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

3.016 Home
Full Screen
Close
Quit

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."[10]

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 $\psi_2(k)$:

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

[10] 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

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.

Creating Images of Lattices for Subsequent Fourier Transform

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A very large matrix of ones (white) and zeroes (black) is created as a set of "pixels" for imaging. The white regions arranged as 8×8 squares in a rectangular patterns. A diffraction pattern from a group of scattering centers such atoms is related to the Fourier transform of the "atom" positions.

```
1  WhiteSquare = Table[1, {i, 8}, {j, 8}];
   BlackSquare = Table[0, {i, 8}, {j, 8}];
   Join[WhiteSquare, BlackSquare, BlackSquare] //
     MatrixForm

2  latcell = Join[
       Transpose[Join[BlackSquare,
           BlackSquare, BlackSquare, BlackSquare]],
       Transpose[Join[BlackSquare, BlackSquare,
           WhiteSquare, BlackSquare]],
       Transpose[Join[BlackSquare, BlackSquare,
           WhiteSquare, BlackSquare]],
       Transpose[Join[BlackSquare, BlackSquare,
           BlackSquare, BlackSquare]]
     ];

3  ListDensityPlot[latcell, MeshStyle → {Hue[1]}];

4  ColumnDuplicateNsq[matrix_, nlog2_] :=
     Nest[Join[#, #] &, matrix, nlog2]

5  ListDensityPlot[ColumnDuplicateNsq[latcell, 2],
     MeshStyle → {Hue[1]}]

6  RowDuplicateNsq[matrix_, nlog2_] :=
     Transpose[ColumnDuplicateNsq[matrix, nlog2]]

7  ListDensityPlot[RowDuplicateNsq[latcell, 2],
     MeshStyle → {Hue[1]}]

8  XtalData = Transpose[
       ColumnDuplicateNsq[RowDuplicateNsq[latcell, 3], 3]];

9  DisplayLater = DisplayFunction → Identity;
   DisplayNow = DisplayFunction → $DisplayFunction;

10 ImagePlot[data_] := ListDensityPlot[data,
     Mesh -> False, ImageSize → 144, DisplayLater]

11 XtalImage = ImagePlot[XtalData]

12 Show[XtalImage, DisplayNow, ImageSize → 400]
```

1: `Table` is used to created "submatrices" of 8×8 ones or zeroes. `Join` will combine the rows of matrices and creates a "tall skinny" matrix from of three square ones. "pixel images" of lattices by placing ones (white) and zeroes (black) in a rectangular grid.

2: *latcell* will be a 32×32 black region with an 16×8 white rectangle near the center. The `Transpose` of four `Join`-ed squares will be a short-fat matrix. Joining four of the resulting `Transpose` operations produces the square matrix.

3: `ListDensityPlot` produces a grayscale image from an array of "pixel values" between 0 (black) and 1 (white).

4: *ColumnDuplicateNsq* takes a matrix as an argument and then recursively duplicates its *rows* into a matrix that has the *same number of columns* as the original. It makes at copy of all the rows at the first iteration, doubling the number of rows—at the second iteration it copies all the rows of the previous result quadrupling the number of rows, and so on. *ColumnDuplicateNsq* uses `Nest` with a *pure function*.

8: The result of calling *RowDuplicateNsq* and *ColumnDuplicateNsq* with "recursion" arguments of 3, creates an $8^3 \times 8^3$ matrix with a square lattice of white rectangles.

9: *DisplayLater* and *DisplayNow* are examples of rule definitions that can be passed to `Show` to delay display or to show a delayed display.

12: *XtalImage* will be used for the Fourier transfrom "diffraction" simulations in the following example.

3.016 Home
Full Screen
Close
Quit

Fast Fourier Transforms and Simulated Diffaction

The fast fourier transform (FFT) is a very fast algorithim 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 subperiodic wavenumbers. In other words, the Discrete Fourier Transform of a $N \times M$ image will be a data set of $N \times M$ wavenumbers:

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

representing the amplitudes of the indicated periodicities.

3.016 Home

Full Screen

Close

Quit

Discrete Fourier Transforms

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Example of taking the DFT of the perfect lattice created above and visualizing the diffraction pattern.

```
1 FourierData = Fourier[XtalData];

FourierColor := ColorFunction → (If[# < .1, Hue[1, 0, 0],
                Hue[.66 * (1 - #), 1, 0.5 + 0.5 #]] &);
2 FourierImagePlot[data_] := ListDensityPlot[Abs[data],
    Mesh -> False, ImageSize → 144,
    FourierColor, DisplayLater]

3 FourierImage = FourierImagePlot[FourierData]

Show[GraphicsArray[{XtalImage, FourierImage,
4       ImagePlot[Chop[InverseFourier[FourierData]]]},
  ImageSize → 1000, DisplayNow]
```

1: *FourierData* is the DFT (obtained with `Fourier`) of *XtalImage* .

