Sampling and Interpolation

In the previous chapter we studied three properties of III that make it so useful in many applications. They are:

- **Periodizing.**
  - Convolving with III periodizes a function.

- **Sampling.**
  - Multiplying by III samples a function.

- **The Fourier transform of III is III.**
  - Through this and the convolution theorem for Fourier transforms, periodizing and sampling are flip sides of the same coin.

We are about to combine these ideas in a spectacular way to treat the problem of sampling and interpolation.

Let me state the problem this way:

- **Given a signal** \( f(t) \) **and a collection of samples** of the signal, i.e., values of the signal at a set of points \( f(t_0), f(t_1), f(t_2), \ldots \), **how can one interpolate the values** \( f(t) \) **at other points?**

This is an old question and a broad one, and it would appear on the surface to have nothing to do with III’s or Fourier transforms or any of that. But we’ve already seen some clues, and in at least one important case the full solution is set to unfold.

### 6.1. Sampling sines and the Idea of a Bandlimited Signal

Imagine putting down a bunch of dots — theoretically, maybe infinitely many — and asking someone to pass a curve through them that agrees everywhere exactly with a predetermined mystery function passing through those dots. Plainly a ridiculous request. Isn’t it?

Try this with a sum of two sinusoids. The curve and the sample points are shown. The sample points in this case are evenly spaced though in general they
need not be. And I don’t mean to be mysterious about this: I played around and
settled on the signal \( f(t) = 4 \cos(2\pi(0.2)t) + 1.5 \sin(2\pi(1.414)t) \) for \( 0 \leq t \leq 12 \). The
sample points are spaced 0.5 apart.

Two simple methods for drawing an approximating curve are linear interpolation\(^1\)
and sample-and-hold.

Comparing with the actual curve, the interpolated curves clearly miss the more
rapid oscillations that take place between sample values. No surprise that this might
happen. We could add more sample points but, to repeat, is it just ridiculous to
expect to be able to do exact interpolation? There will always be some uncertainty
in the interpolated values. Won’t there?

It’s not ridiculous. If a relatively simple hypothesis is satisfied, then exact
interpolation can be done! Here’s one way of getting some intuitive sense of the
problem and what that hypothesis should be.

Suppose we know that the mystery signal is a single sinusoid, say of the form
\( A \sin(2\pi \nu t + \phi) \). A sinusoid repeats, so if we have enough information to pin it
down over one cycle, then we know the whole thing. How many samples — how
many values of the function — within one cycle do we need to know which sinusoid
we have? We need three samples strictly within one cycle. You can think of the
diagram or you can think of the equation. There are three unknowns, the amplitude
\( A \), the frequency \( \nu \), and the phase \( \phi \). We would expect to need three equations
to find the unknowns; hence we need values of the function at three points, three
samples. I’m not saying that it’s so easy to solve for the unknowns, only that it’s
what we might expect.

\(^1\)Otherwise known as connect the dots. There used to be connect the dots books for little kids,
not featuring sinusoids.
What if the signal is a sum of sinusoids, say
\[ \sum_{n=1}^{N} A_n \sin(2\pi n \nu_n t + \phi_n). \]
Sample points for the sum are morally sample points for the individual harmonics, though not explicitly. We need to take enough samples to get sufficient information to determine all of the unknowns for all of the harmonics. Now, in the time it takes for the combined signal to go through one cycle, the individual harmonics will have gone through several cycles, the lowest frequency harmonic through one cycle, the lower frequency harmonics through a few cycles, say, and the higher frequency harmonics through many. We have to take enough samples of the combined signal so that as the individual harmonics go rolling along we’ll be sure to have at least three samples in some cycle of every harmonic.

To simplify and standardize, we assume that we take evenly spaced samples. That’s what you’d typically record with a measuring instrument. Since we’ve phrased things in terms of cycles per second, it’s then also better to think in terms of sampling rate, i.e., samples/sec instead of number of samples. If we are to have at least three samples strictly within a cycle, then the sample points must be strictly less than a half-cycle apart. A sinusoid of frequency \( \nu \) goes through a half-cycle in \( \frac{1}{2\nu} \) seconds so we want
\[
\text{spacing between samples} = \frac{\text{number of seconds}}{\text{number of samples}} < \frac{1}{2\nu}.
\]
The more usual way of putting this is
\[
\text{sampling rate} = \text{samples/sec} > 2\nu.
\]
This is the rate at which we should evenly sample a given sinusoid of frequency \( \nu \) to guarantee that a single cycle will contain at least three sample points. Furthermore, if we sample at this rate for a given frequency, we will certainly have more than three sample points in some cycle of any harmonic at a lower frequency. Note again that the sampling rate has units 1/second and that sample points are 1/(sampling rate) seconds apart.

For the combined signal — a sum of harmonics — the higher frequencies are driving up the sampling rate; specifically, the highest frequency is driving up the rate. To think of the interpolation problem geometrically, high frequencies cause more rapid oscillations, i.e., rapid changes in the function over small intervals. To hope to interpolate such fluctuations accurately we’ll need a high sampling rate. If we sample at too low a rate, we might miss the wiggles entirely. We might mistakenly think we had only a low frequency sinusoid, and, moreover, since all we have to go on are the samples, we wouldn’t even know we’d made a mistake! We’ll come back to just this problem (called aliasing) a little later.

If we sample at a rate greater than twice the highest frequency, our sense is that we will be sampling often enough for all the lower harmonics as well, and we should be able to determine everything. Well, maybe. I’m not saying that we can disentangle the sample values in any easy way to determine the individual harmonics, but in principle it looks like we have enough information.
The problem is if the spectrum is unbounded. If we have a full Fourier series and not just a finite sum of sinusoids, then, in this argument at least, we can’t expect to sample frequently enough to determine the combined signal from the samples; there is no “highest frequency.”

**Bandlimited signals.** It’s time to define ourselves out of trouble. From the point of view above, whatever the caveats, the problem for interpolation is high frequencies. The best thing a signal can be is a finite Fourier series, in which case the signal has a discrete, finite set of frequencies that stop at some point. This is much too restrictive for applications, of course, so what’s the next best thing a signal can be? It’s one for which the spectrum is zero from some point on. These are the bandlimited signals — signals whose Fourier transforms are identically zero outside of a bounded interval, outside of a bounded band of frequencies. More formally:

- A signal $f(t)$ is bandlimited if there is a finite number $p$ such that $\mathcal{F} f(s) = 0$ for all $|s| \geq p/2$.

Interesting that an adjective attached to the signal in the time domain is defined by a property in the frequency domain. The Fourier transform $\mathcal{F} f(s)$ may well have zeros at points $s$ with $|s| \leq p/2$, but for sure it’s identically zero for $|s| \geq p/2$. Recalling terminology from Chapter 4, you will recognize the definition as saying that $f(t)$ is bandlimited if $\mathcal{F} f(s)$ has compact support.\(^2\)

- The smallest number $B$ for which the condition is satisfied is called the bandwidth. I’m sure you’re familiar with that term, but there are some subtleties. We’ll come back to this. In the meantime note carefully that the bandwidth is $B$, not $B/2$.

Singling out the collection of bandlimited signals is an example of one of the working principles laid down by the eminent British mathematician J. E. Littlewood, one of my mathematical forebearers. To wit: “Make your worst enemy your best friend.” The idea is to identify the stumbling block and make an assumption, a definition, that will eliminate or weaken the opposition. Not always easy to do, but an important strategy to keep in mind. The enemy is high frequencies. We wish them away and see what happens.

### 6.2. Sampling and Interpolation for Bandlimited Signals

We’re about to solve the interpolation problem for bandlimited signals. We’ll show that interpolation is possible by finding an explicit formula that does the job. This uses all the important properties of $\Pi_p$, but it goes so fast that you might miss the fun entirely if you read too quickly. I’ll make an effort to slow things down.

Suppose $f(t)$ is a bandlimited signal with $\mathcal{F} f(s)$ identically zero for $|s| \geq p/2$. We periodize $\mathcal{F} f$ using $\Pi_p$ and then cut off to get $\mathcal{F} f$ back again:

$$\mathcal{F} f = \Pi_p(\mathcal{F} f \ast \Pi_p).$$

\(^2\)To repeat from Chapter 4, the support is the smallest closed set containing the set $\{x: f(x) \neq 0\}$. To say that $\mathcal{F} f(s)$ has compact support is to say that it is identically zero outside a bounded set.
6.2. Sampling and Interpolation for Bandlimited Signals

This is the crucial equation. The condition of being bandlimited to $|s| \leq p/2$ means that convolving $\mathcal{F}f(s)$ with $\Pi_p$ shifts the spectrum off itself — no overlaps, not even any (nonzero) touching of the shifted copies since $\mathcal{F}f(\pm p/2) = 0$. We recover the original Fourier transform $\mathcal{F}f(s)$ by multiplying the periodized function $\mathcal{F}f * \Pi_p$ by a rectangle function of width $p$. (Low-pass filtering.)

Here’s the picture to keep in mind. Take a real, bandlimited function $f(t)$ and plot the magnitude $|\mathcal{F}f(s)|$ (the best we can do), which is even (because $f(t)$ is real):

![Magnitude Plot](image1)

The periodized $\mathcal{F}f(s)$, or rather its magnitude, looks like

![Periodized Plot](image2)

I put a plot of $\Pi_p$ in there, too, so you could see how convolution with $\Pi_p$ shifts the spectrum (and adds the shifts together). Then what remains after multiplying by $\Pi_p$ is the original $\mathcal{F}f(s)$:

![Remaining Plot](image3)

Brilliant. We did something (periodize) and then undid it (cut off), and we’re back to where we started, with $\mathcal{F}f(s)$. Have we done nothing? To the contrary,
watch! Take the inverse Fourier transform of \( F_f = \Pi_p(F_f \ast \Pi_p) \):

\[
f(t) = \mathcal{F}^{-1} F_f(t) = \mathcal{F}^{-1}(\Pi_p(F_f \ast \Pi_p))(t) = \mathcal{F}^{-1} \Pi_p(t) \ast \mathcal{F}^{-1}(F_f \ast \Pi_p)(t)
\]

(taking \( \mathcal{F}^{-1} \) turns multiplication into convolution)

\[
= \mathcal{F}^{-1} \Pi_p(t) \ast (\mathcal{F}^{-1} F_f(t) \cdot \mathcal{F}^{-1} \Pi_p(t))
\]

(on this line it’s convolution turning into multiplication)

\[
= p \text{sinc} pt \ast \left( f(t) \cdot \frac{1}{p} \Pi_{1/p}(t) \right) \quad \text{(note that the } p \text{'s cancel)}
\]

\[
= \text{sinc} pt \ast \sum_{k=\infty} f \left( \frac{k}{p} \right) \delta \left( t - \frac{k}{p} \right) \quad \text{(the sampling property of } \Pi_p)
\]

\[
= \sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc} pt \ast \delta \left( t - \frac{k}{p} \right)
\]

\[
= \sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc} \left( t - \frac{k}{p} \right) \quad \text{(the convolution property of } \delta).
\]

6.2.1. The Nyquist-Shannon sampling theorem. We’ve just established the classic Nyquist-Shannon sampling theorem, though it might be better to call it the interpolation theorem. Here it is as a single statement:

- If \( f(t) \) is a signal with \( \mathcal{F}f(s) \) identically zero for \( |s| \geq p/2 \), then
  \[
  f(t) = \sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc} \left( t - \frac{k}{p} \right).
  \]

The attribution is in honor of Harry Nyquist, God of Sampling, who was the first engineer to consider these problems for the purpose of communications, and Claude Shannon, overall genius and founder of information theory, who in 1949 presented the result as it appears here. There are other names associated with sinc interpolation, most notably the mathematician E. Whittaker whose paper was published in 1915.\(^3\)

People usually refer to this expression as the sampling formula, or the interpolation formula, and it’s often written as

\[
f(t) = \sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc}(pt - k).
\]

I generally prefer to emphasize the sample points \( k/p \) within the sinc functions. If we further write

\[
t_k = \frac{k}{p},
\]

\(^3\)Whittaker called the sum of sincs the cardinal series. (I’m sure he didn’t write “sinc.”) Impress your friends with your erudition by using this term, but look up some of the history first.
then the formula is

\[ f(t) = \sum_{k=-\infty}^{\infty} f(t_k) \text{sinc}(p(t - t_k)). \]

As a quick reality check, let's verify that the formula when evaluated at a sample point returns the value of the signal at that point. Plug in \( t_\ell = \ell/p \):

\[
\sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc} \left( \frac{\ell}{p} - \frac{k}{p} \right) = \sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc}(\ell - k).
\]

Now remember that the sinc function is zero at the integers, except that sinc0 = 1, so

\[
\text{sinc}(\ell - k) = \begin{cases} 
1, & \ell = k, \\
0, & \ell \neq k,
\end{cases}
\]

and hence

\[
\sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc}(\ell - k) = f \left( \frac{\ell}{p} \right).
\]

Good thing.

For me, the sampling theorem is identical to the proof of the sampling theorem; it’s so automatic. It all depends on having

\[ \mathcal{F} f = \Pi_p (\mathcal{F} f * \Pi_p). \]

Say it with me: periodize and cut off to get back the original Fourier transform, and then interchange periodizing and sampling via the convolution theorem. It’s almost indecent the way this works. There are variations on the theme, if there are perhaps special aspects of the spectrum. You’ll see examples in the problems, e.g., “islands” where the spectrum is nonzero, separated by intervals where the spectrum vanishes. But the idea of periodize-and-cut-off is always there.

**Bandwidth, sampling rates, and the Nyquist frequency.** There is a series of observations on this wondrous result. The first is yet another reciprocal relationship to keep in mind:

- The sample points are spaced \( 1/p \) apart.
- The **sampling rate** is \( p \), in units of Hz. This is the setting on your sampling machine. The higher the sampling rate, the more closely spaced the sample points.

I point this out because there are higher-dimensional versions of the sampling theorem and more subtle reciprocal relationships. This will be a topic in Chapter 9.

Next some matters of definition. If you look up the sampling theorem elsewhere, you may find other conventions and other notations. Often the assumption is written as \( \mathcal{F} f(s) = 0 \) for \(|s| > \nu_{\text{max}}\), and the conclusion is then that the signal can be interpolated, with the formula as above, using any sampling rate > \( 2\nu_{\text{max}} \). Note the strict inequalities here. (This way of writing the sampling rate does jibe with our initial discussion of sampling a sinusoid.)
The number $2\nu_{\text{max}}$ is often called the Nyquist frequency, as well as being called the bandwidth. You’ll hear people say things like: “For interpolation, sample at a rate greater than twice the highest frequency.” OK, but those same people are not always clear what they mean by “highest frequency.” Is it the largest frequency for which $\mathcal{F}f(s) \neq 0$ or is it the smallest frequency for which $\mathcal{F}f(s)$ is identically zero beyond that point?

Mathematically, the way out is to use the infimum of a set of numbers, abbreviated inf, to define $\nu_{\text{max}}$ and the bandwidth. The infimum of a set of numbers is the greatest lower bound of the set. For example,

$$\sqrt{2} = \inf\{x : x \text{ is a positive rational number and } x^2 \geq 2\}.$$ 

The picture, with the set in the heavier line:

\[
\begin{array}{c}
\sqrt{2} \\
\end{array}
\]

Note that $\sqrt{2}$ is not itself in the set since it’s irrational. The various approximations that you know, 1.4, 1.41, 1.414, ..., are all lower bounds for any number in the set, and $\sqrt{2}$ is the greatest lower bound.\(^4\)

For a signal $f(t)$ we consider the set

$$\{p : \mathcal{F}f(s) = 0 \text{ for all } |s| \geq p/2\}$$

and we find the infimum

$$\inf\{p : \mathcal{F}f(s) = 0 \text{ for all } |s| \geq p/2\}.$$ 

The infimum is what people mean by $\nu_{\text{max}}$, and the bandwidth is twice this:

$$B = 2\inf\{p : \mathcal{F}f(s) = 0 \text{ for all } |s| > p/2\} = 2\nu_{\text{max}}.$$ 

As with the $\sqrt{2}$ example, the number $\nu_{\text{max}}$ may not be in the set whose infimum defines it; i.e., we may not have $\mathcal{F}f(\pm\nu_{\text{max}}) = 0$. (However if $f(t)$ is an integrable function, then $\mathcal{F}f(s)$ is continuous, so the fact that $\mathcal{F}f(s) = 0$ for all $|s| > B/2$ implies that $\mathcal{F}f(\pm B/2) = 0$ as well.) In any event, if $p/2 > \nu_{\text{max}}$, we do have $\mathcal{F}f(s) = 0$ for $|s| \geq p/2$, so including the endpoints $\pm p/2$, and this is what we need for the derivation of the sampling theorem.

Why these niggling remarks? Take, for example, a simple sinusoid, say $f(t) = \sin 2\pi(B/2)t$ (not integrable!), for which $\mathcal{F}f(s) = (1/2i)(\delta(s-B/2) - \delta(s+B/2))$. What is the bandwidth? Everyone in the world would say it’s $B$, and so would I. Sure enough, this conforms to the definition, if you’re willing to grant that the $\delta$’s are zero away from $\pm B/2$.\(^5\) To apply the sampling formula to $\sin 2\pi(B/2)t$ (which, for me, is to apply the derivation of the sampling formula) we’d sample at any rate $> B$.

---

\(^4\)It is a deep property of the real numbers, taken in some form as an axiom, that any set of numbers that doesn’t stretch down to $-\infty$ has an infimum. This is called the completeness property of the real numbers and it underlies many of the existence theorems in calculus.

\(^5\)We know that we don’t really talk about the values of $\delta$’s at points, but here we are around half-way through the book and I’m willing to let that slide.
In fact, to conclude, let’s restate the sampling theorem using the bandwidth.

- If $f(t)$ is a bandlimited signal with bandwidth $B$, then

$$f(t) = \sum_{k=-\infty}^{\infty} f\left(\frac{k}{p}\right) \text{sinc}\left(p\left(t - \frac{k}{p}\right)\right)$$

for any $p > B$.

Got it? The formula is such a nice thing. We’re saving the nasty things for later.\(^6\)

After this abstraction here’s one real-world example, the sampling rate for the audio on compact discs. It’s 44.1 kHz. Where does this number come from? We can hear frequencies up to about 20 kHz; that’s $\nu_{\text{max}}$, so the sampling rate should be upwards of $2\nu_{\text{max}} = 40$ kHz.\(^7\) The precise number 44.1 kHz allows for some slack but really comes from the legacy equipment used for analog recordings.

**Bandlimited and timelimited signals.** You may have noted, with some sadness, that the sampling theorem involves an infinite sum and infinitely many sample points. Two questions come to mind:

- Do I have to worry about convergence of the series?
- Do I really need infinitely many sample points?

On the first point, see the problems for a theorem that guarantees convergence of a sum of the form

$$\sum_{k=-\infty}^{\infty} a_k \text{sinc}\left(p\left(t - \frac{k}{p}\right)\right).$$

That result suffices for almost everything. We’ll be meeting various infinite series, and we won’t be concerned with questions of convergence. The rigor police are back to hanging around outside the room.

The second point is really the more interesting one. It points to a phenomenon that has a consequence for the sampling theorem but is independent of it.

It is unphysical to consider a signal as lasting forever in time. A physical signal $f(t)$ is naturally timelimited, meaning that $f(t)$ is identically zero for $|t| \geq q/2$ for some $q$. There just isn’t any signal beyond some point. (In mathematical language, a physical signal has compact support.) On the other hand, it is very physical to consider a bandlimited signal, one with no frequencies beyond a certain point, or at least no frequencies that our instruments can register. Well, we can’t have both, at least not in the ideal world of mathematics. Here is where mathematical description meets physical expectation, and they disagree. The fact is:

- A signal cannot be both timelimited and bandlimited unless it is identically zero.

What this means in practice is that there must be inaccuracies in a mathematical model of a phenomenon that assumes a signal is both timelimited and bandlimited.

---

\(^6\)See *Lucky Jim* by Kingsley Amis for an appreciation of nice things versus nasty things. Mull over his description of much academic work as “throwing pseudo-light on nonproblems.”

\(^7\)As a trombonist (see Chapter 1), I have spent years sitting in front of a trumpet section. My hearing probably no longer goes up to 20 kHz.
Such a model can be at best an approximation, and one has to be prepared to estimate the errors as they may affect measurements and conclusions.

Here’s one argument why the statement is true: I’ll give another, more refined, statement and proof in Section 6.5. Suppose $f(t)$ is bandlimited, say, $\mathcal{F}f(s)$ is zero for $|s| \geq p/2$. Then

$$\mathcal{F}f = \Pi_p \cdot \mathcal{F}f.$$ 

No periodization here, just cutting off the already limited Fourier transform. Take the inverse Fourier transform of both sides to obtain

$$f(t) = p \text{sinc} pt * f(t).$$ 

Now sinc $pt$ goes on forever; it decays but it has nonzero values all the way out to $\pm \infty$. Hence the convolution with $f$ also goes on forever, so $f(t)$ is not timelimited.

sinc as an identity for convolution. There’s an interesting observation that goes along with the argument I just gave. We’re familiar with $\delta$ acting as an identity element for convolution, meaning

$$f * \delta = f.$$ 

This important property of $\delta$ holds for all signals for which the convolution is defined. We’ve just seen for the more restricted class of bandlimited functions, with spectrum from $-p/2$ to $+p/2$, that the sinc function also has this property:

$$p \text{sinc} pt * f(t) = f(t).$$ 

We can also shift:

$$p \text{sinc} p(t - a) * f(t) = f(t - a).$$ 

What is the consequence of the bandlimited vs. timelimited phenomenon for the sampling theorem? If the signal $f(t)$ is bandlimited, then it cannot be timelimited; $f(t)$ has nonzero values all the way out to $\pm \infty$. Consequently, we should expect to need sample points out to $\pm \infty$. It’s not reasonable to take samples only up to a finite point and still interpolate values of the function way beyond (infinitely beyond) that point. This means, of course, that any practical application of the sampling theorem, which must use a finite sum, will have to be an approximation. These problems are absolutely inevitable. The approaches are via filters, first low-pass filters done before sampling to force a signal to be bandlimited and then other kinds of filters (smoothing) following whatever reconstruction is made from the samples. Particular kinds of filters are designed for particular kinds of signals, e.g., sound or images.

---

8 Later we’ll walk this back in the case of sampling and interpolating periodic signals. See also J. R. Higgins, Sampling Theory in Fourier and Signal Analysis. A nice reference with all sorts of nice topics.
6.2.2. An example. Before we consider how the sampling formula might go rogue, let’s see how it works nicely with the curve we looked at first, back in Section 6.1. The signal is

\[ f(t) = 4\cos(2\pi(0.2)t) + 1.5\sin(2\pi(1.414)t) \]

on the interval \(0 \leq t \leq 12\). The bandwidth is 2.828 so we need a sampling rate greater than this to apply the formula and expect to get the signal back.

Jumping all the way up to \(p = 3\), giving 37 sample points, here’s a plot of the original curve, the sample points, and the finite sum:

\[ \sum_{k=0}^{36} f\left(\frac{k}{3}\right) \text{sinc}3\left(t - \frac{k}{3}\right). \]

The original curve is solid and the approximating curve is dashed. You can spot some differences, particularly where \(f(t)\) is bending the most, naturally, but even with the finite approximation there’s very little daylight between the two curves.

6.2.3. Interpolation and orthogonality. Here’s a brief take on orthogonality playing a role in the sinc interpolation formula, very much analogous to orthogonality and Fourier series.

Recall that the inner product on \(L^2(\mathbb{R})\), the square integrable functions defined on \(\mathbb{R}\), is

\[ (f, g) = \int_{-\infty}^{\infty} f(t)\overline{g(t)} \, dt. \]

Two functions are orthogonal if their inner product is zero.

Another amazing property of sinc functions is that the family of shifted sincs, \(\text{sinc}(t - n), n = 0, \pm 1, \pm 2, \ldots\), of bandwidth 1, is orthonormal. The calculation to
establish this is the general Parseval identity, which you’ll recall says

\[ \int_{-\infty}^{\infty} f(t)g(t) \, dt = \int_{-\infty}^{\infty} \mathcal{F}f(s)\mathcal{F}g(s) \, ds. \]

For the shifted sincs we have

\[ \int_{-\infty}^{\infty} \text{sinc}(t-n) \text{sinc}(t-m) \, dt = \int_{-\infty}^{\infty} (e^{-2\pi isn}\Pi(s))(e^{-2\pi ism}\Pi(s)) \, ds \]

\[ = \int_{-\infty}^{\infty} e^{2\pi is(m-n)}\Pi(s)\Pi(s) \, ds \]

\[ = \int_{-1/2}^{1/2} e^{2\pi is(m-n)} \, ds. \]

Welcome back to that celebrated integral, last seen playing a fundamental role for Fourier series, in Chapter 1. Then, as now, direct integration gives you 1 when \( n = m \) and 0 when \( n \neq m \). And in case you’re fretting over it, the sinc function is in \( L^2(\mathbb{R}) \) and the product of two sinc functions is integrable. Parseval’s identity holds for functions in \( L^2(\mathbb{R}) \).

Sticking with bandwidth 1 for the moment, coupled with the result on orthogonality, the formula

\[ f(t) = \sum_{n=-\infty}^{\infty} f(n) \text{sinc}(t-n) \]

suggests that the family of sinc functions forms an orthonormal basis for the space of bandlimited signals with spectrum in \(-1/2 \leq s \leq 1/2\) and that we’re expressing \( f(t) \) in terms of this basis. That’s exactly what’s going on. The coefficients (the sample values \( f(n) \)) are obtained as the inner product of \( f(t) \) with \( \text{sinc}(t-n) \). We have, again using Parseval,

\[ (f(t), \text{sinc}(t-n)) = \int_{-\infty}^{\infty} f(t) \text{sinc}(t-n) \, dt \]

\[ = \int_{-\infty}^{\infty} \mathcal{F}f(s)\mathcal{F}(\text{sinc}(t-n)) \, ds \quad \text{(by Parseval)} \]

\[ = \int_{-\infty}^{\infty} \mathcal{F}f(s)(e^{-2\pi isn}\Pi(s)) \, ds \]

\[ = \int_{-1/2}^{1/2} \mathcal{F}f(s)e^{2\pi isn} \, ds \]

\[ = \int_{-\infty}^{\infty} \mathcal{F}f(s)e^{2\pi isn} \, ds \quad \text{(because \( f \) is bandlimited)} \]

\[ = f(n) \quad \text{(by Fourier inversion)}. \]

It’s perfect! The interpolation formula says that \( f(t) \) is written in terms of an orthonormal basis, and the coefficient \( f(n) \), the \( n \)th sampled value of \( f(t) \), is exactly the projection of \( f(t) \) onto the \( n \)th basis element:

\[ f(t) = \sum_{n=-\infty}^{\infty} f(n) \text{sinc}(t-n) = \sum_{n=-\infty}^{\infty} (f(t), \text{sinc}(t-n)) \text{sinc}(t-n). \]
Changing to bandwidth $p$ is analogous to changing the period for Fourier series. For the shifted sincs we again use Parseval to calculate the inner product:

$$
\left( \text{sinc} p \left( t - \frac{n}{p} \right), \text{sinc} p \left( t - \frac{m}{p} \right) \right) = \int_{-\infty}^{\infty} \text{sinc} p \left( t - \frac{n}{p} \right) \text{sinc} p \left( t - \frac{m}{p} \right) dt
$$

$$
= \int_{-\infty}^{\infty} \frac{1}{p} e^{-2\pi i s(n/p)} \Pi_p(s) \frac{1}{p} e^{-2\pi i s(m/p)} \Pi_p(s) ds
$$

$$
= \frac{1}{p^2} \int_{-p/2}^{p/2} e^{2\pi i s(p)(m-n)} ds
$$

$$
= \begin{cases} 
1/p, & m = n, \\
0, & m \neq n. 
\end{cases}
$$

Orthogonal but not orthonormal. This is telling us that the orthonormal family to consider is

$$\sqrt{p} \text{sinc} p \left( t - \frac{n}{p} \right), \quad n = 0, \pm 1, \pm 2, \ldots$$

And the sample values? For the projection of $f(t)$, of bandwidth $p$, onto the sincs we have, with Parseval (skipping a few steps),

$$
\left( f(t), \sqrt{p} \text{sinc} p \left( t - \frac{n}{p} \right) \right) = \int_{-\infty}^{\infty} \mathcal{F}f(s) \sqrt{p} \frac{1}{p} e^{2\pi i s(n/p)} \Pi_p(s) ds
$$

$$
= \frac{1}{\sqrt{p}} \int_{-\infty}^{\infty} \mathcal{F}f(s) e^{2\pi i s(n/p)} ds = \frac{1}{\sqrt{p}} f \left( \frac{n}{p} \right).
$$

Then expanding in terms of the orthonormal basis cancels the $\sqrt{p}$’s:

$$
\sum_{k=-\infty}^{\infty} \left( f(t), \sqrt{p} \text{sinc} p \left( t - \frac{k}{p} \right) \right) \sqrt{p} \text{sinc} p \left( t - \frac{k}{p} \right)
$$

$$
= \sum_{k=-\infty}^{\infty} \frac{1}{\sqrt{p}} f \left( \frac{k}{p} \right) \sqrt{p} \text{sinc} p \left( t - \frac{k}{p} \right) = \sum_{k=-\infty}^{\infty} f \left( \frac{k}{p} \right) \text{sinc} p \left( t - \frac{k}{p} \right).
$$

It’s perfect, again.

6.3. Undersampling and Aliasing

The troubles. What if we work a little less hard than as dictated by the bandwidth. What if we undersample a bit and try to apply the interpolation formula with a little lower sampling rate and with the sample points spaced a little farther apart. Will the interpolation formula produce almost a good fit, good enough to listen to or to see? Maybe yes, maybe no. A disaster is a definite possibility.

6.3.1. Sampling sinusoids, still. Let’s revisit the question of sampling and interpolation for a simple sine function and let’s work with an explicit example. Take the signal given by

$$f(t) = \cos \frac{9\pi}{2} t.$$

The frequency of this signal is $9/4$ Hz. If we want to apply the sampling formula, we can take the sampling rate to be anything $> 9/2 = 4.5$. Suppose our sampler
is stuck in low and we can only take one sample every second. Then our samples have values
\[ \cos \frac{9\pi}{2} n, \quad n = 0, 1, 2, 3, \ldots. \]
There is another, lower frequency signal that has the same samples. To find it, take away from \( 9\pi/2 \) the largest multiple of \( 2\pi \) that leaves a remainder of less than \( \pi \) in absolute value, so there’s a spread of less than \( 2\pi \), one full period, to the left and right. You’ll see what I mean as the example proceeds. Here we have
\[ \frac{9\pi}{2} = 4\pi + \frac{\pi}{2}. \]
Then
\[ \cos \frac{9\pi}{2} n = \cos ((4\pi + \frac{\pi}{2}) n) = \cos \frac{\pi}{2} n. \]
The signal \( f(t) \) has the same samples at 0, ±1, ±2, … as the signal
\[ g(t) = \cos \frac{\pi}{2} t, \]
whose frequency is only \( 1/4 \). The two functions are not the same everywhere, but their samples at the integers are equal.

Here are plots of the original signal \( f(t) \) and of \( f(t) \) and \( g(t) \) plotted together, showing how the curves match up at the sample points.\(^9\)

---

\(^9\)They match up wherever they intersect, but we’re only aware of the measurements we make.
6.3. Undersampling and Aliasing

$g(t)$ is an alias of $f(t)$. The lower frequency signal is masquerading as the higher frequency signal via their identical sample values.

You have probably seen physical manifestations of this phenomenon with a strobe light flashing on and off on a moving fan, for example. There are more dramatic examples; see the problems.

