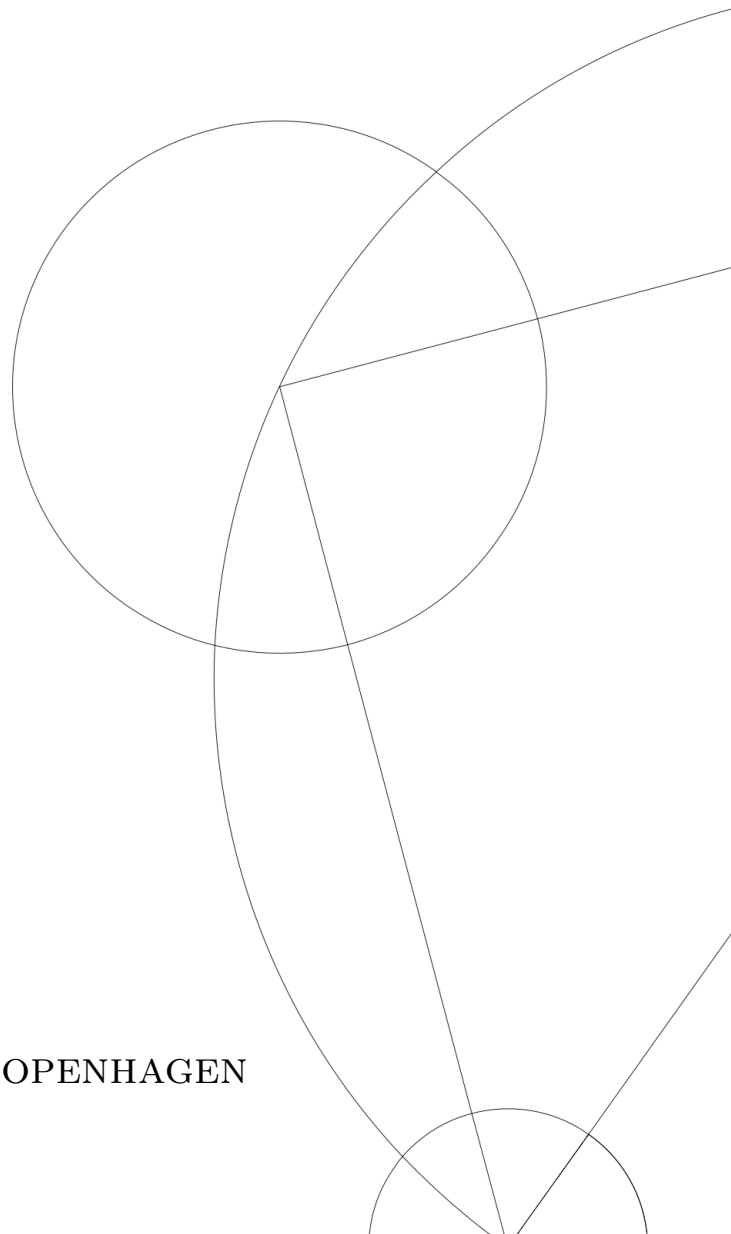
Visualisation of Concept in Condensed Matter Physics

Written by *Nikolai Plambech Nielsen*
June 10, 2018

Supervised by
Mark Spencer Rudner



University of Copenhagen

UNIVERSITY OF
COPENHAGEN

Name of Institute:	Niels Bohr Institute
Name of Department:	Center for Quantum Devices
Author(s):	Nikolai Plambech Nielsen
Email:	lpk331@alumni.ku.dk
Title and subtitle:	Visualisation of Concept in Condensed Matter Physics -
Supervisor(s):	Mark Spencer Rudner
Handed in:	2018-06-13
Defended:	2018-06-??

Name _____

Signature _____

Date _____

Abstract

In this thesis we explore key concepts in the field of condensed matter physics and translate these into visualisations using the computer programming language Python. Concepts explored include the geometry of crystal structures, families of lattice planes, neutron scattering and the band structure of two dimensional materials. 4 distinct programs are written to visualize each of these concepts, and they are bundled into one open-source software package available at https://github.com/NikolaiNielsen/Bachelor, along with documentation on its use.

Contents

1 Introduction

In this thesis we look at phenomena arising from the discrete translational symmetry of crystals. As a first example we look at the Nearly Free electron model in one dimension. This model illustrates some key concepts which will be useful later on, like reciprocal lattice vectors and how the dispersion relation change with the potential strength.

1.1 Nearly Free Electron model in one dimension

The nearly free electron model is, as the name suggests, a model in which electrons are moving in a medium with a small perturbing potential, such that they are nearly free. As such we assume that electrons are essentially free, but experience some small periodic potential that we treat as a perturbation.

In general, the Hamiltonian is

$$H = H_0 + H', \quad H_0 = \frac{p^2}{2m}, \quad H' = V(x) = V(x + na), n \in \mathbb{Z} \tag{1.1}$$

where a is the spacing between atoms. In this case we are dealing with plane waves (at least to zeroth order in perturbation theory) $|k\rangle$, with zeroth order energies $E_0(k) = \hbar^2 k^2/2m$. Now, to calculate anything in perturbation theory, we need the matrix elements of the potential $\langle k'|V|k\rangle$. These are

$$\langle k'|V|k\rangle = \int_{-\infty}^{+\infty} \mathrm{d}x e^{-i(k'-k)x} V(x) \equiv V_{k'-k}. \tag{1.2}$$

Due to the periodicity of the potential we can write this integral as a sum of integrals by setting $x = x' + na$

$$\langle k'|V|k\rangle = \sum_{n=-\infty}^{\infty} e^{-i(k'-k)na} \int_0^a \mathrm{d}x' e^{-i(k'-k)x'} V(x'). \tag{1.3}$$

This sum is infinite for a discrete set of values $k' - k$, namely

$$k' - k = \frac{2\pi j}{a} \equiv G_j, \quad j \in \mathbb{Z} \tag{1.4}$$

where G_j is called a reciprocal lattice vector (with $x_n = na$ being a direct lattice vector). If $k' - k$ is not a reciprocal lattice vector, then this sum is just 0, akin to how roots of unity sum to 0.(**REF FOR THIS?**)

Due to this, there is only a discrete set of values in k-space that contribute to the energy (and wave function) of $|k\rangle$, meaning that $|k\rangle$ can only scatter into another state if it is separated by a reciprocal lattice vector. Further interesting things happen if the energies of some other plane waves are equal to that of $|k\rangle$. Due to the parabolic dispersion of the plane waves, the first case where these conditions apply is for k around π/a and $k' = k + G_{-1} \approx -\pi/a$. Here $E_0(k) = E_0(k')$, and we are at a reciprocal lattice vector (the smallest non-trivial one, in fact). Due to this we need to treat the problem in degenerate perturbation theory.

For this case, there are 4 relevant matrix elements:

$$\langle k|H|k\rangle = E_0(k) + V_0, \quad \langle k'|H|k'\rangle = E_0(k + G_{-1}) + V_0, \quad \langle k'|H|k\rangle = \langle k|H|k'\rangle^* = V_G. \tag{1.5}$$

For simplicity we assume $V_0 = 0$, as it is just a constant. Around these values of k and k' we take the wave function as $|\psi\rangle = \alpha|k\rangle + \beta|k + G_{-1}\rangle$, which means that the energy can be found as the eigenvalue to the Hamiltonian in this "two-dimensional" subspace of these two plane waves:

$$\begin{pmatrix} E_0(k) & V_G^* \\ V_G & E_0(k + G_{-1}) \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = E \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad \Rightarrow \quad (E_0(k) - E)(E_0(k + G_{-1}) - E) - |V_G|^2 = 0. \tag{1.6}$$

In the simple case of k being exactly π/a, the energy is just $E_\pm = E_0(k) \pm |V_G|$, which means there is a band gap of $2|V_G|$ at $\pm\pi/a$. In the not-so-simple case of $k = \pi/a - \delta$, where δ is some small (positive) number, we get

$$E_0(k) = \frac{\hbar^2}{2m}\left[\left(\frac{\pi}{a}\right)^2 + \delta^2 - \frac{2\pi}{a}\right], \quad E_0(k + G_{-1}) = \frac{\hbar^2}{2m}\left[\left(\frac{\pi}{a}\right)^2 + \delta^2 + \frac{2\pi}{a}\right], \tag{1.7}$$

with an eigenenergy of

$$E_{\pm} = \frac{\hbar^2}{2m}\left[\left(\frac{\pi}{a}\right)^2 + \delta^2\right] \pm |V_G|\sqrt{1 + \left(\frac{2\pi\hbar^2\delta}{2m|V_G|}\right)^2}. \tag{1.8}$$

As we are dealing with $\delta \approx 0$, we expand the square root to second order and get

$$E_{\pm} = \frac{\hbar^2\pi^2}{2ma^2} \pm |V_G| + \frac{\hbar^2\delta^2}{2m}\left[1 \pm \frac{\hbar^2\pi^2}{ma^2|V_G|}\right] \tag{1.9}$$

where the second term in brackets is larger than unity (for small perturbations) [1], such that the dispersion is a negative parabola in δ, for E_-. This all means that for $k = \pi/a$ (or equivalently $-\pi/a$), a gap appears in the dispersion relation, and the energy in the lower branch is lowered, such that $\partial E/\partial k = 0$ (see figure 1). As it turns out this is happens for any reciprocal lattice vector, with $k = j\pi/a$ and $k' = -j\pi/a = k + G_{-j}$. These characteristics (scattering only being allowed for states separated by reciprocal lattice vectors, and bands of energy appearing at regular intervals) also appear in systems of higher dimensions, as we will see in sections 3 and 4.

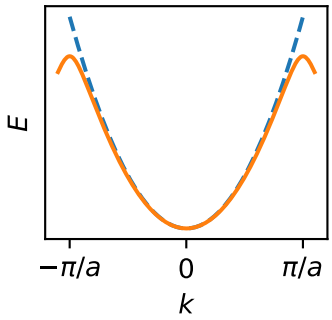


Figure 1: Dispersion relation of electrons in the (nearly) free electron model, for $-\pi/a \leq k \leq \pi/a$. The solid orange line is for a nearly free electron, whilst the dashed blue line is for a free electron.

1.2 Blochs Theorem

In the previous section we considered the case of an electron in a weak periodic potential, where the wave function is essentially a plane wave. But this is not necessarily the case for real materials. So can we be sure that we can treat the wave function as plane waves even when the potential strength is cranked up? It turns out we can, and the proof of that lies with Blochs theorem.

The assumption here is that we are working with a periodic potential, such that $V(\mathbf{r} + \mathbf{R})$ for all lattice points. As such the full Hamiltonian has a discrete translational symmetry, which means that all translation operators $\hat{T}_{\mathbf{R}}$ (that translate by any lattice vector $\mathbf{R}$) commute with the Hamiltonian: $[\hat{H}, \hat{T}_{\mathbf{R}}] = 0$. This further means that we can simultaneously diagonalize these operators and find a shared basis of eigenstates.

The eigenvalue of the translation operator must have a magnitude of unity, as it otherwise would break the translation symmetry inherent to the system. So the only possible eigenvalue is a phase shift. Further we need all translation operators to commute with each other, and $\hat{T}_{\mathbf{R}_1}\hat{T}_{\mathbf{R}_2} = \hat{T}_{\mathbf{R}_1+\mathbf{R}_2}$. With these restrictions the only possible eigenvalue is

$$\hat{T}_{\mathbf{R}}\psi(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{R}}\psi(\mathbf{r}) = \psi(\mathbf{r} + \mathbf{R}), \tag{1.10}$$

where the last equality is just the definition of the translation operator. This means we can write the wave function as a plane wave times some function that has the same periodicity as the crystal:

$$\psi(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}}u(\mathbf{r}), \tag{1.11}$$

where $u(\mathbf{r})$ carries this periodicity of the crystal. As we shall see in section 4, we can write a periodic function as a sum of plane waves over reciprocal lattice vectors. This then also applies to u, as it is periodic in the unit cell. Due to this, we can write the wave function as

$$\psi(\mathbf{r}) = \sum_{\mathbf{G}} u_{\mathbf{G},\mathbf{k}} e^{i(\mathbf{G}+\mathbf{k})\cdot\mathbf{r}}, \tag{1.12}$$

where $u_{\mathbf{G},\mathbf{k}}$ is some number. This will be important in the context of scattering and band structure as we will see in sections 3 and 4.

1.3 General implementation

All of the programs in this project will be implemented in the open source programming language Python, using the standard library, along with the NumPy and Matplotlib packages. Numpy adds the necessary tools for handling arrays and linear algebra operations, whilst Matplotlib is used for plotting the results.

One thing to note is that doing numerical calculations will lead to some numerical error. As such testing for numerical equality will often time yield the "wrong" result. Because of this, we employ the `isclose` function provided by NumPy, which takes 4 arguments: `a`, `b`, `atol` and `rtol`. `a` and `b` are the numerical values to be tested for equality, `atol` is the absolute tolerance and `rtol` is the relative tolerance. The function then tests if $|\mathtt{a}-\mathtt{b}| \leq (\mathtt{atol}+\mathtt{rtol}\cdot|\mathtt{b}|)$, with default values for `atol` and `rtol` being $1\cdot 10^{-8}$ and $1\cdot 10^{-5}$ respectively.

