Lecture 18: The Fourier Transform and its Interpretations

Reading:
Kreyszig Sections: 11.4, 11.7, 11.8, 11.9 (pages496–498, 506–512 513–517, 518–523)

Fourier Transforms

Expansion of a function in terms of Fourier Series proved to be an effective way to represent functions that were periodic in an interval $x \in (-\lambda/2, -\lambda/2)$. Useful insights into "what makes up a function" are obtained by considering the amplitudes of the harmonics (i.e., each of the sub-periodic trigonometric or complex oscillatory functions) that compose the Fourier series. That is, the component harmonics can be quantified by inspecting their amplitudes. For instance, one could quantitatively compare the same note generated from a Stradivarius to an ordinary violin by comparing the amplitudes of the Fourier components of the notes component frequencies.

However there are many physical examples of phenomena that involve nearly, but not completely, periodic phenomena—and of course, quantum mechanics provides many examples of isolated events that are composed of wave-like functions.

It proves to be very useful to extend the Fourier analysis to functions that are not periodic. Not only are the same interpretations of contributions of the elementary functions that compose a more complicated object available, but there are many others to be obtained.

For example:

momentum/position The wavenumber $k_n = 2\pi n/\lambda$ turns out to be proportional to the momentum in quantum mechanics. The position of a function, $f(x)$, can be expanded in terms of a series of wave-like functions with amplitudes that depend

on each component momentum—this is the essence of the Heisenberg uncertainty principle.

diffraction Bragg's law, which formulates the conditions of constructive and destructive interference of photons diffracting off of a set of atoms, is much easier to derive using a Fourier representation of the atom positions and photons.

To extend Fourier series to non-periodic functions, the domain of periodicity will extended to infinity, that is the limit of $\lambda \to \infty$ will be considered. This extension will be worked out in a heuristic manner in this lecture—the formulas will be correct, but the rigorous details are left for the math textbooks.

Recall that the complex form of the Fourier series was written as:

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} \mathcal{A}_{k_n} e^{\imath k_n x} \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda} \\ \mathcal{A}_{k_n} &= \frac{1}{\lambda} \int_{-\lambda/2}^{\lambda/2} f(x) e^{-\imath k_n x} dx \end{aligned} \tag{18-1}$$

where $\mathcal{A}_{k_n}$ is the complex amplitude associated with the $k_n = 2\pi n/\lambda$ reciprocal wavelength or wavenumber.

This can be written in a more symmetric form by scaling the amplitudes with λ—let $\mathcal{A}_{k_n} = \sqrt{2\pi}\mathcal{C}_{k_n}/\lambda$, then

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} \frac{\sqrt{2\pi}\mathcal{C}_{k_n}}{\lambda} e^{\imath k_n x} \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda} \\ \mathcal{C}_{k_n} &= \frac{1}{\sqrt{2\pi}} \int_{-\lambda/2}^{\lambda/2} f(x) e^{-\imath k_n x} dx \end{aligned} \tag{18-2}$$

Considering the first sum, note that the difference in wave-numbers can be written as:

$$\Delta k = k_{n+1} - k_n = \frac{2\pi}{\lambda} \tag{18-3}$$

which will become infinitesimal in the limit as $\lambda \to \infty$. Substituting $\Delta k/(2\pi)$ for $1/\lambda$ in the sum, the more "symmetric result"

appears,

$$f(x) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \mathcal{C}_{k_n} e^{\imath k_n x} \Delta k \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda}$$
$$\mathcal{C}_{k_n} = \frac{1}{\sqrt{2\pi}} \int_{-\lambda/2}^{\lambda/2} f(x) e^{-\imath k_n x} dx \tag{18-4}$$

Now, the limit $\lambda \to \infty$ can be obtained an the summation becomes an integral over a continuous spectrum of wave-numbers; the amplitudes become a continuous function of wave-numbers, $\mathcal{C}_{k_n} \to g(k)$:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(k) e^{\imath k x} dk$$
$$g(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-\imath k x} dx \tag{18-5}$$

The function $g(k = 2\pi/\lambda)$ represents the density of the amplitudes of the periodic functions that make up $f(x)$. The function $g(k)$ is called *the Fourier Transform* of $f(x)$. The function $f(x)$ is called *the Inverse Fourier Transform* of $g(k)$, and $f(x)$ and $g(k)$ are a *the Fourier Transform Pair*.

