



Wir schaffen Wissen – heute für morgen

Paul Scherrer Institut

Michael Wedel

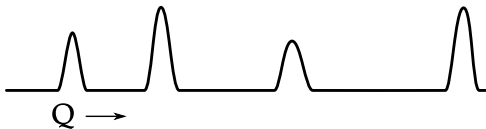
Crystallographic code in Mantid

Original problem

- POLDI: Engineering diffractometer at SINQ/PSI
- Typical samples: Metals, alloys, sometimes with impurities
- Peaks must be indexed for analysis

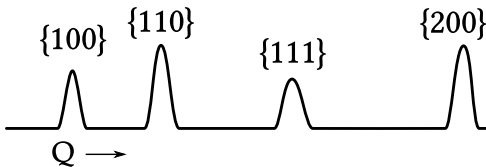
Original problem

- POLDI: Engineering diffractometer at SINQ/PSI
- Typical samples: Metals, alloys, sometimes with impurities
- Peaks must be indexed for analysis



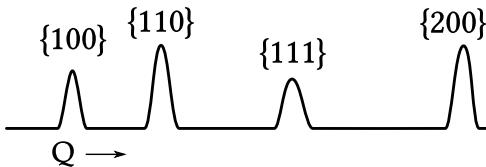
Original problem

- POLDI: Engineering diffractometer at SINQ/PSI
- Typical samples: Metals, alloys, sometimes with impurities
- Peaks must be indexed for analysis



Original problem

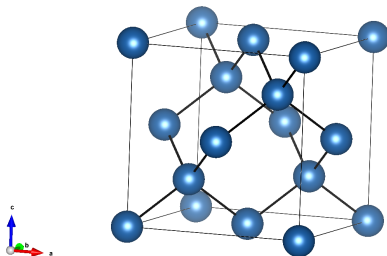
- POLDI: Engineering diffractometer at SINQ/PSI
- Typical samples: Metals, alloys, sometimes with impurities
- Peaks must be indexed for analysis



Boundary conditions

- Crystal structure is known
- Powder diffraction - can not distinguish symmetrically equivalent reflections
- Special reflection conditions should be taken into account

Crystal structure of Silicon



- Space group: $Fd\bar{3}m$
- Unit cell: $a = 5.431$
- Atomic positions:

Element	x	y	z	Occupancy
Si	0	0	0	1.0

Generating Miller indices

```
std::vector<V3D> getHKLs(const UnitCell &cell,
                        double dMin, double dMax) {
    std::vector<V3D> hkl;

    int hMax = static_cast<int>(cell.a() / dMin);
    int kMax = static_cast<int>(cell.b() / dMin);
    int lMax = static_cast<int>(cell.c() / dMin);

    for (int h = -hMax; h <= hMax; ++h) {
        for (int k = -kMax; k <= kMax; ++k) {
            for (int l = -lMax; l <= lMax; ++l) {
                V3D hkl(h, k, l);

                if ( isAllowed(hkl) && inRange(cell.d(hkl), dMin, dMax) ) {
                    hkl.push_back(hkl);
                }
            }
        }
    }

    return hkl;
}
```

Getting independent reflections

- hkl denotes normal vector to a family of lattice planes
- Crystal symmetry makes certain plane families equivalent
- Cubic: $(100), (010), (001), (\bar{1}00), (0\bar{1}0), (00\bar{1}) \rightarrow \{100\}$

- hkl denotes normal vector to a family of lattice planes
- Crystal symmetry makes certain plane families equivalent
- Cubic: $(100), (010), (001), (\bar{1}00), (0\bar{1}0), (00\bar{1}) \rightarrow \{100\}$
- Could be handled by limiting indices ($h > 0, k < h, l < k, \dots$)

Getting independent reflections

- hkl denotes normal vector to a family of lattice planes
- Crystal symmetry makes certain plane families equivalent
- Cubic: $(100), (010), (001), (\bar{1}00), (0\bar{1}0), (00\bar{1}) \rightarrow \{100\}$
- Could be handled by limiting indices ($h > 0, k < h, l < k, \dots$)
- But: Very inconvenient (many different cases)

- hkl denotes normal vector to a family of lattice planes
- Crystal symmetry makes certain plane families equivalent
- Cubic: $(100), (010), (001), (\bar{1}00), (0\bar{1}0), (00\bar{1}) \rightarrow \{100\}$
- Could be handled by limiting indices ($h > 0, k < h, l < k, \dots$)
- But: Very inconvenient (many different cases)

Describe symmetry by point groups

What's a group?

