*)
c       call prt_fd(temp2,STDOUT)
c
      return
      end
c**************************************
c
c  decompose: take first column, and type of action
c             on it; and decompose into isotropy pieces;
c             so type II ==> no decompose
c                     RR ==> (NN --> FR U MR U RMR)
c                     TT ==> (NN --> MT U TMT)

c                     SS ==> (NN --> FS U MS U SMS)
c                     FF ==> (NN --> M0,M1,M2,...,M7)
c                            (FS --> M2,M6)
c                            (MS --> M0,M1,M5),
c                             FR --> M0,M4
c                             MR --> M1,M2,M3
c                             M0 --> M0,etc
c                             MT --> M0,M1,M2,M3
c
```

```
c**************************************
      subroutine decompose (fd_in,fd_out,gp)
      implicit none
      include 'params.'
      include 'pieces.'
      integer fd_in (0:MAXSIZE,0:2)
      integer fd_out (0:MAXSIZE,0:2)
      integer gp(0:MAX_G,0:2)
      integer scan,subgp(0:MAX_G)
      integer type
      integer i,j,k
      integer kk,ll

d      write (6,*) 'on entry to decompose;'
d      call prt_fd (fd_in,STDOUT)

c
c we're only interested in the type of action here
c  essentially, we'd like to switch on type.
      do 1 i = 0,ordg-1
         subgp(i) = i
1     continue
      subgp(i) = EOS

      type = scan(gp,XCOORD,subgp)

c      write (6,901) type
c901   format(1x,'in decompose, type = ',i4)
c      read(*,*)


      if (type .eq. II) then
         call fd_copy (fd_in,fd_out)
         return
      else if (type .eq. RR) then
         i = 0
         j = 0
         do while ( fd_in(i,XCOORD) .ne. EOS)
            if ( fd_in(i,XCOORD) .eq. NN) then
               fd_out(j,XCOORD)   =  FR
               fd_out(j+1,XCOORD) =  MR
               fd_out(j+2,XCOORD) = RMR
               do 10 k = 1,2
                  fd_out(j,k) = fd_in(i,k)
                  fd_out(j+1,k) = fd_in(i,k)
                  fd_out(j+2,k) = fd_in(i,k)
10             continue
               i = i+1
               j = j+3
            else
               do 15 k = 0,2
                  fd_out(j,k) = fd_in(i,k)
15             continue
```

```
               i = i+1
               j = j+1
            endif

         enddo
         fd_out(j,XCOORD) = EOS
         return
c
c  CASE: type = SS
      else if (type .eq. SS) then
         i = 0
         j = 0
         do while ( fd_in(i,XCOORD) .ne. EOS)
            if ( fd_in(i,XCOORD) .eq. NN) then
               fd_out(j,XCOORD)   =  FS
               fd_out(j+1,XCOORD) =  MS
               fd_out(j+2,XCOORD) = SMS
               do 20 k = 1,2
                  fd_out(j,k) = fd_in(i,k)
                  fd_out(j+1,k) = fd_in(i,k)
                  fd_out(j+2,k) = fd_in(i,k)
20             continue
               i = i+1
               j = j+3
            else
               do 25 k = 0,2
                  fd_out(j,k) = fd_in(i,k)
25             continue
               i = i+1
               j = j+1
            endif

         enddo
         fd_out(j,XCOORD) = EOS
         return
c
c  CASE: type = TT
      else if (type .eq. TT) then
         i = 0
         j = 0
         do while ( fd_in(i,XCOORD) .ne. EOS)
            if ( fd_in(i,XCOORD) .eq. NN) then
               fd_out(j,XCOORD)   =  MT
               fd_out(j+1,XCOORD) = TMT
               do 30 k = 1,2
                  fd_out(j,k) = fd_in(i,k)
                  fd_out(j+1,k) = fd_in(i,k)
30             continue
               i = i+1
               j = j+2
            else
               do 35 k = 0,2
```

```
                  fd_out(j,k) = fd_in(i,k)
35              continue
                i = i+1
                j = j+1
             endif

          enddo
          fd_out(j,XCOORD) = EOS
          return

c
c  CASE: type = FF
       else if (type .eq. FF) then
          i = 0
          j = 0
          do while ( fd_in(i,XCOORD) .ne. EOS)
             if ( fd_in(i,XCOORD) .eq. NN) then
                fd_out(j,XCOORD)   =  M0
                fd_out(j+1,XCOORD) =  M1
                fd_out(j+2,XCOORD) =  M2
                fd_out(j+3,XCOORD) =  M3
                fd_out(j+4,XCOORD) =  M4
                fd_out(j+5,XCOORD) =  M5
                fd_out(j+6,XCOORD) =  M6
                fd_out(j+7,XCOORD) =  M7
                do 40 k = 1,2
                do 40 kk = j,j+7
                   fd_out(kk,k) = fd_in(i,k)
40              continue
                i = i+1
                j = j+8
             else if ( fd_in(i,XCOORD) .eq. MT) then
                fd_out(j,XCOORD)   =  M0
                fd_out(j+1,XCOORD) =  M1
                fd_out(j+2,XCOORD) =  M2
                fd_out(j+3,XCOORD) =  M3
                do 45 k = 1,2
                do 45 kk = j,j+3
                   fd_out(kk,k) = fd_in(i,k)
45              continue
                i = i+1
                j = j+4
             else if ( fd_in(i,XCOORD) .eq. FS) then
                fd_out(j,XCOORD)   =  M2
                fd_out(j+1,XCOORD) =  M6
                do 50 k = 1,2
                do 50 kk = j,j+1
                   fd_out(kk,k) = fd_in(i,k)
50              continue
                i = i+1
                j = j+2
```

