ASSIGNMENT 2: SOLUTION

E. ARNOLD, M-S. LIU, R. PRABHU & C.S. RICHARDS
THE UNIVERSITY OF CAMBRIDGE

Written in XeLaTeX

FAST FOURIER TRANSFORM

PREREQUISITES

The chapters *Fourier Fransforms* and *Crystal Structures* from the *Theory Handbook*.

ORIGIN OF THE FFT

The fast Fourier transform (FFT) is used in programming to rapidly calculate a representation of the Fourier transform (FT) of a set of data. Among other things, this could be raw experimental data, a mathematical function, or a lattice. We will demonstrate how the mathematical definition of the Fourier transform gives rise to a discrete expression to approximate the FT, when we have enough data to do so.

The FT $F(k)$ of a function $f(x)$ can be defined as:

$$F(k) = \mathcal{F}[f(x)] = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} \mathrm{d}x,$$

where k is a variable, independent of x, in reciprocal space. k is treated as a constant when performing the integral. Don't worry too much about the interpretation of the FT for now! However, note that:

1. The FT of a function represents exactly the same information as the original function. Nothing has "happened" to any information or object, we have just written the information in a different way.
2. The symbol $\mathcal{F}$ is an operator that requires us to take the FT of its input.

The purpose of the first few sections of this document is to show that this mathematical expression is essentially equivalent to discrete Fourier transform (DFT). In MATLAB, the DFT is defined with the expression:

$$F[k] = \sum_{n=0}^{N-1} f[n] \exp(\frac{-2\pi i k n}{N}),$$

where:

1. n is an arbitrary index,
2. N is the number of possible values of the arbitrary index n.

It is important the reader recognises the significance of the notation $f[n]$ and $F[k]$. This notation does not represent $f(n)$, but the nth object in a list labelled f! The nature of these symbols will become clearer shortly. Note that MATLAB does not use "zero-based indexing".

Assumptions & Defining Variables

To start, we must consider how to best represent data we extract from an experiment. For the purpose of the DFT, note that any data used *must* be discrete; we must make an adjustment to how we perceive the function - writing it as a discrete representation of information, rather than a continuous representation.

To proceed, we must think about how to represent sampling a function uniformly along its input. For reasons that will soon be clear, we take N samples at a set of points n. These are sampled along the variable x, separated by a uniform spacing K. These start at 0, forming the set

$$n = \{0, K, 2K, \dots, (N-1)K\}.$$

The underlying assumption of the DFT is that we know the function $f(x)$ at each possible value of the samples n. Then, as an abuse of mathematics, we assume that we can replace the function $f(x)$ in the Fourier transform with the expression:

$$f(x) \sum_n \delta(x-n),$$

where δ is the Dirac delta function. This is not rigorous mathematics: so do not treat it as such. It is merely used to illustrate an origin of the DFT. This form of $f(x)$ is used to indicate that the information we have; we cannot evaluate $f(x)$ at any value of x except the values of n in the set above: so our representation does not need to include any other values of x.

Calculating the FT

Using the statement above, the Fourier transform reads as:

$$F(k) = \int_{-\infty}^{\infty} f(x) \sum_n \delta(x-n) e^{-2\pi i k x} \mathrm{d}x,$$

which should be highly indicative of where this process leads. Using the defining property of the Dirac delta, the integral becomes

$$F(k) = \sum_n f(n) e^{-2\pi i k x}.$$

This is a very similar form to the DFT! However, we will end our mathematical procedure here - pursue more information in your own time, if you wish to do so.

The final comment to make in this section is that the DFT by itself gives a very awkward result. Plotting a delta function is meaningless. How would it be possible to plot something that can't be plotted? The key to the computational process arises from a more subtle result. Dividing the result of the fast Fourier transform by N gives the coefficients of the delta functions making up a FT (see figure 1), rather than the delta functions themselves. This is very useful! It makes it easy for us to see the coefficients of each frequency making up a signal.