Now let’s analyze this example in the frequency domain, essentially repeating the derivation of the sampling formula for this particular function at the particular sampling rate of 1 Hz. The Fourier transform of $f(t) = \cos\frac{9\pi t}{2}$ is

$$
F f(s) = \frac{1}{2} \left( \delta(s - \frac{9}{4}) + \delta(s + \frac{9}{4}) \right).
$$

To sample at $p = 1$ Hz means, first, that in the frequency domain we:

- Periodize $Ff$ by $\Pi_1$.
- Cut off by $\Pi_1$.

After that we take the inverse Fourier transform and, by definition, this gives the interpolation to $f(t)$ using the sample points $f(0), f(\pm 1), f(\pm 2), \ldots$. The question is whether this interpolation gives back $f(t)$. We know it doesn’t, but what goes wrong?

The Fourier transform of $\cos\frac{9\pi t}{2}$ looks like

For the periodization step, direct calculation results in

$$
Ff(s) * \Pi_1(s) = \frac{1}{2} \left[ \delta(s - \frac{9}{4}) + \delta(s + \frac{9}{4}) \right] * \sum_{k=-\infty}^{\infty} \delta(s - k)
$$

$$
= \frac{1}{2} \sum_{k=-\infty}^{\infty} \left( \delta(s - \frac{9}{4}) * \delta(s - k) + \delta(s + \frac{9}{4}) * \delta(s - k) \right)
$$

$$
= \frac{1}{2} \sum_{k=-\infty}^{\infty} \left( \delta(s - \frac{9}{4} - k) + \delta(s + \frac{9}{4} - k) \right)
$$

(remember the formula $\delta_a * \delta_b = \delta_{a+b}$).
Multiplying by $\Pi_1$ cuts off outside $(-1/2, +1/2)$, and we get $\delta$’s within $-1/2 < s < 1/2$ if, working separately with $\delta \left( s - \frac{9}{4} - k \right)$ and $\delta \left( s + \frac{9}{4} - k \right)$, we have

$$- \frac{1}{2} < -\frac{9}{4} - k < \frac{1}{2}, \quad - \frac{1}{2} < \frac{9}{4} - k < \frac{1}{2},$$

$$\frac{7}{4} < -k < \frac{11}{4}, \quad - \frac{11}{4} < k < -\frac{7}{4},$$

$$- \frac{11}{4} < k < -\frac{7}{4}, \quad \frac{7}{4} < k < \frac{11}{4}.$$  

Thus we get $\delta$’s within $-1/2 < s < 1/2$ for

$$k = -2 \quad \text{and the term} \quad \delta \left( s - \frac{9}{4} - (-2) \right) = \delta \left( s - \frac{1}{4} \right)$$

and

$$k = 2 \quad \text{and the term} \quad \delta \left( s + \frac{9}{4} - 2 \right) = \delta \left( s + \frac{1}{4} \right).$$

All other $\delta$’s in $Ff(s) * \Pi_1(s)$ will be outside the range $-1/2 < s < 1/2$, and the final result is

$$\Pi_1(s)(Ff(s) * \Pi_1(s)) = \frac{1}{2} \left( \delta \left( s + \frac{1}{4} \right) + \delta \left( s - \frac{1}{4} \right) \right).$$

We do not have

$$\Pi_1(Ff * \Pi_1) = Ff.$$  

So if we take the inverse Fourier transform of $\Pi_1(Ff * \Pi_1)$, we do not get $f$ back. But we can take the inverse Fourier transform of $\Pi_1(Ff * \Pi_1)$ anyway, and this produces

$$F^{-1} \left( \frac{1}{2} \left( \delta \left( s - \frac{1}{4} \right) + \delta \left( s + \frac{1}{4} \right) \right) \right) = \frac{1}{2} (e^{\pi i t/2} + e^{-\pi i t/2}) = \cos \frac{\pi}{2} t.$$  

There’s the aliased signal!

$$g(t) = \cos \frac{\pi}{2} t.$$  

It’s still quite right to think of $F^{-1}(\Pi_1(Ff * \Pi_1))$ as an interpolation based on sampling $f(t)$ at 1 Hz. That’s exactly what it is; it’s just not a good one. The sampling formula is

$$F^{-1}(\Pi_1(Ff * \Pi_1))(t) = \sin\left( t * (f(t) * \Pi_1(t)) \right)$$

$$= \sin\left( t * \sum_{k=-\infty}^{\infty} f(k) \delta(t - k) \right)$$

$$= \sum_{k=-\infty}^{\infty} f(k) \sin(t - k) = \sum_{k=-\infty}^{\infty} \cos \frac{9\pi k}{2} \sin(t - k).$$

This sum of sincs provided by the sampling formula isn’t $f(t) = \cos \frac{9\pi}{2} t$; it’s $g(t) = \cos \frac{\pi}{2} t$ (though you’d never know that just from the formula). To say it again, interpolating the samples of $f$ according to the formula at the sampling rate of 1 Hz — too low a sampling rate — has not produced $f(t)$; it has produced $g(t)$, an alias of $f$. Cool.

Before we leave this example let’s take one more look at

$$Ff(s) * \Pi_1(s) = \frac{1}{2} \sum_{k=-\infty}^{\infty} \left( \delta \left( s - \frac{9}{4} - k \right) + \delta \left( s + \frac{9}{4} - k \right) \right).$$
6.3. Undersampling and Aliasing

Being a convolution with $\mathcal{III}_1$, this is periodic of period 1, but, actually, it has a smaller period. To find it, write

$$\frac{1}{2} \sum_{k=-\infty}^{\infty} (\delta(s - \frac{9}{4} - k) + \delta(s + \frac{9}{4} - k)) = \frac{1}{2} \left( \mathcal{III}_1 (s - \frac{9}{4}) + \mathcal{III}_1 (s + \frac{9}{4}) \right)$$

$$= \frac{1}{2} \left( \mathcal{III}_1 (s - 2 - \frac{1}{4}) + \mathcal{III}_1 (s + 2 + \frac{1}{4}) \right)$$

(this is pretty much what we did at the top of the section)

$$= \frac{1}{2} \left( \mathcal{III}_1 (s - \frac{1}{4}) + \mathcal{III}_1 (s + \frac{1}{4}) \right)$$

(because $\mathcal{III}_1$ is periodic of period 1).

This sum of those two $\mathcal{III}$’s is periodic of period $1/2$, for

$$\frac{1}{2} \left( \mathcal{III}_1 (s - \frac{1}{4} + \frac{1}{2}) + \mathcal{III}_1 (s + \frac{1}{4} + \frac{1}{2}) \right) = \frac{1}{2} \left( \mathcal{III}_1 (s + \frac{1}{2}) + \mathcal{III}_1 (s + \frac{3}{2}) \right)$$

$$= \frac{1}{2} \left( \mathcal{III}_1 (s + \frac{1}{2}) + \mathcal{III}_1 (s + 1 - \frac{1}{2}) \right)$$

$$= \frac{1}{2} \left( \mathcal{III}_1 (s + \frac{1}{2}) + \mathcal{III}_1 (s - \frac{1}{4}) \right)$$

(using the periodicity of $\mathcal{III}_1$).

You can also see the reduced periodicity of $(1/2)(\mathcal{III}(s - 9/4) + \mathcal{III}(s + 9/4))$ graphically from the way $\mathcal{III}_1(s - 9/4)$ and $\mathcal{III}_1(s + 9/4)$ line up. Here’s a plot of part of

$$\frac{1}{2} \mathcal{III} \left( s - \frac{9}{4} \right) = \frac{1}{2} \sum_{k=-\infty}^{\infty} \delta(s - k - \frac{9}{4}) .$$

Here’s a plot of part of

$$\frac{1}{2} \mathcal{III} \left( s + \frac{9}{4} \right) = \frac{1}{2} \sum_{k=-\infty}^{\infty} \delta(s - k + \frac{9}{4}) .$$

Here’s a plot of the sum of the two.
6. Sampling and Interpolation

You can see that the $\delta$'s in the sum are spaced 1/2 apart. Cool? You might try working with $\cos 9\pi/2$ in the frequency domain using other sampling rates. See what periodizations look like and what happens when you cut off.

6.3.2. Sampling at the bandwidth. When we talked about the definition of bandwidth $B$ of a signal, we raised the issue of what happens at the endpoints of the spectrum, $\pm B$, whether or not the Fourier transform is zero there, and how this sometimes requires special consideration. Here’s an example of what I had in mind. Take two very simple signals,

$$f(t) = \sin 2\pi t \quad \text{and} \quad g(t) = \cos 2\pi t,$$

each of period 1. The Fourier transforms are

$$\mathcal{F}f(s) = \frac{1}{2i} (\delta(s - 1) - \delta(s + 1)) \quad \text{and} \quad \mathcal{F}g(s) = \frac{1}{2} (\delta(s - 1) + \delta(s + 1)).$$

The bandwidth is 2 for each signal and the Fourier transforms are not zero at the endpoints $\pm p/2 = \pm 1$.

If we apply the sampling formula with $p = 2$ to $\sin 2\pi t$, we get the upsetting news that

$$\sin 2\pi t = \sum_{k=-\infty}^{\infty} \sin \frac{2\pi k}{2} \sin \left(2 \left(t - \frac{k}{2}\right)\right) = \sum_{k=-\infty}^{\infty} \sin k\pi \sin(2t - k) = 0.$$

On the other hand, for $\cos 2\pi t$ the formula gives, again with $p = 2$,

$$\cos 2\pi t = \sum_{k=-\infty}^{\infty} \cos \frac{2\pi k}{2} \sin \left(2 \left(t - \frac{k}{2}\right)\right)$$

$$= \sum_{k=-\infty}^{\infty} \cos k\pi \sin(2t - k) = \sum_{k=-\infty}^{\infty} \sin(2t - 2k) - \sum_{k=-\infty}^{\infty} \sin(2t - 2k - 1),$$

which we might like to believe — at least both the series of sinc functions have period 1. Here’s a plot (really) of

$$\sum_{k=-50}^{50} \cos k\pi \sin(2t - k)$$

for some further encouragement.
Pretty impressive.

It’s easy to see what goes wrong with the sampling formula in the example of \( \sin 2\pi t \). The first step in the derivation is to periodize the Fourier transform, and for \( \sin 2\pi t \) this results in

\[
\mathcal{F} f \ast \text{III}_2 = \frac{1}{2i} (\delta_1 - \delta_{-1}) \ast \sum_{k=-\infty}^{\infty} \delta_{2k} \quad \text{(using the notation } \delta_a \text{ for } \delta(t - a))
\]

\[
= \frac{1}{2i} \sum_{k=-\infty}^{\infty} (\delta_{2k+1} - \delta_{2k-1}) = 0.
\]

The series telescopes and the terms cancel, so the sum is zero.

On the other hand, for \( \sin 2\pi t \), taking the sampling rate to be 2.1, just 0.1 beyond the bandwidth, here’s a plot of

\[
\sum_{k=-50}^{50} \sin \left( \frac{2\pi k}{2.1} \right) \text{sinc} \left( \frac{t - k}{2.1} \right).
\]

Some comfort in that, at least.
For \( \cos 2\pi t \) we find something different in applying the derivation of the sampling formula:

\[
\mathcal{F} g \ast \Pi_2 = \frac{1}{2} (\delta_1 + \delta_{-1}) \ast \sum_{k=-\infty}^{\infty} \delta_{2k} = \frac{1}{2} \sum_{k=-\infty}^{\infty} (\delta_{2k+1} + \delta_{2k-1}) = \sum_{k=-\infty}^{\infty} \delta_{2k+1}.
\]

The series telescopes and this time the terms add. So far so good, but is it true that \( \mathcal{F} g = \Pi_2 (\mathcal{F} g \ast \Pi_2) \), as needed in the second step of the derivation of the sampling formula? We are asking whether

\[
\Pi_2 \cdot \Pi_{2n+1} = \Pi_2 \cdot \sum_{n=-\infty}^{\infty} \delta_{2n+1} = \frac{1}{2} (\delta_1 + \delta_{-1}).
\]

This is correct — cutting off a \( \delta \) at the edge by a rectangle function results in half the \( \delta \), as in the above, and cutting off the \( \delta \)'s outside the support of \( \Pi_2 \) gives zero. See Section 5.7; here's one place where we needed the result from that section.

I chose \( \sin 2\pi t \) and \( \cos 2\pi t \) as simple examples illustrating the extreme cases in setting the sampling rate right at the bandwidth. The signal \( \sin 2\pi t \) is aliased to zero while \( \cos 2\pi t \) is reconstructed without a problem. I'm hesitant to attempt to formulate a general principle here. I think it's best to say, as I did earlier, that any particular endpoint problem should call for special considerations.

### 6.3.3. Aliasing in general

I sometimes think of aliasing as the natural phenomenon associated with the sampling formula. The sampling formula is a machine; you pass it data \( a_k, k = 0, \pm 1, \pm 2, \ldots \), that you've collected, measured sample values determined by a sampling rate \( q \), and it returns a signal

\[
g(t) = \sum_{k=-\infty}^{\infty} a_k \text{sinc} q \left( t - \frac{k}{q} \right)
\]

that has the sample values you gave it,

\[
g(n/q) = a_n.
\]

Remember, this is true for any sampling rate \( q \), because, to remind you,

\[
\text{sinc} q \left( \frac{n}{q} - \frac{k}{q} \right) = \text{sinc}(n-k) = \begin{cases} 1, & n = k, \\ 0, & n \neq k. \end{cases}
\]

It’s not the formula's fault if \( g(t) \) isn’t the signal you wanted, or thought you wanted. If you used too low a sampling rate, you got an alias — the right sample values, but that’s all you can say.
By the way, it’s also true that the sampling formula returns a bandlimited signal of bandwidth $q$. To see this, we have

$$\mathcal{F}g(s) = \sum_{k=-\infty}^{\infty} a_k \mathcal{F} \text{sinc} q \left( t - \frac{k}{q} \right)$$

$$= \sum_{k=-\infty}^{\infty} a_k e^{-2\pi i (k/q)s} \frac{1}{q} \Pi_q(s) = \frac{1}{q} \Pi_q(s) \sum_{k=-\infty}^{\infty} a_k e^{-2\pi i (k/q)s},$$

and you’re cutting off the series outside the interval $-q/2 < s < q/2$.

As in the examples we had above, aliasing occurs because of a breakdown in the derivation of the sampling formula caused by too low a sampling rate. To close out this discussion let’s look back to the signal $f(t)$ we had in Section 6.2 to get a picture of this breakdown for a generic bandlimited signal.

Here’s the plot of the magnitude of its Fourier transform (remember, it’s the magnitude that we typically have to plot).

The bandwidth is $p$. For a sampling rate $q < p$ here’s what happens when you periodize via $\mathcal{F} f \ast \Pi_q$. Think of this in two steps. First the spectrum is shifted by integer multiples of $q$:

We see that the spectrum is not shifted off itself. There are overlaps. Then the shifts are added:
The Fourier transform of \( f \) is
\[
\mathcal{F}f(s) = \sum_{k=-N}^{N} c_k \delta \left( s - \frac{k}{q} \right)
\]
and the spectrum goes from \(-N/q\) to \(N/q\). The sampling formula applies to \( f(t) \), and we can write an equation of the form
\[
f(t) = \sum_{k=-\infty}^{\infty} f(t_k) \text{sinc} p(t - t_k).
\]
It’s a question of what to take for the sampling rate and hence how to space the sample points.

We want to make use of the known periodicity of \( f(t) \). If the sample points \( t_k \) are a fraction of a period apart, say \( q/M \) for an \( M \) to be determined, then the values \( f(t_k) \) with \( t_k = kq/M, k = 0, \pm 1, \pm 2, \ldots \), will repeat after \( M \) samples. We’ll see how this collapses the interpolation formula.

To find the right sampling rate, \( p \), think about the derivation of the sampling formula, the first step being: “periodize \( \mathcal{F}f \).” The Fourier transform \( \mathcal{F}f \) is a train of \( \delta \)'s spaced \( 1/q \) apart and scaled by the coefficients \( c_k \). The natural periodization of \( \mathcal{F}f \) is to keep the spacing \( 1/q \) in the periodized version, essentially making the periodized \( \mathcal{F}f \) a version of \( \Pi_{1/q} \) with the coefficients scaled by the \( c_k \). We do this by convolving \( \mathcal{F}f \) with \( \Pi_p \) where \( p/2 \) is the midpoint between \( N/q \), the last point in the spectrum of \( \mathcal{F}f \), and the point \((N + 1)/q\), which is the next point \( 1/q \) away. Here’s a picture (unscaled \( \delta \)'s).

Thus we find \( p \) from
\[
\frac{p}{2} = \frac{1}{2} \left( \frac{N}{q} + \frac{N + 1}{q} \right) = \frac{(2N + 1)}{2q}, \quad \text{or} \quad p = \frac{2N + 1}{q}.
\]

We periodize \( \mathcal{F}f \) by \( \Pi_p \) (draw yourself a picture of this!), cut off by \( \Pi_p \), and then take the inverse Fourier transform. The sampling formula back in the time domain is
\[
f(t) = \sum_{k=-\infty}^{\infty} f(t_k) \text{sinc} p(t - t_k)
\]
with
\[
t_k = \frac{k}{p}.
\]
With our particular choice of \( p \) let’s now see how the \( q \)-periodicity of \( f(t) \) comes into play. Write
\[
M = 2N + 1,
\]
so that
\[ t_k = \frac{k}{p} = \frac{kq}{M}. \]
Then, to repeat what we said earlier, the sample points are spaced a fraction of a period apart, \( q/M \), and after \( f(t_0), f(t_1), \ldots, f(t_{M-1}) \) the sample values repeat; e.g., \( f(t_M) = f(t_0), f(t_{M+1}) = f(t_1) \), and so on. More succinctly,
\[ t_{k+k'M} = t_k + k'q, \]
and so
\[ f(t_{k+k'M}) = f(t_k + k'q) = f(t_k), \]
for any \( k \) and \( k' \). Using this periodicity of the coefficients in the sampling formula, the single sampling sum splits into \( M \) sums as
\[
\sum_{k=-\infty}^{\infty} f(t_k) \text{sinc}(p(t - t_k)) \\
= f(t_0) \sum_{m=-\infty}^{\infty} \text{sinc}(pt - mM) + f(t_1) \sum_{m=-\infty}^{\infty} \text{sinc}(pt - (1 + mM)) \\
+ f(t_2) \sum_{m=-\infty}^{\infty} \text{sinc}(pt - (2 + mM)) + \cdots + f(t_{M-1}) \sum_{m=-\infty}^{\infty} \text{sinc}(pt - (M - 1 + mM)).
\]

Those sums of sincs on the right are periodizations of sinc \( pt \) and, remarkably, they have a simple closed form expression. The \( k \)th sum is
\[
\sum_{m=-\infty}^{\infty} \text{sinc}(pt - k - mM) = \text{sinc}(pt - k) * \mathbb{III}_{M/p}(t) \\
= \frac{\text{sinc}(pt - k)}{\text{sinc}(\frac{1}{M}(pt - k))} = \frac{\text{sinc}(p(t - t_k))}{\text{sinc}(\frac{1}{q}(t - t_k))}. \\
\]
I’ll give a derivation of this at the end of this section. Using these identities, we find that the sampling formula to interpolate
\[ f(t) = \sum_{k=-N}^{N} c_k e^{2\pi ikt/q} \]
from \( 2N + 1 = M \) sampled values is
\[ f(t) = \sum_{k=0}^{2N} f(t_k) \frac{\text{sinc}(p(t - t_k))}{\text{sinc}(\frac{1}{q}(t - t_k))}, \]
where \( p = \frac{2N + 1}{q}, \ t_k = \frac{k}{p} = \frac{kq}{2N+1}. \)
This is the finite sampling theorem for periodic functions.

Notice, by the way, that the zeros of the denominator of the ratio of sincs are exactly those of the numerator, so we’re spared any embarrassment there. In fact, just as it should be,
\[
\frac{\text{sinc}(p(t_j - t_k))}{\text{sinc}(\frac{1}{q}(t_j - t_k))} = \delta_{jk} \quad \text{(Kronecker delta)},
\]
for
\[
\frac{\operatorname{sinc}(p(t_j - t_k))}{\operatorname{sinc}(q(t_j - t_k))} = \frac{\operatorname{sinc}(p(\frac{t_j}{p} - \frac{k}{p}))}{\operatorname{sinc}(q(\frac{t_j}{q} - \frac{k}{q}))} = \frac{\operatorname{sinc}(j - k)}{\operatorname{sinc}(\frac{1}{2N+1}(j - k))}.
\]

It might also be helpful to write the sampling formula in terms of frequencies. Thus, if the lowest frequency is \(\nu_{\text{min}} = \frac{1}{q}\) and the highest frequency is \(\nu_{\text{max}} = N\nu_{\text{min}}\), then
\[
f(t) = \sum_{k=0}^{2N} f(t_k) \frac{\operatorname{sinc}((2\nu_{\text{max}} + \nu_{\text{min}})(t - t_k))}{\operatorname{sinc}(\nu_{\text{min}}(t - t_k))}, \quad \text{where } t_k = \frac{kq}{2N+1}.
\]
The sampling rate is \(2\nu_{\text{max}} + \nu_{\text{min}}\). Compare this to the sampling rate \(> 2\nu_{\text{max}}\) for a general bandlimited function.

Here’s a simple example of the formula. Take \(f(t) = \cos 2\pi t\). There’s only one frequency, and \(\nu_{\text{min}} = \nu_{\text{max}} = 1\). Then \(N = 1\), the sampling rate is 3, and the sample points are \(t_0 = 0\), \(t_1 = 1/3\), and \(t_2 = 2/3\). The formula says
\[
\cos 2\pi t = \frac{\operatorname{sinc} 3t}{\operatorname{sinc} t} + \cos \left(\frac{2\pi}{3}\right) \frac{\operatorname{sinc}(3(t - \frac{1}{3}))}{\operatorname{sinc}(t - \frac{1}{3})} + \cos \left(\frac{4\pi}{3}\right) \frac{\operatorname{sinc}(3(t - \frac{2}{3}))}{\operatorname{sinc}(t - \frac{2}{3})}.
\]
Does this really work? I’m certainly not going to plow through the trig identities needed to check it! However, here’s a plot of the right-hand side. (Trust me; it’s really the right-hand side.)

Any questions? Ever thought you’d see such a complicated way of writing \(\cos 2\pi t\)?

**Periodizing sinc functions.** In applying the general sampling theorem to the special case of a periodic signal, we wound up with sums of sinc functions that we recognized (sharp-eyed observers that we are) to be periodizations. Then, out of nowhere, came a closed form expression for such periodizations as a ratio of sinc
functions. Here’s where this comes from, and here’s a fairly general result that covers it:

**Lemma.** Let \( p, q > 0 \) and let \( N = \lfloor \frac{pq}{2} \rfloor \), the largest integer \( \leq \frac{pq}{2} \). Then

\[
\text{sinc}(pt) \ast \text{III}_q(t) = \begin{cases} 
\frac{1}{pq} \frac{\sin((2N+1)\pi t/q)}{\sin(\pi t/q)} & \text{if } N < \frac{pq}{2}, \\
\frac{1}{pq} \left( \frac{\sin((2N-1)\pi t/q)}{\sin(\pi t/q)} + \cos(2\pi Nt/q) \right) & \text{if } N = \frac{pq}{2}.
\end{cases}
\]

Using the identity

\[
\sin((2N+1)\alpha) = \sin((2N-1)\alpha) + 2\sin\alpha\cos2\alpha
\]

we can write

\[
\frac{1}{pq} \frac{\sin((2N+1)\pi t/q)}{\sin(\pi t/q)} = \frac{1}{pq} \left( \frac{\sin((2N-1)\pi t/q)}{\sin(\pi t/q)} + 2\cos(2\pi Nt/q) \right),
\]

so, if pressed, we could combine the two cases into a single formula:

\[
\text{sinc}(pt) \ast \text{III}_q(t) = \frac{1}{pq} \left( \frac{\sin((2N-1)\pi t/q)}{\sin(\pi t/q)} + \left( 1 + \left\lceil \frac{pq}{2} \right\rceil - \left\lfloor \frac{pq}{2} \right\rfloor \right) \cos(2\pi Nt/q) \right) .
\]

Here, in standard notation, \( \lceil \frac{pq}{2} \rceil \) is the smallest integer \( \geq \frac{pq}{2} \) and \( \lfloor \frac{pq}{2} \rfloor \) is the largest integer \( \leq \frac{pq}{2} \).

Having written this lemma down so grandly I now have to admit that it’s really just a special case of the sampling theorem as we’ve already developed it, though I think it’s fair to say that this is only obvious in retrospect. The functions on the right-hand side of the equation are each bandlimited — obvious in retrospect — and \( \text{sinc}(pt) \ast \text{III}_q(t) \) is the sampling series. One usually thinks of the sampling theorem as going from a function to a series of sampled values, but it can also go the other way.

This admission notwithstanding, I still want to go through the proof. The distinction between the two cases comes from cutting off just beyond a \( \delta \) versus cutting off right at a \( \delta \). See Section 5.7. In practice, if \( q \) is given, it seems most natural to then choose \( p \) to make use of the first formula rather than the second.

In terms of sinc functions the first formula is

\[
\frac{2N+1}{pq} \frac{\sin((2N+1)\pi t/q)}{\sin(\pi t/q)} .
\]

The factor \( 2N+1 \) can be expressed in terms of \( p \) and \( q \) as

\[
2N + 1 = \left\lceil \frac{pq}{2} \right\rceil + \left\lfloor \frac{pq}{2} \right\rfloor .
\]

It’s also easy to extend the lemma slightly to include periodizing a shifted sinc function \( \text{sinc}(pt+b) \). We only write the formula in the first case, for \( N < \frac{pq}{2} \), which is what we used for the finite sampling formula:

\[
\text{sinc}(pt+b) \ast \text{III}_q(t) = \frac{2N+1}{pq} \frac{\sin(\frac{2N+1}{pq}(pt+b))}{\sin(\frac{1}{pq}(pt+b))} .
\]
One more thing. If \( p = q = 1 \), so that \( N = 0 \), the formula in the lemma gives

\[
\sum_{n=-\infty}^{\infty} \text{sinc}(t - n) = \text{sinc} t \ast \Pi_1(t) = 1.
\]

Striking. Still don’t believe it? Here’s a plot of

\[
\sum_{n=-100}^{100} \text{sinc}(t - n).
\]

Note the scale on the axes — how else was I supposed to display it. There’s a Gibbs-like phenomenon at the edges. This means there’s some issue with what kind of convergence is involved, which is the last thing you and I want to worry about.

We proceed with the proof, which will look awfully familiar. Take the case when \( N < \lfloor pq/2 \rfloor \) and take the Fourier transform of the convolution:

\[
\mathcal{F}(\text{sinc}(pt) \ast \Pi_q(t)) = \mathcal{F}(\text{sinc}(pt)) \cdot \mathcal{F}\Pi_q(t)
\]

\[
= \frac{1}{p} \Pi_p(s) \cdot \frac{1}{q} \Pi_1/q(s)
\]

\[
= \frac{1}{pq} \sum_{n=-N}^{N} \delta \left( s - \frac{n}{q} \right).
\]

See the figure below.

And now take the inverse Fourier transform:

\[
\mathcal{F}^{-1}\left( \frac{1}{pq} \sum_{n=-N}^{N} \delta \left( s - \frac{n}{q} \right) \right) = \frac{1}{pq} \sum_{n=-N}^{N} e^{2\pi i nt/q} = \frac{1}{pq} \frac{\sin(\pi(2N+1)t/q)}{\sin(\pi t/q)}.
\]
There it is. One reason I wanted to go through this is because it is another occurrence of the sum of exponentials and the identity

\[ \sum_{n=-N}^{N} e^{2 \pi i n t/q} = \frac{\sin(\pi (2N+1) t/q)}{\sin(\pi t/q)}, \]

which we’ve seen on other occasions. Reading the equalities backwards we have

\[ \mathcal{F}\left(\frac{\sin(\pi (2N+1) t/q)}{\sin(\pi t/q)}\right) = \mathcal{F}\left(\sum_{n=-N}^{N} e^{2 \pi i n t/q}\right) = \sum_{n=-N}^{N} \delta\left(s - \frac{n}{q}\right). \]

This substantiates the earlier claim that the ratio of sines is bandlimited, and hence we could have appealed to the sampling formula directly instead of going through the argument we just did. But who would have guessed it?

The second case is when \( N/q = p/2 \). This time cutting off with \( \Pi_p \) gives (à la Section 5.7)

\[ \frac{1}{pq} \left( \sum_{n=-(N-1)}^{N-1} \delta\left(s - \frac{n}{q}\right) + \frac{1}{2} (\delta_{N/q} + \delta_{-N/q}) \right). \]

The half \( \delta \)'s account for the cosine.

### 6.5. Appendix: Timelimited vs. Bandlimited Signals

Here’s a more careful treatment of the result that a bandlimited signal cannot be timelimited. We’ll actually prove a more general statement and perhaps I should have said that no interesting signal can be both timelimited and bandlimited because, precisely:

- Suppose \( f(t) \) is a bandlimited signal. If there is some interval \( a < t < b \) on which \( f(t) \) is identically zero, then \( f(t) \) is identically zero for all \( t \).

There’s a very cunning argument for this, due as far as I know to Dym and McKean from their book *Fourier Series and Integrals* mentioned back in Chapter 1. Here we go.

The signal \( f(t) \) is bandlimited so \( \mathcal{F}f(s) \) is identically zero, say, for \( |s| \geq p/2 \). The Fourier inversion formula says\(^{10}\)

\[ f(t) = \int_{-\infty}^{\infty} \mathcal{F}f(s)e^{2 \pi i st} ds = \int_{-p/2}^{p/2} \mathcal{F}f(s)e^{2 \pi i st} ds. \]

Suppose \( f(t) \) is zero for \( a < t < b \). Then for \( t \) in this range,

\[ \int_{-p/2}^{p/2} \mathcal{F}f(s)e^{2 \pi i st} ds = 0. \]

Differentiate with respect to \( t \) under the integral. If we do this \( n \) times, we get

\[ 0 = \int_{-p/2}^{p/2} \mathcal{F}f(s)(2\pi is)^n e^{2 \pi i st} ds = (2\pi i)^n \int_{-p/2}^{p/2} \mathcal{F}f(s)s^n e^{2 \pi i st} ds, \]

---

\(^{10}\)We assume the signal is such that Fourier inversion holds. You can take \( f(t) \) to be a Schwartz function, but some more general signals will do.
so that
\[ \int_{-p/2}^{p/2} \mathcal{F}f(s)s^n e^{2\pi ist} ds = 0. \]

Again, this holds for all \( t \) with \( a < t < b \); pick one, say \( t_0 \). Then
\[ \int_{-p/2}^{p/2} \mathcal{F}f(s)s^n e^{2\pi ist_0} ds = 0. \]

But now for any \( t \) (anywhere, not just between \( a \) and \( b \)) we can write
\[ f(t) = \int_{-p/2}^{p/2} \mathcal{F}f(s)e^{2\pi ist} ds = \int_{-p/2}^{p/2} \mathcal{F}f(s)e^{2\pi is(t-t_0)} e^{2\pi ist_0} ds \]
\[ = \int_{-p/2}^{p/2} \sum_{n=0}^{\infty} \frac{(2\pi i(t-t_0))^n}{n!} s^n e^{2\pi ist_0} \mathcal{F}f(s) ds \]
(\text{using the Taylor series expansion for } e^{2\pi is(t-t_0)})
\[ = \sum_{n=0}^{\infty} \frac{(2\pi i(t-t_0))^n}{n!} \int_{-p/2}^{p/2} s^n e^{2\pi ist_0} \mathcal{F}f(s) ds = \sum_{n=0}^{\infty} \frac{(2\pi i(t-t_0))^n}{n!} 0 = 0. \]

Hence \( f(t) \) is zero for all \( t \).