```
            else if ( fd_in(i,XCOORD) .eq. FR) then
               fd_out(j,XCOORD)   =  M0
               fd_out(j+1,XCOORD) =  M4
               do 55 k = 1,2
               do 55 kk = j,j+1
                  fd_out(kk,k) = fd_in(i,k)
55             continue
               i = i+1
               j = j+2
            else if ( fd_in(i,XCOORD) .eq. MR) then
               fd_out(j,XCOORD)   =  M1
               fd_out(j+1,XCOORD) =  M2
               fd_out(j+2,XCOORD) =  M3
               do 60 k = 1,2
               do 60 kk = j,j+2
                  fd_out(kk,k) = fd_in(i,k)
60             continue
               i = i+1
               j = j+3
            else if ( fd_in(i,XCOORD) .eq. MS) then
               fd_out(j,XCOORD)   =  M0
               fd_out(j+1,XCOORD) =  M1
               fd_out(j+2,XCOORD) =  M5
               do 65 k = 1,2
               do 65 kk = j,j+2
                  fd_out(kk,k) = fd_in(i,k)
65             continue
               i = i+1
               j = j+3
            else if (fd_in(i,XCOORD).ne.M0 .and.
     .               fd_in(i,XCOORD).ne.M1 .and.
     .               fd_in(i,XCOORD).ne.M2 ) then
                  write (6,123)(fd_in(i,j),j=0,2)
123               format(1x,'error in decompose: ',3i4)
                  stop
            else
               do 70 k = 0,2
                  fd_out(j,k) = fd_in(i,k)
70             continue
               i = i+1
               j = j+1
            endif

         enddo
         fd_out(j,XCOORD) = EOS
         return

      endif

      end
c*************************************************************
c
```

```
c            call put_oex_line (lun,row)
c               generates a line in output file based on
c               row of action group:
c                 y(mod(8-i,8),mod(12-i,8),mod(4+i,8) = x(i,j,k)
c
c             row = (R,S,T) (e.g.); however it is in XYZ order,
c             and so must be permuted to agree with the
c             oex transposition;
c
c                    the transpositions are evident in the data
c                    statements for the action.
c
c*****************************************************************
      subroutine put_oex1_line (lun,in_row,phase)
      implicit none
      include 'params.'
      include 'pieces.'
      integer in_row(0:2), lun, phase(0:2)
      integer row(0:2)
      integer i
      character*2 indx(0:2) /'j,','i,','k '/
      character*11 modr(0:2)  /'mod(N-j,N),',
     .                         'mod(N-i,N),',
     .                         'mod(N-k,N)'/
      character*13 modt(0:2)  /'mod(N/2+j,N),',
     .                         'mod(N/2+i,N),',
     .                         'mod(N/2+k,N)'/
      character*15 mods(0:2)  /'mod(3*N/2-j,N),',
     .                         'mod(3*N/2-i,N),',
     .                         'mod(3*N/2-k,N)'/
c
c  fmt(2:4) determined in routine,
c  item contains 'y(', 3 coord actions, '=x(i,j,k)'
      character*20 item(0:5)
      data item(0) /'y('/
      data item(4) /')=x(i,j,k)'/

      character*7 fmt(0:5)
      data fmt(0) /'(9x,a2,'/   ! for 'y('
      data fmt(4) /'a10,'/      ! for ')=x(i,j,k)'
      data fmt(5)/'a14)'/
      character*14 factor(0:7) /' ',                    !000
     .                          '*(-1)**(k)',           !001
     .                          '*(-1)**(j)',           !010
     .                          '*(-1)**(j+k)',         !011
     .                          '*(-1)**(i)',           !100
     .                          '*(-1)**(i+k)',         !101
     .                          '*(-1)**(i+j)',         !110
     .                          '*(-1)**(i+j+k)'  /     !111


c
```

```
c first re-arrange the incoming group element:
c     i,j,k --> j,i,k
      row(0) = in_row(1)
      row(1) = in_row(0)
      row(2) = in_row(2)

      do 10 i = 0,2
         if (row(i).eq.II)then
            fmt(i+1) = 'a2,'
            item(i+1)(1:2) = indx(i)(1:2)
         else if (row(i).eq.RR) then
            fmt(i+1) = 'a11,'
            item(i+1)(1:11) = modr(i)(1:11)
         else if (row(i).eq.TT) then
            fmt(i+1) = 'a13,'
            item(i+1)(1:15) = modt(i)(1:13)
         else if (row(i).eq.SS) then
            fmt(i+1) = 'a15,'
            item(i+1)(1:17) = mods(i)(1:15)
         else
            write (6,*) 'error in put_oex1'
            stop
         endif
10    continue

      i = 4*phase(0)
      item(5)(1: len(factor(i)) ) = factor(i)
c
c cosmetics: do an internal write to convert len(factor) to character
c and concatenate it with 'a'//len to form fmt(5). This will get
c rid of extra blanks in the output file

c      write (6,123)(fmt(i),i=0,4)
c123   format(1x,5a7)
      write (lun,fmt) (item(i),i=0,5)
      return
      end

c*************************************************
c
c   put_oex2_line
c
c
c*************************************************
      subroutine put_oex2_line (lun,in_row,phase)
      implicit none
      include 'params.'
      include 'pieces.'
      integer in_row(0:2), lun, phase(0:2)
      integer row(0:2)
      integer i
      character*2 indx(0:2) /'k,','i,','j '/
      character*11 modr(0:2)  /'mod(N-k,N),',
```

