Brillouin Zone Reduction for Band Structure Integration

John Christensen

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Gus Hart, Adviser

Department of Physics and Astronomy

Brigham Young University

ABSTRACT

Brillouin Zone Reduction for Band Structure Integration

John Christensen
Department of Physics and Astronomy, BYU
Bachelor of Science

Materials discovery is bottle necked by limited computational resources. One common method of computing a materials properties, Density Functional Theory (DFT), requires that the electronic band structure of a materials to be integrated. The electronic band structure of metals is very difficult to perform because of the discontinuities introduced by the Fermi level. Currently the most commonly used integration technique is a simple Riemann sum. Using this method requires very dense sampling because the integral converges slowly. Significant effort has gone into finding integration techniques that converge more quickly. New integration techniques that rely on non-uniform sampling have been proposed, but are not able to take full advantage of conventional symmetry reduction techniques. To ensure optimal symmetry reduction, calculations should be performed in the symmetrically irreducible Brillouin zone (IBZ). We present an algorithm for finding the IBZ for an arbitrary lattice.

Keywords: Irreducible Brillouin Zone, Electronic Band Structure, Density Functional Theory, DFT

ACKNOWLEDGMENTS

I would like to thank the BYU Physics department for providing research funding. Thank you to Dr. Gus Hart for making time to regularly meet and answer questions. Thank you to my wife Alex for always being willing to help and listen to me think out loud.

Contents

Chapter 1

Introduction

1.1 Motivation

Materials are important enough to mankind that historical ages are named after them: stone, bronze, iron, and silicon. Being able to find new materials that are stronger, lighter, or have higher melting temperatures would benefit society as a whole. The Materials Simulation Group at Brigham Young University seeks to accelerate the prediction of new materials by creating computational models of materials instead of mixing elements them in the lab.

Elements of the periodic table can be combined in a practically infinite number of different ways. By creating a model of a given collection and concentration of elements its properties can be predicted using Density Functional Theory (DFT). One of the core calculations performed in DFT is an integration of a material's electronic band structure. By integrating the allowed energies of the electrons in material, the total energy is found. This calculation is performed numerically and converges to an accurate value very slowly. This means that a large number of sampling points must be used to achieve acceptable accuracy. At each sampling point the eigenvalues of the Hamiltonian must be calculated. Finding the eigenvalues is the most computationally time consuming step.

Graduate student Jeremy Jorgensen is working on a new integration technique that carefully selects the location of sampling points to reduce the total number sampling points required to achieve an acceptable accuracy.

Currently, integration techniques sample the band structure with a uniform grid of points. A denser grid increases accuracy. The problem with using denser and denser grids to increase accuracy is that the error in these integrals are not uniformly distributed. Some areas may be smooth and simple to integrate, meaning that increasing sampling densities in these regions is a waste of computation time. Because the electronic band structure of metals has discontinuities at the Fermi level, much of the integration error occurs around these discontinuities. Jorgensen's new techniques does not uniformly sample the band structure. Instead, regions of high error are sampled more densely. By increasing sampling points only in these areas of high error, accuracy is increased without requiring as many sampling points as uniform sampling techniques.

Non-uniform sampling breaks the symmetry reduction techniques that are normally used. These techniques take a grid of points and find points that are equivalent under symmetry. This reduces the number of points where eigenvalues need to be found and speeds up the computation. When points are sampled non-uniformly, it is unlikely that points will have symmetrically equivalent points. In order for the non-uniform grids to take full advantage of symmetry, points should be selected from a symmetrically unique region. This region is known as the Irreducible Brillouin zone (IBZ). This project's goal is to create an algorithm to find the IBZ for any lattice.

1.2 Lattices

Materials can often be modeled as a periodic arrangement of atoms. This periodic nature can be described mathematically as a lattice. A lattice is all linear combinations of linearly independent vectors with integer coefficients. An n-dimensional lattice is defined as

$$\Lambda = \left\{ \sum_{i=1}^{n} a_i v_i \mid a_i \in \mathbb{Z} \right\}.$$

The vectors v_i are known as lattice vectors. I will use 2-dimensional lattices as examples for simplicity, but all the work for this project was done in 3-dimensions.

Lattices repeat infinitely in all directions making it impossible to represent the entire lattice computationally. Because the lattice is periodic, it is sufficient to only look at one period of the lattice. The parallelogram (or parallelpiped in 3 dimensions) formed by the lattice vectors is one period of the lattice. This region is known as the primitive unit cell and is shown in Fig. 1.1. The entire lattice can be recreated by repeating the primitive unit cell in all directions.

