Visualisation of concepts in condensed matter physics

Nikolai Plambech Nielsen, LPK331

2018-06-13

Contents

1 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, which takes 4 parameters: `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 $|\texttt{a}-\texttt{b}| \leq (\texttt{atol}+\texttt{rtol}\cdot|\texttt{b}|)$, with default values for `atol` and `rtol` being $1\cdot10^{-8}$ and $1\cdot10^{-5}$ 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

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

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 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 $(0, 0, 0)$):

Each of these will specify the primitive lattice vectors for a corresponding primitive unit cell. Furthermore the following crystal presets will also be available: Specifications of all of these presets are available in the appendix. (**LINK TO APPENDIX**)

Mathematically speaking, a lattice is infinite. A physical crystal is, 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.

However, plotting just the atoms will quickly become confusing. Because of this we plot grid lines. 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.

2.2 Implementation

In creating a program that plots crystal 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.

To achieve this, we have to be aware of how Matplotlib handles its 3-dimensional plots. One example of this is that the graphics engine will not hide objects plotted outside of the limits of the figure. This may lead to unpredicted artefacts like atoms "outside" of the crystal (atoms plotted on the edge of the crystal in a unit cell that will not be fully populated).

Another problem may be unfilled unit cells, where we expect them to be filled. Say we have specified an fcc lattice, and we want to view the conventional (cubic) unit cell. Plotting just the atoms at the lattice points with coefficients $n_i \in [0, 1, 2]$ for all i will give a parallelepiped and not the anticipated cube. To do this we need to extend the range of coefficients to fill out any incomplete unit cells. In practise this is done by adding the maximum magnitude of coefficients (in this case 2) to the upper range of coefficients, and subtracting it from the lower range. In this case the resulting range of coefficients is $n_i \in [-2, -1, 0, 1, 2, 3, 4]$.

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 (**LINK TO APPENDIX**). 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.

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**).

Next, the program calculates the limits of the plot box. This is done by calculating the 8 possible vectors arising from linear combinations of the primitive lattice vectors, with coefficients from the specified minimum and maximum coefficients and taking the limits of the plot box as the minimum and maximum values for these 8 vectors. For example, say that the specified minimum and maximum coefficients are $[0, 0, 0]$ and $[2, 2, 2]$ respectively, then the 8 vectors are $\mathbf{v}_1 = \mathbf{0}, \mathbf{v}_2 = 2\mathbf{a}_1, \mathbf{v}_3 = 2\mathbf{a}_2, \mathbf{v}_4 = 2\mathbf{a}_3, \mathbf{v}_5 = 2(\mathbf{a}_1 + \mathbf{a}_2), \mathbf{v}_6 = 2(\mathbf{a}_1 + \mathbf{a}_3), \mathbf{v}_7 = 2(\mathbf{a}_2 + \mathbf{a}_3), \mathbf{v}_8 = 2(\mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3)$. These form a parallelepiped from $2\mathbf{a}_1, 2\mathbf{a}_2, 2\mathbf{a}_3$, where the plot box is a cuboid, with edges along the cardinal axes, that just contain the aforementioned parallelepiped.

With the lattice and basis rotated, and the limits of the plot box found, 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.

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.

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.

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 the one dimensional tight binding example, any plane wave with wave vector k and $k+G_m = k+2\pi m/a$ are physically equivalent. The wave vector is only defined up to a factor of $2\pi/a$. This means that the wave vector can be chosen in the interval $-\pi/a \leq k \leq \pi/a$, which is called 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 **REF**, 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, such that $V(\mathbf{r}+\mathbf{R}) = V(\mathbf{r})$ for any lattice point $\mathbf{R}$. With this, 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, 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 two conditions are statements of conservation of crystal momentum (**BUT WHY?**) and energy respectively.

These are the two conditions for scattering on a crystal, but what about the scattering amplitudes? This is where the integral in Eq. (3.11) comes in. It turns out **REF** that the intensity of the scattered wave vector is proportional to the absolute square of this integral, called the *structure factor*:

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

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. (**INCLUDE GEOMETRIC INTERPRETATION?**)

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 $z = 5$.

In this case the intersection happens when

$$p_{0,z} + t \cdot k'_z = 5, \tag{3.20}$$

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 **I DON'T THINK I'VE INTRODUCED THESE YET** of the reciprocal lattice vector giving rise to the corresponding scattering events.

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.