A group (G, \cdot) is a set G with a binary operation \cdot that satisfies the four group axioms.¹

1. *Closure*: For all a, b in G the result of $a \cdot b$ is also in G .

¹ <https://sites.google.com/a/thewe.net/mathematics/Motivation/What-are-groups->

What's a group?

A group (G, \cdot) is a set G with a binary operation \cdot that satisfies the four group axioms.¹

1. *Closure*: For all a, b in G the result of $a \cdot b$ is also in G .
2. *Associativity*: For all a, b, c in G , $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

¹ <https://sites.google.com/a/thewe.net/mathematics/Motivation/What-are-groups->

What's a group?

A group (G, \cdot) is a set G with a binary operation \cdot that satisfies the four group axioms.¹

1. *Closure*: For all a, b in G the result of $a \cdot b$ is also in G .
2. *Associativity*: For all a, b, c in G , $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
3. *Identity element*: There exists an element e in G such that $e \cdot a = a$.

¹ <https://sites.google.com/a/thewe.net/mathematics/Motivation/What-are-groups->

What's a group?

A group (G, \cdot) is a set G with a binary operation \cdot that satisfies the four group axioms.¹

1. *Closure*: For all a, b in G the result of $a \cdot b$ is also in G .
2. *Associativity*: For all a, b, c in G , $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
3. *Identity element*: There exists an element e in G such that $e \cdot a = a$.
4. *Inverse element*: For each a in G , there exists an element b in G such that $a \cdot b = e$.

¹ <https://sites.google.com/a/thewe.net/mathematics/Motivation/What-are-groups->

A symmetry operation maps a geometric object x onto another object x' .

Symmetry groups

A symmetry operation maps a geometric object x onto another object x' .

- Represented by a matrix/vector pair:

$$S = (\mathbf{W}, \mathbf{w})$$

Symmetry groups

A symmetry operation maps a geometric object x onto another object x' .

- Represented by a matrix/vector pair:

$$S = (\mathbf{W}, \mathbf{w})$$

- The group operation \cdot is defined as follows:

$$S_1 \cdot S_2 = (\mathbf{W}_1 \cdot \mathbf{W}_2, (\mathbf{W}_1 \cdot \mathbf{w}_2) + \mathbf{w}_1)$$

A symmetry operation maps a geometric object x onto another object x' .

- Represented by a matrix/vector pair:

$$S = (\mathbf{W}, \mathbf{w})$$

- The group operation \cdot is defined as follows:

$$S_1 \cdot S_2 = (\mathbf{W}_1 \cdot \mathbf{W}_2, (\mathbf{W}_1 \cdot \mathbf{w}_2) + \mathbf{w}_1)$$

- Hermann-Mauguin notation:

$$1, \bar{1}, 2, 2_1, \bar{6}, m, \dots$$

A symmetry operation maps a geometric object x onto another object x' .

