FOURIER METHODS

Teviet Creighton

Richard Price

Discrete Fourier transforms

So far we have dealt with *continuous* functions $x(t)$. In real life we cannot record data infinitely fast nor for an infinite amount of time: we must *sample* the function. The simplest method is to record the value of the function at regular intervals Δt. We assume that we have some number N of such samples:

$$x_k \;=\; x(t_k)\,, \qquad t_k \;=\; k\Delta t \;=\; 0,\ \Delta t,\ 2\Delta t,\ \ldots,\ (N-1)\Delta t\,.$$

We don't know the behaviour of $x(t)$ beyond this point, so we might as well assume that sample $N\Delta t$ is the same as sample 0, $(N+1)\Delta t$ is the same as sample Δt, and so on. That is, we treat it as *periodic* with period $T = N\Delta t$. Then we can write the function as a Fourier series:

$$x(t) \;=\; \sum_{n=-\infty}^{\infty} C_n\, e^{2\pi i n t/T}\,, \qquad C_n \;=\; \frac{1}{T}\int_0^T x(t)\, e^{-2\pi i n t/T}\, dt$$

But to compute C_n we only know $x(t)$ at discrete times t_k, so we must approximate the integral as a sum of discrete values:

$$C_n \;\approx\; \frac{1}{T}\sum_{k=0}^{N-1} x(t_k)\, e^{-2\pi i n t_k/T}\,\Delta t \;=\; \frac{\Delta t}{T}\sum_{k=0}^{N-1} x_k\, e^{-2\pi i n k\Delta t/T} \;=\; \frac{1}{N}\sum_{k=0}^{N-1} x_k\, e^{-2\pi i n k/N} \tag{1}$$

Conventionally, we leave off the factor of $1/N$, and define the **discrete Fourier transform** of x_k:

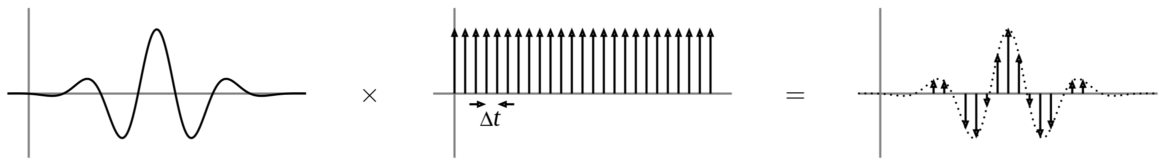
$$\boxed{X_n \;=\; \sum_{k=0}^{N-1} x_k\, e^{-2\pi i n k/N}} \tag{2}$$

Note that the N independent data x_k can give us only N independent Fourier coefficients X_n, since it is clear from equation (2) that $X_{n+N} = X_n$. That is, the Fourier coefficients X_n are *periodic* with period N, just like the original data x_k.

So now that we have the coefficients X_n and $C_n = \frac{1}{N}X_n$, we can plug back into the Fourier series $x(t) = \sum_{n=-\infty}^{\infty} C_n\, e^{2\pi i n t/T}$ and get $x(t)$ as a continuous function of time, right? *WRONG!* Let's try it:

$$\begin{aligned}
x(t) \;&=\; \sum_{n=-\infty}^{\infty} \frac{1}{N}\left(\sum_{k=0}^{N-1} x_k\, e^{-2\pi i n k/N}\right) e^{2\pi i n t/T} \;=\; \frac{1}{N}\sum_{k=0}^{N-1} x_k \sum_{n=-\infty}^{\infty} e^{2\pi i n (t/T - k/N)} \\
&=\; \frac{1}{N}\sum_{k=0}^{N-1} x_k\, \delta\!\left(\frac{t}{T} - \frac{k}{N}\right) \;=\; \frac{1}{N}\sum_{k=0}^{N-1} x_k\, T\, \delta\!\left(t - \frac{kT}{N}\right) \\
&=\; \Delta t \sum_{k=0}^{N-1} x_k\, \delta(t - t_k)
\end{aligned}$$

Instead of a continuous function $x(t)$ we get a series of δ-functions at times t_k weighted by x_k. By approximating the integral for C_n as a sum, we've *thrown away* any information about $x(t)$ between samples. Another way of looking at it is, if we *were* actually to multiply $x(t)$ by the series $\sum_{k=0}^{N-1} \Delta t\, \delta(t - t_k)$ (called a *Dirac comb*), then the "approximation" in equation (1) would be exact. In other words:



Function $x(t)$ × Dirac comb = Sampled function x_k

Inverse discrete Fourier transform

Is there any way to get back just the original samples x_k given the Fourier components X_n? When we plug the coefficients back into the Fourier series, we get back not $x(t_k)$, but $x(t)$ multiplied by a Dirac comb. But remember that only N of the coefficients X_n are independent. What happens if we write the Fourier series as just a sum over those n, rather than an infinite series? We get:

$$\begin{aligned} x(t_k) \overset{?}{=} \frac{1}{N}\sum_{n=0}^{N-1} X_n\, e^{2\pi i n t_k/T} &= \frac{1}{N}\sum_{n=0}^{N-1}\left(\sum_{k'=0}^{N-1} x_{k'}\, e^{-2\pi i n k'/N}\right) e^{2\pi i n k/N} \\ \frac{1}{N}\sum_{n=0}^{N-1} X_n\, e^{2\pi i n k/N} &= \frac{1}{N}\sum_{k'=0}^{N-1} x_{k'} \sum_{n=0}^{N-1} e^{2\pi i n(k-k')/N} \end{aligned}$$

