Fourier Transforms, Convolution Theorem with applications in Digital Image Processing

Kenneth Navarro

Note:
Before beginning, if you see any content/illustrations that you rightfully own and would like me to take down, please email me at kennavarro2022@gmail.com.
Additionally, I wrote these notes in hopes that it could help my friends or others that feel like a lot of the material they see with respect to fourier transforms, convolutions, and other important signal processing concepts tend to be very terse and not well expressed by authors. This document strives to be truly self contained. I realized that I should do something like this only after noticing that

authors tend to leave a lot of important foundational information out when writing about topics that arent so intuitive at first glance (which I feel like tends to be less true in more theoretical writings rather than a more applied topic like this). Also, I already write notes like im talking to a friend about it so I dont see why I shouldnt post this. Hopefully these next ninety pages give a clear/concise/intuitive introduction to fourier transforms, convolutions, aliasing, and much more!

Lastly, there may be some mistakes, and if you're a friend who is reading this, please be sure to shoot me a message/email about this if you notice any.

Also please cut me some slack for the shitty formatting (it gets better over time).

Motivation:

Fourier transforms help us determine which type of frequencies are most present in an image. Also lets us compute (a very important operation in so many areas especially computer science) convolutions rapidly (given you know how to do fast fourier transforms), but these notes will not go over fast fourier transforms, rather it will go over the basic fourier transform and a lot of its properties and related math concepts. After reading this I'd recommend a quick read on fast fourier transforms and looking at more fun examples on digital image processing.

Preliminary Concept

A complex number can be described as

$$C = R + jI$$

Its conjugate

$$C^* = R - jI$$

Its polar representation

$$C = |C|(\cos\theta + j\sin\theta)$$

Its polar representation using eulers identity

$$C = |C|e^{j\theta}$$

Where |C| is the modulus of C.
A complex function extends the notion of a complex number in the most obvious way possible.

> ceding equations are applicable also to complex functions. For example, a complex function, $F(u)$, of a variable u, can be expressed as the sum $F(u) = R(u) + jI(u)$, where $R(u)$ and $I(u)$ are the real and imaginary component functions. As previously noted, the complex conjugate is $F^*(u) = R(u) - jI(u)$, the magnitude is $|F(u)| = \sqrt{R(u)^2 + I(u)^2}$, and the angle is $\theta(u) = \arctan[I(u)/R(u)]$. We return to complex functions several times in the course of this and the next chapter.

Now we know the relationship between sines and cosines because of eulers identity, so it makes sense that given a function f(t) of a continuous variable that has period T (hence periodic) can be given by…

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{T}t}$$

We are able to solve for the coefficients via this formula

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\, e^{-j\frac{2\pi n}{T}t}\, dt \qquad \text{for } n = 0, \pm 1, \pm 2, \ldots$$

A unit impulse of a continuous variable t is

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases} \qquad (4.2\text{-}8a)$$

and is constrained also to satisfy the identity

$$\int_{-\infty}^{\infty} \delta(t)\, dt = 1 \qquad (4.2\text{-}8b)$$

Interpreting t as time an impulse is literally an infinitely long spike at origin that has zero duration. Using the fact that there is only one point in which delta(t) is not zero and that this point is when t is zero, then it makes sense that f only has

magnitude AT t equals to 0. This magnitude is just going to be the spike f(0) and not some constant times f(0) since the integral of the impulse function is 1. If the impulse is located at t_0 then

$$\int_{-\infty}^{\infty} f(t)\,\delta(t - t_0)\,dt = f(t_0)$$

If we let x be a discrete variable then the impulse function of a discrete system is defined as

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases}$$

Clearly, this definition also satisfies the discrete equivalent of Eq. (4.2-8b):

$$\sum_{x=-\infty}^{\infty} \delta(x) = 1 \qquad (4.2\text{-}11b)$$

The sifting property for discrete variables has the form

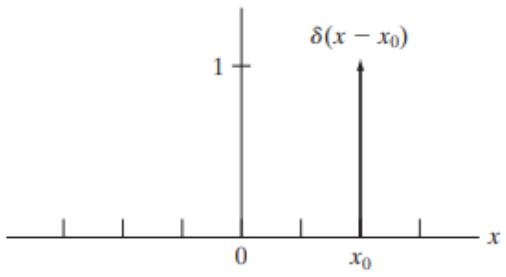
$$\sum_{x=-\infty}^{\infty} f(x)\,\delta(x) = f(0) \qquad (4.2\text{-}12)$$

or, more generally using a discrete impulse located at $x = x_0$,

$$\sum_{x=-\infty}^{\infty} f(x)\,\delta(x - x_0) = f(x_0) \qquad (4.2\text{-}13)$$

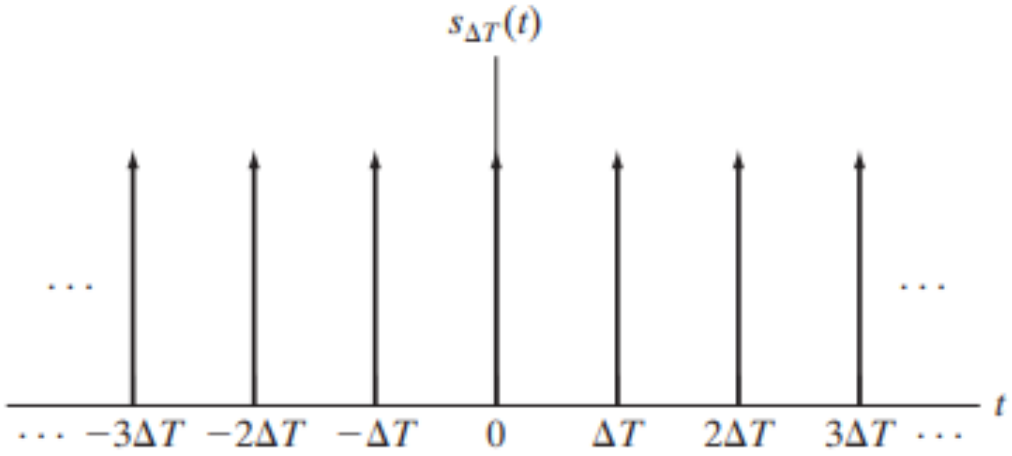
Delta function always 1 when its input is 0 which implies impulse at x_0 if the offset is x_0.

FIGURE 4.2
A unit discrete impulse located at $x = x_0$. Variable x is discrete, and δ is 0 everywhere except at $x = x_0$.



Of particular interest later in this section is an *impulse train*, $s_{\Delta T}(t)$, defined as the sum of infinitely many *periodic* impulses ΔT units apart:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T) \qquad (4.2\text{-}14)$$


$s_{\Delta T}(t)$
$\cdots$
$\cdots$
t
$\cdots$ $-3\Delta T$ $-2\Delta T$ $-\Delta T$ 0 ΔT $2\Delta T$ $3\Delta T$ $\cdots$

The fourier transform of functions of one continuous variable.

$$\Im\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t}\,dt$$

It is important to note that the above function results in a function of u since u is the only variable and t gets integrated out. Therefore we can explicitly denote this via

$$F(\mu) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t}\,dt$$

Of course we can then solve for f(t)

$f(t) = \Im^{-1}\{F(\mu)\}$, written as

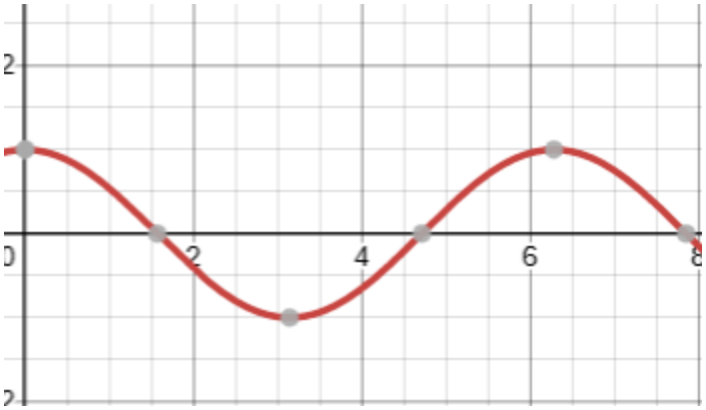
$$f(t) = \int_{-\infty}^{\infty} F(\mu)e^{j2\pi\mu t}\,d\mu$$

$$F(\mu) = \int_{-\infty}^{\infty} f(t)\left[\cos(2\pi\mu t) - j\sin(2\pi\mu t)\right]dt$$

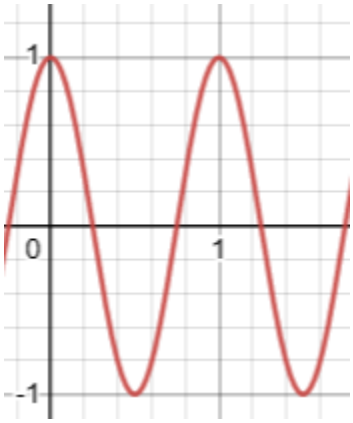
Fourier transform is an expansion of f(t) multiplied by sinusoidal terms whose frequencies r determined simply by u since t gets integrated out. Since u denotes the frequency, we say that the fourier transform is the frequency domain.
If t represents time in seconds, then u is cycles per second (Hertz). t is meters then u is meters/second.

Period is the distance between to identical and consecutive portions of the curve, the reciprocal of the period is the frequency.

If we look at desmos



The grey circle slightly to the right and above 6 is denoted by 2pi.
So this is the period of the function. If I multiply x by 2pi then plot this I get

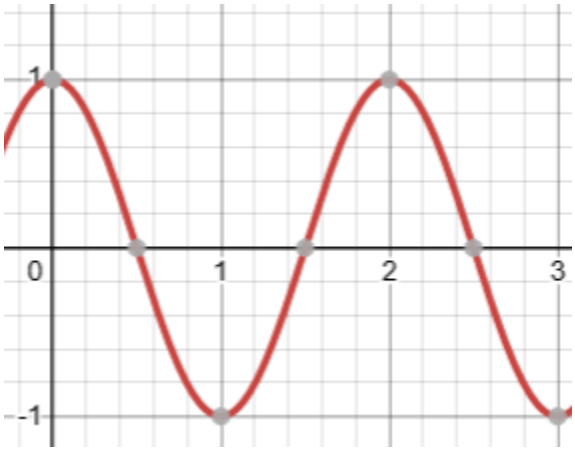


So the period here is simply 1 (since x equal to 1 implies argument is 2pi).
Therefore it makes sense that u has units cycles/second as essentially

$$F(\mu) = \int_{-\infty}^{\infty} f(t)\left[\cos(2\pi\mu t) - j\sin(2\pi\mu t)\right] dt$$

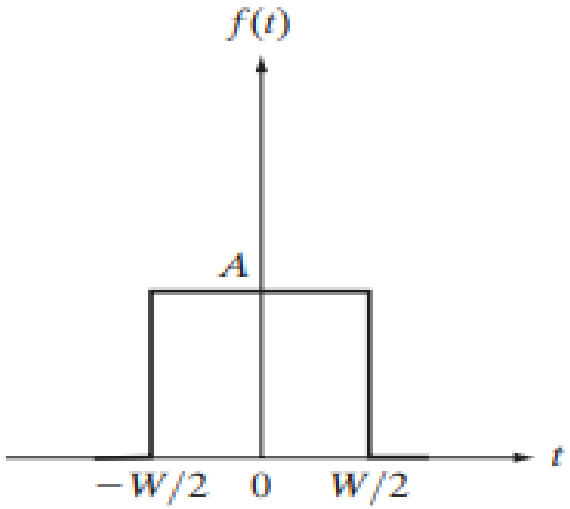
Since 2pit is graphed above we see that its period is 1 and therefore so is its frequency.
We can think of 2pi as accelerating t so that in one second it completes a cycle. Therefore since it takes one second to complete a cycle, and we can increase the speed at which we complete a cycle (u) then u can be seen as the amount of cycles PER second(since we sped up by 2pi). Cosine and sine have phases equal to 2pi naturally, so all we can really do is edit the argument in order to make the time in which the product within the argument hits 2pi quicker, all we are doing is making the product within hit 2pi quicker and we can characterize this as cycles/second according to u when other part of the argument in its product is equal to seconds (if we didnt have this other part, we would not be

able to think about u as cycles per second). Take for example an acceleration by pi, then we notice that t travels a cycle in two seconds, so u would be one cycle cycle per two seconds.



After this explanation it should be clear why this is known as the frequency domain post transformation(due to the 2pi acceleration on t which we then manipulate with u).

Now lets perform the fourier transform on this function



$$F(\mu) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t}\,dt = \int_{-W/2}^{W/2} Ae^{-j2\pi\mu t}\,dt$$

$$= \frac{-A}{j2\pi\mu}\left[e^{-j2\pi\mu t}\right]_{-W/2}^{W/2} = \frac{-A}{j2\pi\mu}\left[e^{-j\pi\mu W} - e^{j\pi\mu W}\right]$$

$$= \frac{A}{j2\pi\mu}\left[e^{j\pi\mu W} - e^{-j\pi\mu W}\right]$$

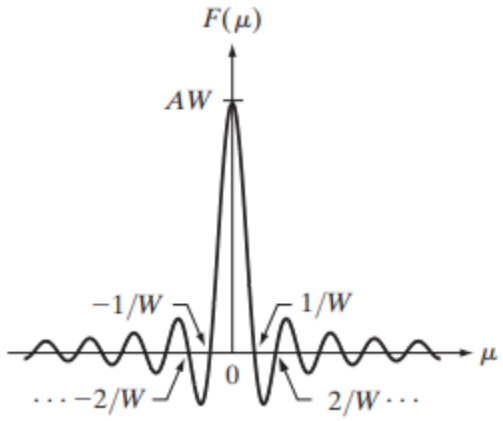
$$= AW\frac{\sin(\pi\mu W)}{(\pi\mu W)}$$

$\sin\theta = (e^{j\theta} - e^{-j\theta})/2j.$

function. The result in the last step of the preceding expression is known as the *sinc* function:

$$\text{sinc}(m) = \frac{\sin(\pi m)}{(\pi m)} \tag{4.2-19}$$

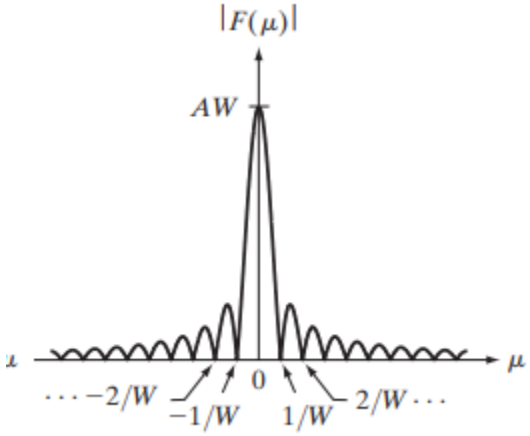
where $\text{sinc}(0) = 1$, and $\text{sinc}(m) = 0$ for all other *integer* values of m. Figure 4.4(b) shows a plot of $F(\mu)$.



Seems like the local maximas are evenly space apart on the frequency domain. Also note that the widths of consecutive zeros in the fourier graph for both the frequency domain and the fourier spectrum are inversely proportional to the width of the box function(above and below)

The fourier transform contains complex terms, therefore customary for displaying stuff to work on the magnitude of the transform (fully real). This is known as the **Fourier Spectrum/Frequency Spectrum**

$$|F(\mu)| = AT\left|\frac{\sin(\pi\mu W)}{(\pi\mu W)}\right|$$



The lobes clearly decrease as a function of distance from origin and the function extends to both +-inf.

■ The Fourier transform of a unit impulse located at the origin follows from Eq. (4.2-16):

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t)\, e^{-j2\pi\mu t}\, dt \\ &= \int_{-\infty}^{\infty} e^{-j2\pi\mu t}\, \delta(t)\, dt \\ &= e^{-j2\pi\mu 0} = e^{0} \\ &= 1 \end{aligned}$$

domain is a constant in the frequency domain. Similarly, the Fourier transform of an impulse located at $t = t_0$ is

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t - t_0) e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{\infty} e^{-j2\pi\mu t} \delta(t - t_0)\, dt \\ &= e^{-j2\pi\mu t_0} \\ &= \cos(2\pi\mu t_0) - j\sin(2\pi\mu t_0) \end{aligned}$$

Deriving the fourier transform of a periodic impulse train

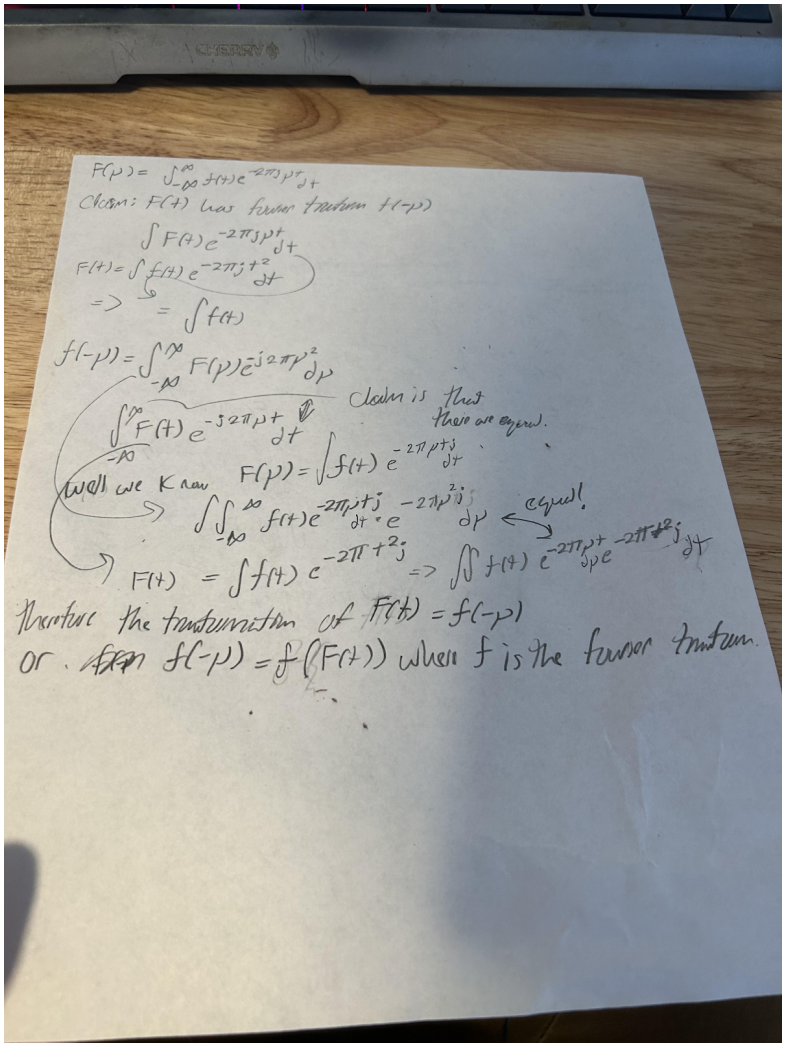
$$F(\mu) = \int_{-\infty}^{\infty} f(t) e^{-j2\pi\mu t}\, dt$$

And we know

$f(t) = \Im^{-1}\{F(\mu)\}$, written as

$$f(t) = \int_{-\infty}^{\infty} F(\mu) e^{j2\pi\mu t}\, d\mu$$

It turns out that since f(t) has a fourier transform of F(u), then f(-u) is the fourier transform of F(t).



Let o(t) denote the impulse function.

Noting the symmetry property above and the fact that the fourier transform of e^(-j2πut0) is o(t-t0), we know the transformation of e^(-j2πt0t) is o(-u - t0). If we let -t0 = a, then the transform of e^(j2πat) is o(-u+a), and we note that o(-u+a) is equivalent to o(u-a), this is true because only u is fluctuating therefore not zero when u equal to a.

The impulse train

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$

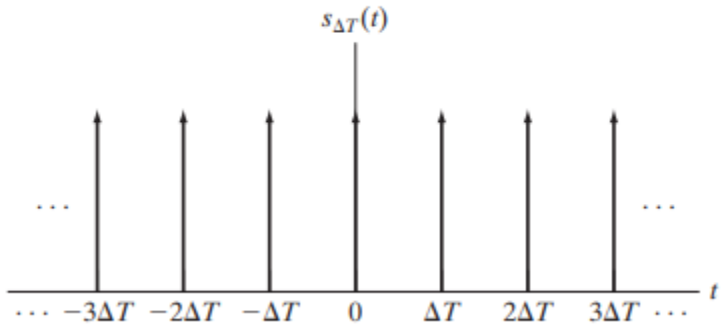
is periodic with period ΔT. So we know it can be expressed as a fourier series.

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{\Delta T}t}$$

$$c_n = \frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2} s_{\Delta T}(t) e^{-j\frac{2\pi n}{\Delta T}t} dt$$

Note that the above 'cn' integral will only capture the part of the impulse train centered at the origin since the integral extends from -deltaT/2 -> +deltaT/2

4.2 ■ P



So

$$c_n = \frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2} \delta(t) e^{-j\frac{2\pi n}{\Delta T}t} dt$$

$$= \frac{1}{\Delta T} e^{0}$$

$$= \frac{1}{\Delta T}$$

Implying that

$$s_{\Delta T}(t) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}$$

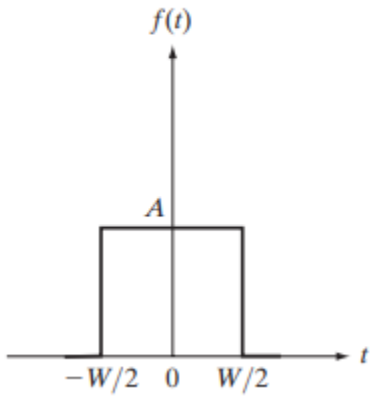
We also know that the fourier transform of

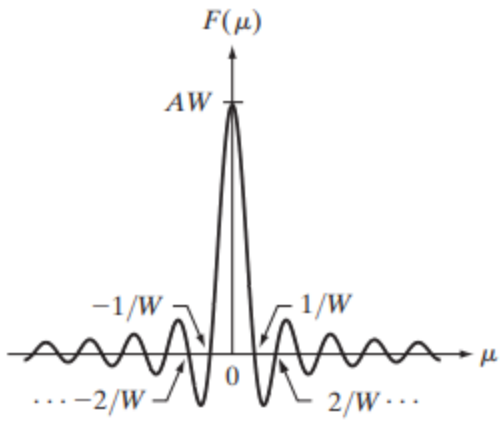
$$\Im\left\{e^{j\frac{2\pi n}{\Delta T}t}\right\} = \delta\left(\mu - \frac{n}{\Delta T}\right)$$

(Let a equal n/deltaT)
So the fourier transform of the impulse train is

$$\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\delta\left(\mu - \frac{n}{\Delta T}\right)$$

It is pretty important to note that this is because the fourier transform is linear.
The most unintuitive part for me was the calculation of the c_n term. Need to probably brush up on that.
Now it is important to note that this result above can be interpreted as stating that the fourier transform of an impulse train with period a is also an impulse train of period 1/a. Note that this inverse proportionality between the impulse train and its fourier transform is actually analogues to a transform weve already seen.





Convolution

We know the continuous convolution operation can be defined as

$$f(t) \star h(t) = \int_{-\infty}^{\infty} f(\tau)\, h(t-\tau)\, d\tau$$

This book describes the notion of a convolution by ‘flipping one function about its origin then sliding past the other and at each displacement in the sliding process performing a computation (for us it was multiplication the summation)’

Now finding the fourier transform of the above

$$\Im\{f(t) \star h(t)\} = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\tau)\, h(t-\tau)\, d\tau \right] e^{-j2\pi\mu t}\, dt$$

$$= \int_{-\infty}^{\infty} f(\tau) \left[\int_{-\infty}^{\infty} h(t-\tau)\, e^{-j2\pi\mu t}\, dt \right] d\tau$$

Then it turns out

The term inside the brackets is the Fourier transform of $h(t - \tau)$. We show later in this chapter that $\Im\{h(t - \tau)\} = H(\mu)e^{-j2\pi\mu\tau}$, where $H(\mu)$ is the Fourier transform of $h(t)$. Using this fact in the preceding equation gives us

$$\Im\{f(t) \star h(t)\} = \int_{-\infty}^{\infty} f(\tau)\left[H(\mu)e^{-j2\pi\mu\tau}\right]d\tau$$

$$= H(\mu)\int_{-\infty}^{\infty} f(\tau)e^{-j2\pi\mu\tau}\,d\tau$$

$$= H(\mu)F(\mu)$$

It turns out that the fourier transform on the convolution operation between two distinct functions (say g and b) result in the product of their individual fourier transforms F(g)*F(b).

the inverse Fourier transform. In other words, $f(t) \star h(t)$ and $H(u)F(u)$ are a Fourier transform pair. This result is one-half of the *convolution theorem* and is written as

$$f(t) \star h(t) \Leftrightarrow H(\mu)F(\mu) \qquad (4.2\text{-}21)$$

The fourier transform of convolution of two functions in the spatial domain is equivalent to their product in the frequency domain.
So the inverse fourier transform of a product of two functions in the frequency domain is equivalent to the convolution of the two functions in the spatial domain.
It also turns out the fourier transform of the product of two different functions is equal to the convolution between their respective fourier transforms.