```
     .                          'mod(N-i,N),',
     .                          'mod(N-j,N)'/
      character*13 modt(0:2)   /'mod(N/2+k,N),',
     .                          'mod(N/2+i,N),',
     .                          'mod(N/2+j,N)'/
      character*15 mods(0:2)   /'mod(3*N/2-k,N),',
     .                          'mod(3*N/2-i,N),',
     .                          'mod(3*N/2-j,N) '/
c
c  fmt(2:4) determined in routine,
c  item contains 'y(', 3 coord actions, '=x(j,i,k)'
      character*20 item(0:5)
      data item(0) /'y('/
      data item(4) /') = x(j,i,k)'/

      character*7 fmt(0:5)
      data fmt(0) /'(9x,a2,'/    ! for 'y('
      data fmt(4) /'a12,'/       ! for ') = x(i,j,k)'
      data fmt(5)/'a14)'/
      character*14 factor(0:7) /' ',                    !000
     .                          '*(-1)**(k)',           !001
     .                          '*(-1)**(j)',           !010
     .                          '*(-1)**(j+k)',         !011
     .                          '*(-1)**(i)',           !100
     .                          '*(-1)**(i+k)',         !101
     .                          '*(-1)**(i+j)',         !110
     .                          '*(-1)**(i+j+k)'  /     !111

c first re-arrange the incoming group element:
c      i,j,k --> k,i,j. in_row is in i,j,k order!
      row(2) = in_row(1)
      row(1) = in_row(0)
      row(0) = in_row(2)

      do 10 i = 0,2
         if (row(i).eq.II)then
            fmt(i+1) = 'a2,'
            item(i+1)(1:2) = indx(i)(1:2)
         else if (row(i).eq.RR) then
            fmt(i+1) = 'a11,'
            item(i+1)(1:11) = modr(i)(1:11)
         else if (row(i).eq.TT) then
            fmt(i+1) = 'a13,'
            item(i+1)(1:15) = modt(i)(1:13)
         else if (row(i).eq.SS) then
            fmt(i+1) = 'a15,'
            item(i+1)(1:17) = mods(i)(1:15)
         else
            write (6,*) 'error in put_oex2'
            stop
         endif
10    continue
```

```
      i = 4*phase(0) + 2*phase(1)
      item(5)(1:20) = factor(i)(1:14)

c      write (6,123)(fmt(i),i=0,4)
c123   format(1x,5a7)
      write (lun,fmt) (item(i),i=0,5)
      return
      end

c*************************************************
c
c   put_oex3_line
c
c
c*************************************************
      subroutine put_oex3_line (lun,row,phase)
      implicit none
      include 'params.'
      include 'pieces.'
      integer row(0:2), lun, phase(0:2)
      integer n   !this is which group element is doing the oex
      integer i
      character*2 indx(0:2) /'i,','j,','k '/
      character*11 modr(0:2)  /'mod(N-i,N),',
     .                         'mod(N-j,N),',
     .                         'mod(N-k,N) '/
      character*13 modt(0:2)  /'mod(N/2+i,N),',
     .                         'mod(N/2+j,N),',
     .                         'mod(N/2+k,N) '/
      character*15 mods(0:2)  /'mod(3*N/2-i,N),',
     .                         'mod(3*N/2-j,N),',
     .                         'mod(3*N/2-k,N) '/
c
c  fmt(2:4) determined in routine,
c  item contains 'y(', 3 coord actions, '=x(i,j,k)'
      character*20 item(0:5)
      data item(0) /'y('/
      data item(4) /') = x(k,i,j)'/

      character*7 fmt(0:5)
      data fmt(0) /'(9x,a2,'/   ! for 'y('
      data fmt(4) /'a12,'/      ! for ') = x(i,j,k)'
      data fmt(5) /'a14)'/       ! for *(-1)**(i+j+k)


      character*14 factor(0:7) /' ',                    !000
     .                          '*(-1)**(k)',           !001

     .                          '*(-1)**(j)',           !010
     .                          '*(-1)**(j+k)',         !011
     .                          '*(-1)**(i)',           !100
     .                          '*(-1)**(i+k)',         !101
```

```
     .                          '*(-1)**(i+j)',        !110
     .                          '*(-1)**(i+j+k)'  /    !111
c
c don't need to re-arrange the incoming group element:

      do 10 i = 0,2
         if (row(i).eq.II)then
            fmt(i+1) = 'a2,'
            item(i+1)(1:2) = indx(i)(1:2)
         else if (row(i).eq.RR) then
            fmt(i+1) = 'a11,'
            item(i+1)(1:11) = modr(i)(1:11)
         else if (row(i).eq.TT) then
            fmt(i+1) = 'a13,'
            item(i+1)(1:15) = modt(i)(1:13)
         else if (row(i).eq.SS) then
            fmt(i+1) = 'a15,'
            item(i+1)(1:17) = mods(i)(1:15)
         else
            write (6,*) 'error in put_oex3'
            stop
         endif
10    continue

      i = 4*phase(0) + 2*phase(1) + phase(2)
      item(5)(1:20) = factor(i)(1:14)

c      write (6,123)(fmt(i),i=0,4)
c123   format(1x,5a7)
      write (lun,fmt) (item(i),i=0,5)
      return
      end
```

5.7 MODULE: INPUT

```
                    SUBROUTINES IN INPUT

*******************************
c
c
c get_g: gets choice of group from user;
c this module has all the groups in it

      subroutine get_g (gp,gpname)

*******************************
```

```
      subroutine get_g (gp,gpname)
      implicit none
      include 'params.'
      include 'pieces.'
      integer gp(0:MAX_G,0:2)
      character*20 gpname

c this group is P1
      integer G10(0:MAX_G,0:2)
      character*20 G10name /'P1'/
      data G10  /  II,EOS,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,EOS,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,EOS,EOS,EOS,EOS,EOS,EOS,EOS,EOS /
c this group is P1bar
      integer G20(0:MAX_G,0:2)
      character*20 G20name /'P1bar'/
      data G20  /  II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS /
c this group is P2
      integer G30(0:MAX_G,0:2)
      character*20 G30name /'P2'/
      data G30  /  II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,II,EOS,EOS,EOS,EOS,EOS,EOS,EOS /
c this group is P21
      integer G40(0:MAX_G,0:2)
      character*20 G40name /'P21'/
      data G40  /  II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,TT,EOS,EOS,EOS,EOS,EOS,EOS,EOS /
c this group is Pm
      integer G50(0:MAX_G,0:2)
      character*20 G50name /'Pm'/
      data G50  /  II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,II,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,II,EOS,EOS,EOS,EOS,EOS,EOS,EOS /
c this group is Pb
      integer G60(0:MAX_G,0:2)
      character*20 G60name /'Pb'/
      data G60  /  II,II,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,TT,EOS,EOS,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,EOS,EOS,EOS,EOS,EOS,EOS,EOS /
c this group is P2/m
      integer G70(0:MAX_G,0:2)
      character*20 G70name /'P2overm'/
      data G70  /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,II,RR,EOS,EOS,EOS,EOS,EOS /
c this group is P2_1/m
```