The same argument \textit{mutatis mutandis} will show:

- If \( f(t) \) is timelimited and if \( \mathcal{F}f(s) \) is identically zero on any interval \( a < s < b \), then \( \mathcal{F}f(s) \) is identically zero for all \( s \).

Then \( f(t) \) is identically zero, too, by Fourier inversions.

\textbf{Remark 1, for eager seekers of knowledge.} This bandlimited vs. timelimited result is often proved by establishing a relationship between timelimited signals and analytic functions (of a complex variable) and then appealing to results from the theory of analytic functions. That connection opens up an important direction for applications of the Fourier transform, but it involves a considerable amount of background and the direct argument makes this approach unnecessary.

\textbf{Remark 2, for overwrought math students and careful engineers.} Where in the preceding argument did we use that \( p < \infty \)? It’s needed in switching integration and summation, in the line
\[ \int_{-p/2}^{p/2} \sum_{n=0}^{\infty} \frac{(2\pi i(t-t_0))^n}{n!} s^n e^{2\pi ist_0} \mathcal{F}f(s) ds \]
\[ = \sum_{n=0}^{\infty} \frac{(2\pi i(t-t_0))^n}{n!} \int_{-p/2}^{p/2} s^n e^{2\pi ist_0} \mathcal{F}f(s) ds. \]

The theorems that tell us “the integral of the sum is the sum of the integral” require as an essential hypothesis that the series converges uniformly.\(^{11}\) In the sum-and-integral expression, above, the variable \( s \) ranges over a finite interval, from \(-p/2\) to \(+p/2\). Over such a finite interval the series for the exponential converges uniformly,

\(^{11}\)Recall that “uniformly” means, loosely, that if we plug a particular value into the converging series, we can estimate the rate at which the series converges \textit{independent} of that particular value. We can make “uniform” estimates, in other words. We saw this sort of thing in the notes on convergence of Fourier series.
essentially because the terms can only get so big — so they can be estimated uniformly — when $s$ can only get so big. We can switch integration and summation in this case. If, however, we had to work with

$$\int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \frac{(2\pi i(t-t_0))^n}{n!} s^n e^{2 \pi ist_0} \mathcal{F}f(s) \, ds,$$

i.e., if we did not have the assumption of bandlimitedness, then we could not make uniform estimates for the convergence of the series, and switching integration and summation would not be justified.

It’s not only unjustified, it’s really wrong. If we could drop the assumption that the signal is bandlimited, we’d be buying into the statement: if $f(t)$ is identically zero on an interval, then it’s identically zero. Think of the implications of such a dramatic statement. In a phone conversation, if you paused for a few seconds to collect your thoughts, your signal would be identically zero on that interval of time, and therefore you would have nothing to say at all, ever again. Be careful.

6.6. Appendix: Linear Interpolation via Convolution

Remember linear interpolation from Section 6.1 as a grown-up version of connect-the-dots? There’s a nice way to describe how to do it and to get a formula using the technology of III’s and convolution. No Fourier transforms, and everything happens in the time domain. You’ll enjoy this.

Suppose we sample a function $f(t)$ at a sampling rate $p$. The sample points, in time, are $t_k = k/p$, spaced $1/p$ apart. The sampled function is

$$f(t) \Pi_{1/p}(t) = \sum_{k=-\infty}^{\infty} f(t_k) \delta(t - k/p) = \sum_{k=-\infty}^{\infty} f(t_k) \delta(t - t_k).$$

This is meant to include the possibility of a finite number of samples, where the $f(t_k)$ are zero beyond a certain point. Now bring in the triangle function

$$\Lambda(t) = \begin{cases} 
1 - |t|, & |t| \leq 1, \\
0, & |t| \geq 1, 
\end{cases}$$

and its scaled version

$$\Lambda_{1/p}(t) = \Lambda(pt) = \begin{cases} 
1 - |pt|, & |t| \leq 1/p, \\
0, & |t| \geq 1/p, 
\end{cases}$$

languishing since Chapter 3:
Convolve this with \( f(t) III_{1/p}(t) \), forming

\[
L(t) = \Lambda_{1/p}(t) * (f(t) III_{1/p}(t)) = \sum_{k=-\infty}^{\infty} f(t_k) \Lambda_{1/p}(t - t_k)
\]

\[
= \sum_{k=-\infty}^{\infty} f\left(\frac{k}{p}\right) \Lambda\left(p \left(t - \frac{k}{p}\right)\right).
\]

That’s the linear interpolation! \( \Lambda_{1/p} \) is playing the role of the sinc.

Below is a picture of

\[
L(t) = 3\Lambda_{1/p}(t - 1/p) + 2.5\Lambda_{1/p}(t - 2/p) - 1\Lambda_{1/p}(t - 3/p) + 1.5\Lambda_{1/p}(t - 4/p)
\]

showing how the triangles combine. Because \( \Lambda \)'s vanish at their endpoints, the first and last interpolated points are on the \( t \)-axis (sample value 0), so you have to take account of that.

To see that the formula is producing what you think it is, first check the sample points. \( \Lambda \) has the magical property (like sinc)

\[
\Lambda(n - k) = \begin{cases} 
1, & n = k \\
0, & n \neq k 
\end{cases}
\]

so

\[
L(t_n) = \sum_{k=-\infty}^{\infty} f(t_k) \Lambda\left(p \left(\frac{n}{p} - \frac{k}{p}\right)\right) = \sum_{k=-\infty}^{\infty} f(t_k) \Lambda(n - k) = \begin{cases} 
f(t_n), & n = k \\
0, & n \neq k 
\end{cases}.
\]

Next, as a sum of piecewise linear functions \( L(t) \) is also piecewise linear. The only question is whether, in the sum defining \( L(t) \), two adjacent \( \Lambda \)'s for two adjacent sample points combine to give you a single segment joining the two sample points. Yes, that’s what happens and I’ll leave the verification to you. This is enough to be sure that \( L(t) \) is the very same connect-the-dots linear interpolation that you drew when you were young.

What about the even simpler sample-and-hold interpolation? That’s a sum of shifted rectangles — also a convolution. I’ll let you write that down, giving a derivation modeled on the one we just did.
6.7. Appendix: Lagrange Interpolation

Finally, a nod to polynomial interpolation, a separate topic and a large one. Going way back, it was desirable to find readily computable approximations of complicated functions by simple functions, particularly in applications arising from solutions to differential equations. Even as computational power has increased, this is still of some interest, though much has changed to say the least. The next step up from linear interpolation is to use polynomials of higher degree, and this has been the classic way to interpolate and approximate. One old method, presented here for your general background and know-how, is due to J.-L. Lagrange.

Suppose we have \( n \) points \( t_1, t_2, \ldots, t_n \) at which we have made \( n \) measurements. We want a polynomial of degree \( n - 1 \) that assumes the measured values at the respective \( t \)'s (2 points, a line, degree 1; 3 points, a quadratic, degree 2; etc.). For this, start with an \( n \)th degree polynomial that vanishes exactly at the \( t_k \). This is \( p(t) = (t - t_1)(t - t_2)\cdots(t - t_n) \).

Next put

\[
p_k(t) = \frac{p(t)}{t - t_k}.
\]

Then \( p_k(t) \) is a polynomial of degree \( n - 1 \); we divide out the factor \( (t - t_k) \) and so \( p_k(t) \) vanishes at the same points as \( p(t) \) except at \( t_k \). Next consider the quotient

\[
\frac{p_k(t)}{p_k(t_k)}.
\]

This is again a polynomial of degree \( n - 1 \). The key property is that \( p_k(t)/p_k(t_k) \) vanishes at the sample points \( t_j \) except at the point \( t_k \) where its value is 1; i.e.,

\[
\frac{p_k(t_j)}{p_k(t_k)} = \begin{cases} 1, & j = k, \\ 0, & j \neq k. \end{cases}
\]

There's the magical property again, just like sinc and just like \( \Lambda \).

To interpolate/approximate a function by a polynomial (to fit a curve through a given set of points) we just scale and add. Suppose we have a function \( f(t) \) and we want a polynomial that has values \( f(t_1), f(t_2), \ldots, f(t_n) \) at the points \( t_1, t_2, \ldots, t_n \). We get this by forming the sum

\[
p(t) = \sum_{k=1}^{n} f(t_k) \frac{p_k(t)}{p_k(t_k)}.
\]

This does the trick. It is known as the Lagrange interpolation polynomial.

The shifted, scaled sinc functions \( \text{sinc} p(t - t_k) \) are the analogs for Fourier interpolation of the \( p_k(t)/p_k(t_k) \). As with linear interpolation and sample-and-hold, and unlike the sinc interpolation formula, we’re not aiming to reconstruct exactly all the values of \( f(t) \) from a set of sample values. On the other hand, there are no restrictions, such as being bandlimited. The aim is to approximate \( f(t) \) by a

\[12\] Joseph-Louis Lagrange made many important contributions to mathematics and its applications. Engineers often first run into his work through the method of Lagrange multipliers in studying optimization problems and are then later tormented in a class on ODEs by his method of variation of parameters. If you took an advanced mechanics course, you probably also learned about Lagrangians and his formulation of mechanics through variational principles. All told, quite remarkable.
polynomial that has the same values as $f(t)$ at a prescribed set of points. Also, for
Lagrange interpolation we’re not assuming that the $t_k$ are evenly spaced, as we do
in the sampling theorem. Moreover, unlike in the sinc interpolation formula, there
are only a finite number of sample points.

As an aside, let me mention that the general topic of orthogonal polynomials
(which the Lagrange polynomials are not, actually) is a vast masterpiece of classical
mathematics. (“Orthogonal” means with respect to an inner product defined via
an integral.) Sometimes such polynomials appear in the service of applications,
including approximations, and sometimes they have their own intrinsic interest.
You saw Legendre polynomials in Problem 1.19. Various families of polynomials
and related functions are built into the standard mathematical software packages.

Experiment.


There are many theorems, of various degrees of generality, on the conver-
gegence of a series of sinc functions, say a series of the form

$$
\sum_{n=-\infty}^{\infty} a_n \operatorname{sinc}\left(\frac{p(t - \frac{n}{p})}{p}\right).
$$

A basic result is on \textit{absolute convergence}, meaning

$$
\sum_{n=-\infty}^{\infty} \left| a_n \operatorname{sinc}\left(\frac{p(t - \frac{n}{p})}{p}\right)\right| < \infty \quad \text{for any } t.
$$

(Look up why this is a property you’d like to have.) In this problem you’ll
show that this property holds for every real number $t$ if and only if

$$
\sum_{n \neq 0} \left| \frac{a_n}{n} \right| < \infty.
$$

Fix a number $t$. If $t = \frac{m}{p}$ for some integer $m$, then the series reduces to a
single term and there’s nothing more to say about its convergence. So we can
assume that $t$ is not of this form, and that’s important for the argument.

(a) Derive the estimate

$$
\left| a_n \operatorname{sinc}\left(\frac{p\left( t - \frac{n}{p}\right)}{p}\right)\right| \leq \left| \frac{a_n}{n} \right| \frac{1}{\pi p \left| \frac{t}{n} - \frac{1}{p} \right|}.
$$
and deduce that if \(|n|\) is sufficiently large, then
\[
|a_n \text{sinc} \left(p \left(t - \frac{n}{p}\right)\right)| \leq \frac{2}{\pi} |\frac{a_n}{n}|.
\]
(The constant \(2/\pi\) isn’t important, just that there’s some constant for which the inequality holds.) Conclude that the condition on the sum of the \(a_n\)’s implies the absolute convergence of the sinc series.

(b) Next suppose that the sinc series converges absolutely. Why does it follow that
\[
\sum_{n \neq 0} \frac{|a_n|}{\pi p |t - \frac{n}{p}|} < \infty?
\]

(c) Show that by taking \(|n|\) large enough we have
\[
\frac{|a_n|}{\pi |n| |p| \frac{t}{p} - \frac{1}{p}} \geq \frac{1}{2\pi} \frac{|a_n|}{n}.
\]
Conclude that the absolute convergence implies the condition on the sum of the \(a_n\).

6.2. A surprising identity

The sinc function satisfies
\[
\sum_{n = -\infty}^{\infty} \text{sinc}^2(t - n) = 1.
\]

Show this in three different ways:

(a) Periodize \(\text{sinc}^2 t\) to have period 1 and find its Fourier series. (Recall the relationship between the Fourier coefficients and the Fourier transform.)

(b) Use the sampling formula applied to \(\text{sinc}(t - x)\) regarded as a function of \(x\) with \(t\) fixed. Show that this leads to the formula
\[
\text{sinc}(t - x) = \sum_{n = -\infty}^{\infty} \text{sinc}(t - n) \text{sinc}(x - n).
\]
This is itself an interesting formula.

(c) Use the Poisson summation formula applied to \(\text{sinc}^2(t - x)\) regarded as a function of \(x\) with \(t\) fixed.

6.3. Energy of a bandlimited signal

The energy of a signal \(g(t)\) is the integral
\[
\int_{-\infty}^{\infty} |g(t)|^2 \, dt.
\]
Suppose that \(g(t)\) is bandlimited with
\[
\mathcal{F}g(s) = 0, \quad |s| \geq \frac{1}{2}.
\]
Express the energy of \(g(t)\), in terms of the sample values of \(g(t)\) at the integers, \(g(n), n = 0, \pm 1, \pm 2, \ldots\) Hint: Use the Fourier series of the periodization of \(\mathcal{F}g(s)\) of period 1.
6.4. Filtering for interpolation

Suppose you have sampled a signal \( f(t) \) at intervals of one unit to obtain \( f_{\text{sampled}}(t) \), shown below:

The arrows represent \( \delta \)-functions of different strengths at 0, 1, and 2.

Sketch the following interpolations of \( f(t) \):

(a) \( f_1(t) = \mathcal{F}^{-1} \{ \text{sinc}(s) \mathcal{F} f_{\text{sampled}}(s) \} \).
(b) \( f_2(t) = \mathcal{F}^{-1} \{ e^{-i\pi s} \text{sinc}(s) \mathcal{F} f_{\text{sampled}}(s) \} \).
(c) \( f_3(t) = \mathcal{F}^{-1} \{ \text{sinc}^2(s) \mathcal{F} f_{\text{sampled}}(s) \} \).

These filters correspond to common interpolation methods: nearest neighbor, zero-order hold, and linear interpolation.

6.5. And yet it flies\textsuperscript{14}

Watch the video “chopper” available at

http://www.youtube.com/watch?v=bZCUB_BiY_4.

The phenomenon you are observing can be attributed to aliasing. Suppose the frame rate of the video camera is \( R_1 \); i.e., the camera is taking \( R_1 \) still shots per second; and the rotation rate of the main rotor is \( R_2 \) rotations per second.

(a) Suppose \( R_1 \) is fixed and the chopper has 5 rotor blades. What values of \( R_2 \) (expressed in terms of \( R_1 \)) cause the rotor to appear stationary as in the video?

(b) In part (a), we assumed that the chopper has 5 rotor blades. Is this assumption valid? If you had seen 6 blades in the video, how many blades do you think the chopper has?

6.6. Handel’s Hallelujah\textsuperscript{15}

In this problem we will explore the effects of sampling with or without anti-aliasing filters. There can be a significant distortion of music due to aliasing if we sample slower than twice the highest frequency. However, if we can suppress the high frequency components before sampling, we can possibly

\textsuperscript{14}From T. John.
\textsuperscript{15}From Logi Vidarsson.
avoid this distortion. In this problem we will use an anti-aliasing filter \( H(s) \) whose Fourier transform is shown below. \( H(s) \) is available to download in the MATLAB file \texttt{anti-aliasing.mat}, which contains \( H(s) \) in the vector \( Hs \).

![Figure 6.1. Anti-aliasing filter.](image)

Built into MATLAB is a snippet of Handel’s “Hallelujah Chorus”. You load it into the workspace by typing

\[
\text{load handel}
\]

This loads two variables into the workspace: \( y \), which contains about 8 seconds of Handel’s “Hallelujah Chorus”, and \( Fs \), which is the sampling frequency used.

Finally, here is the problem. Resample the snippet of Handel’s “Hallelujah Chorus” down to a sampling frequency of \( f_s = 4,096 \) Hz that should be half of the original sampling frequency.

Now apply the anti-aliasing filter to Handel’s “Hallelujah Chorus” so that you cut off all frequencies higher than 2,048 Hz, and then resample down to \( f_s = 4,096 \) Hz. Is there any audible difference between the two versions? Why or why not. Discuss audible differences you heard or did not hear.

\textit{Hints}: To resample at half the sampling rate, you can use

\[
x_{\text{half}} = x(1:2:length(x));
\]

Remember to adjust the sampling rate correctly when you use \texttt{sound} or \texttt{wavwrite}.

Recall that you can use \texttt{fft} to take the Fourier transform, and \texttt{ifft} to take the inverse Fourier transform. \( Hs \) has been arranged in the same way MATLAB’s \texttt{fft} returns Fourier transforms.

To evaluate \( H(s)X(s) \) try using the \( .* \) operator.
6.7. Nyquist rate and spectral islands

The signal $f(t)$ has the Fourier transform $F(s)$ as shown below.

![Fourier transform diagram]

The Nyquist frequency is $2B_2$ since the highest frequency in the signal is $B_2$. The sampling theorem tells us that if we sample above the Nyquist rate, no aliasing will occur. Is it possible, however, to sample at a lower frequency in this case and not get aliasing effects? If it is possible, then explain how it can be done and specify at least one range of valid sampling frequencies below the Nyquist rate that will not result in aliasing. If it is not possible, explain why not.

6.8. Natural sampling

Suppose the signal $f(t)$ is bandlimited with $Ff(s) = 0$ for $|s| \geq B$. Instead of sampling with a train of $\delta$’s we sample $f(t)$ with a train of very narrow pulses. The pulse is given by a function $p(t)$, we sample at a rate $T$, and the sampled signal then has the form

$$g(t) = f(t) \left( \sum_{k=-\infty}^{\infty} Tp(t - kT) \right).$$

(a) Is it possible to recover the original signal $f$ from the signal $g$?
(b) If not, why not. If it is possible, what conditions on the parameters $T$ and $B$ and on the pulse $p(t)$ make it possible?

6.9. Nonuniform sampling: A variation on some earlier problems

We wish to sample and reconstruct signals whose spectrum is known to be nonzero within the frequency bands $0 \leq s \leq 1$ and $2 \leq s \leq 3$. (Suppose the signals have been phase shifted so that there are no negative frequencies.) For example, a signal $f(t)$ whose spectrum $F(s) = \mathcal{F}f(s)$ is as shown below.

![Spectrum diagram]
If we sum shifted copies of \( F(s) \), as in forming the periodization

\[
\sum_{n=-\infty}^{\infty} F(s - n),
\]

we may get some overlapping, and the way the shifts overlap depends on if we shift by even or odd integers.

(a) On one set of axes sketch

\[
\sum_{n \text{ odd}} F(s - n) = F(s - 1) + F(s + 1) + F(s - 3) + F(s + 3) + \cdots.
\]

On a second set of axes sketch

\[
\sum_{n \text{ even}} F(s - n) = F(s) + F(s - 2) + F(s + 2) + F(s - 4) + F(s + 4) + \cdots.
\]

On a third set of axes sketch the full sum

\[
\sum_{n=-\infty}^{\infty} F(s - n).
\]

If we wanted to reconstruct the signal \( f(t) \) from sampled values and use uniform sampling (i.e., applying the usual sampling theorem, where the sampling points are evenly spaced), we would have to use a sampling rate of at least 3. Instead, we consider a modified sampling scheme that uses a different, nonuniform sampling pattern. The spacing of the sample points is determined by choosing a value \( t_0 \) from among \( \{1/4, 1/2, 3/4\} \) — you will be asked which one(s) work. Let

\[
p(t) = \delta(t) + \delta(t - t_0) \quad \text{and} \quad f_{\text{sampled}}(t) = f(t) \left( \sum_{n=-\infty}^{\infty} p(t - n) \right).
\]

The train of \( \delta \)'s for \( t_0 = 1/4 \) is shown below. The other choices of \( t_0 \) have a similar pattern. There are two samples per second, though not evenly spaced within a one-second interval, so we might still say that the sampling rate is 2.

(b) Show that

\[
\mathcal{F} f_{\text{sampled}}(s) = \sum_{n=-\infty}^{\infty} \mathcal{F} p(n) F(s - n).
\]

*Hint:* You did this in the problem on natural sampling: write \( \sum_n p(t - n) = (p \ast \text{III})(t) \).
So the Fourier transform of $f_{\text{sampled}}(t)$ is a sum of shifted copies of $F(s)$, and each copy is scaled by $F_p(n)$, the Fourier transform of the pulse at integer values.

The possibility of reconstructing $f(t)$ from its samples depends on whether the scaled shifts interfere with each other, i.e., whether you can isolate the original spectrum $F(s)$, and this depends on the value you take for $t_0$ as it affects $F_p(n)$.

(c) Which values $t_0$ chosen from $\{1/4, 1/2, 3/4\}$ will allow you to recover the signal?

6.10. Oversampling

Let $f(t)$ be a bandlimited signal with spectrum contained in the interval $-1/2 < s < 1/2$. Suppose you sample $f(t)$ at intervals of 1/2 (that is, at twice the Nyquist rate), to obtain

$$f_{\text{sampled}}(t) = \frac{1}{2} \{1/2(t) f(t).$$

(a) Qualitatively explain why the following equation is correct:

$$f(t) = F^{-1} \{K(s) F f_{\text{sampled}}(s)\}$$

where $K(s)$ is defined by

$$K(s)$$

\[ \begin{array}{c|c|c|c|c}
-1 & -1/2 & 1/2 & 1 \\
\hline
\end{array} \]

(b) Show that you can reconstruct $f(t)$ by

$$f(t) = \frac{1}{2} \sum_{n=-\infty}^{\infty} f \left( \frac{n}{2} \right) k \left( t - \frac{n}{2} \right)$$

where $k(y) = \frac{\cos(\pi y) - \cos(2\pi y)}{\pi y^2}.$

You may use the fact that

$$K(s) = F \left\{ \cos(\pi t) - \cos(2\pi t) \right\}.\frac{\pi^2 t^2}{\pi^2 t^2}.$$

(c) Describe an advantage the reconstruction formula in part (b) has over the usual sinc interpolation formula, say in terms of the accuracy of the series if one only uses a finite number of terms.

6.11. Can undersampling be overcome?\(^\text{16}\)

Let $f(t)$ be a bandlimited signal whose Fourier transform satisfies $|Ff(s)| = 0$ for $|s| \geq 1$. According to the sampling theorem, one has to sample $f(t)$ with $\{1/2(t)\}$ to reproduce the signal without aliasing.

\(^{16}\)From T. John.
You try to test this out in the lab, but there is something wrong with the ideal sampler — it can only sample using $\Pi_1(t)$. This will inevitably cause aliasing, but you think you can devise a scheme to somehow reconstruct $f(t)$ without aliasing.

This problem explores how this can be done by using this faulty sampler and some filter $h(t)$.

(a) Let $h(t) = -1/(\pi it)$. If $g(t) = (f \star h)(t)$, express $G(s)$ in terms of $F(s)$ only. Hint: Consider the cases $s = 0$, $s > 0$, and $s < 0$.

(b) Suppose you sample $f(t)$ with $\Pi_1(t)$ to yield $y(t)$. For the case of $0 < s < 1$, express $Y(s)$ in terms of $F(s)$ and $F(s - 1)$ only.

(c) Suppose you sample $g(t)$ with $\Pi_1(t)$ to yield $x(t)$. For the case of $0 < s < 1$, express $X(s)$ in terms of $F(s)$ and $F(s - 1)$ only.

(d) Using the two equations you have from parts (b) and (c), show how you might reconstruct $f(t)$ without aliasing.

### 6.12. Downconversion

A common problem in radio engineering is “downconversion to baseband.” Consider a signal $f(t)$ whose spectrum $F_f(s)$ satisfies

$$F_f(s) = 0, \quad |s - s_0| \geq B.$$  

To downconvert $F_f(s)$ to baseband means to move the spectrum so that it is centered around 0. Devise a strategy to downconvert using convolution with an appropriate $\Pi$ and a single ideal low-pass filter. What is the new signal in terms of the old? (Note that you can assume $s_0 > 2B$.)

### 6.13. Sampling and (single-sided) modulation

Communication channels are limited, and everyone wants to get his or her message through. In this problem you’ll see how two signals can be combined, the combination sampled, and then the individual signals reconstructed from the samples of the combined signal.

Suppose $g(t)$ and $h(t)$ are bandlimited signals with bandwidth $p$; i.e. (using uppercase letters to denote Fourier transforms),

$$G(s) \equiv 0, \quad H(s) \equiv 0 \quad \text{for } |s| \geq \frac{p}{2}.$$  

Form the signal

$$f(t) = e^{-2\pi i (p/2)t}g(t) + e^{2\pi i (p/2)t}h(t) = e^{-\pi ipt}g(t) + e^{\pi ipt}h(t).$$

Note the minus sign in the first complex exponential and the plus sign in the second.

We’re going to recover $g(t)$ and $h(t)$ in terms of samples of $f(t)$.

(a) What is $F(s) = F_f(s)$ in terms of $G(s)$ and $H(s)$? Is $f(t)$ bandlimited? What is its bandwidth?
Suppose, for simplicity, that plots of $|G(s)|$ and $|H(s)|$ look like

![Diagram showing plots of $|G(s)|$ and $|H(s)|$]

Sketch a plot of $|F(s)|$.

(b) Explain why

$$H(s) = \Pi_p(s) \cdot (\Pi_{2p} * F)(s + \frac{p}{2}) = \Pi_p(s) \cdot \left(\Pi_{2p}(s) \star F\left(s + \frac{p}{2}\right)\right),$$

and use this to derive the interpolation formula

$$h(t) = \frac{1}{2} \sum_{k=-\infty}^{\infty} (-i)^k f\left(\frac{k}{2p}\right) \text{sinc} \left(p\left(t - \frac{k}{2p}\right)\right).$$

Thus the signal $h(t)$ is reconstructed from the samples of $f(t)$.

You are not asked to show this but, likewise,

$$G(s) = \Pi_p(s) \cdot (\Pi_{2p} * F)(s - \frac{p}{2}),$$

and using this, one obtains in the same manner the result

$$g(t) = \frac{1}{2} \sum_{k=-\infty}^{\infty} i^k f\left(\frac{k}{2p}\right) \text{sinc} \left(p\left(t - \frac{k}{2p}\right)\right).$$

We see that both $g(t)$ and $h(t)$ can be interpolated from the same samples of $f(t)$!

6.14. **Demodulation by sampling**\(^{17}\)

(a) Suppose a signal $x_0(t)$ is bandlimited so that $X_0(s) = 0$ for $|s| \geq W$. It is then modulated to carrier frequency $s_0$: $x(t) = x_0(t)e^{2\pi i s_0 t}$. Assume $W > 0$ and $s_0 \gg W$ are known. Show that the original signal $x_0(t)$ can be recovered from an ideal sampler: $y(t) = x(t) \cdot \Pi_T(t)$. What is the largest $T$ that can be used?

(b) Ideal samplers cannot be built in practice. For instance, a more realistic sampling device might provide the average value of the signal over each sample period. That is, for signal $x_0(t)$ and a sampling period of $T$, rather than the sample values being $v_k = x_0(kT)$, we get

$$v_k = \frac{1}{T} \int_{kT - \frac{T}{2}}^{kT + \frac{T}{2}} x_0(t) dt.$$

\(^{17}\)From Adam Wang.
From the samples $w(t) = \sum_{k=-\infty}^{\infty} v_k \delta(t - kT)$, how can we recover the original signal $x_0(t)$? State any assumptions you make about $x_0(t)$ and $T$.

(c) Will the sampling device in (b) work for your demodulating scheme in (a)? Why might this be a bad idea?

6.15. Sampling oscilloscope, warmup

Let $f(t) = \cos 2\pi t$. Suppose we sample $f(t)$ at a rate of $2/3$ Hz and then interpolate using a low-pass filter with cutoff frequency $2/3$ Hz. What signal, $g(t)$, is the result? Sketch $f(t)$ and $g(t)$ on the same axes and comment on what you see. Is $g(t)$ an alias of $f(t)$ for this sampling rate?

The process illustrated in this problem is the basis of the sampling oscilloscope, below.

6.16. Sampling oscilloscope\textsuperscript{18}

Sometimes signals change much faster than electronic devices can sample in order to reconstruct or display the signal on an oscilloscope. However, if a signal is bandlimited and periodic and if we take regular samples spaced somewhat more than the period (a sampling rate that is lower than the Nyquist rate), we can recover a stretched version of the signal. This is the principle of the sampling oscilloscope.

\textsuperscript{18}From A. Oppenheim and A. Willsky via Aakanksha Chowdhery.
In the figures above, drawn in the time domain, the periodic signal \( x(t) \), with period \( T \), is sampled every \( T + \Delta \) producing the sampled function \( v(t) = x(t) \Pi_{T+\Delta}(t) \). The samples are then interpolated by convolving with a scaled sinc function (an ideal low-pass filter), \((b/(T+\Delta)) \text{sinc}(t/(T+\Delta))\), to get \( y(t) \), a stretched version of the original signal \( x(t) \). You will see how to choose the parameters \( \Delta \) and \( b \) to obtain \( y(t) \) by examining the process in the frequency domain.

(a) Consider a periodic signal \( x(t) = \alpha + \beta e^{-2\pi iT}/T + \gamma e^{2\pi iT}/T \), which is periodic with period \( T \).

(i) Find the Fourier transform \( X(s) \) of \( x(t) \). Sketch it for \( \alpha = 2, \beta = \gamma = 1 \).

(ii) Find the Fourier transform \( V(s) \) of \( v(t) \). Sketch it for \( \alpha = 2, \beta = \gamma = 1 \).

(iii) How should \( \Delta \) be chosen so that \( Y(s) \) looks like a compressed version of \( X(s) \) (sketch \( Y(s) \))? How should \( b \) be chosen so that \( y(t) \) is a stretched version of \( x(t) \), i.e., \( y(t) = x(at) \) for some \( a \) (which you will find)?

(b) Consider a signal \( x_0(t) \) with spectrum \( X_0(s) \), drawn below, where \( X_0(s) = 0 \) for \(|s| > W \). Define \( x(t) \) as the periodization of \( x_0(t) \); i.e., \( x(t) = \sum_{n=-\infty}^{\infty} x_0(t - nT) = (x_0 * \Pi_T)(t) \).

(i) Sketch each of the following, using \( T = 1, \Delta = 1/9, \) and \( W = 4 \).

(ii) Determine what constraint applies to \( \Delta \) so that \( y(t) \) is a perfect reconstruction of a scaled version of \( x(t) \). \( \text{(Hint: Surprisingly, this method does not depend on the period of the signal, only the bandwidth \( W \).)} \)
6.17. **Compensating for distortions by adjusting the sampling rate**

You are given a bandlimited, real signal \( f(t) \) with bandwidth \( B \). We will also assume that \( f(t) \) is nonnegative.

\[ \mathcal{F} f(s) \]

The signal is sampled at a rate \( R = 1/T \) to obtain the samples \( f(nT) \), \( n = 0, \pm 1, \pm 2, \ldots \). Some discrete time processing is subsequently done on these sample values, which unintentionally ends up squaring the sample values. Beyond your control, your interpolator has access to \( |f(nT)|^2 \) instead of \( f(nT) \), but no information is lost since \( f \) is nonnegative. The whole system operates the usual way: it constructs

\[ f_s(t) = \sum_n |f(nT)|^2 \delta(t - nT) \]

and filters \( f_s(t) \) using an ideal low-pass filter of bandwidth \( 1/2T \) to obtain \( g(t) \).