Without going into the details, you should be able to convince yourself that when $k \neq k'$, the last sum will be zero: it is the sum of N evenly-spaced samples over $(k-k')$ periods of a sinusoidal function. When $k = k'$, we have $e^0 = 1$, and the last sum is just N. Thus:

$$\begin{aligned} \frac{1}{N}\sum_{n=0}^{N-1} X_n\, e^{2\pi i n k/N} &= \frac{1}{N}\sum_{k=0}^{N-1} x_{k'} \left\{ \begin{array}{cc} 0 & k \neq k' \\ N & k = k' \end{array} \right. \\ &= \frac{1}{N} N x_k \qquad \checkmark \end{aligned}$$

Thus we can define the **inverse discrete Fourier transform** of X_n:

$$\boxed{x_k = \frac{1}{N}\sum_{n=0}^{N-1} X_n\, e^{2\pi i n k/N}} \tag{3}$$

which has the same form as the discrete Fourier transform except for the sign in the exponential and the factor 1/N.

Discrete vs. continuous Fourier transform

In the same way that x_k are samples of $x(t)$ at discrete times $t_k = k\Delta t$, the Fourier components X_n are samples of $\tilde{x}$ at discrete frequencies $f_n = n\Delta f$. But what is the frequency spacing? Using the definituon of the **continuous Fourier transform** of $x(t)$, which I denote:

$$\boxed{\tilde{x}(f) = \int_{-\infty}^{\infty} x(t)\, e^{-2\pi i f t}\, dt} \tag{4}$$

we now just look at the argument of the exponential:

$$2\pi i n k/N = 2\pi i n k \Delta t/(N\Delta t) = 2\pi i n t_k/T = 2\pi i f_n t_k \qquad \text{provided} \qquad f_n = n/T$$

Thus:

$$\boxed{t_k = k\Delta t = kT/N \quad \Leftrightarrow \quad f_n = n\Delta f = n/T} \tag{5}$$

So are the coefficients X_n just the sampled values of $\tilde{x}(f_n)$? Not quite. Note that if $x_k = x(t_k)$ has some dimension such as length, then X_n has the same dimensions, whereas $\tilde{x}(f)$ has dimensions of length×time. To get the actual relationship between X_n and $\tilde{x}(f)$, approximate equation (4) as a sum:

$$\tilde{x}(f_n) \approx \sum_{k=0}^{N-1} x_k\, e^{-2\pi i f_n t_k}\, \Delta t = \sum_{k=0}^{N-1} x_k\, e^{-2ink/N}\, \Delta t$$

Thus:

$$\boxed{x(t_k) = x_k \quad \Leftrightarrow \quad \tilde{x}(f_n) = X_n \Delta t} \tag{6}$$

Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) is a particular computational algorithm for computing the discrete Fourier transform (DFT). It computes all N coefficients X_n at once, and much more efficiently than computing each X_n individually.

If you were to compute the coefficients individually, then each one would require a sum over N values of x_k. This would have to be done separately for each of the N values of X_n. The total number of operations would be of order $\sim N^2$. With the FFT algorithm, the computation is organized to that we get all the coefficients X_n at once with only $\sim N \log(N)$ operations. (The exact prefactors depend on the particular implementation of the algorithm.) Since $\log(N)$ grows very slowly as N increases, this represents a huge savings for large N. The popularization of the FFT algorithm by Cooley and Tukey in 1965 was largely responsible for digital signal processing being the powerful tool that it is today.

The FFT algorithm works best (has the smallest prefactors) when the (large) number of data N is the product of many smaller factors, particularly if N is a power of 2. Many FFT computer codes in use today can handle any value of N; nonetheless, when using FFTs it is common to work with chunks of data of length $N = 2^p$, to maximize efficiency.

Almost all computer implementations of the discrete Fourier transform use some variant of the FFT algorithm. The term "FFT" has become essentially synonymous with "discrete Fourier transform".

Understanding DFTs

Lets begin with some remarks about Fourier transforms in general. They take a function $x(t)$ and return a function $\tilde{x}(f)$. Both functions hold the same information about the underlying process, but present it in different ways. We sometimes call them the *time domain* and *frequency domain* representations of the process. Any feature in $x(t)$ will have a corresponding feature in $\tilde{x}(f)$, and vice-versa. Qualitatively, features in $x(t)$ and $\tilde{x}(f)$ are related by the following rule:

Features of $x(t)$ with short timescales will show up as features in $\tilde{x}(f)$ spread over a wide range in frequency. Features in $\tilde{x}(f)$ that have a narrow frequency range will show up features in $x(t)$ with long timescales.