- Represented by a matrix/vector pair:

$$S = (\mathbf{W}, \mathbf{w})$$

- The group operation \cdot is defined as follows:

$$S_1 \cdot S_2 = (\mathbf{W}_1 \cdot \mathbf{W}_2, (\mathbf{W}_1 \cdot \mathbf{w}_2) + \mathbf{w}_1)$$

- Hermann-Mauguin notation:

$$1, \bar{1}, 2, 2_1, \bar{6}, m, \dots$$

- Jones-faithful notation:

$$(x, y, z), (\bar{x}, \bar{y}, \bar{z}), (\bar{x}, \bar{y}, z), (\bar{x}, \bar{y}, z + 1/2), (y - x, \bar{x}, \bar{z}), (x, y, \bar{z})$$

Example: $2/m$

Example: $2/m$

- Group elements: $G_{2/m} = \{1, \bar{1}, 2, m\}$

Example: $2/m$

- Group elements: $G_{2/m} = \{1, \bar{1}, 2, m\}$
- Group table:

	1	$\bar{1}$	2	m
1	1	$\bar{1}$	2	m
$\bar{1}$	$\bar{1}$	1	m	2
2	2	m	1	$\bar{1}$
m	m	2	$\bar{1}$	1

Example: $2/m$

- Group elements: $G_{2/m} = \{1, \bar{1}, 2, m\}$
- Group table:

	1	$\bar{1}$	2	m
1	1	$\bar{1}$	2	m
$\bar{1}$	$\bar{1}$	1	m	2
2	2	m	1	$\bar{1}$
m	m	2	$\bar{1}$	1

- Example group operation:

$$\begin{array}{c} 2 \\ \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array} = \begin{array}{c} m \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \end{array} \cdot \begin{array}{c} \bar{1} \\ \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \end{array}$$

Generating point groups

²Shmueli, U. Acta Crystallogr. A 40, 559–567 (1984).

Generating point groups

- A point group is a *cyclic group* or product of cyclic groups²

²Shmueli, U. Acta Crystallogr. A 40, 559–567 (1984).

Generating point groups

- A point group is a *cyclic group* or product of cyclic groups²
- Order k of a symmetry operation:

$$S^k = I$$

²Shmueli, U. Acta Crystallogr. A 40, 559–567 (1984).

Generating point groups

- A point group is a *cyclic group* or product of cyclic groups²
- Order k of a symmetry operation:

$$S^k = I$$

- Cyclic group:

$$C = \{S^{k-1}, \dots, S^1, S^0 = I\}$$

²Shmueli, U. Acta Crystallogr. A 40, 559–567 (1984).

Generating point groups

- A point group is a *cyclic group* or product of cyclic groups²
- Order k of a symmetry operation:

$$S^k = I$$

- Cyclic group:

$$C = \{S^{k-1}, \dots, S^1, S^0 = I\}$$

- Product of two groups is again a group:

$$G_{2/m} = G_2 \cdot G_m = C_2 \cdot C_m = \{1, 2\} \cdot \{1, m\}$$

²Shmueli, U. Acta Crystallogr. A 40, 559–567 (1984).

Generating point groups

- A point group is a *cyclic group* or product of cyclic groups²
- Order k of a symmetry operation:

$$S^k = I$$

- Cyclic group:

$$C = \{S^{k-1}, \dots, S^1, S^0 = I\}$$

- Product of two groups is again a group:

$$G_{2/m} = G_2 \cdot G_m = C_2 \cdot C_m = \{1, 2\} \cdot \{1, m\}$$

All point groups can be generated using symmetry operations!

²Shmueli, U. Acta Crystallogr. A 40, 559–567 (1984).

- Represented by Jones-faithful notation (x, y, z)

Geometry::SymmetryOperation

- Represented by Jones-faithful notation (x, y, z)
- Supports rational vector components $(x + 1/2, z, -y)$

Geometry::SymmetryOperation

- Represented by Jones-faithful notation (x, y, z)
- Supports rational vector components $(x + 1/2, z, -y)$

Geometry::SymmetryOperation

- Represented by Jones-faithful notation (x, y, z)
- Supports rational vector components ($x + 1/2, z, -y$)
- Operations on V3D, V3R (3×1 -vector of rational numbers)

```
SymmetryOperation inv = SymmetryOperationFactory::Instance()
                        .createSymOp("-x,-y,-z");
SymmetryOperation mir = SymmetryOperationFactory::Instance()
                        .createSymOp("x,y,-z");
SymmetryOperation twoFoldA = mir * inv;
SymmetryOperation twoFoldB = inv * mir;

assert(twoFoldA == twoFoldB);

// Apply two-fold rotation to V3D
V3D rotated = twoFold * V3D(1,1,1);

// Create several operations at once
std::vector<SymmetryOperation> ops =
    SymmetryOperationFactory::Instance()
    .createSymOps("x,y,z; z,x,y");
```