THE FFT IN MATLAB

It is often desirable to numerically compute the Fourier transform of a set of data; this can occur in one dimension, two dimensions, or even more: the restriction is arbitrary, but should be selected to match the problem at hand. To illustrate how to perform a Fourier transform in MATLAB, we will use the example of the cosine function. How do we begin? It is reasonable to suggest that we must first generate the domain for a function. Having a set of uniformly distributed data points, it is possible for us to apply the FFT. This is generated with the code:

```
%range of domain
Lx = 5*2*pi;

%number of samples in domain
w = 100;

%space between samples
dx = Lx/w;

%set of samples
x=-Lx/2:dx:Lx/2-dx;
```

Nothing in the listing above should seem too outlandish. The next step is to generate the cosine function over the data set. How do we achieve this? MATLAB's built in cosine function is vectorised, and can automatically act over an entire vector. Thus we can write:

```
f = cos(2*x);
```

Recall that we wish to find a representation of the FT. This is easy to achieve! The built in `fft` function does exactly what we want it to; the data is fast fourier transformed, but the reciprocal basis is *not* determined by `fft`. A translated version of the reciprocal variable is obtained below, in conjunction with the FT of our cosine function.

```
%calculate the FT
F = dx/Lx*fft(f);
```

```

%reciprocal variable spacing
dk = 2*pi/Lx;

%range of reicprocal variable
Lk = 2*pi/dx;

%generate vector for reciprocal variable
k = 0:dk:Lk-dk;
```

How do we correct for the misalignment between the calculated reciprocal variable, and the true reciprocal variable? The built-in `fftshift` function is used.

```
%shifts to appropriate basis
Fshift = fftshift(F);

%the reciprocal variable
k_shift = -Lk/2:dk:Lk/2-dk;
```

Plotting the result of this is fortunately a very easy task. Running the code below forces MATLAB to plot the figure.

```
fig = figure;
plot(k_shift, abs(Fshift), '-');
```

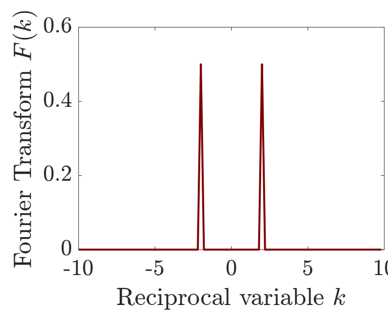

Figure 1 | The FT of $\cos(2x)$, as obtained in MATLAB.

Comment

The reader should note that the definition of the Fourier transform typically used in programming slightly alters the reciprocal variable. Due to the factor of 2π included in the exponential, the basis vectors are a factor of 2π larger than "normal". The more conventional reciprocal variable can be obtained by modifying some lines in the above to change the 2π dependence to 1.

TASKS

1. Use MATLAB to find the Fourier transform of the functions:
   a) $\sin(5x) + \sin(x)$,
   b) x^2,
   c) e^x.
2. Plot the result of each of the Fourier transforms above.

RECIPROCAL LATTICES

PREREQUISITES

The chapters *Crystal Structures* and *Fourier Transforms* from the *Theory Handbook*.

INTRODUCTION

In this chapter, we aim to use the skills developed in the previous chapter to compute the reciprocal lattice for a close-packed plane.

THE DELTA DESCRIPTION OF A LATTICE

Our initial aim is to create a framework from which we can calculate the reciprocal representation of a structure, using FFT methods. This should be much more rapid than attempting to solve the problem manually.

To start, we consider how we can represent a lattice. There are two simple descriptions for the positions of the atoms; we label the number density of atoms $\rho(x, y)$, avoiding confusion that might arise using the symbol n for a second time (*i.e.* the normal symbol), and discuss two equivalent definitions of ρ:

1. The most simplistic description we can use is to state that the number density is 1 on lattice points, and 0 when we are not on lattice points. This represents a description of what the lattice is, but not how to do the algebra. However, this does represent an excellent description for a discrete function. This makes it useful from the computational perspective.
2. A more mathematical definition that helps us to proceed is to state that the number density of atoms ρ follows the relation:

$$\rho(x, y) = \sum_{n=-\infty}^{\infty} \delta(x - x_n) \sum_{m=-\infty}^{\infty} \delta(y - y_m),$$

   where x_n and y_m represent the coordinates of the lattice points (in real space, for now). Thus integrating over a region containing a lattice point yields unity. If there is no lattice point, the integral is 0.