2 Lattices and crystal structure

2.1 Theory

There are three parts to the description of a crystal structure. First is the lattice. This is the mathematical "framework" upon which the physical part of the crystal lies. It can be defined in a number of ways, however the one we will use here is the following:

A lattice is defined as the infinite set of points produces by a linear combination of independent *primitive lattice vectors*, with integer coefficients.

Usually, the primitive lattice vectors are labelled $\mathbf{a}_i$, and the coefficients n_i, so for a d-dimensional lattice, the lattice points $\mathbf{R}$ are given by

$$R = \sum_{i=1}^{d} n_i \mathbf{a}_i \tag{2.1}$$

For this thesis we will mainly focus on the cases of $d = 3$ (visualization of lattices, families of lattice planes and scattering) and $d = 2$ (visualization of the Fermi surface).

A thing to note, is that the choice of primitive lattice vectors is not unique. A new set of primitive lattice vectors can be created by taking a linear combination of the original primitive lattice vectors, with integer coefficients. If the original set is ordered in a matrix $A = \begin{pmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n\end{pmatrix}$, and the new set in a matrix $B = \begin{pmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n\end{pmatrix}$, then $B = MA$, where M is the matrix containing the coefficients. This matrix must have integer entries, and its inverse likewise.

The integer entries of the direct matrix makes sure that any lattice point generated with the new set of primitive lattice vectors will also have integer coefficients when expressed in the old set of primitive lattice vectors. The integer entries of the inverse matrix then makes sure that this process will also happen in reverse. This characteristic will come into play when trying to detect the lattice based on the users input of primitive lattice vectors.

The second part is the unit cell. This is the building block of the lattice. It is a region of space which, when stacked will completely tile the space. Like with the choice of primitive lattice vectors, the choice of unit cell is not unique. In particular we distinguish between two type of unit cells: the smallest possible unit cell and everything else. The smallest possible unit cell is called a *primitive* unit cell, and must contain only one lattice point (it cannot contain no lattice points, as then it would not recreate the lattice when tiled). Any unit cell containing more than one lattice point is called a *conventional unit cell*. Usually a conventional unit cell is chosen for ease of calculation (as we will see in the scattering simulation), where the primitive lattice vectors constitute an orthogonal set.

The third part of the crystal structure is the basis. This is a description of the physical objects that make up the structure, and their position in relation to the lattice. In our case the objects are of course atoms.

The basis is specified as a list of vectors that are to be added to the lattice points, specifying the position of the atoms in the crystal.

The user can create any type of crystal they want by specifying any set of primitive lattice vector and supplying any desired basis. However a small selection of crystals will be available as presets.

These include the 14 Bravais lattices with a 1 atom basis (at $(0,0,0)$), named in table 1. Each of these will specify the primitive lattice vectors for a corresponding primitive unit cell. Furthermore five other crystal presets will be available, named in table 2. Specifications of all of these presets are available in the appendix A.

Simple cubic	base centred cubic (bcc)
face centred cubic (fcc)	tetragonal
body centred tetragonal	orthorhombic
body centred orthorhombic	face centred orthorhombic
base centred orthorhombic	simple monoclinic
base centred monoclinic	hexaonal
triclinic	rhombohedral

Table 1: The 14 Bravais lattices

fcc, conventional	bcc, conventional
zincblende	wurtzite
diamond (zincblende with 1 atom)	

Table 2: Other available crystal presets

2.2 Implementation

Mathematically speaking, a lattice is infinite. A physical crystal, of course, is not. However, even though a real crystal is finite, plotting all of the atoms would be infeasible, both due to the number of atoms, each atom would be way too small (if it was feasible, what would be the point of this project then?). So for the purposes of this project, only a couple of unit cells will be plotted. A good amount seems to be 8 unit cells. 2 in each of the directions specified by the primitive lattice vectors. This keeps the size of the plot relatively small, whilst still showing the important parts of the crystal structure. As such, in the following we assume that $n_i \in \{0,1,2\}$.

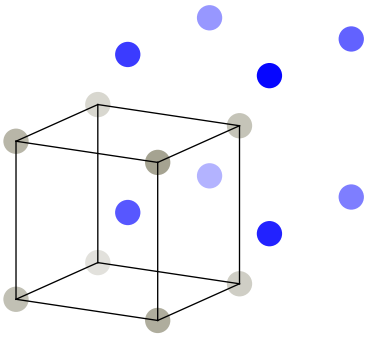


Figure 2: One conventional unit cell for a bcc lattice. If we naively plot all atoms associated with a lattice point, we will end up with unfinished unit cells.

In creating a program that plots these structures, the thought should always be on how the end product looks. Mainly we want the plot to be as clear and instructive as possible. This constitutes plotting only full unit cells: no unit cell can have missing atoms. However, the screen is still just a two dimensional projection of the actual three dimensional phenomena and without some form of depth perception, the crystal will just look like a weird two dimensional pattern. This is what grid lines fix. They give the required depth perception to allow the user to comprehend the crystal as a three dimensional structure, and not as a two dimensional sheet of dots.

For the crystals that can be expressed as a lattice with orthogonal primitive lattice vectors and a basis (cubic, tetragonal and orthorhombic), we usually want to plot orthogonal grid lines, whilst for other crystals plotting grid lines along the lattice vectors will be more useful.

In the case of plotting primitive unit cells, the ones furthest away from the origin (say with $n_1 = n_2 = n_3 = 2$) may not be filled. Say the basis consists of only one atom. Then this atom will just be placed on the lattice points, and there is no problem, as this is the edge of the unit cell. However, if the basis consists of two atoms (or more), where one is placed at $(0,0,0)$ and the other has some positive coordinates, then this second atom will be in a unit cell which is not supposed to be plotted (it would necessitate plotting the lattice points corresponding to $n_i = 3$). An example of this is figure 2, which shows one unit cell of a simple cubic lattice, with a two atom basis (corresponding to a conventional

bcc unit cell). A way to correct for this is to write each plotted atom as a linear combination of the primitive lattice vectors with coefficients n_i' (where the coefficients here are real numbers, as they do not necessarily align with the lattice points). If the coefficients are $0 \leq n_i' \leq 2$ for all n_i', then we plot the point.

For plotting conventional unit cells when the user inputs primitive lattice vectors (for cubic, tetragonal and orthorhombic lattices), we need something similar, otherwise a situation like in figure 3 may occur. To fix this we calculate the side lengths of the cuboid plot box that just contains the parallelepiped arising from plotting the lattice points with the specified coefficients ($n_i \in \{0, 1, 2\}$), and fill this plot box completely with atoms.

To calculate this, the program creates the 8 possible vectors arising from linear combinations of the primitive lattice vectors, with coefficients from the minimum and maximum coefficients and taking the limits of the plot box as the minimum and maximum values for these 8 vectors. For the specified lattice points, these 8 vectors are:

$$\begin{array}{llll} \mathbf{v}_1 = \mathbf{0}, & \mathbf{v}_2 = 2\mathbf{a}_1, & \mathbf{v}_3 = 2\mathbf{a}_2, & \mathbf{v}_4 = 2\mathbf{a}_3, \\ \mathbf{v}_5 = 2(\mathbf{a}_1 + \mathbf{a}_2), & \mathbf{v}_6 = 2(\mathbf{a}_1 + \mathbf{a}_3), & \mathbf{v}_7 = 2(\mathbf{a}_2 + \mathbf{a}_3), & \mathbf{v}_8 = 2(\mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3). \end{array} \tag{2.2}$$

These are the vertices of the aforementioned parallelepiped. The side lengths of the plot box are then the maximum values of the coordinates minus the minimum values. Then we plot atoms for a larger range of coefficients to completely fill out this plot box. If some atoms fall outside of the box we hide them.

2.3 Step-by-step

Initially the program will either load the chosen crystal preset in such a way as to make the resulting plot as informative as possible (eg. place lattice vectors along cardinal axes for an orthogonal lattice, to make plotting grid lines easier). If the user manually specifies the lattice and basis, the program will try to classify the lattice according to the specifications in the appendix A. Next the program checks whether or not the crystal should be rotated to make plotting prettier.

In general the rotation algorithm tries to align one lattice vector with the x-axis. $\mathbf{a}_1$ is preferred, but is only chosen to lie along the x-axis if it forms an orthogonal pair with at least one other primitive lattice vector. The second primitive lattice vector of the pair ($\mathbf{a}_2$ being preferred) is then aligned along the y-axis. If the three primitive lattice vectors form an orthogonal set, then the last vector of the set will now be aligned along the z-axis.

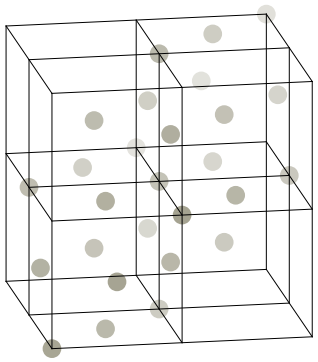


Figure 3: An attempt to plot eight conventional unit cells for an FCC lattice, by using primitive lattice vectors with coefficients in the set $\{0, 1, 2\}$. This does not fill out all eight unit cells, so we need to plot more lattice points than this. In practise we need $\{-2, -1, 0, 1, 2, 3, 4\}$

The actual rotation is done by rotating the whole crystal (each primitive lattice vector and all vectors in the basis) along the cross product between the initial vector and the destination vector. Rotating the crystal such that $\mathbf{a}_1$ lies along the x-axis is done by rotating along $\mathbf{a}_1 \times \hat{\mathbf{x}}$, with an angle of $\sin\theta = |\mathbf{a}_1 \times \hat{\mathbf{x}}|/|\mathbf{a}_1|$.

However, this might rotate the crystal the wrong way, depending on the orientation between the two vectors. Because of this the program checks whether or not the rotated initial vector and the destination vector is parallel (that is, if the rotated $\mathbf{a}_1$ is parallel to $\hat{\mathbf{x}}$). If this is not the case, the whole crystal is rotated about the same vector, with an angle -2θ.

Five of the Bravais lattices have specialised rotation functions: hexagonal, base centred monoclinic and the three face centred lattices.

For the hexagonal, the program detects which primitive lattice vectors constitute the triangular lattice and orients them such that one is along the x-axis and the other is in the xy-plane (easily done by rotating the third primitive lattice vector such that it is parallel with the z-axis). The same approach is used for the base centred monoclinic, but here the program always aligns $\mathbf{a}_1$ along the x-axis, and

$\mathbf{a}_2$ in the xy-plane. Here however, the easy option of rotating $\mathbf{a}_3$ is not available. Instead the program uses the fact that the vector rejection of $\mathbf{a}_2$ with $\mathbf{a}_1$ is orthogonal to $\mathbf{a}_1$ (the vector rejection of $\mathbf{a}_2$ with $\mathbf{a}_1$ being $\mathbf{a}_2$ minus the projection of $\mathbf{a}_2$ along $\mathbf{a}_1$). This vector rejection is then in the yz-plane, and the crystal can be rotated along $\mathbf{a}_1$ with the angle the rejection makes with $\hat{\mathbf{y}}$: $\cos\theta = \mathbf{a}_{2,rej} \cdot \hat{\mathbf{y}}/|\mathbf{a}_{2,rej}|$.

For the face centred lattices, the ideal, rotated lattice is created from the magnitude of the primitive lattice vectors: if $|\mathbf{a}_1| = a, |\mathbf{a}_2| = b, |\mathbf{a}_3| = c$, then $\mathbf{a}_1' = (a/2, b/2, 0), \mathbf{a}_2' = (a/2, 0, c/2), \mathbf{a}_3' = (0, b/2, c/2)$. First the crystal is rotated such that $\mathbf{a}_1$ aligns with $\mathbf{a}_1'$, by using their cross product. Then the crystal is rotated such that the now rotated $\mathbf{a}_2$ aligns with $\mathbf{a}_2'$, via the vector rejection of $\mathbf{a}_2$ with $\mathbf{a}_2'$. (**THIS METHOD AND THE ONE FOR THE HEXAGONAL LATTICE ASSUMES THAT THE PRIMITIVE LATTICE VECTORS ARE SPECIFIED AS IN THE APPENDIX**).

With the lattice and basis rotated, the program finds the limits of the plot box as written above, after which the crystal is generated. This is done by looping over the three ranges specified by the minimum and maximum coefficients, creating each lattice point $\mathbf{R}$ by Eq. (2.1). For each lattice point n atomic positions are created by adding one of the n vectors in the basis to the lattice point. Furthermore lists of the colours and sizes associated with each atom are also created at this point. After creating all the atoms, the program deletes any that may lie outside the limits of the plot box.

The only thing missing now is to create the grid lines and plot everything. The program has two ways of creating grid lines: along the primitive lattice vectors and along the cardinal axes.