(a) Suppose the sampling rate \( R \) is equal to the Nyquist rate for \( f \) (i.e., \( R = 2B \)). Sketch the spectrum of the output \( g(t) \).

(b) How will you pick the sampling rate \( R \) so that \( f(t) \) can be recovered from \( g(t) \)?

6.18. **An infinite product for \( \text{sinc} \)**

Let \( f(x) \) be the infinite product of cosines defined as

\[ f(x) = \cos \frac{\pi x}{2} \cos \frac{\pi x}{4} \cos \frac{\pi x}{8} \ldots = \prod_{k=1}^{\infty} \cos \frac{\pi x}{2^k}. \]

(a) Find \( f(0) \).

(b) Find \( f(\pm 1), f(\pm 2), \ldots \). Generalize the pattern to find \( f(n) \) when \( n \) is a nonzero integer.

(c) Argue that \( f \) has bandwidth \( 1/2 \). Recall that the largest frequency in the product of cosines \( \cos(2\pi \nu_1 t) \cos(2\pi \nu_2 t) \) is \( \nu_1 + \nu_2 \).

(d) Conclude that \( f(x) = \text{sinc} \, x \).

---

19 From A. Siripuram.
20 From J. Gill.

Consider an interpolation of a signal $x(t)$ from its samples $x(n)$ in the form

$$x(t) = \sum_{n=-\infty}^{\infty} x(n)s_n(t).$$

In the case of Nyquist-Shannon interpolation we have $s_n(t) = \text{sinc}(t - n)$. The interpolation process is called *stable in the energy sense* if small errors in the sample values do not lead to large errors in the interpolated signal. To make this precise, suppose we have an error $e_n$ in recording the $n$th sample $x(n)$. Then the overall error is given by

$$e(t) = \sum_{n=-\infty}^{\infty} e_n s_n(t).$$

The interpolation process is stable in the energy sense if

$$\int_{-\infty}^{\infty} |e(t)|^2 dt \leq C \sum_{n=-\infty}^{\infty} |e_n|^2,$$

for some constant $C$.

(a) Show that the Nyquist-Shannon interpolation is stable in the energy sense. (For this, use Parseval’s theorem and the fact that the functions $\{\text{sinc}(t - n)\}$ for $n = 0, \pm 1, \pm 2, \ldots$ are orthonormal.)

Note that for energy-stable interpolation, even though the overall energy in the error $e(t)$ is not too large, it is possible that the value of the error $e(t)$ is very large at some particular $t$. For this reason, we can define *stability in the pointwise sense* if

$$|e(t)|^2 \leq C \sum_{n=-\infty}^{\infty} |e_n|^2 \quad \text{for all } t,$$

where $C$ is a constant.

(b) Use the Cauchy-Schwarz inequality (see Chapter 1) to show that the Nyquist-Shannon interpolation is stable in the pointwise sense. You’ll need the result from Problem 6.2.

6.20. **Sampling using the derivative**

Suppose that $f(t)$ is a bandlimited signal with $\mathcal{F}f(s) = 0$ for $|s| \geq 1$ (bandwidth 2). According to the sampling theorem, knowing the values $f(n)$ for all integers $n$ (sampling rate of 1) is not sufficient to interpolate the values $f(t)$ for all $t$. However, if *in addition* one knows the values of the derivative $f'(n)$ at the integers, then there is an interpolation formula with a sampling rate of 1. In this problem you will derive that result.

---

21 From A. Siripuram.
22 From D. Kammler.
Let $F(s) = \mathcal{F}f(s)$ and let $G(s) = \frac{1}{2\pi i} (\mathcal{F}f')'(s) = sF(s)$.

(a) For $0 \leq s \leq 1$ show that
\[
(\mathcal{I} \ast F)(s) = F(s) + F(s - 1),
\]
and then show that
\[
F(s) = (1 - s)(\mathcal{I} \ast F)(s) + (\mathcal{I} \ast G)(s).
\]

(b) For $-1 \leq s \leq 0$ show that
\[
(\mathcal{I} \ast F)(s) = F(s) + F(s + 1),
\]
and then show that
\[
F(s) = (1 + s)(\mathcal{I} \ast F)(s) - (\mathcal{I} \ast G)(s).
\]

(c) Using parts (a) and (b) show that for all $s, \ -\infty < s < \infty$,
\[
F(s) = \Lambda(s)(\mathcal{I} \ast F)(s) - \Lambda'(s)(\mathcal{I} \ast G)(s),
\]
where $\Lambda(s)$ is the triangle function
\[
\Lambda(s) = \begin{cases} 
1 - |s|, & |s| \leq 1, \\
0, & |s| \geq 1.
\end{cases}
\]

(d) From part (c) derive the interpolation formula
\[
f(t) = \sum_{n=-\infty}^{\infty} f(n)\text{sinc}^2(t - n) + \sum_{n=-\infty}^{\infty} f'(n)(t - n)\text{sinc}^2(t - n).
\]

6.21. In this problem\(^{23}\) we will show that for $0 \leq t, \tau \leq 1/2$,
\[
\min(t, \tau) = \sum_{n=-\infty}^{\infty} \sum_{n \text{ odd}} \frac{4\sin n\pi \tau \sin n\pi t}{n^2\pi^2}.
\]
(This relationship is useful, for example, in identifying the Fourier series expansion of a Wiener process.)

For a fixed $\tau$, consider the function $f$ given in the figure.

\[f(t)\]

---

\(^{23}\)From A. Siripuram.
(a) Since $f$ has a *time width* of 1, we should be able to sample $\mathcal{F}f(s)$ at a sampling rate of 1 and still recover $f$. Consider sampling $\mathcal{F}f(s)$ by multiplying it with $\Pi(s-1/2)$:

$$G(s) = \mathcal{F}f(s)\Pi(s-1/2).$$

Let $g(t)$ be the Fourier inverse of $G(s)$. Show that

$$g(t) = \sum_{n=-\infty}^{\infty} (-1)^n f(t-n).$$

(b) Note that $g$ is periodic. What is its period? Find the Fourier series for $g$ (you can leave your answer in terms of $\mathcal{F}f$).

(c) Argue why $\mathcal{F}f(s)$ satisfies

$$\pi is \mathcal{F}f(s) = \tau \text{sinc}^2 \tau s - \tau \cos \pi s.$$

From this, it follows that

$$\mathcal{F}f(s) = \frac{1}{\pi is} (\tau \text{sinc}^2 \tau s - \tau \cos \pi s).$$

(You do not have to prove this).

(d) Using the result of (c) in (b), deduce that

$$g(t) = \sum_{n=-\infty}^{\infty} \frac{4}{n^2 \pi^2} \sin n\pi \tau \sin n\pi t.$$

In particular, for $0 \leq t, \tau \leq 1/2$,

$$\min(t, \tau) = \sum_{n=-\infty}^{\infty} \frac{4 \sin n\pi \tau \sin n\pi t}{n^2 \pi^2}.$$

6.22. *Sampling with half a shah*

We define a new sampling function $\Pi(t)$, which is a train of evenly spaced $\delta$-function pairs, as shown in the figure. This sampling function is parameterized by two parameters: $q$, the spacing between the $\delta$-function pairs, and $b$, the spacing within the pairs of $\delta$-functions.

We will apply this sampling scheme to a function $f(t)$, which has a Fourier transform $\mathcal{F}f(s)$ and is bandlimited with bandwidth $p$ (i.e., $\mathcal{F}f(s) = 0$ for $|s| \geq p/2$).
(a) Write II(t) as the sum of two Shah functions (in terms of q and b).

(b) Define g(t) to be the sampled version of f(t) using our new sampling scheme, so that g(t) = II(t)f(t). Write the Fourier transform of g(t) in terms of Ff(s), q, and b.

(c) Consider the case where b = q/2. What is the maximum bandwidth p for which we can guarantee reconstruction of f(t) from its sampled version g(t)? What is a possible reconstruction scheme?

(d) Now consider the case when b = q/4. We further assume that f(t) is real and even. What is the maximum bandwidth p for which we can guarantee reconstruction of f(t) from its sampled version g(t)? For a signal of maximum bandwidth, what is a possible reconstruction scheme? Hint: Consider reconstruction based only on the real or the imaginary part of the Fourier transform of g(t).

6.23. Papoulis Generalized Sampling Theorem

Let f(t) be a function with bandwidth 2B and Fourier transform F(s), so that F(s) = 0 for |s| ≥ B. If we sample f(t) at the Nyquist rate 2B, we know we can reconstruct the function through sinc interpolation. This problem will explore a more general sampling and reconstruction scheme due to A. Papoulis.

(a) First input the signal f(t) into K linear systems with known impulse responses ak(t). The outputs of the systems are denoted gk, so

\[ g_k(t) = (a_k * f)(t) \quad \text{for } k = 0, \ldots, K - 1, \]

or in the frequency domain

\[ G_k(s) = A_k(s)F(s) \quad \text{for } k = 0, \ldots, K - 1, \]

where the Ak(s) are the transfer functions of the systems. Note that since f(t) has bandwidth 2B, so do all of the gk(t).

\[ 24 \text{ From Raj Bhatnagar.} \]

\[ 25 \text{ Athanasios Papoulis was a Greek-American mathematician who made many contributions to signal processing, of which this sampling theorem may be the best known. He wrote a book on Fourier transforms, too, like everybody has.} \]
(b) Next, sample each of the outputs $g_k(t)$ for $k = 0, \ldots, K - 1$ at the sub-Nyquist rate

$$B_K := 2B/K$$

to obtain the samples $g_k(nT_K)$ where $T_K = 1/B_K$.
(c) Then, even though we are sampling below the Nyquist rate, we can reconstruct our signal $f(t)$ from the interpolation formula

$$f(t) = \sum_{k=0}^{K-1} \sum_{n=-\infty}^{\infty} g_k(nT_K)p_k(t - nT_K),$$

where the functions $p_k$ are determined only from $A_k$ and $B_K$.

**Background.** Before we discuss how to prove this result (and specify $p_k$), let’s discuss why this result is true. In the interval $|s| < B_K$, we have the original spectrum $A_0(s)F(s)$ as well as $K - 1$ aliases, $A_k(s)F(s-kB_K)$. Think of the $F(s-kB_K)$ for $k = 0, \ldots, K - 1$ as $K$ unknowns. If we can find a linear system of $K$ equations relating $F(s-kB_K)$ to known quantities, then we can recover the original spectrum and find $f(t)$.

**Details.** Here’s how we find $p_k(t)$. First, for $-B \leq s < -B + B_K$, define the matrices

$$A = \begin{bmatrix}
A_0(s) & A_1(s) & \cdots & A_{K-1}(s) \\
A_0(s+B_K) & A_1(s+B_K) & \cdots & A_{K-1}(s+B_K) \\
\vdots & \vdots & \ddots & \vdots \\
A_0(s+(K-1)B_K) & A_1(s+(K-1)B_K) & \cdots & A_{K-1}(s+(K-1)B_K)
\end{bmatrix},$$

$$P = \begin{bmatrix}
P_0(s,t) \\
P_1(s,t) \\
\vdots \\
P_{K-1}(s,t)
\end{bmatrix},$$

and

$$E = \begin{bmatrix}
1 \\
e^{2\pi iB_Kt} \\
\vdots \\
e^{2\pi i(B-1)B_Kt}
\end{bmatrix}.$$

Then solve the linear system $P = A^{-1}E$ (assume that $\det A \neq 0$ so that the system has a solution). Notice that $A$ is a function of $s$ while $E$ is a function of $t$. $P_k$ is a function of both: in $s$, it is bandlimited to $(-B, -B + B_K)$, and in $t$, it has period $1/kB_K = T_K/k$. Finally, we define

$$p_k(t) := \frac{1}{B_K} \int_{-B}^{-B+B_K} P_k(s,t)e^{2\pi ist}ds.$$

**The Problem.**

(a) Denote the periodization of $P_k(s,t)e^{2\pi ist}$ in the $s$-domain by

$$\tilde{P}_k(s,t) := \sum_{m=-\infty}^{\infty} P_k(s + mB_K, t)e^{2\pi i(s+mB_K)t}.$$

Show that the $p_k(t - nT_K)$ are the Fourier series coefficients of $\tilde{P}_k(s,t)$.
(b) Explain why

$$\sum_{k=0}^{K-1} A_k(s)\tilde{P}_k(s,t) = e^{2\pi ist} \quad \text{for} \quad -B \leq s < B.$$
Hint: Consider multiplying the first equation in the linear system $A = PE$ by $e^{2\pi ist}$. What happens if you do the same with the second equation in the system?

(c) Using part (b), we have

$$f(t) = \int_{-B}^{B} F(s)e^{2\pi ist} \, ds = \int_{-B}^{B} F(s) \sum_{k=0}^{K-1} A_k(s) \tilde{P}_k(s,t) \, ds.$$ 

Use this to show the interpolation formula

$$f(t) = \sum_{k=0}^{K-1} \sum_{n=-\infty}^{\infty} g_k(nT_K)p_k(t - nT_K).$$

6.24. Chopper amplifiers\textsuperscript{26}

It is difficult to build an amplifier for DC signals (or, more generally, low frequency and bandlimited signals) that provides high gain and low noise. A device that overcomes these problems is the chopper amplifier, and its design is a nice example of passing between the time and frequency domains. A figure illustrating the entire process is below, following the statements of the problem.

The basic component, the chopper, is essentially a rapid on/off switch, acting on a signal in the time domain by multiplication with a periodized rect function,

$$c(t) = \sum_{k=-\infty}^{\infty} \Pi_T(t - 2kT).$$

Start with a signal $x(t)$, let $X(s) = \mathcal{F}x(s)$, and suppose

$$X(s) \equiv 0 \text{ for } |s| \geq W.$$ 

(a) Form the chopped signal

$$x_1(t) = c(t)x(t).$$ 

Show that the Fourier transform of $x_1(t)$ is

$$X_1(s) = \frac{1}{2} \sum_{k=-\infty}^{\infty} \text{sinc}(k/2)X\left(s - \frac{k}{2T}\right).$$

(b) What relationship must $W$ and $T$ satisfy for there to be no overlaps in the shifts of $X(s)$?

(c) Assuming the conditions of part (b), we next produce two spectral islands of $X_1(s)$, with a gain of $A$, by forming

$$X_2(s) = A \left( \Pi_{1/2T} \left( s + \frac{1}{2T} \right) + \Pi_{1/2T} \left( s - \frac{1}{2T} \right) \right) X_1(s)$$

$$= \frac{A}{2} \text{sinc}(1/2) \left( X\left(s - \frac{1}{2T}\right) + X\left(s + \frac{1}{2T}\right) \right).$$

(This is a band-pass filter.)

\textsuperscript{26}From Oppenheim and Willsky via Adam Wang.
Let $x_2(t)$ be the corresponding signal in the time domain, chop $x_2(t)$ to form

$$x_3(t) = c(t)x_2(t),$$

and, writing $X_3(s) = \mathcal{F}x_3(s)$, show that

$$X_3(s) = \frac{A}{4} \text{sinc}(1/2) \sum_{k=-\infty}^{\infty} \left( \text{sinc} \left( \frac{k - 1}{2} \right) + \text{sinc} \left( \frac{k + 1}{2} \right) \right) X(s - \frac{k}{2T}).$$

(d) Finally, multiply $X_3(s)$ by $\Pi_{1/2T}(s)$ (a low-pass filter) to obtain $Y(s)$, and let $y(t)$ be the corresponding signal in the time domain.

Find $y(t)$. How does $y(t)$ compare to the original signal $x(t)$? In particular, what is the overall gain, $B$, of the system?
Chapter 7

Discrete Fourier Transform

7.1. The Modern World

Many would say that the modern world began in 1965 when J. Cooley and J. Tukey published their account of an efficient method for numerical computation of Fourier series.¹ By that measure the modern world might actually have started rather earlier, for the idea at the heart of the algorithm is clearly present in an unpublished paper of Gauss that appeared posthumously in 1866.² Take your pick.

Whatever the past, the demands of the present and future are that we process continuous signals and their spectra by means of discrete methods; computers in all their forms can work with finite sums only. To turn the continuous into the discrete and finite requires, for our purposes, that a signal be both timelimited and bandlimited, something we know cannot be true, and that we take a finite number of samples, something we know cannot suffice. But it works. It works to the extent that a large fraction of the world’s economy depends upon it working, and that’s not a bad measure of success. Consider this a proof by economics.

Some would argue that one shouldn’t think in terms of “turning the continuous into the discrete” at all, but rather that measurements and data in the real world come to us in discrete forms, and that’s how we should understand the world and work with it. Period. Though things seem to have settled down, battles over “discrete” versus “continuous” can rise to a fever pitch. One such battle can be fought over the different approaches to the discrete Fourier transform, abbreviated DFT. As with everything else in this book, there are choices to make. My choice is to make the discrete look like the continuous as much as we can. We’re thus taking sides, at least initially, in favor of “from continuous to discrete” as a way of motivating the definition of the DFT. For one thing, we have built up a lot of

¹The title of the paper is “An algorithm for the machine calculation of complex Fourier series.” It appeared in Math. Comput. 19 (1965), 297–301. Incidentally, Tukey is also credited with coining the term “bit” as an abbreviation for “binary digit” — how about that for immortality!

²Much historical scholarship has gone into this. Google the paper “Gauss and the history of the Fast Fourier Transform” by M. T. Heideman, D. H. Johnson, and C. S. Burrus for a thorough discussion.
intuition and understanding of the Fourier transform, its properties, and its uses, and a reasonable goal is to leverage that intuition as we now work with the discrete Fourier transform. For a more discrete-centric approach, and more detail on many of the topics we’ll cover here, I highly recommend the book *The DFT: An Owner’s Manual* by W. Briggs and V. Henson.

### 7.2. From Continuous to Discrete

Start with a signal $f(t)$ and its Fourier transform $\mathcal{F}f(s)$, each a function of a continuous variable. We want to:

- Find a discrete version of $f(t)$ that’s a reasonable approximation to $f(t)$.
- Find a discrete version of $\mathcal{F}f(s)$ that’s a reasonable approximation to $\mathcal{F}f(s)$.
- Find a way to relate the discrete version of $\mathcal{F}f(s)$ to the discrete version of $f(t)$ that’s a reasonable approximation to the way $\mathcal{F}f(s)$ is related to $f(t)$.

Good things to try for, but it’s not quite straightforward. Here’s the setup.

We suppose that $f(t)$ is zero outside of $0 \leq t \leq L$. We also suppose that the Fourier transform $\mathcal{F}f(s)$ is zero, or effectively zero (beyond our ability to measure, negligible energy, whatever) outside of $0 < s < 2B$ ($B$ for bandwidth). We are taking the support of $\mathcal{F}f$ to be the interval from 0 to $2B$ instead of $-B$ to $B$ only because it will make the *initial indexing* of sample points easier; this will not be an issue in the end. We’ll also take $L$ and $B$ to both be integers so we don’t have to round up or down in any of the considerations that follow; you can think of that as our first concession to the discrete.

Thus we are regarding $f(t)$ as both timelimited and bandlimited, with the knowledge that this can only be approximately true. Remember, however, that we’re ultimately going to come up with a definition of a discrete Fourier transform that will make sense in and of itself regardless of shaky initial assumptions. After the definition is written down we could (and some do) erase all that came before it, perhaps casting only a brief, wistful glance back from the discrete to the continuous with a few comments on how the former approximates the latter. Many treatments of the discrete Fourier transform that start with the discrete and stay with the discrete do just that. We’re trying not to do that.

According to the sampling theorem (misapplied here, yes, but play along), we can reconstruct $f(t)$ perfectly from its samples if we sample at the rate of $2B$ samples per second. Since $f(t)$ is defined on an interval of length $L$ and the samples are $1/2B$ apart, that means that we want a total of

$$N = \frac{L}{1/2B} = 2BL$$

(note that $N$ is therefore even)

evenly spaced samples. Starting at 0 these are at the points

$$t_0 = 0, \quad t_1 = \frac{1}{2B}, \quad t_2 = \frac{2}{2B}, \ldots, \quad t_{N-1} = \frac{N-1}{2B}.$$

Draw yourself a picture. To know the values $f(t_k)$ is to know $f(t)$ reasonably well.
Thus we state:

- The discrete version of $f(t)$ is the list of sampled values $f(t_0), \ldots, f(t_{N-1})$.

Next, represent the discrete version of $f(t)$ (the list of sampled values) continuously with the aid of a finite impulse train (a finite III-function) at the sample points. Namely, using

$$\sum_{n=0}^{N-1} \delta(t-t_n),$$

let

$$f_{\text{sampled}}(t) = f(t) \sum_{n=0}^{N-1} \delta(t-t_n) = \sum_{n=0}^{N-1} f(t_n) \delta(t-t_n).$$

This is what we have considered previously as the sampled form of $f(t)$, hence the subscript “sampled.” The Fourier transform of $f_{\text{sampled}}$ is

$$\mathcal{F}f_{\text{sampled}}(s) = \sum_{n=0}^{N-1} f(t_n) \mathcal{F}\delta(t-t_n) = \sum_{n=0}^{N-1} f(t_n)e^{-2\pi ist_n}.$$

This is close to what we want — it’s the continuous Fourier transform of the sampled form of $f(t)$.

Now let’s change perspective and look at sampling in the frequency domain. The function $f(t)$ is limited to $0 \leq t \leq L$, and this determines a sampling rate for reconstructing $\mathcal{F}f(s)$ from its samples in the frequency domain. The sampling rate is $L$ and the spacing of the sample points is $1/L$. We sample $\mathcal{F}f(s)$ over the interval from 0 to $2B$ in the frequency domain at points spaced $1/L$ apart. The number of sample points is

$$\frac{2B}{1/L} = 2BL = N,$$

the same number of sample points as for $f(t)$. The sample points for $\mathcal{F}f(s)$ are of the form $m/L$, and there are $N$ of them. Starting at 0,

$$s_0 = 0, s_1 = \frac{1}{L}, \ldots, s_{N-1} = \frac{N-1}{L}.$$

The discrete version of $\mathcal{F}f(s)$ that we take is not $\mathcal{F}f(s)$ evaluated at these sample points $s_n$. Rather, it is $\mathcal{F}f_{\text{sampled}}(s)$ evaluated at the sample points. We base the discrete approximation of $\mathcal{F}f(s)$ on the sampled version of $f(t)$. To ease the notation write $F(s)$ for $\mathcal{F}f_{\text{sampled}}(s)$. Then:

- The discrete version of $\mathcal{F}f(s)$ is the list of values

$$F(s_0) = \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i s_0 t_n}, \quad F(s_1) = \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i s_1 t_n}, \ldots,$$

$$F(s_{N-1}) = \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i s_{N-1} t_n}.$$
By this definition, we now have a way of going from the discrete version of $f(t)$ to the discrete version of $F_f(s)$, namely the expressions

$$F(s_m) = \sum_{n=0}^{N-1} f(t_n)e^{-2\pi is_m t_n}.$$ 

These sums, one for each $m$ from $m = 0$ to $m = N - 1$, are supposed to be an approximation to the Fourier transform going from $f(t)$ to $F_f(s)$.

In what sense is this a discrete approximation to the Fourier transform? Since $f(t)$ is timelimited to $0 \leq t \leq L$, we have

$$F_f(s) = \int_0^L e^{-2\pi ist} f(t) dt.$$ 

Thus at the sample points $s_m$,

$$F_f(s_m) = \int_0^L e^{-2\pi is_m t} f(t) dt,$$

and to know the values $F_f(s_m)$ is to know $F_f(s)$ reasonably well. Now use the sample points $t_k$ for $f(t)$ to write a Riemann sum approximation for the integral. The spacing $\Delta t$ of the points is $1/2B$, so

$$F_f(s_m) = \int_0^L f(t)e^{-2\pi is_m t} dt \approx \sum_{n=0}^{N-1} f(t_n)e^{-2\pi is_m t_n} \Delta t$$

$$= \frac{1}{2B} \sum_{n=0}^{N-1} f(t_n)e^{-2\pi is_m t_n} = \frac{1}{2B} F(s_m).$$

This is the final point:

- Up to the factor $1/2B$, the values $F(s_m)$ provide an approximation to the values $F_f(s_m)$.

Writing a Riemann sum as an approximation to the integral defining $F_f(s_m)$ essentially discretizes the integral, and this is an alternate way of getting to the expression for $F(s_n)$, up to the factor $2B$. We short-circuited this route by working directly with $F_{f_{sampled}}(s)$.

You may find the “up to the factor $1/2B$” unfortunate in this part of the discussion, but it’s in the nature of the subject. In fact, back in Chapter 2 we encountered a similar kind of “up to the factor ...” phenomenon when we obtained the Fourier transform as a limit of the Fourier coefficients for a Fourier series.

### 7.3. The Discrete Fourier Transform

We are almost ready for a definition, with one final observation remaining to clear the way. The sample points are

$$t_n = \frac{n}{2B}, \quad s_m = \frac{m}{L},$$

and so

$$s_m t_n = \frac{mn}{2BL} = \frac{mn}{N}.$$
Hence
\[ F(s_m) = \sum_{n=0}^{N-1} f(t_n) e^{-2\pi is_m t_n} = \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i m / 2BL} = \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i m / N}. \]

The last form of the exponential, \( e^{-2\pi i m / N} \), highlights the indices of the inputs (index \( n \)) and outputs (index \( m \)) and the number of sample points (\( N \)). It hides the sample points themselves. We thus take the needed last step in moving from the continuous to the discrete.

We identify the point \( t_n \) with its index \( n \), and the list of \( N \) values \( f(t_0), f(t_1), \ldots, f(t_{N-1}) \) with the list of values of the discrete signal \( f[n], n = 0, 1, \ldots, N - 1, \) by defining
\[ f[n] = f(t_n). \]
Here we use the bracket notation (common practice) for functions of a discrete variable, as well as using the underline notation to distinguish \( f \), as a function, from \( f \). So, again, as a discrete function
\[ f[0] = f(t_0), f[1] = f(t_1), \ldots, f[N - 1] = f(t_{N-1}). \]

Likewise we identify the point \( s_m \) with its index \( m \), and we identify the list of \( N \) values \( F(s_0), F(s_1), \ldots, F(s_{N-1}) \) with values of the discrete signal \( F[m], m = 0, 1, \ldots, N - 1, \) by
\[ F[m] = F(s_m). \]
Written out,
\[ F[0] = F(s_0), F[1] = F(s_1), \ldots, F[N - 1] = F(s_{N-1}). \]
Purely in terms of the discrete variables \( n \) and \( m \),
\[ F[m] = \sum_{n=0}^{N-1} e^{-2\pi i mn / N} f[n]. \]
It remains to turn this around and make a definition solely in terms of discrete variables and discrete signals, pretending that we never heard of continuous signals and their sampled values.

Officially:
- Let \( f[n], n = 0, \ldots, N - 1, \) be a discrete signal of length \( N \). Its discrete Fourier transform (DFT) is the discrete signal \( Ff \) of length \( N \), defined by
\[ Ff[m] = \sum_{n=0}^{N-1} f[n] e^{-2\pi i mn / N}, \quad m = 0, 1, \ldots, N - 1. \]

We’ll write \( F_{\text{DFT}} f \), with the subscript, if we need to call attention to the \( N \).

It’s perfectly legitimate to let the inputs \( f[n] \) be complex numbers, though for applications they’ll typically be real. The computed values \( Ff[m] \) are generally complex, being sums of complex exponentials.

Giving software its due, MATLAB thinks the DFT is:
- An operation that accepts as input a list of \( N \) numbers and returns as output a list of \( N \) numbers, as specified above.
There are actually a number of things to say about the inputs and outputs of this operation, and I’ll try not to say them all at once. One can think of the inputs and outputs as $N$-vectors, but note that (for now) the indexing goes from $0$ to $N - 1$ rather than from $1$ to $N$ as usual. MATLAB gags on that, it’s not alone, and indexing is one of the things we’ll talk about.

There are some instances when the vector point of view is more natural and others where it’s more natural to think in terms of discrete signals and their values. I’ll use the underline notation for both. You have to keep both interpretations in your head. It’s not hard.

### 7.4. Notations and Conventions 1

I said that I wanted to set things up to look as much like the continuous case as possible, and the usual operations on functions are part of that. For example, for

$$x = (x[0], x[1], \ldots, x[N - 1]) \quad \text{and} \quad y = (y[0], y[1], \ldots, y[N - 1]),$$

we have

$$x \cdot y = (x[0]y[0], x[1]y[1], \ldots, x[N - 1]y[N - 1])$$

(componentwise product, not to be confused with the dot product of vectors — that comes later),

$$\frac{x}{y} = \left( \frac{x[0]}{y[0]}, \frac{x[1]}{y[1]}, \ldots, \frac{x[N - 1]}{y[N - 1]} \right)$$

(when the individual quotients make sense),

$$x^p = (x[0]^p, x[1]^p, \ldots, x[N - 1]^p)$$

(when the individual powers make sense),

and so on. We even allow a function of one variable (think sine or cosine, for example) to operate componentwise via

$$f((x[0], x[1], \ldots, x[N - 1])) = (f(x[0]), f(x[1]), \ldots, f(x[N - 1])).$$

You wouldn’t usually write expressions like this if you were thinking of $x$ and $y$ as vectors, but standard mathematical software packages (MATLAB, Mathematica) incorporate such componentwise operations on lists. Be careful, but keep an open mind.

We’ll also use the notation $[r : s]$ for the tuple of numbers $(r, r + 1, r + 2, \ldots, s)$. Two special signals. We’ll write

$$0 = (0, 0, \ldots, 0)$$

for the zero signal and

$$1 = (1, 1, \ldots, 1)$$

for the signal of all 1’s. A little notation can go a long way.

Finally, I’m aware of the fact that for various reasons it’s better to write an $N$-vector as a column (an $N \times 1$ matrix). This may have been drummed into you in a linear algebra class. Nevertheless, most of the time we’ll write vectors as horizontal $N$-tuples, as above. It’s typography. Sue me.
7.4. Notations and Conventions

7.4.1. Discrete complex exponentials. The definition of the discrete Fourier transform, like that of the continuous Fourier transform, involves a complex exponential. Much of the theory and practice of the DFT depends on the properties of the complex exponential, and it’s helpful to have some special notations to save writing.

We let
\[ \omega = e^{2\pi i/N}. \]

Occasionally we’ll decorate this with a subscript to
\[ \omega_N = e^{2\pi i/N} \]
when we want to emphasize the \( N \). From Euler’s formula,
\[ \text{Re} \, \omega_N = \cos(2\pi/N), \quad \text{Im} \, \omega_N = \sin(2\pi/N). \]

\( \omega_N \) is an \( N \)th root of unity, meaning
\[ \omega_N^N = e^{2\pi i N/N} = e^{2\pi i} = 1. \]
There are \( N \) distinct numbers in the list of the powers \( \omega_n^N \) as \( n \) goes from 0 to \( N - 1 \):
\[ 1 = \omega_N^0, \omega_N^1, \omega_N^2, \ldots, \omega_N^{N-1}, \]
and each of the \( \omega_N^n \) is itself an \( N \)th root of unity; i.e.,
\[ \omega_N^{nN} = 1. \]
Then, in general for any integers \( n \) and \( k \),
\[ \omega_N^{k+nN} = \omega_N^k. \]
All obvious, but these properties come up often enough that it’s worth pointing them out.

Also note that
\[ \omega_N^{N/2} = e^{2\pi i N/2N} = e^{i\pi} = -1 \quad \text{and hence} \quad \omega_N^{kN/2} = (-1)^k. \]
These come up, too.

On the notation, some people write \( \omega_N = e^{-2\pi i/N} \) (minus instead of plus in the exponential) and others write \( W = e^{2\pi in/N} \). I’ve seen all sorts of things, so be aware of different conventions as you peruse the literature.