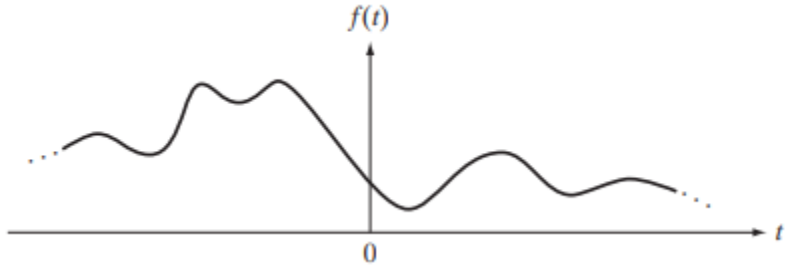
$$f(t)h(t) \Leftrightarrow H(\mu) \star F(\mu)$$

Inversely the inverse fourier transform on the convolution of two individual fourier transforms results in the two original functions (pre fourier transform) being multiplied by eachother.
‘Multiplication in spatial domain is analogous to convolution in frequency domain’.
It turns out that the two equations we see above are what form the convolution theorem. This plays a fundamental role for filtering in the frequency domain.

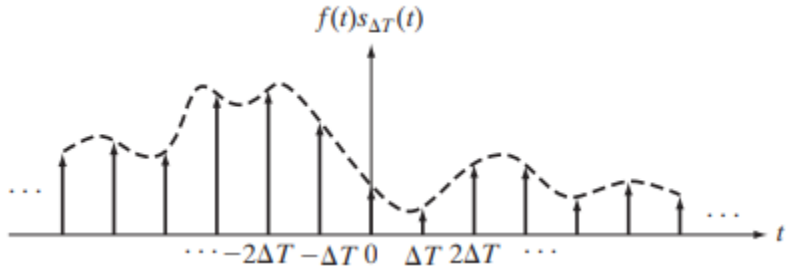
Sampling and Fourier Transform of Sampled Functions

Sampling:
Continuous functions must be broken down into a sequence of discrete values before being able to be processed ina computer. Accomplished via sampling and quantization. We now view sampling in more detail.

Lets consider a function f(t).



Suppose we want to sample at uniform intervals ΔT of the independent variable t. Assuming the function extends from negative to positive infinity, one way to sample is by multiplying f(t) with a sampling function equal to a train of impulses ΔT units apart. Intuitively from our understanding of impulse trains it makes sense the result is the visual below



This function is equivalent to

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\,\delta(t - n\Delta T)$$

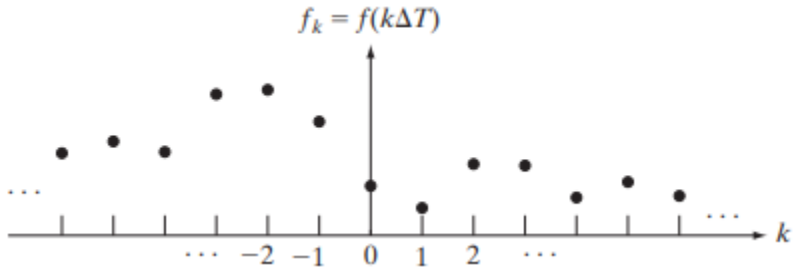
Now to get the ‘strength’ of the weighted impulse (with respect to a specific impulse in the train) Is given by

$$f_k = \int_{-\infty}^{\infty} f(t)\,\delta(t - k\Delta T)\,dt$$

$$= f(k\Delta T)$$

<<<< Sifting Property!

Notice here that by ‘respect to a specific impulse’ i mean with respect to one of the arrows above and how the entire function f(t) reacts only at that one specific point. We can measure that with the above formulation and it results in a graph that we see below (assuming we do it for all k)



Is going to be the sum of all f_k for all k. But there is not really a point to finding that out.

The fourier transform of sampled functions

If we let F(u) denote the fourier transform of a cont. function f(t). As we know the corresponding function

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T)$$

is the product of f(t) and an impulse train.

We know from the convolution theorem that the product of two functions in the spatial domain is equal to the convolution of two products in the frequency domain. Thus the fourier transform of the ‘discrete sampling function’ is given by

$$\begin{aligned}\tilde{F}(\mu) &= \Im\{\tilde{f}(t)\} \\ &= \Im\{f(t)s_{\Delta T}(t)\} \\ &= F(\mu) \star S(\mu)\end{aligned}$$

As previously derived we noted that the fourier transform of the impulse train is

$$S(\mu) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\delta\left(\mu - \frac{n}{\Delta T}\right)$$

Now performing convolution on this, we get

$$
\begin{aligned}
\tilde{F}(\mu) &= F(\mu) \star S(\mu) \\
&= \int_{-\infty}^{\infty} F(\tau)\, S(\mu - \tau)\, d\tau \\
&= \frac{1}{\Delta T} \int_{-\infty}^{\infty} F(\tau) \sum_{n=-\infty}^{\infty} \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\
&= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} F(\tau)\, \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\
&= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F\left(\mu - \frac{n}{\Delta T}\right)
\end{aligned}
$$

This shows that the fourier transform of the discrete sampling function is an infinite periodic sequence of copies of F(u). The separation between copies is defined by 1/ΔT.
Although we did the transform on a discrete sampling function, its transform is continuous because it consists of copies of F(u) which is a continuous function (The sum of continuous functions result in a continuous function by analysis arguments).

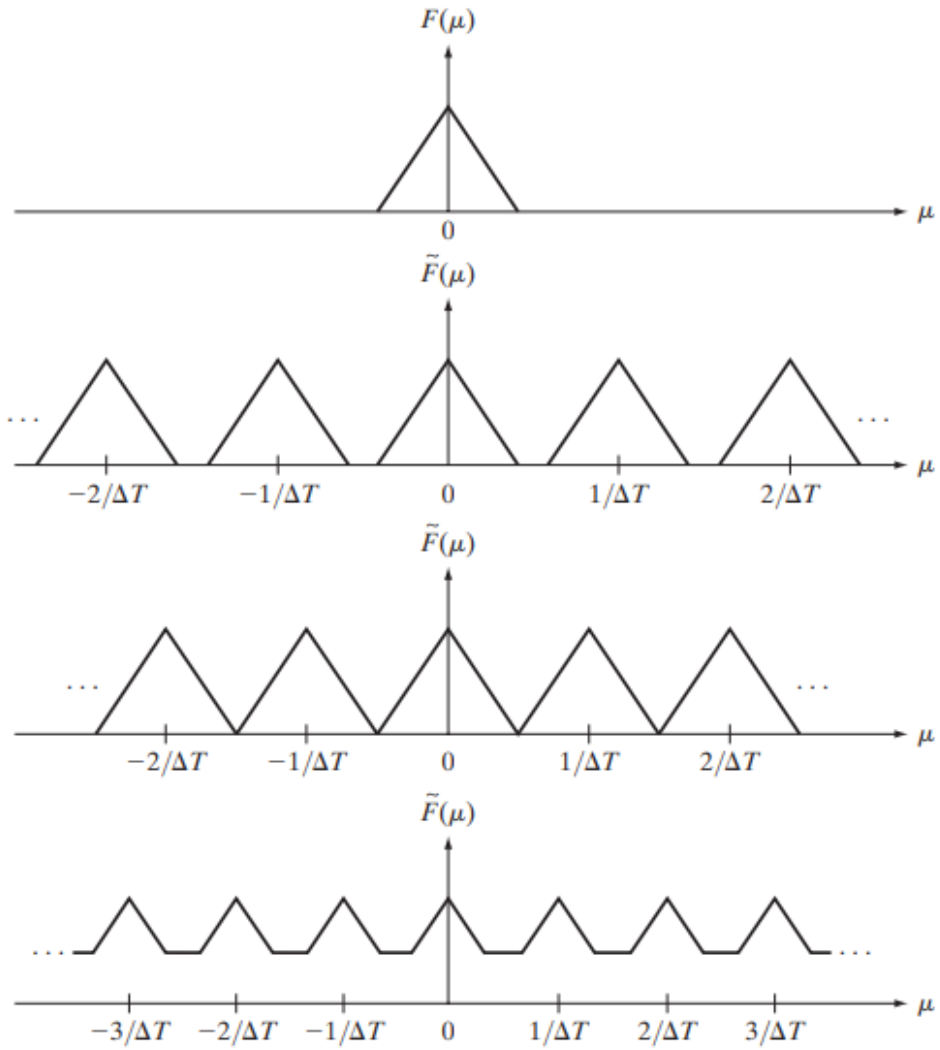


Figure a is the regular fourier transform of the function, while the other graphs are the fourier transforms of the function times the discrete sampling function. The second graph is known as over sampling, the third graph is known as critically sampling (just barely enough), and the third graph is known as undersampling. Maybe the intuition behind it is that we are not giving enough space for the function to display its true characteristics when we pick such a small gap in between each consecutive sample. Meanwhile in the second graph we made the gap too big and thus lost information due to the constant gap nature of the transform.
So all we have went over so far is essentially taking the fourier transform of one type of sampling function (the continuous function times the impulse train).

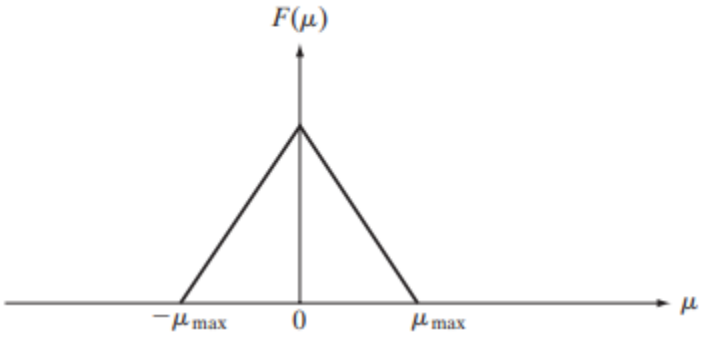
The Sampling Theorem

We will now look at conditions under which a continuous function can be **recovered uniquely** from a given set of samples.

A function f(t) whose fourier transform is zero for frequencies outside a finite interval

$[-\mu_{max}, \mu_{max}]$

about the origin is known as a 'band limited' function (we are calling f(t) band limited not its transform)
For example this function below is a band limited function.



As said previously, a lower value of 1/ΔT will cause a smaller gap (merging) between the triangles/periods meanwhile a larger value of 1/ΔT will cause a larger gap (seperation) between the triangles/periods. We are able to recover a function f(t) from its sampled version if we can isolate a copy of F(u) from the periodic sequence of copies of the function contained in F~(u) (the transform of the discretely sampled function f~). Note that since…

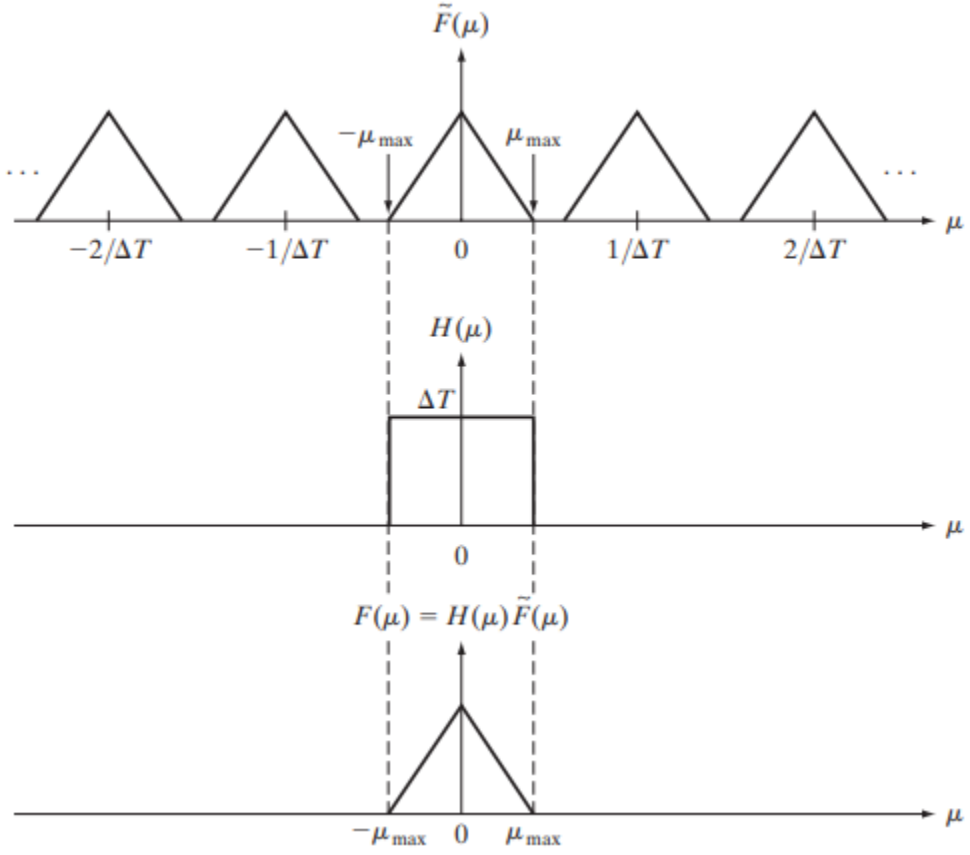
previous section that $\tilde{F}(\mu)$ is a *continuous*, *periodic* function with period $1/\Delta T$. Therefore, all we need is one complete period to characterize the entire transform. This implies that we can recover $f(t)$ from that single period by using the inverse Fourier transform.

So, it is possible to extract a representation of F(u) assuming that the 1/ΔT in F~(u)'s input is of the proper size. Well if we want to make sure the gap is big enough such that we dont experience an under sampling effect, we can realize that if u_max is c, then for us to cover the entire triangle given by a singal period, we just need our difference to be larger than 2*c in order to capture the left side of u_max (-u_max centered around origin) and well also be able to capture u_max in this case (over sampling seems to be okay for recovering the unique representation of the function associated with the transform).
'No information is lost if a continuous band limited function is represented by samples acquired at a rate greater than twice the highest frequency content of the function'

function. Conversely, we can say that the *maximum* frequency that can be "captured" by sampling a signal at a rate $1/\Delta T$ is $\mu_{max} = 1/2\Delta T$. Sampling at the Nyquist rate sometimes is sufficient for perfect function recovery, but there are cases in which this leads to difficulties, as we illustrate later in Example 4.3. Thus, the sampling theorem specifies that sampling must exceed the Nyquist rate.

The Nyquist rate is when we set 1/ΔT = 2*u_max = 2*c.

Since f(t) and its associated discrete sampler s(a) = o(u - n/a) are different functions, I suppose all we have to do is examine f(t) itself, and then make sure 1/a > 2*/height of box (if a box function).
There are multiple ways to sample a function, another example is to use the H function shown in this diagram which is essentially a box as wide as 2*u_max centered at the origin with ΔT height and notice how the reconstruction can in principle occur. Note that we have a function below in the first diagram which is sampled at higher than the nyquist rate, and therefore when we choose a sampler that is of enough height and distance, we can recover the original function. Honestly ΔT seems to be 1 here (for the H function) because it doesnt not seem to be changing the height of F~(u) when retrieving F(u).



Once we obtain F(u), we can obtain f(t) via the inverse fourier transform

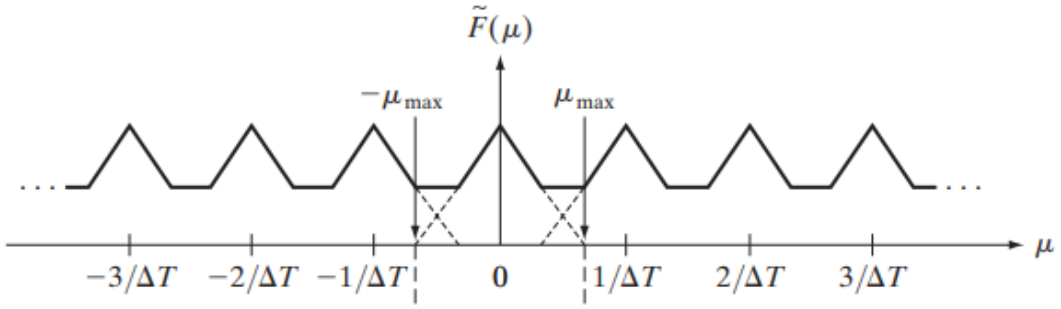
$$f(t) = \int_{-\infty}^{\infty} F(\mu) e^{j2\pi\mu t} d\mu$$

Theoretically its possible to recover a band limited function from samples of the function obtained at a rate exceeding twice the highest frequency content of the function. The requirement that f(t) is bandwidth limited implies that f(t) extends from -inf -> +inf (due to the integral)

The function H in the above case is known as a lowpass filter because it passes frequencies at the low end of the frequency range (centered about origin and frequency in absolute value up to and equal to u_max) and filters everything else out (eliminates all higher frequencies). Notice that this is the frequency domain. However this is known as an ideal lowpass filter due to infinitesmally small transition time from a height of ΔT to a height of 0 (at position -u_max and +u_max), which can not be achieved via current physical electronic components.
Filters are critical in reconstruction/recovering the original function from its samples, filters in this realm specifically are known as reconstruction filters.

Aliasing

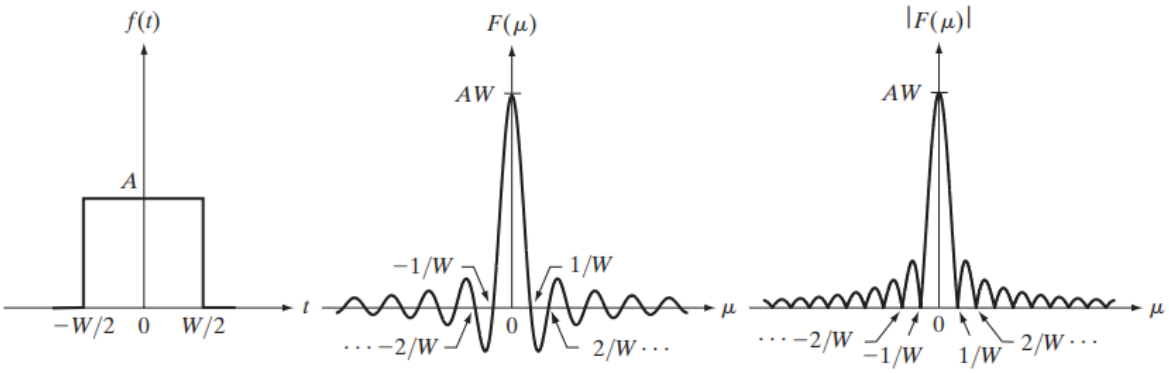
What happens if a band limited function is under sampled? The effect of sampling below the nyquist rate is that periods overlap therefore becoming impossible to isolate a single period of the transform (regardless on what kind of filter is used!). If we use the ideal low pass filter as before and apply this to the frequency domain, we will get adjacent signals intertwining with eachother and have corrupted information therefore resulting in an inaccurate transform. This effect is known as frequency aliasing/aliasing.
Aliasing is a process where high frequency components of a continuous function masquerade as lower frequencies in the samples function (the higher frequency portion of the signal will be the part of the signal which is gone and therefore lost in terms of information and will be masked as the state before it, specifically look at the straight lines) for example.



Aliasing turns out to always be present in practice, since even with band-limited functions, infinite frequency components arise when we limit duration of function. For example if we want to limit duration of f(t) (assumed bandwidth limited) to [0,T] we multiply f(t) by

$$h(t) = \begin{cases} 1 & 0 \leq t \leq T \\ 0 & \text{otherwise} \end{cases}$$

h(t) has same shape as bottom left figure below, so H(u) (its transform) will have frequency domain with the same shape as the second graph.



The second graph has components extending to positive and negative infinity. The fourier transform of the product of h and f where h is the sample function defined as above is the convolution of the transforms of the individual functions (FConv(H)). So even if the transform of f (F) is band limited, convolving it with H (involving sliding one function across the other) will result with frequency components extending to infinity since H goes infinitely far away in both directions. Therefore no function of finite duration can be band limited.
No function of finite duration can be band limited.
A function that is band limited must extend from -inf to +inf.

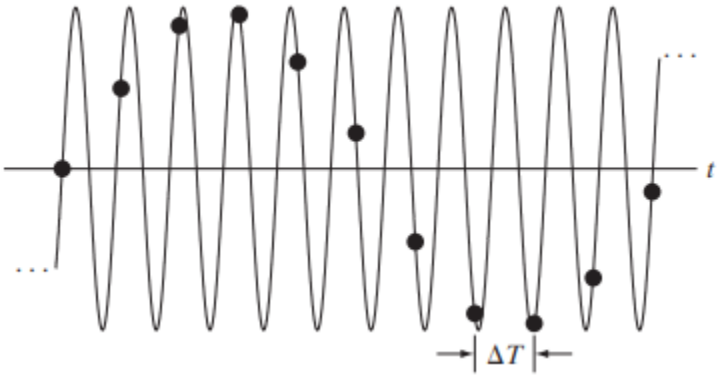
Aliasing is present in sampled signals because when even if the function is band limited, infinite frequency components will arise when we take the fourier transform of the sampling function. The first figure above is what we call the sampling function and the second figure above is its fourier transform which extends to infinity. So even if f (assume the sampling function is h) is band limited (has a finite transform) it will be convolution with something that has an infinitely long transform everytime we perform sampling. This implies that some under sampling will occur because technically we wont be able to ever capture the entirety of the sampling function H (regardless of how big we choose 1/ΔT to be)!
Now that we know aliasing is an inevitable fact of working with sampled records of finite length, we can reduce the effects of aliasing via smoothing the input function to attenuate the higher frequencies (moves majority of information closer to origin in frequency domain by doing so). In the case of an image maybe by defocusing. This process is called anti aliasing (done before sampling).

This is a great example below to understand an applied version of aliasing.

■ Figure 4.10 shows a classic illustration of aliasing. A pure sine wave extending infinitely in both directions has a single frequency so, obviously, it is band-limited. Suppose that the sine wave in the figure (ignore the large dots for now) has the equation $\sin(\pi t)$, and that the horizontal axis corresponds to time, t, in seconds. The function crosses the axis at $t = \ldots -1, 0, 1, 2, 3 \ldots$.

The period, P, of $\sin(\pi t)$ is 2 s, and its frequency is $1/P$, or $1/2$ cycles/s. According to the sampling theorem, we can recover this signal from a set of its samples if the sampling rate, $1/\Delta T$, exceeds twice the highest frequency of the signal. This means that a sampling rate greater than 1 sample/s $[2 \times (1/2) = 1]$, or $\Delta T < 1$ s, is required to recover the signal. Observe that sampling this signal at *exactly* twice the frequency (1 sample/s), with samples taken at $t = \ldots -1, 0, 1, 2, 3 \ldots$, results in $\ldots \sin(-\pi), \sin(0), \sin(\pi), \sin(2\pi), \ldots$, which are all 0. This illustrates the reason why the sampling theorem requires a sampling rate that exceeds twice the highest frequency, as mentioned earlier.