```
      integer G80(0:MAX_G,0:2)
      character*20 G80name /'P21overm'/
      data G80  /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS/
c this group is P2_1/m  ???$$$ not a group (typo) but goes thru!
c      data G    /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
c     .             II,RR,RR,TT,EOS,EOS,EOS,EOS,EOS,
c     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS/
c this group is P2_1/m , in a different order
c      data G    /  II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS,
c     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
c     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS/
c this group is P2/b
      integer G90(0:MAX_G,0:2)
      character*20 G90name /'P2overb'/
      data G90  /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,II,RR,EOS,EOS,EOS,EOS,EOS /
c this group is P21/b
      integer G100(0:MAX_G,0:2)
      character*20 G100name /'P21overb'/
      data G100 /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS /
c this group is P222
      integer G110(0:MAX_G,0:2)
      character*20 G110name /'P222'/
      data G110 /  II,RR,II,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,II,RR,RR,EOS,EOS,EOS,EOS,EOS /
c this group is P2221
      integer G120(0:MAX_G,0:2)
      character*20 G120name /'P2221'/
      data G120 /  II,II,RR,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS /
c this group is P21212
      integer G130(0:MAX_G,0:2)
      character*20 G130name /'P21212'/
      data G130 /  II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,II,RR,RR,EOS,EOS,EOS,EOS,EOS /
c this group is P212121
      integer G140(0:MAX_G,0:2)
      character*20 G140name /'P212121'/
      data G140 /  II,SS,TT,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,TT,RR,SS,EOS,EOS,EOS,EOS,EOS /
c this group is Pmm2
      integer G150(0:MAX_G,0:2)
      character*20 G150name /'Pmm2'/
```

```
      data G150 /  II,RR,II,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,II,II,II,EOS,EOS,EOS,EOS,EOS/
c this group is Pmc21
      integer G160(0:MAX_G,0:2)

      character*20 G160name /'Pmc21'/
      data G160 /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,II,RR,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,II,TT,TT,EOS,EOS,EOS,EOS,EOS/
c this group is Pcc2
      integer G170(0:MAX_G,0:2)
      character*20 G170name /'Pcc2'/
      data G170 /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,II,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,II,TT,TT,EOS,EOS,EOS,EOS,EOS/
c this group is Pma2
      integer G180(0:MAX_G,0:2)
      character*20 G180name /'Pma2'/
      data G180 /  II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,II,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,II,II,II,EOS,EOS,EOS,EOS,EOS/
c this group is Pca21
      integer G190(0:MAX_G,0:2)
      character*20 G190name /'Pca21'/
      data G190 /  II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,II,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,TT,TT,II,EOS,EOS,EOS,EOS,EOS/
c this group is Pnc2
      integer G200(0:MAX_G,0:2)
      character*20 G200name /'Pnc2'/
      data G200 /  II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS,
     .             II,II,TT,TT,EOS,EOS,EOS,EOS,EOS/
c this group is Pmn21 ??? is this right?
c      integer G210(0:MAX_G,0:2)
c      character*20 G210name /'Pmn21'/
c      data G210 /  II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
c     .             II,RR,RR,II,EOS,EOS,EOS,EOS,EOS,
c     .             II,II,TT,TT,EOS,EOS,EOS,EOS,EOS/
c this group is Pmn21 ??? is this right?
      integer G210(0:MAX_G,0:2)
      character*20 G210name /'Pmn21'/
      data G210 /  II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,

     .             II,II,RR,RR,EOS,EOS,EOS,EOS,EOS,
     .             II,II,TT,TT,EOS,EOS,EOS,EOS,EOS/
c this group is Pba2
      integer G220(0:MAX_G,0:2)
      character*20 G220name /'Pba2'/
      data G220 /  II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS,
     .             II,II,II,II,EOS,EOS,EOS,EOS,EOS/
```

