2D and 3D Fourier transforms

The 2D Fourier transform

The reason we were able to spend so much effort on the 1D transform in the previous chapter is that the 2D transform is very similar to it. The integrals are over two variables this time (and they're always from $-\infty$ to ∞ so I have left off the limits). The FT is defined as

$$G(u,v) = \iint g(x,y) e^{-i2\pi(xu+yv)}\, dx\, dy \qquad (1)$$

and the inverse FT is

$$g(x,y) = \iint G(u,v) e^{i2\pi(xu+yv)} du dv\,. \qquad (2)$$

The Gaussian function is special in this case too: its transform is a Gaussian.

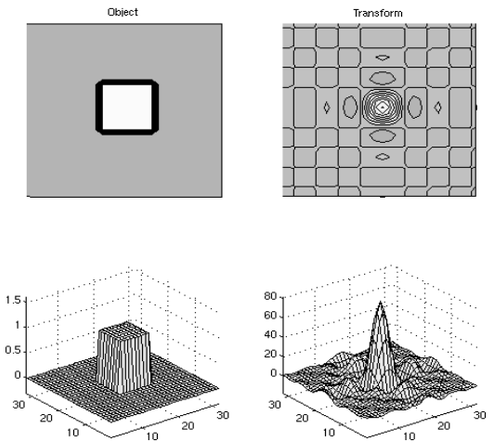
$$e^{-\pi(x^2+y^2)} \xrightarrow{FT} e^{-\pi(u^2+v^2)} \qquad (3)$$

The Fourier transform of a 2D delta function is a constant

$$\delta(x)\delta(y) \xrightarrow{FT} 1 \qquad (4)$$

and the product of two rect functions (which defines a square region in the *x,y* plane) yields a 2D sinc function:

$$\mathrm{rect}(x)\,\mathrm{rect}(y) \xrightarrow{FT} \mathrm{sinc}(u)\,\mathrm{sinc}(v)\,. \qquad (5)$$

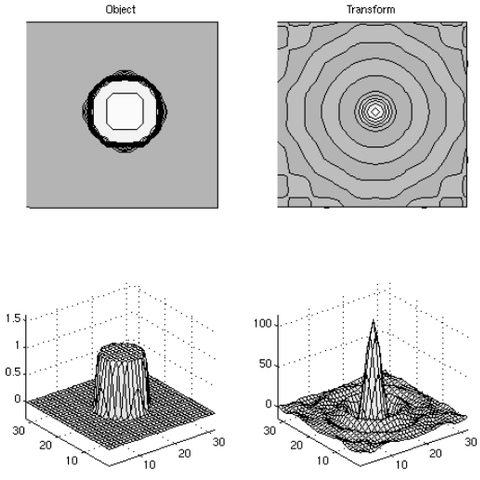


The function rect(x)rect(y) is shown on the left. Its transform is the function sinc(u)sinc(v) shown on the right. (Ignore the units in the axes, they are the units of the discrete FT used to make the figure.)

One special 2D function is the circ function, which describes a disc of unit radius. Its transform is a Bessel function,

$$\mathrm{circ}(r) \xrightarrow{FT} \frac{J_1(2\pi\rho)}{\rho} \qquad (6)$$

here the variables r and ρ represent $(x^2+y^2)^{1/2}$ and $(u^2+v^2)^{1/2}$, respectively. Again, as in the case of the rect function, something with "sharp edges" in one domain transforms into something with ripples in the other.



The circ function (shown on the left) has the transform on the right (a Bessel function, also known as the Airy function.) The Airy function is circularly symmetric, but doesn't quite look like that here because of aliasing artifacts from the discrete FFT on a computer (more about that later).

The 2D FT has a set of properties just like the 1D transform.

1. Linearity
$$g+h \xrightarrow{FT} G+H$$

2. Scale
$$g(ax,by) \xrightarrow{FT} \frac{1}{|ab|} G\left(\frac{u}{a},\frac{v}{b}\right)$$

3. Shift
$$g(x-a,y-b) \xrightarrow{FT} G(u,v)e^{-i2\pi(au+bv)}$$

4. Convolution
$$g * h \xrightarrow{FT} G \cdot H$$

5. Rotation
$$g(x',y') \xrightarrow{FT} G(u',v'),$$
where (x',y') and (u',v') are rotated about the origin through the same angle.