For the discrete Fourier transform it’s helpful to bundle the powers of \( \omega \) together as a discrete signal of length \( N \). We define the discrete signal \( \omega \) (or \( \omega_N \) if the \( N \) is wanted for emphasis) by
\[ \omega[m] = \omega^m, \quad m = 0, \ldots, N - 1. \]
Writing the values as a vector,
\[ \omega = (1, \omega, \omega^2, \ldots, \omega^{N-1}). \]
This is the discrete complex exponential. Recasting what we said above, the real and imaginary parts of $\omega[m]$ are

$$\text{Re}\,\omega[m] = \cos\left(\frac{2\pi m}{N}\right), \quad \text{Im}\,\omega[m] = \sin\left(\frac{2\pi m}{N}\right), \quad m = 0, 1, \ldots, N - 1.$$ 

The integer powers of $\omega$ are

$$\omega^k[m] = \omega^{km} \quad \text{or} \quad \omega^k = (1, \omega^k, \omega^{2k}, \ldots, \omega^{(N-1)k}).$$

There’s a symmetry here:

$$\omega^k[m] = \omega^{km} = \omega^m[k].$$

The power $k$ can be positive or negative, and of course

$$\omega^{-k}[m] = \omega^{-km}, \quad \text{or} \quad \omega^{-k} = (1, \omega^{-k}, \omega^{-2k}, \ldots, \omega^{-(N-1)k}).$$

In terms of complex conjugates,

$$\omega^{-k} = \overline{\omega^k}.$$

Note also that

$$\omega^0 = 1.$$

In all of this you can see why it’s important to use notations that distinguish a discrete signal or a vector from a scalar. Don’t let your guard down.

Finally,

$$\omega^k[m + nN] = \omega^{km + knN} = \omega^{km} = \omega^k[m].$$

This is the periodicity property of the discrete complex exponential:

- $\omega^k$ is periodic of period $N$ for any integer $k$.

As simple as it is, this is a crucial fact in working with the DFT.

The DFT rewritten. Introducing the discrete complex exponential allows us to write the formula defining the discrete Fourier transform with a notation that, I think, really looks like a discrete version of the continuous transform. The DFT of a discrete signal $\mathbf{f}$ is

$$\hat{\mathbf{f}} = \sum_{k=0}^{N-1} f[k] \omega^{-k}.$$ 

At an index $m$,

$$\hat{\mathbf{f}}[m] = \sum_{k=0}^{N-1} f[k] \omega^{-km} = \sum_{k=0}^{N-1} f[k] \omega^{-km} = \sum_{k=0}^{N-1} f[k] e^{-2\pi ikm/N}.$$ 

We still think of the indices $m$ as frequencies, but the definition has $m = 0, 1, \ldots, N - 1$. Where are the negative frequencies, you may well ask? Soon, very soon, you will know.

The frequencies for which $\hat{\mathbf{f}}[m] \neq 0$ constitute the spectrum of $\mathbf{f}$, though, as with the continuous Fourier transform, we sometimes also refer to the values $\hat{\mathbf{f}}[m]$ as the spectrum.

---

3It’s also referred to as the vector complex exponential (keeping an open mind).
We note one special value:

\[ F[f_0] = \sum_{k=0}^{N-1} f[k] \omega^{-k} = \sum_{k=0}^{N-1} f[k], \]

the sum of the values of the input \( f \). Some people define the DFT with a \( 1/N \) in front so that the zeroth component of the output is the average of the components of the input, just as the zeroth Fourier coefficient of a periodic function is the average of the function over one period. We’re not doing this, but the choice of where and whether to put a factor of \( 1/N \) in a formula haunts the subject. Wait and see.

We also note that the DFT is linear. To state this formally as a property,

\[ F(f_1 + f_2) = F f_1 + F f_2 \quad \text{and} \quad F(\alpha f) = \alpha F f. \]

Showing this is easy:

\[
F(f_1 + f_2) = \sum_{k=0}^{N-1} (f_1 + f_2)[k] \omega^{-k} = \sum_{k=0}^{N-1} (f_1[k] + f_2[k]) \omega^{-k} \\
= \sum_{k=0}^{N-1} f_1[k] \omega^{-k} + \sum_{k=0}^{N-1} f_2[k] \omega^{-k} = F f_1 + F f_2.
\]

The DFT in matrix form. Switching the point of view to vectors and matrices, as a linear transformation taking vectors in \( \mathbb{C}^N \) to vectors in \( \mathbb{C}^N \), computing the DFT is exactly multiplying by a matrix. With \( F = F f \),

\[
\begin{pmatrix}
F[0] \\
F[1] \\
F[2] \\
\vdots \\
F[N-1]
\end{pmatrix} = \begin{pmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & \omega^{-1} & \omega^{-1} & \ldots & \omega^{-1(N-1)} \\
1 & \omega^{-2} & \omega^{-2} & \ldots & \omega^{-2(N-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{-(N-1)} & \omega^{-(N-1)} & \ldots & \omega^{-(N-1)^2}
\end{pmatrix} \begin{pmatrix}
f[0] \\
f[1] \\
f[2] \\
\vdots \\
f[N-1]
\end{pmatrix}.
\]

The discrete Fourier transform is the big old \( N \times N \) matrix

\[
F = \begin{pmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & \omega^{-1} & \omega^{-2} & \ldots & \omega^{-(N-1)} \\
1 & \omega^{-2} & \omega^{-4} & \ldots & \omega^{-2(N-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{-(N-1)} & \omega^{-(N-1)} & \ldots & \omega^{-(N-1)^2}
\end{pmatrix}.
\]

Of course, if someone gave you the matrix for the DFT as a starting point, you would write down the corresponding transform. You would write

\[ F f[m] = \sum_{k=0}^{N-1} f[k] \omega^{-km} = \sum_{k=0}^{N-1} f[k] \omega^{-k[m]}. \]

You would do that.
In the usual compact matrix notation,
\[(\mathcal{F})_{mn} = \omega^{-mn} = e^{-2\pi imn/N}.\]
The columns of \(\mathcal{F}\) are just the negative powers of the discrete complex exponentials (as are the rows). Again, we take the indices for \(\mathcal{F}\) to go from 0 to \(N-1\) instead of from 1 to \(N\), and the same for the rows and columns of \(\mathcal{F}\). Again I issue the warning, be careful how your favorite software package indexes in computing the DFT.

Right away we can see that the matrix for the DFT is *symmetric* (equal to its transpose):
\[(\mathcal{F})_{mn} = (\mathcal{F})_{nm}.\]
In fact, it has the additional property of being (almost) unitary, but that comes a little later.

### 7.5. Two Grids, Reciprocally Related

Before surrendering completely to the discrete world, we look back fondly one more time to the continuous. Invoke our understanding that the DFT finds the sampled Fourier transform of a sampled signal. We have a grid of points in the time domain and a grid of points in the frequency domain where the discrete version of the signal and the discrete version of its Fourier transform are known. From the discrete point of view, the values of the signal at the points in the time domain are all we know about the signal, and the values we compute according to the DFT formula are all we know about its transform.

A quick recap: In the time domain the signal is limited to an interval of length \(L\). In the frequency domain the transform is limited to an interval of length \(2B\). The grid points in the time domain are spaced \(1/2B\) apart. The grid points in the frequency domain are spaced \(1/L\) apart, so note (again) that the spacing in one domain is determined by properties of the signal in the other domain. The two grid spacings are related to the third quantity in the setup, the number of sample points, \(N\). The equation is
\[N = 2BL.\]
Now, any two of the quantities \(B\), \(L\), or \(N\) determine the third via this relationship. The equation is often written another way, in terms of the grid spacings. If \(\Delta t = 1/2B\) is the grid spacing in the time domain and \(\Delta \nu = 1/L\) is the grid spacing in the frequency domain, then
\[N = \frac{1}{\Delta t \Delta \nu} \quad \text{or} \quad \frac{1}{N} = \Delta t \Delta \nu.\]
These two equivalent equations are referred to as the *reciprocity relations*. For a fixed number of sample points \(N\), making \(\Delta t\) small means making \(\Delta \nu\) large, and vice versa (relatively — depends on the size of \(N\)). Put that in your head.

Here's why all of this is important. For a given problem you want to solve, for a given signal you want to analyze by taking the Fourier transform, you typically either *know* or *choose* two of the following items:

- How long the signal lasts, i.e., how long you’re willing to sit there taking measurements — that’s \(L\).
• How many measurements you make — that’s \( N \).
• How often you make a measurement — that’s \( \Delta t \).
• How fine a resolution you want in the spectrum — that’s \( \Delta \nu \).

Once two of these are determined, everything else is set. See the book by Briggs and Henson, mentioned earlier, for further discussion and examples.

7.6. Getting to Know Your Discrete Fourier Transform

We want to develop some general properties of the DFT as well as to compute the DFT of some special signals, much as we did when we first introduced the continuous Fourier transform. For the DFT there’s less of an emphasis on finding explicit transforms, though a few examples are important, the discrete \( \delta \) for example.

Many properties of the DFT correspond pretty directly to properties of the continuous Fourier transform, but there are differences. You should try to make the most of these correspondences, if only to decide for yourself when they are close and when they are not. Use what you know from your work in the continuous setting. We’ll find formulas that are interesting and that are very important in applications.

Derivations using the DFT are often easier, technically, than those for the Fourier transform in that there are no worries about integrals converging, etc., but discrete derivations can have their own complications. The skills you need are in manipulating sums, particularly sums of complex exponentials, and in manipulating matrices. In both instances, if you’re troubled, it’s often a good idea to first calculate a few examples, say for DFTs of size three or four, or to write out the first several terms in a sum to see what the patterns are.

One thing to be mindful of in deriving formulas, in particular when working with sums, is how the index of summation (analogous to the variable of integration) enters and not to get it confused or in conflict with other indices that are in use; see Appendix C, on geometric sums. Derivations might also involve changing the index of summation, analogous to changing the variable of integration. This procedure (technique) seems easier in the case of integrals than in sums, maybe because of all the practice you’ve had making substitutions to find integrals. At any rate, you’ll see the sorts of things that come up in the derivations in the chapter, and in the problems.

Two important facts. Before embarking, I am pleased to announce at once that all interesting properties of the DFT derive essentially from two facts. They are:

• Orthogonality of the discrete complex exponentials.
• Periodicity of the discrete complex exponentials, and hence periodicity of the inputs and outputs.

After treating ourselves to the DFT of one special signal, we’ll discuss each of these in turn.
7.6.1. The discrete ķ. Let’s begin with an explicit calculation rather than a general property, the DFT of the discrete ķ. And good news! No need to go through the theory of distributions to define a ķ in the discrete case, no pairings, no “where did we start and where did we finish.” We can define it directly and easily by setting

\[ \delta_0 = (1, 0, \ldots, 0). \]

In words, there’s a 1 in the zeroth slot and 0’s in the remaining \( N - 1 \) slots. We didn’t specify \( N \), and so, strictly speaking, there’s a \( \delta_0 \) for each \( N \), but since \( \delta_0 \) will always arise in a context where the \( N \) is otherwise specified, we’ll set aside that detail and we won’t modify the notation.

To find the DFT of \( \delta_0 \), we have, for any \( m = 0, 1, \ldots, N - 1 \),

\[
\mathcal{F} \delta[m] = \sum_{k=0}^{N-1} \delta_0[k] \omega^{-k} \left[ m \right] \\
= 1 \omega^0[m] \quad \text{(only the term } k = 0 \text{ survives)} \\
= 1.
\]

The same calculation done all at once is

\[
\mathcal{F} \delta_0 = \sum_{k=0}^{N-1} \delta_0[k] \omega^{-k} = \omega^0 = 1.
\]

Great — it looks just like \( \mathcal{F} \delta = 1 \) in the continuous case, and no tempered distributions in sight!

The shifted discrete ķ is just what you think it is,

\[ \delta_n = (0, \ldots, 0, 1, 0, \ldots, 0) \]

with a 1 in the \( n \)th slot and zeros elsewhere. This is another name for the Kronecker δ, written for indices \( m \) and \( n \) as

\[ \delta_n[m] = \delta_{mn} = \begin{cases} 1, & m = n, \\ 0, & m \neq n. \end{cases} \]

For the DFT of \( \delta_n \), we have

\[
\mathcal{F} \delta_n = \sum_{k=0}^{N-1} \delta_n[k] \omega^{-k} = \omega^{-n}.
\]

These are our first explicit transforms, and if we believe that the discrete case can be made to look like the continuous case, the results are encouraging. We state them again.

\[ ^4 \text{If you’re waiting for a discrete version of the rectangle function Π and its Fourier transform, which was our first example in the continuous setting, that comes later.} \]
The DFTs of the discrete δ₀ and shifted δₙ are
\[ Fδ₀ = 1 \quad \text{and} \quad Fδₙ = ω⁻ⁿ. \]

We'll establish other properties of discrete δₙ's (convolution, sampling) later.

**The vector view.** There's a vector/matrix point of view that you should keep in mind. Regarded as vectors, the δ's are the natural basis vectors δ₀, δ₁, ..., δ₉₋₁ of \( \mathbb{C}^{N} \). True, we're indexing from 0 to \( N - 1 \) instead of from 1 to \( N \), but get over it. Thinking of the DFT as a matrix, \( Fδₙ \) is the \( n \)th column of \( F \), and this is \( ω⁻ⁿ \). Good.

Or think in terms of vectors and \( F \) as a linear transformation from \( \mathbb{C}^{N} \) to \( \mathbb{C}^{N} \). For an arbitrary discrete signal (vector) \( f \) we can write
\[ f = \sum_{k=0}^{N-1} f[k]δₖ, \]

and if we had defined the DFT by what it does to a basis, we’d say it’s the linear transformation for which \( Fδₖ = ω⁻ᵏ \). We would find from this
\[ Ff = \sum_{k=0}^{N-1} f[k]Fδₖ = \sum_{k=0}^{N-1} f[k]ω⁻ᵏ. \]

Good.

**7.6.2. Orthogonality of the discrete complex exponentials.** To state this crucial property of the discrete complex exponentials we again regard
\[ ω = (1, ω, ω², ..., ω^{N-1}) \]
and its \( k \)th power
\[ ω^k = (1, ω^k, ω^{2k}, ..., ω^{(N-1)k}) \]
as vectors. Then the inner products satisfy:
- For \( k \) and \( ℓ \) in \([0 : N - 1]\),
\[ \omega^k · \omega^ℓ = \begin{cases} 0, & k \neq ℓ, \\ N, & k = ℓ. \end{cases} \]

For the norms, for each \( k \),
\[ ||ω^k|| = \sqrt{N}. \]

Thus the powers of the discrete complex exponentials are orthogonal and *almost* orthonormal. That’s the result in the title of this section. Later we’ll give another version of orthogonality that incorporates periodicity.

We could make the powers orthonormal by considering instead
\[ \frac{1}{\sqrt{N}}(1, ω^k, ω^{2k}, ..., ω^{(N-1)k}). \]

We won’t do this, and neither does anyone else, pretty much. But we’ll pay for it in the way a factor of \( N \) comes up in various formulas.
To remind you, in the continuous case the analogous result is that the family of functions \( (1/\sqrt{T})e^{2\pi int/T} \) (periodic of period \( T \)) is orthonormal with respect to the inner product on \( L^2([0, T]) \):

\[
\int_0^T \frac{1}{\sqrt{T}} e^{2\pi int/T} \frac{1}{\sqrt{T}} e^{-2\pi int/T} \, dt = \frac{1}{T} \int_0^T e^{2\pi i(m-n)t/T} \, dt = \begin{cases} 1, & m = n, \\ 0, & m \neq n. \end{cases}
\]

Orthogonality in the continuous case may seem easier to establish than in the discrete case because, sometimes, integration may seem easier than summation. Depends on what you're used to. There are two things we'll need for the derivation. The first is that \( \omega^k = 1 \) when \( k \) is any integer. The second is a geometric sum:

\[
1 + z + z^2 + \cdots + z^{N-1} = \begin{cases} \frac{1-z^N}{1-z}, & z \neq 1, \\ N, & z = 1. \end{cases}
\]

See Appendix C (again) if you need some review.

We’ll use this formula for \( z \) a power of \( \omega \). Observe that if \( k \neq \ell \), then, as both \( k \) and \( \ell \) are in \( [0 : N - 1] \),

\[
\omega^{k-\ell} = e^{2\pi i(k-\ell)/N} \neq 1 \quad \text{while} \quad \omega^{(k-\ell)N} = e^{2\pi i(k-\ell)} = 1.
\]

With this, let’s compute the inner product \( \omega^k \cdot \omega^\ell \). In full detail,

\[
\begin{aligned}
\omega^k \cdot \omega^\ell &= \sum_{n=0}^{N-1} \omega^k[n] \omega^\ell[n] = \sum_{n=0}^{N-1} \omega^k[n] \omega^{-(\ell)[n]} \\
&= \sum_{n=0}^{N-1} \omega^{k-\ell}[n] = \sum_{n=0}^{N-1} \omega^{(k-\ell)n} = \sum_{n=0}^{N-1} (\omega^{k-\ell})^n \\
&= \frac{1 - \omega^{(k-\ell)N}}{1 - \omega^{k-\ell}} \\
&= \begin{cases} 0, & k \neq \ell, \\ N, & k = \ell. \end{cases}
\end{aligned}
\]

Done. By taking complex conjugates we also deduce that

\[
\omega^{-k} \cdot \omega^{-\ell} = \begin{cases} 0, & k \neq \ell, \\ N, & k = \ell. \end{cases}
\]

\( \mathcal{F} \) is invertible. From this result we conclude that the \( N \) distinct vectors \( 1, \omega, \omega^2, \ldots, \omega^{(N-1)} \) are a basis of \( \mathbb{C}^N \); they are orthogonal, so linearly independent, and there are \( N \) of them. Likewise \( 1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(N-1)} \) are also a basis of \( \mathbb{C}^N \). From the earlier result that \( \mathcal{F} \delta_k = \omega^{-k} \) we can then conclude that \( \mathcal{F} \) is invertible because it’s a linear transformation that takes a basis to a basis. This doesn’t tell us what the inverse is, however. We have to work a little harder for that.

The DFT of the discrete complex exponential. With the orthogonality of the discrete complex exponentials established, a number of other important results are within easy reach. For example, we can now find \( \mathcal{F} \omega^k \).
By definition,

$$F\omega^k = \sum_{n=0}^{N-1} \omega^k[n] \omega^{-n},$$

and its $\ell$th component is then

$$F\omega^k[\ell] = \sum_{n=0}^{N-1} \omega^k[n] \omega^{-n}[\ell]$$

$$= \sum_{n=0}^{N-1} \omega^{kn} \omega^{-n\ell} = \omega^k \cdot \omega^\ell = \begin{cases} 0, & \ell \neq k, \\ N, & \ell = k. \end{cases}$$

That is, $F\omega^k = (0, \ldots, N, \ldots, 0)$ with an $N$ in the $k$th slot. We recognize this, and we are pleased.

- The discrete Fourier transform of $\omega^k$ is
  $$F\omega^k = N\delta_k.$$

- In particular, taking $k = 0$ we have
  $$F1 = N\delta_0.$$

Perhaps we are almost pleased. There’s a factor of $N$ that comes in that we don’t see, in any way, in the continuous case. Here it traces back, ultimately, to $||\omega||^2 = N$.

The appearance of a factor $N$ or $1/N$ in various formulas is always wired somehow to $||\omega||^2 = N$. It’s one thing that makes the discrete case appear different from the continuous case, and it’s a pain in the neck to keep straight. This is what you heard me warning and complaining about.

7.6.3. Inverting the DFT. By now it should be second nature to you to expect that any useful transform ought to be invertible. The DFT is no exception; we commented on that just above. The DFT does have an inverse and the key to finding the formula is the orthogonality of the discrete complex exponentials. With a nod to the matrix form of the DFT it’s actually pretty easy to realize what $F^{-1}$ is, if somewhat unmotivated. (I’ll give an alternate way to identify $F^{-1}$ a little later.)

All the inner products $\omega^k \cdot \omega^\ell$ can be bundled together in the product of two matrices. Consider the conjugate transpose (the adjoint), $F^*$, of the DFT matrix. Since the DFT matrix is already symmetric, this just takes the conjugate of all the entries. The $k$th row of $F^*$ is

$$\begin{pmatrix} 1 & \omega^k & \omega^{2k} & \ldots & \omega^{(N-1)k} \end{pmatrix}.$$
The $\ell$th column of $F$ is
\[
\begin{pmatrix}
1 \\
\omega^{-1}\ell \\
\omega^{-2}\ell \\
\vdots \\
\omega^{-(N-1)\ell}
\end{pmatrix}.
\]

The $k\ell$-entry of $F^*F$ is the product of the $k$th row of $F^*$ and the $\ell$th column of $F$, which is, as highlighted in bold,
\[
(F^*F)_{k\ell} = \sum_{n=0}^{N-1} \omega^{kn}\omega^{-n\ell}.
\]

This sum is the inner product,
\[
\sum_{n=0}^{N-1} \omega^{kn}\omega^{-n\ell} = \omega^k : \omega^\ell = \begin{cases} 
0, & \ell \neq k, \\
N, & \ell = k.
\end{cases}
\]

All the off-diagonal elements of $F^*F$ are zero, and the elements along the main diagonal are all equal to $N$. Combining the two statements,
\[
F^*F = NI,
\]
where $I$ is the $N \times N$ identity matrix. A similar calculation would also give $F F^* = NI$, multiplying the matrices in the other order.

And now the inverse of the DFT appears before us. Rewrite $F^*F = NI$ as
\[
\left(\frac{1}{N}F^*\right)F = I
\]
and, behold, the inverse of the DFT is
\[
F^{-1} = \frac{1}{N}F^*.
\]
You would also write this as
\[
F^{-1}f[m] = \frac{1}{N} \sum_{k=0}^{N-1} f[k]\omega^{km} = \frac{1}{N} \sum_{k=0}^{N-1} f[k]\omega^k[m],
\]
for $m = 0, 1, \ldots, N - 1$. You would do that.
Viewed together, the two transforms

\[ Ff = \sum_{k=0}^{N-1} f[k] \omega^{-k} \quad \text{and} \quad F^{-1}f = \frac{1}{N} \sum_{k=0}^{N-1} f[k] \omega^{k} \]

are imitating the relationship between the continuous-time Fourier transform and its inverse in that there’s just a change in the sign of the complex exponential to go from one to the other — except, of course, for the ever irritating factor of \(1/N\).

**A brief digression on matrices.** This material is probably familiar to you from a course in linear algebra, but I thought it would be helpful to include. We’ve already used some of the terminology, and we’ll see these things again when we study linear systems themselves in Chapter 8.

For the definitions it’s first necessary to remember that the transpose of a matrix \(A\), denoted by \(A^T\), is obtained by interchanging the rows and columns of \(A\). If \(A\) is an \(M \times N\) matrix, then \(A^T\) is an \(N \times M\) matrix, and in the case of a square matrix (\(M = N\)) taking the transpose amounts to reflecting the entries across the main diagonal. As a linear transformation, if \(A: \mathbb{R}^N \to \mathbb{R}^M\), then \(A^T: \mathbb{R}^M \to \mathbb{R}^N\).

If, for shorthand, we write \(A\) generically in terms of its entries, as in \(A = (a_{ij})\), then we write \(A^T = (a_{ji})\); note that the diagonal entries \(a_{ii}\), where \(i = j\), are unaffected by taking the transpose.

Square matrices can have a special property with respect to taking the transpose — they get to be symmetric: a square matrix \(A\) is symmetric if

\[ A^T = A. \]

In words, interchanging the rows and columns gives back the same matrix — it’s symmetric across the main diagonal. (The diagonal entries need not be equal to each other!) The DFT, though a complex matrix, is symmetric.

A different notion also involving a matrix and its transpose is orthogonality. A square matrix is orthogonal if

\[ A^T A = I, \]

where \(I\) is the identity matrix. A matrix is orthogonal if and only if its columns (or rows) are orthonormal with respect to the inner product and hence form an orthonormal basis of \(\mathbb{R}^N\). Now be careful, “symmetric” and “orthogonal” are independent notions for matrices. A matrix can be one and not the other.

For matrices with complex entries (operating on real or complex vectors) the appropriate notion corresponding to simple symmetry in the real case is Hermitian symmetry. For this we form the transpose and take the complex conjugate of the entries. If \(A\) is a complex matrix, then we use \(A^*\) to denote the conjugate transpose, known as the adjoint of \(A\). A square matrix \(A\) is Hermitian if

\[ A^* = A. \]

The DFT is not Hermitian.

Finally, a square matrix is unitary if

\[ A^* A = I. \]
Once again, “Hermitian” and “unitary” are independent notions for complex matrices. And once again, a matrix is unitary if and only if its columns (or rows) are orthonormal with respect to the complex inner product and hence form an orthonormal basis of $\mathbb{C}^N$.

The fundamental result above on the DFT is

$$\mathcal{F}^* \mathcal{F} = \mathcal{F} \mathcal{F}^* = NI,$$

and one could fairly say that $\mathcal{F}$ is “almost” unitary. It misses by the factor $N$, which is there because the discrete complex exponentials are orthogonal but not orthonormal. Oh, the agony.

7.6.4. Periodicity. Here comes the second important fact about the DFT, the periodicity of the inputs and outputs. This is a difference between the discrete and continuous cases rather than a similarity. The definition of the DFT, specifically the periodicity of the discrete complex exponentials suggests, even compels, some additional structure to the outputs and inputs.

As with the input values, the output values $F[m]$ are defined initially only for $m = 0$ to $m = N - 1$, but their definition as

$$F[m] = \sum_{k=0}^{N-1} f[k] \omega^{-km}$$

implies a periodicity property. Since

$$\omega^{-k(m+N)} = \omega^{-km},$$

we have

$$\sum_{k=0}^{N-1} f[k] \omega^{-k(m+N)} = \sum_{k=0}^{N-1} f[k] \omega^{-km} = F[m].$$

If we consider the left-hand side as the DFT formula producing an output $F$, then that output would be $F[m+N]$. The equations together then say that $F[m+N] = F[m]$, a periodicity property. More generally, and by the same kind of calculation, we would have

$$F[m + nN] = F[m]$$

for any integer $n$. Thus, instead of just working with $F$ as defined for $m = 0, 1, \ldots, N - 1$ it’s natural to extend it to be a periodic signal of period $N$, defined on all the integers.


$$F[p] = F[q]$$

if $p - q$ is a multiple of $N$, positive or negative.

Or put another way

$$F[p] = F[q]$$

if $p \equiv q \mod N$. 
(I’m assuming that you’re familiar with the “≡” notation as used in modular arithmetic.) We then have the formula

\[ F[m] = \sum_{k=0}^{N-1} f[k] \omega^{-km} \]

for all integers \( m \).

Because of these observations and unless instructed otherwise:

- We will always assume that the output of the DFT is a periodic signal of period \( N \).

Furthermore, the form of the inverse DFT means that an input \( f \) to the DFT also extends naturally to be a periodic signal of period \( N \). It’s the same argument, for if we start with \( F = F f \), input \( f \) and output \( F \), then

\[ f[m] = F^{-1}[m] = \frac{1}{N} \sum_{k=0}^{N-1} F[k] \omega^{k}[m], \]

and the right-hand side defines a periodic signal of period \( N \). Thus:

- We will always assume that the input to the DFT is a periodic signal of period \( N \).

So, briefly, when the DFT is involved, we assume that all our discrete signals are periodic.

As for how this differs from the continuous case, we certainly can consider periodicity — that’s what the subject of Fourier series is all about, after all — but when working with the Fourier transform, we don’t have to consider periodicity. In the discrete case we really do.

**The discrete complex exponential and discrete δ as periodic signals.** To take an important example, if, according to the periodicity dictum, we consider the discrete complex exponential as a periodic discrete signal, then we can define it simply by

\[ \omega[m] = \omega^m = e^{2\pi im/N}, \quad m \text{ an integer}. \]

The orthogonality property looks a little more general:

\[ \omega^k \cdot \omega^\ell = \begin{cases} 0, & k \not\equiv \ell \mod N, \\ N, & k \equiv \ell \mod N. \end{cases} \]

The discrete \( \delta \), as a periodic signal, also has a more general looking definition:

\[ \delta_n[m] = \begin{cases} 0, & m \not\equiv n \mod N, \\ 1, & m \equiv n \mod N. \end{cases} \]

**On the definition of the DFT.** If we were developing the DFT from a purely mathematical point of view, we would probably incorporate periodicity as part of the initial definition, and this is sometimes done. For the setup, we let

\[ \mathbb{Z}_N = \{0, 1, \ldots, N - 1\}, \]

with addition understood to be modulo \( N \). The symbol “\( \mathbb{Z} \)” is for the German word *Zahlen*, which means “numbers.” Tell your friends. So \( \mathbb{Z} = \{0, \pm1, \pm2, \ldots \} \).
the tension over this and other similar sorts of things. It goes:

- The DFT can be defined over any set of $N$ consecutive indices.

What this means most often in practice is that we can write

$$\mathcal{F} f = \sum_{k=p}^{p+N-1} f[k] \omega^{-k},$$

for any integer $p$. We’ll explain this thoroughly in just a bit. It’s a little tedious, but not difficult, and it’s important to understand. Thinking of an input (or output) $f$ as a periodic discrete signal (as we must, but something your software package can’t really do), then you don’t have to worry about how it’s indexed. It goes on forever, and any block of $N$ consecutive values, $f[p], f[p+1], \ldots, f[p+N-1]$, should be as good as any other because the values of $f$ repeat. You still have to establish the quoted remark, however, to be assured that finding the DFT gives the same result on any such block. This is essentially a discrete form of the statement for continuous-time periodic functions that the Fourier coefficients can be calculated by integrating over any period.

**Negative frequencies.** In case you hadn’t noticed, negative frequencies have appeared on the scene! Indeed, once the signal has been extended periodically to be defined on all the integers there’s no issue with considering negative frequencies. However, unless and until periodicity comes to the rescue, be aware that many standard treatments of the DFT cling to indexing from 0 to $N-1$ and talk about which frequencies in that range should be thought of as the negative frequencies “by convention.” This is also an issue with software packages.

Suppose that $N$ is even. This makes things a little easier and is often assumed.\(^6\) Suppose also that we consider real inputs $f = (f[0], f[1], \ldots, f[N-1])$. Something special happens at the midpoint, $N/2$, of the spectrum. We find

$$\mathcal{F} f[N/2] = \sum_{k=0}^{N-1} f[k] \omega^{-k}[N/2] = \sum_{k=0}^{N-1} f[k] \omega^{-kN/2}$$

$$= \sum_{k=0}^{N-1} f[k] e^{-\pi i k} = \sum_{k=0}^{N-1} (-1)^{-k} f[k] \quad (\text{using } \omega^{N/2} = -1).$$

The value of the transform $\mathcal{F} f[N/2]$ is an alternating sum of the components of the input signal $f$. In particular, $\mathcal{F} f[N/2]$ is real.