Creating grid lines along the primitive lattice vectors works by taking each lattice point $\mathbf{R}_n$, and finding the lattice point $\mathbf{R}_m$ the furthest away from it, in the (postive) direction of these lattice vectors, such that $\mathbf{R}_m = R_n + \alpha\mathbf{a}_i$, where $\alpha > 0$, for all $i \in [1, 2, 3]$, and creating lines between $\mathbf{R}_n$ and $\mathbf{R}_m$. This does create duplicate grid lines, but these do not show on the final plot, since they are all plotted with the same width and colour.

Creating grid lines along the cardinal axes works by finding the minimum spacing between lattice points on these axes (called a_x, a_y and a_z), and using these as the spacing between grid lines. The program then finds the maximum and minimum coordinates of lattice points along the cardinal axes (called x_{min}, x_{max}, etc.). This is used to create ranges corresponding to each lattice points on the cardinal axes: The range for the x-axis starting at x_{min} and ending at x_{max} with steps of a_x.

These ranges then specify a grid of lattice points on the xy, xz and yz-planes. The program then creates lines orthogonal to these planes, stretching from z_{min} to z_{max} for the points in the xy-plane, and similarly for the other two planes.

Next a blank figure is created with an orthogonal projection and limits as calculated above. The atoms are plotted with colours and sizes specified by the user. The grid lines are plotted with a uniform size and colour and lastly the primitive lattice vectors are plotted with corresponding labels.

2.4 Examples

As an example, we plot a simple cubic lattice with a two atom basis (corresponding to the basis of a bcc lattice with conventional unit cells), where the atoms on the lattice points are grey and the body centred atoms are blue (figure 4), we write the following:

```
1 Lattice(lattice_name="conventional bcc",
2         colors=["xkcd:cement", "b"])
```

Or we could plot a hexagonal lattice with a one atom basis (figure 5):

```
1 Lattice(lattice_name="hexagonal")
```

3 Reciprocal lattice and scattering

3.1 Theory

The reciprocal lattice is an incredibly useful construct, as it allows for an easy description of wave phenomena, where the main variable is the wave vector $\mathbf{k}$. The reciprocal lattice is a lattice (as defined in the previous section), but in reciprocal space. This again means that the free variables is not position, but the wave vector.

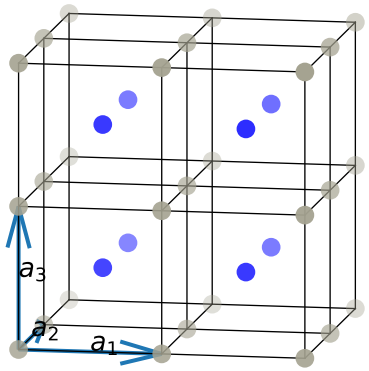


Figure 4: A simple cubic lattice with a two atom basis: One atom on the lattice points and another in the middle of each unit cell.

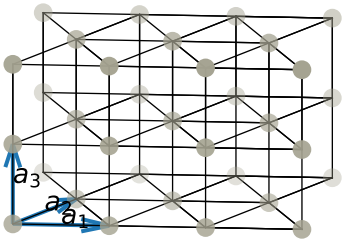


Figure 5: A hexagonal lattice, which consists of a series of triangular lattices stacked on top of each other.

The definition of a reciprocal lattice is all of the points $\mathbf{G}$, that satisfy

$$e^{i\mathbf{G}\cdot\mathbf{R}} = 1, \tag{3.1}$$

for any lattice point in real space $\mathbf{R}$. Further, the primitive lattice vectors of the reciprocal lattice all satisfy the property

$$\mathbf{a}_i \cdot \mathbf{b}_j = 2\pi\delta_{ij}, \tag{3.2}$$

where $delta_{ij}$ is the familiar Kronecker delta. This property of the reciprocal lattice vectors is what makes $\mathbf{G}$ a lattice in reciprocal space. The real space lattice points have the coordinates $\mathbf{R} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3$, and say an arbitrary point in the reciprocal space is constructed as

$$\mathbf{G} = m_1\mathbf{b}_1 + m_2\mathbf{b}_2 + m_3\mathbf{b}_3, \tag{3.3}$$

then plugging these into Eq. (3.1) we get

$$e^{i(n_1\mathbf{a}_1+n_2\mathbf{a}_2+n_3\mathbf{a}_3)\cdot(m_1\mathbf{b}_1+m_2\mathbf{b}_2+m_3\mathbf{b}_3)} = e^{2\pi i(n_1m_1+n_2m_2+n_3m_3)}, \tag{3.4}$$

where the second equality follows from the defining property of the primitive lattice vectors in reciprocal space. For $\mathbf{G}$ to be part of the reciprocal lattice, for any choice of integer n_i, then m_i need also be integers. As such the form of the reciprocal lattice matches that of the real space lattice. Usually (in the case of lattice planes and scattering) we express the 3 coefficients for the reciprocal lattice vectors using Miller indices (hkl), where $m_1 = h, m_2 = k, m_3 = l$.

Now, to actually construct the reciprocal primitive lattice vectors, the following formulas are used:

$$\mathbf{b}_1 = \frac{2\pi\,\mathbf{a}_2 \times \mathbf{a}_3}{\mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)}, \quad \mathbf{b}_2 = \frac{2\pi\,\mathbf{a}_3 \times \mathbf{a}_1}{\mathbf{a}_2 \cdot (\mathbf{a}_3 \times \mathbf{a}_1)}, \quad \mathbf{b}_3 = \frac{2\pi\,\mathbf{a}_1 \times \mathbf{a}_2}{\mathbf{a}_3 \cdot (\mathbf{a}_1 \times \mathbf{a}_2)}, \tag{3.5}$$

where the cross product in the numerator ensures the property of the Kronecker delta, and the denominator serves as a sort of normalization, such that the dot product equals 2π. Further, by dimensional analysis, $\mathbf{b}_i$ has dimensions m^{-1}, as expected for wave vectors.

The reciprocal lattice can also be understood in terms of families of lattice planes. First, a lattice plane is defined as any plane that contains at least 3 lattice points (this also guarantees that an infinite set of lattice points is contained by the plane, by virtue of the non-uniqueness of the primitive lattice vectors).

A family of lattice planes is an infinite series of parallel planes, equally spaced, such that all points of the lattice is contained in the planes, and that all planes contain lattice points.

First consider the series of planes containing the points $\mathbf{r}_m$, defined by

$$\mathbf{G} \cdot \mathbf{r}_m = 2\pi m, \tag{3.6}$$

for some integer m. This form ensures that all lattice points are contained in the planes. The minimum distance between planes can then be calculated from

$$\mathbf{G} \cdot (\mathbf{r}_2 - \mathbf{r}_1) = 2\pi, \tag{3.7}$$

as the minimum distance will be when $(\mathbf{r}_2 - \mathbf{r}_1)$ and $\mathbf{G}$ are parallel, yielding

$$d = \frac{2\pi}{|\mathbf{G}|}. \tag{3.8}$$

Now any reciprocal lattice vector will give yield a set of equidistant, parallel planes, but not all of these sets will be families of lattice planes. The family of lattice planes in the direction $\hat{\mathbf{G}}$ will have have some distance $d = 2\pi/|\mathbf{G}|$, where $\mathbf{G}$ is a reciprocal lattice vector. An infinite set of reciprocal lattice vectors share this direction, but increasing the magnitude of the reciprocal lattice vector $\mathbf{G}$ will decrease the distance between the set of planes, and at some point there will be planes that do not contain any lattice points.

As such, there is some minimum magnitude of the reciprocal lattice vector that defines a given family of lattice planes. This means that for a family of lattice planes, the distance is $d = 2\pi/|\mathbf{G}_{min}|$.

3.1.1 The Brillouin zone

In discussing Bloch's theorem we see that any wave function in a periodic medium can be expressed as a sum of plane waves, with wave vectors differing by a reciprocal lattice vector. In one dimension we see that changing $k \to k + 2\pi/a$ yields the same wave function and the states are therefore physically equivalent. This means that the wave vector is only defined up to a factor of $2\pi/a$, and can be chosen in the interval $-\pi/a \leq k \leq \pi/a$. We call wave vectors in this range *crystal momentum*, and the range itself the *first Brillouin zone*.

The same principle applies to higher dimensions. The first Brillouin zone is defined as any point $\mathbf{k}$ in reciprocal space, that is closer to $\mathbf{0}$ than any other point on the reciprocal lattice. The n'th Brillouin zone is then defined as any point $\mathbf{k}$, where $\mathbf{0}$ is the n'th nearest point on the reciprocal lattice.

For a 2 dimensional, square lattice the first Brillouin zone corresponds to a square with sides $2\pi/a$, centred on the origin.

3.1.2 Scattering

In a scattering experiment, an incoming collection of waves (be it an electrons, neutrons, x-ray photons or something completely different) with wave vector $\mathbf{k}$ is incident upon a crystal. Some of these will be scattered into states with wave vector $\mathbf{k}'$ whilst others will pass on through.

To find the general conditions for scattering, we start with Fermi's Golden Rule [1], which is a measure of the transition rate between states. It is given as

$$\Gamma(\mathbf{k}', \mathbf{k}) = \frac{2\pi}{\hbar} |\langle \mathbf{k}'|V|\mathbf{k}\rangle|^2 \delta(E_{\mathbf{k}'} - E_{\mathbf{k}}), \tag{3.9}$$

where the matrix element is

$$\langle \mathbf{k}'|V|\mathbf{k}\rangle = \int_{-\infty}^{+\infty} \frac{e^{-i\mathbf{k}'\cdot\mathbf{r}}}{\sqrt{L^3}} V(\mathbf{r}) \frac{e^{i\mathbf{k}\cdot\mathbf{r}}}{\sqrt{L^3}} \, \mathrm{d}\mathbf{r} = \frac{1}{L^3} \int_{-\infty}^{+\infty} e^{-i(\mathbf{k}'-\mathbf{k})\cdot\mathbf{r}} \, V(\mathbf{r}) \, \mathrm{d}\mathbf{r}. \tag{3.10}$$

This is just the Fourier transform of the potential! Further, we assume the potential is periodic in the unit cell, allowing us to use the same trick as with the Nearly Free Electron Model: Since $V(\mathbf{r} + \mathbf{R}) = V(\mathbf{r})$ for any lattice point $\mathbf{R}$, we can define $\mathbf{r} = \mathbf{R} + \mathbf{x}$ where $\mathbf{x}$ is a position within the unit cell, and then split up the integral into an infinite sum over lattice points:

$$\langle \mathbf{k}'|V|\mathbf{k}\rangle = \frac{1}{L^3} \int_{-\infty}^{+\infty} e^{-i(\mathbf{k}'-\mathbf{k})\cdot(\mathbf{R}+\mathbf{x})} V(\mathbf{R}+\mathbf{x}) \, \mathrm{d}\mathbf{x} = \frac{1}{L^3} \sum_{\mathbf{R}} e^{-i(\mathbf{k}'-\mathbf{k})\cdot\mathbf{R}} \int_{\substack{\text{unit-}\\ \text{cell}}} e^{-i(\mathbf{k}'-\mathbf{k})\cdot\mathbf{x}} V(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{3.11}$$

Now, as in the one dimensional case, this sum of complex exponentials has two possible outcomes. If $\mathbf{k}' - \mathbf{k}$ is a reciprocal lattice vector, all the terms are unity and the sum adds up to the total number

of unit cells in the crystal. If $\mathbf{k}' - \mathbf{k}$ is not a reciprocal lattice vector, then the terms will just oscillate, like roots of unity, summing to 0. This condition is called the Laue condition:

$$\mathbf{k}' - \mathbf{k} = \mathbf{G}. \tag{3.12}$$

Furthermore, when the scattered wave leaves the crystal, it has to have the same magnitude of the wave vector as the incoming wave, as required from the delta function in Fermi's Golden Rule:

$$|\mathbf{k}'| = |\mathbf{k}|. \tag{3.13}$$

Together these conditions are statements of conservation of crystal momentum and energy respectively. With the conditions met we still need to find the actual scattering amplitudes. This is where the integral in Eq. (3.11) comes in. It turns out that the intensity of the scattered wave vector is proportional to the absolute square of this integral, called the *structure factor*: [1]

$$I \propto |S(\mathbf{G})|^2, \quad S(\mathbf{G}) = \int_{\substack{\text{unit-}\\ \text{cell}}} e^{-i\mathbf{G}\cdot\mathbf{x}} V(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{3.14}$$

This is as far as is workable without specifying the form of the potential. For now we will assume we are working with neutron scattering. Since neutrons are not charged, they only scatter from the atomic cores by the nuclear force, and not from electrons. For this reason we model the potential of each atom in the unit cell as a delta function with some appropriate potential strength associated:

$$V(\mathbf{x}) \sum_{\text{atoms } j} f_j \delta(\mathbf{x} - \mathbf{x}_j), \tag{3.15}$$

where the potential strength is called the *form factor*. With this potential, the structure factor becomes

$$S(\mathbf{G}) = \sum_{\text{atoms } j} f_j e^{i\mathbf{G}\cdot\mathbf{x}_j}. \tag{3.16}$$

For the purposes of the program, we will further restrict the available lattice to a simple cubic with a basis. This will allow us to illustrate the core ideas of scattering whilst keeping any clutter to a minimum.