The rotation property is the only one we haven't seen before. You can understand it this way: if we define the vectors $\mathbf{x}=(x,y)$ and $\mathbf{u}=(u,v)$ then we can rewrite the definition of the FT (eqn. 13) as

$$G(\mathbf{u}) = \iint g(\mathbf{x}) e^{-i2\pi \mathbf{x}\cdot\mathbf{u}}\, d\mathbf{x} \qquad (7)$$

where the vectors appear only through a dot product in the exponential function. Thus if you rotate the coordinate system for **x**, a corresponding rotation of **u** will give the same result for the dot product.

More formally, let **R** be a rotation matrix. Then the FT of a rotated function $g(\mathbf{Rx})$ can be gotten through the substitution $\mathbf{x}' = \mathbf{Rx}$ in this way:

$$\begin{aligned} G_{\mathbf{R}}(\mathbf{u}) &= \iint g(\mathbf{Rx}) e^{-i2\pi \mathbf{x}\cdot\mathbf{u}}\, d\mathbf{x} \\ &= \iint g(\mathbf{x}') e^{-i2\pi (\mathbf{R}^{-1}\mathbf{x}')\cdot\mathbf{u}}\, d\mathbf{x}' \\ &= \iint g(\mathbf{x}') e^{-i2\pi \mathbf{x}'\cdot(\mathbf{Ru})}\, d\mathbf{x}' \\ &= G(\mathbf{Ru}) \end{aligned}$$

Like the 1D lattice, a 2D lattice with unit spacing transforms into the same function in the other domain. Making use of the scaling rule, it is then easy to show that the general 2D lattice transforms this way:

$$\left[\sum_{n=-\infty}^{\infty} \delta(ax-n)\right]\left[\sum_{m=-\infty}^{\infty} \delta(by-m)\right] \xrightarrow{FT} \frac{1}{ab}\left[\sum_{n=-\infty}^{\infty} \delta\left(\frac{u}{a}-n\right)\right]\left[\sum_{m=-\infty}^{\infty} \delta\left(\frac{v}{b}-m\right)\right] \qquad (8)$$

This means that a lattice with spacings $1/a$ and $1/b$ transforms to a lattice with spacings a and b, respectively. This is the origin of the term "reciprocal space" for the Fourier transform space.

2D Power spectrum

The 2D power spectrum is an important tool in electron microscopy. Recall from last time that the power spectrum is the magnitude squared of the FT, so in 2D it is

$$S(u,v) = |F(u,v)|^2. \qquad (9)$$

If the random signal has been filtered by some filter function H such that $F = G \cdot H$, then the expectation value of the spectrum is given by

$$S(u,v) = |G(u,v)|^2 |H(u,v)|^2 \qquad (10)$$

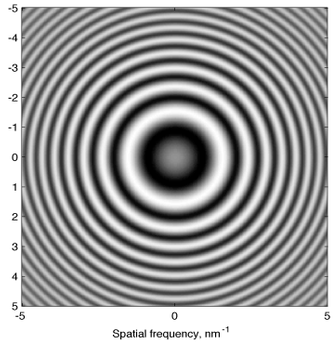
Let's let H be the contrast transfer function. In two dimensions the CTF should be circularly symmetric (why do you think?) We'll define the magnitude s of the spatial frequency as

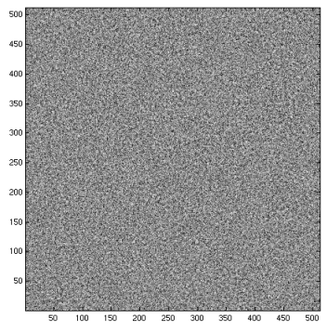
$$s = \sqrt{u^2 + v^2}$$

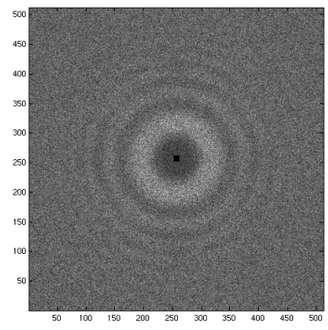
so the CTF has the form—ignoring spherical aberration but including the envelope function

$$H(s) = \sin(-\pi\delta s^2 - \alpha) e^{-Bs^2/4}. \qquad (11)$$

Below left is the theoretical CTF, the middle is a micrograph, and on the right is the power spectrum of the micrograph. Note that the power spectrum looks a lot like the CTF (actually according to (10) it is proportional to the CTF squared).