```
c this group is Pna21
      integer G230(0:MAX_G,0:2)
      character*20 G230name /'Pna21'/
      data G230 /  II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS,
     .             II,TT,TT,II,EOS,EOS,EOS,EOS,EOS/
c this group is Pnn2
      integer G240(0:MAX_G,0:2)
      character*20 G240name /'Pnn2'/
      data G240 /  II,RR,SS,TT,EOS,EOS,EOS,EOS,EOS,
     .             II,RR,TT,SS,EOS,EOS,EOS,EOS,EOS,
     .             II,II,TT,TT,EOS,EOS,EOS,EOS,EOS/
c this group is Pmmm
      integer G250(0:MAX_G,0:2)
      character*20 G250name /'Pmmm'/
      data G250 /  II,RR,II,RR,RR,II,RR,II,EOS,
     .             II,RR,RR,II,RR,II,II,RR,EOS,
     .             II,II,RR,RR,RR,RR,II,II,EOS/
c this group is Pnnn
      integer G260(0:MAX_G,0:2)
      character*20 G260name /'Pnnn'/
      data G260 /  II,RR,SS,TT,II,RR,SS,TT,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS,
     .             II,II,SS,SS,RR,RR,TT,TT,EOS/
c this group is Pccm
      integer G270(0:MAX_G,0:2)
      character*20 G270name /'Pccm'/
      data G270 /  II,RR,RR,II,RR,II,II,RR,EOS,
     .             II,RR,II,RR,RR,II,RR,II,EOS,
     .             II,II,TT,TT,RR,RR,SS,SS,EOS/
c this group is Pban
      integer G280(0:MAX_G,0:2)
      character*20 G280name /'Pban'/
      data G280 /  II,RR,SS,TT,II,RR,SS,TT,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS,
     .             II,II,RR,RR,RR,RR,II,II,EOS/
c this group is Pmma
      integer G290(0:MAX_G,0:2)
      character*20 G290name /'Pmma'/
      data G290 /  II,RR,SS,TT,RR,II,TT,SS,EOS,
     .             II,II,II,II,RR,RR,RR,RR,EOS,
     .             II,RR,II,RR,RR,II,RR,II,EOS/
c this group is Pnna
      integer G300(0:MAX_G,0:2)
      character*20 G300name /'Pnna'/
      data G300 /  II,SS,II,SS,RR,TT,RR,TT,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS,
     .             II,II,SS,SS,RR,RR,TT,TT,EOS/
c this group is Pmna
      integer G310(0:MAX_G,0:2)
      character*20 G310name /'Pmna'/
      data G310 /  II,II,SS,SS,RR,RR,TT,TT,EOS,
```

```
     .             II,RR,II,RR,RR,II,RR,II,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS/
c this group is Pcca
      integer G320(0:MAX_G,0:2)
      character*20 G320name /'Pcca'/
      data G320 /  II,SS,TT,RR,RR,TT,SS,II,EOS,
     .             II,RR,RR,II,RR,II,II,RR,EOS,
     .             II,II,SS,SS,RR,RR,TT,TT,EOS/
c this group is Pbam
      integer G330(0:MAX_G,0:2)
      character*20 G330name /'Pbam'/
      data G330 /  II,RR,TT,SS,RR,II,SS,TT,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS,
     .             II,II,RR,RR,RR,RR,II,II,EOS/
c this group is Pccn
      integer G340(0:MAX_G,0:2)
      character*20 G340name /'Pccn'/
      data G340 /  II,SS,TT,RR,RR,TT,SS,II,EOS,
     .             II,SS,RR,TT,RR,TT,II,SS,EOS,
     .             II,II,SS,SS,RR,RR,TT,TT,EOS /

c this group is Pbcm
      integer G350(0:MAX_G,0:2)
      character*20 G350name /'Pbcm'/
      data G350 /  II,RR,II,RR,RR,II,RR,II,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS,
     .             II,TT,RR,SS,RR,SS,II,TT,EOS/
c this group is Pnnm
      integer G360(0:MAX_G,0:2)
      character*20 G360name /'Pnnm'/
      data G360 /  II,RR,TT,SS,RR,II,SS,TT,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS,
     .             II,II,SS,SS,RR,RR,TT,TT,EOS/
c this group is Pmmn
      integer G370(0:MAX_G,0:2)
      character*20 G370name /'Pmmn'/
      data G370 /  II,RR,SS,SS,RR,II,TT,TT,EOS,
     .             II,RR,SS,TT,II,RR,TT,SS,EOS,
     .             II,II,RR,RR,II,II,RR,RR,EOS/
c this group is Pbcn
      integer G380(0:MAX_G,0:2)
      character*20 G380name /'Pbcn'/
      data G380 /  II,SS,TT,RR,RR,TT,SS,II,EOS,
     .             II,SS,SS,II,RR,TT,TT,RR,EOS,
     .             II,TT,RR,SS,RR,SS,II,TT,EOS/
c this group is Pbca
      integer G390(0:MAX_G,0:2)
      character*20 G390name /'Pbca'/
      data G390 /  II,TT,RR,SS,RR,SS,II,TT,EOS,
     .             II,SS,TT,RR,RR,TT,SS,II,EOS,
     .             II,RR,SS,TT,RR,II,TT,SS,EOS/
c this group is Pnma
```

```
      integer G400(0:MAX_G,0:2)
      character*20 G400name /'Pnma'/

      data G400 /  II,TT,RR,SS,RR,SS,II,TT,EOS,
     .             II,SS,TT,RR,RR,TT,SS,II,EOS,
     .             II,SS,RR,TT,RR,TT,II,SS,EOS/
      integer i,j,k,input
c
c
c
c  output main menu, get choice
1     continue
      write (6,*) '  1. P1       (1)      21. Pmn21 (31)'
      write (6,*) '  2. P1bar    (2)      22. Pba2  (32)'
      write (6,*) '  3. P2       (3)      23. Pna21 (33)'
      write (6,*) '  4. P21      (4)      24. Pnn2  (34)'
      write (6,*) '  5. Pm       (6)      25. Pmmm  (47)'
      write (6,*) '  6. Pc       (7)      26. Pnnn  (48)'
      write (6,*) '  7. P2/m    (10)      27. Pccm  (49)'
      write (6,*) '  8. P21/m   (11)      28. Pban  (50)'
      write (6,*) '  9. P2/c    (13)      29. Pmma  (51)'
      write (6,*) ' 10. P21/c   (14)      30. Pnna  (52)'
      write (6,*) ' 11. P222    (16)      31. Pmna  (53)'
      write (6,*) ' 12. P2221   (17)      32. Pcca  (54)'
      write (6,*) ' 13. P21212  (18)      33. Pbam  (55)'
      write (6,*) ' 14. P212121 (19)      34. Pccn  (56)'
      write (6,*) ' 15. Pmm2    (25)      35. Pbcm  (57)'
      write (6,*) ' 16. Pmc21   (26)      36. Pnnm  (58)'
      write (6,*) ' 17. Pcc2    (27)      37. Pmmn  (59)'
      write (6,*) ' 18. Pma2    (28)      38. Pbcn  (60)'
      write (6,*) ' 19. Pca21   (29)      39. Pbca  (61)'
      write (6,*) ' 20. Pnc2    (30)      40. Pnma  (62)'

      write (6,*)'     Which space group do you want? ==> '

      read(5,5,err=2,end=2000) input
5     format(i4)


      go to (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,
     .       160,170,180,190,200,210,220,230,240,250,