One thing this allows us to show is systemic absences. For a cubic lattice with a basis, the structure factor becomes

$$S(\mathbf{G}) = S_{hkl} = \sum_{\text{atoms } j} f_j e^{i(h\mathbf{b}_1 + k\mathbf{b}_2 + l\mathbf{b}_3)\cdot\mathbf{x}_j} = \sum_{\text{atoms } j} f_j e^{2\pi i(hx_j + ky_j + lz_j)} \tag{3.17}$$

where the coordinates of $\mathbf{x}_j$ are in units of the lattice constant a. For a bcc lattice, which corresponds to a simple cubic lattice, with two identical atoms at $(0,0,0)$ and $(1/2,1/2,1/2)$ (in units of the lattice constant), the structure factor becomes

$$S_{hkl} = f\ (1 + e^{2\pi i(h/2 + k/2 + l/2)}) = f\ (1 + e^{\pi i(h+k+l)}) = f\ (1 + (-1)^{h+k+l}), \tag{3.18}$$

meaning that for there to be any scattering for a bcc lattice, $h + k + l$ must be even. This is what is known as a systemic absence. There is also a systemic absence for fcc lattices, which can be thought of as a simple cubic lattice, with identical atoms at $(0,0,0), (1/2,1/2,0), (1/2,0,1/2)$ and $(0,1/2,1/2)$:

$$S_{hkl} = f\ (1 + e^{\pi i(h+k)} + e^{\pi i(k+l)} + e^{\pi i(h+l)}). \tag{3.19}$$

Here all of $h + k, k + l$ and $h + l$ must be even, corresponding to h, k, l all being either even or odd. As such there are systemic absences in both bcc and fcc lattices, but not simple cubic lattices, where all combinations of h, k and l can lead to scattering.

3.2 Implementation

The scattering program creates two figures. One interactive figure with the physical setup, including the desired crystal structure, incoming probe beam, detector screen and detected scattering events. The second shows only a top down view of the detector screen with the detected scattering events.

For the first figure, the scattering program builds upon the lattice plotting program. It plots the desired lattice (simple cubic with a basis), calculates scattering for this crystal given a list of form factors and an incoming wave, and displays the results on a simulated detection screen.

Calculating the actual scattering is done by first creating the reciprocal lattice with Eq. (3.5), and next creating an array of reciprocal lattice vectors with indices $h, k, l \in [-5, -4, \ldots, 5]$ (excluding $h = k = l = 0$ as this just results in a "scattered" wave vector equal to the incident wave vector). This of course does not constitute the whole possible range reciprocal lattice vectors, but a line has to be drawn somewhere. This interval includes $11^3 - 1 = 1330$ different reciprocal lattice vectors, which should be plenty to get an understanding of scattering.

This array of reciprocal lattice vectors is then used to calculate the "scattered" wave vectors by Eq. (3.12). These do not necessarily meet the other criteria of energy conservation, though, and there are further criteria to consider. First is of course any systemic absences. These wave vectors do meet the criteria of both conservation of crystal momentum and energy, but they still do not show up on the detection screen due to the systemic absence.

To calculate the systemic absence of a scattered wave vector, the structure factor for a given reciprocal lattice vector has to be calculated. This is done with Eq. (3.16). Next the program calculates the intensity $I_{hkl} = |S_{hkl}|^2$ and checks if it is equal to 0. If so, then the scattered wave vector is subject to a systemic absence.

The second additional criteria is the direction of the scattered wave vector. The scattered wave vector has to point in the direction of the detection screen, otherwise it will not physically "hit" the screen. In the program the detection screen is placed parallel to the xy-plane, with $z = 5$. As such any scattered wave vector will need to have $k'_z > 0$ to be detected.

These three criteria (conservation of energy, lack of systemic absence, and proper direction) are all calculated by the program, and if a scattered wave vector does not fulfil all of them, it is discarded.

Left are only a handful, if any, of scattered wave vectors. For each of these, the impact point of the scattered wave vector on the detection plane has to be calculated. This is done by calculating the intersection between a line and a plane. The line is defined by the point of impact $\mathbf{p}_0$ of the incident wave vector along with the scattered wave vector $\mathbf{k}'$. The plane is defined as mentioned above, with $z = 3$. In this case the intersection happens when

$$p_{0,z} + t \cdot k'_z = 3, \tag{3.20}$$

for some value of t. This value of t can then be used to calculate the point of intersection as

$$\mathbf{p} = \mathbf{p}_0 + t\mathbf{k}'. \tag{3.21}$$

These points are then plotted in the first figure along with the detection plane and the incoming wave vector (scaled so its length is equal to the wavelength).

The points are also shown on the second figure, showing the top down view of the detection plane, along with the Miller indices of the reciprocal lattice vector giving rise to the corresponding scattering events.

To make the program more user friendly, we (by default) define the incoming wave vector in units of $2\pi/a$, such that the user does not need to always include these factors.

Two additional features for the scattering program are available. The first is the "show all" feature. This plots all outgoing wave-vectors and their associated lines (starting at the impact point for the incident beam, and ending at the intersection between the outgoing wave vector and the detection plane).

The second is a "highlighting" feature. This feature works by taking a set of Miller indices and highlighting the scattering associated with said set (if scattering occurs for this reciprocal lattice vector). It plots the relevant scattering event in a different colour, and plots the associated lattice planes.

These planes are created in one of two ways. If the plane is *not* perpendicular to the xy-plane (corresponding to a normal vector $\mathbf{n}$ with a z-component different from 0) the equation of the plane is used with the origin as the starting point $\mathbf{r}_0$. For the normal vector, the program uses the displacement vector $\mathbf{d} = 2\pi\hat{\mathbf{G}}/|\mathbf{G}|$:

$$0 = \mathbf{d} \cdot (\mathbf{r} - \mathbf{r}_0) = \mathbf{n} \cdot \mathbf{r}, \quad \Leftrightarrow \quad z = -\frac{d_x x + d_y y}{d_z}. \tag{3.22}$$

The program then calculates the z-component of the plane from a given array of x and y-values. This, of course, only creates one plane. Each subsequent plane is created by displacing the original plane by an amount $d/\cos\theta = d^2/d_z$. This process is repeated in the positive direction until the smallest z-value of the uppermost plane is outside of the plot box, and likewise in the negative direction, yielding a full set of parallel planes, spaced by $\mathbf{d}$ (at least within the plot box).

However, if the plane *is* perpendicular to the xy-plane, this method does not work. Here the program creates a plane from the span of two vectors perpendicular to the displacement vector. Since the plane is perpendicular to the xy-plane, one such vector is $\hat{\mathbf{z}}$, the unit vector in the z-direction. Then the second vector can just be taken as the cross product between $\mathbf{d}$ and $\hat{\mathbf{z}}$. Again the origin is taken as the starting point:

$$\mathbf{r} = \mathbf{r}_0 + s\hat{\mathbf{z}} + t(\hat{\mathbf{z}} \times \mathbf{d}). \tag{3.23}$$

To get any subsequent planes, an integer multiple of the displacement vector $\mathbf{d}$ is added to the starting plane.

For both of these methods the planes need to be limited so they are only plotted within the plot box. This is done by replacing the z-component of any point outside the plot box with `NaN`, which will cause Matplotlib to not plot the associated point. This is especially pertinent in the second case, where not only the z-component can be outside the plot box, but x and y-components can be. Because of this $\hat{\mathbf{z}}/4$ and $(\hat{\mathbf{z}} \times \hat{\mathbf{G}})/4$ are used to increase the resolution of each plane, along with a large range of values for s and t (in this case taken to be integers, though the same effect could be achieved by using a more densely packed interval for s and t and keeping the original vectors).

Further, a second program is added alongside the scattering program. This second program just allows the user to plot any crystal, along with any set of lattice planes, defined by a set of Miller indices supplied by the user.

3.3 Examples

Say we want to simulate neutron scattering on a cubic lattice with a four atom basis (one at lattice points, the others at the centre of the faces of the unit cell). We set the relative scattering lengths as 1 for the atoms on the lattice points and 0.5 for the others. We set $\mathbf{k}_{in} = (0, 0, -1.5)$ in units of $2\pi/a$. Lastly we highlight the scattering corresponding to the Miller indices (112). This is all done by the following line, and produces figure 7.

```
Scattering(k_in=np.array([0, 0, -1.5])
           basis=np.array([[0, 0, 0],
                           [0.5, 0.5, 0],
                           [0.5, 0, 0.5],
                           [0, 0.5, 0.5]]),
           highlight=(1,1,2)
           scattering_length=np.array([1, 0.5, 0.5, 0.5]))
```

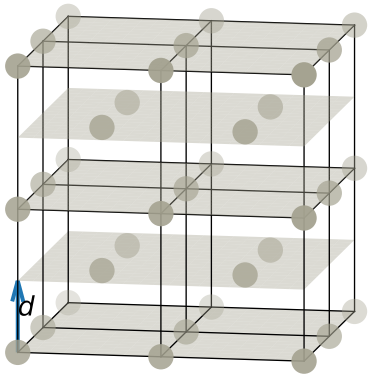


Figure 6: A bcc lattice showing the (001) family of lattice planes.

If instead we want to plot scattering on regular fcc lattice, with the same wave vector, we just set the scattering length for all atoms to 1. This gives figure 8.

```
Scattering(k_in=np.array([0, 0, -1.5])
           basis=np.array([[0, 0, 0],
                           [0.5, 0.5, 0],
                           [0.5, 0, 0.5],
                           [0, 0.5, 0.5]]),
           scattering_length=np.array([1, 0.5, 0.5, 0.5]))
```

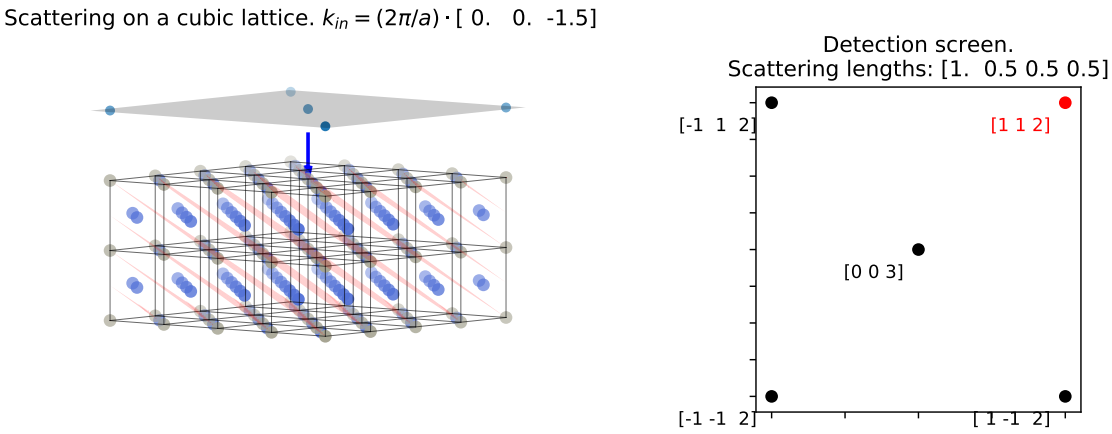


Figure 7: Scattering on a cubic lattice with a four atom basis (one at lattice points and the others on the faces of the unit cell), with form factors of 1 for the lattice point atoms, and 0.5 for the others. The choice of incoming wave vector gives rise to 5 scattering events, corresponding to 5 different sets of miller indices. Note the highlighted point and associated family of planes. They are not lattice planes, which means that for a proper fcc lattice, this set of Miller indices will be subject to systemic absence.