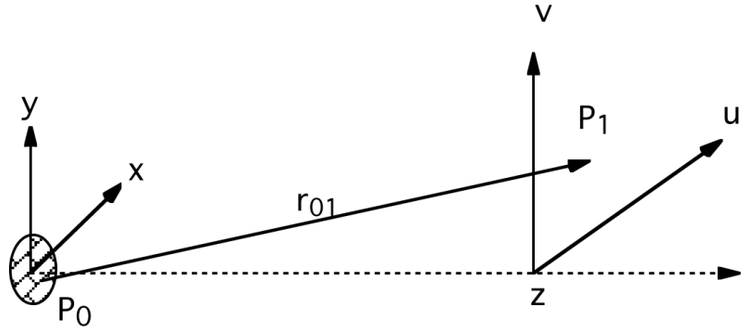






The 2D FT and diffraction

The diffraction pattern is the Fourier transform of the amplitude pattern of a source of radiation. Consider the following system. A plane wave is propagating in the +*z* direction, passing through a scattering object at z=0, where its amplitude becomes $A_o(x,y)$. We want to know the amplitude of the wave at the detector in the *u,v* plane, which is a distance *z* from the *x,y* plane.



Classical wave theory says that we can solve this problem by treating each point in the *x,y* plane as the source of a spherical wavefront. Here is the wave function of a spherical wave of magnitude A_0 that starts from the origin,

$$\Psi(r) = \frac{A_0}{r} e^{i2\pi r/\lambda},$$

and the wave function at the detector plane is gotten by adding up a spherical wave emanating from each point in the x,y plane. Using the weak-phase approximation we'll scale the amplitude of each diffracted wave by $1/i\lambda$ and get:

$$\psi_d(u,v) = \iint_{x,y} \frac{A_0(x,y)}{i\lambda\sqrt{(u-x)^2+(v-y)^2+z^2}} e^{i2\pi\sqrt{(u-x)^2+(v-y)^2}/\lambda}\, dxdy.$$

Making various approximations, specifically assuming *u* and *v* are large compared to *x* and *y* and that *z* is much larger than all other dimensions, the Fraunhofer formula for the wave amplitude at the detector plane is

$$\Psi_d(u,v) = \frac{e^{i2\pi z/\lambda}}{i\lambda z} e^{i2\pi(u^2+v^2)/2\lambda z} \iint_{(x,y)} A_o(x,y) e^{-i2\pi(xu+yv)/\lambda z}\, dx\, dy \qquad (12)$$

or, letting G be the Fourier transform of the amplitude distribution $A(x,y)$,

$$\Psi_d(u,v) = \frac{e^{i2\pi z/\lambda}}{i\lambda z} e^{i\pi(u^2+v^2)/\lambda z}\; G\left(\frac{u}{\lambda z}, \frac{v}{\lambda z}\right) \qquad (13)$$

Where G is the Fourier transform of the amplitude distribution $A_0(x,y)$. The intensity of X-rays or photons at the detector will be the squared magnitude of Ψ_d, in which the various phase factors drop out and we have

$$I(u,v) = \frac{1}{\lambda z}\left|G\left(\tfrac{u}{\lambda z}, \tfrac{v}{\lambda z}\right)\right|^2 \qquad (14)$$

The intensity at the detector is proportional to the magnitude squared of the FT of the scattering pattern. This is the basis of electron crystallography. Working from the measured intensity at the detector, the first step is to find what the amplitude is at the detector (which you get by taking the square root of the intensity and assigning phases). Then the IFT give you the original pattern of amplitudes A_o at the object.

The 3D Fourier transform

In the same way, there exists a 3D Fourier transform as well. It is defined as a triple integral, and it has all the properties of the 2D FT, including rotations.

$$G(u,v,w) = \iiint g(x,y,z) e^{-i2\pi(xu+yv+wz)}\, dx\, dy dz \qquad (15)$$

Central to the theory of 3D reconstruction is the "central slice theorem". It is based on the fact that for any 3D distribution of density $g(x,y,z)$ there is a 3D Fourier transform volume $G(u,v,w)$. If you take a projection through g to obtain a 2D image, it turns out that the Fourier transform of that image has the same values as slice through G. If you have data to fill up the Fourier volume with slices, then you can do an inverse transform to obtain the density map g. We'll talk more about this next time.