This qualitative rule can be illustrated quantitatively with two functions whose Fourier transforms look identical to the original functions, just with a rescaling of the dependent variable.

1. The Gaussian function

Consider a time-domain function that is a normalized Gaussian centred on $t = 0$ with standard deviation σ_t:

$$x(t) \quad = \quad \frac{1}{\sqrt{2\pi\sigma_t^2}}\, e^{-t^2/2\sigma_t^2}$$

The Fourier transform can be computed directly:

$$\begin{aligned}
\tilde{x}(f) \quad &= \quad \int_{-\infty}^{\infty} x(t)\, e^{-2\pi i f t}\, dt \quad = \quad \frac{1}{\sqrt{2\pi\sigma_t^2}} \int_{-\infty}^{\infty} e^{-t^2/2\sigma^2}\, e^{-2\pi i f t}\, dt \\
&= \quad \frac{1}{\sqrt{2\pi\sigma_t^2}} \int_{-\infty}^{\infty} e^{-t^2/2\sigma_t^2 \,-\, 2\pi i f t \,+\, 2\pi^2 f^2 \sigma_t^2}\, e^{-2\pi^2 f^2 \sigma_t^2}\, dt \\
&= \quad e^{-2\pi^2 f^2 \sigma_t^2}\, \frac{1}{\sqrt{2\pi\sigma_t^2}} \int_{-\infty}^{\infty} e^{-(t+2\pi i f \sigma_t^2)^2/2\sigma_t^2}\, dt
\end{aligned}$$

But the last integral is just the integral of a Gaussian function centred on $-2\pi i f \sigma_t^2$ with standard deviation σ_t, which should just cancel the normalization factor:

$$\tilde{x}(f) \quad = \quad e^{-2\pi^2 f^2 \sigma_t^2} \quad = \quad e^{-f^2/2\sigma_f^2} \qquad \text{where} \qquad \sigma_f \quad = \quad \frac{1}{2\pi\sigma_t} \tag{7}$$

So the Fourier transform of a Gaussian in time is a Gaussian in frequency, where the width in frequency is inversely proportional to the width in time. That is, a short-duration Gaussian is a wide-bandwidth Gaussian, and a long-duration Gaussian is a narrow-bandwidth Gaussian, in keeping with the qualitative rule above.

2. The Dirac comb

Formally, the Dirac δ "function" is not a true function (its value is undefined when its argument is zero), but is a useful conceptual tool for understanding Fourier transforms and other integral transforms. Consider a "Dirac comb", an infinite series of δ-functions in time with spacing T:

$$x(t) \quad = \quad \sum_{k=-\infty}^{\infty} \delta(t - kT)$$

This is manifestly periodic, so we write it as a Fourier series $x(t) = \sum_{n=-\infty}^{\infty} C_n\, e^{2\pi i n t/T}$ with Fourier coefficients:

$$C_n \quad = \quad \frac{1}{T}\int_{-T/2}^{T/2} x(t)\, e^{-2\pi i n t/T}\, dt \quad = \quad \frac{1}{T}\int_{-T/2}^{T/2} \delta(0)\, e^{-2\pi i n t/T}\, dt \quad = \quad \frac{1}{T}\, e^{-2\pi i n \cdot 0/T} \quad = \quad \frac{1}{T}$$

We'll use a trick to compute the Fourier transform of $x(t)$: we'll take its Fourier series and manipulate it so that it has the form of an inverse Fourier transform, and whatever function appears inside that inverse transform must then be $\tilde{x}(f)$. So:

$$\begin{aligned} x(t) &= \sum_{n=-\infty}^{\infty} \frac{1}{T} e^{2\pi i n t/T} = \sum_{n=-\infty}^{\infty} \frac{1}{T} \int_{-\infty}^{\infty} e^{2\pi i f t}\, \delta\big(f - \tfrac{n}{T}\big)\, df \\ &= \int_{-\infty}^{\infty} e^{2\pi i f t} \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta\big(f - \tfrac{n}{T}\big)\, df = \int_{-\infty}^{\infty} e^{2\pi i f t}\, \tilde{x}(f)\, df \end{aligned} \tag{8}$$

where $\tilde{x}(f) = \frac{1}{T}\sum_{n=-\infty}^{\infty} \delta(f - \frac{n}{T})$ is a Dirac comb in frequency with spacing $1/T$. Thus a wide spacing in time gives a narrow spacing in frequency, and vice versa, in keeping with the rule of thumb.

Periodicity and sampling

This behaviour of the Dirac comb, in combination with the convolution theorem, helps us to understand the relationship between the continuous Fourier transform and the Fourier series in a pictorial rather than a mathematical way.