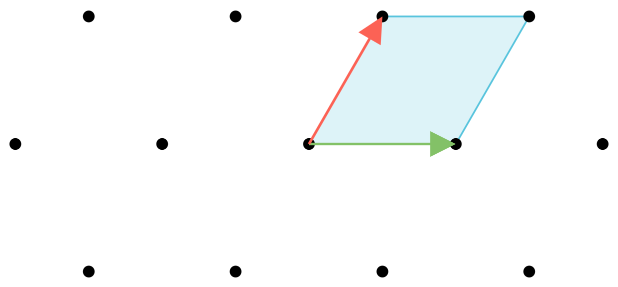
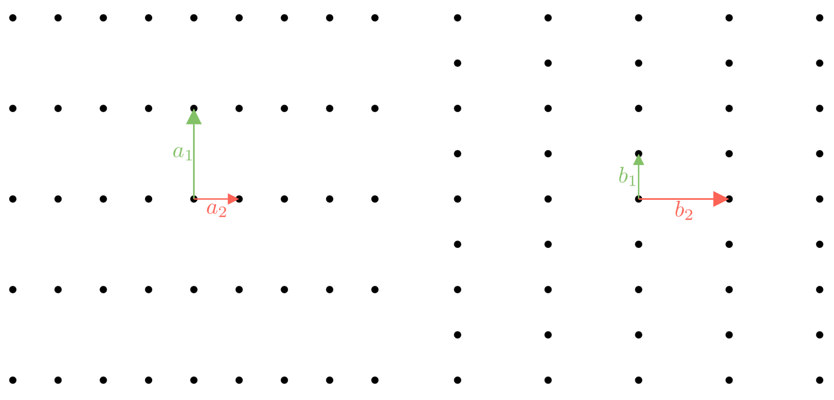


Figure 1.1 The unit cell of a hexagonal lattice

Similar to looking at the frequency space of an audio signal, looking at the Fourier transform of the lattice is useful. The Fourier transform of the lattice is also a lattice, but instead of existing in real space, it now exists in reciprocal space (also known as k-space or momentum space). The properties we are interested in are easily computed in reciprocal space.

Taking the Fourier transform of the 2-D real space lattice defined by lattice vectors $(\mathbf{a_1}, \mathbf{a_2})$ gives



(a) Real space **(b)** Reciprocal space

Figure 1.2 The real and reciprocal spaces for a rectangular lattice

the reciprocal lattice vectors $(\mathbf{b_1}, \mathbf{b_2})$

$$\mathbf{b_1} = 2\pi \frac{\mathbf{Ra_2}}{\mathbf{a_1} \cdot \mathbf{Ra_2}} \tag{1.1}$$

$$\mathbf{b_2} = 2\pi \frac{\mathbf{Ra_1}}{\mathbf{a_2} \cdot \mathbf{Ra_1}} \tag{1.2}$$

Where **R** is a 90° rotation.

Similarly in 3-D, the reciprocal lattice vectors are

$$\mathbf{b_1} = 2\pi \frac{\mathbf{a_2} \times \mathbf{a_3}}{\mathbf{a_1} \cdot (\mathbf{a_2} \times \mathbf{a_3})} \tag{1.3}$$

$$\mathbf{b_2} = 2\pi \frac{\mathbf{a_3} \times \mathbf{a_1}}{\mathbf{a_2} \cdot (\mathbf{a_3} \times \mathbf{a_1})} \tag{1.4}$$

$$\mathbf{b_3} = 2\pi \frac{\mathbf{a_1} \times \mathbf{a_2}}{\mathbf{a_3} \cdot (\mathbf{a_1} \times \mathbf{a_2})}. \tag{1.5}$$

Similar to the primitive unit cell in real space, there is a region of reciprocal space that contains all the information of the lattice. This region is known as the Brillouin zone.

1.3 The Brillouin zone

The Brillouin zone (BZ) is the region in reciprocal space where the closest lattice point is the origin (Fig. 1.3). Just like the unit cell in real space, the BZ covers one period of the reciprocal lattice.

Because calculations that are needed to predict a material's properties are done in reciprocal space, all calculations can be performed in just the BZ.

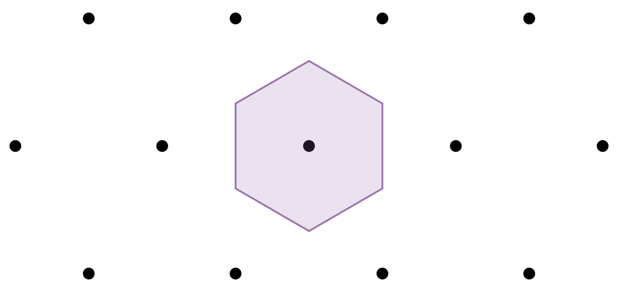


Figure 1.3 The Brillouin zone of a hexagonal lattice.

The BZ can be constructed by using a series of cutting planes. If the lattice is Minkowski reduced (the choice of basis vectors are as close to orthogonal as possible), then the BZ can be constructed using only the origin's nearest neighboring lattice points. By bisecting each line that connects the origin to its nearest neighbors, then taking the area enclosed by the bisections, the BZ can be found. This process is illustrated in Fig. 1.4.