Higher Dimensional Fourier Transforms

Of course, many interesting periodic phenomena occur in two dimensions (e.g., two spatial dimensions, or one spatial plus one temporal), three dimensions (e.g., three spatial dimensions or two spatial plus one temporal), or more.

The Fourier transform that integrates $\frac{dx}{\sqrt{2\pi}}$ over all x can be extended straightforwardly to a two dimensional integral of a function $f(\vec{r}) = f(x, y)$ by $\frac{dxdy}{2\pi}$ over all x and y—or to a three-dimensional integral of $f(\vec{r}) \frac{dxdydz}{\sqrt{(2\pi)^3}}$ over an infinite three-dimensional volume.

A wavenumber appears for each new spatial direction and they represent the periodicities in the x-, y-, and z-directions. It

is natural to turn the wave-numbers into a **wave-vector**

$$\vec{k} = (k_x, k_y, k_z) = (\frac{2\pi}{\lambda_x}, \frac{2\pi}{\lambda_y}, \frac{2\pi}{\lambda_y}) \tag{18-6}$$

where λ_i is the wavelength of the wave-function in the i^{th} direction.

The three dimensional Fourier transform pair takes the form:

$$\begin{aligned} f(\vec{x}) =& \frac{1}{\sqrt{(2\pi)^3}} \iiint_{-\infty}^{\infty} g(\vec{k}) e^{\imath \vec{k}\cdot\vec{x}} dk_x dk_y dk_z \\ g(\vec{k}) =& \frac{1}{\sqrt{(2\pi)^3}} \iiint_{-\infty}^{\infty} f(\vec{x}) e^{-\imath \vec{k}\cdot\vec{x}} dx dy dz \end{aligned} \tag{18-7}$$

Properties of Fourier Transforms

Dirac Delta Functions

Because the inverse transform of a transform returns the original function, this allows a definition of an interesting function called the Dirac delta function $\delta(x-x_o)$. Combining the two equations in Eq. 18-5 into a single equation, and then interchanging the order of integration:

$$\begin{aligned} f(x) =& \frac{1}{2\pi} \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} f(\xi) e^{-\imath k \xi} d\xi \right\} e^{\imath k x} dk \\ f(x) =& \int_{-\infty}^{\infty} f(\xi) \left\{ \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{\imath k (x-\xi)} dk \right\} d\xi \end{aligned} \tag{18-8}$$

Apparently, a function can be defined

$$\delta(x - x_o) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{\imath k (x-\xi)} dk \tag{18-9}$$

that has the property

$$f(x_o) = \int_{-\infty}^{\infty} \delta(x - x_o) f(x) dx \tag{18-10}$$

in other words, δ picks out the value at $x = x_o$ and returns it outside of the integration.

Parseval's Theorem

The delta function can be used to derive an important *conservation theorem*.

If $f(x)$ represents the density of some function (i.e., a wave-function like $\psi(x)$), the square-magnitude of f integrated over all of space should be the total amount of material in space.

$$\int_{-\infty}^{\infty} f(x)\bar{f}(x)dx = \int_{-\infty}^{\infty} \left\{ \left(\frac{1}{\sqrt{2\pi}} g(k) e^{-\imath kx} dk \right) \left(\frac{1}{\sqrt{2\pi}} \bar{g}(\kappa) e^{-\imath \kappa x} d\kappa \right) \right\} dx \tag{18-11}$$

where the complex-conjugate is indicated by the over-bar. This exponentials can be collected together and the definition of the δ-function can be applied and the following simple result can is obtained

$$\int_{-\infty}^{\infty} f(x)\bar{f}(x)dx = \int_{-\infty}^{\infty} g(k)\bar{g}(k)dk = \tag{18-12}$$

which is Parseval's theorem. It says, that the magnitude of the wave-function, whether it is summed over real space or over momentum space must be the same.