As a last example, say we just want to plot the (001) family of lattice planes for a bcc lattice, then we write the following and get figure 6.

```
1 Reciprocal(lattice_name="bcc",
2              indices=(0,0,1))
```

4 Band structure

4.1 Theory

In a crystal, the potential must necessarily be periodic in the unit cell, otherwise the discrete translational symmetry is broken. As such the following equation holds for any lattice vector $\mathbf{R}$:

$$V(\mathbf{r} + \mathbf{R}) = V(\mathbf{r}) \tag{4.1}$$

Due to this periodicity, the potential can also be expressed as a sum over reciprocal lattice vectors. To see this, we start with the potential operator:

$$\hat{V} = \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}\, V(\mathbf{r}) \left|\mathbf{r}\right\rangle \left\langle\mathbf{r}\right| \tag{4.2}$$

Inserting two continuous identities for $\mathbf{k}$ and $\mathbf{k}'$, which is the equivalent of taking the Fourier transform of the potential:

$$\hat{V} = \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}\; V(\mathbf{r}) \left[\int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}' \left|\mathbf{k}'\right\rangle \left\langle\mathbf{k}'\right|\right] \left|\mathbf{r}\right\rangle \left\langle\mathbf{r}\right| \left[\int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k} \left|\mathbf{k}\right\rangle \left\langle\mathbf{k}\right|\right], \tag{4.3}$$

$$= \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}' \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k} V(\mathbf{r}) \left|\mathbf{k}'\right\rangle \left\langle\mathbf{k}'|\mathbf{r}\right\rangle \left\langle\mathbf{r}|\mathbf{k}\right\rangle \left\langle\mathbf{k}\right|. \tag{4.4}$$

The two middle brakets are $\langle\mathbf{k}'|\mathbf{r}\rangle = \sqrt{v}^{-1} e^{-i\mathbf{k}'\cdot\mathbf{r}}$ and $\langle\mathbf{r}|\mathbf{k}\rangle = \sqrt{v}^{-1} e^{i\mathbf{k}\cdot\mathbf{r}}$, where v is the volume of the material and $\sqrt{v}^{-1}$ is the normalization factor. This then becomes

$$\hat{V} = \frac{1}{v} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}' \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}\, V(\mathbf{r}) e^{-i(\mathbf{k}'-\mathbf{k})\cdot\mathbf{r}} \left|\mathbf{k}'\right\rangle \left\langle\mathbf{k}\right|. \tag{4.5}$$

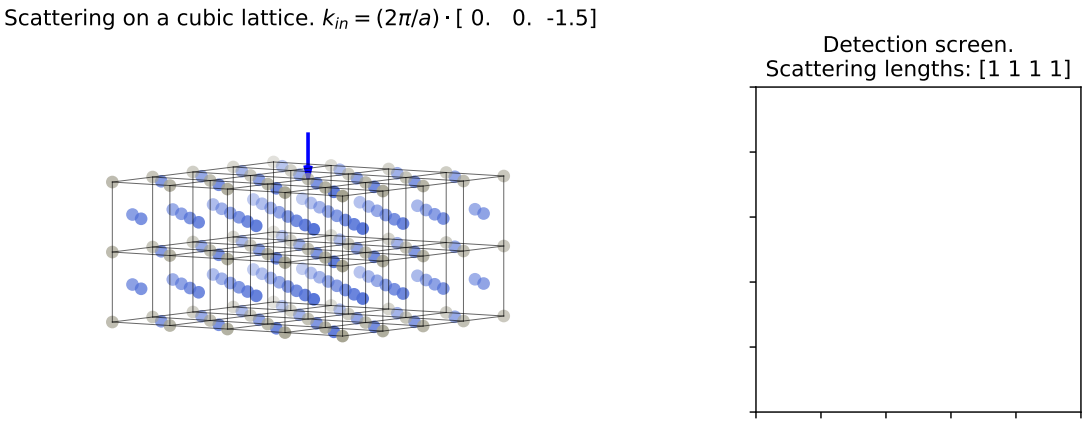


Figure 8: The same scattering setup as in figure 7, but with equal form factors for the four atoms (effectively an fcc lattice). Note the absence of scattering events due to systemic absence, since for each of the sets of Miller indices either $h+k$, $k+l$ or $h+l$ is odd.

Next we use the now familiar trick of letting $\mathbf{r} = \mathbf{R} + \mathbf{x}$. This allows us to write the operator as

$$\hat{V} = \frac{1}{v} \sum_{\mathbf{R}} e^{i(\mathbf{k}-\mathbf{k}')\cdot\mathbf{R}} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}' \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k} \int_{\substack{\text{unit-}\\\text{cell}}} \mathrm{d}\mathbf{x}\ V(\mathbf{x})\ e^{-i(\mathbf{k}'-\mathbf{k})\cdot\mathbf{x}} \left|\mathbf{k}'\right\rangle \left\langle\mathbf{k}\right| \tag{4.6}$$

This sum, like before, equals 0 for $\mathbf{k} - \mathbf{k}' \neq \mathbf{G}$. However, when $\mathbf{k} - \mathbf{k}' = \mathbf{G}$ this sum becomes infinite (given the infinite amount of lattice points) **THERE NEEDS TO BE JUSTIFICATION FOR THIS FOR FINITE MATERIALS**, with a prefactor of $(2\pi)^D/v$ [1]. This then gives

$$\hat{V} = \frac{(2\pi)^D}{v^2} \sum_{G} \delta(\mathbf{k} - \mathbf{k}' - \mathbf{G}) \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}' \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k} \int_{\substack{\text{unit-}\\\text{cell}}} \mathrm{d}\mathbf{x}\ V(\mathbf{x})\ e^{i\mathbf{G}\cdot\mathbf{x}} \left|\mathbf{k}'\right\rangle \left\langle\mathbf{k}\right|, \tag{4.7}$$

$$= \frac{(2\pi)^D}{v^2} \sum_{G} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}\ V_{\mathbf{G}} \left|\mathbf{k} - \mathbf{G}\right\rangle \left\langle\mathbf{k}\right|, \tag{4.8}$$

where $V_{\mathbf{G}}$ is the Fourier transform of the potential, in $\mathbf{G}$, over the unit cell. Now, inserting two additional identities, but this time for $\mathbf{r}$ and $\mathbf{r}'$ gives

$$\hat{V} = \frac{(2\pi)^D}{v^2} \sum_{G} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}\ V_{\mathbf{G}} \left[\int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}' \left|\mathbf{r}'\right\rangle \left\langle\mathbf{r}'\right|\right] \left|\mathbf{k} - \mathbf{G}\right\rangle \left\langle\mathbf{k}\right| \left[\int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r} \left|\mathbf{r}\right\rangle \left\langle\mathbf{r}\right|\right], \tag{4.9}$$

$$= \frac{(2\pi)^D}{v^2} \sum_{G} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}' \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}\ V_{\mathbf{G}} \frac{1}{v} e^{i(\mathbf{k}-\mathbf{G})\cdot\mathbf{r}} e^{-i\mathbf{k}\cdot\mathbf{r}'} \left|\mathbf{r}'\right\rangle \left\langle\mathbf{r}\right|, \tag{4.10}$$

$$= \frac{(2\pi)^D}{v^3} \sum_{G} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}' \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}\ V_{\mathbf{G}}\ e^{-i\mathbf{G}\cdot\mathbf{r}} e^{i\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}')} \left|\mathbf{r}'\right\rangle \left\langle\mathbf{r}\right|. \tag{4.11}$$

We can get rid of two of these integrals, if we use the fact that [2]

$$\delta(\mathbf{r} - \mathbf{r}') = \frac{1}{(2\pi)^D} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{k}\ e^{i\mathbf{k}\cdot(\mathbf{r}-\mathbf{r}')}, \tag{4.12}$$

as we then get

$$\hat{V} = \frac{(2\pi)^{2D}}{v^2} \sum_{\mathbf{G}} \int_{-\infty}^{+\infty} \mathrm{d}\mathbf{r}\ V_{\mathbf{G}}\ e^{-i\mathbf{G}\cdot\mathbf{r}} \left|\mathbf{r}\right\rangle \left\langle\mathbf{r}\right| \tag{4.13}$$

which, if the prefactors and the sum is taken inside the integral, is the same form as we started with! Thus we get our desired result of

$$V(\mathbf{r}) = \frac{(2\pi)^{2D}}{v^2} \sum_{\mathbf{G}} V_{\mathbf{G}}\, e^{-i\mathbf{G}\cdot\mathbf{r}}, \quad V_{\mathbf{G}} = \frac{1}{v} \int_{\substack{\text{unit-}\\\text{cell}}} \mathrm{d}\mathbf{x}\ V(\mathbf{x})\ e^{i\mathbf{G}\cdot\mathbf{x}}. \tag{4.14}$$

EXPLAIN PREFACTORS OR GET RID OF THEM

This allows us to write the Schrödingers equation in a form where the dispersion relation is easily calculated numerically. First we Fourier transform the equation:

$$\int_{-\infty}^{+\infty} e^{-i\mathbf{k}\cdot\mathbf{r}} \left[\frac{\mathbf{p}^2}{2m} + V(\mathbf{r})\right] \psi(\mathbf{r})\ \mathrm{d}\mathbf{r} = \int_{-\infty}^{+\infty} e^{-i\mathbf{k}\cdot\mathbf{r}}\, E\, \psi(\mathbf{r})\ \mathrm{d}\mathbf{r} = E\, \tilde{\psi}(\mathbf{k}). \tag{4.15}$$

The kinetic energy term is just

$$-\frac{\hbar^2}{2m} \int_{-\infty}^{+\infty} e^{-i\mathbf{k}\cdot\mathbf{r}}\, \nabla^2 \psi(\mathbf{r})\ \mathrm{d}\mathbf{r} = \frac{\hbar^2 \mathbf{k}^2}{2m} \tilde{\psi(\mathbf{k})}, \tag{4.16}$$

whilst the potential energy term is

$$\mathcal{F}[V(\mathbf{r})\psi(\mathbf{r})] = \int_{-\infty}^{+\infty} e^{-i\mathbf{k}\cdot\mathbf{r}}\, V(\mathbf{r})\, \psi(\mathbf{r})\ \mathrm{d}\mathbf{r} = \int_{-\infty}^{+\infty} e^{-i\mathbf{k}\cdot\mathbf{r}} \left[\sum_{\mathbf{G}} e^{i\mathbf{G}\cdot\mathbf{r}}\, V_{\mathbf{G}}\right] \psi(\mathbf{r})\ \mathrm{d}\mathbf{r}, \tag{4.17}$$

$$= \sum_{\mathbf{G}} V_{\mathbf{G}} \int_{-\infty}^{+\infty} e^{-i(\mathbf{k}-\mathbf{G})\cdot\mathbf{r}}\, \psi(\mathbf{r})\ \mathrm{d}\mathbf{r}, \tag{4.18}$$

where the integral is just the Fourier transform of the wave function, in $\mathbf{k} - \mathbf{G}$:

$$\mathcal{F}[V(\mathbf{r})\psi(\mathbf{r})] = \sum_{\mathbf{G}} V_{\mathbf{G}}\, \tilde{\psi}(\mathbf{k} - \mathbf{G}). \tag{4.19}$$

With this expression, the whole equation becomes

$$\sum_{\mathbf{G}} \left[\frac{\hbar^2 \mathbf{k}^2}{2m} \delta_{\mathbf{G},0} + V_{\mathbf{G}}\right] \tilde{\psi}(\mathbf{k} - \mathbf{G}) = E\, \tilde{\psi}(\mathbf{k}). \tag{4.20}$$

This equation gives the energy for a single value of $\mathbf{k}$, by relating the state $\tilde{\psi}(\mathbf{k})$ to all other states with the same crystal momentum. If we then consider all the different equations for states with the same crystal momentum $\tilde{\psi}(\mathbf{k} - \mathbf{G})$, we can describe them all as a matrix equation, where the eigenvalues are the energies for the state with wave vector $\mathbf{k}$, in all of the different bands. The first (lowest) eigenvalue is thus the energy of $\tilde{\psi}(\mathbf{k})$ in the lowest band.

To calculate the dispersion relation for a given lattice and potential we then need to find the eigenvalue of the above matrix equation for a range of different $\mathbf{k}$. However, due to the unbounded nature of $\mathbf{G}$, the matrix and vector will both have an infinite amount of elements. For the purposes of the program we need to only allow some set of $\mathbf{G}$, making the matrix and vector finite dimensional.

This can be thought of through the lens of perturbation theory. If we have a free particle (corresponding to no allowed value of $\mathbf{G}$), we just get a 1×1 "matrix", whose eigenvalue trivially is the energy of a free particle. Adding the potential for $\mathbf{G} = 0$ gives the first order perturbation, shifting the state by some constant energy (the matrix equation still only has one allowed state).

Allowing the set of next smallest values for $\mathbf{G}$ would then constitute a second order perturbation, where the particle is allowed to scatter into these states. Further allowing a larger set of $\mathbf{G}$ will give a more accurate calculation of the dispersion relation for the particle, until finally it becomes exact when the full, infinite spectrum of values for $\mathbf{G}$ is included.

4.2 Implementation

So far in these calculations we have not specified the dimensionality of the system. In the following however, we will restrict ourselves to two dimensions and a square lattice with lattice spacing a. The reciprocal lattice vectors $\mathbf{G}$ can then be indexed with the coefficients m_1 and m_2, as $\mathbf{G} = \frac{2\pi}{a}(m_1\hat{\mathbf{x}} + m_2\hat{\mathbf{y}})$. This also means that the sum over $\mathbf{G}$ can be expressed as a double sum over m_1 and m_2.

To describe the matrix it is helpful to first describe the vector in the equation. Let us call this $|\psi\rangle$. This consists of $\tilde{\psi}(\mathbf{k}+\mathbf{G})$ for all the allowed $\mathbf{G}$. Let us also call these $\psi[m_1, m_2]$:

$$|\psi\rangle = \begin{pmatrix} \vdots \\ \tilde{\psi}(\mathbf{k}-\mathbf{G}_1) \\ \tilde{\psi}(\mathbf{k}) \\ \tilde{\psi}(\mathbf{k}+\mathbf{G}_1) \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ \psi_{[0,-1]} \\ \psi_{[0,0]} \\ \psi_{[0,1]} \\ \vdots \end{pmatrix}, \tag{4.21}$$

where $\mathbf{G}_1 = \frac{2\pi}{a}\hat{\mathbf{y}}$. Correspondingly we write $V_{\mathbf{G}}$ as $V_{[m_1,m_2]}$.