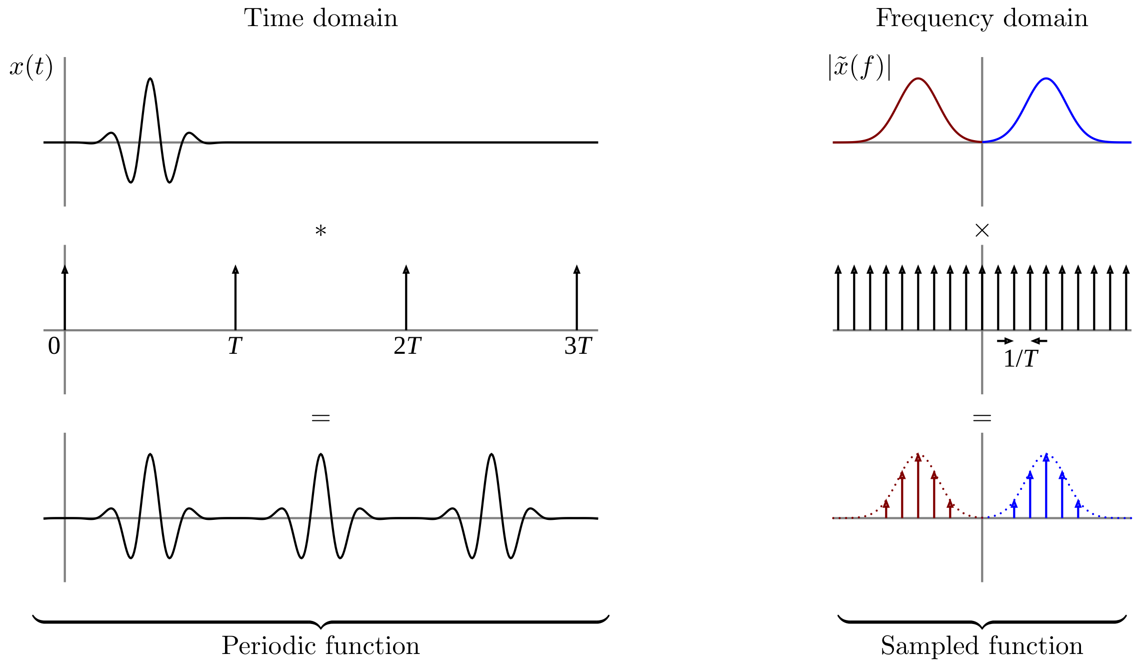
The Fourier series presumes that the function $x(t)$ is periodic: if it is not periodic then the Fourier series expansion *makes* it so. On the other hand, the Fourier transform presumes that $x(t)$ is finitely integrable. Let us suppose $x(t)$ spans a finite time and has a Fourier transform $\tilde{x}(f)$. We can *impose* periodicity with period T by *convolving* $x(t)$ with a Dirac comb with spacing T:

$$y(t) = x(t) * \sum_k \delta(t - kT) = \int_{-\infty}^{\infty} x(\tau) \sum_k \delta(t - \tau - kT)\, d\tau$$

since, assuming $x(t)$ is restricted to the domain $t \in (0, T)$, we have:

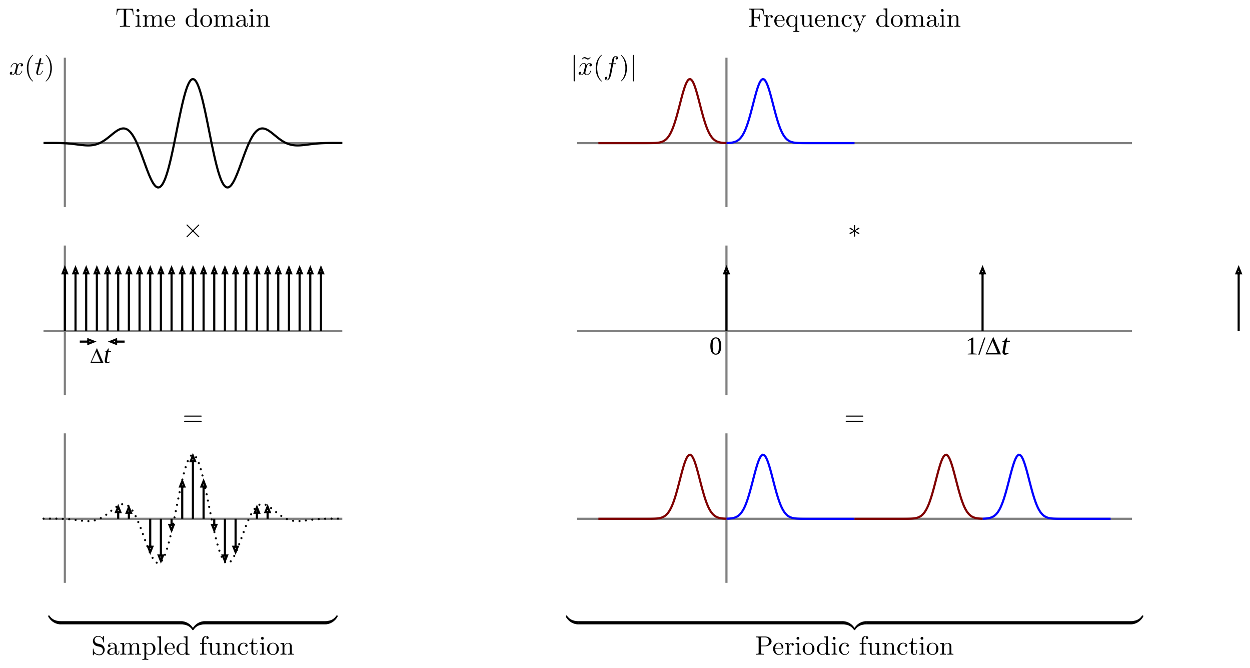
$$y(t + nT) = \int_0^T x(\tau) \sum_k \delta(t - \tau + [n-k]T)\, d\tau = \int_0^T x(\tau)\, \delta(t - \tau)\, d\tau = x(t)$$