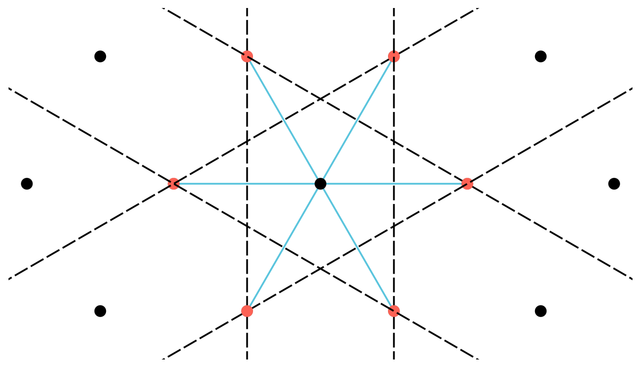


Figure 1.4 Construction of the BZ. The lines (blue) that connect the origin to its nearest neighbors (red) are bisected (black dashed). The area circumscribed by the dashed lines is the BZ.

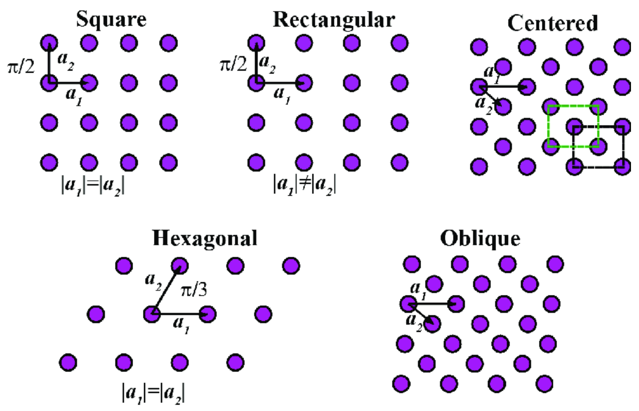


Figure 1.5 The five 2-dimensional Bravais lattices.

1.4 Symmetry

It is useful to categorize lattices based on their symmetry. For example, if a 2-dimensional lattice has two lattice vectors of equal length and an angle of $\frac{\pi}{3}$ between them, it is classified as hexagonal as seen in Fig. 1.5. In 2-dimensions any lattice can be categorized as one of the five Bravais lattices (Fig. 1.5). In 3-dimensions there are 14 Bravais lattices. Each Bravais lattice has a set of symmetries associated with the lattice.

To illustrate why symmetry is useful, consider performing the integral

$$\int_{-1}^{1} x^2 dx$$

numerically using the middle Riemann sum method. First a grid such as the one shown in Fig. 1.6 must be selected. When performing the numeric integral the function being integrated will be sampled at each grid point. Increasing the density of the grid increases the number of sampled points and the accuracy of the integral, but at the cost of computation time. By taking advantage of the symmetry of the problem it is possible to maintain accuracy while reducing the number of sampling points.

To numerically perform the integral using the middle Riemann sum, first create a grid of equally spaced points in the given range (Fig 1.6). Then evaluate the function at the sampling points, create

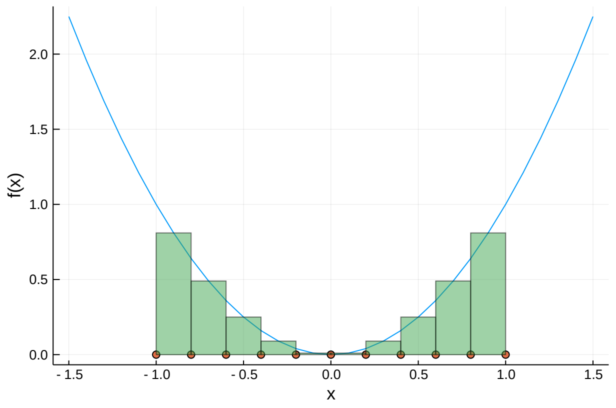


Figure 1.6 A simple middle Riemann sum numeric integration method.

the appropriate rectangles, and add up their areas. Using this method yields

$$\int_{-1}^{1} x^2 dx \approx \sum_{i=1}^{10} (x_i + \frac{\Delta x}{2})^2 \Delta x = 0.660$$

where x_i is the set of grid points and Δx is the width of each rectangle.

The error using this numeric integral is

$$\frac{2}{3} - 0.660 = 0.00\overline{66}.$$

This result was achieved using 10 sampling points, but by utilizing the symmetry of this function the same result can be found using only the 5 sampling points in the range 0 to 1 and multiplying the result by 2. Let x_i' be the smaller set of grid points.

$$2\int_{0}^{1} x^2 dx \approx 2\sum_{i=1}^{5} (x_i' + \frac{\Delta x}{2})^2 \Delta x = 0.660.$$

The same accuracy was achieved using half as many sampling points!

This case is trivial, and has very little symmetry (just one reflection) compared to most lattices. A cubic lattice, for example, has 48 symmetries. This means that by utilizing the symmetry of the problem, 48 times fewer points can be used to achieve the same result found without utilizing symmetry.

To utilize all the symmetries of a lattice, a precise definition of symmetry is needed. The symmetries of a lattice are the set of all operations (such as reflection or rotations) that when applied to the lattice makes it look the same as it did before the operation. For example, if the hexagonal lattice shown in Fig. 1.3 is rotated about the origin by 60° the lattice would look identical before and after the rotation. This means that a 60° rotation is one of the symmetric operators of a hexagonal lattice.