FINDING THE RECIPROCAL LATTICE USING FFT

Theory

What happens when we take the Fourier transform of a set of points? It turns out that we get another set of points. This new set of points is known as the "reciprocal lattice". The mathematics behind this is simple: we simply take the Fourier transform, and "swap" the position of the integral and the sum. The density of points in the reciprocal lattice $P(k_x, k_y)$ can be written in terms of the reciprocal variables k_x and k_y, corresponding to x and y (respectively) as:

$$\begin{aligned} P(k_x, k_y) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \rho(x, y) e^{-2\pi i k_x x} e^{-2\pi i k_y y} \mathrm{d}x \mathrm{d}y; \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(x - x_n) \sum_{m=-\infty}^{\infty} \delta(y - y_m) e^{-2\pi i k_x x} e^{-2\pi i k_y y} \mathrm{d}x \mathrm{d}y; \\ &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(x - x_n) e^{-2\pi i k_x x} \mathrm{d}x \sum_{m=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(y - y_m) e^{-2\pi i k_y y} \mathrm{d}y, \end{aligned}$$

which should be clear to the reader can be evaluated using the defining property of the Dirac delta. Following through with the calculation gives the sum:

$$P(k_x, k_y) = \sum_{n=-\infty}^{\infty} e^{-2\pi i k_x x_n} \sum_{m=-\infty}^{\infty} e^{-2\pi i k_y y_m},$$

which, incredibly, is a type of Fourier series that can describe a set of Dirac delta functions! The reader should refer to the *Theory Handbook* if they cannot see why this is.

The task of the investigator is to find the positions of these delta functions in reciprocal space, thus constructing a "reciprocal lattice".

Note that in order to proceed from the integral to the most recent equation, x and y must be orthogonal. When we naturally extend the method to three dimensions, we will also require the z coordinate to be orthogonal to the other two coordinates! The conclusion we can draw from this is that x, y and z exist in Cartesian space along a normalised canonical basis: otherwise, the calculation will fail. The investigator must consider the representation of their crystal structure in Cartesian space to use the FT method we have described.

Example: A Rectangular 2D Lattice

For the purpose of this example, we have a very simple structure. Simple enough that an analytical approach using mathematics is not tiresome or contrived, but actually very useful to compare against a computational result; this comparison follows naturally from the ability to plot the result in two dimensions.

The key process in this example revolves around the use of the formula above. Since the positions of the atoms are uniformly spaced in both directions, we can write the Fourier transform as:

$$P(k_x, k_y) = \sum_{n=-\infty}^{\infty} e^{-2\pi i k_x x_1 n} \sum_{m=-\infty}^{\infty} e^{-2\pi i k_y y_1 m},$$

which is evidently a product of two geometric series. This happens to be the Fourier series for the expression:

$$\frac{4\pi^2}{x_1 y_1} \sum \delta(k_x - \frac{n}{x_1}) \sum \delta(k_y - \frac{m}{y_1}),$$

which we can compare to the accepted definition of the reciprocal lattice. This model suggests the permitted coordiates (k_x, k_y) that are valid positions for lattice points are:

$$(k_x, k_y) = (\frac{n}{x_1}, \frac{m}{y_1}),$$

which we can easily check! The coefficients of the deltas do not interest us at the moment; all we want to know are the positions of the reciprocal lattice points.

TASKS

1. Finish the problem involving the square lattice, using MATLAB's native `fft2` function. As with all coding problems, feel free to look up as many tips as you can find on the internet! Stackexchange is always useful.
2. In this question, we are concerned with a hexagonal lattice. Set the lattice parameters to whatever you wish.
   a) State the positions of all of the lattice points in a hexagonal lattice in real space, in terms of integer multiples of the lattice vectors.
   b) Look up, and write down, the positions of the reciprocal lattice points for the same real space lattice.
   c) Use MATLAB to find the positions of the reciprocal lattice points, by calculating the Fourier transform of the real space lattice.
   d) Plot the reciprocal space lattice. Use a small circle to denote the position of a lattice point.
   e) Export the graph as a vector graphics file.
3. Explain why it is difficult to resolve to resolve the discrete positions of the lattice points using the Fourier transform.

SUMMARY

Having completed this tutorial, you should be comfortable with:
1. Calculating the FT of functions using MATLAB
2. Calculating reciprocal lattices in MATLAB
3. Exporting figures to vector graphic files

FURTHER READING

1. Riley, Hobson & Bence *Mathematical Methods for Physics and Engineering*. 3rd Edition, Chapter 13.
   Good for a background understanding of FT techniques.