By the convolution theorem, the Fourier transform $\tilde{y}(f)$ of this periodic function will be $\tilde{x}(f)$ *multiplied* by the Fourier transform of the Dirac comb, which is just a Dirac comb in frequency with spacing $1/T$. This can be represented pictorially:



That is, *periodicity in time* leads to *sampling in frequency.* The Fourier series coefficients C_n give *complete* information about the function in the frequency domain, because the imposition of periodicity nullifies the Fourier transform except at discrete frequency samples $f_n = n/T$.

Similarly, if the function is *sampled in time*, then it must be *periodic in frequency.* We illustrate this pictorially below:

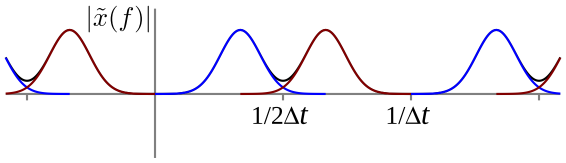


When discussing the discrete Fourier transform, we had mentioned that the Fourier coefficients X_n had the periodicity relationship $X_{N+n} = X_n$. Here we see that this periodicity in frequency also applies to *continuous* Fourier transforms of *sampled* time series. A discrete Fourier transform is sampled in time *and* is treated as periodic in time; thus it must be periodic *and* sampled in frequency as well. To summarize:

Periodic in time	$\Leftrightarrow$	Sampled in frequency
Sampled in time	$\Leftrightarrow$	Periodic in frequency

Aliasing and Nyquist sampling

If a continuous function $x(t)$ is sampled at time intervals Δt, then its Fourier transform $\tilde{x}(f)$ is forced to be periodic in frequency with period $1/\Delta t$. This leads to some odd behaviour if the original $\tilde{x}(f)$ extended beyond $|f| = 1/2\Delta t$. Then the *negative* frequency components of subsequent periods start to overlap with the *positive* frequency components, as illistrated below:



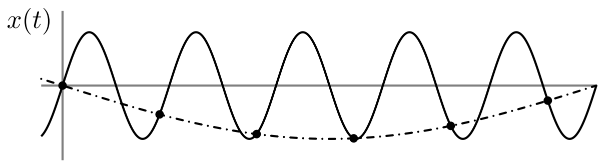
This overlapping of high-frequency components is called **aliasing**. The frequency $\frac{1}{2\Delta t} = \frac{1}{2} \times$ sampling rate at which it occurs is known as the **Nyquist frequency**.

Aliasing leads to a loss of information about the original data, because one cannot readily separate the positive-frequency contributions with $|f| < \frac{1}{2\Delta t}$ from the negative-frequency contributions with $|f| > \frac{1}{2\Delta t}$. To avoid this ambiguity, one must be sure to sample the original function $x(t)$ at a sampling rate that is at least twice the highest frequency present in the data. This minimum sampling rate is known (somewhat confusingly) as the **Nyquist rate**. Thus:

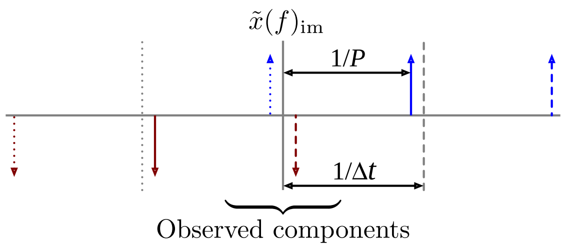
$$\boxed{\begin{array}{lcl} \text{Nyquist frequency} & \equiv & \frac{1}{2} \times \text{sample rate} \\ \text{Nyquist rate} & \equiv & 2 \times \text{highest frequency} \end{array}}$$

In other words, to ensure that the highest frequency is less than the Nyquist frequency, you must sample the function faster than the Nyquist rate.

This may all look like some weird feature of the discrete Fourier transform, but it is actually a fundamental effect of sampling. If you don't sample fast enough, you start to miss half-cycles of the higher frequency components, making them appear as if they were lower frequency. To illustrate, consider sampling a sine wave with a sampling interval Δt slightly less than its true period P:



Since you can't resolve the half-cycles of the wave that lie between the samples, you cannot distinguish between the "true" high-frequency signal and the low-frequency interpolation (dot-dashed line). In the frequency domain, we see not the high-frequency Fourier components of the original sinusoid (solid lines) but the aliased copies of these components from earlier and later periods (dotted and dashed lines):



Avoiding aliasing

There are two parts to avoiding aliasing effects:

1. Use a physical filter to remove high-frequency components before sampling.
2. Sample at a rate greater than twice the highest freqency present in the signal.