The large dots in Fig. 4.10 are samples taken uniformly at a rate of less than 1 sample/s (in fact, the separation between samples exceeds 2 s, which gives a sampling rate lower than 1/2 samples/s). The sampled signal *looks* like a sine wave, but its frequency is about *one-tenth* the frequency of the original. This sampled signal, having a frequency well below anything present in the original continuous function is an example of aliasing. Given just the samples in Fig. 4.10, the seriousness of aliasing in a case such as this is that we would have no way of knowing that these samples are not a true representation of the original function. As you will see in later in this chapter, aliasing in images can produce similarly misleading results. ■

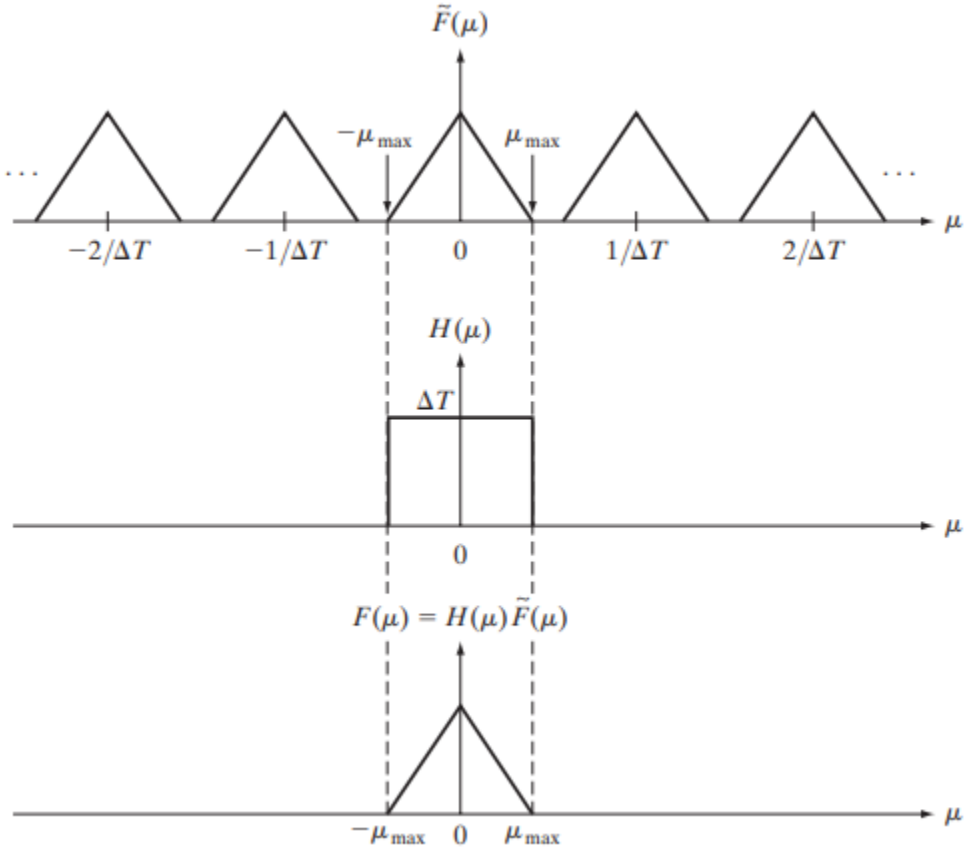


To begin we have a sign wave which we know is band limited since it has a finite period. Assume that the period is 2 seconds. Therefore the frequency is 1 cycle per 2 seconds or rather ½ cycles per second (one cycle is equivalent to the distance travelled period). Now we know that for us to capture the full cycle of one instance, we need to have 1/ΔT be greater than twice the highest frequency which is 2 * (½ cycle per second), this implies that 1/ΔT > 1 or rather we need to sample quicker than once per second or rather that we need to do over one sample per

second. In the above illustration, the black dots show an example of aliasing were we do not follow the above rule for full information capture and actually sample every two seconds, which is a sampling rate lower than ½ samples/s. The sample signal looks like a sign wave but its frequency (inverse of its period is about 1/10th the original, which is equivalent to around 10 * the original period), we are able to visually see this above as if we count how long it takes for one point of the newly formed sine wave (from first black dot to last block dot) to reach its original position on the newly formed consecutive curve). One of the problems with the above is that even when we do aliasing it still “looks” like a sine function, but it is subtly very inaccurate due to its massive difference inf frequency.

Function Reconstruction from Sampled Data

In practice, the reconstruction of a function with a set of sampled reduces to interpolating between the samples (linear or non linear interpolation as previously discussed). Even the simple act of image display requires reconstruction of image from its samples via the display medium. The following graph and its associated discussion in the book (regarding the fact that the sampling rate needs to be greater than twice the highest frequency) is a theoretically perfect example of function reconstruction on a band limited function using samples in the frequency domain.



$$f(t) = \Im^{-1}\{F(\mu)\}$$
$$= \Im^{-1}\{H(\mu)\tilde{F}(\mu)\}$$
$$= h(t) \star \tilde{f}(t)$$

f(t) convolution between the sampling function h and the sampled function f~.

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T)$$

$$f(t) \star h(t) = \int_{-\infty}^{\infty} f(\tau)\, h(t - \tau)\, d\tau$$

Results in

$$f(t) = \sum_{n=-\infty}^{\infty} f(n\Delta T)\, \mathrm{sinc}\left[(t - n\Delta T)/n\Delta T\right]$$

The ‘spatial domain’ representation for f(t). It is not surprising that the inverse fourier transform of H (SIMPLY A BOX FUNCTION HERE) results in a sinc function as the fourier transform of the box function results in a sinc, and the only difference between the two transforms is a sign change.
The function has the important property that the reconstructed function is identical to the sample values at multiple integer increments of ΔT.

to the sample values at multiple integer increments of ΔT. That is, for any $t = k\,\Delta T$, where k is an integer, $f(t)$ is equal to the kth sample $f(k\Delta T)$. This follows from Eq. (4.3-12) because $\mathrm{sinc}(0) = 1$ and $\mathrm{sinc}(m) = 0$ for any other integer value of m. Between sample points, values of $f(t)$ are *interpolations* formed by the sum of the sinc functions.

Obtaining the Discrete Fourier Transform from the Continuous Transform of a Sampled Function

We saw previously how the fourier transform of a band limited sampled function extending from -inf to +inf is another periodic function which is continuous that also extends from -inf to +inf. Of course in practice there are a finite number of samples. So how do we derive the discrete fourier transform with respect to these sample sets?

To begin we notice that here that the fourier transform of the sampling function s times the spatial domain function f is given by

$$\begin{aligned}
\tilde{F}(\mu) &= F(\mu) \star S(\mu) \\
&= \int_{-\infty}^{\infty} F(\tau) S(\mu - \tau)\, d\tau \\
&= \frac{1}{\Delta T} \int_{-\infty}^{\infty} F(\tau) \sum_{n=-\infty}^{\infty} \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \qquad (4 \\
&= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} F(\tau) \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\
&= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F\left(\mu - \frac{n}{\Delta T}\right)
\end{aligned}$$

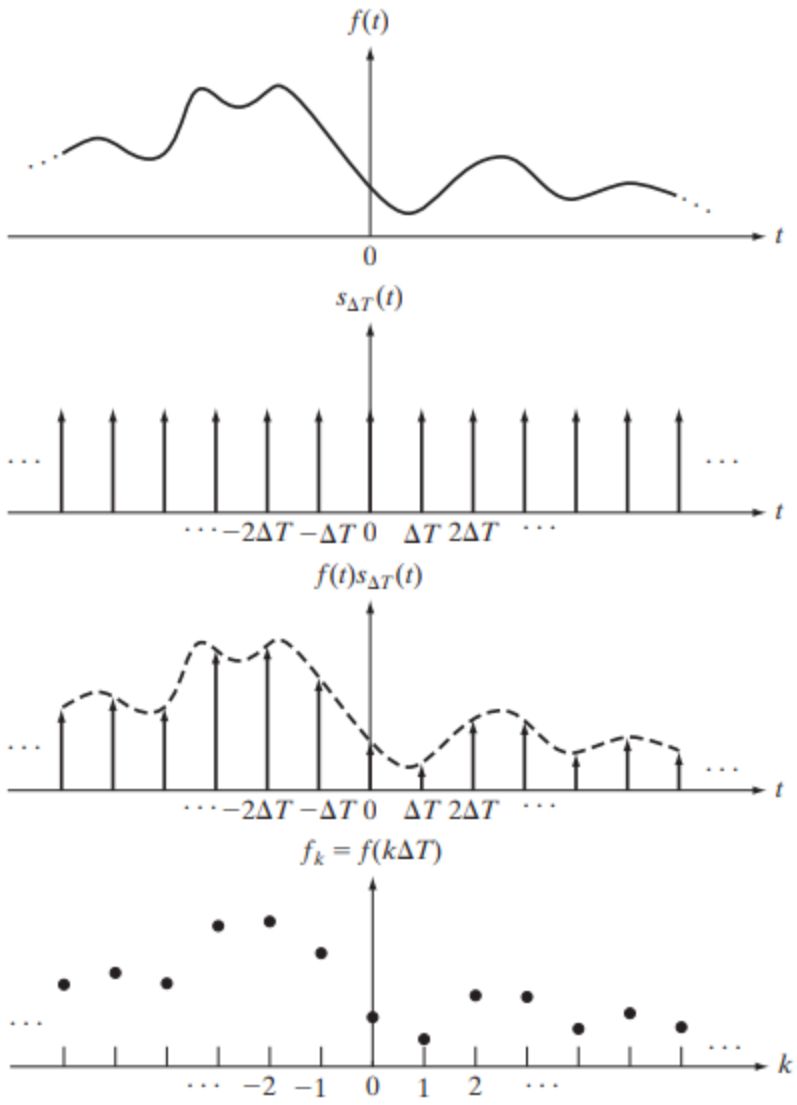
We notice that this writes F~ in terms of F, not in terms of f or f~. So lets get F~ in terms of f~ (this is clearly possible via the standard fourier transformation in the continuous setting)

$$\tilde{F}(\mu) = \int_{-\infty}^{\infty} \tilde{f}(t)\, e^{-j2\pi\mu t}\, dt$$

Then now if we do a couple substitutions accordingly we arrive to

$$\begin{aligned}
\tilde{F}(\mu) &= \int_{-\infty}^{\infty} \tilde{f}(t) e^{-j2\pi\mu t}\, dt \\
&= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(t)\, \delta(t - n\Delta T) e^{-j2\pi\mu t}\, dt \\
&= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} f(t)\, \delta(t - n\Delta T) e^{-j2\pi\mu t}\, dt \\
&= \sum_{n=-\infty}^{\infty} f_n\, e^{-j2\pi\mu n\Delta T}
\end{aligned}$$

It is important to note that for earlier we defined f_n = f(nΔT). It is important to note that f_n is a discrete function in this case, the reason f_n is discrete is because of this



(Discrete function due to sampling component in f~)
we also know its discrete fourier transform

$$= \sum_{n=-\infty}^{\infty} f_n e^{-j2\pi\mu n\Delta T}$$

is continuous since we earlier saw that

$$
\begin{aligned}
\tilde{F}(\mu) &= F(\mu) \star S(\mu) \\
&= \int_{-\infty}^{\infty} F(\tau)\, S(\mu - \tau)\, d\tau \\
&= \frac{1}{\Delta T} \int_{-\infty}^{\infty} F(\tau) \sum_{n=-\infty}^{\infty} \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\
&= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} F(\tau)\, \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\
&= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F\left(\mu - \frac{n}{\Delta T}\right)
\end{aligned}
$$

We know F~ is a continuous function and infinitely periodic with period 1/ΔT. Therefore all we need to characterize F~ is a single period. Sampling one period is the basis for the discrete fourier transform.
Suppose we want to get M equally spaced samples of F~ taken over the course of u = 0 to u = 1/ΔT. This is accomplished by taking the sum over the following frequencies

$$
\mu = \frac{m}{M \Delta T} \qquad m = 0, 1, 2, \ldots, M - 1
$$

The reason we go from 0 to M-1 is because its the same as 1 to M, since when m = 0 or m = M, it is going to be the same value assuming that its a periodic function split uniformly among M samples.

$$
F_m = \sum_{n=0}^{M-1} f_n e^{-j 2\pi m n / M} \qquad m = 0, 1, 2, \ldots, M - 1
$$

$$
\begin{aligned}
X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N} kn} \\
&= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N} kn\right) - i \cdot \sin\left(\frac{2\pi}{N} kn\right) \right],
\end{aligned}
\qquad \text{(Eq.1)}
$$

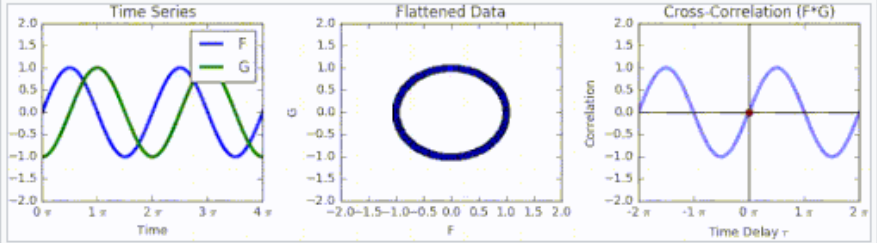
The k'th value in the fourier sequence is given by multiply each input sample from the 0'th to the n-1'st by the sinusoidal function given at that specific point defined by k. I still dont understand how this was derived.

Cross Correlation

In signal processing, cross correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. This can be seen as a sliding dot product. It is used for searching longer signals for a shorter known feature. Has similarities with the convolution of two functions.

Explanation [edit]

As an example, consider two real valued functions f and g differing only by an unknown shift along the x-axis. One can use the cross-correlation to find how much g must be shifted along the x-axis to make it identical to f. The formula essentially slides the g function along the x-axis, calculating the integral of their product at each position. When the functions match, the value of $(f \star g)$ is maximized. This is because when peaks (positive areas) are aligned, they make a large contribution to the integral. Similarly, when troughs (negative areas) align, they also make a positive contribution to the integral because the product of two negative numbers is positive.



Animation of how cross-correlation is calculated. The left graph shows a green function G that is phase-shifted relative to function F by a time displacement of τ. The middle graph shows the function F and the phase-shifted G represented together as a Lissajous curve. Integrating F multiplied by the phase-shifted G produces the right graph, the cross-correlation across all values of τ.

With complex-valued functions f and g, taking the conjugate of f ensures that aligned peaks (or aligned troughs) with imaginary components will contribute positively to the integral.

In econometrics, lagged cross-correlation is sometimes referred to as cross-autocorrelation.[8]:p. 74

The above explains why the cross correlation of two functions is

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} \overline{f(t)} g(t+\tau)\, dt$$

Another interesting thing to look at is

Properties [edit]

- The cross-correlation of functions $f(t)$ and $g(t)$ is equivalent to the convolution (denoted by $*$) of $\overline{f(-t)}$ and $g(t)$. That is:

  $$[f(t) \star g(t)](t) = [\overline{f(-t)} * g(t)](t).$$

- $[f(t) \star g(t)](t) = [\overline{g(t)} \star \overline{f(t)}](-t).$
- If f is a Hermitian function, then $f \star g = f * g.$
- If both f and g are Hermitian, then $f \star g = g \star f.$
- $(f \star g) \star (f \star g) = (f \star f) \star (g \star g).$
- Analogous to the convolution theorem, the cross-correlation satisfies

  $$\mathcal{F}\{f \star g\} = \overline{\mathcal{F}\{f\}} \cdot \mathcal{F}\{g\},$$

  where $\mathcal{F}$ denotes the Fourier transform, and an $\overline{f}$ again indicates the complex conjugate of f, since $\mathcal{F}\left\{\overline{f(-t)}\right\} = \overline{\mathcal{F}\{f(t)\}}$. Coupled with fast Fourier transform algorithms, this property is often exploited for the efficient numerical computation of cross-correlations[9] (see circular cross-correlation).
- The cross-correlation is related to the spectral density (see Wiener–Khinchin theorem).
- The cross-correlation of a convolution of f and h with a function g is the convolution of the cross-correlation of g and f with the kernel h:

  $$g \star (f * h) = (g \star f) * h.$$

Matched Filter

A matched filter is obtained by correlating a known delayed signal template with an unknown signal to detect the amount of presence of the template in the unknown signal. Equivalent to convolving the unknown signal with a conjugated time reversed version of the template.

The discrete fourier transform can be seen as the cross correlation of the input sequence x_n and the complex sinusoid at frequency k/N, therefore acting like a matched filter for that frequency.

$$= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N} kn\right) - i \cdot \sin\left(\frac{2\pi}{N} kn\right)\right],$$

The summation is from n = 0 to n = N-1 (iterating over all data points), Meanwhile the k/N gives the frequency for the sinusoidal function. Rememer that k/N is cycles per second. So we are measuring the template similarity between the n samples and the sinusoid with that frequency match. Whichever F_k from k=0,...,N-1 has the cycles/second that best match with the x_n for n=0,...,N-1 samples will be the highest value in the discrete fourier transform.

In terms of imaging

Of course then when k/N is at its highest frequency, if the data points plotted match mostly with this that implies that more information is with higher frequency regions, meanwhile if the data matches better with the lower frequency portions then there is more information amongst lower frequency portions of the image. Here is an example of actually applying the discrete fourier transform.

This example demonstrates how to apply the DFT to a sequence of length $N = 4$ and the input vector

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 - i \\ -i \\ -1 + 2i \end{pmatrix}.$$

Calculating the DFT of $\mathbf{x}$ using **Eq.1**

$$X_0 = e^{-i2\pi 0 \cdot 0/4} \cdot 1 + e^{-i2\pi 0 \cdot 1/4} \cdot (2 - i) + e^{-i2\pi 0 \cdot 2/4} \cdot (-i) + e^{-i2\pi 0 \cdot 3/4} \cdot (-1 + 2i) = 2$$
$$X_1 = e^{-i2\pi 1 \cdot 0/4} \cdot 1 + e^{-i2\pi 1 \cdot 1/4} \cdot (2 - i) + e^{-i2\pi 1 \cdot 2/4} \cdot (-i) + e^{-i2\pi 1 \cdot 3/4} \cdot (-1 + 2i) = -2 - 2i$$
$$X_2 = e^{-i2\pi 2 \cdot 0/4} \cdot 1 + e^{-i2\pi 2 \cdot 1/4} \cdot (2 - i) + e^{-i2\pi 2 \cdot 2/4} \cdot (-i) + e^{-i2\pi 2 \cdot 3/4} \cdot (-1 + 2i) = -2i$$
$$X_3 = e^{-i2\pi 3 \cdot 0/4} \cdot 1 + e^{-i2\pi 3 \cdot 1/4} \cdot (2 - i) + e^{-i2\pi 3 \cdot 2/4} \cdot (-i) + e^{-i2\pi 3 \cdot 3/4} \cdot (-1 + 2i) = 4 + 4i$$

results in $\mathbf{X} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 - 2i \\ -2i \\ 4 + 4i \end{pmatrix}.$

Inverse transform [edit]

The discrete Fourier transform is an invertible, linear transformation

$$\mathcal{F}: \mathbb{C}^N \to \mathbb{C}^N$$

with $\mathbb{C}$ denoting the set of complex numbers. Its inverse is known as Inverse Discrete Fourier Transform (IDFT). In other words, for any $N > 0$, an *N*-dimensional complex vector has a DFT and an IDFT which are in turn N-dimensional complex vectors.

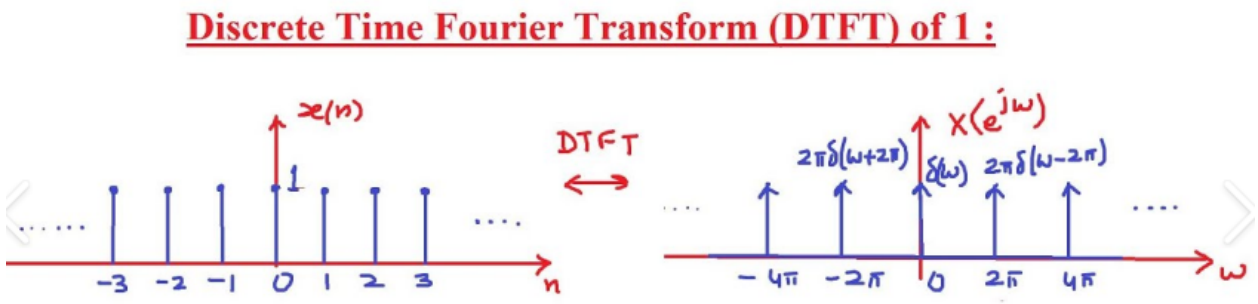
The inverse transform is given by:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i \frac{2\pi}{N} kn} \quad \text{(Eq.3)}$$

Back to the book
So we know that the standard fourier transform will result in

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j 2\pi mn/M} \qquad m = 0, 1, 2, \ldots, M-1$$

Then this implies that we are given M samples in the time domain, then we can retrieve M samples in the frequency domain. The picture below shows the shift in domain value



Now that we know the standard discrete fourier transform, the following is the inverse discrete fourier transform

$$f_n = \frac{1}{M}\sum_{m=0}^{M-1} F_m e^{j2\pi mn/M} \qquad n = 0, 1, 2, \ldots, M-1$$

The two transformations form a fourier transformation pair because if we plug f_n into the F_m equation we get F_m = F_m and vice versa. Notice how the ΔT is missing from the formulation, well that implies that the fourier transform can be done any M finite uniformly sampled discrete samples (makes sense as that variable really doesnt effect anything assuming we are taking things uniformly and within a known interval that describes behavior of entire set).

We will use x,y from now on to denote image coordinate variables and u and v for frequency variables. These are understood to be integers. We then get the equations

$$F(u) = \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \qquad u = 0, 1, 2, \ldots, M-1$$

The above we see that x is the pixel coordinates in the 1D image. This is why we have f(x) to denote the pixel intensity at coordinate x and x iterating from 0 to the M-1'th sample in its 1D coordinate axes. We compare the discrete distribution of intensities of each pixel to the complex sinusoidal at frequency u/M.

$$f(x) = \frac{1}{M}\sum_{u=0}^{M-1} F(u) e^{j2\pi ux/M} \qquad x = 0, 1, 2, \ldots, M-1$$

It turns out that both the forward and inverse discrete transforms are infinitely periodic (with period M). Well this is pretty intuitive since its a sinusoidal function that is taken over M samples that are supposed to fully characterize the thing.

$$F(u) = F(u + kM)$$

$$f(x) = f(x + kM)$$

Ofc k integer

The discrete equivalent to

$$f(t) \star h(t) = \int_{-\infty}^{\infty} f(\tau)\, h(t - \tau)\, d\tau$$

Is then

$$f(x) \star h(x) = \sum_{m=0}^{M-1} f(m) h(x - m)$$

Since the functions are periodic, so are their convolutions. This is why the equation is referred to as circular convolution and is a direct consequence of the periodicity of the discrete fourier transform and its inverse

periodicity of the DFT and its inverse. This is in contrast with the convolution you studied in Section 3.4.2, in which values of the displacement, x, were determined by the requirement of sliding one function completely past the other, and were not fixed to the range $[0, M - 1]$ as in circular convolution. We discuss

The convolution theorem is applicable to discrete variables.

Relationship between sampling and frequency intervals

If $f(x)$ consists of M samples of a function $f(t)$ taken ΔT units apart, the duration of the record comprising the set $\{f(x)\}$, $x = 0, 1, 2, \ldots, M-1$, is

$$T = M\Delta T \tag{4.4-11}$$

The corresponding spacing, Δu, in the discrete frequency domain follows from Eq. (4.4-3):

$$\Delta u = \frac{1}{M\Delta T} = \frac{1}{T} \tag{4.4-12}$$

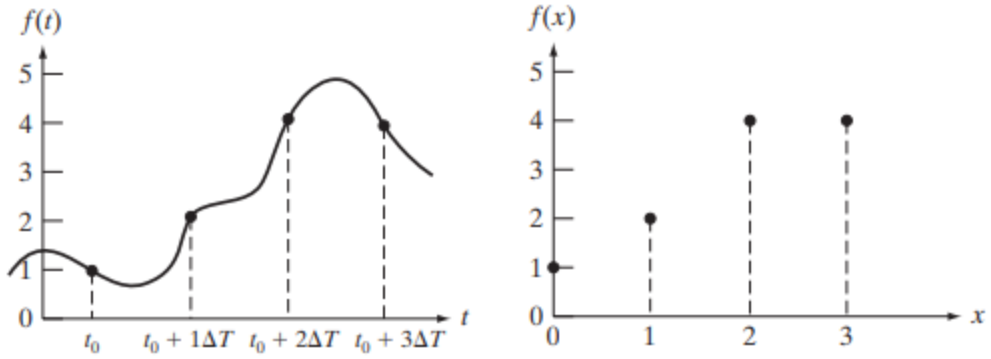
The entire frequency range spanned by the M components of the DFT is

$$\Omega = M\Delta u = \frac{1}{\Delta T} \tag{4.4-13}$$

Thus, we see from Eqs. (4.4-12) and (4.4-13) that that the resolution in frequency, Δu, of the DFT depends on the duration T over which the continuous function, $f(t)$, is sampled, and the range of frequencies spanned by the DFT depends on the sampling interval ΔT. Observe that both expressions exhibit *inverse* relationships with respect to T and ΔT.

The resolution (units here) in frequency depends on the number of samples with the given spacing between the samples.
The range of frequencies spanned by the DFT depends on simply the distance between samples. Of course when there is a larger spacing there will be less frequencies taken into account when doing the transform and when way less spacing we take into account a larger variety of resulting frequencies!