     .       260,270,280,290,300,310,320,330,340,350,
     .       360,370,380,390,400) input

2     write (6,*) 'input out of range, type <RET> to try again'
      read(5,*)
      go to 1
c
c exit
2000  stop

10    continue
```

```
      call gcopy (G10,gp)
      gpname(1:20) = G10name(1:20)
      go to 1000
20    continue
      call gcopy (G20,gp)
      gpname(1:20) = G20name(1:20)
      go to 1000
30    continue
      call gcopy (G30,gp)
      gpname(1:20) = G30name(1:20)
      go to 1000
40    continue
      call gcopy (G40,gp)
      gpname(1:20) = G40name(1:20)
      go to 1000
50    continue
      call gcopy (G50,gp)
      gpname(1:20) = G50name(1:20)
      go to 1000
60    continue
      call gcopy (G60,gp)
      gpname(1:20) = G60name(1:20)
      go to 1000
70    continue
      call gcopy (G70,gp)
      gpname(1:20) = G70name(1:20)
      go to 1000
80    continue
      call gcopy (G80,gp)
      gpname(1:20) = G80name(1:20)
      go to 1000
90    continue
      call gcopy (G90,gp)
      gpname(1:20) = G90name(1:20)
      go to 1000
100   continue
      call gcopy (G100,gp)
      gpname(1:20) = G100name(1:20)
      go to 1000
110   continue
      call gcopy (G110,gp)
      gpname(1:20) = G110name(1:20)
      go to 1000
120   continue
      call gcopy (G120,gp)
      gpname(1:20) = G120name(1:20)
      go to 1000
130   continue
      call gcopy (G130,gp)
      gpname(1:20) = G130name(1:20)
      go to 1000
140   continue
```

```
      call gcopy (G140,gp)
      gpname(1:20) = G140name(1:20)
      go to 1000
150   continue
      call gcopy (G150,gp)
      gpname(1:20) = G150name(1:20)
      go to 1000
160   continue
      call gcopy (G160,gp)
      gpname(1:20) = G160name(1:20)
      go to 1000
170   continue
      call gcopy (G170,gp)

      gpname(1:20) = G170name(1:20)
      go to 1000
180   continue
      call gcopy (G180,gp)
      gpname(1:20) = G180name(1:20)
      go to 1000
190   continue
      call gcopy (G190,gp)
      gpname(1:20) = G190name(1:20)
      go to 1000
200   continue
      call gcopy (G200,gp)
      gpname(1:20) = G200name(1:20)
      go to 1000
210   continue
      call gcopy (G210,gp)
      gpname(1:20) = G210name(1:20)
      go to 1000
220   continue
      call gcopy (G220,gp)
      gpname(1:20) = G220name(1:20)
      go to 1000
230   continue
      call gcopy (G230,gp)
      gpname(1:20) = G230name(1:20)
      go to 1000
240   continue
      call gcopy (G240,gp)
      gpname(1:20) = G240name(1:20)
      go to 1000
250   continue
      call gcopy (G250,gp)
      gpname(1:20) = G250name(1:20)
      go to 1000
260   continue
      call gcopy (G260,gp)
      gpname(1:20) = G260name(1:20)
      go to 1000
```

```
270   continue
      call gcopy (G270,gp)
      gpname(1:20) = G270name(1:20)
      go to 1000
280   continue
      call gcopy (G280,gp)
      gpname(1:20) = G280name(1:20)
      go to 1000
290   continue
      call gcopy (G290,gp)
      gpname(1:20) = G290name(1:20)
      go to 1000
300   continue
      call gcopy (G300,gp)
      gpname(1:20) = G300name(1:20)
      go to 1000
310   continue
      call gcopy (G310,gp)
      gpname(1:20) = G310name(1:20)
      go to 1000
320   continue
      call gcopy (G320,gp)
      gpname(1:20) = G320name(1:20)
      go to 1000
330   continue
      call gcopy (G330,gp)
      gpname(1:20) = G330name(1:20)
      go to 1000
340   continue
      call gcopy (G340,gp)
      gpname(1:20) = G340name(1:20)
      go to 1000
350   continue
      call gcopy (G350,gp)
      gpname(1:20) = G350name(1:20)
      go to 1000
360   continue
      call gcopy (G360,gp)
      gpname(1:20) = G360name(1:20)
      go to 1000
370   continue
      call gcopy (G370,gp)
      gpname(1:20) = G370name(1:20)
      go to 1000
380   continue
      call gcopy (G380,gp)
      gpname(1:20) = G380name(1:20)
      go to 1000
390   continue
      call gcopy (G390,gp)
      gpname(1:20) = G390name(1:20)
      go to 1000
```

```
400   continue
      call gcopy (G400,gp)
      gpname(1:20) = G400name(1:20)
      go to 1000