Discrete Fourier transform and terminology

In this course we will be talking about computer processing of images and volumes involving Fourier transforms. The computer operates on data that have been sampled at regular, finite intervals and produces results that we view as individual pixels or voxels. There is an alternative Fourier transform in which the integrals are replaced by sums for working on such finite data sets. This is called the discrete Fourier transform (DFT). Its theory and use are very similar to those of the continuous FT that we have discussed here. There are some considerations about sampling intervals that have to be made, but otherwise things work just fine.

In the 1960s people discovered a beautiful algorithm for computing the DFT that is extremely efficient. It is called the fast Fourier transform. Its results are exactly the same as the DFT, only obtained much

more quickly. Here is an example. Suppose you want to do the 2D DFT of a 1000 x 1000 pixel image. To get the 1000 x 1000 element DFT, you have to do 10^{12} arithmetic operations (just think, you have to use all values of *x, y, u* and *v* in the calculation). It is spectacular that this calculation can be done more than 10^4 times faster with the FFT. The FFT of a 1000 x 1000 image takes less than 0.1 second on my laptop computer, requiring about 10^8 operations!

So, whenever someone talks about computing an FT on a computer, they usually say they are "taking the FFT". They are actually taking the discrete FT using the FFT algorithm.

More about the DFT can be found in the Appendix below.

Appendix. The discrete Fourier transform

The Fourier transform is defined as an integral over all of space. How could we evaluate this on a computer? We will have to take a finite number of discrete samples both in the real (x,y) space and in the (u,v) frequency space.

Let's first do this in one dimension, and we'll model discrete samples by multiplying functions by series of delta functions. Recall the Fourier transform pair

$$\sum_{n=-\infty}^{\infty} \delta(ax-n) \xrightarrow{FT} \frac{1}{a}\sum_{m=-\infty}^{\infty} \delta\left(\frac{u}{a}-m\right) \qquad (1.1)$$

We can use this to understand the properties of the discrete Fourier transform (DFT). Start with our favorite FT pair,

$$e^{-\pi x^2} \xrightarrow{FT} e^{-\pi u^2}$$

which we'll call g_0 and G_0. But now we'll multiply g_0 by a series of delta functions (a scaled Shah function) to yield

$$g_1(x) = e^{-\pi x^2} \sum_{n=-\infty}^{\infty} \delta(ax-n) \qquad (1.2)$$

where *a* is set to 4, that is we sample *x* in units of 1/4. We know what the FT of (1.2) will be: it will be the convolution of $G(u)$ with a series of delta functions, so we'll get *copies of G* spaced every 4 units on the *u* axis. See the middle row of Figure A1. This spacing looks to be sufficient that if we were just to cut out the central copy of the Gaussian function, between $u = -2$ to $u = 2$, we'd have the original function back. This demonstrates a very important theorem, the *sampling theorem*. If the FT of some function g_0 is zero outside a limited interval of *u*, it can be reconstructed perfectly from discrete samples. The only requirement is that the sampling frequency must be at least twice the highest frequencies $\pm u$ for which $G(u)$ is nonzero.

You will understand what will happen if this criterion, called the *Nyquist criterion*, is violated. Imagine that we had sampled $g(x)$ more coarsely, say in x steps of ½. Then the periodicity of the repeats in G_1 would be only two units and there would be overlap of the copies of the Gaussians. This overlap is called *aliasing* and it reflects the loss of information—we can no longer reconstruct g_1 perfectly from G_1.