The matrix in the above equation can be split into two different matrices: One for the kinetic energy T, which is just a diagonal matrix, and one for the potential energy V. The diagonal elements of the kinetic energy matrix are just the energy of the state, if no potential was present:

$$T = \frac{\hbar^2}{2m}\begin{pmatrix} \ddots & & & & & \\ & (\mathbf{k}-\mathbf{G}_1)^2 & & & & \\ & & \mathbf{k}^2 & & & \\ & & & (\mathbf{k}+\mathbf{G}_1)^2 & & \\ & & & & \ddots & \end{pmatrix}. \tag{4.22}$$

The potential energy matrix is a bit more complicated. It is best described with an example. Say we allow $m_1, m_2 \in \{-1, 0, 1\}$. The sum still has an infinite amount of terms, but we only allow 9 of these in out matrix equation. These terms have pairs of coefficients for ψ that are in the ordered list

$$[m_1, m_2] \in \{[-1,-1], [-1,0], [-1,1], [0,-1], [0,0], [0,1], [1,-1], [1,0], [1,1]\}. \tag{4.23}$$

If we then calculate (part of) the row equation for $m_1 = -1, m_2 = 0$ (which is the second row) we get

$$\begin{aligned} E\psi_{[-1,0]} &= \sum_{m_1'=-\infty}^{\infty}\sum_{m_2'=-\infty}^{\infty} V_{[m_1',m_2']}\ \psi_{[-1-m_1',-m_2']}, \\ &= \cdots + V_{[-2,-1]}\psi_{[1,1]} + V_{[-2,0]}\psi_{[1,0]} + V_{[-2,1]}\psi_{[1,-1]} + \cdots \\ &\quad + V_{[-1,-1]}\psi_{[-2,1]} + V_{[-1,0]}\psi_{[-2,0]} + V_{[-1,1]}\psi_{[-2,-1]} + \cdots \\ &\quad + V_{[0,-1]}\psi_{[-1,1]} + V_{[0,0]}\psi_{[-1,0]} + V_{[0,1]}\psi_{[-1,-1]} + \cdots \\ &\quad + V_{[1,-1]}\psi_{[0,1]} + V_{[1,0]}\psi_{[0,0]} + V_{[1,1]}\psi_{[0,-1]} + \cdots \end{aligned}$$

Now, the 4th to 6th shown terms contain "disallowed" states, i.e. they have coefficients for ψ outside of the allowed range. The other 9 shown terms are "allowed" states however. Because $\psi_{[-1,-1]}$, which is multiplied by $V_{[0,1]}$, is the first pair of coefficients in the ordered set of allowed coefficients, this factor will also be the first element in the corresponding row of the matrix. This row is:

$$\begin{pmatrix} V_{[0,1]} & V_{[0,0]} & V_{[0,-1]} & V_{[1,1]} & V_{[1,0]} & V_{[1,-1]} & V_{[-2,1]} & V_{[-2,0]} & V_{[-2,-1]} \end{pmatrix}. \tag{4.24}$$

This illuminates the structure of the potential energy matrix. The algorithm for constructing it is as follows:

1. Choose a range of values for m_1 and m_2 and arrange them in an oriented set.

2. For each pair of coefficients in this set, ($[m_1, m_2]$, corresponding to some row), calculate $[m_1 - m_1', m_2 - m_2']$, where $[m_1', m_2']$ is all the pairs of coefficients from the same set, and corresponds to the columns of the matrix.

3. This new pair of coefficients, $[m_1 - m_1', m_2 - m_2']$, will be coefficients for the potential at the corresponding matrix element: $V_{[m_1-m_1',m_2-m_2']}$. (The corresponding matrix element is the one with row/column corresponding to the position of $[m_1, m_2]/[m_1', m_2']$ in the ordered set).

With this algorithm in mind, we can see a couple of characteristics of the matrix:

- The diagonal elements are all $V_{[0,0]}$, corresponding to the state not being scattered into any other state.
- The matrix is hermitian, if $V_{[m_1,m_2]} = V^*_{[-m_1,-m_2]}$, which is the same condition one gets when using perturbation theory to calculate the band structure in the Nearly Free Electron Model, see section 1.1. This is all very fortunate since the potential matrix must necessarily be hermitian for the whole Hamiltonian to be hermitian, which it has to be, since it corresponds to an observable quantity, namely energy.

On the last characteristic: Since the potential is real and symmetric (rather, it is even about any lattice point $\mathbf{R}$, in both the x and y-direction), it is guaranteed that $V_{[m_1,m_2]} = V^*_{[-m_1,-m_2]}$. This is because the Fourier transform of a real and even function is also real [2]. As such, for any periodic potential (that is even in both x and y about lattice points) the potential matrix, and therefore also the whole Hamiltonian is hermitian!

With all of this in mind, the potential energy matrix for $m_1, m_2 \in \{-1, 0, 1\}$ is a 9×9 matrix with the following coefficients:

$$V = \begin{pmatrix} V_{[0,0]} & V_{[0,-1]} & V_{[0,-2]} & V_{[-1,0]} & V_{[-1,-1]} & V_{[-1,-2]} & V_{[-2,0]} & V_{[-2,-1]} & V_{[-2,-2]} \\ V_{[0,1]} & V_{[0,0]} & V_{[0,-1]} & V_{[-1,1]} & V_{[-1,0]} & V_{[-1,-1]} & V_{[-2,1]} & V_{[-2,0]} & V_{[-2,-1]} \\ V_{[0,2]} & V_{[0,1]} & V_{[0,0]} & V_{[-1,2]} & V_{[-1,1]} & V_{[-1,0]} & V_{[-2,2]} & V_{[-2,1]} & V_{[-2,0]} \\ V_{[1,0]} & V_{[1,-1]} & V_{[1,-2]} & V_{[0,0]} & V_{[0,-1]} & V_{[0,-2]} & V_{[-1,0]} & V_{[-1,-1]} & V_{[-1,-2]} \\ V_{[1,1]} & V_{[1,0]} & V_{[1,-1]} & V_{[0,1]} & V_{[0,0]} & V_{[0,-1]} & V_{[-1,1]} & V_{[-1,0]} & V_{[-1,-1]} \\ V_{[1,2]} & V_{[1,1]} & V_{[1,0]} & V_{[0,2]} & V_{[0,1]} & V_{[0,0]} & V_{[-1,2]} & V_{[-1,1]} & V_{[-1,0]} \\ V_{[2,0]} & V_{[2,-1]} & V_{[2,-2]} & V_{[1,0]} & V_{[1,-1]} & V_{[1,-2]} & V_{[0,0]} & V_{[0,-1]} & V_{[0,-2]} \\ V_{[2,1]} & V_{[2,0]} & V_{[2,-1]} & V_{[1,1]} & V_{[1,0]} & V_{[1,-1]} & V_{[0,1]} & V_{[0,0]} & V_{[0,-1]} \\ V_{[2,2]} & V_{[2,1]} & V_{[2,0]} & V_{[1,2]} & V_{[1,1]} & V_{[1,0]} & V_{[0,2]} & V_{[0,1]} & V_{[0,0]} \end{pmatrix}. \tag{4.25}$$

Specific potentials implemented in the program include a two dimensional Dirac Comb, and a harmonic potential:

$$V_{\text{dirac}}(\mathbf{r}) = V_0 a^2 \sum_{\mathbf{R}} \delta(\mathbf{r} - \mathbf{R}), \quad V_{\text{harmonic}}(\mathbf{r}) = V_0 \left[\cos\left(\frac{2\pi}{a} x\right) + \cos\left(\frac{2\pi}{a} y\right) \right], \tag{4.26}$$

where the a^2 in the Dirac comb potential arises from the fact that the Dirac delta function carries units m^{-2}. Now, $V_{[m_1,m_2]}$ for the Dirac Comb potential is easily calculated. We let the unit cell run from $-a/2$ to $a/2$ in both x and y, to make sure the delta functions are well within the integration limits. Then it just becomes:

$$V_{[m_1,m_2]} = V_0 \frac{a^2}{a^2} \int_{-a/2}^{a/2} \int_{-a/2}^{a/2} \mathrm{d}x \; \mathrm{d}y \; e^{i2\pi(m_1 x + m_2 y)/a} \delta(\mathbf{r}) = V_0, \tag{4.27}$$

and the potential matrix is just a matrix full of ones, scaled by V_0. For the harmonic potential we let the unit cell run from 0 to a in both directions and get

$$V_{[m_1,m_2]} = \frac{V_0}{a^2} \int_0^a \int_0^a \mathrm{d}x \; \mathrm{d}y \; e^{i2\pi(m_1 x + m_2 y)/a} \left[\cos\left(\frac{2\pi}{a} x\right) + \cos\left(\frac{2\pi}{a} y\right) \right] \tag{4.28}$$

This can be split into two integrals, I_1 and I_2. The first is

$$I_1 = \frac{V_0}{a^2} \int_0^a \int_0^a \mathrm{d}x \; \mathrm{d}y \; e^{i2\pi(m_1 x + m_2 y)/a} \cos\left(\frac{2\pi}{a} x\right), \tag{4.29}$$

$$= \frac{V_0}{2a^2} \int_0^a \mathrm{d}y \; e^{i2\pi m_2 y/a} \int_0^a \mathrm{d}x \; e^{i2\pi m_1 x/a} \left[e^{i2\pi x/a} + e^{-i2\pi x/a} \right], \tag{4.30}$$

$$= \frac{V_0}{2a^2} \int_0^a \mathrm{d}y \; e^{i2\pi m_2 y/a} \int_0^a \mathrm{d}x \; \left[e^{i2\pi(m_1+1)x/a} + e^{-i2\pi(m_1-1)x/a} \right] \tag{4.31}$$

Now if $m_2 = 0$, the integrand in the y-integral is just 1, and the whole integral evaluates to a. If $m_2 \neq 0$ this is not the case. However, the antiderivative evaluated at both ends is the same, and the whole integral is 0:

$$\int_0^a \mathrm{d}y\ e^{i2\pi m_2 y/a} = \frac{a}{i2\pi m_2}\left[e^{i2\pi m_2 y/a}\right]_0^a = 0 \tag{4.32}$$

The same goes for the integrals in x, but with $m_1 \pm 1$ instead of m_2. As such

$$I_1 = \frac{V_0 a^2}{2a^2}\left[\delta_{m_2,0}(\delta_{m_1,1} + \delta_{m_1,-1})\right], \tag{4.33}$$

with a similar result for I_2. $V_{[m_1,m_2]}$ is then

$$V_{[m_1,m_2]} = \begin{cases} \frac{V_0}{2} & \text{if } [m_1, m_2] \in \{[0,1],[0,-1],[1,0],[-1,0]\}, \\ 0 & \text{else.} \end{cases} \tag{4.34}$$

And the potential energy matrix is

$$V_{\text{harmonic}} = \frac{V_0}{2}\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{4.35}$$

With this we have the full Hamiltonian matrix, and we can solve it for a range of $\mathbf{k}$ to get the dispersion relation for a particle in a square lattice.

To calculate this, the program needs values for V_0, a and $\hbar^2/2m$ along with the range of coefficients for $\mathbf{G}$. To make this more simple for the user, we define $k_0 \equiv 2\pi/a$ (the same factor as in the scattering program) and $E_0 \equiv \hbar^2 k_0^2/m$. These allow us to rewrite the equation in a dimensionless form:

$$\sum_{\tilde{\mathbf{G}}}\left[\frac{\tilde{\mathbf{k}}^2}{2}\delta_{\tilde{\mathbf{G}},0} + \tilde{V}_{\tilde{\mathbf{G}}}\right]\tilde{\psi}(\tilde{\mathbf{k}} - \tilde{\mathbf{G}}) = \tilde{E}\tilde{\psi}(\tilde{\mathbf{k}}), \tag{4.36}$$

with $\tilde{\mathbf{G}} \equiv \mathbf{G}/k_0 = m_1\hat{\mathbf{x}} + m_2\hat{\mathbf{y}}$, $\tilde{\mathbf{k}} \equiv \mathbf{k}/k_0$ $\tilde{V}_{\tilde{\mathbf{G}}} \equiv V_{\mathbf{G}}/E_0$ and $\tilde{E} \equiv E/E_0$. This also redefines the first Brillouin zone to be $-1/2 \leq \tilde{k} \leq 1/2$ in each direction.

With this the Hamiltonian matrix is created. The program also takes a value for the number of points in the first Brillouin-zone in k-space and creates linearly spaced points from $-\pi/a$ to π/a.

For each point in the Brillouin zone the program then creates the corresponding kinetic energy matrix, adds the potential energy matrix, and finds the eigenvalues for the resulting matrix. These values are stored in a multidimensional array (represented as a series of matrices, each with size $n_k \times n_k$, where n_k is the number of points in the Brillouin zone, in each direction). When the energies for all coordinates are calculated, the lowest energies for each of the points (corresponding to the first matrix in the multidimensional array) will then be the dispersion of the particle in the first band.