Geometry::PointGroup

- Was introduced in 2012 for checking equivalence of hkl s

Geometry::PointGroup

- Was introduced in 2012 for checking equivalence of hkl s
- Refactored to use group-theoretical code
- New functionality added

- Was introduced in 2012 for checking equivalence of *hkl*s
- Refactored to use group-theoretical code
- New functionality added

```
PointGroup_sptr m3m = PointGroupFactory::Instance()  
                    .createPointGroup("m-3m");  
  
std::vector<V3D> equivalents = m3m->getEquivalents(V3D(1,0,0));  
  
// Vector contains all equivalent hkl's for (100)  
assert(equivalents.size() == 6);  
  
V3D family100 = m3m->getReflectionFamily(V3D(1,0,0));  
V3D family010 = m3m->getReflectionFamily(V3D(0,1,0));  
  
// Returns the same value for two equivalent reflections  
assert(family100 == family010);
```

Generating independent reflections

```
std::vector<V3D> getHKLs(const UnitCell &cell,
                        const PointGroup_sptr &pointGroup,
                        double dMin, double dMax) {
    // set instead of vector
    std::set<V3D> hkls;

    // lots of nested for loops...

    if ( isAllowed(hkl)
        && inRange(cell.d(hkl), dMin, dMax) ) {
        // Generate reflection family from hkl
        V3D unique = pointGroup->getReflectionFamily(hkl);
        hkls.insert(unique);
    }

    // closing loops...

    return std::vector<V3D>(hkls.begin(), hkls.end());
}
```

Generating independent reflections

```
std::vector<V3D> getHKLs(const UnitCell &cell,
                        const PointGroup_sptr &pointGroup,
                        double dMin, double dMax) {
    // set instead of vector
    std::set<V3D> hkls;

    // lots of nested for loops...

    if ( isAllowed(hkl)
        && inRange(cell.d(hkl), dMin, dMax) ) {
        // Generate reflection family from hkl
        V3D unique = pointGroup->getReflectionFamily(hkl);
        hkls.insert(unique);
    }

    // closing loops...

    return std::vector<V3D>(hkls.begin(), hkls.end());
}
```

Set of independent reflections with Geometry::PointGroup

Problem solved?

- Space group $Fd\bar{3}m$ - face centered lattice
- Checking allowed reflections with
`Geometry::ReflectionCondition?`

Problem solved?

- Space group $Fd\bar{3}m$ - face centered lattice
- Checking allowed reflections with
Geometry::ReflectionCondition?
- F allows (222) ($h + k = 2n, h + l = 2n, k + l = 2n$)

Problem solved?

- Space group $Fd\bar{3}m$ - face centered lattice
- Checking allowed reflections with
Geometry::ReflectionCondition?
- F allows (222) ($h + k = 2n, h + l = 2n, k + l = 2n$)
- Problem: Silicon atoms on Wyckoff-position $8a$

8	b	-4	3	m	$1/2, 1/2, 1/2$	$1/4, 3/4, 1/4$	$hkl :$
8	a	-4	3	m	$0, 0, 0$	$3/4, 1/4, 3/4$	$h = 2n + 1$ or $h + k + l = 4n$

Problem solved?

- Space group $Fd\bar{3}m$ - face centered lattice
- Checking allowed reflections with
Geometry::ReflectionCondition?
- F allows (222) ($h + k = 2n, h + l = 2n, k + l = 2n$)
- Problem: Silicon atoms on Wyckoff-position $8a$

8	b	-4 3 m	1/2, 1/2, 1/2	1/4, 3/4, 1/4	hkl :
8	a	-4 3 m	0, 0, 0	3/4, 1/4, 3/4	$h = 2n + 1$ or $h + k + l = 4n$

$2 + 2 + 2 = 6 \neq 4n - (222)$ is forbidden!