SOLUTIONS

FOURIER TRANSFORMS: MAIN TASKS

Task 1: Question

Use MATLAB to find the Fourier transform of the functions:
a) $\sin(5x) + \sin(x)$,
b) x^2,
c) e^x.

Task 1: Solution

The best method here is to simply plot the solution. We stated that we must use MATLAB, thus finding an analytical solution by hand fails to answer the question. Hence we skip the answer here, as it is included in the answer to task 2.

Task 2: Question

Plot the result of each of the Fourier transforms above.

Task 2: Solution

This is an easy task, following the routine presented in the document; the solution is listed below.

```
%-------------------------------------------------------
% Define the general quantities used to Fourier transform
%-------------------------------------------------------
% range of domain
Lx = 10*2*pi;

% number of samples in domain
w = 1000;

% space between samples
dx = Lx/w;

% set of samples
x = -Lx/2:dx:Lx/2-dx;

% reciprocal variable spacing
dk = 2*pi/Lx;
```

```

% range of reicprocal variable
Lk = 2*pi/dx;

% the reciprocal variable
k_shift = -Lk/2:dk:Lk/2-dk;
%------------------------------------------------------
%------------------------------------------------------




%------------------------------------------------------
% Fourier transform and plot sin(x) + sin(5x)
%------------------------------------------------------
% define a function
f = sin(5*x) + sin(x);

% calculate the FT
F = dx/Lx*fft(f);

% shifts to appropriate basis
Fshift = fftshift(F);

% hold on
hold on

% initialise figure
figure();

% plot the FT
p1 = plot(k_shift, abs(Fshift), '-');

% hold off
hold off

% Requires R2020a or later
exportgraphics(gcf,'sin5x sinx.pdf', 'ContentType', 'vector')
%------------------------------------------------------
%------------------------------------------------------




%------------------------------------------------------
% Fourier transform and plot x^2
%------------------------------------------------------
% define a function
f = x.^2;

% calculate the FT
F = dx/Lx*fft(f);

% shifts to appropriate basis
Fshift = fftshift(F);

% hold on
hold on

% initialise figure
figure();
```

```

% plot the FT
p1 = plot(k_shift, abs(Fshift), '-');

% hold off
hold off

% Requires R2020a or later
exportgraphics(gcf,'x2.pdf', 'ContentType', 'vector')
%-----------------------------------------------------------
%-----------------------------------------------------------




%-----------------------------------------------------------
% Fourier transform and plot e^x
%-----------------------------------------------------------
% define a function
f = exp(x);
%calculate the FT
F = dx/Lx*fft(f);

%shifts to appropriate basis
Fshift = fftshift(F);

% hold on
hold on

% initialise figure
figure();

% plot the FT
p1 = plot(k_shift, abs(Fshift), '-');

% hold off
hold off

% Requires R2020a or later
exportgraphics(gcf,'ex.pdf', 'ContentType', 'vector')
%---------------------------------------------------------
%-----------------------------------------------------------
```

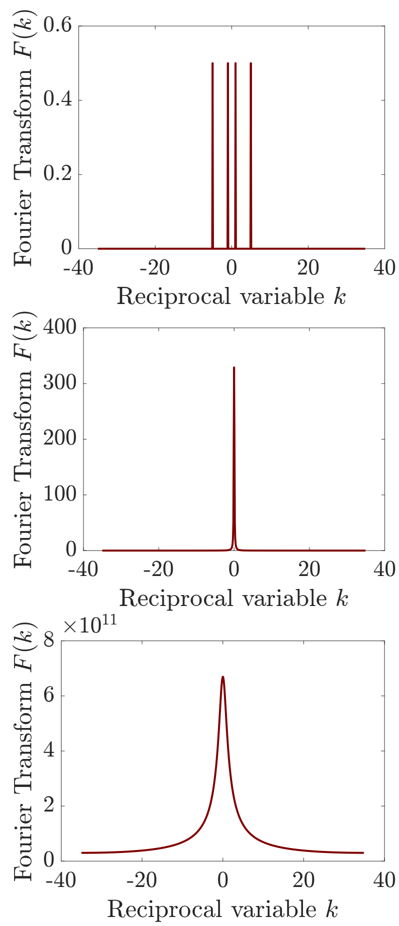

Figure 2 | The Fourier transforms of $\sin(5x) + \sin(x)$, x^2 and e^x. It should be noted that the Fourier transform of e^x does not converge.