Now we will see an example of four samples being taken from a continuous function f(t) ΔT units apart. Then the next figure wil show the sampled values in x domain(can be seen as 1D image domain). The input value (from 0 - 3 inclusive) here has no direct relationship with the ordering of samples of f(t).



$$F(0) = \sum_{x=0}^{3} f(x) = \left[f(0) + f(1) + f(2) + f(3)\right]$$

$$= 1 + 2 + 4 + 4 = 11$$

The next value of $F(u)$ is

$$F(1) = \sum_{x=0}^{3} f(x)\, e^{-j2\pi(1)x/4}$$

$$= 1e^{0} + 2e^{-j\pi/2} + 4e^{-j\pi} + 4e^{-j3\pi/2} = -3 + 2j$$

Similarly, $F(2) = -(1 + 0j)$ and $F(3) = -(3 + 2j)$. Observe that *all* values of $f(x)$ are used in computing *each* term of $F(u)$.

If instead we were given $F(u)$ and were asked to compute its inverse, we would proceed in the same manner, but using the inverse transform. For instance,

$$
\begin{aligned}
f(0) &= \frac{1}{4}\sum_{u=0}^{3} F(u)e^{j2\pi u(0)} \\
&= \frac{1}{4}\sum_{u=0}^{3} F(u) \\
&= \frac{1}{4}[11 - 3 + 2j - 1 - 3 - 2j] \\
&= \frac{1}{4}[4] = 1
\end{aligned}
$$

which agrees with Fig. 4.11(b). The other values of $f(x)$ are obtained in a similar manner. ■

Extending to functions of two variables

The following is a pretty concise way of putting it all together.

4.5.1 The 2-D Impulse and Its Sifting Property

The impulse, $\delta(t, z)$, of two continuous variables, t and z, is defined as in Eq. (4.2-8):

$$\delta(t, z) = \begin{cases} \infty & \text{if } t = z = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.5-1a}$$

and

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(t, z)\, dt\, dz = 1 \tag{4.5-1b}$$

As in the 1-D case, the 2-D impulse exhibits the *sifting property* under integration,

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t, z)\,\delta(t, z)\, dt\, dz = f(0, 0) \tag{4.5-2}$$

or, more generally for an impulse located at coordinates (t_0, z_0),

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t, z)\,\delta(t - t_0, z - z_0)\, dt\, dz = f(t_0, z_0) \tag{4.5-3}$$

As before, we see that the sifting property yields the value of the function $f(t, z)$ at the location of the impulse.

For discrete variables x and y, the 2-D discrete impulse is defined as

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.5-4}$$

and its sifting property is

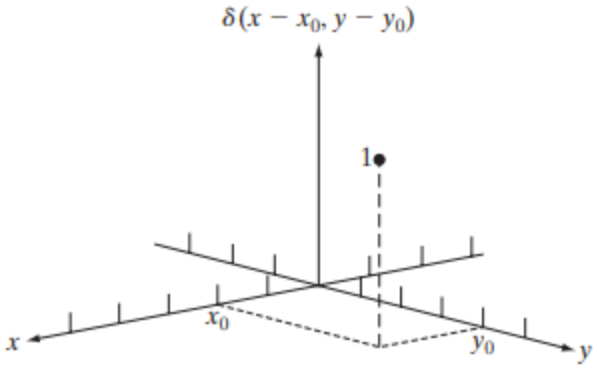
$$\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} f(x, y)\,\delta(x, y) = f(0, 0) \tag{4.5-5}$$

where $f(x, y)$ is a function of discrete variables x and y. For an impulse located at coordinates (x_0, y_0) (see Fig. 4.12) the sifting property is

$$\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} f(x, y)\,\delta(x - x_0, y - y_0) = f(x_0, y_0) \tag{4.5-6}$$

As before, the sifting property of a discrete impulse yields the value of the discrete function $f(x, y)$ at the location of the impulse.

A nice visualization of the above is



The 2D Continuous Fourier Transform Pair

Again, the book does a good job of being concise so I am just gonna attach the screenshot.

Let $f(t, z)$ be a continuous function of two continuous variables, t and z. The two-dimensional, continuous Fourier transform pair is given by the expressions

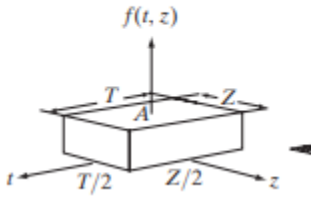
$$F(\mu, \nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t, z)\, e^{-j2\pi(\mu t + \nu z)}\, dt\, dz \qquad (4.5\text{-}7)$$

and

$$f(t, z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu, \nu)\, e^{j2\pi(\mu t + \nu z)}\, d\mu\, d\nu \qquad (4.5\text{-}8)$$

where μ and ν are the frequency variables. When referring to images, t and z are interpreted to be continuous *spatial* variables. As in the 1-D case, the domain of the variables μ and ν defines the *continuous frequency domain*.

Notice how the 2D case has the two different e's multiplied by their respective 1D domains/axes.



The above function will have a corresponding fourier transform of

$$F(\mu,\nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t,z)e^{-j2\pi(\mu t+\nu z)}\,dt\,dz$$

$$= \int_{-T/2}^{T/2}\int_{-Z/2}^{Z/2} A e^{-j2\pi(\mu t+\nu z)}\,dt\,dz$$

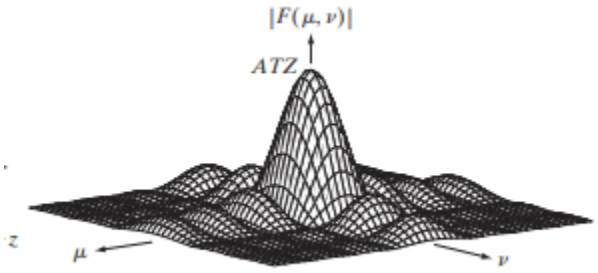
$$= ATZ\left[\frac{\sin(\pi\mu T)}{(\pi\mu T)}\right]\left[\frac{\sin(\pi\nu Z)}{(\pi\nu Z)}\right]$$

I think the authors made a mistake, if you're integrating with respect to dt first then based off of the figure above we would first integrate from -T/2 to T/2 (but the authors have us integrating from -Z/2 to Z/2.

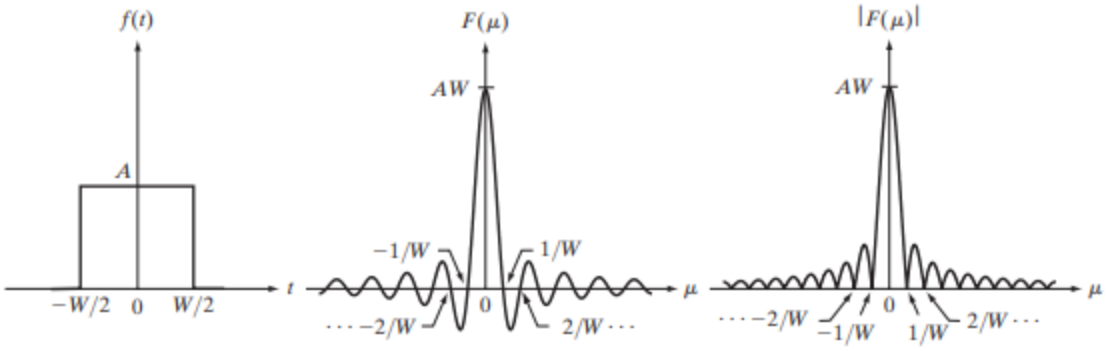
If we take the absolute of the resulting integration, we get

$$|F(\mu,\nu)| = ATZ\left|\frac{\sin(\pi\mu T)}{(\pi\mu T)}\right|\left|\frac{\sin(\pi\nu Z)}{(\pi\nu Z)}\right|$$

And the corresponding graph



Notice how block is longer along t axis, so spectrum is more 'contracted' along u axis. Comparing this with figure 4.4 we see that the spacing between points relatively on the same part as the consecutive curve is 1/W. Then it makes sense why the u axis is more contracted since the larger the box in that respective domain is the more close the consecutive spacings will be. (The smaller the phase).



2D Sampling & 2D Sampling Theorem

In a manner similar to the 1-D case, sampling in two dimensions can be modeled using the sampling function (2-D impulse train):

$$s_{\Delta T \Delta Z}(t, z) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - m\Delta T, z - n\Delta Z) \tag{4.5-9}$$

where ΔT and ΔZ are the separations between samples along the t- and z-axis of the continuous function $f(t, z)$. Equation (4.5-9) describes a set of periodic impulses extending infinitely along the two axes (Fig. 4.14). As in the 1-D case illustrated in Fig. 4.5, multiplying $f(t, z)$ by $s_{\Delta T \Delta Z}(t, z)$ yields the sampled function.

Function $f(t, z)$ is said to be *band-limited* if its Fourier transform is 0 outside a rectangle established by the intervals $[-\mu_{\max}, \mu_{\max}]$ and $[-\nu_{\max}, \nu_{\max}]$; that is,

$$F(\mu, \nu) = 0 \quad \text{for } |\mu| \geq \mu_{\max} \text{ and } |\nu| \geq \nu_{\max} \tag{4.5-10}$$

The *two-dimensional sampling theorem* states that a continuous, band-limited function $f(t, z)$ can be recovered with no error from a set of its samples if the sampling intervals are

$$\Delta T < \frac{1}{2\mu_{\max}} \tag{4.5-11}$$

and

$$\Delta Z < \frac{1}{2\nu_{\max}} \tag{4.5-12}$$

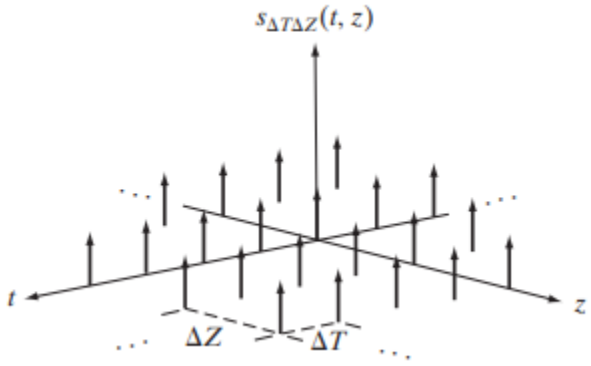
or, expressed in terms of the sampling rate, if

$$\frac{1}{\Delta T} > 2\mu_{\max}$$

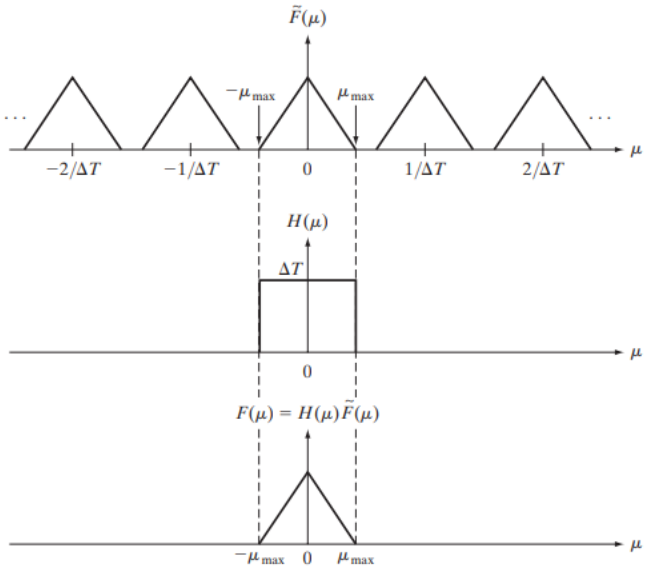
and

$$\frac{1}{\Delta Z} > 2\nu_{\max}$$

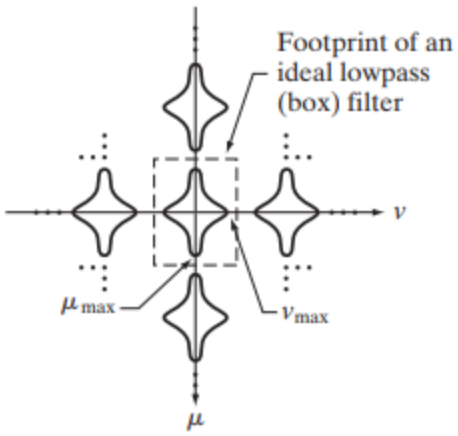
This is literally the most obvious extension possible. An example of 2D periodic impulses can be given by the following visual.



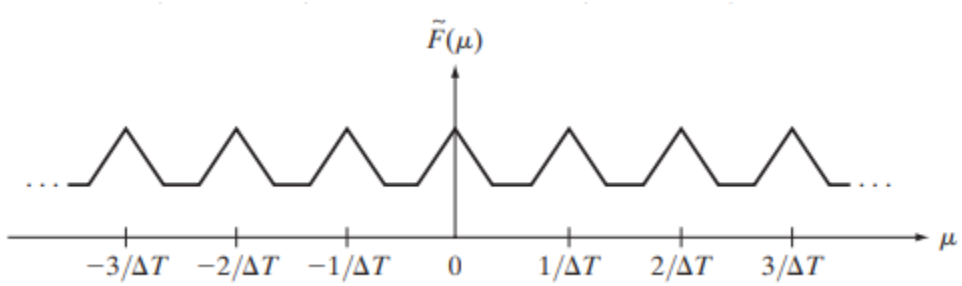
Now when we extend this idea of an ‘ideal lowpass filter’ from 1 dimensions to 2 dimensions



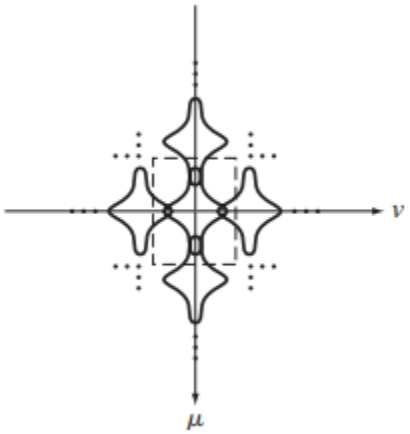
Now an ideal low pass filter for the 2d case will having the following “footprint” (height/z-axis 0 component)



Note that the low pass filter is independent of the sampling scheme. The above figure shows an over sampled situation and the below shows an under sampled situation.



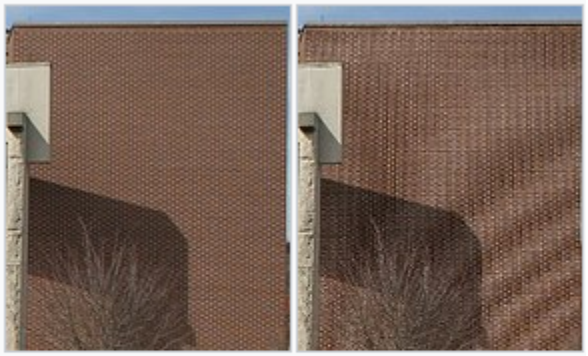
Is



Note that the footprint is the same in both cases, but one function was sampled properly (greater than the highest frequency in both dimensions) while the other was not.

Before we go over this I want to make sure the intuition for why aliasing occurs in practice regardless is truly locked in. Aliasing is a result of sampling under or equal twice the max frequency of the signal. It is the effect of overlapping frequency components due to this. It causes appearance of frequencies in the amplitude-frequency spectrum that are not in the original signal (since you apply the fourier transform on the sampled function). This image is an example of aliasing (notice how the higher intensities are masked a bit on the right (resulting in the moire pattern).

An example of aliasing



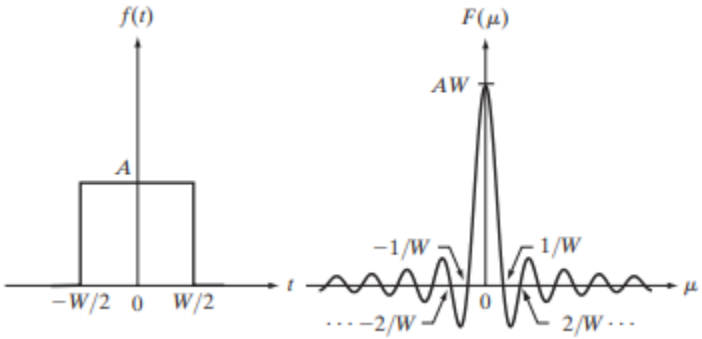
This full-sized image shows what a properly sampled image of a brick wall should look like with a screen of sufficient resolution.

When the resolution is reduced, aliasing appears in the form of a moiré pattern.

There exists aliasing known as temporal aliasing which is what happens when you dont capture the right interval of a sequence of images that are supposed to make up an object.
Aliasing can be avoided by applying low pass filters to the input before sampling. For example take temporal aliasing in music. If a piece of music sampled at 32,000 Hz (samples/second), well we know anything about 16,000Hz will cause aliasing when the music is put through a digital to analog converter, the high frequencies will appear as low frequencies (wrong alias) and so one method to prevent this is a filter that removes components above the nyquist frequency. I thought the nyquist rate was the sampling rate equivalent to two times the max frequency, so cutting off frequency to get to below the nyquist frequency is interesting to me? Maybe what its really saying is that if we use an independent filter (that has no knowledge of the type of frequencies it may encounter, then cutting the unknown frequencies below a certain point will allow the filter to capture what it has to in order to reproduce the proper image). The pre

processing anti aliasing portion will probably look a little off from the original image, however I believe they're reconstruction techniques to get it back to its original point.

It seems to me that the reason the book uses as to why aliasing will always occur in practice is because if we take a look at the fourier transform of a sampling function….



We notice how the height gets smaller as a function of distance from the origin. This extends on to infinitely far out. Therefore resulting in different values **infinitely far out** when it gets convolved with the original function (say h). It then makes sense why you can reduce aliasing by using a low pass filter on the original spatial domain, because when you do this, you lessen the height of the convolution of the two functions as it goes to infinity.

It also now makes sense that band limited functions extend infinitely, because if they are band limited then they have repetition/periodicness in their input which must extend to infinity otherwise the function would really be periodic.

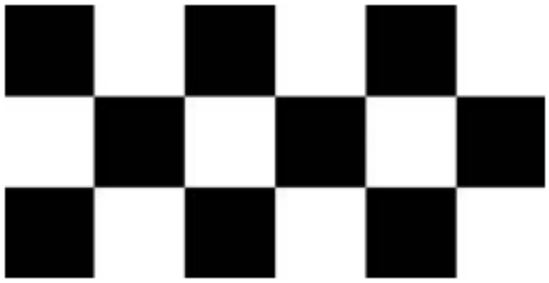
Aliasing in images

As in the single dimensional case, a continuous function f(t,z) can band limited in general only if it extends to positive and negative infinity on both coordinate axes. Of course the act of limiting the duration of the function introduces some aliasing due to the extension to infinity in the frequency domain introduced by the sampling function. Because we can not sample a function infinitely, aliasing will always be present in digital images.

Spatial aliasing is due to under sampling

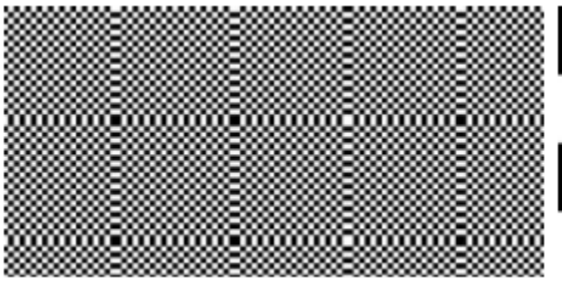
Some things that spatial aliasing causes are jaggedness of line features, spurious highlights, and presence of frequency patterns that are not actually in the original image.

Suppose we have an ideal imaging system (noiseless and produces exact image of what it sees), but the number of samples it can take is fixed at 96x96 pixels. Using this system to digitize checkerboard patterns, it can resolve patterns of up to 96x96 squares.
Think about it like we have a checkerboard with size 96x96 pixels, and the number of squares in that checkerboard vary per example. Then we can refer to the size of each square in the checkerboard as taking up a certain amount of pixels. In the case where each square in the checkerboard takes up 16 pixels, we see 6x6 square board, such as

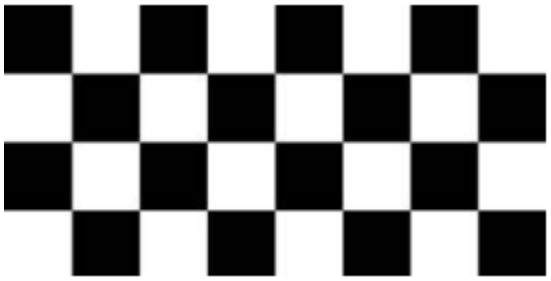


Now support each square in the checkerboard takes up 6 pixels each, this will result in a 16x16 checkerboard (96/6 = 16)

Now suppose each square is slightly less than the size of one pixel (more than 96x96 squares in the 96x96 pixel view, specifically below is the visual when each square in the checkerboard takes up 0.9174 pixels) (96/.9174)



The above is a clear example of very bad aliasing
However there is bad aliasing thats ‘masqueraded’ as a normal photo (take for example that each square is .4798pixels). Then we get



Take a moment and notice how we can interpret the ‘over sampling’ and ‘under sampling’ we talked about so much earlier with signals. Here we have a sampling amount of 96x96, and it turns out that when that is less than the number of squares in the checkerboard (in the last two pictures above), it results in aliasing (as in we are undersampling). For example to properly sample the third photo of shown here where each square is the size of .9174 pixels, we would need to increase the 96x96 pixel ‘sampling net’ to the size of (96/.9174)x(96x.9174) of course rounding up. Similarly for the next sample we would need to increase the sampling net size to (96/.4798)x(96/.4798). We were able to properly/oversample in cases where squares were size 6px and 16px.

So accordingly, the effect of aliasing is able to be reduced by slightly defocusing the scene so that high frequencies are taken attenuated (defocusing results in less squares in checkerboard in this case). Obviously the anti aliasing filtering must be done before the image is sampled, since sampling is where we pick up aliasing in the first place.
Side note: On modern digital cameras, the ‘anti aliasing’ option is for resampling. Not regarding reducing aliasing in original sampled image. Commercial digital cameras have good anti aliasing filters built in.

Image interpolation and resampling

Similarly to the single dimensional example, to perfectly reconstruct a band limited image function given a set of samples requires 2 dimensional convolution in spatial domain with a sinc function. In practice we look for approximations because we do not have infinite summations.

Common 2D interpolation practices involve image resizing (zooming and shrinking). Zooming can be seen as over sampling, while shrinking may be seen as under sampling. This makes sense as earlier we saw how when each square in the checkerboard take up 16 pixels, we only needed a 6x6 pixel camera to capture the details, however our camera was 96x96 pixels, therefore over sampling heavily. Therefore resulting in a more 'zoomed in' photo! Notice how when we required the 16x16 camera for when a square took up just 6 pixels each, it is slightly less zoomed in (look at the first two checkerboard examples above).
A special case of nearest neighbor interpolation that ties into over sampling is zooming via pixel replication (this is applicable when wanting to increase the size of an image by an integer number of times).
Zooming via pixel replication is very simple, we just duplicate the columns and then we duplicate the rows. Below is an illustrated example.

1	2
3	4

1	1	2	2
3	3	4	4

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Another example of zooming is **Zero Order Hold**
Where we take the original matrix

1	2
3	4

Then we sum the adjacent entries rowwise (3 and 7 here respectively). Then we divide each one by 2 and place them in the middle of the two columns in their respective rows. Resulting in,

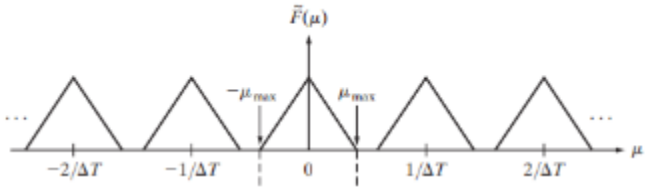
1	1	2
3	3	4

Then you can do column wise zooming by doing the same operation but in the other dimension (the final result is)

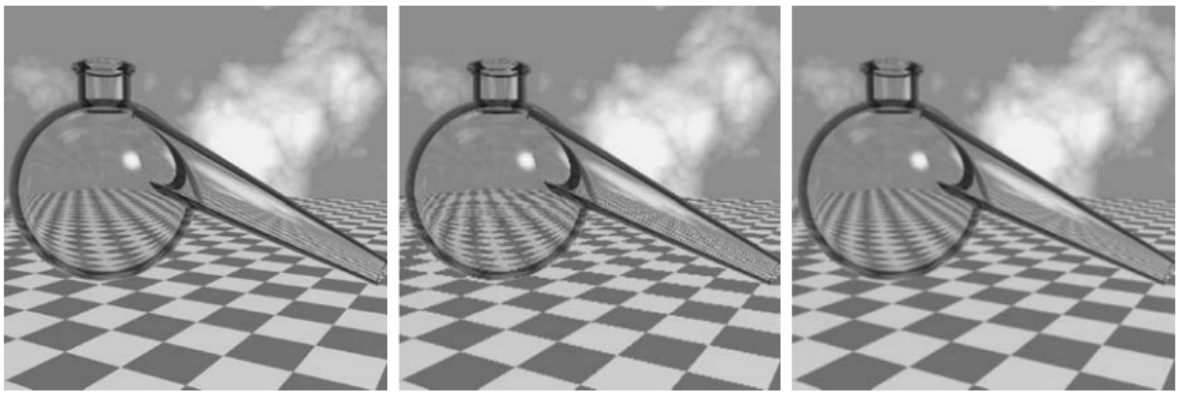
1	1	2
2	2	3
3	3	4

Image shrinking is done similar to zooming. To undersample we can do row-column removal where we shrink an image by ½ of its original size by deleting every other row/column.
Practical note:
It is a idea when reducing aliasing to blur an image before shrinking.
Another idea is to super sample then reduce its rows and columns via deletion (shrinking).
Effects of aliasing are **generally** worse when size of digital image reduced. Picture on left is properly sampled, picture in middle is result of performing row column deletion, picture in the right is result of 3x3 averaging filter. It is apparent that the aliasing effects (jaggedness effects here) are very prominent in the second picture which is directly associated with under sampling as that is what we are doing (higher frequency like the color white masking itself as a lower frequency like black). It is important to note that the second two pictures were resized to their original dimension via pixel replication as previously described to simplify comparisons.