Furthermore, the spectrum splits at $N/2$. I explain. For a start, look at $\mathcal{F} f[(N/2) + 1]$ and $\mathcal{F} f[(N/2) - 1]$:

$$\mathcal{F} f\left[\frac{N}{2} + 1\right] = \sum_{k=0}^{N-1} f[k] \omega^{-k} \left[\frac{N}{2} + 1\right] = \sum_{k=0}^{N-1} f[k] \omega^{-k} \omega^{-Nk/2} = \sum_{k=0}^{N-1} f[k] \omega^{-k} (-1)^{-k},$$

$$\mathcal{F} f\left[\frac{N}{2} - 1\right] = \sum_{k=0}^{N-1} f[k] \omega^{-k} \left[\frac{N}{2} - 1\right] = \sum_{k=0}^{N-1} f[k] \omega^{k} \omega^{-Nk/2} = \sum_{k=0}^{N-1} f[k] \omega^{k} (-1)^{-k}.$$
Comparing the two calculations, we see that
\[ \mathcal{F}f\left[ \frac{N}{2} + 1 \right] = \mathcal{F}f\left[ \frac{N}{2} - 1 \right]. \]
Similarly, we get
\[ \mathcal{F}f\left[ \frac{N}{2} + 2 \right] = \sum_{k=0}^{N-1} f[k]\omega^{-2k}(-1)^{-k}, \quad \mathcal{F}f\left[ \frac{N}{2} - 2 \right] = \sum_{k=0}^{N-1} f[k]\omega^{2k}(-1)^{-k}, \]
so that
\[ \mathcal{F}f\left[ \frac{N}{2} + 2 \right] = \mathcal{F}f\left[ \frac{N}{2} - 2 \right]. \]
This pattern persists down to the pair of values
\[ \mathcal{F}f[1] = \sum_{k=0}^{N-1} f[k]\omega^{-k} \]
and
\[ \mathcal{F}f[N-1] = \sum_{k=0}^{N-1} f[k]\omega^{-(N-1)k} = \sum_{k=0}^{N-1} f[k]\omega^{-kN} = \sum_{k=0}^{N-1} f[k]\omega^k, \]
i.e., to
\[ \mathcal{F}f[1] = \mathcal{F}f[N-1]. \]
This is where it stops; recall that \( \mathcal{F}f[0] \) is the sum of the components of \( f \).

This result is analogous to the symmetry relation \( \mathcal{F}f(-s) = \overline{\mathcal{F}f(s)} \) in the continuous case, which we’ll come back to. Because of it, when the spectrum is indexed from 0 to \( N - 1 \), the convention is to say that the frequencies from \( m = 1 \) to \( m = N/2 - 1 \) are the positive frequencies and those from \( N/2 + 1 \) to \( N - 1 \) are the negative frequencies. Whatever adjectives one uses, the important upshot is that for a real input \( f \) all the information in the spectrum is in:

- The first component \( \mathcal{F}f[0] \) (the “DC” component, the sum of components of the input).
- The components \( \mathcal{F}f[1], \mathcal{F}f[2], \ldots, \mathcal{F}f[N/2 - 1] \).
- The special value \( \mathcal{F}f[N/2] \) (the alternating sum of the components of the input).

The remaining components of \( \mathcal{F}f \) are just the complex conjugates of those from 1 to \( N/2 - 1 \).
Fine. But periodicity makes this much more clear. It makes honest negative frequencies correspond to negative frequencies “by convention.” Suppose we have a periodic input $f$ and periodic output $F = \mathcal{F} f$ indexed from $-(N/2)+1$ to $N/2$. We would certainly say in this case that the negative frequencies go from $-(N/2)+1$ to $-1$, with corresponding outputs $F[-(N/2)+1], F[-(N/2)+2], \ldots, F[-1]$. Where do these frequencies go if we reindex from $0$ to $N-1$? Using periodicity,

$$F[-\frac{N}{2} + 1] = F[-\frac{N}{2} + N] = F[\frac{N}{2} + 1].$$

$$F[-\frac{N}{2} + 2] = F[-\frac{N}{2} + 2 + N] = F[\frac{N}{2} + 2],$$

and so on up to $F[-1] = F[-1 + N]$.

The “honest” negative frequencies at $-(N/2) + 1, \ldots, -1$ are by periodicity the “negative frequencies by convention” at $N/2 + 1, \ldots, N - 1$. Convince yourself as well why it’s more natural to index from $-(N/2) + 1$ up to $N/2$, as we have, rather than from $-N/2$ to $(N/2) - 1$.

We worked through this in so much detail to help you understand and interpret what you’re seeing when you compute and plot DFTs using any of the standard software packages, where indexing (usually) goes from $1$ to $N$. Armed with this new understanding, look at Problem 7.5.

**7.7.1. Different definitions for the DFT.** The purpose of this section is to get a precise understanding of the following statement:

“The DFT can be defined over any set of $N$ consecutive indices,”

mentioned earlier. Once understood precisely, this statement is behind the different indexing conventions that are in use for the DFT. Sort of mathy, but a nice exercise in periodicity, which is all that’s involved. You can skip the derivation and still have a happy life.

For periodic functions $f(t)$ in the continuous setting, say functions of period $1$, the $n$th Fourier coefficient is

$$\hat{f}(n) = \int_0^1 e^{-2\pi i nt} f(t) \, dt.$$

The periodicity of $f(t)$ implies that $\hat{f}(n)$ can be obtained by integrating over any interval of length $1$, meaning

$$\hat{f}(n) = \int_a^{a+1} e^{-2\pi i nt} f(t) \, dt$$

for any $a$. Morally, we’re looking for the discrete version of this. Morally, we’ll give the same argument that we did back in Chapter 1, which goes: take the derivative
with respect to $a$:

$$\frac{d}{da} \int_{a}^{a+1} e^{-2\pi int} f(t) \, dt = e^{-2\pi i(a+1)n} f(a + 1) - e^{-2\pi i a n} f(a)$$

$$= e^{-2\pi i a n} e^{-2\pi i n} f(a + 1) - e^{-2\pi i a n} f(a)$$

$$= e^{-2\pi i a n} f(a) - e^{-2\pi i a n} f(a) = 0,$$

using $e^{2\pi in} = 1$ and the periodicity of $f(t)$. Since the derivative is identically zero, the value of the integral is the same for any value of $a$. In particular, taking $a = 0$,

$$\int_{a}^{a+1} e^{-2\pi int} f(t) \, dt = \int_{0}^{1} e^{-2\pi int} f(t) \, dt = \hat{f}(n).$$

Back to the discrete case. Take a good backward glance at what’s been done and start the whole thing over, so to speak, by giving a more general definition of the DFT:

We consider discrete signals $f$ that are periodic of period $N$. Let $\mathcal{P}$ and $\mathcal{Q}$ be index sets of $N$ consecutive integers, say,

$$\mathcal{P} = [p : p + N - 1], \quad \mathcal{Q} = [q : q + N - 1].$$

The DFT based on $\mathcal{P}$ and $\mathcal{Q}$ is defined by

$$\mathcal{G}f[m] = \sum_{k \in \mathcal{P}} f[k] \omega^{-mk} = \sum_{k = p}^{p+N-1} f[k] \omega^{-mk}, \quad m \in \mathcal{Q}.$$

I’ve called the transform $\mathcal{G}$ to distinguish it from $\mathcal{F}$, which in the present setup corresponds to the special choice of index sets $\mathcal{P} = \mathcal{Q} = [0 : N - 1]$.

Since $f$ is periodic of period $N$, knowing $f$ on any set of $N$ consecutive numbers determines it everywhere. We hope the same will be true of a transform of $f$, but this is what must be shown. Thus one wants to establish that the definition of $\mathcal{G}$ is independent of $\mathcal{P}$ and $\mathcal{Q}$. This is a sharper version of the informal statement in the first quote, above, but we have to say what “independent of $\mathcal{P}$ and $\mathcal{Q}$” means.

First $\mathcal{Q}$. Allowing for a general set $\mathcal{Q}$ of $N$ consecutive integers to index the values $\mathcal{G}f[m]$ doesn’t really come up in applications and is included in the definition for the sake of generality. We can dispose of the question quickly. To show that the transform is independent of $\mathcal{Q}$, we do the following:

(a) Extend $\mathcal{G}f$ to be periodic of period $N$; thus $\mathcal{G}f[m]$ is defined, by periodicity, for all integers $m$.

(b) Use the periodicity of the exponentials in the definition of the transform to show that the extension is again given by the formula for $\mathcal{G}f$. 
To wit, for (b), let \( m \in \mathbb{Z} \) and write \( m = n + \ell N \) where \( n \in \mathcal{Q} \) and \( \ell \) is an integer. Then
\[
\sum_{k=p}^{p+N-1} f[k] \omega^{-mk} = \sum_{k=p}^{p+N-1} f[k] \omega^{-(n+\ell N)k} = \sum_{k=p}^{p+N-1} f[k] \omega^{-mk} \omega^{-\ell Nk} = \sum_{k=p}^{p+N-1} f[k] \omega^{-nk} \quad \text{(since } \omega^{-\ell Nk} = 1 \text{)}
\]
\[
= \mathcal{G} f[n] \quad \text{(from the original definition of } \mathcal{G}, \text{ which needs } n \in \mathcal{Q})
\]
\[
= \mathcal{G} f[n + \ell N] \quad \text{(because we’ve extended } \mathcal{G} \text{ to be periodic)}
\]
\[
= \mathcal{G} f[m].
\]

Reading from bottom right to top left \(^7\) then shows that
\[
\mathcal{G} f[m] = \sum_{k=p}^{p+N-1} f[k] \omega^{-mk}
\]
for all \( m \in \mathbb{Z} \). That’s all \( m \in \mathbb{Z} \), with a formula for \( \mathcal{G} \) depending only on \( \mathcal{P} \). That’s the point, and it shows that the definition of \( \mathcal{G} \) is independent of the initial choice of the index set \( \mathcal{Q} \).

Now for independence from \( \mathcal{P} \). Write, more compactly,
\[
\mathcal{G}_p f = \sum_{k=p}^{p+N-1} f[k] \omega^{-k}.
\]

Analogous to taking the derivative with respect to \( a \) of the integral from \( a \) to \( a + 1 \) that defines the Fourier coefficient, apply the difference operator to \( \mathcal{G}_p f \) as a function of \( p \), meaning \( \Delta \mathcal{G}_p f = \mathcal{G}_{p+1} f - \mathcal{G}_p f \). Then
\[
\Delta \mathcal{G}_p f = \mathcal{G}_{p+1} f - \mathcal{G}_p f
\]
\[
= \sum_{k=p+1}^{p+N} f[k] \omega^{-k} - \sum_{k=p}^{p+N-1} f[k] \omega^{-k}
\]
\[
= f[p + N] \omega^{-(p+N)} - f[p] \omega^{-p} \quad \text{(all other terms cancel)}
\]
\[
= f[p + N] \omega^{-p}\omega^{-N} - f[p] \omega^{-p}
\]
\[
= f[p] \omega^{-p} - f[p] \omega^{-p} = 0
\]
using \( \omega^{-N} = 1 \) and the periodicity of \( f \). Since \( \Delta \mathcal{G}_p f = 0 \), it follows that the values of \( \mathcal{G}_p f \) are independent of \( p \), which is just what we wanted to show. In particular, taking \( p = 0 \) we have
\[
\mathcal{G}_p f = \sum_{k=p}^{p+N-1} f[k] \omega^{-k} = \sum_{k=0}^{N-1} f[k] \omega^{-k} = \mathcal{F} f.
\]

\(^7\) As in: Where did we start; where did we finish?
In other words, any DFT is the DFT as we originally defined it, when considered on periodic signals. And we have made the argument to show this look very much like the one for the continuous case.

Finally, in this same circle of ideas, so, too, do we have: “the inverse DFT can be defined over any set of $N$ consecutive indices.” No argument from me, and no argument given here.

### 7.8. Getting to Know Your DFT, Better

Time to push harder on the similarities and differences between the discrete and continuous settings. Start with the following:

**7.8.1. Reversed signals and their DFTs.** For a discrete signal, $f$, defined on the integers, periodic or not, the corresponding reversed signal, $f^-$, is defined by

$$f^-[m] = f[-m].$$

If $f$ is periodic of period $N$, as we henceforth again assume, and if we write it as the vector

$$f = (f[0], f[1], \ldots, f[N - 1]),$$

so that $f^- = (f[0], f[-1], \ldots, f[-N + 1])$, then by periodicity

$$f^- = (f[N], f[N - 1], \ldots, f[1]) \text{ (using } f[N] = f[0]),$$

which makes the description of $f^-$ as “reversed” even more apt (though, as in many irritating instances, the indexing is a little off). Defined directly in terms of its components this is

$$f^-[n] = f[N - n]$$

and this formula is good for all integers $n$. This description of $f^-$ is often quite convenient.

Note that reversing a signal satisfies the principle of superposition (is linear as an operation on signals):

$$(f + g)^- = f^- + g^- \text{ and } (\alpha f)^- = \alpha f^-.$$ 

It’s even more than that, for we also have

$$(fg)^- = (f^-)(g^-).$$ 

Let’s consider two special cases of reversed signals. First, clearly

$$\delta^-_0 = \delta^-_0,$$

and though we’ll pick up more on evenness and oddness later, this says that $\delta^-_0$ is even. For the shifted $\delta$,

$$\delta^-_k = \delta^-_{-k}.$$

I’ll let you verify that. With this result we can write

$$f^- = \left( \sum_{k=0}^{N-1} f[k] \delta^-_k \right) = \sum_{k=0}^{N-1} f[k] \delta^-_{-k}.$$
One might say that the $\delta_k$ are a basis for the forward signals and the $\delta_{-k}$ are a basis for the reversed signals.

Next let’s look at $\omega$. First, we have

$$\omega^\tau = (\omega[N], \omega[N-1], \omega[N-2], \ldots, \omega[1]) = (1, \omega^{N-1}, \omega^{N-2}, \ldots, \omega).$$

But now notice (as we could have noticed earlier) that

$$\omega^{N-1} \omega = \omega^N = 1 \Rightarrow \omega^{N-1} = \omega^{-1}. $$

Likewise

$$\omega^{N-2} \omega^2 = \omega^N = 1 \Rightarrow \omega^{N-2} = \omega^{-2}. $$

Continuing in this way we see, very attractively,

$$\omega^\tau = (1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(N-1)}) = \omega^{-1}. $$

In the same way we find, equally attractively,

$$(\omega^k)^\tau = \omega^{-k}. $$

Of course then also

$$(\omega^{-k})^\tau = \omega^k. $$

**Duality.** This has an important consequence for the DFT, our first discrete duality result. Let’s consider $Ff^\tau$, the DFT of the reversed signal. To work with the expression for $Ff^\tau$ we’ll need to use periodicity of $f$ and do a little fancy footwork changing the variable of summation in the definition of the DFT. Here’s how it goes:

$$Ff^\tau = \sum_{k=0}^{N-1} f^\tau[k] \omega^{-k}$$

$$= \sum_{k=0}^{N-1} f[N-k] \omega^{-k} \quad \text{ (reversing } f)$$

$$= \sum_{\ell=0}^{N} f[\ell] \omega^{\ell-N} \quad \text{ (letting } \ell = N-k)$$

$$= \sum_{\ell=0}^{N} f[\ell] \omega^{\ell} \quad \text{ (since } \omega^{-N} = 1).$$

But using $f[N] = f[0]$ and $\omega^N = \omega^0 = 1$ we can clearly write

$$\sum_{\ell=0}^{N} f[\ell] \omega^{\ell} = \sum_{\ell=0}^{N-1} f[\ell] \omega^{\ell}$$

$$= \left( \sum_{\ell=0}^{N-1} f[\ell] \omega^{-\ell} \right)^\tau = (Ff)^\tau. $$

We have shown $Ff^\tau = (Ff)^\tau$. Cool. A little drawn out, but cool.
This then tells us that
\[ \mathcal{F} \omega^{-k} = (\mathcal{F} \omega^k)^{-} = (N \delta_k)^{-} = N \delta_{-k}. \]

In turn, from here we get a second duality result. Start with
\[ \mathcal{F} f = \sum_{k=0}^{N-1} f[k] \omega^{-k} \]
and apply \( \mathcal{F} \) again. This produces
\[ \mathcal{F} \mathcal{F} f = \sum_{k=0}^{N-1} f[k] \mathcal{F} \omega^{-k} = N \sum_{k=0}^{N-1} f[k] \delta_{-k} = N f^{-}. \]

To give the two results their own display:

- Duality relations for the DFT are
  \[ \mathcal{F} f^{-} = (\mathcal{F} f)^{-} \quad \text{and} \quad \mathcal{F} \mathcal{F} f = N f^{-}. \]

We’ll do more on reversed signals, evenness and oddness, etc., but it’s interesting to note that the duality results also lead to the definition of the inverse DFT. We take our cue from the duality results in the continuous case that say
\[ \mathcal{F}^{-1} f = \mathcal{F} f^{-} = (\mathcal{F} f)^{-}. \]

This equation tells us how we might try defining the inverse DFT. Because I know what’s going to happen, I’ll put a factor of \( 1/N \) where it belongs and claim that
\[ \mathcal{F}^{-1} f = \frac{1}{N} \mathcal{F} f^{-}. \]

So equivalently
\[ \mathcal{F}^{-1} f = \frac{1}{N} (\mathcal{F} f)^{-} \quad \text{and also} \quad \mathcal{F}^{-1} f = \frac{1}{N} \sum_{n=0}^{N-1} f[n] \omega^n, \]
the second formula being what we know from our earlier work. Let’s see why \( \mathcal{F}^{-1} f = (1/N) \mathcal{F} f^{-} \) really does give us an inverse of \( \mathcal{F} \).

It’s clear that \( \mathcal{F}^{-1} \) as defined this way is linear. We also need to know
\[ \mathcal{F}^{-1} \omega^{-k} = \frac{1}{N} \mathcal{F} (\omega^{-k})^{-} \quad \text{(definition of \( \mathcal{F}^{-1} \))} \]
\[ = \frac{1}{N} \mathcal{F} \omega^k \quad \text{(using \( (\omega^{-k})^{-} = \omega^k \))} \]
\[ = \frac{1}{N} N \delta_k = \delta_k. \]

With this,
\[ \mathcal{F}^{-1} \mathcal{F} f = \mathcal{F}^{-1} \left( \sum_{k=0}^{N-1} f[k] \omega^{-k} \right) \]
\[ = \sum_{k=0}^{N-1} f[k] \mathcal{F}^{-1} \omega^{-k} = \sum_{k=0}^{N-1} f[k] \delta_k = f. \]
A similar calculation shows that
\[ \mathcal{F}_\mathcal{F}^{-1} f = f. \]

Good show. We have shown that \((1/N)\mathcal{F} f\) really, really does give an inverse to \(\mathcal{F}\). A longer route to the inverse DFT, but it took us past the signposts marking the way that said, “The discrete corresponds to the continuous. Use what you know.”

Let’s also note that
\[ \mathcal{F}^{-1} \delta_k = \frac{1}{N} \omega^k. \]

**Symmetries.** Remember those evenness, oddness results for the Fourier transform? Same sort of thing for the DFT. A discrete signal \(f\) is even if \(f^- = f\), i.e., if \(f[-n] = f[n]\). It’s odd if \(f^- = -f\), i.e., \(f[-n] = -f[n]\). The signal and its DFT have the same symmetry. If, for example, \(f\) is even, then using duality,
\[ (\mathcal{F} f)^- = \mathcal{F} f^- = \mathcal{F} f. \]
The same sort of argument shows that if \(f\) is odd, then so is \(\mathcal{F} f\).

Finally, if \(f\) is a real signal, then DFT has the conjugate symmetry that we saw in the continuous setting, namely,
\[ \mathcal{F} f[-n] = \mathcal{F} f[n]. \]

Let’s go through this, if only for some practice in duality, sums, and summing over any consecutive \(N\) indices to compute the DFT. Onward!

\[
(\mathcal{F} f)^- = \mathcal{F} f^- \quad \text{(duality)}
\]
\[
= \sum_{k=0}^{N-1} f^-[k] \omega^{-k}
\]
\[
= \sum_{k=0}^{N-1} f[-k] \omega^{-k}
\]
\[
= \sum_{\ell=0}^{-(N-1)} f[\ell] \omega^{-\ell} \quad \text{(letting } \ell = -k)\]
\[
= \left( \sum_{\ell=0}^{-(N-1)} f[\ell] \omega^{-\ell} \right) \quad \text{(using } \omega^{-\ell} = \overline{\omega}^\ell, \text{ and } \overline{f} = f \text{ because } f \text{ is real})\]
\[
= \mathcal{F} \overline{f} \quad \text{(because we can sum over any block of } N \text{ indices).}
\]

We have a few additional properties and formulas for the DFT that are analogous to those in the continuous case. This will mostly be a listing of results, often without much discussion. Use it as a reference.
7.8.2. Parseval’s identity. There’s a version of Parseval’s identity for the DFT, featuring an extra factor of $N$ that one has to keep track of:

$$\mathcal{F} f \cdot \mathcal{F} g = N (f \cdot g).$$

That’s the dot product. The derivation goes like this, using properties of the complex inner product and the orthogonality of the discrete complex exponentials:

$$\mathcal{F} f \cdot \mathcal{F} g = \left( \sum_{k=0}^{N-1} f[k] \omega^{-k} \right) \cdot \left( \sum_{\ell=0}^{N-1} g[\ell] \omega^{-\ell} \right) = \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} f[k] \overline{g[\ell]} (\omega^{-k} \cdot \omega^{-\ell}) = N \sum_{k=0}^{N-1} f[k] \overline{g[k]} = N (f \cdot g).$$

If $f = g$, then the identity becomes

$$||\mathcal{F} f||^2 = N ||f||^2.$$ 

Parseval’s identity is still another way of saying that $\mathcal{F}$ is almost unitary as a matrix. A unitary matrix $A$ has the property that

$$A \cdot \cdot A g = f \cdot g,$$

so $A$ preserves the inner product. $\mathcal{F}$ almost does this.

7.8.3. Shifts and the shift theorem. In formulating the shift theorem for the DFT it’s helpful to introduce the delay operator for a discrete signal $f$. For an integer $p$ we define the delayed signal $\tau_p f$ by

$$\tau_p f[n] = f[n - p].$$

The version of the shift theorem for the DFT looks just like its continuous cousin:

$$\mathcal{F} (\tau_p f) = \omega^{-p} \mathcal{F} f.$$

The verification of this is left as a problem. Note that we need $f$ to be defined on all integers for shifting and the shift theorem to make sense.

7.8.4. The modulation theorem. Modulation also works as in the continuous case. The modulation of a discrete signal $f$ is, by definition, a signal of the form

$$(\omega^n f)[m] = \omega^n[m] f[m].$$

We can find $\mathcal{F}(\omega^n f)$ directly from the definition:

$$\mathcal{F}(\omega^n f) = \sum_{k=0}^{N-1} \omega^n[k] f[k] \omega^{-k},$$

and so the $m$th component is

$$\mathcal{F}(\omega^n f)[m] = \sum_{k=0}^{N-1} f[k] \omega^{kn} \omega^{-km} = \sum_{k=0}^{N-1} f[k] \omega^{-k(m-n)}.$$
But if we shift $f$ by $n$, we obtain
\[ \tau_n(\mathcal{F}f) = \tau_n \left( \sum_{k=0}^{N-1} f[k] \omega^{-k} \right) = \sum_{k=0}^{N-1} f[k] \tau_n \omega^{-k}, \]
and the $m$th component of the right-hand side is
\[ \left( \sum_{k=0}^{N-1} f[k] \tau_n \omega^{-k} \right)[m] = \sum_{k=0}^{N-1} f[k] (\tau_n \omega^{-k})[m] = \sum_{k=0}^{N-1} f[k] \omega^{-k(m-n)}, \]
just as we had above. We conclude that
\[ \mathcal{F}(\omega^n f) = \tau_n(\mathcal{F}f). \]
This is the modulation theorem.

7.8.5. Convolution. Convolution combined with the DFT is the basis of digital filtering. This may be the greatest employment of computation in the known universe. It’s a topic in the next chapter.

In considering how convolution is to be defined, let me ask again the question we asked in the continuous case: How can we use one signal to modify another to result in scaling the frequency components? In the continuous case we discovered convolution in the time domain precisely by looking at multiplication in the frequency domain, and we’ll do the same thing now.

Given $F$ and $G$, we can consider their componentwise product $FG$. The question is:

- If $F = \mathcal{F}f$ and $G = \mathcal{F}g$, is there an $h$ so that $FG = \mathcal{F}h$?

The technique to analyze this is to interchange the order of summation, much as we often interchanged the order of integration (e.g., $dx \, dy$ instead of $dy \, dx$) in deriving formulas for Fourier integrals. We did exactly that in the process of coming up with convolution in the continuous setting.

For the DFT and the question we have posed,
\[ (\mathcal{F}^{-1}(FG))[m] = \frac{1}{N} \sum_{n=0}^{N-1} F[n] G[n] \omega^{mn} \]
\[ = \frac{1}{N} \sum_{n=0}^{N-1} \left( \sum_{k=0}^{N-1} f[k] \omega^{-kn} \right) \left( \sum_{\ell=0}^{N-1} g[\ell] \omega^{-\ell n} \right) \omega^{mn} \]
\[ = \sum_{k=0}^{N-1} f[k] \sum_{\ell=0}^{N-1} g[\ell] \left[ \frac{1}{N} \sum_{n=0}^{N-1} \omega^{-kn} \omega^{-\ell n} \omega^{mn} \right] \]
\[ = \sum_{k=0}^{N-1} f[k] \sum_{\ell=0}^{N-1} g[\ell] \left[ \frac{1}{N} \sum_{n=0}^{N-1} \omega^{n(m-k-\ell)} \right]. \]
Now look at the final sum in brackets. As in earlier calculations, this is a finite geometric series whose sum is $N$ when $m-k-\ell \equiv 0 \mod N$ and is zero if $m-k-\ell \not\equiv 0 \mod N$. This takes the periodicity of the inputs and outputs into account, and
we really must work modulo $N$ in order to do that because $m - k - \ell$ could be less than 0 or bigger than $N - 1$. Thus what survives is when $\ell \equiv m - k \mod N$, and the final line above becomes

$$\sum_{k=0}^{N-1} f[k] g[m - k].$$

Therefore, if

$$h[m] = \sum_{k=0}^{N-1} f[k] g[m - k], \quad m = 0, \ldots, N - 1,$$

then $Fh = FG$. Again notice that the periodicity of $g$ has to be used in defining $h$ because the index on $g$ will be negative for $m < k$. Also observe that $h$ is periodic.

To summarize:

- **Convolution of discrete signals.** Let $f$ and $g$ be periodic discrete signals. Define the convolution of $f$ and $g$ to be the periodic discrete signal

\[(f * g)[m] = \sum_{k=0}^{N-1} f[k] g[m - k].\]

Then

$$F(f * g) = (FF)(FG).$$

The product on the right-hand side is the componentwise product of the DFTs. You can verify (if you have the patience) the various algebraic properties of convolution, namely linearity, commutativity, and associativity.

- It’s also true that the DFT turns a product into a convolution:

$$F(fg) = \frac{1}{N}(FF*FG).$$

(An extra factor of $1/N$. Agony.)

This equation can be derived from the first convolution property, using duality. Let $F = F^{-1} f$ and $G = F^{-1} g$. Then $f = FF$, $g = FG$, and

$$fg = (FF)(FG) = F(FG).$$

Hence

$$F(fg) = FF(FG) = N(FG)^{-}\frac{1}{N}(FG)^{-} = \frac{1}{N}(FF*FG).$$

So you know where the $N$’s go, here are the corresponding results for the inverse DFT. You can derive them via duality:

$$F^{-1}(f * g) = N(F^{-1}fF^{-1}g),$$

$$F^{-1}(fg) = F^{-1}fF^{-1}g.$$
Shifts and convolution. We note one general property combining convolution with delays, namely that the discrete shift works with discrete convolution just as it does in the continuous case:

\[(\tau_p f) * g)[n] = \sum_{k=0}^{N-1} \tau_p f[n - k]g[k] \]

\[= \sum_{k=0}^{N-1} f[n - k - p]g[k] = (f * g)[n - p] = \tau_p(f * g)[n].\]

Thus, since convolution is commutative,

\[(\tau_p f) * g = \tau_p(f * g) = f * (\tau_p g).\]

7.8.6. More properties of \(\delta\). Two of the most useful properties of the continuous \(\delta\), if we can get away with the term “continuous” in connection with \(\delta\), are what it does when multiplied or convolved with a smooth function. For the discrete \(\hat{\delta}\) we have similar results. For multiplication,

\[f\hat{\delta}_0 = (f[0] \cdot 1, f[1] \cdot 0, \ldots, f[N - 1] \cdot 0) = (f[0], 0, \ldots, 0) = f[0]\hat{\delta}_0.\]

For convolution,

\[(f * \hat{\delta}_0)[m] = \sum_{n=0}^{N-1} f[m - n]\hat{\delta}_0[n] = f[m];\]

i.e., \(f * \hat{\delta}_0 = f\).

There are analogous properties for the shifted discrete \(\hat{\delta}\). For multiplication,

\[f\hat{\delta}_k = f[k]\hat{\delta}_k.\]

For convolution,

\[f * \hat{\delta}_k = f * \tau_k\hat{\delta}_0 = \tau_k(f * \hat{\delta}_0) = \tau_kf,\]

or in components

\[(f * \hat{\delta}_k)[m] = f[m - k],\]

again in agreement with what we would expect from the continuous case.

Observe, incidentally, that

\[\hat{\delta}_k = \tau_k\hat{\delta}_0, \quad \hat{\delta}_k[m] = \hat{\delta}_0[m - k],\]

so a shifted discrete \(\hat{\delta}\) really does appear as \(\hat{\delta}\) delayed by \(k\).

There’s more. Note that

\[\hat{\delta}_p \hat{\delta}_q = \begin{cases} \hat{\delta}_p, & p = q, \\ 0, & p \neq q, \end{cases}\]

and that

\[\hat{\delta}_p * \hat{\delta}_q = \hat{\delta}_{p+q} .\]
The former operation, multiplying δ’s, is against the law in the continuous case, but not in the discrete case, where we’re just multiplying honest discrete signals. A cute observation making use of the first result and the convolution theorem is that

\[ \omega^p \ast \omega^q = \begin{cases} N\omega^p, & p = q, \\ 0, & p \neq q. \end{cases} \]

Of course you can also see this directly, but it might not occur to you to look.

### 7.9. The Discrete Rect and Its DFT

Unlike for the Fourier transform, where computing a storehouse of particular transforms is necessary for an understanding and for applications, we’ve only explicitly computed and needed the DFT for δ’s and for discrete complex exponentials. We want to add to that short list the DFT of a discrete rectangle function, a discrete version of the very first example that we computed for the continuous Fourier transform. There are times when such an expression is useful, say in the design of digital filters as we’ll discuss in the next chapter, or in a discrete version of the Nyquist-Shannon sampling theorem as we’ll discuss in the next section. The formulas work out nicely, by and large, and further advance the notion that the discrete case can be made to look like the continuous case.

In this section we’ll index discrete signals from \(-\frac{N}{2} + 1\) to \(\frac{N}{2}\), so in particular we’re assuming that \(N\) is even. The discrete rectangle is a sum of shifted δ’s going from \(-p/2\) to \(p/2\), \(0 < p < N/2\), so we also assume that \(p\) is even. We let

\[ \Pi_{p,\alpha} = \alpha(\delta_{p/2} + \delta_{-p/2}) + \sum_{k=-p/2+1}^{p/2-1} \delta_k, \]

where \(\alpha\) is any real number. Why the extra parameter \(\alpha\) in setting the value at the endpoints ±\(p/2\)? In the continuous case one also encounters different normalizations of the rect function at the points of discontinuity, but it hardly makes a difference in any formulas and calculations; most of the time there’s an integral involved, and changing the value of a function at a few points has no effect on the value of the integral. Not so in the discrete case, where sums instead of integrals are the operations that one encounters. With an eye toward flexibility in applications, we’re allowing an \(\alpha\).