2: *FourierColor* is a special `ColorFunction` for visualizing diffraction patterns. If the intensity is very low (¡0.1), the result will be black; otherwise it will scale from blue at low intensities to red at the highest intensity. *FourierImage* is a function to display the result of a DFT, it uses `Abs` to get the magnitudes of the imaginary data set created by `Fourier`.

4: Notice that the DFT has very sharp features, this is because the underlying lattice is perfect. Each feature represents a different periodic function in the direction $\vec{k} = (k_x, k_y)$.

Visualizing Diffraction Patterns

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Visualization examples are created and a function is constructed to move the *longest wavelength (i.e.,* $\vec{k} \approx 0$ *periodicities* to the center of the resulting pattern.

1: Using `GraphicsArray`, the original image, its diffraction image, and the inverse Fourier transform of the diffraction image are viewed side-by-side.

2: Diffraction images are usually observed with the long wavelength features at the center of the image, instead of at the corners. *KZeroAtCenter* divides the original matrix data into four approximately equal-sized parts, and then exchanges the data from the northwest and southeast parts of the original matrix and exchanges the northeast and southwest data. The result is an image with $\vec{k} \approx 0$ at the center.

3: *FourierImagePlot* takes input Fourier-transformed data, rearranges the diffraction image and produces an image with $\vec{k} \approx 0$ at the center.

5: This will be a row similar to (1) above, but with the diffraction pattern adjusted.

```
1  Show[GraphicsArray[{XtalImage, FourierImage,
       ImagePlot[Chop[InverseFourier[FourierData]]]},
   ImageSize → 1000, DisplayNow]
```

Microscopists are used to seeing the "k=0" point in the center of the fourier image (i.e., the periodic information at the center). We can write a function that translates the k=0 point to the center of the image and redisplay the result:

```
2  KZeroAtCenter[matdat_] := Block[
     {rows = Dimensions[matdat][[1]],
      cols = Dimensions[matdat][[2]],
      halfcol, halfrow, colrem, rowrem},
     halfcol = Round[cols/2]; halfrow = Round[rows/2];
     colrem = cols - halfcol; rowrem = rows - halfrow;
     Return[
       BlockMatrix[
         {
           {Take[matdat, -rowrem, -colrem],
            Take[matdat, -rowrem, halfcol]},
           {Take[matdat, halfrow, -colrem],
            Take[matdat, halfrow, halfcol]}
         }
       ]
     ]
   ]
```

```
3  FourierImagePlot[data_] :=
     ListDensityPlot[KZeroAtCenter[Abs[data]], Mesh -> False,
       ImageSize → 144, FourierColor, DisplayLater]
```

```
4  FourierImage = FourierImagePlot[FourierData]
```

```
5  Show[GraphicsArray[{XtalImage, FourierImage,
       ImagePlot[Chop[InverseFourier[FourierData]]]},
   ImageSize → 1000, DisplayNow]
```

Diffraction Patterns of Defective Lattices

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Data from the perfect lattice is used to create a defect by *scalar multiplying* with another matrix with a *small hole* created with zeroes.

1: *HoleFunc* uses the size of the data to create a matrix of ones with a rectangular region of zeroes at a specified position and size.

3: A 12×12 hole is created at position $28, 28$.

4: The hole is multiplied by the original perfect crystal to create a defect and the diffraction pattern is obtained.

5: The defect gives rises to "diffuse" scattering near $\vec{k} = 0$.

```
1  HoleFunc[data_, xc_, yc_, twicew_, twiceh_] :=
     Module[{nrows, ncols}, nrows = Dimensions[data][[1]];
       ncols = Dimensions[data][[2]]; Table[
         If[And[Abs[j - xc] <= twicew, Abs[i - yc] <= twiceh],
           0, 1], {i, nrows}, {j, ncols}]];

2  XtalData = Transpose[
       ColumnDuplicateNsq[RowDuplicateNsq[latcell, 3], 3]];

3  hole = HoleFunc[XtalData, 28, 28, 6, 6];

4  XtalData = Transpose[
       ColumnDuplicateNsq[RowDuplicateNsq[latcell, 3], 3]];
   XtalData = hole * XtalData;
   XtalImage = ImagePlot[XtalData];
   FourierData = Fourier[XtalData];
   FourierImage = FourierImagePlot[FourierData];

5  Show[GraphicsArray[{XtalImage, FourierImage,
       ImagePlot[Chop[InverseFourier[FourierData]]]},
     ImageSize → 1000, DisplayNow]
```

3.016 Home

Full Screen

Close

Quit

Diffraction Patterns from Lattices with Thermal 'Noise'

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Functions to create a larger family of two-dimensional lattices are developed with a variable that simulates random deviation from a perfect position.