Convolution Theorem

The *convolution* of two functions is given by

$$F(x) = p_1(x) \star p_2(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p_1(\eta) p_2(x - \eta) d\eta \tag{18-13}$$

If p_1 and p_2 can be interpreted as densities in probability, then this convolution quantity can be interpreted as "the total joint probability due to two probability distributions whose arguments add up to x."[9]

The proof is straightforward that the convolution of two functions, $p_1(x)$ and $p_2(x)$, is a Fourier integral over the product of their Fourier transforms, $\psi_1(k)$ and $\psi_2(k)$:

$$p_1(x) \star p_2(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p_1(\eta) p_2(x-\eta) d\eta = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \psi_1(k) \psi_2(k) e^{\imath k x} dk \tag{18-14}$$

This implies that Fourier transform of a convolution is a direct product of the Fourier transforms $\psi_1(k)\psi_2(k)$.

Another way to think of this is that "the net effect on the spatial function due two interfering waves is contained by product the fourier transforms." Practically, if the effect of an aperture (i.e., a sample of only a finite part of real space) on a wave-function is desired, then it can be obtained by multiplying the Fourier transform of the aperture and the Fourier transform of the entire wave-function.

[9] To think this through with a simple example, consider the probability that two dice sum up 10. It is the sum of $p_1(n)p_2(10-n)$ over all possible values of n.

3.016 Home

Full Screen

Close

Quit

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Creating Images of Lattices for Subsequent Fourier Transform

A matrix of intensities (or, the density of scattering objects) is created as a set of "pixels" for imaging. We will use data like this to simulate

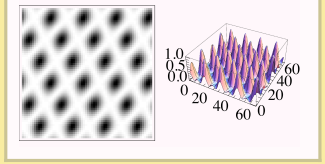
Here we create an image simulating what might be seen in an electron microscope. We will use this data to perform simulated diffraction through use of Fourier Transforms.

```
ISize = 64;
AtomDensity =
  N[Table[(1 + Sin[4 (x + y)
          2 Pi / ISize])
      (1 + Sin[2 (x - 2 y) 2
          Pi / ISize]) / 4,
    {x, 1, ISize}, {y,
     1, ISize}]];
```

1

```
GraphicsRow[
 {ArrayPlot[AtomDensity],
  ListPlot3D[AtomDensity]},
 ImageSize → Full]
```

2



1: `Table` to form a discrete set of points that we will use to approximate am image which image such as might be seen in an transmission electron microscope. For our first set of data, we use interference of two sine waves to produce a simulation of the density of scattering centers in an atomic lattice. Most of the physical aspects of atomic imaging and diffraction can be simulated with the two-dimensional techniques that are produced in these notes. We start with a 64×64 set of discrete points, this is fairly small but it will produce fairly computationally inexpensive results.

2: `ArrayPlot` produces a gray-scale image from an array of "pixel values" between 0 (black) and 1 (white); we use Plot3D to get an additional visualization of the density of scatterers.

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

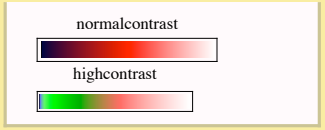
Improving Visualization Contrast with `ColorFunction`

Two examples of `ColorFunction`, *normalcontrast* and *highcontrast* are produced that aid in the interpretation of simulated data. The first, *normalcontrast* , provides a way to use a color, red, to interpolate between black at low intensities and white at high intensities. The second, *highcontrast* , compresses the color change at the low-intensity end; this provides a means to visualize "noise" at low intensities.

To view the data better, we create two versions of a contrast function, the first *(normalcontrast)* is useful when we wish to view the entire range of intensities, the second version *(highcontrast)* when we wish to resolve differences at the low-end of the intensities.

A

`ContrastGraphics` **4**



A: Three input expressions are not shown, but available in the links provided above. The first two define the two *Pure Functions* that will be used at the `ColorFunction` option to graphics objects. The third produces a scale that relates the colors to the intensities.

4: *ContrastGraphics* , defined in item A, shows the relation of colors to intensity.