1000  continue
      return
      end
```

5.8 MODULE: PUTS

```
    All of these subroutines put out a line of fortran code.
      subroutine put_f (lun,label,index)
      subroutine put_m (lun,label,index)
      subroutine put_mm (lun,label,index)
      subroutine put_fs (lun,label,index)
      subroutine put_ms1 (lun,label,index)
      subroutine put_ms2 (lun,label,index)
      subroutine put_fd_ff (lun,label,index)
      subroutine put_m0 (lun,label,index)
      subroutine put_m1 (lun,label,index)
      subroutine put_m2 (lun,label,index)
      subroutine put_tt (lun,label,index)
      subroutine put_tmt (lun,label,index)
      subroutine put_fd_rr (lun,label,index)
      subroutine put_xx (lun,label,index)
      subroutine put_M3 (lun,label,index)
      subroutine put_M4 (lun,label,index)
      subroutine put_M5 (lun,label,index)
      subroutine put_M6 (lun,label,index)
      subroutine put_M7 (lun,label,index)
      subroutine put_cont (lun,label)
      subroutine put_fteven (lun)
      subroutine put_fteven2 (lun)
      subroutine put_ftshalf (lun)
      subroutine put_ftslov2 (lun)
      subroutine put_ftahalf (lun)
      subroutine put_ftalov2 (lun)
      subroutine put_ft21 (lun)
      subroutine put_ft2136 (lun)
      subroutine put_ft2157 (lun)
      subroutine put_ft2137 (lun)
      subroutine put_ft1 (lun)
      subroutine put_head(lun)
      subroutine put_head2(lun)
      subroutine put_end (lun)
      subroutine put_if (lun,num,xy_flg)
      subroutine put_else (lun)
      subroutine put_endif (lun)
      subroutine put_call (lun,piece)
      subroutine put_call2 (lun,piece,ft_type)
*******************************
```

```
c
c
c
c  loops that come up in xform routines

c F:
c      do i = 0,N/2,N/2
       subroutine put_f (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 0,N/2,N/2')
       return
       end

c M:
c      do i = 1,N/2-1
       subroutine put_m (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 1,N/2-1')
       return
       end

c GM:
c      do i = N/2+1,N-1
       subroutine put_mm (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = N/2+1,N-1')
       return
       end

c FS:
c      do i = N/4,3*N/4,N/2
       subroutine put_fs (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = N/4,3*N/4,N/2')
       return
       end

c MS:
c      do i = 0,N/4-1
```

```
c and
c      do i = N/2+1,3*N/4-1
       subroutine put_ms1 (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 0,N/4-1')
       return
       end
       subroutine put_ms2 (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = N/2+1,3*N/4-1')
       return
       end

c FT21 loop = fd_ff fundamental domain for full action
c      do i = 0,N/4
       subroutine put_fd_ff (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 0,N/4')
       return
       end

c M0
c      do i = 0,0
       subroutine put_m0 (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 0,0')
       return
       end

c M1
c      do i = 1,N/4-1
       subroutine put_m1 (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 1,N/4-1')
       return
       end
```

```
c M2
c      do i = N/4,N/4
       subroutine put_m2 (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = N/4,N/4')
       return
       end

c TT
c      do i = 0,N/2-1
       subroutine put_tt (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 0,N/2-1')
       return
       end

c TMT
c      do i = N/2,N-1
       subroutine put_tmt (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = N/2,N-1')
       return
       end

c FD_RR
c      do i = 0,N/2
       subroutine put_fd_rr (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 0,N/2')
       return
       end

c XX
c      do i = 0,N-1
       subroutine put_xx (lun,label,index)
       implicit none
       integer lun,label
       character index
       write (lun,100)label,index
100    format ('      do ',i3,' ',a,' = 0,N-1')
```

```
      return
      end

c M3
c      do i = N/4+1,N/2-1
      subroutine put_M3 (lun,label,index)
      implicit none
      integer lun,label
      character index
      write (lun,100)label,index
100   format ('      do ',i3,' ',a,' = N/4+1,N/2-1')
      return
      end

c M4
c      do i = N/2,N/2
      subroutine put_M4 (lun,label,index)
      implicit none
      integer lun,label
      character index
      write (lun,100)label,index
100   format ('      do ',i3,' ',a,' = N/2,N/2')
      return
      end

c M5
c      do i = N/2+1,3N/4 - 1
      subroutine put_M5 (lun,label,index)
      implicit none
      integer lun,label
      character index
      write (lun,100)label,index
100   format ('      do ',i3,' ',a,' = N/2+1,3*N/4 - 1')
      return
      end

c M6
c      do i = 3N/4,3N/4
      subroutine put_M6 (lun,label,index)
      implicit none
      integer lun,label
      character index
      write (lun,100)label,index
100   format ('      do ',i3,' ',a,' = 3*N/4,3*N/4')
      return
      end

c M7
c      do i = 3N/4 + 1, N-1
      subroutine put_M7 (lun,label,index)
      implicit none
      integer lun,label
```

```
      character index
      write (lun,100)label,index
100   format ('      do ',i3,' ',a,' = 3*N/4 + 1,N-1')
      return
      end

c
c 10    continue
      subroutine put_cont (lun,label)
      implicit none

      integer lun,label
      write (lun,100)label
100   format (i3,'   continue')
      return
      end

c*********************************************************************
c
c
c           routines follow to put out things like
c
c              call fteven (n,x(0,i,j),y(0,i,j))
c
c*********************************************************************
      subroutine put_fteven (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call fteven (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_fteven2 (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call fteven2 (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_ftshalf (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call ftshalf (n,x(0,i,j),y(0,i,j))' )
      return
      end


      subroutine put_ftsiov2 (lun)
      implicit none
      integer lun
```

```
      write (lun,100)
100   format ('         call ftslov2 (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_ftahalf (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call ftahalf (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_fta1ov2 (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call fta1ov2 (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_ft21 (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call ft21 (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_ft2135 (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call ft2135 (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_ft2157 (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call ft2157 (n,x(0,i,j),y(0,i,j))' )
      return
      end

      subroutine put_ft2137 (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call ft2137 (n,x(0,i,j),y(0,i,j))' )
      return
      end
```