Another key theme is that smoothing the image before resampling results in less aliasing (smoothing frequency can be seen in the 2D signals graph below as reducing how far out ‘u_max’ goes out). It is important to remember that frequency in an image is essentially the rate of change of intensities of nearby pixels, therefore attenuating higher frequencies in an image is the same thing as making pixels around eachother become more similar in value. Therefore it makes sense that ‘averaging’ a region attenuates higher frequencies on a graph like below and on an image like above.



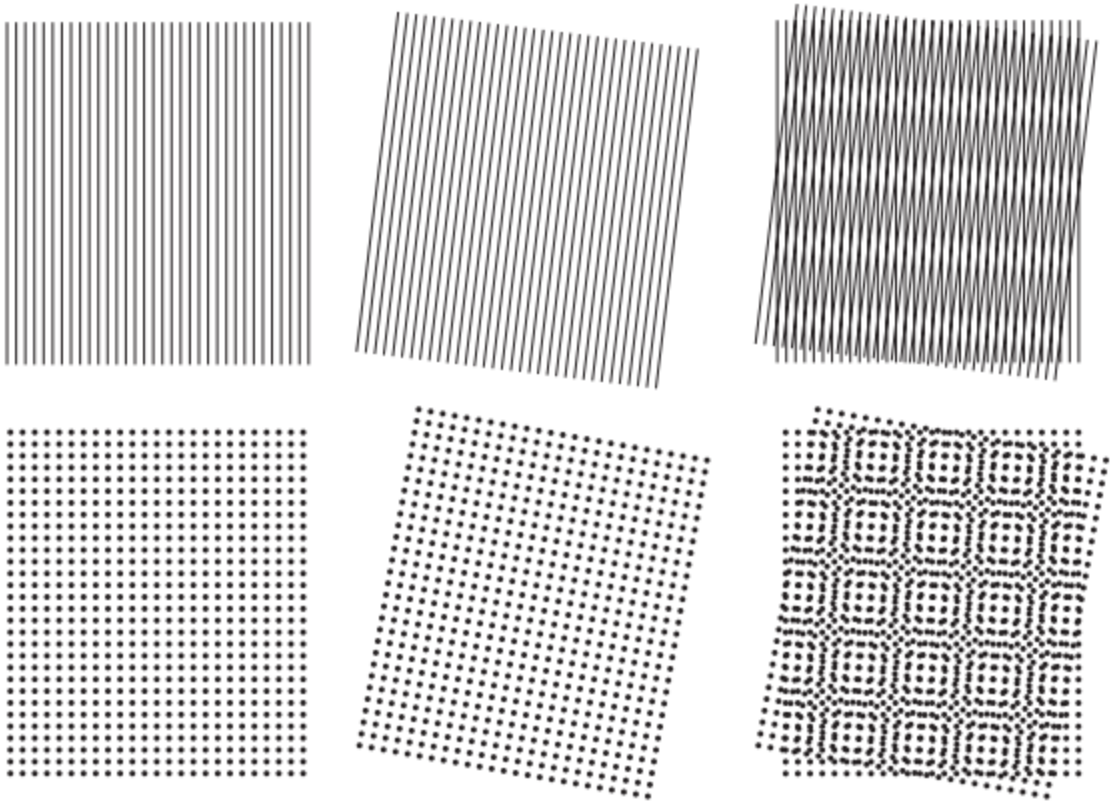
Another example of averaging helping attenuate the higher frequencies in an image.



a b c

FIGURE 4.18 Illustration of jaggies. (a) A 1024 × 1024 digital image of a computer-generated scene with negligible visible aliasing. (b) Result of reducing (a) to 25% of its original size using bilinear interpolation. (c) Result of blurring the image in (a) with a 5 × 5 averaging filter prior to resizing it to 25% using bilinear interpolation. (Original image courtesy of D. P. Mitchell, Mental Landscape, LLC.)

Moire Pattern



Rightmost side is the moire pattern, occurs when you take 2 ‘periodic’ images and then rotate one of them and super impose it on the other. Resulting ina nice pattern.

The 2D Discrete Fourier Transform and Its Inverse

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi(ux/M + vy/N)}$$

$$u = 0, 1, 2, \ldots, M-1 \text{ and } v = 0, 1, 2, \ldots, N-1.^{\dagger}$$

Again (t,z) and (n,s) denote the 2d continuous spatial and frequency domain variables. In the 2d discrete case (x,y) for spatial cariables and (u,v) for frequency domain variables.

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)} \qquad (4.5\text{-}16)$$

for $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$. Equations (4.5-15) and (4.5-16) constitute the 2-D *discrete Fourier transform pair*. The rest of this

Before we go into properties of the 2 dimensional discrete case, it is good to look at a reminder of how fourier transforms are used specifically in image processing from another source.

So far a lot of the content has been foundational and regarding the fourier transform on general signals, under sampling, over sampling, derivations of the DFT, Interepereting filters, distribution matching..etc

However we have not seen anything regarding why these are ACTUALLY useful for images, simply hand wavey type comments. Below goes over a nice and very clear explanation on its actual purpose in imaging. I recommend scrolling back up to the 'Matched Filter' & 'Cross Correlation' portion so that way the intuition really locks in for why we get the results we do directly below and why looking at a gresycale with its center at the middle even has any meaning to begin with.

Here is the link that the following practical example is associated with ~ [Image Transforms - Fourier Transform (ed.ac.uk)]

'**The fourier transform is an important image processing tool which can be used to decompose an image into its sine and cosine components, the output of the transformation represents the image in the fourier/frequency domain while the input image is the spatial domain equivalent, in the fourier domain each point represents a particular frequency contained in the spatial domain image.**

We will only concern ourselves with the discrete case in covering this chapter because that is what we apply to images (pixels are not continuous). We know already that the 2D NxN pixel image can be transformed via the following (where f(i,j) denotes intensity at coordinate (i,j) in the image).

$$F(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j)\, e^{-\iota 2\pi (\frac{ki}{N} + \frac{lj}{N})}$$

,

Earlier we noted how we can interperet the discrete fourier transform as a cross correlation measure between the complex sinusoidal with frequency k/N (and l/N

here) and the discrete N^2 samples found in the image. Another way to describe this is that each point of F(k,l) is obtained by multiplying the entire spatial image via the corresponding base function and summing the result.
The basis functions have increasing frequencies, **F(N-1, N-1) represents highest frequency.**
The inverse fourier transform is given by a simple sign change of the exponentials argument and is given by

$$f(a,b) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l)\, e^{\iota 2\pi (\frac{ka}{N} + \frac{lb}{N})}$$

Notice how for the fourier transform described above this figure, it involves a double sum. However the fourier transform is **separable** therefore it can be done in two single sums rather than a double sum!
Below we see what that lets us do (2N operations rather than the N^2 operations implied in the double sum)
First we do

$$P(k,b) = \frac{1}{N} \sum_{a=0}^{N-1} f(a,b)\, e^{-\iota 2\pi \frac{ka}{N}}$$

(again remembering that f(a,b) corresponds to pixel intensity at coordinate (a,b)) and then plug this into

$$F(k,l) = \frac{1}{N} \sum_{b=0}^{N-1} P(k,b)\, e^{-\iota 2\pi \frac{lb}{N}}$$

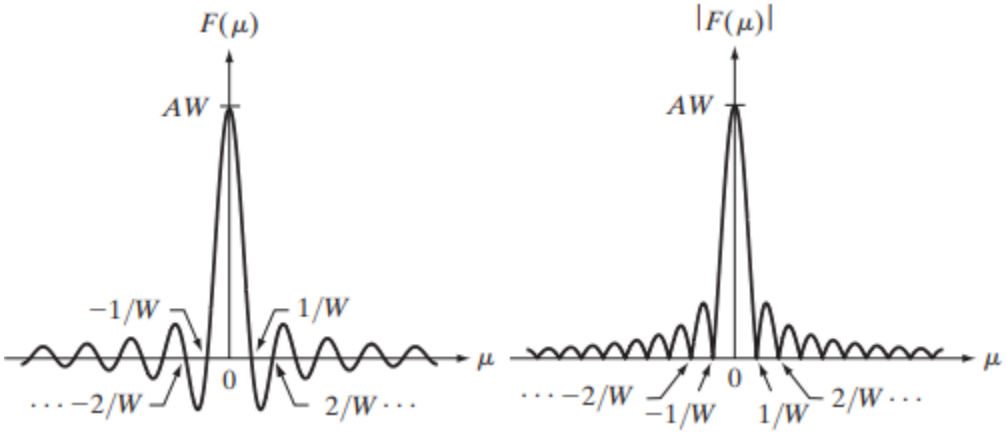
There is an even faster method that reduces compute of this transform from 2N to Nlog_2(N), however we will not go over that right now. This method is known as the fast fourier transform.

Why use the fourier transform on images

If we want to access some of the geometric characteristics of said image. Decomposing the image into its sinusoidal components, we are able to examine certain frequencies of the image. Most of the implementations involving fourier transforms have F(0,0) (the image mean) displayed in the center of the image (via shifting the image). Now with this convention, the further away from the center of an image the point is, the higher its corresponding frequency.
Now with this in mind, if we apply the fourier transform to this image

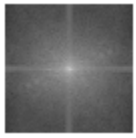
And then do the required shifting as well as take the magnitude of the image we get

(**If taking the absolute value feels random, note that the below two graphs contain essentially the same information, the right side is the absolute value graph of the fourier transform on the left)**



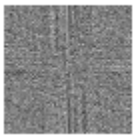
BACK TO THE ALMOST FULLY BLACK IMAGE

Notice the values are too large to be displayed on the screen in the frequency domain, so we apply a logarithmic transform (maintains relative intensity differences between frequencies).



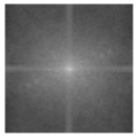
Then we notice that the magnitude is lower for higher frequencies (less bright the further from center of image we go), therefore the lower frequency components

of the image contain more information. We can also see there are two dominating magnitudes in the fourier image (in horizontal and vertical direction), which implies patterns exist somewhere in the patterns of the background of the original image **(since the frequency components have high magnitude in a line shape this is why we made the assumption of a 'pattern' in original image)**
Below is an image corresponding to phase, which isnt very useful itself for any information retrieval, but is necessary for reconstructing the original image as we see next.



Lets try reconstructing this

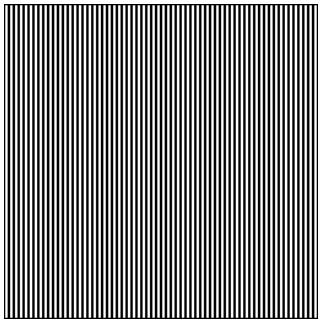
Using just



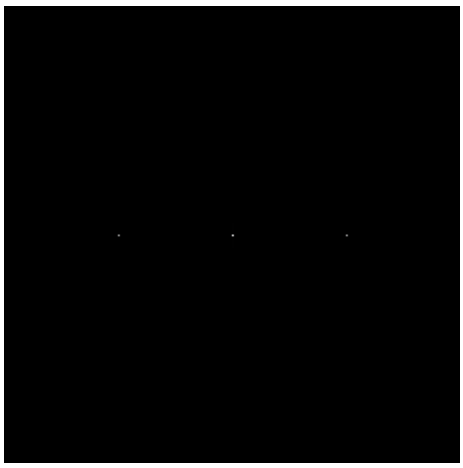
We get

And that is clearly a terrible reconstruction **(Hence we need the phase fourier image too)**
So it seems like in the case where we have a fourier transform that tells us that most of the information is stored in lower frequency components, we can apply some averaging to attenuate the higher frequency components as they do not contain much information.
If most intensities associated with the pixels are aligned with the lower frequencies, then this corresponds to bright points along lower frequencies (**closer to middle after we shift**). I am basing this intuition off of the 1D DFT case. **This paragraph should help tie everything together morally/intuitively on why we have dedicated so much time to understanding this transform.**

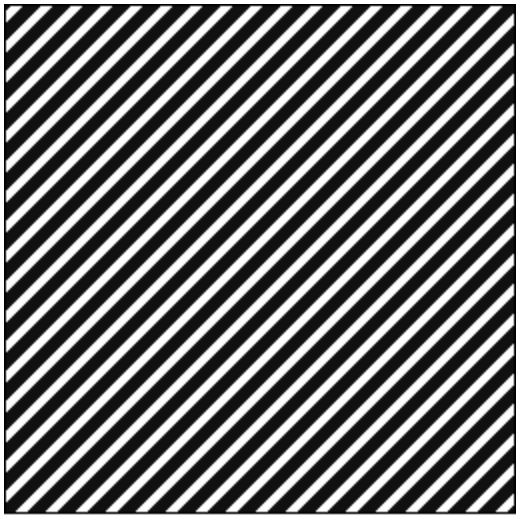
Now the page linked earlier goes into even more detail.
For example if given the image



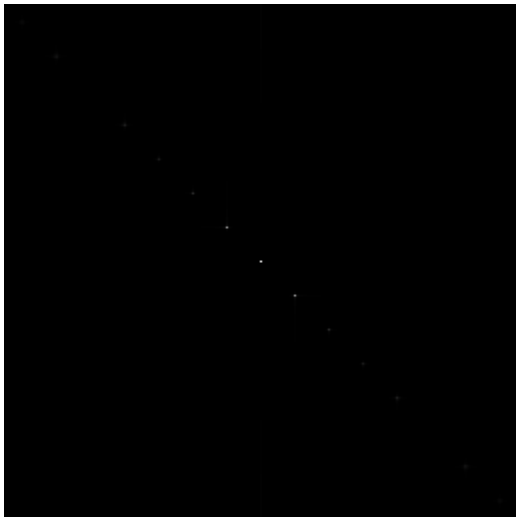
And then performing the fourier transform



We notice that we get 3 dots and they are symmetric about the center.
Geometrically we can interpret this as saying that the image intensity in the spatial domain chances the most if we go along it horizontally (frequency highest along horizontal axis), which we are able to visually confirm.
Now if we perform fourier transform on



We get



Which corresponds to highest frequency along diagonal. Note that in the original image we were not able to give a geometric interpretation of what we were seeing in the original image with the magnitude of the fourier transform because the image did not display as strong of a pattern as these images are now showing. Additionally the pattern shown is independent of actual placement on the image (I think… I say this due to the translation that is imposed of moving the smallest frequency to the center of the image).

A fourier transformed image can be used for frequency filtering.
We will touch more on smoothing, sharpening, and more, later on.

Some properties of the 2D discrete fourier transform

Relationship between spatial andfrequency intervals.
Suppose that a continuous f(t,z) sampled to form digital image f(x,y), consisting of MxN samples taken in the t and z directions. Let ΔT and ΔZ denote separations between samples. Then the separations between corresponding discrete frequency domain variables are given by!

$$\Delta u = \frac{1}{M\Delta T}$$

$$\Delta v = \frac{1}{N\Delta Z}$$

One can interpret the above as noting how, 1/ΔT is the total range of the values we are considering in the frequency domain and how we divide that into M samples along that range. Similar interpretation for delta v.

The fourier transform pair satisfies the following properties

$$f(x, y)e^{j2\pi(u_0x/M+v_0y/N)} \Leftrightarrow F(u - u_0, v - v_0)$$

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-j2\pi(x_0u/M+y_0v/N)}$$

Multiplying f(x,y) by the above exponential shifts the origin of the discrete fourier transform to u_0, v_0.
Conversely multiplying F(u,v) by the negative exponential shifts the origin of f(x,y) to (x_0,y_0).

Using the polar coordinates

$$x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$$

results in the following transform pair:

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0) \qquad (4.6\text{-}5)$$

which indicates that rotating $f(x, y)$ by an angle θ_0 rotates $F(u, v)$ by the same angle. Conversely, rotating $F(u, v)$ rotates $f(x, y)$ by the same angle.

Periodicity
Similar to single dimension, the two dimensional fourier transform and its inverse are infinitely periodic in the u and v directions

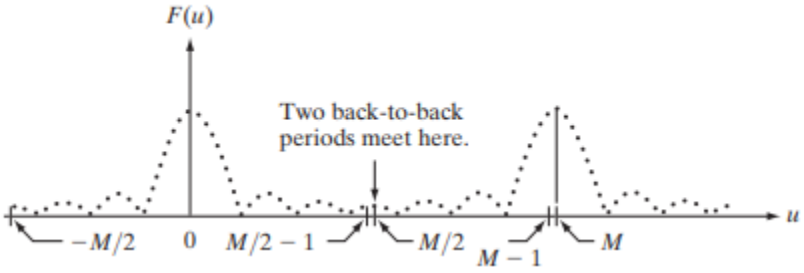
$$F(u, v) = F(u + k_1M, v) = F(u, v + k_2N) = F(u + k_1M, v + k_2N) \quad (4.6\text{-}6)$$

and

$$f(x, y) = f(x + k_1M, y) = f(x, y + k_2N) = f(x + k_1M, y + k_2N) \quad (4.6\text{-}7)$$

where k_1 and k_2 are integers.

Periodicity is important when implementing dft based algorithms, suppose we have the following graph



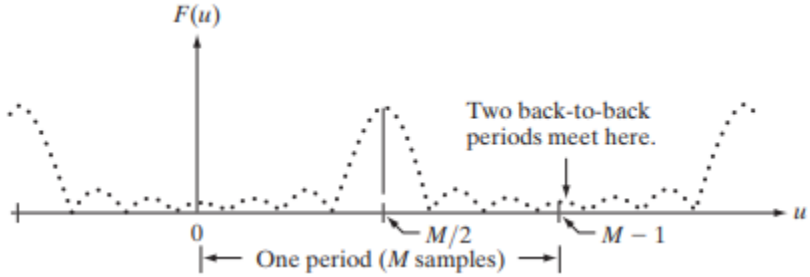
Notice how we have two back to back periods from 0 to M-1 inclusive. What if instead of this, we want our period to be one full period that is continuous and has no stop in the middle? Well we can simply shift the origin of the above fourier transform over such that its origin is at M/2.
Well if we want to do this, note that from one of the properties listed above we have
It follows from Eq. (4.6-3) that

$$f(x)e^{j2\pi(u_0x/M)} \Leftrightarrow F(u - u_0)$$

Therefore the origin F(0) will now be located at u_0. Letting u_0 = M/2, we get that the exponential term on the left becomes e^(jπx) = (-1)^x since x is an integer. So,

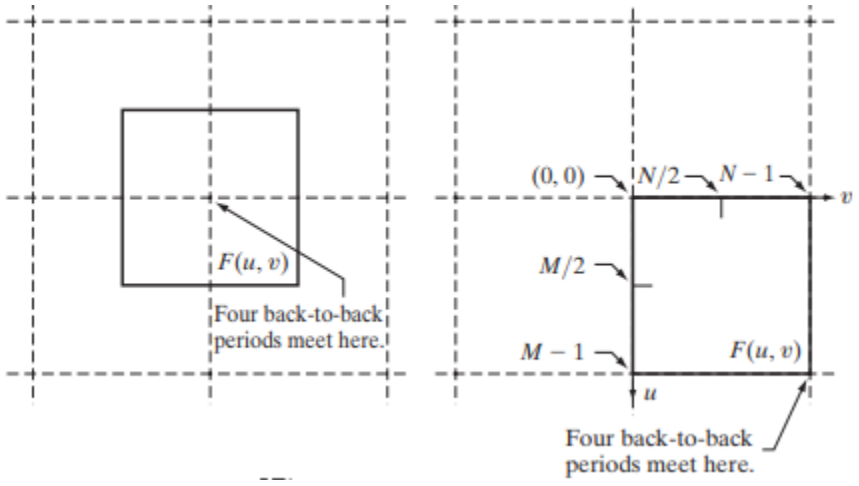
$$f(x)(-1)^x \Leftrightarrow F(u - M/2)$$

Remember that the above notation implies that after we multiply f(x) by (-1)^x we can then perform the fourier transform on this product to retrieve what is on the right of the double sided arrow, and inversely we can perform the inverse fourier transform on what is on the right side of the arrow to get what is on its left. Anyways, the resulting graph of performing the transform on the left side is ..



This was exactly our goal.

As in the 1 dimensional case, graphically we can see how we would do this in two dimensions (rather than back to back, the problem becomes back to back to back to back)



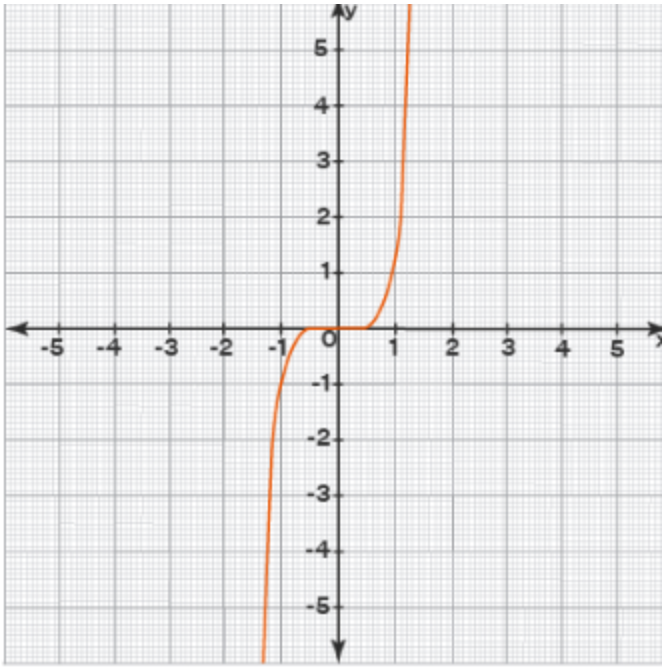
is simplified if we shift the data so that $F(0, 0)$ is at $(M/2, N/2)$. Letting $(u_0, v_0) = (M/2, N/2)$ in Eq. (4.6-3) results in the expression

$$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2) \qquad (4.6\text{-}8)$$

Using this equation shifts the data so that $F(0, 0)$ is at the center of the *frequency rectangle* defined by the intervals $[0, M - 1]$ and $[0, N - 1]$, as

Symmetry Properties

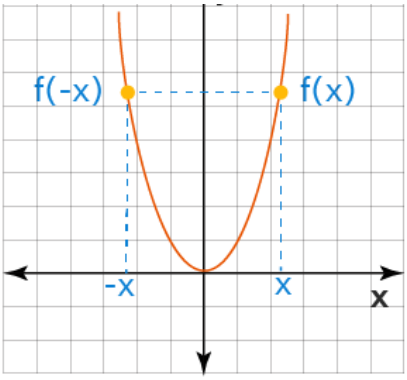
A result that comes from calculus is that a function is odd if that function (lets suppose defined by f) has the property that f(-x) = -f(x). This makes sense as the negative value is preserved (for example take f(c) = c^3, of course f(-c) = -c^3 and the negative sign is kept but notice how this also equals -f(c). Odd functions have property that they are symmetric about origin. Take this for example

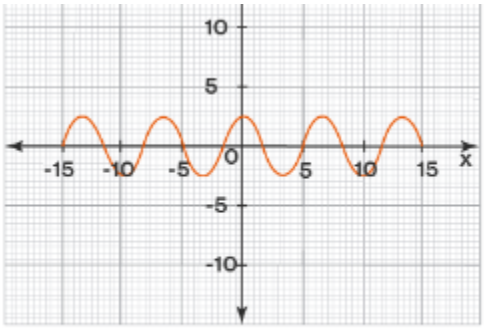


This makes sense as negative inputs give the negative of the equivalent positive input.
It is important to not associate odd functions with polynomials that contain only odd powers, for example f(x) = x^3 + 2 is not odd as -x^3 + 2 =/= -x^3 - 2.