1: *MakeLattice* takes input for the width and height of the resulting lattice image, structures for the lattice vectors and the number of repeats to produce, a size for the 'atom,' and a random amplitude from which to simulate noise. This function is not very well-constructed and doesn't always work perfectly. I'll improve it someday.

3: This will display the original 'perfect' lattice, its resulting diffraction pattern, and the inverse fourier image of the diffraction pattern.

5: This will illustrate the effect of adding thermal noise: a diffuse ring will be superimposed on the original diffraction pattern.

```
1  MakeLattice[W_, H_, latvecA_,
     latvecB_, size_, randrange_] :=
    Module[{result = Table[0, {i, H}, {j, W}], lata = -1, latb = -1,
      xpos, ypos, untouched = Table[True, {i, H}, {j, W}]},
     For[lata = 0, lata ≤ latvecA[[3]],
      For[latb = 0, latb ≤ latvecB[[3]], xpos =
          Mod[lata * latvecA[[1]] + latb * latvecB[[1]], H, 1];
        ypos = Mod[lata * latvecA[[2]] + latb * latvecB[[2]],
            W, 1]; If[untouched[[ypos, xpos]],
          untouched[[ypos, xpos]] = False;
          xpos += Random[Integer, randrange];
          ypos += Random[Integer, randrange];
          For[j = 1, j ≤ size, For[i = 1, i ≤ size,
              result[[Mod[ypos + j, H, 1],
                  Mod[xpos + i, W, 1]]] = 1; i++]; j++]];
       latb++];
      lata++]; result
    ]

2  latdata = MakeLattice[400, 400, {0, 20, 40},
      {16, 4, 25}, 4, {0, 0}]; fourlat = Fourier[latdata];

3  Show[GraphicsArray[
      {ImagePlot[latdata], FourierImagePlot[fourlat],
       ImagePlot[Chop[InverseFourier[fourlat]]]}],
    ImageSize → 1000, DisplayNow]
```

The noise is simulated by making small random displacements of each "atom" about its site in the perfect crystal, then computing the Fourier transform of the resulting somewhat imperfect crystal...

```
4  thermallatdata =
     MakeLattice[400, 400, {0, 20, 40}, {16, 4, 25}, 4, {-2, 2}];
   thermalfourlat = Fourier[thermallatdata];

5  Show[GraphicsArray[{ImagePlot[thermallatdata],
       FourierImagePlot[thermalfourlat],
       ImagePlot[Chop[InverseFourier[thermalfourlat]]]}],
    ImageSize → 1000, DisplayNow]
```

3.016 Home

Full Screen

Close

Quit

Using an Aperature to Select Particular Perioidicities in a Diffraction Pattern

notebook (non-evaluated) pdf (evaluated) html (evaluated)

This is a function designed to select particular regions from fourier transforms of a perfect lattice and the perturbation of a perfect lattice, and then display eight images in two columns. The left column of graphics illustrates (from top to bottom) the "clean" input image, the entire fourier transform with the rectangular aperature illustrated, the "reconstructed image" that derives from the fourier transform of the aperature region, and finally a magnified image of the fourier transform within the aperature only.

A,B The adjusted ($\vec{k} = 0$ at center) input data from the Fourier transforms of the reference lattice and one that will create a 'diffuse' pattern.

B,C The diffraction images of the data.

E,F Data from only a selected portion of values $\Delta\vec{k}_x, \Delta\vec{k}_y$ of the input data. This data should only have the periodicities of the original lattice for these selected values.

G–L The images associated with all the data and their reconstructions.