The set of all the symmetries of a lattice is known as the point group and contains both reflectional and rotational symmetries. One easily overlooked operator is the identity operator. The identity operator does nothing to the lattice. While it might seem that if it does nothing it shouldn't matter, we defined a symmetric operator as any operator that does not change the way the lattice looks. Applying the identity operator to the lattice does not change the lattice, so it must be included in the point group. By using all the symmetries in the point group, the number of sampling points required to achieve a given accuracy is reduced by a factor of N where N is the number of operations in the point group. The potential reduction factor for 3-dimensional lattices is given in Table 1.1.

Lattice	Volume Reduction	Bravais Lattices
Cubic	$\frac{1}{48}$	3
Hexagonal	$\frac{1}{24}$	1
Tetragonal	$\frac{1}{16}$	2
Rhombohedral	$\frac{1}{12}$	1
Orthorhombic	$\frac{1}{8}$	4
Monoclinic	$\frac{1}{4}$	2
Triclinic	$\frac{1}{2}$	1

Table 1.1 Volume reduction factor for lattices with differing amounts of symmetries.

By using the symmetries of the lattice, the BZ can be reduced to a smaller volume known as the

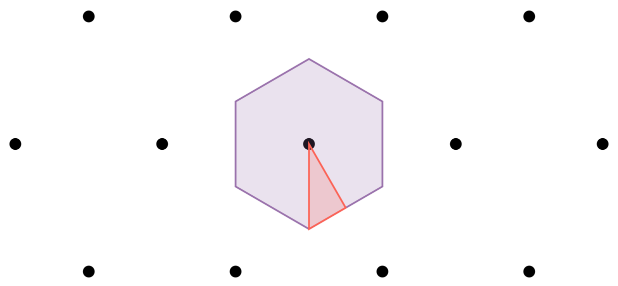


Figure 1.7 The Brillouin zone (purple) and Irreducible Brillouin zone (red) for a hexagonal lattice

Irreducible Brillouin zone (IBZ). The reduction in volume is

$$\frac{V_{\text{IBZ}}}{V_{\text{BZ}}} = \frac{1}{N}$$

where N is the order of the point group, V_{IBZ} is the volume of the IBZ, and V_{BZ} is the volume of the BZ. This means for a hexagonal lattice, the area of the IBZ is $\frac{1}{12}$ the area of the BZ as shown in Fig. 1.7. The goal of this project is to create an algorithm to find the IBZ for any given lattice.

IBZs for each Bravais lattice have been tabulated in the past [1, 2]. While it would be possible to use the tabulated results for any arbitrary lattice by transforming the lattice into a canonical representation, using the tabulated results, and finally transforming back, we created a more elegant approach to directly calculate the IBZ for any lattice. This algorithm works quickly so there is no need to use tabulated results.

Chapter 2

Methods

The algorithm takes the Brillouin zone and the point group of a lattice as inputs, and returns the Irreducible Brillouin zone. First a high level conceptual walk through is presented, and the in-depth implementation follows.

2.1 Algorithm Overview

The goal of the algorithm is to take the Brillouin zone and reduce it down to its symmetrically irreducible part (the IBZ). This is done by using each symmetry operation in the point group of the lattice to iteratively reduce the BZ down to its irreducible part.

Let the point group be represented as set of matrices $\mathbf{G}$. Let the Brillouin zone be represented as a polytope (an n-dimensional flat-sided shape) $\mathbf{P}$. Let $\mathbf{V}$ be the set of vertices of $\mathbf{P}$. For each operation $g \in \mathbf{G}$, perform the following steps:

1. Pick a point $v \in \mathbf{V}$, and transform it by multiplying it by g. Call this transformed point v'.

2. If the transformation does not move v (meaning $v = v'$), put g in a set of stabilizer operations $\mathbf{S}$ and return to step 1 (these are dealt with later).

3. Create a line segment that connects v to v'.

4. Create a cutting plane that bisects this line segment.

5. Reduce **P** by removing all parts **P** on the side of the plane closest to v'.

6. Repeat these steps with the next operation g, until each operation has been used.

The Brillouin zone is now partially reduced, but the stabilizer operations have not yet reduced the volume. Pick a new point $v \in \mathbf{V}$, and perform the above steps again but now only use the stabilizer operations $g \in \mathbf{S}$. Continue this process until the only element in **S** is the identity matrix (the identity matrix never moves a vertex v and is always considered a stabilizer). After this process, all of the symmetries of the BZ have been used to reduce the BZ and only the IBZ remains. A proof that this algorithm always gives the IBZ was created with help from Dr. Tyler Jarvis from BYU's Math Department and can be found in Appendix A.

2.2 Implementation

In order to put this high level algorithm into practice, it is necessary to determine how the BZ and partially reduced BZ are tracked during the algorithm. Two methods are used to represent these geometries: convex hulls and half-space representation.

2.2.1 Convex Hulls

A convex hull is used to represent the BZ. A convex hull is the smallest convex polytope (a polytope with all interior angles less than 180°) that entirely encloses a set of points. In two dimensions, the convex hull can be thought of as a rubber band stretched over a collection of nails in a board as shown in Fig. 2.1. In three dimensions, the convex hull can be pictured as a sheet being stretched over a cloud of points.