However, we want to find the Fermi sea and surface, and not just the dispersion relation. For a crystal with monovalent atoms the electrons fill up half a band, corresponding to half the area of the first Brillouin zone (in two dimensions). To represent this we use the fact that we have discretised the Brillouin zone into n_k^2 points. The Fermi sea is then the values of $\mathbf{k}$ with have an energy lower than the $n_k^2/2$'th lowest energy.

So to plot only the Fermi sea for a 2D material, we order the energies in the band from lowest to highest, find the middle value, and only plot points that have an energy lower than, or equal to, this energy.

For divalent atoms a whole band is filled, and the occupied states may spill into the second Brillouin zone. As such we need to calculate the dispersion for states in this zone as well. Then we order the energies of the band, take the energy of the n_k^2'th point as the Fermi energy, and plot all points whose energy is lower than or equal to it.

4.3 Examples

Say we just want to find the dispersion of a free particle. Then we just call the program with $\tilde{V}_0 \equiv V_0/E_0 = 0$. For good measure we can specify that the potential should be the Dirac potential:

```
Band_structure(V0=0,
               potential='dirac')
```

And if we want the high-potential limit of the harmonic potential (which happens to be $\tilde{V}_0 = 1$) we write

```
Band_structure(V0=1)
```

Because the program defaults to using the harmonic potential. These lines produce figures 9 and 10 respectively.

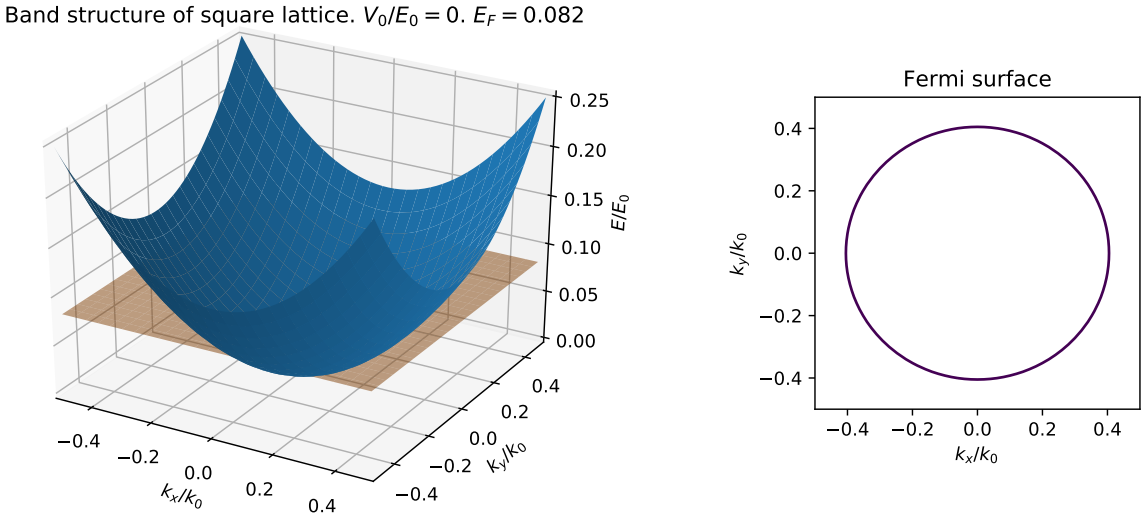


Figure 9: The dispersion relation for a free particle in a square lattice of monovalent atoms, along with the Fermi surface.

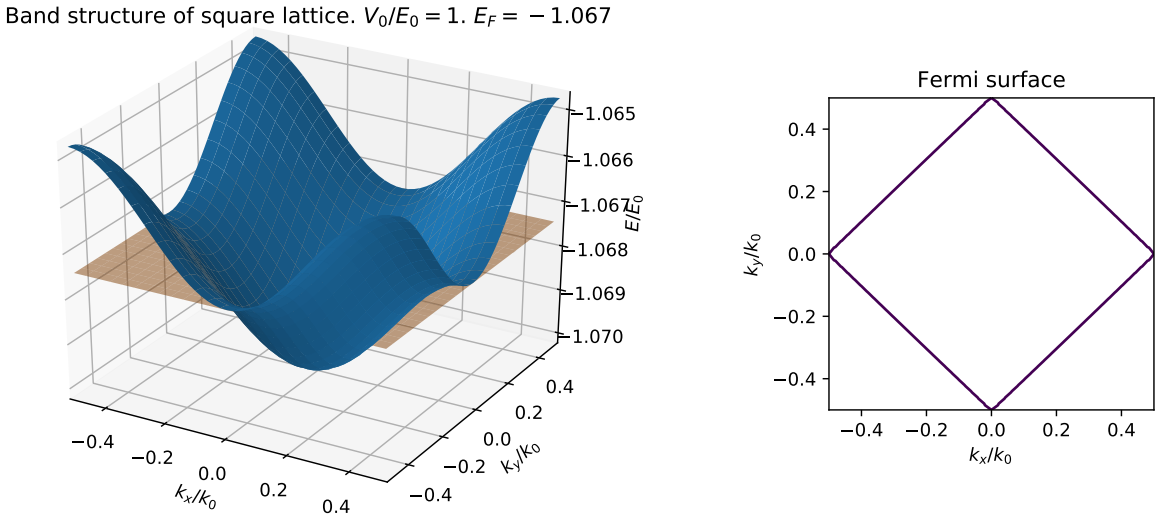


Figure 10: The dispersion relation of a particle in a strong harmonic potential, on a square lattice of monovalent atoms, along with the corresponding Fermi surface.

5 Discussion

In this thesis we set out to create a suite of programs to visualise key concept in the field of condensed matter physics. These concepts included crystal structures, families of lattice planes, a simulation of

neutron scattering, along with band structures of two dimensional materials.

The end result is a code package written in Python which accomplishes just that. It includes 4 main functions: `Lattice`, `Reciprocal`, `Scattering` and `Band_structure`. These 4 functions take in a wealth of arguments, supplied by the user, and produce some sort of figure. Be that the crystal structure of hexagonal lattice with a one atom basis (figure 5), the (001) family of lattice planes for a bcc lattice (figure 6), neutron scattering on an fcc lattice (figures 7 and 8) or the band structure of a monovalent two dimensional material with a high strength potential (figure 10).

More can be done with these programs though, and they are by no means a comprehensive resource. For the lattice plotting program, there is currently no perfect algorithm to detect type of lattice that the user has input.

The algorithm employed to detect lattices rely on the user inputting primitive lattice vectors with the specifications from appendix A. These lattices can be rotated and scaled to the hearts desires, and the program will still detect them. But the algorithm does not take into account the non-uniqueness of primitive lattice vectors.

The "easy" way to take this into account would be to use the method specified in section 2.1, calculating the matrix M for the different lattices and checking whether or not it meets the necessary criteria. However, this method cannot work if the user also rotates the lattice. Different magnitudes of input primitive lattice vectors will also complicate things, as the program would have to check a range of different sets of primitive lattice vectors for each lattice. For example, say the user inputs an fcc lattice with $a_1 = a \cdot (1, 0, 0)$, $a_2 = a \cdot (0, 1/2, 1/2)$ and $a_3 = a \cdot (1/2, 0, 1/2)$. There is currently no way for the program to "know" which magnitude to use for the primitive lattice vectors used for comparison, and the only recourse is to brute-force check all combinations.

So currently no catch-all solution of classification is implemented. A thing to note is that the current algorithms only depend on the primitive lattice vectors. A different approach may be to construct the full crystal and look for a specific "type" of unit cell, corresponding to one of the Bravais lattices. However, this will necessarily involve more complicated calculations, which unfortunately there was not time for.

Further, the programs only calculate the relevant quantities for one specific set of values per function call. That means, that if the user wants to see how systemic absences appear in scattering experiments, or how the Fermi surface distorts for stronger potentials, they will need to manually call the relevant function multiple times with different form factors or potential strength. This gets tedious after a while.

A solution to this is to create some sort of GUI which will display input boxes or sliders for the relevant quantities, automatically call the functions and display the results. This can be done, for example with the python package Flask, which allows the creation of web-applications. These can be run locally on the users machine and allows for javascript integration. This is especially useful (necessary, even!) for accomplishing the goal of added interactivity, as Matplotlib figures can be rendered as javascript objects.

With regards to distortion of the Fermi surface: The next (small) step would be to maybe include different two dimensional lattices. A rectangular lattice would be as simple as altering a value or two in the code, for example. But perhaps a better change would be to bump up the dimensions of the lattices to three. Doing this, however, means we run into problems with the number of available dimensions. It seems we live in a universe with only three spatial dimensions, which means we will use up all of those just specifying the geometry of the lattice, leaving no dimension for any other quantities of interest - like the dispersion relation of the particles in the crystal.

This limits us to only looking at isosurfaces of energy - like the Fermi surface, where we in two dimensions could view the entire Fermi sea (and the rest of the dispersion relation). While this would be enough to view the distortion of the Fermi surface (especially if combined with a web-app as specified above), there is the added issue of the computational time required. Currently, for a two dimensional lattice, with n_k values of $\mathbf{k}$ in each direction, and n_G allowed values of $\mathbf{G}$ (again in each direction), necessitates finding the eigenvalues of n_k^2 matrices of size $n_G^2 \times n_G^2$. By default these values are around 100 and 7 respectively, meaning the program diagonalises 10^4 49×49 matrices. Increasing the dimensionality would increase both the number of matrices to be diagonalised, and the size of these. We would increase the amount of matrices by a factor of n_k, and the size of each of these would go from $n_G^2 \times n_G^2$ to $n_G^3 \times n_G^3$. A very considerable increase.

As it currently stands the program does take a while to compute the band structure (around a second or so, for a decent resolution), and to get the same resolution in three dimensions would take much, much longer. So if three dimensions are to be considered, the program would need to be thoroughly optimised. One way to do this would be to employ an algorithm to find just the lowest eigenvalue of these matrices, as we are only concerned with the lowest band - i.e. the Fermi surface.

A line had to be drawn somewhere, and a two dimensional lattice seemed like a reasonable compromise between added dimensionality and computational complexity.

One last issue is in regards to the programming language and packages chosen. Python is an interpreted language, which means that it necessarily trades computational speed for added ease of development. There are alternatives to this, like Nim or Julia, which are properly compiled or just-in-time compiled respectively. They do not, however, have as large a community as Python does, and therefore not as big a support for third-party packages like Matplotlib.

Further, Matplotlib has one glaring issue in that it does not support a fully fledged 3D graphics engine. This means that all the three dimensional figures in this thesis and the programs are actually just 2D projections of underlying 3D data. This creates artefacts like how there is no proper support for the intersection of surface plots. This can be seen in the band structure program, where the horizontal plane, indicating the Fermi energy, does not properly intersect the dispersion relation.

There are other packages, like Mayavi which do support proper 3D plotting. However, these problems (and their potential solution in this package) were discovered too late in the process of writing, so could not be solved in time.

In conclusion: 4 programs have been created which illustrate concepts in the field of condensed matter physics. While they are not without their flaws or potential for improvement, they do still hold merit, and could be a valuable tool if used in combination with traditional book-based learning.

References

[1] S. H. Simon, *The Oxford Solid State Basics*. Oxford University Press, 2013.

[2] K. F. Riley and M. P. Hobson, *Essential Mathematical Methods for the Physical Sciences*. Cambridge University Press, 2011.

A Lattice Classification

As mentioned in the main text, the choice of primitive lattice vectors is not unique, and the following is just an example of one such choice. It is however the choice used in the program for classification.

Cubic lattices

All of these lattices can be described with a cubic conventional unit cell, with side lengths a. All vectors as such in units of a.

Simple cubic

The simplest lattice. Orthogonal axes and lattice vectors of equal lengths:

$$\mathbf{a_1} = (1, 0, 0), \quad \mathbf{a_2} = (0, 1, 0), \quad \mathbf{a_3} = (0, 0, 1)$$

Face Centered cubic

In the primitive unit cell we chose

$$\mathbf{a_1} = (1/2, 1/2, 0), \quad \mathbf{a_2} = (1/2, 0, 1/2), \quad \mathbf{a_3} = (0, 1/2, 1/2).$$

With this choice, all internal angles are 60 degrees ($\cos\theta = 1/2$). They each form 45 degree angles with respect to two cardinal axis ($\cos\theta = \sqrt{2}/2$), and are orthogonal with respect to last. All lengths equal ($\sqrt{2}/2$).

In a conventional unit cell, the primitive lattice vectors are the same as the simple cubic lattice, but the basis consists of four (row) vectors:

$$\text{basis} = \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \end{pmatrix}$$

Body centered cubic

The choice of primitive lattice vectors is

$$\mathbf{a_1} = (1, 0, 0), \quad \mathbf{a_2} = (0, 1, 0), \quad \mathbf{a_3} = (1/2, 1/2, 1/2).$$

The internal angles for this choice are: $\mathbf{a_1}$ and $\mathbf{a_2}$ are orthogonal. $\mathbf{a_1}$ and $\mathbf{a_2}$ with $\mathbf{a_3}$ has $\cos\theta = \sqrt{3}/3$ (roughly 54.74 degrees). $\mathbf{a_1}$ and $\mathbf{a_2}$ have length 1, while $\mathbf{a_3}$ has length $\sqrt{3}/2$.