Now, what happens if we sample G_1, say in steps of ¼ in u, to yield the function G_2. If we evaluate its inverse transform g_2 we see that it also consists of copies of the original function

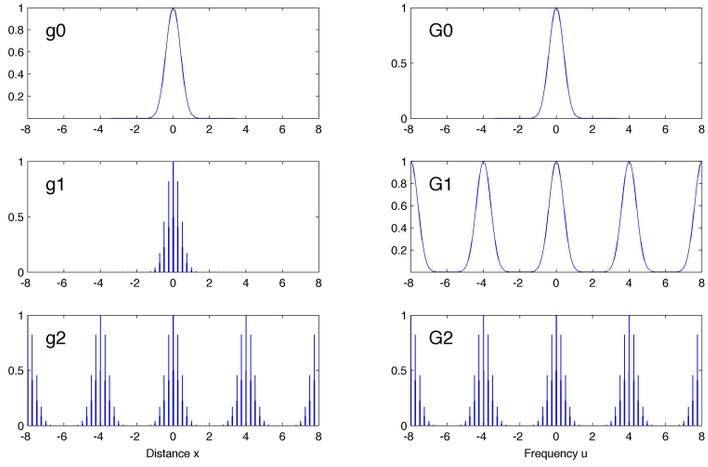


Figure A1. The continuous functions g0 and G0 are sampled, resulting in periodic copies.

(see the bottom row of Fig. A1). Now we have discrete samples in both x and u, which is exactly what we'd need to process data on a computer. Discrete samples mean that both g and G have become periodic functions, but as long as we make the periods long enough (by sampling finely enough) we can work with these discrete samples just fine, by chopping out one period to work with.

So, if we restrict both x and u to integer values, we can define the discrete Fourier transform and its inverse (DFT and IDFT, respectively) as

$$G(u) = \sum_{x=0}^{N-1} g(x) e^{-i2\pi ux/N} \qquad (1.3)$$

$$g(x) = \frac{1}{N} \sum_{u=0}^{N-1} G(u) e^{i2\pi ux/N} \qquad (1.4)$$

The values of x range from 0 to N-1, as do the values of u. This can be confusing, as demonstrated in Fig. A2. Part A of the figure shows how I'd like to visualize a Gaussian function in an array with N=16 for use with the DFT, with index values of x= -8, -7,…-2, -1, 0 , 1, 2…7. Part B shows however the conventional representation for using the DFT on a computer. The successive points correspond to my x values of 0, 1, ...6, 7, -8, -7, …-1. Remember that the DFT treats its input as if it were periodic, so it will "splice together" the end points of the interval; this means that the representations in A and B are equivalent. Note also that some programming languages have array indices starting with 1 instead of 0. (Matlab and Fortran are like this.) As I'm using Matlab you'll see that in panel B the point corresponding to the peak of the Gaussian is at position 1 on the x axis, which is actually showing array indices.

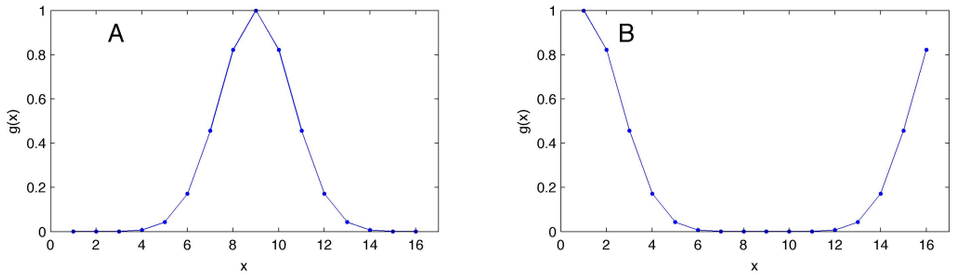


Figure A2. Two discrete representations of a Gaussian function using *N*=16 points.

Two commonly-used Matlab functions for "unwrapping" functions for use in the DFT are fftshift and ifftshift. To convert part B of the figure to part A, one uses fftshift; to convert A to B you use ifftshift. If the number of points *N* is even, fftshift and ifftshift do exactly the same thing. So if you have an array *g* that is indexed like A, you can compute the DFT and get a centered Gaussian back by typing

```
G=fftshift(fft(ifftshift(g)));
```

In Matlab you have the functions fft and ifft (one-dimensional forward and inverse transforms) and fftn and ifftn (n-dimensional transforms).

The FFT algorithm happens to work best when the number of points *N* is a power of 2 or a product of small factors. That is one reason why people often use array sizes like 1024 instead of, say, 1020.