Just as \(\Pi_{p,\alpha}\) comes in two parts, so, too, does its DFT. The first part is easy:

\[ \mathcal{F}(\alpha(\delta_{p/2} + \delta_{-p/2})) = \alpha(\omega^{-p/2} + \omega^{p/2}) = 2\alpha \text{Re}\{\omega^{p/2}\}; \]

hence

\[ \mathcal{F}(\alpha(\delta_{p/2} + \delta_{-p/2}))[m] = 2\alpha \cos(\pi pm/N). \]

The second part takes more work:

\[ \mathcal{F}\left( \sum_{k=-p/2+1}^{p/2-1} \delta_k \right)[m] = \sum_{k=-p/2+1}^{p/2-1} \omega^{-k}[m] = \sum_{k=-p/2+1}^{p/2-1} \omega^{-km}, \]

8Here’s a hint: \(\alpha = 1/2\) is often the best choice — but not always. This opens up wounds from the endpoint battles in the continuous case, I know.
and the right-hand side is a geometric sum. Before plunging in, we have, directly,

\[ \mathcal{F}_{\Pi_{p,\alpha}}[0] = 2\alpha + p - 1. \]

Now, if you look at Appendix C, you will see a formula (in the problems) that helps. When \( m \neq 0 \),

\[ \sum_{k=-p/2+1}^{p/2-1} \omega^{-km} = \frac{\omega^{-m(-\frac{p}{2}+1)} - \omega^{-m\frac{p}{2}}}{1 - \omega^{-m}}. \]

There’s a little hocus pocus that goes into modifying this expression, not unfamiliar to us but I’ll provide the details because I want to write it in a particular form. I’m aiming to bring in a ratio of sines once we substitute \( \omega = e^{2\pi i/N} \).

For the numerator,

\[ \omega^{-m(-\frac{p}{2}+1)} - \omega^{-m\frac{p}{2}} = \omega^{-\frac{mp}{2}}(\omega^{\frac{mp}{2}(p-1)} - \omega^{\frac{mp}{2}(p-1)}) \]

\[ = \omega^{-\frac{mp}{2}}(e^{\pi im(p-1)} - e^{-\pi im(p-1)}) \]

\[ = \omega^{-\frac{mp}{2}}2i \sin\left(\pi m(p-1)/N\right). \]

Same idea for the denominator:

\[ 1 - \omega^{-m} = \omega^{-m/2}(\omega^{m/2} - \omega^{-m/2}) = \omega^{-m/2}2i \sin(\pi m/N). \]

Putting it all together we get

\[ \mathcal{F}_{\Pi_{p,\alpha}}[m] = \begin{cases} 2\alpha + p - 1, & m = 0, \\ 2\alpha \cos(\pi pm/N) + \frac{\sin(\pi m(p-1)/N)}{\sin(\pi m/N)}, & m \neq 0. \end{cases} \]

One additional point before we continue with special cases. As with all of our discrete signals we regard \( \Pi_{p,\alpha} \) as periodic of period \( N \), so the pattern of \( \delta \)'s repeats. Since \( p \) is even and \( p - 1 \) is odd, we observe that \( \mathcal{F}_{\Pi_{p,\alpha}} \) is likewise periodic of period \( N \), which it had better be. Because \( p - 1 \) is odd, both the numerator and the denominator of the second term change sign if \( m \) is replaced by \( m + N \), while, because \( p \) is even, the cosine term is unchanged. Thus the equations above should actually read \( m \equiv 0 \mod N \) and \( m \not\equiv 0 \mod N \). The DFT is also even, which it had better be.

The most common choices of \( \alpha \) are \( \alpha = 1 \) and \( \alpha = 1/2 \). (If \( \alpha = 0 \), we just have a thinner version of a \( \Pi \) with \( \alpha = 1 \).) For \( \alpha = 1 \) we use the addition formula for sines to write

\[ 2 \cos(\pi pm/N) \sin(\pi m/N) + \sin(\pi(p-1)m/N) = \sin(\pi(p+1)m/N). \]

Thus

\[ \mathcal{F}_{\Pi_{p,1}}[m] = \begin{cases} p + 1, & m \equiv 0 \mod N, \\ \frac{\sin(\pi(p+1)m/N)}{\sin(\pi m/N)}, & m \not\equiv 0 \mod N. \end{cases} \]
For $\alpha = 1/2$, 
\[ \cos(\pi pm/N) \sin(\pi m/N) + \sin(\pi(p - 1)m/N) = \cos(\pi m/N) \sin(\pi pm/N), \]
and 
\[ \mathcal{F} \Pi_{p,1/2}[m] = \begin{cases} 
  p, & m \equiv 0 \mod N, \\
  \frac{\cos(\pi m/N) \sin(\pi pm/N)}{\sin(\pi m/N)}, & m \not\equiv 0 \mod N.
\end{cases} \]
We’ll see this in the next section.

7.10. Discrete Sampling and Interpolation

For the fun of it, let’s derive a discrete version of the sampling theorem.\(^9\) Bolstered by the faith that things should work as in the continuous case, the main ingredients should be

1. a class of bandlimited discrete signals,
2. a discrete III and its DFT,
3. a discrete $\Pi$ and its DFT.

We’ll work with periodic signals of length $N$, and we’ll index from $-\frac{N}{2} + 1$ to $\frac{N}{2}$.

**Bandlimited discrete signals.** We say that a discrete signal $f$ is bandlimited to a set $\mathcal{I}$ of consecutive indices if $\mathcal{F}f[m] = 0$ for $m \not\in \mathcal{I}$. In words, $\mathcal{F}f$ vanishes outside $\mathcal{I}$; there may be places $m \in \mathcal{I}$ where $\mathcal{F}f[m] = 0$ as well (we’re not ruling that out), but for sure the DFT is always 0 off $\mathcal{I}$.

That’s a fairly general definition, but for our purposes, to make things a little simpler and to make them look like the continuous case, let’s suppose that $f$ has $\mathcal{F}f[m] = 0$ for $|m| \geq \frac{p}{2}$. We thus assume that $p$ is even and, for reasons having to do with the discrete III, *we also assume that $p$ divides $N$*. These are definitely restrictions on the argument to follow, but really not severe ones.

**Discrete III.** Remember that the usual $\Pi_p$ is defined as a sum of $\delta$’s spaced $p$ apart,

\[ \Pi_p(x) = \sum_{k=-\infty}^{\infty} \delta(x - kp), \]
and is a periodic distribution of period $p$. To define a discrete III we first of all have to recognize that, like all our discrete signals, it’s periodic of period $N$. So what we want is an evenly spaced set of $\delta$’s *within* one period. We can do this if the spacing $p$ is *a divisor of $N$*, setting

\[ \Pi_p = \sum_{k=0}^{\frac{N}{p}-1} \delta_{kp}. \]

From this comes the divisibility assumption mentioned above.

At an index $m$,
\[ \Pi_p[m] = \begin{cases} 
  1, & m \equiv 0, p, 2p, \ldots, (\frac{N}{p} - 1)p, \mod N, \\
  0, & \text{otherwise}.
\end{cases} \]

\(^9\)Thanks to Aditya Siripuram and William Wu for their contributions to this subject, beyond this section.
The $\delta$'s are spaced $p$-apart, and there are $N/p$ of them in one $N$-period of $\mathbf{III}_p$. Phrased differently, we put a $\delta$ at each $p$th sample. Here’s a plot of $\mathbf{III}_4$ when $N = 20$. Only the $\delta$'s at the positive integers are shown, but remember that the pattern repeats periodically.

Note that if $N$ is a prime number (the first time a number-theoretic consideration has come up!), then in $[0 : N - 1]$ there’s only $\mathbf{III}_1$, the sum of $N$ shifted $\delta$’s, and $\mathbf{III}_N$, consisting of the single $\delta_0$. Also note that $\mathbf{III}_p$ is even.

Just as in the continuous case, the two fundamental properties of $\mathbf{III}_p$ are:

- **Periodizing.**
  For a discrete signal $f$ of length $N$, convolving with $\mathbf{III}_p$,
  \[
  (f * \mathbf{III}_p)[m] = \sum_{k=0}^{N-1} (f * \delta_{kp})[m] = \sum_{k=0}^{N-1} f[m - kp],
  \]
  produces a signal that is periodic of period $p$ (as well as periodic of period $N$).

- **Sampling.**
  For a discrete signal $f$ of length $N$, multiplying by $\mathbf{III}_p$,
  \[
  (f \cdot \mathbf{III}_p)[m] = \sum_{k=0}^{N-1} (f \cdot \delta_{kp})[m] = \sum_{k=0}^{N-1} f[kp]\delta[m - kp],
  \]
  samples $f$ at the points $0, p, 2p, \ldots, (N/p - 1)p$.

What is $F \mathbf{III}_p$? Watch how this works out. Recall that $F \delta_j = \omega^{-j}$, so

\[
F \mathbf{III}_p = \sum_{k=0}^{N-1} \omega^{-kp}.
\]

Then a calculation of the geometric sum gives

\[
F \mathbf{III}_p[m] = \sum_{k=0}^{N-1} \omega^{-kpm} = \sum_{k=0}^{N-1} (\omega^{-pm})^k
\]

\[
= \begin{cases} 
\frac{N}{p}, & pm \equiv 0 \mod N, \\
0, & pm \not\equiv 0 \mod N,
\end{cases}
\]

\[
= \begin{cases} 
\frac{N}{p}, & m \equiv 0 \mod \frac{N}{p}, \\
0, & m \not\equiv 0 \mod \frac{N}{p}.
\end{cases}
\]
Up to the factor $N/p$ this is another $\mathbb{III}$, with spacing $N/p$. Compactly, and beautifully,

$$\mathcal{F} \mathbb{III}_p = \frac{N}{p} \mathbb{III}_{N/p}.$$ 

By duality,

$$\mathcal{F}^{-1} \mathbb{III}_p = \frac{1}{N} (\mathcal{F} \mathbb{III}_p)^{-} = \frac{1}{p} \mathbb{III}_{N/p}.$$

**Discrete rectangle function.** In the last section I introduced the discrete rectangle with a parameter $\alpha$, and the question is which $\alpha$ is suited for our problem. The answer is $\alpha = 1/2$. That’s not obvious till we actually get to the derivation of the sampling theorem, so you’ll have to wait for a few paragraphs. Since we’ll only be working with $\alpha = 1/2$, let’s simplify the notation for now and just write $\Pi_p$ for $\Pi_{p, 1/2}$. It heightens the drama.

Playing the role of a discrete sinc, let

$$\text{dinc}_p[m] = \begin{cases} 
1, & m \equiv 0 \mod N, \\
\frac{1}{p} \cos(\pi m/N) \sin(\pi pm/N) \sin(\pi m/N), & m \not\equiv 0 \mod N.
\end{cases}$$

For the DFT we thus have

$$\mathcal{F} \Pi_p[m] = p \text{dinc}_p[m].$$

This is as close as I could come to making things look like the continuous case, where

$$\mathcal{F} \Pi_p(s) = p \text{sinc} ps.$$ 

For the inverse transform, by duality,

$$\mathcal{F}^{-1} \Pi_p[m] = \frac{1}{N} (\mathcal{F} \Pi_p)^{-}[m] = \frac{p}{N} \text{dinc}_p[-m] = \frac{p}{N} \text{dinc}_p[m],$$

and

$$\mathcal{F} \text{dinc}_p = \frac{N}{p} \Pi_p.$$ 

Here’s a plot of $\text{dinc}_p$ for $p = 6$ and $N = 30.$

---

10I made up the name “dinc,” for discrete sinc. It hasn’t caught on. You can help.
A discrete sampling theorem. We’re all set. Our assumptions are that $F[f][m] = 0$ for $|m| \geq \frac{p}{2}$ and that $p$ divides $N$.

Just as in the continuous setting, forming the convolution $\Pi_p \ast F[f]$ shifts the spectrum off itself, and then cutting off by $\Pi_p$ recovers $F[f]$. We have the fundamental equation

$$F[f] = \Pi_p (\Pi_p \ast F[f]).$$

Now take the inverse DFT:

$$\hat{f} = F^{-1} (\Pi_p(\Pi_p \ast F[f]))$$

$$= (F^{-1} \Pi_p) \ast (F^{-1} (\Pi_p \ast F[f]))$$

$$= \frac{p}{N} dinc_p \ast N \left( \frac{1}{p} \Pi_{N/p} \ast \hat{f} \right)$$

(the factor $N$ comes in because of the convolution theorem for $F^{-1}$, and with $\alpha = 1/2$ we get factors of $p$ cancelling)

$$= dinc_p \ast \sum_{k=0}^{p-1} f[Nk/p] \delta_{Nk/p}.$$

Thus at an index $m$,

$$\hat{f}[m] = \sum_{k=0}^{p-1} f \left[ \frac{Nk}{p} \right] dinc_p \left[ m - \frac{Nk}{p} \right].$$

Perfect.

As a check, you can verify using the definition of $dinc_p$ that the formula returns the sample values:

$$\sum_{k=0}^{p-1} f \left[ \frac{Nk}{p} \right] dinc_p \left[ \frac{N\ell}{p} - \frac{Nk}{p} \right] = f \left[ \frac{N\ell}{p} \right].$$

The next topic along these lines is aliasing, often discussed in the context of upsampling and downsampling. There are some problems on the latter, and a little googling will suggest applications, but a more thorough treatment is for a dedicated course on digital signal processing.

7.11. The FFT Algorithm

I’m sure you’ve been waiting for this. It’s time to consider the practical problem of how the DFT is computed. In this section we’ll go through the famous Cooley-Tukey Fast Fourier Transform algorithm.

You can’t beat the matrix form

$$(F)_{mn} = \omega^{-mn}, \quad m, n = 0, 1, \ldots, N - 1.$$

as a compact way of writing the DFT. It contains all you need to know, and finding the DFT is just multiplication by an $N \times N$ matrix. But you can beat it into submission. The FFT is an algorithm for computing the DFT with fewer than the $N^2$ multiplications that would seem to be required to find $F = F[Nf]$ by multiplying $f$ by the $N \times N$ matrix. Here’s how it works.
Reduction calculations: Merge and sort. To set the stage for a discussion of the FFT algorithm, I thought it first might be useful to see an example of a somewhat simpler but related idea, a way of reducing the total number of steps in a multistep calculation by a clever arranging and rearranging of the individual steps.

Consider the classic (and extremely important) problem of sorting \(N\) numbers from smallest to largest. Say the numbers are

\[
\begin{array}{cccccccc}
\text{Zeroth step} & 5 & 2 & 3 & 1 & 6 & 4 & 8 & 7 \\
\text{First step} & 5 & 2 & 3 & 6 & 4 & 8 & 7 & 1 \\
\text{Second step} & 5 & 3 & 6 & 4 & 8 & 7 & 1 & 2 \\
\text{Third step} & 5 & 6 & 4 & 8 & 7 & 1 & 2 & 3 \\
\text{Fourth step} & 5 & 6 & 8 & 7 & 1 & 2 & 3 & 4 \\
\end{array}
\]

From this list we want to create a new list with the numbers ordered from 1 to 8. The direct assault on this problem is to search the entire list for the smallest number, remove that number from the list and put it in the first slot of the new list, then search the remaining original list for the smallest number, remove that number and put it in the second slot of the new list, and so on:

In general, how many operations does such an algorithm require to do a complete sort? If we have \(N\) numbers, then each step requires roughly \(N\) comparisons — true, the size of the list is shrinking, but the number of comparisons is of the order \(N\) — and we have to repeat this procedure \(N\) times. (There are \(N\) steps, not counting the zeroth step which just inputs the initial list.) Thus the number of operations used to sort \(N\) numbers by this procedure is of the order \(N^2\), or, as it’s usually written \(O(N^2)\) (read “Big Oh”).

The problem with this simple procedure is that the \((n+1)\)st step doesn’t take into account the comparisons done in the \(n\)th step. All that work is wasted, and it’s wasted over and over. An alternate approach, one that makes use of intermediate comparisons, is to sort sublists of the original list, merge the results, and sort again. Here’s how it goes. We’ll assess the efficiency after we see how the method works.\(^{11}\)

Start by going straight through the list and breaking it up into sublists that have just two elements; say that’s step zero.

\[
\begin{array}{cccc}
5 & 2 & 3 & 1 & 6 & 4 & 8 & 7 \\
\end{array}
\]

\(^{11}\)This is a pretty standard topic in many introductory algorithm courses in computer science. I wanted an EE source, and I’m following the discussion in K. Steiglitz, A Digital Signal Processing Primer. I’ve referred to this book before; highly recommended.
Step one is to sort each of these (four) 2-lists, producing two sets of 2-lists, called “top” lists and “bottom” lists just to keep them straight (and we’ll also use top and bottom in later work on the FFT):

\[
\begin{align*}
\text{top} &\{ \begin{array}{c} 2 \\ 1 \\ 5 \\ 3 \end{array} \\
\text{bottom} &\{ \begin{array}{c} 4 \\ 6 \\ 7 \\ 8 \end{array} 
\end{align*}
\]

This step requires four comparisons.

Step two merges these 2-lists into two sorted 4-lists (again called top and bottom). Here’s the algorithm, applied separately to the top and bottom 2-lists. The numbers in the first slots of each 2-list are the smaller of those two numbers. Compare these two numbers and promote the smaller of the two to the first slot of the top (respectively, bottom) 4-list. That leaves a 1-list and a 2-list. Compare the single element of the 1-list to the first element of the 2-list and promote the smaller to the second slot of the top (resp. bottom) 4-list. We’re down to two numbers. Compare them and put them in the three and four slots of the top (resp. bottom) 4-list. For the example we’re working with, this results in the two 4-lists:

\[
\begin{align*}
\text{top} &\{ \begin{array}{c} 1 \\ 2 \\ 3 \\ 5 \end{array} \\
\text{bottom} &\{ \begin{array}{c} 4 \\ 6 \\ 7 \\ 8 \end{array} 
\end{align*}
\]

With this example this step requires five comparisons.

Following this same sort procedure, step three is to merge the top and bottom sorted 4-lists into a single sorted 8-list:

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{array}
\]

With this example this step requires five comparisons.

In this process we haven’t cut down (much) the number of comparisons we have to make at each step, but we have cut down the number of steps from 8 to 3\textsuperscript{12}. In general how many operations are involved in getting to the final list of sorted numbers? It’s not hard to see that the number of comparisons involved in merging two sublists is of the order of the total length of the sublists. Thus with \( N \) numbers total (at the start) the number of comparisons in any merge-sort is \( O(N) \):

\[
\text{number of comparisons in a merge-sort} = O(N).
\]

How many merge-sort steps are there? At each stage we halve the number of sublists, or, working the other way, from the final sorted list each step “up” doubles the number of sublists. Thus if there are \( n \) doublings (\( n \) steps), then \( 2^n = N \), or

\[
\text{number of merge-sort steps} = \log_2 N.
\]

We conclude that

\[
\text{number of steps to sort} \ N \ \text{numbers} = O(N \log N).
\]

\textsuperscript{12} It wasn’t an accident that I took eight numbers here. The procedure is most natural when we have a list of numbers to sort that is a power of 2, something we’ll see again when we look at the FFT.
Also a worthy simplification. (We have called attention to this in the “big Oh.”) If \( N \) is large, this is a huge savings in steps from the \( O(N^2) \) estimate for the simple sort that we did first. For example, if \( N \) is one million, then \( O(N^2) \) is a million million or \( 10^{12} \) steps while \( N \log_{10} N = 10^6 \times 6 \), a mere six million operations.

That’s a correct accounting of the number of operations involved, but why is there a savings in using merge-sort rather than a straight comparison? By virtue of the sorting of sublists, we only need to compare first elements of the sublists in the merge part of the algorithm. In this way the \((n + 1)\)st step takes advantage of the comparisons made in the \( n \)th step, the thing that is not done in the straight comparison method.

A sample calculation of the DFT. As we consider how we might calculate the DFT more efficiently than by straight matrix multiplication, let’s do a sample calculation with \( N = 4 \) so we have on record what the answer is and what we’re supposed to come up with by other means. The DFT matrix is

\[
F_4 = \begin{pmatrix}
1 & 1 & 1 & 1 \\
1 & \omega_4^{-1} & \omega_4^{-2} & \omega_4^{-3} \\
1 & \omega_4^{-2} & \omega_4^{-4} & \omega_4^{-6} \\
1 & \omega_4^{-3} & \omega_4^{-6} & \omega_4^{-9}
\end{pmatrix}.
\]

We want to reduce this as much as possible, “reduction” being somewhat open to interpretation.

Using \( \omega_4 = e^{2\pi i/4} \) and \( \omega_4^{-4} = 1 \) we have

\[
\omega_4^{-6} = \omega_4^{-4}\omega_4^{-2} = \omega_4^{-2} \quad \text{and} \quad \omega_4^{-9} = \omega_4^{-8}\omega_4^{-1} = \omega_4^{-1}.
\]

In general, to simplify \( \omega_4 \) to a power, we take the remainder of the exponent on \( \omega_4 \), and

\[
\begin{pmatrix}
1 & 1 & 1 & 1 \\
1 & \omega_4^{-1} & \omega_4^{-2} & \omega_4^{-3} \\
1 & \omega_4^{-2} & \omega_4^{-4} & \omega_4^{-6} \\
1 & \omega_4^{-3} & \omega_4^{-6} & \omega_4^{-9}
\end{pmatrix} = \begin{pmatrix}
1 & 1 & 1 & 1 \\
1 & \omega_4^{-1} & \omega_4^{-2} & \omega_4^{-3} \\
1 & \omega_4^{-2} & \omega_4^{-4} & \omega_4^{-6} \\
1 & \omega_4^{-3} & \omega_4^{-6} & \omega_4^{-9}
\end{pmatrix}.
\]

But we don’t stop there. Note that

\[
\omega_4^{-2} = (e^{2\pi i/4})^{-2} = e^{-\pi i} = -1,
\]

also a worthy simplification. (We have called attention to this \( N/2 \)th power of \( \omega_N \) before.) So, finally,

\[
F_4 = \begin{pmatrix}
1 & 1 & 1 & 1 \\
1 & \omega_4^{-1} & \omega_4^{-2} & \omega_4^{-3} \\
1 & \omega_4^{-2} & \omega_4^{-4} & \omega_4^{-6} \\
1 & \omega_4^{-3} & \omega_4^{-6} & \omega_4^{-9}
\end{pmatrix} = \begin{pmatrix}
1 & 1 & 1 & 1 \\
1 & \omega_4^{-1} & -1 & -\omega_4^{-1} \\
1 & -1 & 1 & -1 \\
1 & -\omega_4^{-1} & -1 & \omega_4^{-1}
\end{pmatrix}.
\]

Therefore we find

\[
F_4f = \begin{pmatrix}
1 & 1 & 1 & 1 \\
1 & \omega_4^{-1} & -1 & -\omega_4^{-1} \\
1 & -1 & 1 & -1 \\
1 & -\omega_4^{-1} & -1 & \omega_4^{-1}
\end{pmatrix} \begin{pmatrix}
f[0] \\
f[1] \\
f[2] \\
f[3]
\end{pmatrix} = \begin{pmatrix}
\end{pmatrix}.
\]
The matrix looks simpler, true, but it still took 16 multiplications to get the final answer. You can see those two special components that we called attention to earlier, the sum of the inputs in the zero slot and the alternating sum of the inputs in the $N/2 = 2$ slot.

This is about as far as we can go without being smart. Fortunately, there have been some smart people on the case.

### 7.11.1. Half the work is twice the fun: The Fast Fourier Transform.

We agree that the DFT has a lot of structure. The trick to a faster computation of a DFT of order $N$ is to use that structure to rearrange the products to bring in DFTs of order $N/2$. Here’s where we use that $N$ is even; in fact, to make the algorithm most efficient in being applied repeatedly we’ll eventually want to assume that $N$ is a power of 2.

We need a few elementary algebraic preliminaries on the $\omega_N$, all of which we’ve used before. We also need to introduce some temporary (!) notation or we’ll sink in a sea of subscripts and superscripts. Let’s write powers of $\omega$ with two arguments:

$$\omega[p, q] = \omega^q_p = e^{2\pi iq/p}.$$  

I think this will help. It can’t hurt. For our uses $p$ will be $N$, $N/2$, etc.

First, notice that

$$\omega[N/2, -1] = e^{-2\pi i/(N/2)} = e^{-4\pi i/N} = \omega[N, -2].$$  

Therefore powers of $\omega[N/2, -1]$ are even powers of $\omega[N, -1] = \omega_N^{-1}$:

$$\omega[N/2, -n] = \omega[N, -2n]$$

and, in general,

$$\omega[N, -2nm] = \omega[N/2, -nm].$$

What about odd powers of $\omega_N^{-1} = \omega[N, -1]$? An odd power is of the form $\omega[N, -(2n + 1)]$ and so

$$\omega[N, -(2n + 1)] = \omega[N, -1] \omega[N, -2n] = \omega[N, -1] \omega[N/2, -n].$$

Thus also

$$\omega[N, -(2n + 1)m] = \omega[N, -m] \omega[N/2, -nm].$$

Finally, recall that

$$\omega[N, -N/2] = e^{(-2\pi i/N)(N/2)} = e^{-\pi i} = -1.$$  

**Splitting the sums.** Here’s how we’ll use this. For each $m$ we want to split the single sum defining $F[m]$ into two sums, one over the even indices and one over the odd indices:

$$F[m] = \sum_{n=0}^{N-1} f[n] \omega[N, -nm]$$

$$= \sum_{n=0}^{N/2-1} f[2n] \omega[N, -(2n)m] + \sum_{n=0}^{N/2-1} f[2n + 1] \omega[N, -(2n + 1)m].$$
Everything is accounted for here; all the terms are there — make sure you see that. Also, although both sums go from 0 to \(N/2 - 1\), notice that for the first sum the first and last terms are \(f[0]\omega[N, -m]\) and \(f[N - 1]\omega[N, -(N - 1)m]\), and for the second they are \(f[1]\omega[N, -m]\) and \(f[N - 1]\omega[N, -(N - 1)m]\).

Next, according to our observations on powers of \(\omega\) we can also write \(F[m]\) as

\[
F[m] = \sum_{n=0}^{N/2-1} f[2n] \omega[N/2, -nm] + \sum_{n=0}^{N/2-1} f[2n+1] \omega[N/2, -nm] \omega[N, -m]
\]

\[
= \sum_{n=0}^{N/2-1} f[2n] \omega[N/2, -nm] + \omega[N, -m] \sum_{n=0}^{N/2-1} f[2n+1] \omega[N/2, -nm].
\]

Let’s study these sums more closely. There are \(N/2\) even indices and \(N/2\) odd indices, and we appear, in each sum, almost to be taking a DFT of order \(N/2\) of the \(N/2\)-tuples \(f[\text{even indices}]\) and \(f[\text{odd indices}]\). Why “almost?” The DFT of order \(N/2\) accepts as input an \(N/2\)-length signal and returns an \(N/2\)-length signal. But the sums above give all \(N\) entries of the \(N\)-length \(F\) as \(m\) goes from 0 to \(N - 1\). We’re going to do two things to bring in \(F_{N/2}\).

- First, if we take \(m\) to go from 0 to \(N/2 - 1\), then we get the first \(N/2\) outputs \(F[m]\). We write, informally,

\[
F[m] = (F_{N/2} f_{\text{even}})[m] + \omega[N, -m] (F_{N/2} f_{\text{odd}})[m],\quad m = 0, 1, \ldots, N/2 - 1.
\]

That makes sense; \(N/2\)-tuples go in and \(N/2\)-tuples come out.

- Second, what is the story for an index \(m\) in the second half of the range, from \(N/2\) to \(N - 1\)? Instead of letting \(m\) go from \(N/2\) to \(N - 1\) we can write these indices in the form \(m + N/2\), where \(m\) goes from 0 to \(N/2 - 1\). What happens with \(F[m + N/2]\)?

We again have

\[
F[m + N/2] = F_{N/2} f_{\text{even}}[m + N/2] + \omega[N, -(m + N/2)] F_{N/2} f_{\text{odd}}[m + N/2].
\]

But look: for \(F_{N/2}\) the outputs and inputs are periodic of period \(N/2\). That is,

\[
F_{N/2} f_{\text{even}}[m + N/2] = F_{N/2} f_{\text{even}}[m],
\]

\[
F_{N/2} f_{\text{odd}}[m + N/2] = F_{N/2} f_{\text{odd}}[m].
\]

Now by a calculation,

\[
\omega[N, -(m + N/2)] = -\omega[N, -m].
\]

Thus

\[
F[m + N/2] = F_{N/2} f_{\text{even}}[m] - \omega[N, -m] F_{N/2} f_{\text{odd}}[m].
\]

We are there.
7.11. The FFT Algorithm

The description of the FFT algorithm. It’s really very significant what we’ve done here. Let’s summarize:

- We start with an input \( f \) of length \( N \) and want to compute its output \( F = \mathcal{F}_N f \), also of length \( N \).
- The steps we take serve to compute the component outputs \( F[m] \) for \( m = 0, 1, \ldots, N - 1 \) by computing a DFT on two sequences, each of half the length, and arranging properly. That is:
  1. Separate \( f[n] \) into two sequences, the even and odd indices (0 is even), each of length \( N/2 \).
  2. Compute \( \mathcal{F}_{N/2} f_{\text{even}} \) and \( \mathcal{F}_{N/2} f_{\text{odd}} \).
  3. The outputs \( F[m] \) are obtained by arranging the results of this computation according to

\[
F[m] = (\mathcal{F}_{N/2} f_{\text{even}})[m] + \omega[N, -m] (\mathcal{F}_{N/2} f_{\text{odd}})[m],
\]

\[
F[m + N/2] = (\mathcal{F}_{N/2} f_{\text{even}})[m] - \omega[N, -m] (\mathcal{F}_{N/2} f_{\text{odd}})[m],
\]

for \( m = 0, 1, \ldots, N/2 \).

Another look at \( \mathcal{F}_4 \). Let’s do the case \( N = 4 \) as an example, comparing it to our earlier calculation. The first step is to rearrange the inputs to group the even and odd indices. This is done by a permutation matrix

\[
M = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

whose effect is

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
f[0] \\
f[1] \\
f[2] \\
f[3]
\end{pmatrix} =
\begin{pmatrix}
f[0] + f[2] \\
-\omega^{-1}[0] f[0] + \omega^{-1}[2] f[2] \\
\end{pmatrix}.
\]

The matrix \( M \) is defined by what it does to the natural basis (discrete \( \delta \)'s!) \( \delta_0, \delta_1, \delta_2, \delta_3 \) of \( \mathbb{R}^4 \); namely, \( M\delta_0 = \delta_1 \), \( M\delta_1 = \delta_2 \), \( M\delta_2 = \delta_3 \), and \( M\delta_3 = \delta_0 \).

Next, the even and odd indices are fed respectively to two DFTs of order \( 4/2 = 2 \). This is the crucial reduction in the FFT algorithm.

\[
\begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & \omega^{-1} & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & \omega^{-1}
\end{pmatrix}
\begin{pmatrix}
f[0] \\
f[2] \\
f[1] \\
f[3]
\end{pmatrix} =
\begin{pmatrix}
f[0] + f[2] \\
-\omega^{-1}[0] f[0] + \omega^{-1}[2] f[2] \\
\end{pmatrix}.
\]

On the left we have a block diagonal matrix. It’s a \( 4 \times 4 \) matrix with the \( 2 \times 2 \) \( \mathcal{F}_2 \) matrices down the diagonal and zeros everywhere else. We saw this step, but we didn’t see the intermediate result written just above on the right because our formulas passed right away to the reassembly of the \( F[m] \)'s. That reassembly is the final step.
So far we have

\[ F_2 f_{\text{even}} = \begin{pmatrix} f_0 + f[2]_0 \\ f_0 + f[2]_2 \end{pmatrix}, \quad F_2 f_{\text{odd}} = \begin{pmatrix} f[1] + f[3]_0 \\ f[1] + f[3]_2 \end{pmatrix} \]

and in each case the indexing is \( m = 0 \) for the first entry and \( m = 1 \) for the second entry. The last stage, to get the \( F[m] \)'s, is to recombine these half-size DFTs in accordance with the even and odd sums we wrote down earlier. In putting the pieces together we want to leave the even indices alone, put a \( +\omega^{-m}_4 \) in front of the \( m \)th component of the first half of the \( F_2 \) of the odds, and put a \( -\omega^{-m}_4 \) in front of the \( m \)th component of the \( F_2 \) of the second half of the odds. This is done by the matrix

\[
B_4 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & \omega^{-1}_4 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -\omega^{-1}_4 \end{pmatrix}.
\]

It works like this:

\[
\]

where we (finally) used \( \omega^{-1}_2 = e^{-2\pi i/2} = -1 \). This checks with what we got before.