For the conventional unit cell we have a two atom basis:

$$\text{basis} = \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \end{pmatrix}$$

Tetragonal

Tetragonal lattices also have orthogonal axes. But the difference between them and cubic lattices, is that tetragonal lattices only have 2 primitive lattice vectors of same length. As such we choose

$$\mathbf{a_1} = (a, 0, 0), \quad \mathbf{a_2} = (0, a, 0), \quad \mathbf{a_3} = (0, 0, b)$$

This is also the choice of primitive lattice vectors with conventional unit cells for the following variants of the tetragonal lattice:

Body centered

This almost has the same primitive lattice vectors as the body centred cubic lattice, except that we increase the z-coordinate on $\mathbf{a_3}$:

$$\mathbf{a_3} = (a/2, a/2, b/2).$$

Then $\cos\theta = a/\sqrt{2a^2+b^2}$ is the angle between $\mathbf{a_1}$ (or $\mathbf{a_2}$) and $\mathbf{a_3}$. And we have $|\mathbf{a_3}| = \frac{\sqrt{2a^2+b^2}}{2}$, giving $\cos\theta = |\mathbf{a_1}|/2|\mathbf{a_3}|$:

$$\begin{aligned}
\mathbf{a_1} = (a,0,0), \quad \mathbf{a_2} = (0,a,0), \quad \mathbf{a_3} &= (a/2, a/2, b/2) \\
|\mathbf{a_1}| = a = |\mathbf{a_2}|, \quad |\mathbf{a_3}| &= \frac{\sqrt{2a^2+b^2}}{2}, \\
\cos\theta_{12} = 0, \quad \cos\theta_{23} = \cos\theta_{31} &= \frac{|\mathbf{a_1}|}{2|\mathbf{a_3}|} = \frac{|\mathbf{a_2}|}{2|\mathbf{a_3}|}
\end{aligned}$$

Face centered

Again we just increase the z-coordinate for the primitive lattice vectors, and get:

$$\begin{aligned}
\mathbf{a_1} = (a/2, a/2, 0), \quad \mathbf{a_2} = (a/2, 0, b/2), \quad \mathbf{a_3} &= (0, a/2, b/2), \\
|\mathbf{a_1}| = \frac{\sqrt{2}}{2}a, \quad |\mathbf{a_2}| = \frac{\sqrt{a^2+b^2}}{2} &= |\mathbf{a_3}|, \\
\cos\theta_{12} = \frac{|\mathbf{a_1}|}{2|\mathbf{a_2}|} = \cos\theta_{31} = \frac{|\mathbf{a_1}|}{2|\mathbf{a_3}|}, \quad \cos\theta_{23} &= \frac{b^2}{a^2+b^2} = \frac{2\mathbf{a_2}^2 - \mathbf{a_1}^2}{2\mathbf{a_2}^2} = \frac{2\mathbf{a_3}^2 - \mathbf{a_1}^2}{2\mathbf{a_3}^2}
\end{aligned}$$

base centered

The base centred tetragonal lattice is like a tetragonal lattice, but with an atom in the middle of the base of each unit cell:

$$\begin{aligned}
\mathbf{a_1} = (a/2, a/2, 0), \quad \mathbf{a_2} = (0, a, 0), \quad \mathbf{a_3} &= (0, 0, b) \\
|\mathbf{a_1}| = \frac{\sqrt{2}}{2}a, \quad |\mathbf{a_2}| = a, \quad |\mathbf{a_3}| &= b \\
\cos\theta_{12} = \frac{\sqrt{2}}{2}, \quad \cos\theta_{31} = \cos\theta_{23} &= 0
\end{aligned}$$

Orthorhombic

The orthorhombic lattice still has orthogonal primitive lattice vectors, but they are all of different magnitudes: Let's assume the conventional unit cell has

$$\mathbf{a_1} = (a,0,0), \quad \mathbf{a_2} = (0,b,0), \quad \mathbf{a_3} = (0,0,c)$$

Body centered

$$\begin{aligned}
\mathbf{a_1} = (a,0,0), \quad \mathbf{a_2} = (0,b,0), \quad \mathbf{a_3} &= (a/2, b/2, c/2), \\
|\mathbf{a_1}| = a, \quad |\mathbf{a_2}| = b, \quad |\mathbf{a_3}| &= \frac{\sqrt{a^2+b^2+c^2}}{2}, \\
\cos\theta_{12} = 0, \quad \cos\theta_{31} &= \frac{a}{\sqrt{a^2+b^2+c^2}} = \frac{|\mathbf{a_1}|}{2|\mathbf{a_3}|}, \\
\cos\theta_{23} &= \frac{b}{\sqrt{a^2+b^2+c^2}} = \frac{|\mathbf{a_2}|}{2|\mathbf{a_3}|}
\end{aligned}$$

And the third length, c, can be expressed as $c^2 = 4\mathbf{a_3}^2 - \mathbf{a_1}^2 - \mathbf{a_2}^2$

These two last are also the angles $\mathbf{a_3}$ make the x-axis and y-axis respectively. Angle with z-axis is $\cos\theta_{3z} = \frac{c}{\sqrt{a^2+b^2+c^2}} = 2(4\mathbf{a_3}^2 - \mathbf{a_1}^2 - \mathbf{a_2}^2)/|\mathbf{a_3}|$

Face centered

$$\begin{aligned}
\mathbf{a_1} = (a/2, b/2, 0), \quad \mathbf{a_2} = (a/2, 0, c/2), \quad \mathbf{a_3} = (0, b/2, c/2),\\
|\mathbf{a_1}| = \frac{\sqrt{a^2+b^2}}{2}, \quad |\mathbf{a_2}| = \frac{\sqrt{a^2+c^2}}{2}, \quad |\mathbf{a_3}| = \frac{\sqrt{b^2+c^2}}{2}\\
\cos\theta_{12} = \frac{a^2}{\sqrt{a^2+c^2}\cdot\sqrt{a^2+b^2}},\\
\cos\theta_{31} = \frac{b^2}{\sqrt{b^2+c^2}\cdot\sqrt{a^2+b^2}},\\
\cos\theta_{23} = \frac{c^2}{\sqrt{a^2+c^2}\cdot\sqrt{b^2+c^2}},
\end{aligned}$$

and the spacings are

$$a^2 = 2(\mathbf{a_1}^2 + \mathbf{a_2}^2 - \mathbf{a_3}^2),\ b^2 = 2(\mathbf{a_1}^2 - \mathbf{a_2}^2 + \mathbf{a_3}^2),\ c^2 = 2(-\mathbf{a_1}^2 + \mathbf{a_2}^2 + \mathbf{a_3}^2).$$

As such the angles can be written

$$\cos\theta_{12} = \frac{\mathbf{a_1}^2 + \mathbf{a_2}^2 - \mathbf{a_3}^2}{2\cdot|\mathbf{a_1}|\cdot|\mathbf{a_2}|}, \quad \cos\theta_{31} = \frac{\mathbf{a_1}^2 - \mathbf{a_2}^2 + \mathbf{a_3}^2}{2\cdot|\mathbf{a_1}|\cdot|\mathbf{a_3}|}, \quad \cos\theta_{23} = \frac{-\mathbf{a_1}^2 + \mathbf{a_2}^2 + \mathbf{a_3}^2}{2\cdot|\mathbf{a_2}|\cdot|\mathbf{a_3}|}$$

Base centered

$$\begin{aligned}
\mathbf{a_1} = (a, 0, 0), \quad \mathbf{a_2} = (a/2, b/2, 0), \quad \mathbf{a_3} = (0, 0, c)\\
|\mathbf{a_2}|^2 = \frac{a^2+b^2}{4}, \quad b^2 = 4\mathbf{a_2}^2 - \mathbf{a_1}^2,\\
\cos\theta_{12} = \frac{a}{\sqrt{a^2+b^2}} = \frac{|\mathbf{a_1}|}{2|\mathbf{a_2}|}, \quad \cos\theta_{31} = \cos\theta_{23} = 0
\end{aligned}$$

Simple monoclinic

The simple monoclinic lattice can be viewed as an infinite series of rectangular lattices stacked on top of each other, but with each by some amount with respect to the one below it. We have:

$$|\mathbf{a_1}| = a,\ |\mathbf{a_2}| = b,\ |\mathbf{a_3}| = c, \quad \cos\theta_{12} = \cos\theta_{23} = 0,\ \cos\theta_{31} \neq 0,$$

where $\mathbf{a_1}$ and $\mathbf{a_2}$ are along the x and y-axes, respectively. $\mathbf{a_3} = c\cdot(\cos\theta_{31}, 0, \sin\theta_{31})$

Base centered monoclinic

$$\begin{aligned}
\mathbf{a_1} = (a, 0, 0), \quad \mathbf{a_2} = (a/2, b/2, 0), \quad \mathbf{a_3} = c\cdot(\cos\theta_{31}, 0, \sin\theta_{31})\\
|\mathbf{a_1}| = a, \quad |\mathbf{a_2}| = \frac{\sqrt{a^2+b^2}}{2}, \quad |\mathbf{a_3}| = c\\
\cos\theta_{12} = \frac{|\mathbf{a_1}|}{2|\mathbf{a_2}|}, \quad \cos\theta_{23} = \frac{a\cos\theta_{31}}{\sqrt{a^2+b^2}} = \frac{\mathbf{a_1}\cdot\mathbf{a_3}}{2\cdot|\mathbf{a_2}|\cdot|\mathbf{a_3}|}
\end{aligned}$$

Hexagonal

The hexagonal lattice is an infinite series of triangular lattices, regularly stacked on top of each other. $\mathbf{a_1}$ and $\mathbf{a_2}$ are orthogonal to $\mathbf{a_3}$, and 60 degrees between them:

$$\begin{aligned}
\mathbf{a_1} = a\cdot(1, 0, 0),\ \mathbf{a_2} = a\cdot(1/2, \sqrt{3}/2, 0),\ \mathbf{a_3} = (0, 0, b)\\
|\mathbf{a_1}| = |\mathbf{a_2}| \neq |\mathbf{a_3}| \quad \cos\theta_{12} = 1/2,\ \cos\theta_{31} = \cos\theta_{23} = 0
\end{aligned}$$

Triclinic

For this lattice we have no equal sides, nor any equal angles:

$$|\mathbf{a_1}| \neq |\mathbf{a_2}| \neq |\mathbf{a_3}|, \quad \theta_{12} \neq \theta_{31} \neq \theta_{23}.$$

An example (from aflowlib.org/CrystalDatabase/triclinic_lattice.html) is

$$\mathbf{a_1} = (a, 0, 0), \quad \mathbf{a_2} = b \cdot (\cos\gamma, \sin\gamma, 0), \quad \mathbf{a_3} = (c_x, c_y, c_z),$$
$$c_x = c\cos\beta, \quad c_y = c\frac{\cos\theta - \cos\beta \cdot \cos\gamma}{\sin\gamma}, \quad c_z = \sqrt{c^2 - c_x^2 - c_y^2}$$

where γ is the angle between $\mathbf{a_1}$ and $\mathbf{a_2}$, β is the angle between $\mathbf{a_1}$ and $\mathbf{a_3}$, and θ is the angle between $\mathbf{a_2}$ and $\mathbf{a_3}$.

Rhombohedral

$|\mathbf{a_1}| = |\mathbf{a_2}| = |\mathbf{a_3}|$ and $\theta_{12} = \theta_{31} = \theta_{23}$ but they're not right angles. An example is

$$\mathbf{a_1} = (a, b, b), \quad \mathbf{a_2} = (b, a, b), \quad \mathbf{a_3} = (b, b, a),$$

which looks like the addition of a simple cubic lattice and a face centred cubic one, with different lattice spacings.

Wurtzite

The Wurtzite lattice is made up of equally spaced triangular lattices (with 120 degrees instead of 60 degrees), with a 4 atom basis:

$$\mathbf{a_1} = a \cdot (1/2, -\sqrt{3}/2, 0), \quad \mathbf{a_2} = a \cdot (1/2, \sqrt{3}/2, 0), \quad \mathbf{a_3} = (0, 0, b),$$
$$\text{basis} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -a\sqrt{3}/3 & b/2 \\ 0 & 0 & u \cdot b \\ 0 & -a\sqrt{3}/3 & (1/2 + u) \cdot b \end{pmatrix}$$

where $u \approx 3/8$ (for the preset in the program we use $u = 3/8$).

Zincblende (and diamond)

The Zincblende is a two atom face centred cubic crystal:

$$\mathbf{a_1} = (1/2, 1/2, 0), \quad \mathbf{a_2} = (1/2, 0, 1/2), \quad \mathbf{a_3} = (0, 1/2, 1/2),$$
$$\text{basis} = \begin{pmatrix} 0 & 0 & 0 \\ 1/4 & 1/4 & 1/4 \end{pmatrix}$$