- Structure factor amplitudes for each hkl :

$$F(hkl) = \sum_j b_j \cdot \exp [2\pi i \cdot (hx_j + ky_j + lz_j)]$$

- Sum over all atoms j *in the unit cell*

- Structure factor amplitudes for each hkl :

$$F(hkl) = \sum_j b_j \cdot \exp [2\pi i \cdot (hx_j + ky_j + lz_j)]$$

- Sum over all atoms j *in the unit cell*
- Si atom at 0,0,0 is mapped to 8 symmetrically equivalent positions:

$$\begin{array}{cccc} 0, 0, 0 & 0, 1/2, 1/2 & 1/2, 0, 1/2 & 1/2, 1/2, 0 \\ 3/4, 1/4, 3/4 & 3/4, 3/4, 1/4 & 1/4, 1/4, 1/4 & 1/4, 3/4, 3/4 \end{array}$$

- Structure factor amplitudes for each hkl :

$$F(hkl) = \sum_j b_j \cdot \exp [2\pi i \cdot (hx_j + ky_j + lz_j)]$$

- Sum over all atoms j *in the unit cell*
- Si atom at 0,0,0 is mapped to 8 symmetrically equivalent positions:

$$\begin{array}{cccc} 0, 0, 0 & 0, 1/2, 1/2 & 1/2, 0, 1/2 & 1/2, 1/2, 0 \\ 3/4, 1/4, 3/4 & 3/4, 3/4, 1/4 & 1/4, 1/4, 1/4 & 1/4, 3/4, 3/4 \end{array}$$

- For (222) contributions cancel out – **not allowed!**

- Structure factor amplitudes for each hkl :

$$F(hkl) = \sum_j b_j \cdot \exp [2\pi i \cdot (hx_j + ky_j + lz_j)]$$

- Sum over all atoms j *in the unit cell*
- Si atom at 0,0,0 is mapped to 8 symmetrically equivalent positions:

$$\begin{array}{cccc} 0, 0, 0 & 0, 1/2, 1/2 & 1/2, 0, 1/2 & 1/2, 1/2, 0 \\ 3/4, 1/4, 3/4 & 3/4, 3/4, 1/4 & 1/4, 1/4, 1/4 & 1/4, 3/4, 3/4 \end{array}$$

- For (222) contributions cancel out – **not allowed!**
- Implemented in Geometry::BraggScatterer

Geometry::SpaceGroup

- Generation of equivalent coordinates in the unit cell
- Translational symmetry (screw axes, glide mirror planes, lattice centering)
- Exactly the same generation algorithm as for point groups if w is limited to interval $[0, 1)$
- Algorithmic generation only on first request, afterwards *Prototype* pattern

- Generation of equivalent coordinates in the unit cell
- Translational symmetry (screw axes, glide mirror planes, lattice centering)
- Exactly the same generation algorithm as for point groups if w is limited to interval $[0, 1)$
- Algorithmic generation only on first request, afterwards *Prototype* pattern

```
SpaceGroup_const_sptr Fd3m = SpaceGroupFactory::Instance()  
                           .createSpaceGroup("F d -3 m");  
  
// Si-atom at (0,0,0)  
std::vector<V3D> atoms = Fd3m * V3D(0,0,0);  
assert(atoms.size() == 8);
```