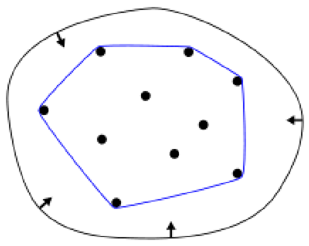
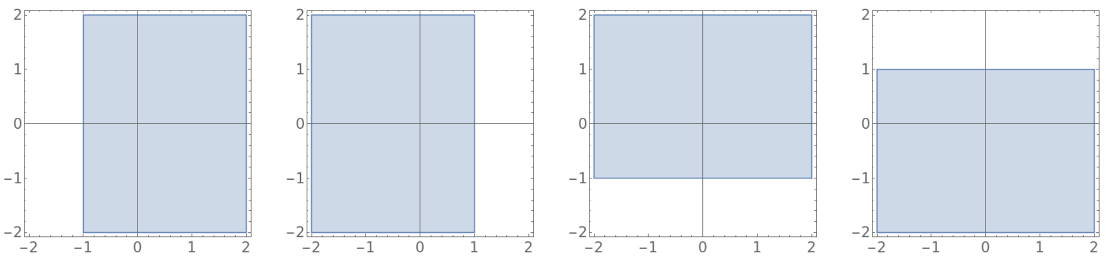


Figure 2.1 The 2-dimensional convex hull (blue) of a set of points. The black circle illustrates a rubber band constricting around the points, creating the convex hull.



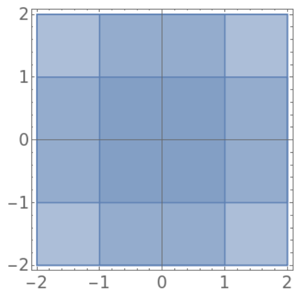
(a) $x > -1$ **(b)** $x < 1$ **(c)** $y > -1$ **(d)** $y < 1$

Figure 2.2 Examples of half-spaces in 2 dimensions

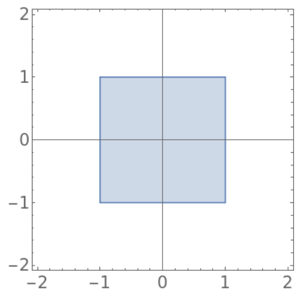
The BZ is guaranteed to be a convex hull because it is constructed by a series of cutting planes. This type of construction does not allow for the creation of any angles greater than greater 180°. Our code uses a julia wrapper of the qhull package [3]: a program written in C that makes it simple to calculate the convex hull from a collection of points. We use this representation to easily access the vertices of the BZ needed to perform the algorithm described in Section 2.1.

2.2.2 Half-space representation

The partially reduced BZ (The polytope that is being reduced throughout the algorithm) is represented as an intersection of half-spaces. A half-space is one of the two regions of space divided by a plane (or line for 2-dimensions) as shown in Fig. 2.2. An intersection of multiple half-spaces can represent any convex polytope as shown in Fig. 2.3. We use an intersection of half-spaces to represent our partially reduced BZ because it is simple to perform step 5 of the algorithm. To do the



(a) Half-spaces from Fig. 2.2 overlaid



(b) The intersection of half-spaces from Fig. 2.2

Figure 2.3 An illustration of how a square can be defined by an intersection of four half-spaces

reduction step, simply find the intersection of the partially reduced BZ and the half-space found in step 4.

Without using the half-space representation, one would need to find everywhere the cutting plane intersects the BZ and create new vertices. This approach can be especially prone to finite precision errors. Instead, it is simpler to let the julia Polyhedra library [4] do the hard work of reducing the BZ.

A convex hull representation simplifies access to the vertices of the polytope. A convex hull is used to represent the initial BZ because the vertices of the BZ ,**V**, are needed throughout the algorithm. Using a half-space representation simplifies the reduction of volumes. The partially reduced BZ is represented using a half-space.

After all reduction has taken place, a half-space representation of the IBZ remains. The IBZ in half-space representation is converted to a convex hull to return to the user. This is done so that the user can easily access the vertices and faces of the IBZ. The code implementing this algorithm can be found in Appendix B.

Chapter 3

Results

3.1 Validation

In order to verify that the algorithm returns the correct IBZ, two tests were created: checking the volume of the IBZ compared to the BZ and checking that the IBZ "unfolds" to the BZ. As mentioned before, the volume of the BZ is expected to be reduced by a factor of $1/N$ where N is the number of operators in the point group. This allows us to verify that

$$V_{\text{IBZ}} = \frac{V_{\text{BZ}}}{N}. \tag{3.1}$$

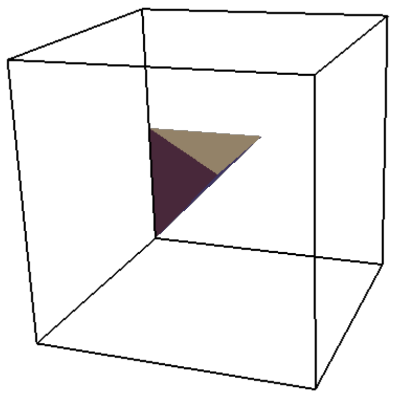
It is important to note that this check alone does not ensure that the IBZ is correct. Any polyhedron that happens to have the correct volume would pass this test. A test that takes into account the faces and vertices of the IBZ is also necessary.