3.016 Home
Full Screen
Close
Quit

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

ImagePlot

This will be our swiss-army knife visualizer for atomic image and diffraction image graphics.

```
ImagePlot[data_?MatrixQ,
  label_:None,
  colfunc_:highcontrast,
  imagesize_:Medium] :=
 Module[{absdata =
    Abs[data], min, max},
  ArrayPlot[absdata,
   ColorFunction→colfunc,
   BaseStyle→
    {Tiny, FontFamily→
      "Helvetica"},
   PlotLabel→label,
   ImageSize→imagesize]]
```

1

1: *ImagePlot* takes a rectangular array of (possibly complex-valued) intensities are produces graphics from them. It takes three optional functions for the `PlotLabel`, `ColorFunction`, and `ImageSize` which will have default values if not given. It uses `Abs` to find the magnitude of each pixel and then `ArrayPlot` to visualize it. Note, the units of the graphics are the number of pixels along the horizontal and vertical edges.

3.016 Home

Full Screen

Close

Quit

The fast fourier transform (FFT) is a very fast algorithm for compute discrete Fourier transforms (DFT) (i.e., the Fourier transform of a data set) and is widely used in the physical sciences. For image data, the Fourier transform is the diffraction pattern (i.e., the intensity of reflected waves from a set of objects, the pattern results from positive or negative reinforcement or coherence).

However, for FFT simulations of the diffraction pattern from an image, the question arises on what to do with the rest of space *which is not the original image*. In other words, the Fourier transform is taken over all space, but the image is finite. In the examples that follow, the rest of space is occupied by *periodic duplications* of the original image. Thus, because the original image is rectangular, there will always be an additional rectangular symmetry in the diffraction pattern due to scattering from the duplicate features in the neighboring images.

The result of a discrete Fourier transform is a also a discrete set. There are a finite number of pixels in the data, the same finite number of sub-periodic wave-numbers. In other words, the Discrete Fourier Transform of a $N \times M$ image will be a data set of $N \times M$ wave-numbers:

$$\begin{aligned}\text{Discrete FT Data} = 2\pi\left(\frac{1}{N\text{pixels}}, \frac{2}{N\text{pixels}}, \ldots, \frac{N}{N\text{pixels}}\right) \\ \times 2\pi\left(\frac{1}{M\text{pixels}}, \frac{2}{M\text{pixels}}, \ldots, \frac{M}{M\text{pixels}}\right)\end{aligned} \tag{18-15}$$

representing the amplitudes of the indicated periodicities.

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Discrete Fourier Transforms on Simulated Lattices

Example of taking the Discrete Fourier Transform (DFT) of the simulated lattice created above and visualizing it Fourier transform.

We create a function that shows the original data, its Fourier transform, and then its inverse transform (hopefully) back to the original image.

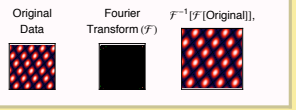
```
FourierRow[data_] :=
 Module[{fourdat =
    Fourier[data]},
  GraphicsRow[
   {ImagePlot[data, "",
     normalcontrast],
    ImagePlot[fourdat] ,
    ImagePlot[
     InverseFourier[
      fourdat], "",
     normalcontrast]},
   ImageSize → Small]]
```

1

Peaks will be located located near (kx,ky) = 2 π (a,b)/(size), where (a,b) = {(0,0), (size,0), (0,size), (size, size)}. These correspond to the longest wavelength periodicities.

```
FourierRow[AtomDensity]
```

2



1: *FourierRow* is a function to visualize, from left to right, the input intensity data, its Fourier transform, and the inverse Fourier Transform of the Fourier Transform. If the final image isn't the same as the first, then something went dreadfully wrong.

2: Notice that the Fourier Transform has very sharp features at the corners of the figure; this is because the original data lattice is a linear combination of sine waves. There are three unique peaks in the pattern. The first, and brightest, sits at $(k_x, k_y) = (2\pi/\lambda_x, 2\pi/\lambda_y) = (1, N_y)$. This peak comes from each pixel interfering with itself in a periodic repetition of the underlying rectangular (in this case, square) lattice. As the number of pixels becomes large, this peak converges to the infinite wavelength limit (or the data with no periodic correlations). This brightest peak is, in fact, the super position of four periodic peaks; the other three are at: $(1 + N_x, 1)$, $(1 + N_x, 1 + N_y)$, and $(N_x, 1 + N_y)$. The reason it appears in the upper left is related to our approximation of dx and dy with $1/N_x$ and $1/N_y$ and the underlying periodicity of the image: there are four choices on which corner to use as the bright spot, upper-left is the one that appears.