Windowing

Figure A3 demonstrates another problem that arises from the finite sum in the DFT, that we never had with the FT because the latter uses an infinite integral. Function g_1 is a cosine function that has 8 cycles in the *N*=80 points, and its DFT looks a lot like the two delta functions we expect from the FT of a cosine function. (Right?) The cosine function g_2 has a slightly higher frequency, so 8.5 cycles fit in the 80 points. Its DFT looks horrible. Why all the ripples if we're taking the transform of a nice smooth cosine function? Remember that the DFT thinks its input is a periodic function. If we shift the display of g_2 by half of the window (bottom row in the figure) you can see that g_2 actually contains a discontinuity which is what causes the trouble.

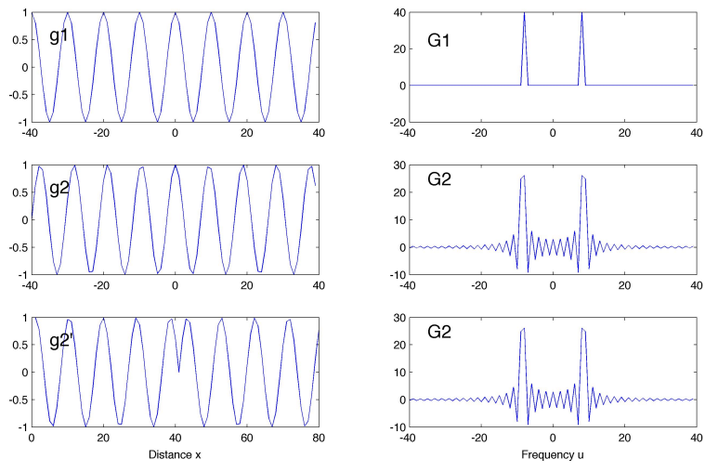


Figure A3. DFTs of a cosine function. *N*=80.

One can get rid of the problems with a discontinuity by using a windowing function $w(x)$ that tapers gently toward zero at the edges. Here in Fig. A4 are the same functions except that g_1' and g_2' are the products of g_1 and g_2 with a smooth window function. It gives better looking transforms.

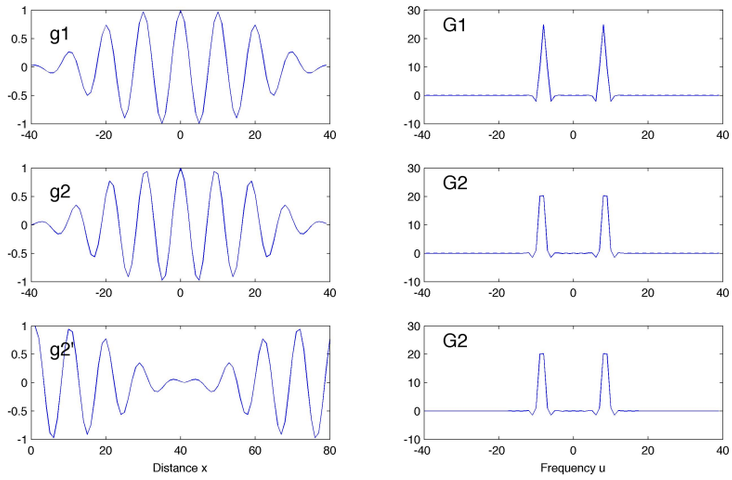


Figure A4. A window function has been applied to the cosine.

Note that in the upper two panels of this figure I used Matlab's `fftshift` function to move the center of the waveforms to the center of each plot; but I changed the labeling of the x-axis to run from -40 to +39, rather than from 1 to 80.

Scaling the DFT

Once we start using integer values of x and u for the DFT, we have to keep in mind the relationship between the integer values and real-world quantities. For example, let x index the pixels of a micrograph. Then one unit in x corresponds to a pixel size dx, e.g. 1.7 Å. When we take the DFT to get the result in terms of the spatial frequency u, one unit in u corresponds to a frequency of $du = 1/(Ndx)$, that is $1/$(size of the micrograph). So for example if N=1024 then $du = 5.7 \times 10^{-4}$ Å^{-1}. We can choose the integer u values to range from $-N/2$ to $N/2$-1 and therefore correspond to real-world spatial frequencies of about -0.3 to 0.3 inverse angstroms, that is $\pm 1/(2dx)$. The maximum accessible frequency, $1/(2dx)$ is called the Nyquist frequency.