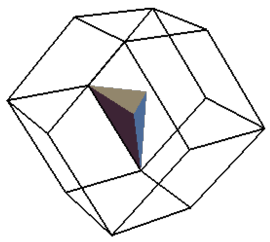
The IBZ can be checked by "unfolding" it, or by walking through the algorithm backward. To do this take each vertex of the IBZ and apply one of the symmetry operators. Now find the convex hull of the original and transformed vertices. Take the vertices of this convex hull and apply a different operator. Continue this process until all operators have been used. The final convex hull should be the BZ. Note that this test is also not enough to verify that the IBZ is correct. A "partially reduced" BZ such as the polyhedra that would be encountered in the intermediates steps of the unfolding check, would also unfold to the BZ. By checking the IBZ volume *and* unfolding the IBZ, the IBZ can correctly verified.

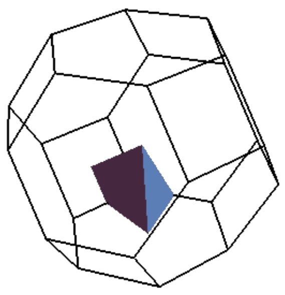
These checks were used to verify the validity of the algorithm for a large number of lattices. For each of the 14 bravais lattices, 50 random lattices were generated and the BZs and IBZs were found. All tests confirmed that the correct IBZ was found.

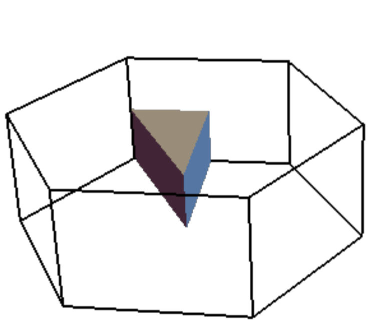
3.2 Examples

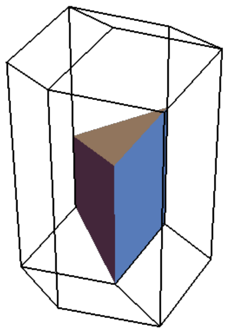
To illustrate the output of the algorithm we give an example of the BZ and IBZ for each of the 14 Bravais lattices (Tab. 3.1)

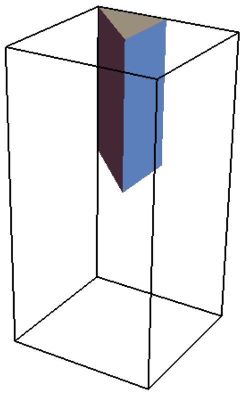

simple cubic

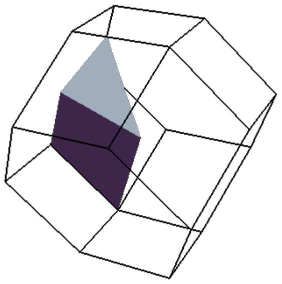

face centered cubic

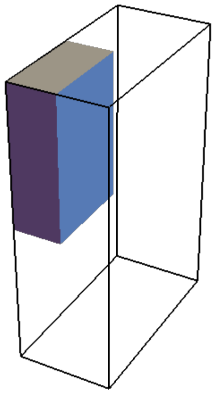

body centered cubic

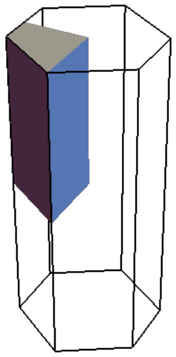

Hexagonal

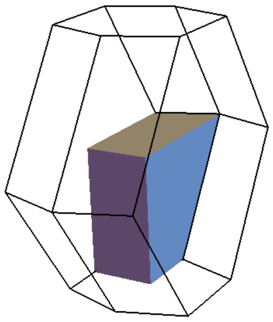

Rhombehdral

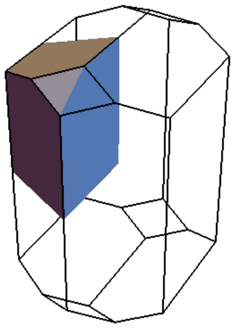

Simple Tetragonal

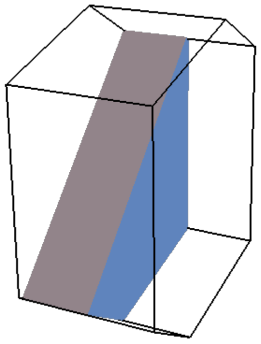

Body Centered Tetragonal


Simple Orthorhombic


Base Centered Orthorhombic


Body Centered Orthorhombic


Face Centered Orthorhombic


Simple Monoclinic

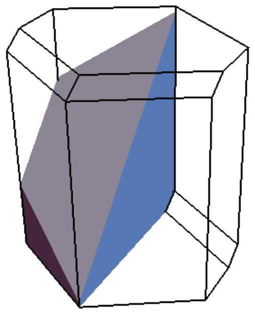
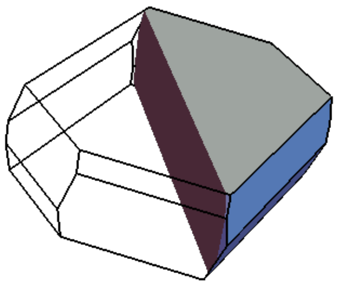
	