Sometimes this is done in stages. The physical process is physically filtered to some degree, and then sampled at a high sampling rate. Then a *digital filter* is used on the data to extract the frequency band of interest, and the result is then *downsampled* (resampled at a lower rate).

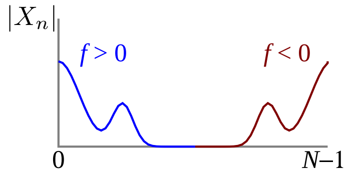
Data storage and indexing of FFTs

Most FFT algorithms take an array of data $\{x_k\}$, $k = 0, \ldots, N-1$, and return an array of Fourier components $\{X_n\}$, $n = 0, \ldots, N-1$. Some data analysis programs, such as MATLAB, use indecies running from $1, \ldots, N$ rather than $0, \ldots, N-1$. You need to remember that the *first* element of the array still stores the *zero-frequency* component, and so on. That is:

$$\boxed{\begin{array}{ll} \text{If the index } n = 1, \ldots, N, & \text{then the } n^{\text{th}} \text{ Fourier coefficient is at frequency } f_n = \frac{n-1}{T}. \\ \text{If the index } n = 0, \ldots, N-1, & \text{then the } n^{\text{th}} \text{ Fourier coefficient is at frequency } f_n = \frac{n}{T}. \end{array}}$$

Our convention so far has been $0, \ldots, N-1$, and we will stick with this, as it is simpler and more common.

If the signal was sampled above the Nyquist rate, then all the positive-frequency components should appear at $n < N/2$. What about the negative components? Because of periodicity, the component X_{-n} will be normally be stored as X_{N-n}. So, if you simply plot the FFT of a function, you will normally see something like what is shown to the right.



Data storage and indexing of real FFTs

We noted earlier that, given N data $x_0, \ldots, x_{N-1}$, we get N independent Fourier components $X_0, \ldots, X_{N-1}$, where $X_{n+N} = X_n$. Each X_n will typically be a complex number. In general, the x_k may also be complex. So each set of data contains $2N$ independent real numbers (the real and imaginary parts of each sample).

Now suppose, as is often the case, that x_k are all *real*, so there are only N independent real numbers. The X_n are still complex, but their components are no longer independent. Specifically, using * to represent complex conjugation, and noting that $x_k^* = x_k$ for real x_k, we have:

$$X_{-n} = \sum_{k=0}^{N-1} x_k \, e^{-2\pi i(-n)k/N} = \sum_{k=0}^{N-1} x_k \, e^{2\pi ink/N} = \sum_{k=0}^{N-1} x_k^* \, e^{2\pi ink/N} = X_n^*$$

or, since X_n is periodic, $\boxed{X_{N-n} = X_n^*}$

Note in particular that $X_0 = X_{-0} = X_0^*$, so X_0 is real. Also, if N is *even*, $X_{N/2} = X_{N-(N/2)} = X_{N/2}^*$, so $X_{N/2}$ is also real. So we have $\frac{N}{2} - 1$ complex positive-frequency components $X_1, \ldots, X_{(N/2)-1}$, plus two purely real components X_0 and $X_{N/2}$, for a total of N independent real numbers, as expected.

Many FFT implementations have special functions for computing FFTs of real data, and returning the Fourier components back in the original array. Typically this is done by storing the $\frac{N}{2}-1$ complex components $X_1, \ldots, N_{(N/2)-1}$ in place of the $N-2$ real numbers $x_2, \ldots, x_{N-1}$. The purely real zero-frequency X_0 and Nyquist frequency $X_{N/2}$ are stored in place of x_0 and x_1. So you should be aware that, with such routines, the "imaginary" part of the zero-frequency component is actually the Nyquist frequency component.

The MATLAB computing environment does *not* have specialized real FFT routines, so this issue does not arise in any of the exercises associated with this course. However, *most* implementations of the FFT *do* include such routines, so this is a feature you should be aware of, to avoid confusion down the road.

Parseval's theorem for DFTs

Recall Parseval's theorem for continuous Fourier transforms:

$$\int_{-\infty}^{\infty} |x(t)|^2\,dt \;=\; \int_{-\infty}^{\infty} |\tilde{x}(f)|^2\,df \tag{9}$$

To get the correponding formula for the discrete Fourier transform, we replace the integrals with sums, replace df and dt with Δf and Δt, and recall that $\tilde{x}(f_n) = X_n \Delta t$. (This is not a mathematically rigorous proof, but gives the right result.) Thus:

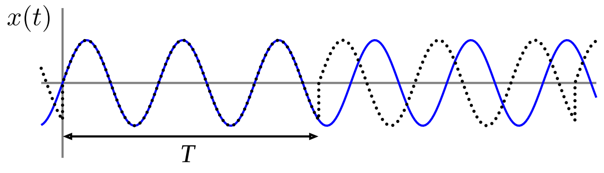
$$\begin{aligned}\sum_k |x_k|^2\,\Delta t &= \sum_n |X_n\,\Delta t|^2\,\Delta f \;=\; \Delta t^2\,\Delta f \sum_n |X_n|^2 \\ \sum_k |x_k|^2 &= \Delta t\,\Delta f \sum_n |X_n\,\Delta t|^2\end{aligned}$$

and $\Delta t\,\Delta f \;=\; \Delta t/T \;=\; 1/N$, so:

$$\boxed{\sum_{k=0}^{N-1} |x_k|^2 \;=\; \frac{1}{N}\sum_{n=0}^{N-1} |X_n|^2} \tag{10}$$

Window functions

Suppose we have some continuous function $x(t)$. When we compute the Fourier transform of a finite timespan T, we are assuming that the function is periodic with period T. I discussed earlier that this was equivalent to *convolving* the function with a Dirac comb with spacing T: that is, overlaying copies of $x(t)$ offset by successive intervals T. But this is only strictly true if $x(t)$ is contained entirely within the interval $t \in (0, T)$. If $x(t)$ represents an *ongoing* process, what we are really doing is *chopping it off* at the endpoints, and then attaching copies to make it periodic. This is illustrated below with a sine function: the solid curve represents the original function, while the dotted curve represents the function for which we are computing the Fourier transform:



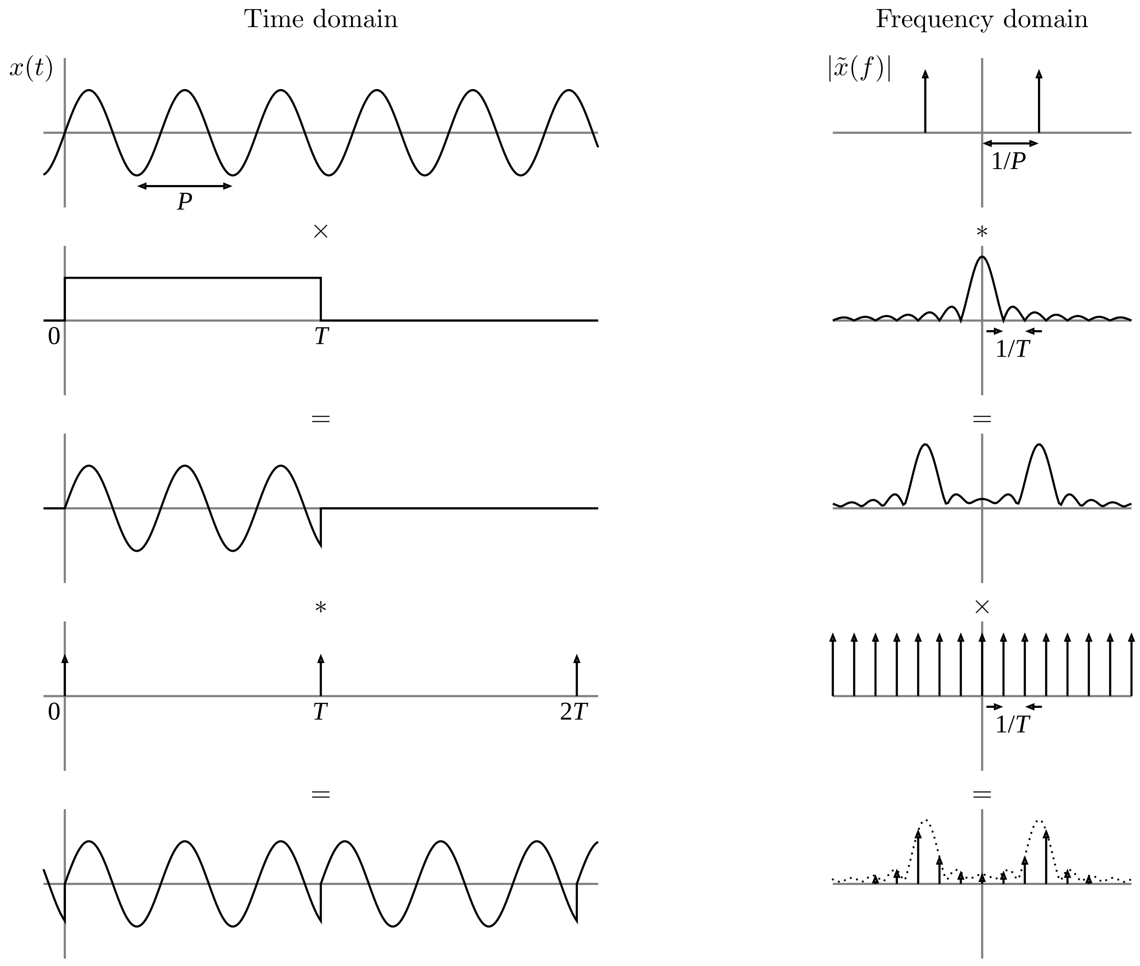
According to our rule of thumb, short timescales ⇔ wide frequency scales, so the sharp discontinuities in the function where we force it to be periodic will lead to a wide spread in frequencies. Again, we can imagine this pictorially. When we chop off the function in time, we are *multiplying* $x(t)$ with a rectangular function:

$$y(t) \;=\; \begin{cases} 1 & 0 \le t \le T \\ 0 & \text{otherwise} \end{cases}$$