M Producing the array of graphics.

```
Compare[sharpdata_, sharpfourierdata_,
    diffusedata_, diffusefourierdata_, ApCenterx_,
    ApCentery_, ApTwicewidth_, ApTwiceheight_] :=
 (*Details left out of code in this displayed version,
  full code in notebook, html, and pdf of results*);
Module[{(*... local variables...*)},
  (*A*)shiftedsharpfourier = KZeroAtCenter[sharpfourierdata];
  (*B*)shifteddiffusefourier = KZeroAtCenter[diffusefourierdata];
  (*C*)sharpfourimage = ListDensityPlot[Abs[shiftedsharpfourier]];
  (*D*)
  diffusefourimage = ListDensityPlot[Abs[shifteddiffusefourier]];
  (*... Compute Aperature Function... *);
  (*E*)sharpfourieraperature =
    aperature * shiftedsharpfourier;
  (*F*)diffusefourieraperature =
    aperature * shifteddiffusefourier;
  (*G*)sharpapimage = Show[sharpfourimage, Aperature];
  (*H*)diffuseapimage =
    Show[diffusefourimage, Aperature];
  (*I*)sharpfourmagimage =
    ListDensityPlot[Abs[shiftedsharpfourier]];
  (*J*)diffusefourmagimage =
    ListDensityPlot[Abs[shifteddiffusefourier]];
  (*K*)sharprevfourimage =
    ListDensityPlot[Abs[Chop[InverseFourier[
          KZeroAtCenter[sharpfourieraperature]]]]];
  (*L*)diffuserevfourimage = ListDensityPlot[
    Abs[Chop[InverseFourier[
          KZeroAtCenter[diffusefourieraperature]]]]];
  (*M*)Show[GraphicsArray[{
      {ImagePlot[sharpdata], ImagePlot[diffusedata]},
      {sharpapimage, diffuseapimage},
      {sharprevfourimage, diffuserevfourimage},
      {sharpfourmagimage, diffusefourmagimage}
    }
   ];
  ]
]
```

3.016 Home

Full Screen

Close

Quit

Visualizing Simulated Selected Area Diffraction

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of selecting particular periodicities from ideal input and 'noisy' input data.

1: An ideal lattice and its Fourier transform are computed.

2: A 'thermal' perturbation of the ideal lattice and its Fourier transform are computed.

3: An example of selecting only those periodicities within 50 increments of $\vec{k} = 0$.

9: Creating input data where the 'phonon modes' have anisotropic amplitudes.

10: If the 'thermal vibration' appears only in the vertical direction, the resulting diffraction pattern gets 'streaked' in the horizontal direction.

```
1  latdata =
     MakeLattice[400, 400, {0, 20, 40}, {16, 4, 25}, 4, {0, 0}];
   fourlat = Fourier[latdata];

2  thermallatdata =
     MakeLattice[400, 400, {0, 20, 40}, {16, 4, 25}, 4, {-1, 1}];
   thermalfourlat = Fourier[thermallatdata];

3  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 0, 0, 50, 50]

4  Compare[latdata, fourlat, thermallatdata,
     thermalfourlat, 100, 100, 25, 25]

5  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 20, 30, 15, 15]

6  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 30, 30, 15, 15]

7  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 35, 25, 15, 15]

   Modify the function MakeLattice to make "noise" anisotropic

8  MakeLattice[W_, H_, latvecA_, latvecB_, size_,
     Xrandrange_, Yrandrange_] := («details in notebook»)

   The following data only has fluctuations in the vertical direction:

9  thermallatdata = MakeLattice[400, 400,
     {0, 20, 40}, {16, 4, 25}, 4, {0, 0}, {-4, 4}];
   thermalfourlat = Fourier[thermallatdata];

   The resulting Fourier transform gets "streaked" horizontally

10 Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 0, 0, 200, 200]
```

Discrete Fourier Transforms of Real Images

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A image in graphics format, such as a .png, contains intensity as a function of position. If the function is gray-scale data, then each pixel typically takes on 2^8 discrete gray values between 0 and 255. This data can be input into MATHEMATICA® and then Fourier transformed. Images used here and below can be obtained from http://pruffle.mit.edu/3.016/Images.

1: An image in a number of different graphics formats can be imported into MATHEMATICA® with `Import`.

3: The image data is stored in a complicated format, but the gray values (indexed as integers between 0 and 255) are stored as the first item in the first list.

6: This illustrates the original image, its diffraction pattern, and reconstructed image.

Importing an image into Mathematica, .png is some of many graphics data types that Mathematica can process.

(Note: all of the images below can be downloaded from http://pruffle.mit.edu/3.016/Images/

```
1 AnImage = Import["/Users/ccarter/classes/
        3016/Images/fourier_xtal_data.png"];
2 Show[AnImage, DisplayNow]
3 ImageData = AnImage[[1, 1]] / 255;
4 Dimensions[ImageData]
5 FourierImageData = Fourier[ImageData];
6 Show[GraphicsArray[{ImagePlot[ImageData],
      FourierImagePlot[FourierImageData],
      ImagePlot[Chop[InverseFourier[FourierImageData]]]}],
  ImageSize → 1000, DisplayNow]
```

Selected Area Diffraction on Image Data

notebook (non-evaluated) pdf (evaluated) html (evaluated)

An function, *ImageFourierAperature* , that reads in the name of an image file and information about which regions of $\vec{k}$-space to select is developed.