The second brightest peak is located at $^\circ 45$ to the highest intensity peak. The peak derives from the $\sin(x + y)$ in one of superposed waves—it is the signature of the planes oriented at $^\circ 45$ $(\bar{1}1)$ in the original data. Each peak corresponds to the superposition of two waves, and its intensity is one-half of the brightest spot which is the superposition of four.

There are just as many low-intensity peaks as mid-intensity. These peaks are derive from the $x - 2y$ modulated sine-wave; their intensity is less because the distance between the corresponding planes is larger.

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Simulating Diffraction Patterns

Materials scientists, microscopists, and crystallographers observe the long wavelength peak at the middle of the diffraction pattern. We develop a data manipulation function that takes input data and outputs the same data, but with our approximation to $\vec{k} = 0$ at the center.

Materials Scientists, Microscopists, and Crystallographers are used to seeing the $\vec{k}$=0 spot at the center of the diffraction image; so we write a function that takes the Fourier data and manipulates to so as to move spots to the center.

```
KZeroMiddle[
  fourmat_?MatrixQ] :=
 Module[{nrows, ncols},
  {nrows, ncols} =
   Dimensions[fourmat];
  RotateRight[fourmat,
   {Quotient[nrows, 2],
    Quotient[ncols, 2]}]]
```

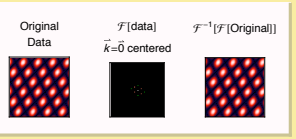
1

And, we modify our FourierRow function to use the k-at-zero transformation: FourierRowK0

A

```
FourierRowK0[AtomDensity]
```

3



1: *KZeroMiddle* uses `RotateRight` to "rolls" the data array so that the left edge goes to the center, followed by the right edge which ends up just to its left at the center; the two columns at the center roll to both edges. The same operation is performed in the vertical direction. To find the center, we use `Quotient` instead of dividing the number of columns and rows by 2 to anticipate the cases where there is an odd number of rows or columns. Fourier transform of the diffraction image are viewed side-by-side.

A–3: *FourierRowK0* duplicates the functionality of *FourierRow* , but the Fourier data is filtered with *KZeroMiddle* before display. The definition of the graphics function is straightforward and suppressed in these class notes versions. This simulates an observed fraction pattern, but with colors instead of gray-scale to indicate intensity of the image's periodicities. *KZeroAtCenter* divides the original matrix data into four approximately equal-sized parts,

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

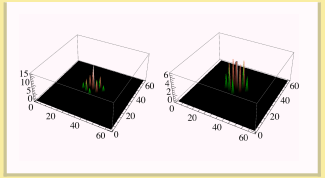
Alternative Representations of Diffraction Data

Because our data is organized as intensities over the x–y plane we can use the z-direction to add another component to visualization. We can also use the same contrast function that we employed for the two-dimensional simulation.

```
Spots3D[data_,
  range_:All] :=
 ListPlot3D[KZeroMiddle[
   Abs[Fourier[data]]],
  ColorFunction →
   (highcontrast[#3] &),
  PlotRange → range]
```
1

```
Spots3DRow[data_,
  range_] := Module[{plt},
  plt = Spots3D[data];
  GraphicsRow[
   {plt, Show[plt,
     PlotRange → range]}]]
```
2

```
Spots3DRow[
 AtomDensity, {0, 7}]
```
3



1: *Spots3D* uses `ListPlot3D` to convert our discrete two-dimensional data into a `Graphics3D` object. We create a default argument for the range of intensities to be plotted, and use *highcontrast* on the z-values.

2–3: *Spots3DRow* takes Fourier data and creates visualizations for the all the intensities, and a second argument for a range, which permits us to observe the finer structure of the diffraction intensities. In this case, because the data is the superposition of two sine-waves, discrete approximations to sharp peaks are observed.