```
      subroutine put_ft1 (lun)
      implicit none
      integer lun
      write (lun,100)
100   format ('         call ft1 (n,x(0,i,j),y(0,i,j))' )
      return
      end

c*************************************************************
c
c*************************************************************
      subroutine put_head(lun)
      implicit none
      integer lun
      write (lun,100)
100   format('      implicit none',/,
     .       '      include ''dimension.''',/,
     .       '      complex x(0:n-1,0:n-1,0:n-1)',/,
     .       '      complex y(0:n-1,0:n-1,0:n-1)',/,
     .       '      integer i,j,k')

      return
      end
c*************************************************************
c
c   this header is for expand routines
c
c*************************************************************
      subroutine put_head2(lun)
      implicit none
      integer lun
      write (lun,100)
100   format('      implicit none',/,
     .       '      include ''dimension.''',/,
     .       '      complex x(0:n-1,0:n-1,0:n-1)',/,
     .       '      integer i,j,k')
      return
      end
c*************************************************************
c
c*************************************************************
      subroutine put_end (lun)
      implicit none
      integer lun
      write (lun,100)
100   format('      return',/,
     .       '      end')
      return
      end
c*************************************************************
c
c*************************************************************
```

```
      subroutine put_if (lun,num,xy_flg)
      implicit none
      integer lun,num,xy_flg
      character*31 ifmods (0:2)
      data ifmods       /'      if (mod(j,2).eq.0) then',
     .                  '      if (mod(i,2).eq.0) then',
     .                  '      if (mod(i+j,2).eq.0) then' /

      if (num.eq.2) then
         write (lun,100)
100      format('      if (mod(i,2).eq.0) then')
      else if (num.eq.3) then
         write (lun,200) ifmods(xy_flg-1)
200      format(a31)
      endif
      return
      end
c*************************************************************
c
c*************************************************************
      subroutine put_else (lun)
      implicit none
      integer lun
      write (lun,100)
100   format('      else')
      return
      end
c*************************************************************
c
c*************************************************************
      subroutine put_endif (lun)
      implicit none
      integer lun
      write (lun,100)
100   format('      endif')
      return
      end
c*************************************************************
c
c
c
c*************************************************************
      subroutine put_call (lun,piece)
      implicit none
      include 'params.'
      include 'pieces.'

      integer piece,lun
c
c
c  we're assuming that FR means cosine
c                      FS means shalf
c                      M0 means ft21
```

```
c                       MT means ftahalf

      if (piece .eq. FR) then
         call put_fteven(lun)
      else if (piece .eq. FS) then
         call put_ftshalf(lun)
      else if (piece .eq. M0) then
         call put_ft21(lun)
      else if (piece .eq. NN) then
         call put_ft1(lun)
      else if (piece .eq. MT) then
         call put_ftahalf(lun)
      else
         write (6,10) piece
10       format(ix,'error in put_call, piece = ',i5)
         stop
      endif
      return
      end
c*************************************************************
c
c     this routine prints out calls to FT's with phase shifts
c      those used after translational symmetries.
c
c*************************************************************
      subroutine put_call2 (lun,piece,ft_type)
      implicit none
      include 'params.'
      include 'pieces.'
      integer piece,lun
      character*2 ft_type
c
c
c  we're assuming that FR means cosine
c                       FS means shalf
c                       M0 means ft21
c                       MT means ftahalf

      if (piece .eq. FR) then
         call put_fteven2(lun)
      else if (piece .eq. FS) then
         call put_ftsiov2(lun)
      else if (piece .eq. M0) then
         if (ft_type .eq. '35')then
            call put_ft2135(lun)
         else if (ft_type .eq. '37') then
            call put_ft2137(lun)
         else if (ft_type .eq. '57') then
            call put_ft2157(lun)
         endif
      else if (piece .eq. NN) then
         call put_ft1(lun)
```

```
      else if (piece .eq. MT) then
         call put_ftaiov2(lun)
      else
         write (6,10) piece
10       format(1x,'error in put_call, piece = ',i5)
         stop
      endif
      return
      end
c*******************************************************
c
c
c      z_to_x (gp,gp2)  permute Y Z X to X Y Z
c
c*******************************************************
      subroutine z_to_x (gp,gp2)
      implicit none
      include 'params.'

      include 'pieces.'
      integer i
      integer gp(0:MAX_G,0:2)
      integer gp2(0:MAX_G,0:2)


      do 10 i = 0, ordg
         gp2 (i,XCOORD) = gp (i,ZCOORD)
         gp2 (i,YCOORD) = gp (i,XCOORD)
         gp2 (i,ZCOORD) = gp (i,YCOORD)
10    continue

      return
      end
```

Bibliography

[1] L. Auslander and M. Shenefelt. Fourier transforms that respect crystallographic symmetries. *IBM Journal of Research and Development*, 31(2):213–223, March 1987.

[2] Martin J. Buerger. *Contemporary Crystallography*. McGraw Hill, Inc., New York, N.Y., 1970.

[3] Charles W. Curtis and Irving Reiner. *Representation Theory of Finite Groups and Associate Algebras*. Interscience Publishers, New York, N.Y., 1962.

[4] Lynn F.Ten Eyck. Crystallographic fast fourier transforms. *Acta Cryst.*, A29:183–191, 1973.

[5] Daniel R. Farkas. Crstallographic groups and their mathematics. *Rocky Mountain Journal of Mathematics*, 11(4):551 – 551, Fall 1981.

[6] Norman F.M. Henry and Kathleen Lonsdale, editors. *International Tables for X-Ray Crstallography*. Volume I. Symmetry Groups, The Kynoch Press, Birmingham, England, 1952.

[7] I. N. Herstein. *Topics in Algebra*. Xerox College Publishing, Waltham, Mass., 1964.

[8] Serge Lang. *Algebra*. Addison-Wesley Publishing Co., Inc., Reading, Massachusetts, 1965.

[9] L.Auslander, Myoung Shenefelt, and Michael Cook. Computational methods for crystallographic groups. 1989. Submitted for publication.

[10] Myoung Shenefelt. *Group Invariant Finite Fourier Transforms*. PhD thesis, The Graduate Center of City University of New York, 1988.