Base Centered Monoclinic	Triclinic

Table 3.1 The BZ (wire-frame) and IBZ (blue polyhedra) of the 14 Bravais lattices.

As discussed in Appendix A, the IBZ the algorithm returns is not completely symmetrically unique. Some faces of the IBZ can be symmetrically equivalent to other faces of the IBZ. Because the faces have no volume this does not present any problems when integrating over the volume of the IBZ. Sampling points on the faces of the IBZ could be calculated redundantly, but the number of sampling points on the faces of the IBZ will in practice be much fewer than the total number of sampling points. This redundancy would only be significant for grids with very few points. In these cases more care should be taken when evaluating points on the faces of the IBZ.

A road map of the code can be found in Appendix B. The code can be found at https://github.com/JohnEdChristensen/IBZ.

Appendix A

Proof

Proof. Let $\mathbf{P}$ be a polytope that is the volume enclosed by the initial BZ. Let $\mathbf{V}$ be the vertices of $\mathbf{P}$. Let $\mathbf{G}$ be the point group operators of the lattice. Iterative steps will be taken to to reduce $\mathbf{P}$ down to the IBZ.

The IBZ (found after k steps) must satisfy the following properties:

1. $\forall x \in \mathbf{P}_k$ and $\forall g \in \mathbf{G} - \{\mathbf{I}\}$, $gx \notin \mathbf{P}_k$.

   Each point in the IBZ is moved outside the IBZ for all operations in the point group except for the identity operator.

2. $\forall x \in \mathbf{P}_0 \, \exists g$ such that $gx \in \mathbf{P}_k$.

   For each point in the BZ there is an operator that will move that point inside the IBZ.

For the first step let

$$\mathbf{P}_0 = \mathbf{P}, \mathbf{V}_0 = \mathbf{V}, \mathbf{G}_0 = \mathbf{G}.$$

Choose $v_1 \in \mathbf{V}$ and let $\mathbf{H}_0 = \mathbf{G}_0 - \mathrm{Stab}_{\mathbf{G}_0}(\mathrm{v}_1)$, where $\mathrm{Stab}_{\mathbf{G}_0(\mathrm{v}_1)}$ are the operators $g \in \mathbf{G}_0$ where g does not does not move v_1, that is $\|v_1 - gv_1\| = 0$.

Let

$$\mathbf{P}_1 = \{x \mid \|x - v_1\| < \|x - gv_1\| \, \forall g \in \mathbf{H}_0\}.$$

$\mathbf{P}_1$ is the polytope that has been reduced by every operator $gx \in \mathbf{H}_0$. This partially fulfills condition (1) as demonstrated in the following lemma.

Lemma A.0.1. *For $\forall x \in \mathbf{F}_1, \forall g \in \mathbf{H}_0$ we have $gx \notin \mathbf{P}_1$.*

Proof.

$$\forall g \in \mathbf{H_0} \text{ and } x \in \mathbf{P}_1$$

we have

$$\begin{aligned}\|gx - v_1\| &= \|x - g^{-1}v_1\| \\ &> \|x - v_1\| \\ &= \|gx - gv_1\| \\ &\implies gx \notin \mathbf{F}_1.\end{aligned}$$

□

This lemma tells us that any operator that was used to reduce $\mathbf{P}_1$ will move any point $x \in \mathbf{P}_1$ outside $\mathbf{P}_1$.

This concludes the first iteration. A new point will now be chosen so that the operators that have not yet been used to reduce $\mathbf{P}_1$ can taken into account ($g \in \mathbf{G} - \mathbf{H}_0$).

Let $\mathbf{G}_1 = \mathrm{Stab}_{\mathbf{G}_1}(\mathrm{v}_1)$ and $\mathbf{V}_1$ be the vertices of $\mathbf{P}_1$. Choose a new vertex $v_2 \in \mathbf{V}_1 - v_1$ and let $\mathbf{H}_1 = \mathbf{G}_1 - \mathrm{Stab}_{\mathbf{G_1}}(\mathrm{v}_2)$. Let

$$\mathbf{P}_2 = \{x \mid \|x - v_2\| < \|x - gv_2\| \, \forall g \in \mathbf{H}_1\}.$$

The lemma applies again, so $\forall x \in \mathbf{P}_2$ and $\forall g \in \mathbf{H}_1$ we have $gx \notin \mathbf{P}_2$. Which means $\forall g \in \mathbf{H}_0 \cup \mathbf{H_1} = \mathbf{G}_0 - \mathrm{Stab}_{\mathbf{G}_1}(\mathrm{v}_2)$, and therefore $gx \notin \mathbf{P}_1$. This concludes the second iteration.