3.016 Home
Full Screen
Close
Quit

Diffraction Patterns of Defective Lattices

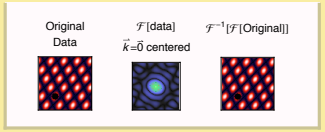
In this example, the simulated atomic density is modified to simulate the removal of one atom—in other words, we simulate a vacancy.

Algorithm to Create a Defect in a Simulated Atomic Density Image : Here, the algorithm will create new data which will be called *AtomDensityWithDefect*

A

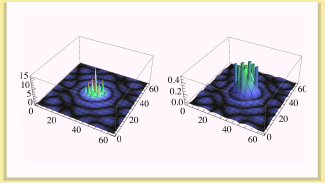
```
FourierRowK0[
 AtomDensityWithDefect]
```

2




```
Spots3DRow[
 AtomDensityWithDefect,
 {0, 1 / 2}]
```

3



A: *AtomDensityWithDefect* are simulated data with a vacancy (definition-algorithm suppressed in class notes). It selects one of the maximum intensity positions at random, and then sets data in, disk centered at that position, to zero.

2–3: The vacancy affects the diffraction pattern with diffuse, low-intensity scattering near $\vec{k} = 0$. The 3D version shows more clearly that the peaks remain the dominate feature, and we have to decrease the range to very small intensities to find the defect scattering all. Nevertheless, the intensity that is shed from the peaks into the entire spectrum reproduces the defect on the reconstructed image.

3.016 Home
Full Screen
Close
Quit

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Diffraction Patterns from Lattices with Thermal 'Noise'

Functions to create a two-dimensional lattices of squares with a specifiable amount of randomness in their position are created. are developed with a variable that simulates random deviation from their ideal lattice positions.

```
Function to Create A Lattice of Squares :
NoisyLattice[TotalSize, LatticeVector1, LatticeVector2, SquareSize, RandomDisplacements]
```
A

```
NoNoise =
  NoisyLattice[64, {8, 4}, {16, 16}, 1, {0, 0}];
```
2

```
FourierRowK0[NoNoise]
GraphicsRow[
 {Spots3D[NoNoise],
  Spots3D[NoNoise, {0, 2}]}]
```
3

```
SomeNoise =
  NoisyLattice[64, {8, 4}, {16, 16}, 1, {1, 1}];
FourierRowK0[SomeNoise]
GraphicsRow[
 {Spots3D[SomeNoise],
  Spots3D[SomeNoise, {0, 2}]}]
```
4

A: *NoisyLattice* 's first argument is the size N of the $N \times N$ that is returned. It also takes input for the two lattice-vectors, the size of squares to place near the lattice positions, and a vector that specifies the magnitude of random displacements in the x and y directions. (The definition, which is a bit long and complicated, is suppressed in the class-notes.) This function will produce smaller unit cells if the lattice vectors are divisors of the data size.

2–3: This simulates data from 'perfect' lattice of squares. Note, that in this case, the entire diffraction pattern is filled. This is because the original data are not sine-waves, but superposed square-wave-patterns. This diffraction pattern is called the *reciprocal lattice* by materials scientists and crystallographers.

4: The data from *SomeNoise* will illustrate the effect of adding isotropic thermal noise (in a real crystal, the amplitude of the noise will be larger in the elastically soft directions, it would not be difficult to modify this function to take a matrix of compliances to multiply the random displacements) A diffuse ring of scattering about $\vec{k} = 0$ is superimposed onto the 'ideal' diffraction pattern.

3.016 Home
Full Screen
Close
Quit

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Computational Microscopy

We produce functions to create circular apertures and interactively move and resize the apertures on the diffraction pattern.

Transmission electron microscopy works by accelerating electrons with a voltage difference and sending them towards a target. The electrons interact with the target and scatter. After the electrons have passed through the sample, they are focused with a magnetic objective lens (or typically lenses). This lens produces a plane at which electrons scattered in the same direction arrive at the same point—this is the diffraction pattern. Because diffraction transforms periodic elements into points, it is closely related to the fourier transform. An image of the target is created beyond this diffraction plane. An operator of an electron microscope can toggle between looking at the diffraction-plane or the image-plane.

The "Bright-Field Image" consists of using a central aperture around the direct beam to block off all others from contributing to the image.