Properties of Odd Functions
-Sum of two functions is odd
-Difference between two odd functions is odd
-The product of two odd functions is even
-Another way to rephrase the above is that the quotient when dividing one odd function by another is even (analogous to integers, if a/b = 2q for integers a,b,q then a = 2qb => a*b = 2qb^2, if we let c = qb^2 then its super obvious that a*b is even).
-Composition of two odd functions is odd
-Composition of an even function and odd function even

Even functions
These are functions (lets define by g) s.t. g(x) = g(-x). Of course one requirement is that the powers in any of the expressions is even. Examples include g(x) = x^2, x^4, x^6, … , x^(2n), another example is that cos(x) = cos(-x), therefore g(x) = cos(x) is an even function.
It should be obvious that the graph will be symmetric across Y axis





Properties

The sum of two even functions is even.
The difference between the two even functions is even.
The product of two even functions is even.
The quotient of the division of two even functions is even.
The composition of two even functions is even.
The composition of an even and odd function is even.

Why mention odd and even functions in the first place?

It turns out that using functional analysis its possible to prove that any function can be written as the sum of an even function and an odd function. Say we have the following,

$$w(x, y) = w_e(x, y) + w_o(x, y)$$

This function can be decomposed into its even and odd constituents as seen above, with each of those equalling…

$$w_e(x, y) \triangleq \frac{w(x, y) + w(-x, -y)}{2}$$

$$w_o(x, y) \triangleq \frac{w(x, y) - w(-x, -y)}{2}$$

Of course, just like we talked about before in the 1 dimensional input case, it holds for 2 dimensions as well…

$$w_e(x, y) = w_e(-x, -y)$$

$$w_o(x, y) = -w_o(-x, -y)$$

In order to translate the above in terms of discrete fourier transforms and its indices, the way we can refer to symmetry is via the **center point** of the given sequence. Indices to the right of the center point will be seen as positive inputs and those to the left are negative inputs.
It will be more convenient to think in terms of non negative indices, so we will add an offset to the inputs with respect to the rows and columns. Remembering that x denotes horizontal and y denotes vertical it makes sense that M denotes number of columns and N denotes number of rows. Therefore resulting in the new formulation…

$$w_e(x, y) = w_e(M - x, N - y)$$

$$w_o(x, y) = -w_o(M - x, N - y)$$

As previously stated, the product of two evens or two odd functions is even. Also the product of an odd and even function is odd, therefore in terms of discrete points, the following sum makes sense

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} w_e(x, y)\, w_o(x, y) = 0$$

Before we look at the next example, notice that if we think of the midpoint as being the center and every point to right as being positive and every point to left as being negative, then for a function to be even based off of the above we know that its M/2 + i indice input has to equal its M/2 - i indice input, which is equivalent to stating that f(i) = f(M-i) (can be thought of as for however much i travels forward, i will assist in travelling backwards when taking from M). So now it should be clear that for this sequence

$$f = \begin{Bmatrix} f(0) & f(1) & f(2) & f(3) \end{Bmatrix}$$

$$= \begin{Bmatrix} 2 & 1 & 1 & 1 \end{Bmatrix}$$

To test for evenness we need to check that

$$f(x) = f(4 - x)$$

In other words check if
f(0)=f(4), f(1)=f(3), f(2)=f(2).

It turns out to be even.
Additionally any 4 sequence of form

$$\{a \quad b \quad c \quad b\}$$

Is even.

Now applying what we discussed earlier on odd functions, it makes sense that an odd function in terms of our discrete set will have to satisfy the property that f(x) = -f(M-x).

Any four point odd sequence has form

$$\{0 \quad -b \quad 0 \quad b\}$$

The preceding discussion indicates that evenness and oddness of sequences depend also on the length of the sequences. For example, we already showed that the sequence $\{0 \quad -1 \quad 0 \quad 1\}$ is odd. However, the sequence $\{0 \quad -1 \quad 0 \quad 1 \quad 0\}$ is neither odd nor even, although the "basic" structure appears to be odd. This is an important issue in interpreting DFT results. We show later in this section that the DFTs of even and odd functions have some very important characteristics. Thus, it often is the case that understanding when a function is odd or even plays a key role in our ability to interpret image results based on DFTs.

Applying this to 2D is a clear extension where we do the same offset and sign changes but to two variables instead of one. Using that extension onto this discrete 2D set

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

We get that this set is odd, however if we added another row or columns of zeros, we would get neither odd nor even.

It turns out that **the fourier transform of a real function is conjugate symmetric**. Hence

$$F^*(u, v) = F(-u, -v)$$

The derivation for this proof is quite simple, we just have to remember that conjugating a fully imaginary number will just change its sign while conjugating a fully real value will leave it untouched. Then we see that…

$$F^*(u, v) = \left[\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)} \right]^*$$

$$= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f^*(x, y) e^{j2\pi(ux/M + vy/N)}$$

$$= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi([-u]x/M + [-v]y/N)}$$

$$= F(-u, -v)$$

^f(x,y) unaddected by the dot.

Additionally the **fourier transform of a complex function is conjugate antisymmetric** $F^*(-u, -v) = -F(u, v)$.

This can be proved similarly to the f(x,y) being fully real

Properties listed

	Spatial Domain†		Frequency Domain†
1)	$f(x, y)$ real	$\Leftrightarrow$	$F^*(u, v) = F(-u, -v)$
2)	$f(x, y)$ imaginary	$\Leftrightarrow$	$F^*(-u, -v) = -F(u, v)$
3)	$f(x, y)$ real	$\Leftrightarrow$	$R(u, v)$ even; $I(u, v)$ odd
4)	$f(x, y)$ imaginary	$\Leftrightarrow$	$R(u, v)$ odd; $I(u, v)$ even
5)	$f(-x, -y)$ real	$\Leftrightarrow$	$F^*(u, v)$ complex
6)	$f(-x, -y)$ complex	$\Leftrightarrow$	$F(-u, -v)$ complex
7)	$f^*(x, y)$ complex	$\Leftrightarrow$	$F^*(-u - v)$ complex
8)	$f(x, y)$ real and even	$\Leftrightarrow$	$F(u, v)$ real and even
9)	$f(x, y)$ real and odd	$\Leftrightarrow$	$F(u, v)$ imaginary and odd
10)	$f(x, y)$ imaginary and even	$\Leftrightarrow$	$F(u, v)$ imaginary and even
11)	$f(x, y)$ imaginary and odd	$\Leftrightarrow$	$F(u, v)$ real and odd
12)	$f(x, y)$ complex and even	$\Leftrightarrow$	$F(u, v)$ complex and even
13)	$f(x, y)$ complex and odd	$\Leftrightarrow$	$F(u, v)$ complex and odd

Examples of these properties

Property	$f(x)$		$F(u)$
3	$\{1\ \ 2\ \ 3\ \ 4\}$	$\Leftrightarrow$	$\{(10)\ (-2 + 2j)\ (-2)\ (-2 - 2j)\}$
4	$j\{1\ \ 2\ \ 3\ \ 4\}$	$\Leftrightarrow$	$\{(2.5j)\ (.5 - .5j)\ (-.5j)\ (-.5 - .5j)\}$
8	$\{2\ \ 1\ \ 1\ \ 1\}$	$\Leftrightarrow$	$\{(5)\ (1)\ (1)\ (1)\}$
9	$\{0\ \ -1\ \ 0\ \ 1\}$	$\Leftrightarrow$	$\{(0)\ (2j)\ (0)\ (-2j)\}$
10	$j\{2\ \ 1\ \ 1\ \ 1\}$	$\Leftrightarrow$	$\{(5j)\ (j)\ (j)\ (j)\}$
11	$j\{0\ \ -1\ \ 0\ \ 1\}$	$\Leftrightarrow$	$\{(0)\ (-2)\ (0)\ (2)\}$
12	$\{(4 + 4j)\ (3 + 2j)\ (0 + 2j)\ (3 + 2j)\}$	$\Leftrightarrow$	$\{(10 + 10j)\ (4 + 2j)\ (-2 + 2j)\ (4 + 2j)\}$
13	$\{(0 + 0j)\ (1 + 1j)\ (0 + 0j)\ (-1 - j)\}$	$\Leftrightarrow$	$\{(0 + 0j)\ (2 - 2j)\ (0 + 0j)\ (-2 + 2j)\}$

Fourier Spectrum and Phase Angle
Since the fourier transform is generally complex we can describe it in polar form

$$F(u, v) = |F(u, v)|e^{j\phi(u,v)}$$

w/ **Fourier Spectrum**

$$|F(u, v)| = \left[R^2(u, v) + I^2(u, v)\right]^{1/2}$$

And with **phase angle**

$$\phi(u, v) = \arctan\left[\frac{I(u, v)}{R(u, v)}\right]$$

Lastly, the **Power Spectrum** is

$$\begin{aligned} P(u, v) &= |F(u, v)|^2 \\ &= R^2(u, v) + I^2(u, v) \end{aligned}$$

If a function is conjugate symmetric, then the spectrum is even about origin (take the magnitude of the derivation we did earlier to prove conjugate symmetry of transform of a real function f(x,y))

$$|F(u, v)| = |F(-u, -v)|$$

The phase angle has odd symmetry ab origin,

$$\phi(u, v) = -\phi(-u, -v)$$

We know that if we set u and v equal to 0 in the earlier transform equation, we get the simple sum over all the indices. This sum will clearly be proportional to the average value/intensity of the indices/pixels. To see this..

$$F(0, 0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

$$\begin{aligned} F(0, 0) &= MN\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \\ &= MN\bar{f}(x, y) \end{aligned}$$

Note that (u,v) = (0,0) is implying that we are at zero frequency. Since MN is typically very big, |F(0,0)| tends to be the largest component of spectrum. Since u and v are zero at origin, F(0,0) is called the dc component of the transform (direct current => Zero Frequency).
Before reading this next part, we need these 3 formulas we have already seen to keep up.

$$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$$

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-j2\pi(x_0u/M + y_0v/N)} \qquad (4.6\text{-}4)$$

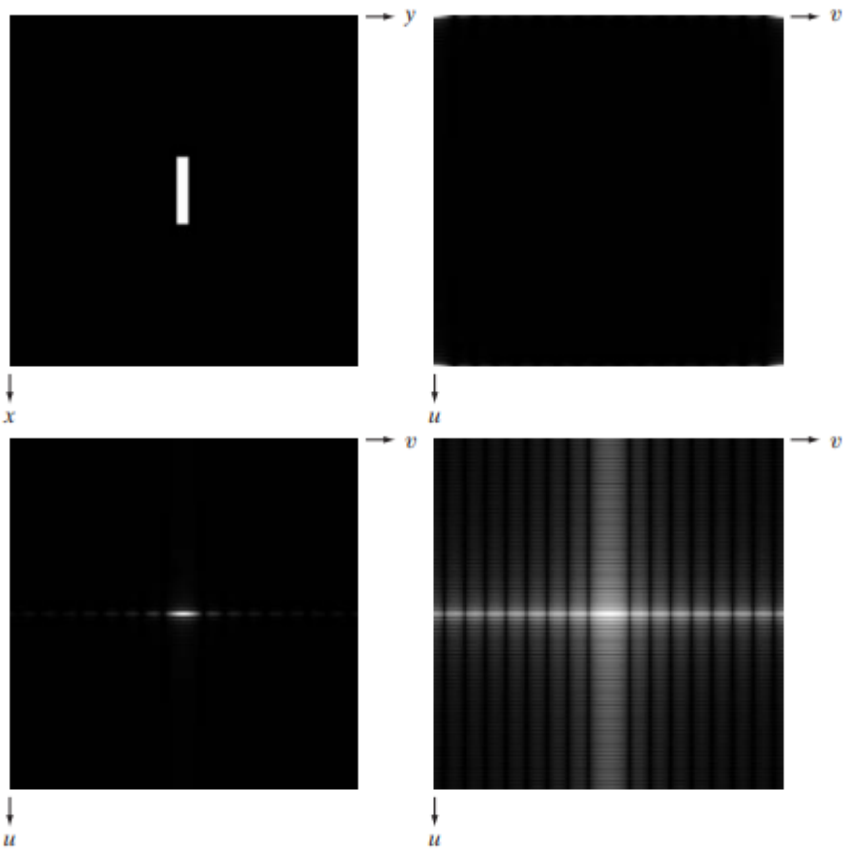
Using the polar coordinates

$$x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$$

results in the following transform pair:

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0) \qquad (4.6\text{-}5)$$

‘ Rotating f(x,y) by angle θ_o will rotate F(u,v) by the same angle, same goes for rotating F and its affect on f ’
Also remembering that cos^2(x) + sin^2(x) = 1.

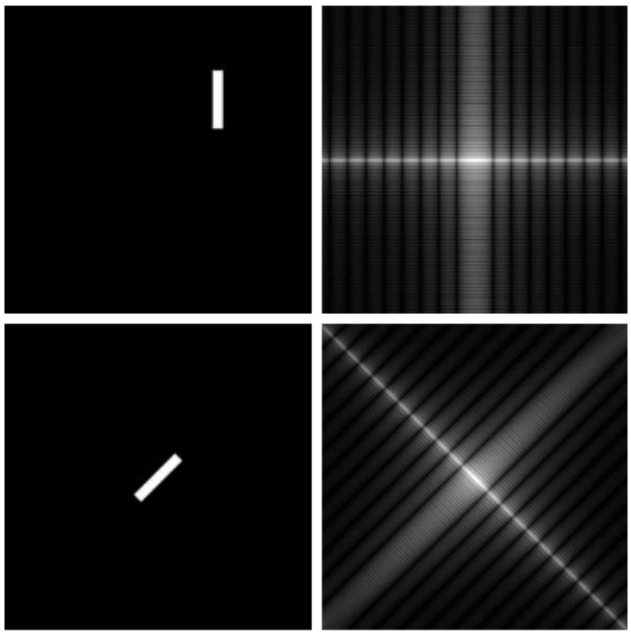


a b
c d

FIGURE 4.24
(a) Image.
(b) Spectrum showing bright spots in the four corners.
(c) Centered spectrum. (d) Result showing increased detail after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The coordinate convention used throughout the book places the origin of the spatial and frequency domains at the top left.

■ Figure 4.24(a) shows a simple image and Fig. 4.24(b) shows its spectrum, whose values were scaled to the range [0, 255] and displayed in image form. The origins of both the spatial and frequency domains are at the top left. Two things are apparent in Fig. 4.22(b). As expected, the area around the origin of the

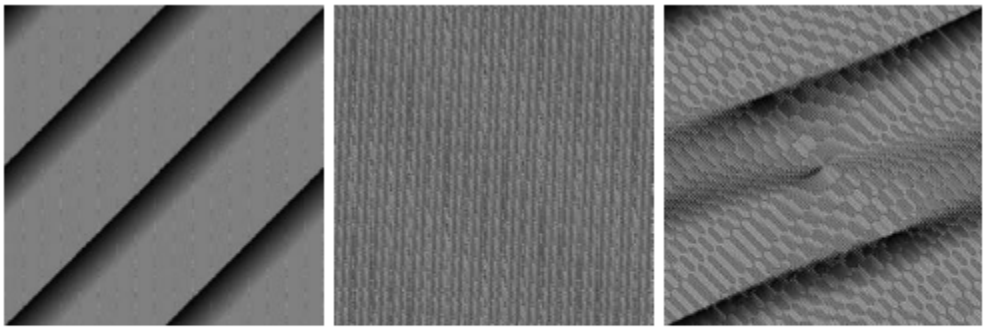
transform contains the highest values (and thus appears brighter in the image). However, note that the four corners of the spectrum contain similarly high values. The reason is the periodicity property discussed in the previous section. To center the spectrum, we simply multiply the image in (a) by $(-1)^{x+y}$ before computing the DFT, as indicated in Eq. (4.6-8). Figure 4.22(c) shows the result, which clearly is much easier to visualize (note the symmetry about the center point). Because the dc term dominates the values of the spectrum, the dynamic range of other intensities in the displayed image are compressed. To bring out those details, we perform a log transformation, as described in Section 3.2.2. Figure 4.24(d) shows the display of $(1 + \log|F(u, v)|)$. The increased rendition of detail is evident. Most spectra shown in this and subsequent chapters are scaled in this manner.



a b
c d

FIGURE 4.25
(a) The rectangle in Fig. 4.24(a) translated, and (b) the corresponding spectrum. (c) Rotated rectangle, and (d) the corresponding spectrum. The spectrum corresponding to the translated rectangle is identical to the spectrum corresponding to the original image in Fig. 4.24(a).

It follows from Eqs. (4.6-4) and (4.6-5) that the spectrum is insensitive to image translation (the absolute value of the exponential term is 1), but it rotates by the same angle of a rotated image. Figure 4.25 illustrates these properties. The spectrum in Fig. 4.25(b) is identical to the spectrum in Fig. 4.24(d). Clearly, the images in Figs. 4.24(a) and 4.25(a) are different, so if their Fourier spectra are the same then, based on Eq. (4.6-15), their phase angles must be different. Figure 4.26 confirms this. Figures 4.26(a) and (b) are the phase angle arrays (shown as images) of the DFTs of Figs. 4.24(a) and 4.25(a). Note the lack of similarity between the phase images, in spite of the fact that the only differences between their corresponding images is simple translation. In general, visual analysis of phase angle images yields little intuitive information. For instance, due to its 45° orientation, one would expect intuitively that the phase angle in



a b c

FIGURE 4.26 Phase angle array corresponding (a) to the image of the centered rectangle in Fig. 4.24(a), (b) to the translated image in Fig. 4.25(a), and (c) to the rotated image in Fig. 4.25(c).

Now the following paragraph is going over a more mathematical way to say the same thing I said earlier regarding the intuition of why/how the fourier transform is useful in measuring the prominence of certain frequencies in an image. It additionally explains why understanding the phase image is important.

The components of the spectrum of the DFT determine the amplitudes of the sinusoids that combine to form the resulting image. At any given frequency in the DFT of an image, a large amplitude implies a greater prominence of a sinusoid of that frequency in the image. Conversely, a small amplitude implies that less of that sinusoid is present in the image. Although, as Fig. 4.26 shows, the contribution of the phase components is less intuitive, it is just as important. The phase is a measure of displacement of the various sinusoids with respect to their origin. Thus, while the magnitude of the 2-D DFT is an array whose components determine the intensities in the image, the corresponding phase is an array of angles that carry much of the information about where discernable objects are located in the image. The following example clarifies these concepts further.

Take for example this image of a subject,

If we compute the phase image of this subject using the following equation

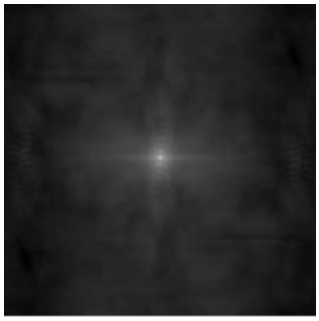
$$\phi = \tan^{-1}\frac{Im(Y)}{Re(Y)}$$

Where Y(x,y) is fourier transform output of pixel (x,y).
We get,

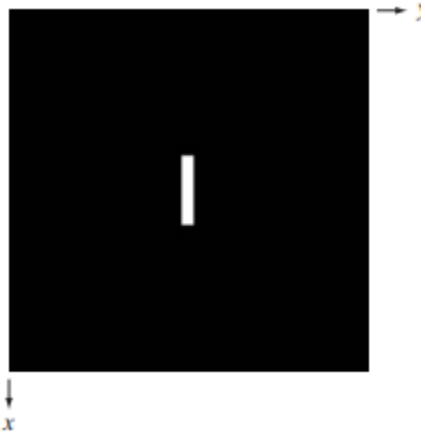
Which at first seems to be completely and utterly useless
It turns out that if we compute the inverse discrete fourier transform using just the phase image, we get back

(In practice computing the IDFT of just the phase angle corresponds to setting |F(u,v)| in the inverse formula equal to 1)

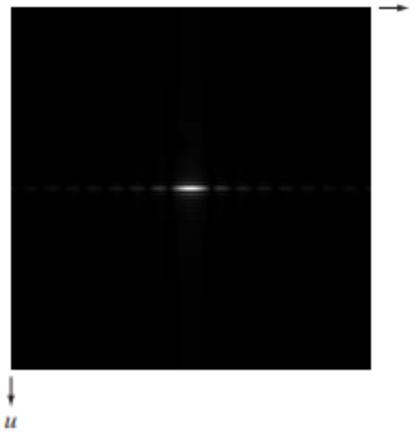
If we compute the magnitude of the fourier transform (i.e. set the exponential term equal to 1 => 0 phase angle), we get



Notice how the intensity here we can map to the above **IDFT on the phase** image and it would make sense as most of the higher intensity is where the face is in the original image. Additionally if we want to measure importance of phase compared to magnitude in computing the original image well the next image shows reconstruction using phase angle image seen above and magnitude corresponding to the following rectangle

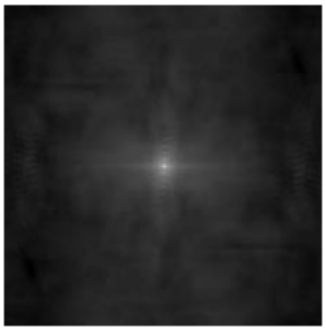


Namely **(After translating DC component to middle)**,

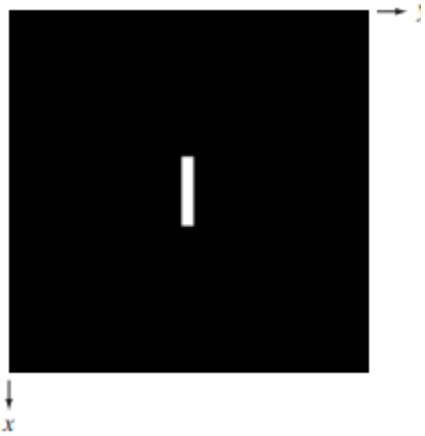


, resulting in..

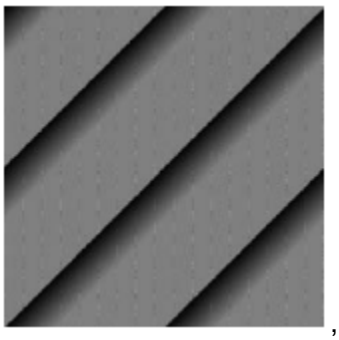
Meanwhile if we take the magnitude of the subject in the image photo,



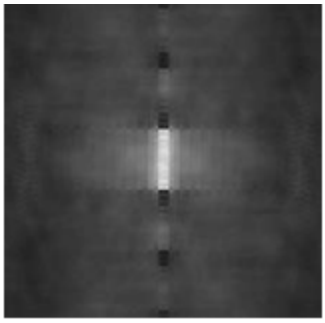
And combine it with the phase of



Namely,


,

We get,



Which is obviously a way worse reconstruction than

The 2D Convolution Theorem

Circular convolution is when we convolve 2 periodic signals, in a sense this is intuitive because we start where we began at a certain point along the domains inputs. The following is the circular convolution,

$$f(x, y) \star h(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n) \qquad (4.6\text{-}23)$$

, x and y in this case iterate through a single period in this sequence

The 2D Convolution Theorem

$$f(x, y) \star h(x, y) \Leftrightarrow F(u, v)H(u, v)$$

&

$$f(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v)$$

Note that we should again not confuse the double sided arrow with the mathematical notation for if and only if, rather it means that if we take the fourier transform of whatever the expression on the left side of the arrow is we will get whats on the right side, and similarly if we take the **inverse** fourier transform of whats on the right side of the arrow we get whats on the left. The main focus for now will be on the second of the above 3 mathematical expressions, relating how the inverse fourier transform of the product of two functions in the frequency domain return their convolution in the spatial domain.