RECIPROCAL LATTICES: MAIN TASKS

Task 1: Question

Verify that the FT calculation for the square planar is correct, by calculating the cross product of the basis vectors for a square planar lattice.

Task 1: Solution

The three (unnormalised) basis vectors for a square planar lattice are:

$$\mathbf{a}_1 = x_1\mathbf{x};$$
$$\mathbf{a}_2 = y_1\mathbf{y};$$
$$\mathbf{a}_3 = \mathbf{z}.$$

Thus the reciprocal lattice vectors are:

$$\begin{aligned}
\mathbf{b_1} &= \frac{\mathbf{a}_2 \wedge \mathbf{a}_3}{[\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]}; \\
&= \frac{1}{x_1}\mathbf{x}; \\
\mathbf{b_2} &= \frac{\mathbf{a}_3 \wedge \mathbf{a}_1}{[\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]}; \\
&= \frac{1}{y_1}\mathbf{y}.
\end{aligned}$$

It should be noted that the third reciprocal lattice vector $\mathbf{b_3}$ is completely arbitrary, as it corresponds to unwanted information. Thus the solution is any integer combination of the reciprocal lattice vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ listed above. This recreates the required set of coordinates.

Task 2: Question

In this question, we are concerned with a hexagonal lattice. Set the lattice parameters to whatever you wish.

a) State the positions of all of the lattice points in a hexagonal lattice in real space, in terms of integer multiples of the lattice vectors.
b) Look up, and write down, the positions of the reciprocal lattice points for the same real space lattice.
c) Use MATLAB to find the positions of the reciprocal lattice points, by calculating the Fourier transform of the real space lattice.
d) Plot the reciprocal space lattice. Use a small circle to denote the position of a lattice point.
e) Export the graph as a vector graphics file.

Task 2: Solution

Part *a)* is easily solved. The required points are:

$$\mathbf{r} = m\mathbf{a}_1 + n\mathbf{a}_2,$$

where the coefficients m and n are integers, and $\mathbf{a}_1$ and $\mathbf{a}_2$ are the basis vectors. These basis vectors can be written in terms of their cartesian components as:

$$\mathbf{a}_1 = x_1\mathbf{x};$$
$$\mathbf{a}_2 = x_1(-\frac{1}{2}\mathbf{x} + \frac{\sqrt{3}}{2}\mathbf{y});$$
$$\mathbf{a}_3 = \mathbf{z},$$

where x_1 is the lattice parameter, and $\mathbf{a}_3$ is an out-of-plane basis vector. Thus, in exactly the same way to task 1, we calculate the relevant reciprocal lattice vectors to be:

$$\mathbf{b}_1 = \frac{1}{x_1}(\mathbf{x} + \frac{1}{\sqrt{3}}\mathbf{y});$$
$$\mathbf{b}_2 = \frac{2}{x_1\sqrt{3}}\mathbf{y}.$$

In this task, we are asked to use the Fourier transform.When find a reciprocal lattice, it is not wise to use the DFT as the method to reach the required result; this example, using the DFT, is merely illustrative, demonstrating that it produces the same result as that of the analytical application of the FT, without producing an exact expression for the coordinates of the reciprocal lattice points: the DFT cannot reproduce the Dirac delta functions that we would need to describe a lattice.