The "Dark-Field Image" consists of selecting a specific diffracted peak with the aperture and using that to form an image.

A "Structure image," or a "lattice image," uses the direct beam and one or more diffracted beams to form the image. In this case, the apertures are typically much larger than for bright- or dark-field imaging.

Aperture size is effectively limited because of spherical aberrations that become significant for beams that are "off-axis" by a significant amount. So, in practice one can only use part of the Fourier spectrum (reciprocal space) to produce an image in TEM. You always lose some structure information in the image formation.

Function to Create Two Circular Aperatures (i.e, to remove all data from a Fourier Transform except the regions inside two specified circles:

CircAps[Center1, Center2, Radius1, Radius2, FourierData]

A

Function to Perform Diffraction Spot Microscopy on Real-Space Images

DiffractionMicroscopy[RealData] **Creates a Interactive Structure with Four Windows Arranged in a Square.**

NorthWest: All Fourier Data from Real Image with Movable Aperatures, use a clicked mouse to move aperatures to various diffraction peaks.

NorthEast: The original real space image.

SouthWest: The Fourier image of the aperature filtered data. (Don't try to move these aperatures)

SouthEast: The reconstructed image from the aperature filtered data.

B

A: *CircAps* is a function (definition suppressed in class-notes) designed to take the positions of the centers of two circular apertures, their radii, and input data (which is intended to be the Fourier transform of scattering density. It returns the data with zeroes everywhere except within the apertures, where it has the same value as the input data.

B: *DiffractionMicroscopy* (definition suppressed, but available at the links given above) takes an array of values representing scattering density, and creates an interactive simulation which allows the user to move the apertures with the mouse, and their radii with slider controls. This definition is only about 20 lines of code, and demonstrates the economy of MATHEMATICA® 6's new `Manipulate` function. We will demonstrate examples below.

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Visualizing Simulated Selected Area Diffraction

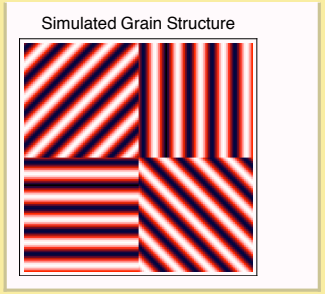
We create a function that creates a square array of square "grains". Each grain will be simplified by creating a sinusoidal modulation in each of the $N_g \times N_g$ with a random orientation picked from an equally spaced set of angles in $(0, N_g^2\pi)$.

GrainStructure[TotalSize, GrainSize]
Creates an array of "square grains" with stripes oriented in somewhat random orientations

A

```
Grains =
  GrainStructure[128, 64];
ImagePlot[Grains,
 "Simulated
   Grain Structure",
 normalcontrast, Large]
```

2



A: This is a definition of the function *GrainStructure* (suppressed in class-notes). Its first argument is the number of pixels along one side of the image, and the second is the number of pixels along the side of the grains. It is best to make the grain size a divisor of the image size.

2: This is an example of creating data and imaging it for a 2×2 grain structure.

3.016 Home

Full Screen

Close

Quit

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

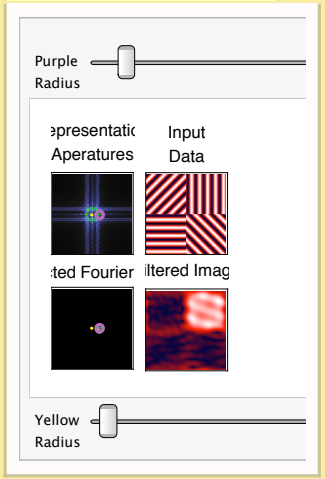
Simulated Diffraction Imaging on a Polycrystal

We use our simulated grain structure as input to our diffraction simulator, *DiffractionMicroscopy* .

Diffraction Spot Microscopy on grain structure.

```
DiffractionMicroscopy[
 Grains]
```

1



1: Here is an example of our interactive function, *DiffractionMicroscopy* , on a polycrystal. You can move each aperture by mouse-dragging and control their sizes with the sliders. You can only move the apertures in the upper-left image. Try these simple experiments first:

Image Orientation of a Single Grain: Move the purple aperture over on of the green diffraction peaks. Notice that only one grain is imaged in the reconstruction. Because the yellow aperture is picking up data from the $\vec{k} = 0$ spot, the other aperture is producing the modulation of a single sine wave.