If we want to end up coming the spatial convolution of two functions we need to take into account the **periodicity** problem mentioned earlier. Before we go into this though, the following is a reminder of how to prove the convolution theorem.

$$x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(\tau) x_2(t-\tau) d\tau$$

Now, from the definition of Fourier transform, we have,

$$X(\omega) = F[x_1(t) * x_2(t)] = \int_{-\infty}^{\infty} [x_1(t) * x_2(t)] e^{-j\omega t} dt$$

$$\Rightarrow F[x_1(t) * x_2(t)] = \int_{-\infty}^{\infty} [\int_{-\infty}^{\infty} x_1(\tau) x_2(t-\tau) d\tau] e^{-j\omega t} dt$$

By interchanging the order of integration, we get,

$$\Rightarrow F[x_1(t) * x_2(t)] = \int_{-\infty}^{\infty} x_1(\tau) [\int_{-\infty}^{\infty} x_2(t-\tau) e^{-j\omega t} dt] d\tau$$

By replacing $(t - \tau) = u$ in the second integration, we get,

$$t = (u + \tau) \text{ and } dt = du$$

$$\therefore F[x_1(t) * x_2(t)] = \int_{-\infty}^{\infty} x_1(\tau) [\int_{-\infty}^{\infty} x_2(u) e^{-j\omega(u+\tau)} du] d\tau$$

$$\Rightarrow F[x_1(t) * x_2(t)] = \int_{-\infty}^{\infty} x_1(\tau)[\int_{-\infty}^{\infty} x_2(u)e^{-j\omega u}du]e^{-j\omega\tau}d\tau$$

$$\Rightarrow F[x_1(t) * x_2(t)] = \int_{-\infty}^{\infty} x_1(\tau)X_2(\omega)e^{-j\omega\tau}d\tau.$$

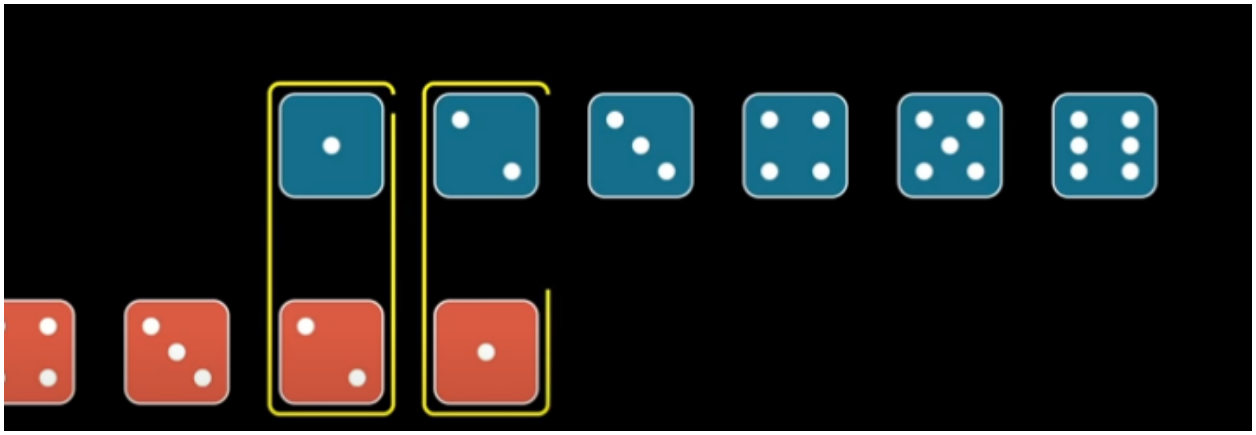
$$\therefore F[x_1(t) * x_2(t)] = X_1(\omega).X_2(\omega)$$

Therefore, the convolution between x1 and x2 is given by the inverse fourier transform of X1 times X2.

This 3b1b video is excellent and i think helps w/ the next part.

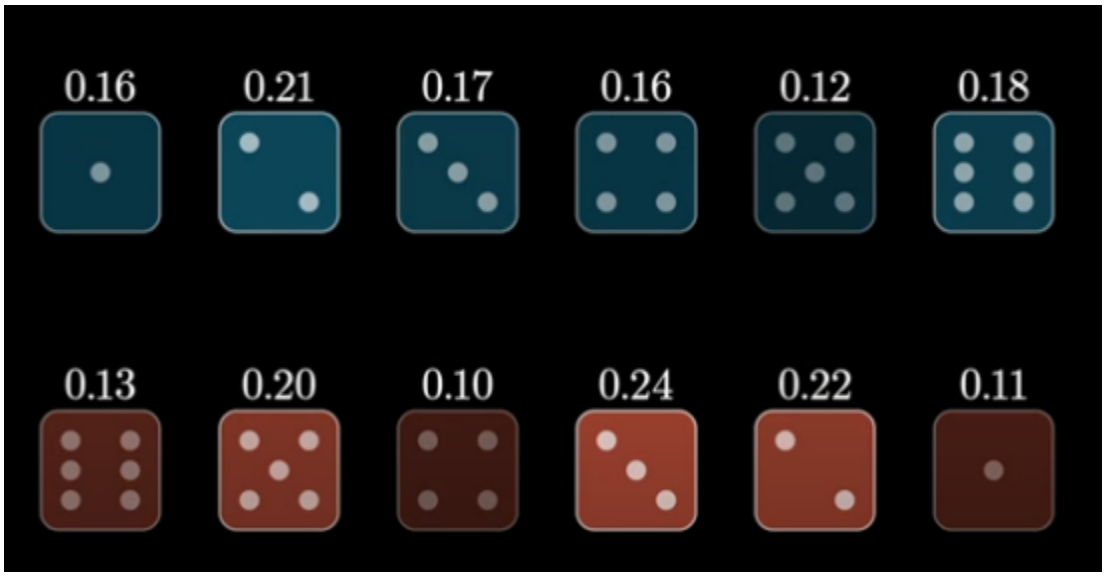
Explaining some parts of the 3b1b video

In the part regarding the addition of two random variables, he does not specify it but essentially the reason you flip (may or may not be obvious) on row of the dice before applying the offset -> multiplying -> adding is because when you do this flip, the total amount of number lost by the top row will be gained by the bottom row, resulting in the sum of each bottom and top part that is under the top row to all be equal in sum (number). **For example** if i give u {1,2,3,4,5,6} and I told you to take a copy of this set/array and figure out how to orient this copy such that when you add entry wise the original set/array and its copy you get back the same value in every index, you would get the copy {1,2,3,4,5,6} then flip it to get {6,5,4,3,2,1} then add it entry wise with the original (unflipped) version of it, and then the result would be {7,7,7,7,7,7}, this is the main idea behind the flipping before doing the dice number counting, because then for a given sum X of two numbers, every entry below the above array is going to correspond to that number X (this also is strictly increasing).

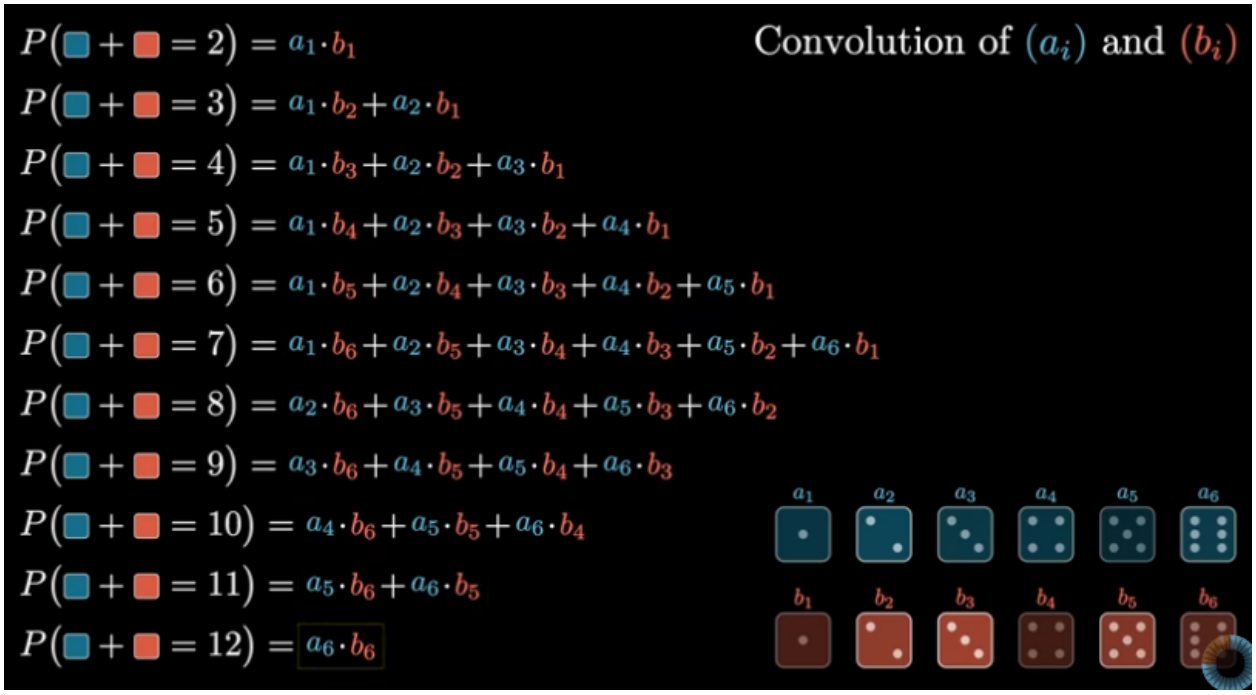


Notice how when you slide again, the 1 (**top**) gets placed with the 3, the two gets placed with the 2 and the 3 gets placed with the 1. Well this is the idea because only these dice will add up to the given sum being considered at the moment (**in the above visual the sum being**

considered at the moment is two). Another way to think about it is if we have a set minimum and maximum possible sum being considered (in the case of dice the minimum is 2 and the maximum is 12), we get the bottom row (**after flipping**) and slide it below the top row until the sum above and below is equal to the minimum (**then we know that sliding anymore will result in a sum above the minimum since the dice are strictly increasing towards the left on the orange side**), then we do the addition and repeat the process **(taking the next number as the new min)** until we hit the maximum which is 12. This **should exemplify** why we flip the bottom row first, because this allows for a strictly increasing portion on the bottom towards the left which then takes into account the strictly increasing portion on the top towards the right (**both obviously growing at same rate**).
The above is an example of a discrete convolution.
Now if you attach non uniform probabilities to every single potential dice number you get back

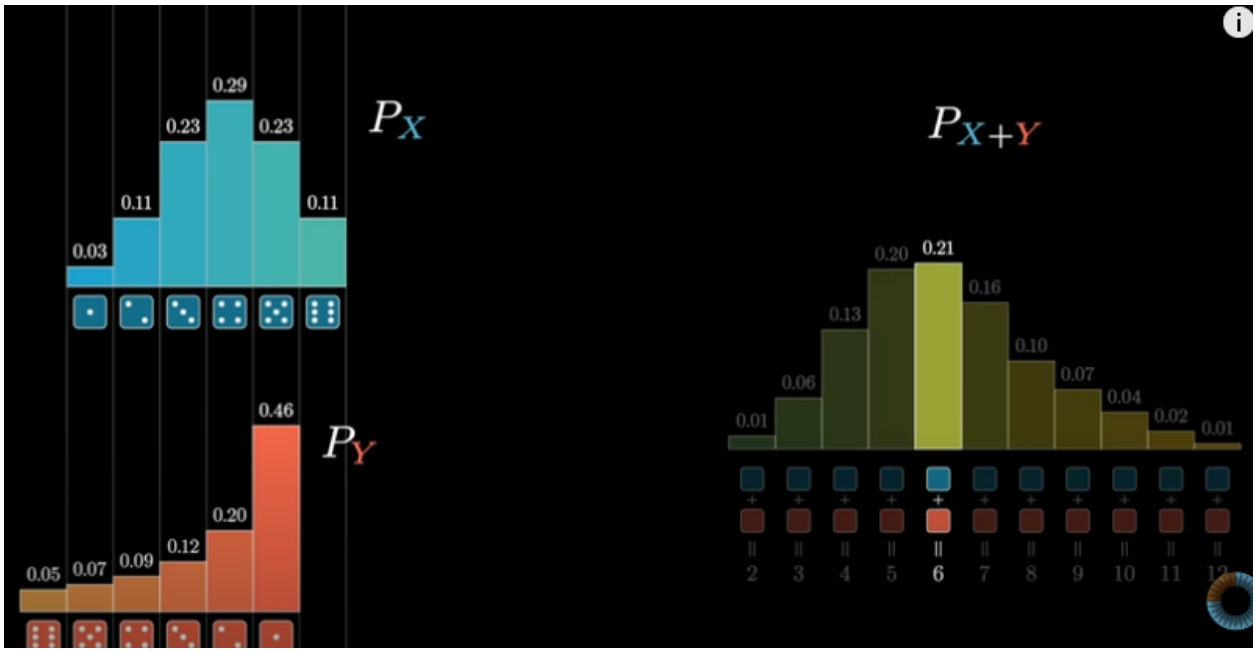


Then you repeat the same process as before but instead of considering each non uniform label before doing the multiplication. It makes sense that it results in this,

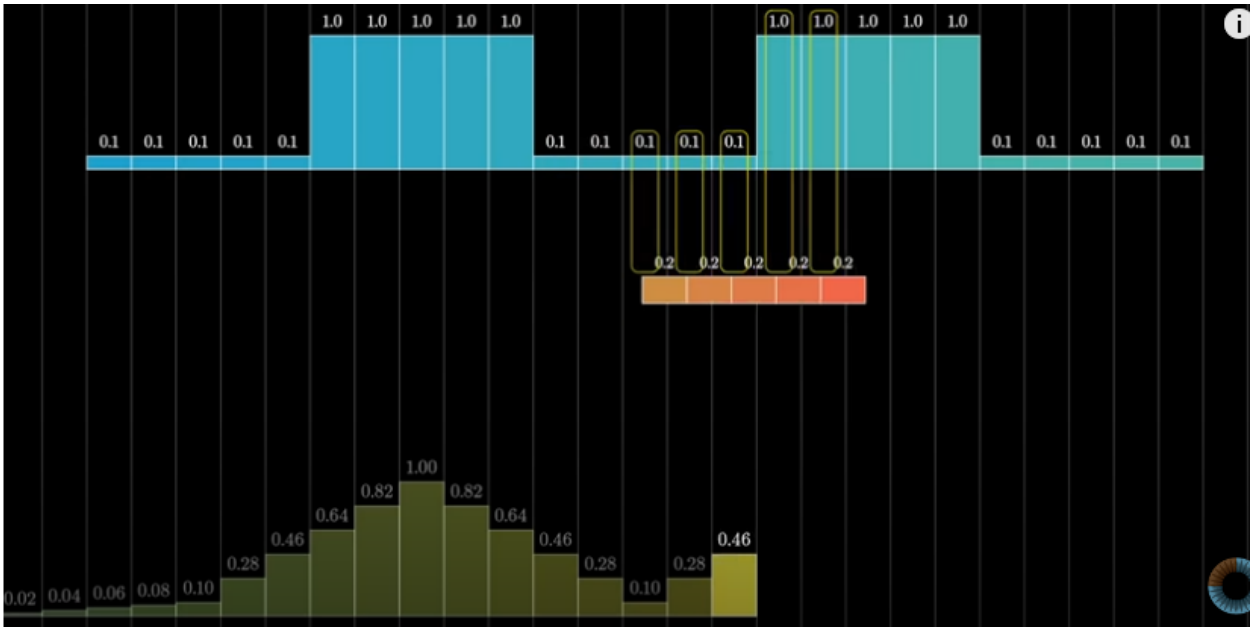


In this case since the indice corresponds to the actual value taken on by the dice, we can think of the convolution operation in this case being defined by

$$(a * b)_n = \sum_{\substack{i,j \\ i+j=n}} a_i \cdot b_j$$

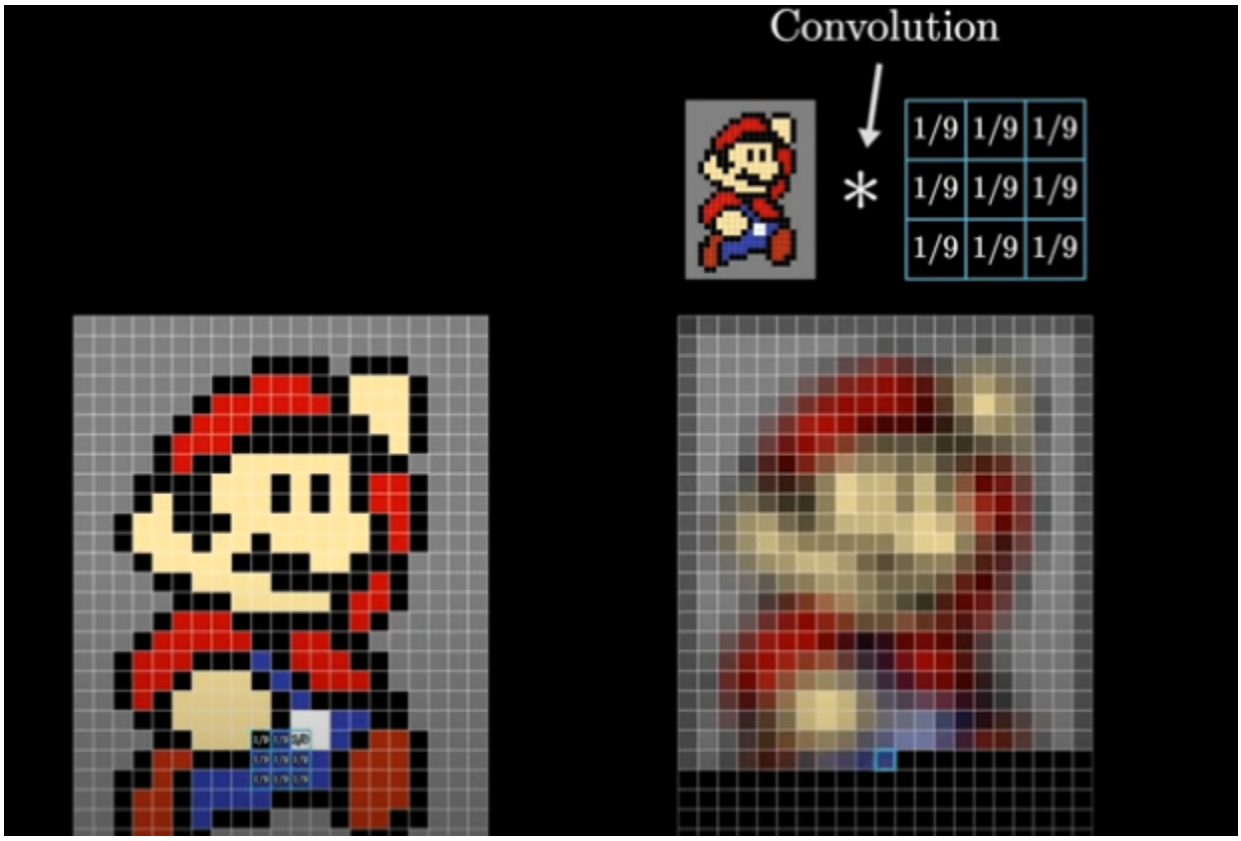


Another example is using a convolution to essentially do a moving average within a certain window size!



The above is a uniform moving average, but it can be non uniform as well (change values associated with each indice in the smaller orange list we slide across the blue values).

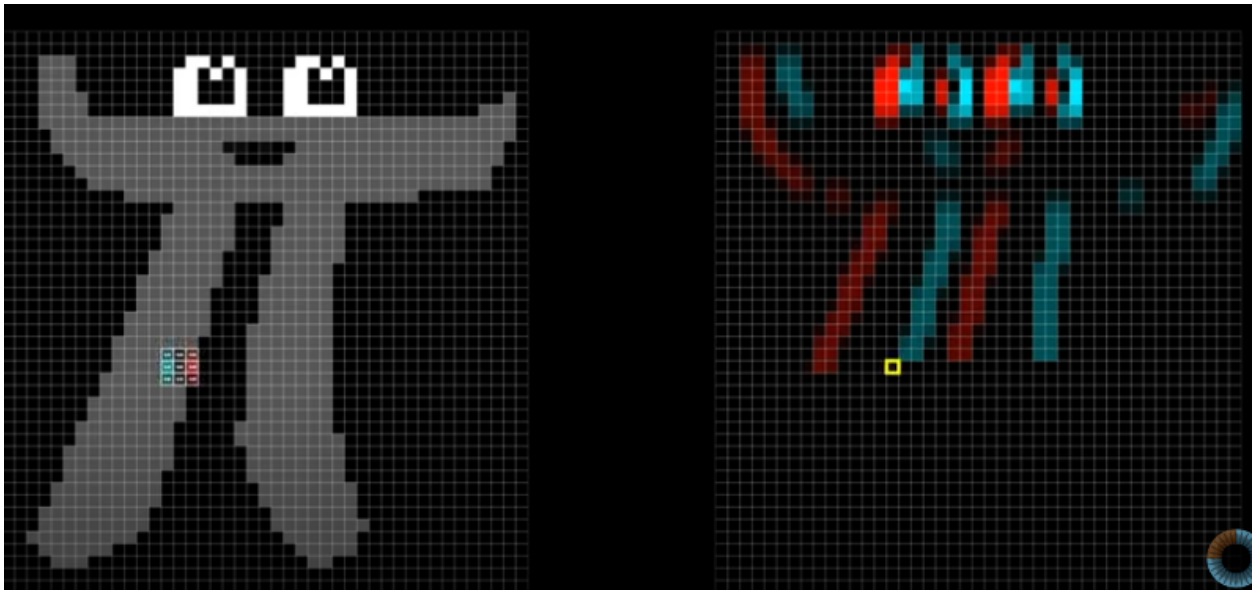
The convolution operation in 2 dimensions is a very simple extension of the 1d case, we literally just apply it as a square across a square rather than a line across a line essentially



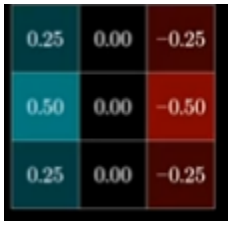
Note that to keep the mathematics precise (**according to definition of convolution**) before we convolve the right array with the mario image, we need to perform a 180 degree rotation on the array and then use that to convolve (in this case it obviously doesnt matter since the array is symmetric).

The video goes on to describe more about blurring in image processing, for more info on this I highly recommend checking out sources like wikipedia, and recommend books from richard szeliski on Computer Vision as well as rafael gonzales on digital image processing.

However this following visual is cool too hard to ignore, essentially what you are doing to the left with the convolutional kernel (what you call the 2d array you use to convolve) in this case is taking the 'first derivative' of the image which essentially marks the areas on the image which have the highest rate of change in pixel intensities (marks highest frequency portions of image).

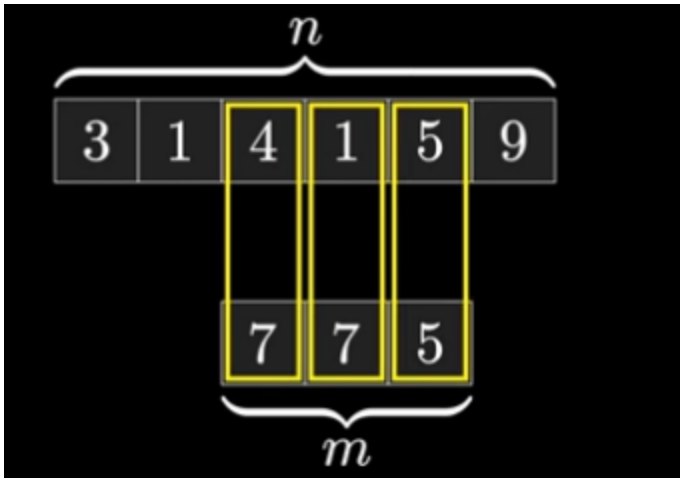


The convolutional kernel that is associated with this kind of transformation is given by something like



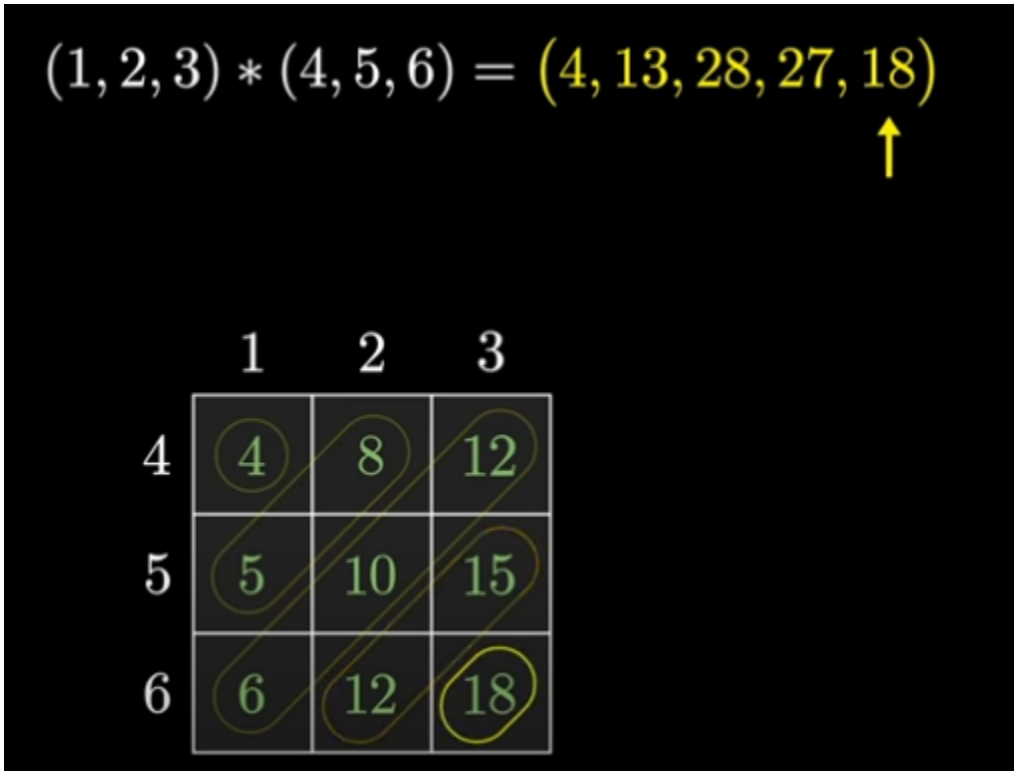
Note that you can really perform this with any a b c on left so long as the middle is full zero and the right is the negatives. Also this is more for vertical ‘edges’ noticeable in the original image meanwhile rotating this filter 90 degrees is more for horizontal edges.