Continue this process finding $\mathbf{P}_k$ until the only element of $\mathrm{Stab}(v_k)$ is the identity operator. At this point, $\forall g \in \mathbf{G} - \mathbf{I}$ and $\forall x \in \mathbf{P}_k$ we have $gx \notin \mathbf{P}_k$, so $\mathbf{P}_k$ satisfies property (1).

To show $\mathbf{P}_k$ satisfies property (2) Consider the following: $\forall x \in \mathbf{P}_i$ choose $g \in \mathbf{H}_i$ such that

$$\|x - gv_i\| \leq \|x - hgv_i\| \forall h \in \mathbf{G}_i.$$

Let $\mathbf{L}$ be the set of points on a line from gv_i to x.

$$\|l - gv_i\| < \|l - hgv_i\| \quad \forall h \in \mathrm{Stab}_{\mathbf{G}_\mathrm{i}}(\mathrm{gv_i}) \text{ and } \forall \mathrm{l} \in \mathbf{L}.$$

In the limit as $l = x$

$$g^{-1} l \in \mathbf{P}_{i+1} \,\mathrm{and}\, \mathrm{g}^{-1}\mathrm{x} \in \overline{\mathbf{P}}_{\mathrm{i+1}}.$$

$$\implies \forall x \in \mathbf{P}_0 \, \exists g \text{ such that } gx \in \overline{\mathbf{P}}_k.$$

Where $\overline{\mathbf{P}}_k$ is the closure (includes the faces) of $\mathbf{P}_k$. This isn't exactly what we sought out to prove, but it is close enough. To find the truly Irreducible Brillouin zone, the faces of $\overline{\mathbf{P}}_k$ that are symmetrically equivalent to another face would need to be removed. Because faces have no volume (the set of points on a face form a set of measure 0), this has no affect when integrating over the IBZ. For our purposes $\overline{\mathbf{P}}_k$ is sufficient

$\square$

Appendix B

Code

The package is located at https://github.com/JohnEdChristensen/IBZ. Source code is located in "src/ibz.jl". Test cases can be found in "test/runtests.jl". The main algorithm is demonstrated in the code below.

```julia
@doc """
reduce_bz(lat, at, atom_pos)
Returns the reduced Brillouin zone
# Arguments
- `lat:Array{Float64,3,3}`: The lattice vectors
- `at:Array{Int64,n}`: The atom types
- `atom_pos:Array{Int64,n,3}`: position of the atoms
"""
function reduce_bz(lat, at, atom_pos)
    bz = make_bz(lat,false)
    verts =  collect(points(polyhedron(bz,CDDLib.Library())))

    obz = copy(hcat(verts...)')
    obz = chull(obz)

    symmetry = pyimport("phenum.symmetry")
    #transpose to use phenumlib
    spaceGroup = symmetry.get_spaceGroup(transpose(lat),at,atom_pos)[1]
    ops = [spaceGroup[i,:,:] for i=1:size(spaceGroup)[1]]

    #keep track of the indices of the stabilzer operators
    #(the operators that did not move a given point)
    stabilizer_index = []
    index = 1
    while size(ops)[1] != 1
        for i=1:(size(ops)[1])
            m = ops[i]
            vert = obz.points[obz.vertices[index],:]
            rotated_vert = m*vert
            # check if m is a stabilizer, if it is, mark it to be kept
            if (vert ≈ rotated_vert)
                append!(stabilizer_index,i)
            else
                cut_plane = (vert - rotated_vert)

                bz = bz ∩ HalfSpace(cut_plane,0)
            end
        end
        #if we only have stabilizers move to a new point
        if size(stabilizer_index)[1] == size(ops)[1]
            index = (index + 1)%size(obz.vertices)[1]
        else
            index = 1
        end
        #remove all but the stabilizers
        ops = [ops[i,:,:][1] for i in stabilizer_index]
        stabilizer_index= []
    end

    #convert to verts
    verts =  collect(points(polyhedron(bz,CDDLib.Library())))
    # convert to 2d array
    return chull(copy(hcat(verts...)'))


end #function
```

Bibliography

[1] R. W. Grosse-Kunstleve, B. Wong, M. Mustyakimov, and P. D. Adams, “Exact direct-space asymmetric units for the 230 crystallographic space groups,” Acta Crystallographica Section A Foundations of Crystallography **67,** 269–275 (2011).

[2] M. I. Aroyo, D. Orobengoa, G. de la Flor, E. S. Tasci, J. M. Perez-Mato, and H. Wondratschek, “Brillouin-zone database on theBilbao Crystallographic Server,” Acta Crystallographica Section A Foundations and Advances **70,** 126–137 (2014).

[3] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The Quickhull algorithm for convex hulls,” ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE **22,** 469–483 (1996).

[4] B. Legat *et al.*, “JuliaPolyhedra/Polyhedra.jl: v0.5.1,”, 2019.

Index