- Combines unit cell, space group and scatterers

- Combines unit cell, space group and scatterers

```
SpaceGroup_const_sptr Fd3m = SpaceGroupFactory::Instance()
                             .createSpaceGroup("F d -3 m");

BraggScatterer_sptr SiAtom =
    BraggScattererFactory::Instance()
        .createScatterer("IsotropicAtomBraggScatterer",
                         "Element=Si;U=0.05");

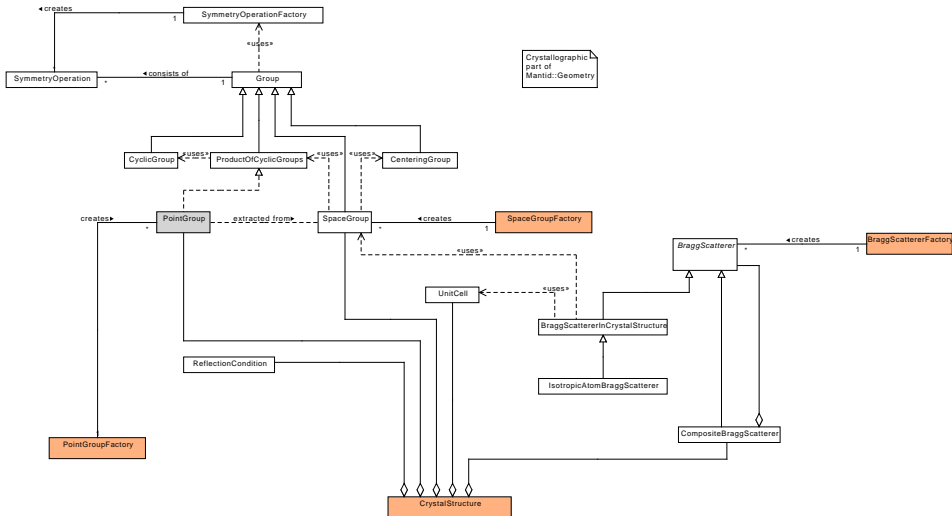
CompositeBraggScatterer_sptr basis =
    CompositeBraggScatterer::create(
        std::vector<BraggScatterer_sptr>(1, SiAtom));

UnitCell cell(5.431, 5.431, 5.431);

CrystalStructure Si(cell, Fd3m, basis);

// Get unique hkl's with correct absences
std::vector<V3D> hkl's = Si.getUniqueHKLs(0.75,10.0,
                                           CrystalStructure::UseStructureFactor);
// Based on centering only, we would get 22
assert(hkl's.size() == 15)
```

Class diagram of crystallographic code



What has been done?

- Extended existing point group code
- Added space groups based on group theoretical approach
- Introduced scatterers (composite patterns - very flexible)
- Crystal structure class
- Lots of unit tests and space group system test

What has been done?

- Extended existing point group code
- Added space groups based on group theoretical approach
- Introduced scatterers (composite patterns - very flexible)
- Crystal structure class
- Lots of unit tests and space group system test

Future improvements

- Refactor point group code
- Better factory for scatterers
- Export everything (?) to Python
- Ticket #10134 - Introduce *Reflection* concept?

- Help from the team: Anders, Andrei, Vicky (point group code & tests)
- Space group help: Dr. Igor Baburin (TU Dresden)
- Literature:
 - *International Tables for Crystallography Volume A*
 - *Symmetry Relationships between Crystal Structures: Applications of Crystallographic Group Theory in Crystal Chemistry* by Ulrich Müller (ISBN: 9780191648809)

Thank you for your attention!

Generates SpaceGroup prototype object on first call to getPrototype().

```

AbstractSpaceGroupGenerator
-m_prototype: SpaceGroup const_sptr
+getPrototype(): SpaceGroup const_sptr
-generateGroup(): Group const_sptr
  
```

Stores instances of AbstractSpaceGroupGenerator implementations which supplies prototype objects that are copied.

```

SpaceGroupFactoryImpl
-m_generatorMap: map<string, AbstractSpaceGroupGenerator_sptr>
+subscribeUsingGenerator<T>(...)
  
```

AlgorithmicSpaceGroupGenerator

Uses the algorithm described in paper by U. Shmueli from 1984 to generate space groups from generators in ITA.

TabulatedSpaceGroupGenerator

Generates a group from a list of symmetry operations without further computations.

```

DECLARE_GENERATED_SPACEGROUP(198, "P 21 3",
                               "-x+1/2,-y,z+1/2; -x,y+1/2,-z+1/2; z,x,y")
  
```