Imaging a Single Grain: Shrink the yellow aperture to zero; this is a "dark-field" simulation. Move the purple aperture over a single peak; notice that a single grain is imaged, but its modulation has disappeared.

Imaging a Defect: Keeping the yellow aperture at zero-radius, move the purple apertures over one of the streaks in the diffraction pattern. Notice that a *grain boundary* is imaged—pay attention to the orientation of the grain boundary relative to that of the streak in the diffraction image.

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

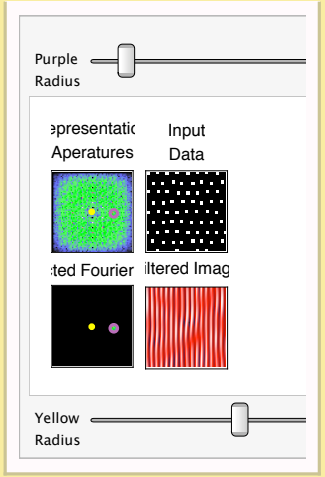
notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Bright-Field and Dark-Field Imaging of a Lattice with Thermal Noise

We use our ideal lattice of squares and a similar one with a bit of thermal noise to continue our investigation of diffraction phenomena.

```
DiffractionMicroscopy[
 NoNoise]
```
1

```
DiffractionMicroscopy[
 SomeNoise]
```
2



1: Here, we use our computational microscope, *DiffractionMicroscopy* , on an ideal lattice. Here are some experiments to try:
Image a Set of Planes Leave the yellow aperture over the $\vec{k} = 0$ peak. Move the purple aperture one of the nearby spots. Notice the orientation and periodicity of the planes in the reconstructed image.
Detect Periodicity Leave the yellow aperture at $\vec{k} = 0$. Move the purple aperture from one of the nearby peaks to a similarly oriented one along the same ray from the origin. Notice that the orientation of the planes do not change, but the period of modulation is increased. (Also, recall that this is a complex superposition of sine-waves to make squares, in the simple superposition of sine-waves, there we fewer peaks.)

2: Here, we do the same example with our randomly perturbed lattice of squares. There is a significant amount of diffuse scattering, but by observing carefully, you will see the the same peaks that were present for the perfect lattice.
Discover Robustness of Imaging with Noise If you leave the yellow aperture over $\vec{k} = 0$, and move the purple aperture over one of the "perfect peaks," you will see a reconstruction of perfect planes even though they are barely discernible in the original image. If you shrink the purple radius, while leaving it over the peak, you will see the quality of the planes improves as less diffuse scattering is included. This simulates how a microscopist can image individual atom planes at finite temperatures where atoms are vibrating around their equilibrium positions.

3.016 Home
Full Screen
Close
Quit

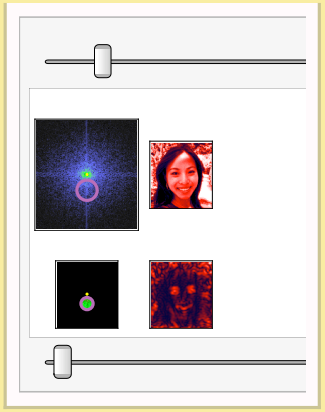
notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Selected Area Diffraction on Image Data

DiffractionMicroscopy is used on data that is extracted from a gray-scale image-file.

```
AnImage = Import[
  "http://pruffle.mit.edu/
    3.016-2007/MsChang.
    pgm"]
```
1

```
DiffractionMicroscopy[
 1 - ChangData]
```
2



1: We read in an image that is stored on the 'net. We do a bit of plastic surgery on this data to put it into a form that is ready for our microscope (plastic surgery algorithm suppressed in class-notes)

2: We perform selected area diffraction on this image. Notice that we can highlight different aspects of the image by selecting different aperture locations.

3.016 Home

Full Screen

Close

Quit

Index

3.016 Home

Full Screen

Close

Quit