This is called a **window function**, as it represents the finite span over which we are "viewing" our physical process: multiplying by a window function is called "windowing" the data. But multiplying $x(t) \times y(t)$ in the time domain means *convolving* $\tilde{x}(t) * \tilde{y}(f)$ in the frequency domain, where:

$$\begin{aligned}\tilde{y}(f) &= \int_{-\infty}^{\infty} y(t)\,e^{-2\pi i f t}\,dt \;=\; \int_0^T e^{-2\pi i f t}\,dt \;=\; \frac{1}{-2\pi i f}\left[e^{-2\pi i f T} - 1\right] \\ &= \frac{1}{\pi f}\,e^{-\pi i f T}\,\frac{1}{2i}\left[e^{\pi i f T} - e^{-\pi i f T}\right] \;=\; \frac{\sin(\pi f T)}{\pi f}\,e^{-\pi i f T} \;=\; T\,e^{-\pi i f T}\,\mathrm{sinc}(\pi f T)\end{aligned}$$

where $\mathrm{sinc}(x) \equiv \sin(x)/x$. The entire procedure of applying the window and imposing periodicity can be visualized as follows:



As predicted, the effect of windowing on a nonperiodic function is to spread it out in frequency, wiith the sinc function decreasing as $\sim 1/f$ for large f.

What if our function *is* in fact perfectly periodic with period T? That is, suppose T is a multiple of P. In the time domain, multiplying by the window function and imposing periodicity has no effect on the function $x(t)$. In the frequency domain we are still convolving with the sinc function and then sampling at frequency intervals $1/T$, but because $1/P$ is a multiple of $1/T$, all our frequency samples lie on the zeroes of the sinc function, except for the two samples at the peaks $\pm 1/P$. So in this case convolving with the sinc function and sampling has no effect on $\tilde{x}(f)$.

The effect of windowing limits our ability to reconstruct the original frequency of the signal, or to recover weaker signals covered by the "wings" of a strong signal. Can anything be done about this? Well, the $1/f$ wings arise from the sharp discontinuity of the windowed function, which results from the sharp edges of the window. If instead we multiply by a smoother window, then the wings will not be as pronounced. For example, if we choose a triangular-shaped window:

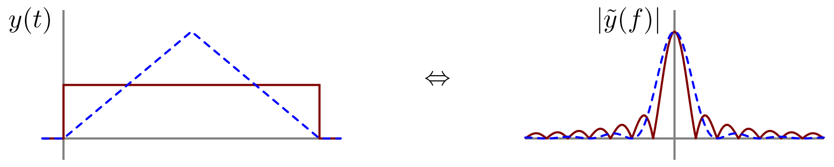
$$y(t) \;=\; \begin{cases} 4t/T & 0 \le t \le T/2 \\ 4 - 4t/T & T/2 \le t \le T \\ 0 & \text{otherwise} \end{cases} \qquad \Leftrightarrow \qquad \tilde{y}(f) \;=\; T\, e^{-\pi i f T}\, \frac{1-\cos(\pi f T)}{\frac{1}{2}\pi^2 f^2 T^2}$$

(work it out!), we see that in the frequency domain the wings fall off as $1/f^2$. This falloff is due to the fact that, even though the window function doesn't have sharp *edges* (where the function is discontinuous), it does have sharp *points* (where the *derivative* of the function is discontinuous). In general:

$$\boxed{\frac{d^n y}{dt^n} \text{ is discontinuous} \qquad \Leftrightarrow \qquad \tilde{y}(f) \;\sim\; \frac{1}{f^{n+1}} \text{ for large } f}$$

Note that if $y(t)$ *and* all its derivatives are continuous, then $\tilde{y}(f)$ will fall off exponentially for large f. For example, if $y(t)$ is Gaussian, then $\tilde{y}(f)$ is also Gaussian.

Using smooth window functions has its disadvantages, though. The more you smooth out the edges, the more of your original data you are chopping away. So although you are eliminating some of the very short timescale features, you are also concentrating the data into a shorter overall timespan. In the frequency domain, this means that as you lower the wings of $\tilde{y}(f)$, you are broadening the central peak, as illustrated for the rectangular and triangular windows:



This leads to the second rule of thumb for window functions:

Faster fall-off of wings	$\Leftrightarrow$	Broader central peak
Narrower central peak	$\Leftrightarrow$	Slower fall-of of wings

The choice of a windowing function somewhat of an art, and depends on whether you want to resolve nearby features of similar strength (requiring a narrower central peak), or whether you want to resolve weak features that might be obscured by stronger features (requiring lower wings). A wide peak with low wings will reduce *spectral bias*, meaning that the Fourier spectrum is less sensitive to whether, for instance, a particular signal frequency is or is not an exact multiple of $1/T$. Sometimes the *speed* of computing the window function is a consideration, making polynomials in t preferable to sines, cosines, or exponentials.