Performing the task is a surprisingly easy feat. We must create a set of points that behave as Dirac deltas. This is easy enough to mimic - we simply span the plane with points of finite height at the lattice points. The discontinuity this represents allows us to effectively calculate the discrete Fourier transform of the points to produce the reciprocal lattice. Thus, at each lattice point, we place a step of height 1. The code for this is listed below. First, we setup the basis vectors for the real space lattice:

```
% define the lattice constant
a = 2.71;

% define the first basis vector
a1 = [a 0]';

% define the angle to rotate the first basis vector by to obtain
    the other
theta = 120;

% use this angle in a rotational matrix to perform the rotation
RotM = [cosd(theta) -sind(theta); sind(theta) cosd(theta)];

% produce the second basis
a2 = RotM*a1;
```

Our next step is to produce a coordinate grid in real space and reciprocal space. First, we consider the real space lattice:

```
% Define a rectangular cell which "respects" the periodicity of
    the lattice.
```

```
% In our case, the cell includes the equivalent area of two unit-cells.

% define the range of the space in real space
Lx = [-a/2 a/2];
Ly = [-norm(a1+2*a2)/2 norm(a1+2*a2)/2];

% setup range of real variable in x
lx = linspace(Lx(1),Lx(2),31);

% clear the last entry to respect periodicity
lx(end) = [];

% setup range of real variable in y
ly = linspace(Ly(1), Ly(2), 51);

% clear the last entry to respect periodicity
ly(end) = [];

% setup a usable coordinate grid in real space
[x,y] = meshgrid(lx,ly);

% setup matrix of zeros the same size as the coordinate grid
z = zeros(size(x));

% set the positions where the real lattice is non zero, as multiples of the basis vectors
z(1, 1) = 1;
z(x==0 & y==0) = 1;
```

It should be noted that the last few lines in the listing above generate the lattice points in real space. Our attention is then directed towards the reciprocal lattice. The coordinate for the reciprocal lattice can be generated with:

```
% find the increments along x in reciprocal space
dKx = 2*pi./Lx;

% find the increments along y in reciprocal space
dKy = 2*pi./Ly;

% find the range along x in reciprocal space
Kx = 2*pi/diff(lx(1:2));

% find the range along y in reciprocal space
Ky = 2*pi/diff(ly(1:2));

% setup the range of the reciprocal variable in x
kx_ = linspace(-Kx/2,Kx/2,31);

% clear the last entry to respect periodicity
kx_(end) = [];

% setup the range of the reciprocal variable in y
ky_ = linspace(-Ky/2,Ky/2,51);

% clear the last entry to respect periodicity
ky_(end) = [];

% produce a coordinate grid
[kx, ky] = meshgrid(kx_, ky_);
```

Now that we have all of the relevent points and grids generated, we can draw our attention to producing the reciprocal lattice points. These can be generated using the Fourier transform *in two dimensions*, which is easily realised in Matlab with the `fft2` function. Thus the listing for calculating the reciprocal lattice is given below. We find the index of all positions where the Fourier transform is greater than a cut off position.

```
% Fourier transform the lattice points
Z = real(fftshift(fft2(z)));

% find the positions where the reciprocal variable is non negligible
indx_G = abs(Z) > 1e-3;
```

Our final step is to plot, and export, the figure.

```
% initialise a new figure
figure;

% plot the reciprocal lattice points
% this method of finding the lattice points can be used as the FT commutes with multiplication by a scalar
plot(kx(indx_G), ky(indx_G), 'o', "MarkerSize", 10)

% set the range of the figure axes
axis([-6 6 -5 5]);

% set the axes to be equal in ratio
axis equal

% label the figure axes
xlabel("Kx coordinate")
ylabel("Ky coordinate")

% Requires MATLAB version R2020a or later
exportgraphics(gcf,'FT lattice.pdf','ContentType','vector')
```

The resulting figure for this code is displayed in the margin.

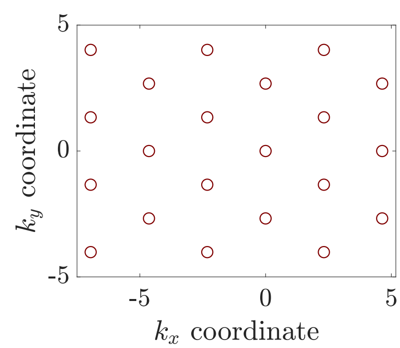

Figure 3 | A plot of the positions of the reciprocal lattice points for a close packed plane.

Task 3: Question

Explain why it is difficult to resolve to resolve the discrete positions of the lattice points using the Fourier transform.

Task 3: Solution

The discrete Fourier transform, unless an infinite number of points are used, will not return Dirac delta functions. Thus the user would have to pick out the positions of the Deltas from a continuous function - which is not necessarily easy. Looking up analytic formulas for reciprocal lattices is much easier than using the Fourier transform.