A *fourierdata* is the shifted fourier transform of the image-file's data.

C *aperature* is a matrix of zeroes and ones representing the region in $\vec{k}$-space to be retained.

H This will display four images. To the right of the orginal image will be the diffraction pattern with an indication of where the aperature is located. Below the diffraction image will be a magnification of the aperature region. Below the original image will be the reconstructed image obtained from only those periodicities available in the aperature.

```
ImageFourierAperature[imagefile_,
    Apxmin_, Apxmax_, Apymin_, Apymax_] :=
  Module[{theimage = Import[imagefile], dims,
      nrows, ncols, fourierdata, fourimage,
      fourieraperature, apimage, fourmagimage,
      revfourierimage, aperature, xll, yll, xur, yur},
    (*A*) fourierdata = KZeroAtCenter[
      Fourier[(theimage[[1, 1]] / 255)]];
    (*B*) fourimage = ListDensityPlot[Abs[fourierdata],
      Mesh -> False, ImageSize → 144,
      FourierColor, DisplayLater];
    dims = Dimensions[fourierdata];
    nrows = dims[[1]]; ncols = dims[[2]];
    xll = Round[ncols / 2 + Apxmin * ncols / 2];
    yll = Round[nrows / 2 + Apymin * nrows / 2];
    xur = Round[ncols / 2 + Apxmax * ncols / 2];
    yur = Round[nrows / 2 + Apymax * nrows / 2];
    xll = If[xll < 1, 1, xll]; xur = If[xur > ncols, ncols, xur];
    yll = If[yll < 1, 1, yll]; yur = If[yur > nrows, nrows, yur];
    (*C*) aperature =
      Table[If[And[i ≥ yll, i ≤ yur, j ≥ xll, j ≤ xur], 1, 0],
1       {i, nrows}, {j, ncols}];
    (*D*) fourieraperature = aperature * fourierdata;
    (*E*) apimage = Show[fourimage, Graphics[{
          Hue[.1667, 1, 1], Thickness[2 / nrows],
          Line[{{xll - 1, yll - 1},
              {xur + 1, yll - 1}, {xur + 1, yur + 1},
              {xll - 1, yur + 1}, {xll - 1, yll - 1}}]}]];
    xll = If[xll < 1, 1, xll]; xur = If[xur > nrows, nrows, xur];
    yll = If[yll < 1, 1, yll]; yur = If[yur > ncols, ncols, yur];
    (*F*) fourmagimage = ListDensityPlot[
      Abs[fourierdata[[Range[yll, yur], Range[xll, xur]]]],
      Mesh -> False, FourierColor, DisplayLater];
    (*G*) revfourimage = ListDensityPlot[Abs[Chop[
        InverseFourier[KZeroAtCenter[fourieraperature]]]],
      Mesh → False, DisplayLater];
    (*H*) Show[GraphicsArray[{
        {theimage, apimage},
        {revfourimage, fourmagimage}
      }
    ], ImageSize → 1000,
    GraphicsSpacing → {.001, .0}, DisplayNow];
  ]
```

3.016 Home

Full Screen

Close

Quit

Visualizing Selected Area Diffraction on Image Data

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples and interpretations of using the function *ImageFourierAperature* on several input files. Images used here and below can be obtained from http://pruffle.mit.edu/3.016/Images.

1-4 Examples of selecting diffraction spots from an image of a penrose tiling. The 'streaks' come from the lines between tiles. Picking out particular $\vec{k}$ image lines of similar tilt in the reconstructed image.

5: Picking out particular periodicities allows one to image a selected set of oriented grains in a polycrystal.

6: Of course, one is not limited to playing with images of crystals and tilings...

```
1 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/penrose.png",
    −1, 1, −1, 1]
2 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/penrose.png",
    −0.1, 0.1, −0.2, 0.2]
3 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/penrose.png",
    .04, .14, .05, .15]
4 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/penrose.png",
    .14, .24, .11, .21]
5 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/polycrystal.png",
    .1, .3, .2, .4]
6 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/AB.gif", −1, 1, −1, 1]
7 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/AB.gif",
    −0.5, 0.5, −0.05, 0.05]
8 ImageFourierAperature[
    "/Users/ccarter/classes/3016/Images/AB.gif",
    −0.05, 0.05, −0.5, 0.5]
```

Index