Note that in pure mathematics convolution will produce array bigger than the two arrays you started with, but in thinks like convolutional nets often times convolutions are performed to downsize and ‘summarize’ the semantics of an image typically resulting in a smaller image (which can be achieved when we limit the convolution to only portions where the convolutional kernel is fully within the matrix it is convolving with respect to)

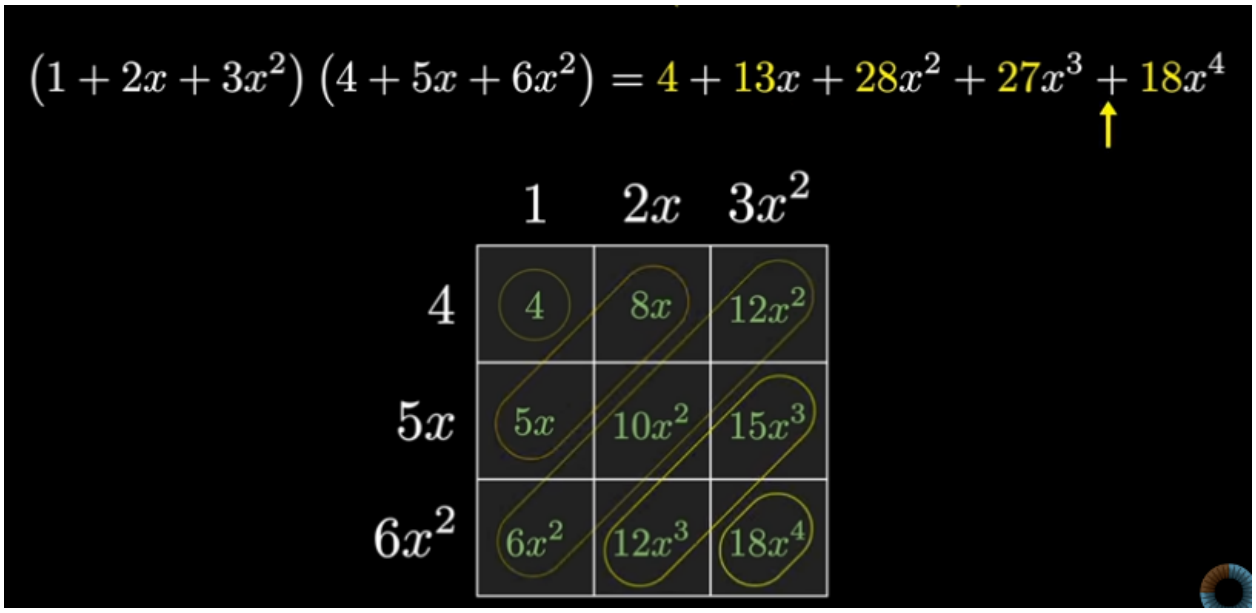


What i mean by that is in the above example stopping 5 after it hits 9 and performs its computation and starting the leftward 7 at 3 (**not starting with bottom right 5 at top left 3 or ending at bottom left 7 with top right 9**)

In computer science the flipping around the kernel portion of convolution is often ignored because we either are learning the filter anyway **(in something like the neural net)** or we are setting up the filter manually in such a way that we dont have to rotate it.



We can also think about polynomials like a convolution!



So in the above case we can see that a convolution will mimic the behavior of the multiplication of two polynomials.
A way you can see that the convolution operation does this is by focusing on the second expression

$$(4+5x+6x^2)$$

And the it multiplies the first term in this expression by the first term in the left side expression which is 1, then by 2x (**at the same time multiplying 5x by 1**), then if you notice this will be the same thing as distribution of each term (except rather than distributing 4 completely before distributing 5x, we distribute 4 once then 4 again with 5x then 4 again with 5x with 6x^2).
Another way of seeing this is that in the above visual that has the multiplication chart, is that when you go to the row denoted by 4 you will notice that in the first column of that row only the row 4 is within the illustrated **ellipse,** then you will notice when you go to the second column in the 4th row it is in the same ellipse as 5x for its first column, then on the third column of the first row youre in the 2nd column of 5x row and 1st column of 6x^2 row.

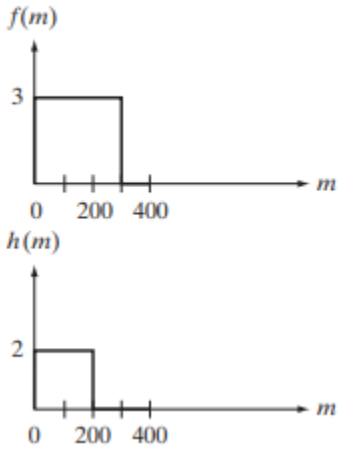
It turns out that the book by gonzales has a notation error here

$$f(x) \star h(x) = \sum_{m=0}^{399} f(x)h(x-m)$$

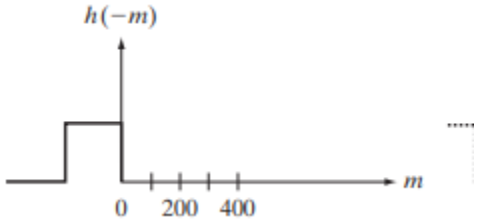
In reality the right side needs to have an f(m) term instead of an f(x) term.
Its important to note that convolution is indeed commutative
{a,b,c} * {x,y,z} = x*a,x*b+y*a,x*c+y*b+z*a,y*c+z*b,z*c
If we decide to do
{x,y,z}*{a,b,c} = x*a,x*b+y*a,x*c+y*b+z*a,y*c+z*b,z*c

Hence commutative (since the multiplication of two complex numbers is commutative). Note that multiplication is not always commutative which is why we need to check.a given set of matrices are not commutative under multiplication generally.

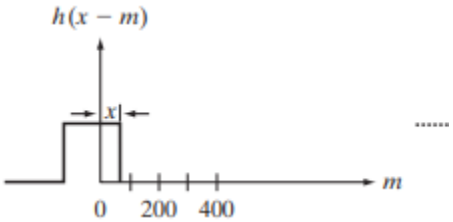
Graphical illustration of convolution
We first take two functions f and h in the domain in which we are summing over,



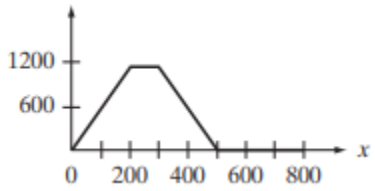
We then flip h (typically better to flip the one that is shorter in terms of input ranges)



Then we slide this flipped version an amount x

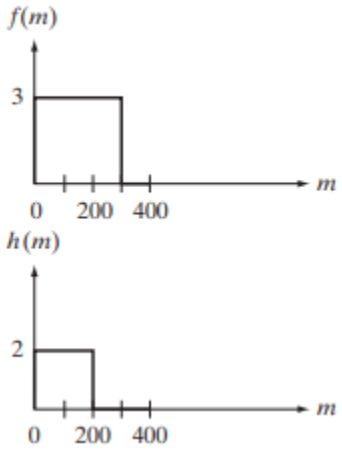


We then multiply the height of h from 0 to x with the height of f from 0 to x pointwise and then sum all of those. Then we keep sliding and summing until h is completely past f. This will result in a function that graphically looks like

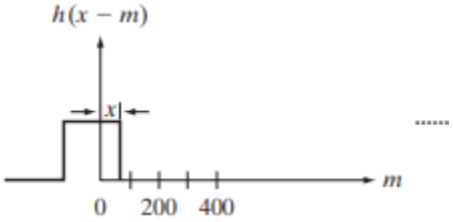


Now an important question to ask at this point is to consider the values that x needs to range over for this to completely work **(Hint: Look at where the bold line on the above graph with domain x starts and stops)**.

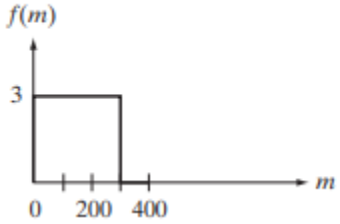
Well if we look at these two graphs we notice that the boldface line extends from 0 to 400.



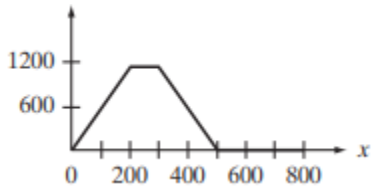
So, in this case we notice that when x hits 400 in the below graph



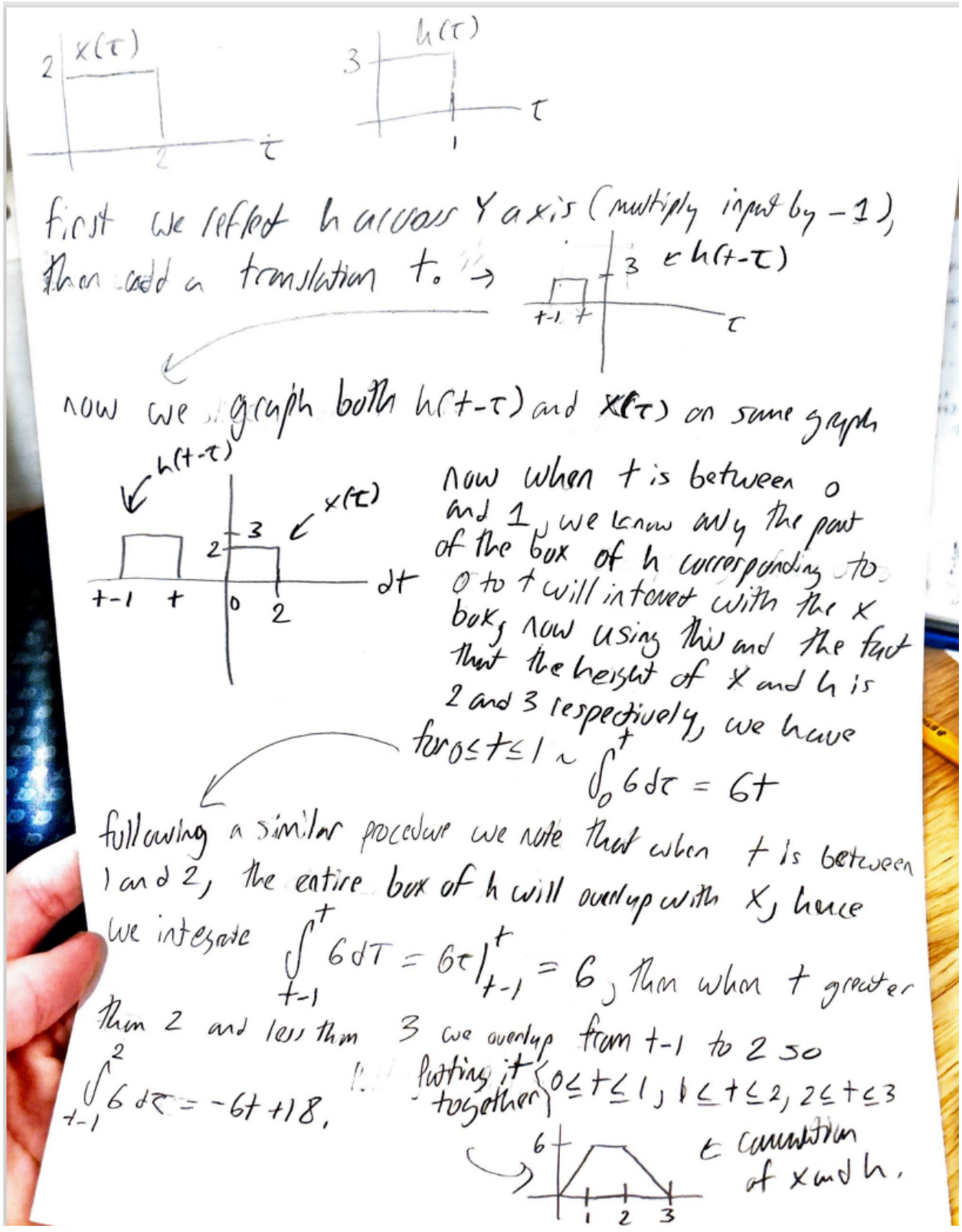
That it will have a 1-1 correspondence with



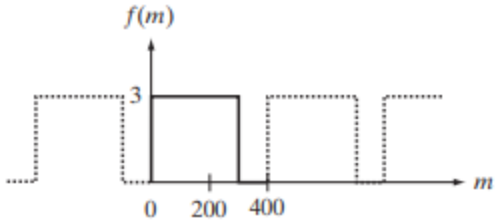
And at this point x has only travelled 400 points, for us to completely slide h(x-m) out of the way of f (such that its tail finally doesnt hit f anymore) we clearly then need to travel another 400 points. This is why in the final convolved function

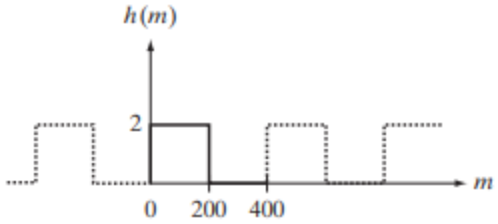


We have that the range of values x takes on is 0 to 799.
Additionally with our new intuition of sliding one function **entirely** across the other, it should be clear why the domain will be defined from 0 to n + m - 1 where n denotes number of defined inputs for f, and m denotes number of defined inputs for h.
For an exact example of performing a convolution on two signals, the following picture should help

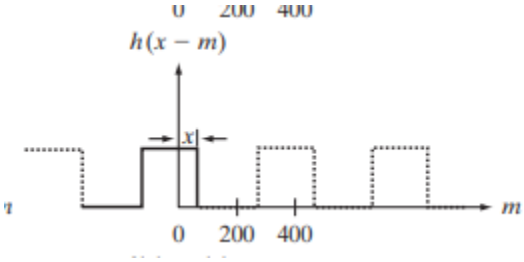


Now we need to go over the convolution of two periodic functions. Suppose we are working with the same functions as before except this time we extend it such that its periodic, then we can imagine f and h defined as

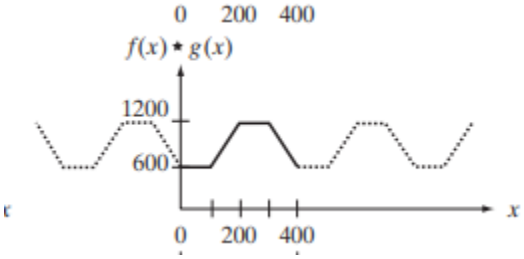




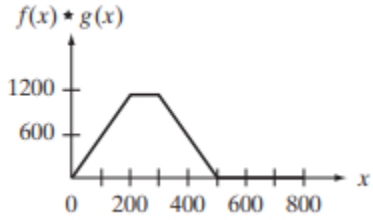
With h reflected and translated as



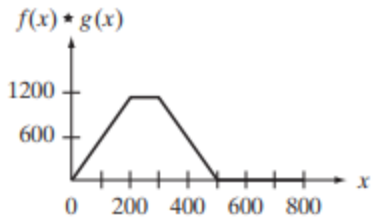
Then performing a convolution as we did before results in



Which is clearly a problem since we are trying to reconstruct a signal like



This problem is known as the **wraparound error**, the solution to this problem is given in a 1988 paper which essentially states that we need to append zeros to each function. This corresponds to adding zeros to both f and h, the amount of zeros we add to f is P - M where P is equal to **atleast** M + N -1 where M and N correspond to the number of samples in f and h respectively. Similarly the amount of zeros we add to h is P - N. We add these zeros to the trailing end of each function. Doing this then performing the convolution will result again in a periodic function however where each period has a signal with this shape,



Now the 2D Extension of this is tougher to visualize however regarding the padding error the following excerpt is a pretty decent description.

pending zeros to the functions. Let $f(x, y)$ and $h(x, y)$ be two image arrays of sizes $A \times B$ and $C \times D$ pixels, respectively. Wraparound error in their circular convolution can be avoided by padding these functions with zeros, as follows:

$$f_p(x, y) = \begin{cases} f(x, y) & 0 \le x \le A-1 \quad \text{and} \quad 0 \le y \le B-1 \\ 0 & A \le x \le P \quad \text{or} \quad B \le y \le Q \end{cases} \tag{4.6-27}$$

and

$$h_p(x, y) = \begin{cases} h(x, y) & 0 \le x \le C-1 \quad \text{and} \quad 0 \le y \le D-1 \\ 0 & C \le x \le P \quad \text{or} \quad D \le y \le Q \end{cases} \tag{4.6-28}$$

with

$$P \ge A + C - 1 \tag{4.6-29}$$

and

$$Q \ge B + D - 1 \tag{4.6-30}$$

The resulting padded images are of size $P \times Q$. If both arrays are of the same size, $M \times N$, then we require that

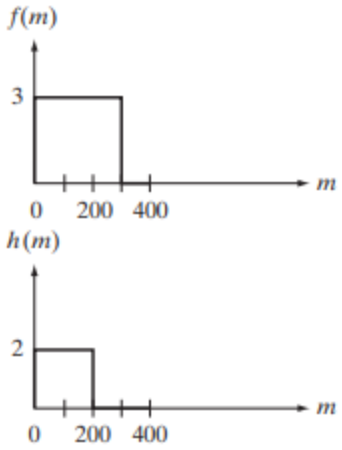
$$P \ge 2M - 1 \tag{4.6-31}$$

and

$$Q \ge 2N - 1 \tag{4.6-32}$$

In practice when computing discrete fourier transforms, they tend to be faster when arrays are of even size. Good idea to select P and Q s.t. They are even integers.

It should be noticed that the following functions that we have already seen



become zero before the end of the sampling interval, if this was not the case then a discontinuity would occur if we zero padded the function to stop the **wraparound error**. This excerpt gives a simple explanation on the way to view this (**remembering the infinite frequency components that comes from the fourier transform of a box function**).

appended to the function to eliminate wraparound error. This is analogous to multiplying a function by a box, which in the frequency domain would imply convolution of the original transform with a sinc function (see Example 4.1). This, in turn, would create so-called *frequency leakage*, caused by the high-frequency components of the sinc function. Leakage produces a blocky effect on images. Although leakage never can be totally eliminated, it can be reduced significantly by multiplying the sampled function by another function that tapers smoothly to near zero at both ends of the sampled record to dampen the sharp transitions (and thus the high frequency components) of the box. This approach, called *windowing* or *apodizing*, is an important consideration when fidelity in image reconstruction (as in high-definition graphics) is desired. If you are faced with the need for windowing, a good approach is to use a 2-D Gaussian function (see Section 4.8.3). One advantage of this function is that its Fourier transform is Gaussian also, thus producing low leakage.

Summary of DFT and Expressions

Name	Expression(s)
1) Discrete Fourier transform (DFT) of $f(x, y)$	$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$
2) Inverse discrete Fourier transform (IDFT) of $F(u, v)$	$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}$
3) Polar representation	$F(u, v) = \lvert F(u, v) \rvert e^{j\phi(u,v)}$
4) Spectrum	$\lvert F(u, v) \rvert = \left[R^2(u, v) + I^2(u, v) \right]^{1/2}$ $R = \text{Real}(F); \quad I = \text{Imag}(F)$
5) Phase angle	$\phi(u, v) = \tan^{-1}\left[\frac{I(u, v)}{R(u, v)} \right]$
6) Power spectrum	$P(u, v) = \lvert F(u, v) \rvert^2$
7) Average value	$\bar{f}(x, y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) = \frac{1}{MN} F(0, 0)$

Name	Expression(s)
8) Periodicity (k_1 and k_2 are integers)	$F(u, v) = F(u + k_1 M, v) = F(u, v + k_2 N)$ $= F(u + k_1 M, v + k_2 N)$ $f(x, y) = f(x + k_1 M, y) = f(x, y + k_2 N)$ $= f(x + k_1 M, y + k_2 N)$
9) Convolution	$f(x, y) \star h(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) h(x - m, y - n)$
10) Correlation	$f(x, y) \star h(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m, n) h(x + m, y + n)$
11) Separability	The 2-D DFT can be computed by computing 1-D DFT transforms along the rows (columns) of the image, followed by 1-D transforms along the columns (rows) of the result. See Section 4.11.1.
12) Obtaining the inverse Fourier transform using a forward transform algorithm.	$MNf^*(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M + vy/N)}$ This equation indicates that inputting $F^*(u, v)$ into an algorithm that computes the forward transform (right side of above equation) yields $MNf^*(x, y)$. Taking the complex conjugate and dividing by MN gives the desired inverse. See Section 4.11.2.

Name	DFT Pairs
1) Symmetry properties	See Table 4.1
2) Linearity	$af_1(x, y) + bf_2(x, y) \Leftrightarrow aF_1(u, v) + bF_2(u, v)$
3) Translation (general)	$f(x, y)e^{j2\pi(u_0x/M + v_0y/N)} \Leftrightarrow F(u - u_0, v - v_0)$ $f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-j2\pi(ux_0/M + vy_0/N)}$
4) Translation to center of the frequency rectangle, $(M/2, N/2)$	$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$ $f(x - M/2, y - N/2) \Leftrightarrow F(u, v)(-1)^{u+v}$
5) Rotation	$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$ $x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\varphi \quad v = \omega\sin\varphi$
6) Convolution theorem†	$f(x, y) \star h(x, y) \Leftrightarrow F(u, v)H(u, v)$ $f(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v)$

Name	DFT Pairs
7) Correlation theorem†	$f(x, y) \star h(x, y) \Leftrightarrow F^*(u, v)\,H(u, v)$ $f^*(x, y)h(x, y) \Leftrightarrow F(u, v) \star H(u, v)$
8) Discrete unit impulse	$\delta(x, y) \Leftrightarrow 1$
9) Rectangle	$\text{rect}[a, b] \Leftrightarrow ab\dfrac{\sin(\pi ua)}{(\pi ua)}\dfrac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
10) Sine	$\sin(2\pi u_0x + 2\pi v_0y) \Leftrightarrow$ $j\dfrac{1}{2}\left[\delta(u + Mu_0, v + Nv_0) - \delta(u - Mu_0, v - Nv_0)\right]$
11) Cosine	$\cos(2\pi u_0x + 2\pi v_0y) \Leftrightarrow$ $\dfrac{1}{2}\left[\delta(u + Mu_0, v + Nv_0) + \delta(u - Mu_0, v - Nv_0)\right]$

The following Fourier transform pairs are derivable only for continuous variables, denoted as before by t and z for spatial variables and by μ and ν for frequency variables. These results can be used for DFT work by sampling the continuous forms.

12) *Differentiation* (The expressions on the right assume that $f(\pm\infty, \pm\infty) = 0$.)	$\left(\dfrac{\partial}{\partial t}\right)^m\left(\dfrac{\partial}{\partial z}\right)^n f(t, z) \Leftrightarrow (j2\pi\mu)^m(j2\pi\nu)^nF(\mu, \nu)$ $\dfrac{\partial^m f(t, z)}{\partial t^m} \Leftrightarrow (j2\pi\mu)^mF(\mu, \nu); \dfrac{\partial^n f(t, z)}{\partial z^n} \Leftrightarrow (j2\pi\nu)^nF(\mu, \nu)$
13) *Gaussian*	$A2\pi\sigma^2e^{-2\pi^2\sigma^2(t^2+z^2)} \Leftrightarrow Ae^{-(\mu^2+\nu^2)/2\sigma^2}$ (A is a constant)