One way to view this procedure is as a factorization of \( F_4 \) into simpler matrices. It works like this:

\[
\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1}_4 & \omega^{-2}_4 & \omega^{-3}_4 \\ 1 & \omega^{-2}_4 & \omega^{-4}_4 & \omega^{-6}_4 \\ 1 & \omega^{-3}_4 & \omega^{-6}_4 & \omega^{-9}_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1}_4 & -1 & -\omega^{-1}_4 \\ 1 & -1 & 1 & -1 \\ 1 & -\omega^{-1}_4 & 1 & -\omega^{-1}_4 \end{pmatrix},
\]

\[
\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & \omega^{-1}_4 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -\omega^{-1}_4 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \omega^{-1}_2 \\ 0 & 1 & 0 & 1 \omega^{-1}_2 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -\omega^{-1}_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.
\]
Look at all the zeros! There are 48 entries total in the three matrices that multiply together to give $\mathcal{F}_4$, but only 20 entries are nonzero.

In the same way, the general shape of the factorization to get $\mathcal{F}_N$ via $\mathcal{F}_{N/2}$ is

$$
\mathcal{F}_N = \begin{pmatrix}
I_{N/2} & \Omega_{N/2} \\
I_{N/2} & -\Omega_{N/2}
\end{pmatrix}
\begin{pmatrix}
\mathcal{F}_{N/2} & 0 \\
0 & \mathcal{F}_{N/2}
\end{pmatrix}
\begin{pmatrix}
sort the even \\
and odd indices
\end{pmatrix}.
$$

$I_{N/2}$ is the $N/2 \times N/2$ identity matrix. $0$ is the zero matrix (of size $N/2 \times N/2$ in this case). $\Omega_{N/2}$ is the diagonal matrix with entries $1, \omega^{-1}_N, \omega^{-2}_N, \ldots, \omega^{-(N/2-1)}_N$ down the diagonal.$^{13}$ $\mathcal{F}_{N/2}$ is the DFT of half the order, and the permutation matrix puts the $N/2$ even indices first and the $N/2$ odd indices second.

Thus, the way this factorization works is:

- The inputs are $f[0], f[1], \ldots, f[N-1]$.
- The matrix on the right is a permutation matrix that puts the even indices in the first $N/2$ slots and the odd indices in the second $N/2$ slots.
  - Alternatively, think of the operation as first starting with $f[0]$ and taking every other $f[n]$ — this collects $f[0], f[2], f[4]$, and so on — and then starting back with $f[1]$ and taking every other $f[n]$ — this collects $f[1], f[3], f[5]$, and so on. As we iterate the process, this will be a more natural way of thinking about the way the first matrix chooses how to send the inputs on to the second matrix.
- The outputs of the first matrix operation are a pair of $N/2$-vectors. The matrix in the middle accepts these as inputs. It computes half-size DFTs on these half-size inputs and outputs two $N/2$-vectors, which are then passed along as inputs to the third matrix.
- The third matrix, on the left, reassembles the outputs from the half-size DFTs and outputs the $\mathcal{F}[0], \mathcal{F}[1], \ldots, \mathcal{F}[N-1]$.
  - This is similar in spirit to a step in the “merge-sort” algorithm for sorting numbers. Operations (comparisons in that case, DFTs in this case) are performed on smaller lists that are then merged to longer lists.
- The important feature, as far as counting the multiplications go, is that suddenly there are a lot of zeros in the matrices.

As to this last point, we can already assess some savings in the number of operations when the even/odd splitting is used versus the straight evaluation of the DFT from its original definition. If we compute $E = \mathcal{F}_N f$ just as a matrix product, there are $N^2$ multiplications and $N^2$ additions for a total of $2N^2$ operations. On the other hand, with the splitting, the computations in the inner block matrix of two DFTs of order $N/2$ require $2(N/2)^2 = N^2/2$ multiplications and $2(N/2)^2 = N^2/2$ additions. The sorting and recombining by the third matrix require another $N/2$ multiplications and $N$ additions — and this is linear in $N$. Thus the splitting method needs on the order of $N^2$ operations while the straight DFT needs $2N^2$.

$^{13}$The notation $\Omega_{N/2}$ isn’t the greatest — it’s written with $N/2$ because of the dimensions of the matrix, though the entries are powers of $\omega_N$. Still, it will prove useful to us later on, and it appears in the literature.
We’ve cut the work in half, pretty much, though it’s still of the same order. We’ll get back to this analysis later.

**Divide and conquer.** At this point it’s clear what we’d like to do: repeat the algorithm, each time halving the size of the DFT. The factorization from $N$ to $N/2$ is the top level:

$$
\mathcal{F}_N = \begin{pmatrix} I_{N/2} & \Omega_{N/2} \\ I_{N/2} & -\Omega_{N/2} \end{pmatrix} \begin{pmatrix} \mathcal{F}_{N/2} & 0 \\ 0 & \mathcal{F}_{N/2} \end{pmatrix} \begin{pmatrix} \text{sort the even} \\ \text{and odd indices} \end{pmatrix}.
$$

At the next level “down” we don’t do anything to the matrices on the ends, but we factor each of the two $\mathcal{F}_{N/2}$’s the same way, into a permutation matrix on the right, a block matrix of $\mathcal{F}_{N/4}$’s in the middle, and a reassembly matrix on the left. (I’ll come back to the sorting — it’s the most interesting part.) That is,

$$
\mathcal{F}_{N/2} = \begin{pmatrix} I_{N/4} & \Omega_{N/4} \\ I_{N/4} & -\Omega_{N/4} \end{pmatrix} \begin{pmatrix} \mathcal{F}_{N/4} & 0 \\ 0 & \mathcal{F}_{N/4} \end{pmatrix} \begin{pmatrix} \text{sort $N/2$-lists to} \\ \text{two $N/4$-lists} \end{pmatrix},
$$

and putting this into the top-level picture, the operations become nested (or recursive):

$$
\mathcal{F}_N = \begin{pmatrix} I_{N/2} & \Omega_{N/2} \\ I_{N/2} & -\Omega_{N/2} \end{pmatrix} \begin{pmatrix} \mathcal{F}_{N/4} & 0 \\ 0 & \mathcal{F}_{N/4} \end{pmatrix} \begin{pmatrix} \text{sort the $N/2$-even} \\ \text{and $N/2$-odd indices} \end{pmatrix}.
$$

To be able to repeat this, to keep halving the size of the DFT, we now see that we need to take $N$ to be a power of 2. The construction then continues, going down levels till we get from $\mathcal{F}_N$ to $\mathcal{F}_1$. Note that the DFT of order 1 takes a single input and returns it unchanged; i.e., it is the identity transform.

When the halving is all over, here’s what happens. The work is in the initial sorting and in the reassembling, since the final DFT in the factorization is $\mathcal{F}_1$, which leaves alone whatever it gets. Thus, reading from right to left, the initial inputs $f[0], \ldots, f[N-1]$ are first sorted and then passed back up through a number of reassembly matrices, ultimately winding up as the outputs $\mathcal{F}[0], \ldots, \mathcal{F}[N - 1]$.

It’s clear, with the abundance of zeros in the matrices, that there should be a savings in the total number of operations, though it’s not clear how much. The **entire trip**, from $f$’s to $\mathcal{F}$’s, is called the Fast Fourier Transform (FFT). It’s fast because of the reduction in the number of operations. Remember, the FFT is not a new transform, it is just computing the DFT of the initial inputs.

**7.11.2. Factoring the DFT matrix.** Rather than trying now to describe the general process in more detail, let’s look at an example more thoroughly and from the matrix point of view. One comment about the matrix factorization description versus other sorts of descriptions of the algorithm: Since the initial ideas of Cooley and Tukey there have been many other styles of FFT algorithms proposed and implemented, similar in some respects to Cooley and Tukey’s formulation and different in others. It became a mess. In a 1992 book, *Computational Frameworks*
for the Fast Fourier Transform, Charles van Loan showed how many of the ideas could be unified via a study of different matrix factorizations of the DFT. This is not the only way to organize the material, but it has been very influential.

Let’s take the case \( N = 16 \), just to live it up. Once again, the initial input is a length 16 signal (or vector) \( f \) and the final output is another 16-tuple, \( \mathbf{F} = \mathcal{F}_{16} f \).

At the top level, we can write this as

\[
\mathbf{F} = \mathcal{F}_{16} f = \begin{pmatrix} I_8 & \Omega_8 \\ I_8 & -\Omega_8 \end{pmatrix} \begin{pmatrix} \mathcal{F}_8 & 0 \\ 0 & \mathcal{F}_8 \end{pmatrix} \begin{pmatrix} \mathcal{F}_{\text{even}} \\ \mathcal{F}_{\text{odd}} \end{pmatrix} = \begin{pmatrix} I_8 & \Omega_8 \\ I_8 & -\Omega_8 \end{pmatrix} \begin{pmatrix} \mathcal{G} \\ \mathcal{H} \end{pmatrix},
\]

where \( \mathcal{G} \) and \( \mathcal{H} \) are the results of computing \( \mathcal{F}_8 \) on \( f_{\text{even}} \) and \( f_{\text{odd}} \), respectively. Write this as

\[
\mathbf{F} = B_{16} \begin{pmatrix} \mathcal{G} \\ \mathcal{H} \end{pmatrix}, \quad B_{16} = \begin{pmatrix} I_8 & \Omega_8 \\ I_8 & -\Omega_8 \end{pmatrix},
\]

where \( B \) is supposed to stand for “butterfly” — more on this, later.

But how, in the nesting of operations, did we get to \( \mathcal{G} \) and \( \mathcal{H} \)? The next level down (or in) has

\[
\mathcal{G} = \mathcal{F}_8 f_{\text{even}} = \begin{pmatrix} I_4 & \Omega_4 \\ I_4 & -\Omega_4 \end{pmatrix} \begin{pmatrix} \mathcal{G}' \\ \mathcal{H}' \end{pmatrix} = B_8 \begin{pmatrix} \mathcal{G}' \\ \mathcal{H}' \end{pmatrix},
\]

where \( \mathcal{G}' \) and \( \mathcal{H}' \) are the result of computing \( \mathcal{F}_4 \)’s on the even and odd subsets of \( f_{\text{even}} \). Got it? Likewise we write

\[
\mathcal{H} = \mathcal{F}_8 f_{\text{odd}} = \begin{pmatrix} I_4 & \Omega_4 \\ I_4 & -\Omega_4 \end{pmatrix} \begin{pmatrix} \mathcal{G}'' \\ \mathcal{H}'' \end{pmatrix} = B_8 \begin{pmatrix} \mathcal{G}'' \\ \mathcal{H}'' \end{pmatrix},
\]

where \( \mathcal{G}'' \) and \( \mathcal{H}'' \) are the result of computing \( \mathcal{F}_4 \)’s on the even and odd subsets of \( f_{\text{odd}} \).

Combining this with what we did first, we have

\[
\mathbf{F} = \mathcal{F}_{16} f = B_{16} \begin{pmatrix} B_8 & 0 \\ 0 & B_8 \end{pmatrix} \begin{pmatrix} \mathcal{G}' \\ \mathcal{H}' \\ \mathcal{G}'' \\ \mathcal{H}'' \end{pmatrix}. \]

Continue this for two more steps — it remains to find DFTs of order 4 and 2. The result then looks like

\[
\mathbf{F} = \mathcal{F}_{16} f = B_{16} \begin{pmatrix} \mathcal{G}' \\ \mathcal{H}' \\ \mathcal{G}'' \\ \mathcal{H}'' \end{pmatrix}. \]

\[
= B_{16} \begin{pmatrix} B_8 & 0 \\ 0 & B_8 \end{pmatrix} \begin{pmatrix} B_4 & 0 & 0 & 0 \\ 0 & B_4 & 0 & 0 \\ 0 & 0 & B_4 & 0 \\ 0 & 0 & 0 & B_4 \end{pmatrix} \begin{pmatrix} B_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.
\]

\[
\cdot \begin{pmatrix} 16 \times 16 \text{ permutation matrix that sorts the inputs} \end{pmatrix} f.
\]
Note that

\[ B_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \]

Each \( B_2 \) receives a pair of inputs coming from a pair of \( F_1 \)'s, and since the \( F_1 \)'s don’t do anything, each \( B_2 \) receives a pair of the original inputs \( f[m] \), but shuffled from the ordering \( f[0], f[1], \ldots, f[15] \). We’ll get back to the question of sorting the indices, but first let’s be sure that it’s worth it.

And the point of this is, again? There are lots of zeros in the factorization of the DFT. After the initial sorting of the indices (also lots of zeros in that matrix) there are 4 reassembly stages. In general, for \( N = 2^n \) there are \( n = \log_2 N \) reassembly stages after the initial sorting. The count \( \log_2 N \) for the number of reassembly stages follows in the same way as the count for the number of merge-sort steps in the sorting algorithm, but I want to be a little more precise this time.

We now consider the computational complexity of the FFT algorithm in general. Let \( C(N) \) denote the number of elementary operations involved in finding the DFT via the FFT algorithm; these include additions and multiplications. We reassemble \( F_N \) from two \( F_{N/2} \)'s by another set of elementary operations. From our earlier considerations, or from the factorization, the number of operations can easily be shown to be proportional to \( N \). Thus the basic recursion relationship is

\[ C(N) = 2C(N/2) + KN. \]

We can solve this recurrence equation as follows: let

\[ n = \log_2 N \]

and let

\[ T(n) = \frac{C(N)}{N} \]

so that

\[ C(N) = NT(n). \]

Then \( n - 1 = \log_2(N/2) \) and thus

\[ T(n - 1) = \frac{C(N/2)}{N/2} = 2 \frac{C(N/2)}{N}, \quad \text{or} \quad NT(n - 1) = 2C(N/2). \]

Substituting into the recurrence relationship for \( C \) then gives

\[ NT(n) = NT(n - 1) + KN \]

or simply

\[ T(n) = T(n - 1) + K. \]

This already implies that \( T(n) \) is linear. But \( C(1) \) is obviously 0, because there aren’t any operations needed to compute the DFT of a signal of length 1. Hence \( T(0) = C(1) = 0, \ T(1) = K, \) and in general

\[ T(n) = Kn. \]

In terms of \( C \) this says

\[ C(N) = KN \log_2 N. \]
Various implementations of the FFT try to make the constant $K$ as small as possible. The best one around now, I think, brings the number of multiplications down to $N \log_2 N$ and the number of additions down to $3\log_2 N$. Remember that this is for complex inputs. Restricting to real inputs cuts the number of operations in half.

As I pointed out when talking about the problem of sorting, when $N$ is large, the reduction in computation from $N^2$ to $N \log_2 N$ is an enormous savings. For example, take $N = 32,768 = 2^{15}$. Then $N^2 = 2^{30} = 1,073,741,824$, about a billion, while $2^{15} \log_2 2^{15} = 491,520$, a measly half a million. (Cut down to half that for real signals.) That’s a substantial reduction.

### 7.11.3. Sorting the indices. If we think of recursively factoring the inner DFT matrix, then in implementing the whole FFT the first thing that’s done is to sort and shuffle the inputs. It’s common to display a flow diagram for the FFT, and much of the pictorial splendor in many treatments of the FFT is in showing how the $f[m]$’s are shuffled and passed on from stage to stage. The flow chart of the complete FFT algorithm is called a butterfly diagram — hence the naming of the matrices $B$. You can find butterfly diagrams in any of the standard works.

The principle of sorting the inputs is as stated earlier. Start with the first input, $f[0]$, and take every other one. Then start with the second input, $f[1]$ (which was skipped over in the first pass), and take every other one. This produces $f_{\text{even}}$ and $f_{\text{odd}}$. The next sorting repeats the process for the subsequences $f_{\text{even}}$ and $f_{\text{odd}}$, and so on.

For $N = 8$ (I don’t have the stamina for $N = 16$ again) this looks like

<table>
<thead>
<tr>
<th>$f[0]$</th>
<th>$f[0]$</th>
<th>$f[0]$</th>
<th>$f[0]$</th>
</tr>
</thead>
</table>

Though there’s no more shuffling from the third to the fourth column we’ve written the last column to indicate that the inputs go in, one at a time, in that order to the waiting $B$’s.

---

The FFT algorithm with \( N = 8 \) is thus

\[
E = B_8 \begin{pmatrix} B_4 & 0 \\ 0 & B_4 \end{pmatrix} \begin{pmatrix} B_2 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 \\ 0 & 0 & B_2 & 0 \\ 0 & 0 & 0 & B_2 \end{pmatrix} \begin{pmatrix} f[0] \\ f[1] \\ f[2] \\ f[3] \\ f[4] \\ f[5] \\ f[6] \\ f[7] \end{pmatrix}.
\]

The sorting can be described in a neat way via binary numbers. Each sort puts a collection of inputs into a “top” bin or a “bottom” bin. Let’s write 0 for top and 1 for bottom (as in 0 for even and 1 for odd). Assigning digits from right to left, the least significant bit is the first sort, the next most significant bit is the second sort, and the most significant bit (for the three sorts needed when \( N = 8 \)) is the final sort. We thus augment the table, above, to (read the top/bottom descriptions right to left):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 top</td>
<td>00 top-top</td>
<td>000 top-top-top</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 top</td>
<td>00 top-top</td>
<td>10 bottom-top</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 top</td>
<td>10 bottom-top</td>
<td>01 top-bottom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 bottom</td>
<td>01 top-bottom</td>
<td>01 bottom-bottom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 bottom</td>
<td>01 bottom-bottom</td>
<td>11 bottom-bottom-bottom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 bottom</td>
<td>11 bottom-bottom-bottom</td>
<td>111 bottom-bottom-bottom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 bottom</td>
<td>11 bottom-bottom-bottom</td>
<td>111 bottom-bottom-bottom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The numbers in the final column are exactly the binary representations for 0, 4, 2, 6, 1, 5, 3, 7.

Now notice that we get from the initial natural ordering

\[
\begin{align*}
  f[0] &= 000 \\
  f[1] &= 001 \\
  f[2] &= 010 \\
  f[3] &= 011 \\
  f[4] &= 100 \\
  f[5] &= 101 \\
  f[6] &= 110 \\
  f[7] &= 111
\end{align*}
\]

by reversing the binary representation of the numbers in the first table.

This same procedure for sorting works for all $N$. In summary:

(1) Write the numbers 0 to $N - 1$ in binary (with leading 0’s so all numbers have the same length). That enumerates the slots from 0 to $N - 1$.

(2) Reverse the binary digits of each slot number. For a binary number $m$ call this reversed number $\bar{m}$.

(3) The input $f[\bar{m}]$ goes in slot $m$.

This step in the FFT algorithm is called bit reversal, for obvious reasons. In fact, people have spent plenty of time coming up with efficient bit reversal algorithms. In running an FFT routine, as in MATLAB, you don’t do the sorting, of course. The program takes care of that. If, in the likely case, you don’t happen to have $2^n$ samples in whatever data you’ve collected, then a common dodge is to add zeros to get up to the closest power of 2. This is referred to as zero padding, and FFT routines will automatically do it for you. But, like anything else, it can be dangerous if used improperly. We discuss zero padding below, and there are some problems on it.

**Bit reversal via permutation matrices.** To write down the permutation matrix that does this sorting, you perform the “every other one” algorithm to the rows (or columns) of the $N \times N$ identity matrix, reorder the rows according to that, and then repeat. Thus for $N = 8$ there are two steps:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\text{sort & rearrange rows} \rightarrow
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

sort & rearrange top & bottom halves \rightarrow
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}.

---

\(^{15}\text{See P. Rosel, Timing of some bit reversal algorithms, }\textit{Signal Processing} \textbf{18} (1989), 425–433, for a survey of 12 (!) different bit-reversal algorithms.\)
And sure enough
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
f[0] \\
f[1] \\
f[2] \\
f[3] \\
f[4] \\
f[5] \\
f[6] \\
f[7] \\
\end{bmatrix} =
\begin{bmatrix}
f[0] \\
f[4] \\
f[2] \\
f[6] \\
f[1] \\
f[5] \\
f[3] \\
f[7] \\
\end{bmatrix}.
\]

7.11.4. Zero padding. As we have seen, the FFT algorithm for computing
the DFT is set up to work with an input length that is a power of 2. While
not all implementations of the FFT require an input to be of that length, many
programs only accept inputs of certain lengths, and when this requirement is not
met, it’s common to add enough zeros to the end of the signal to bring the input
up to the length required. That’s zero padding. Let’s spell it out. Let
\[
f = (f[0], f[1], \ldots, f[N - 1])
\]
be the original input. For an integer \( M > N \), define
\[
g[n] = \begin{cases} 
  f[n], & 0 \leq n \leq N - 1, \\
  0, & N \leq n \leq M - 1.
\end{cases}
\]
Then
\[
G[m] = F_M g[m] = \sum_{n=0}^{M-1} \omega_M^{-mn} g[n] = \sum_{n=0}^{N-1} \omega_M^{-mn} f[n].
\]
Work a little bit with \( \omega_M^{-mn} \):
\[
\omega_M^{-mn} = e^{-2\pi imn/M} = e^{-2\pi imnN/MN} = e^{-2\pi in(mN/M)/N} = \omega_N^{n(mN/M)}.
\]
Thus whenever \( mN/M \) is an integer, we have
\[
G[m] = \sum_{n=0}^{N-1} \omega_N^{-n(mN/M)} f[n] = F[mN/M].
\]
We could also write this equation for \( F \) in terms of \( G \) as
\[
F[m] = G[mM/N]
\]
whenever \( mM/N \) is an integer. This is what we’re more interested in: the program
computes the zero padded transform \( G = F_M g \), and we’d like to know what the outputs \( F[m] \) of our original signal are in terms of the \( G \)’s. The answer is that
the \( m \)th component of \( F \) is the \( mM/N \)th component of \( G \) whenever \( mM/N \) is an integer.

We pursue this, starting with getting rid of the ridiculous proviso that \( mM/N \)
is an integer. We can choose \( M \), so let’s choose
\[
M = kN
\]
The FFT Algorithm

7.11. The FFT Algorithm

For some integer $k$; so $M$ is twice as large as $N$, or 3 times as large as $N$, or whatever. Then $mM/N = km$, always an integer, and

$$F[m] = G[km],$$

which is much easier to say in words:

- If $f$ is zero padded to a signal $g$ of length $M$, where $M = kN$, then the $m$th component of $F = \mathcal{F}f$ is the $km$th component of $G = \mathcal{F}g$.

**Time and frequency grids.** Remember the two grids, in time and in frequency, that arise when we use a discrete approximation of a continuous signal? Zero padding the inputs has an important consequence for the spacing of the grid points in the frequency domain. Suppose that the discrete signal $f = (f[0], f[1], \ldots, f[N-1])$ comes from sampling a continuous signal $f(t)$ at points $t_n$, so that $f[n] = f(t_n)$. Suppose also that the $N$ sample points in the time domain of $f(t)$ are spaced $\Delta t$ apart. Then the length of the interval on which $f(t)$ is defined is $N\Delta t$, and the spectrum $\mathcal{F}f(s)$ is spread out over an interval of length $1/\Delta t$. Remember, knowing $N$ and $\Delta t$ determines everything. Going from $N$ inputs to $M = kN$ inputs by padding with zeros lengthens the interval in the time domain to $M\Delta t$ but it doesn’t change the spacing of the sample points. For the sample points associated with the discrete signals $f$ and $F = \mathcal{F}_N f$ we have

$$\Delta t \Delta \nu_{\text{unpadded}} = \frac{1}{N},$$

by the reciprocity relations (see Section 7.5), and for $g$ and $G = \mathcal{F}_M g$ we have

$$\Delta t \Delta \nu_{\text{padded}} = \frac{1}{M} = \frac{1}{kN}.$$

The $\Delta t$ in both equations is the same, so

$$\frac{\Delta \nu_{\text{padded}}}{\Delta \nu_{\text{unpadded}}} = \frac{1}{k} \quad \text{or} \quad \Delta \nu_{\text{padded}} = \frac{1}{k} \Delta \nu_{\text{unpadded}},$$

that is, the spacing of the sample points in the frequency domain for the padded sequence has decreased by the factor $1/k$. At the same time, the total extent of the grid in the frequency domain has not changed because it is $1/\Delta t$ and $\Delta t$ has not changed. What this means is:

- Zero padding in the time domain refines the grid in the frequency domain.

There’s a warning that goes along with this. Using zero padding to refine the grid in the frequency domain is a valid thing to do only if the original continuous signal $f$ is already known to be zero outside of the original interval. If not, then you’re killing off real data by filling $f$ out with zeros. See the book by Briggs and Henson for this and for other important practical concerns.
7.1. Discrete triangle function

We can obtain a discrete triangle function $\Delta[n]$ by sampling the usual triangle function $\Lambda(t)$. We need to specify the length, $N$, of $\Delta[n]$ and the sampling rate $m$. Define

$$\Delta[n] = \Lambda(n/m), \quad n = -N, \ldots, N.$$ 

For the plots below, what are the values of $m$ and $N$?
7.2. Convolving discrete rectangle functions

We’re familiar with the result $\Pi \ast \Pi = \Lambda$ for the continuous-time rectangle and triangle functions. Use software to try this out with discrete signals. Take, e.g., $\Pi = \sum_{n=-4}^{4} \delta_n$ and plot $\Pi \ast \Pi$. Verify algebraically what you see. Now use $\Pi' = (1/2)(\delta_{-4} + \delta_4) + \sum_{n=-3}^{3} \delta_n$ (i.e., making the value $1/2$ at the endpoints), and plot $\Pi' \ast \Pi'$. How does this compare to your first plot? Try some other experiments.

7.3. Practice with the DFT

Find the discrete Fourier transform of each of the sequences below. (You should not have to grind through all the calculations!) Note: You might check your results in MATLAB, or another package, but the point of the problem is to get some hands-on experience with calculating DFTs.

(a) $f = (1, 1, 1, 1, 1, 1, 1, 1)$.
(b) $\tilde{f} = (1, 1, 1, 1, 0, 0, 0, 0)$.
(c) $\tilde{f} = (0, 0, 1, 1, 1, 1, 0, 0)$.
(d) $\tilde{f} = (1, 1, -1, 1, -1, 1, -1, 1)$.
(e) $\tilde{f} = (0, 0, 1, 0, 0, 0, 1, 0)$.

7.4. DFT of real-valued signals

Let $f$ and $g$ be real discrete signals of length $N$ with DFTs $F$ and $G$, respectively. Consider the possibility of computing both of these DFTs simultaneously by using an $N$-point DFT on the complex signal $h[n] = f[n] + ig[n]$. If $H = \mathcal{F}h$, find separate expressions (if possible) for $F$ and $G$ in terms of $H$. If it is not possible to separate $F$ and $G$ from $H$, explain.

7.5. A true story

Professor Osgood and a graduate student were working on a discrete form of the sampling theorem. This included looking at the DFT of the discrete rect function

$$f[n] = \begin{cases} 1, & |n| \leq \frac{N}{6}, \\ 0, & -\frac{N}{2} + 1 \leq n < -\frac{N}{6}, \quad \frac{N}{6} < n \leq \frac{N}{2}. \end{cases}$$

The grad student, ever eager, said, “Let me work this out.” A short time later the student came back saying, “I took a particular value of $N$ and I plotted the DFT using MATLAB (their FFT routine). Here are plots of the real part and the imaginary part.”
(a) Produce these figures.
Professor Osgood said, “That can’t be correct.”
(b) Is Professor Osgood right to object? If so, state what the basis of his objection is and produce the correct plot. If not, explain why the student is correct.\(^\text{16}\)

7.6. DFT variety

Consider two sequences \( f[n] \) and \( g[n] \) of length \( N = 6 \). These two sequences are related by the DFT relationship \( F[m] = (-1)^m G[m] \). The sequence \( g[n] \) is shown below.

(a) Express the sequence \( f[n] \) in terms of \( g[n] \). Also write out the sequence \( f[n] \) as an \( N \)-tuple \((f[0], \ldots, f[5])\).

\( \text{Hint: Write } (-1)^m \text{ as a complex exponential.} \)

\(^{16}\)Readers of this book will know that Professor Osgood has (on occasion) been wrong.
(b) Evaluate the summation $\sum_{m=0}^{5} |F[m]|^2$.
(c) Let $h = f \ast g$. Evaluate $h[4]$, the fifth element of this convolution.

7.7. Approximation of continuous Fourier transform

We can view the DFT as a discrete approximation to the Fourier transform of a continuous-time signal, and we know how to take the DFT in MATLAB (fft, fftshift, ifftshift, ...) by sampling the continuous-time signal. In this problem, we want to approximate the Fourier transform of the timelimited signal $f(t)$ shown below.

We sample $f(t)$ in three different versions, sample (a), sample (b), and sample (c):

Don’t worry — you don’t need to evaluate the Fourier transform of $f(t)$ or write a MATLAB code. Possible results, obtained using MATLAB, are shown below. What you need to do is match (a), (b), and (c) to the approximation of the Fourier transform in (A) through (F). Hint: Start with finding a plot that matches sample (a) first. Remember the relationship $2BL = N$ between the bandwidth and the limitation in time.
7.8. Sampling a periodic signal: Another interpretation of the DFT

Let $x(t)$ be a periodic signal of period 1. Let $N > 1$ and sample $x(t)$ by multiplying by a finite $\delta$-function of spacing $1/N$, forming

$$g(t) = x(t) \sum_{k=0}^{N-1} \delta(t - k/N).$$

Let $c_n$ be the $n$th Fourier coefficient,

$$c_n = \int_{-1/2}^{1/2} e^{-2\pi i nt} g(t) \, dt = \int_{0}^{1} e^{-2\pi i nt} g(t) \, dt.$$

(The integral from $-1/2$ to $1/2$ is more convenient to use for the calculations in this problem.)

Finally, let $f$ be the discrete signal with values at the sample points,

$$f[n] = x(n/N), \quad n = 0, 1, \ldots, N - 1.$$

What is the relationship between $c_n$ and the DFT of $f[n]$?

---

**Diagram Description:**

- **A**: A graph showing a periodic signal with a frequency range from 0 to 3 Hz.
- **B**: A graph of a sampled signal with a frequency range from 0 to 3 Hz.
- **C**: A graph showing the signal after multiplication with the $\delta$-function.
- **D**: A graph of the sampled signal with a frequency range from 0 to 5 Hz.
- **E**: A graph showing the signal after multiplication with the $\delta$-function.
- **F**: A graph of the sampled signal with a frequency range from 0 to 5 Hz.
Problems and Further Results

7.9. Some particular cases

Let \( \mathbf{x} = (p, q, r, s, t, u) \) be a vector of length 6 and assume that \( \mathbf{x} \) has the 6-point DFT \( \mathcal{F}\mathbf{x} = (P, Q, R, S, T, U) \), where \( p, \ldots, u, P, \ldots, U \) are complex scalars. For each signal (i)–(v), indicate its corresponding discrete Fourier transform from parts (A)–(J). Provide a brief explanation for each answer.

Note that the 3-point DFT is computed for the signal of length 3 in part (i) and the 2-point DFT is computed for the signal of length 2 in part (ii).

(i) \((p + s, q + t, r + u)\).
(ii) \((p, s)\).
(iii) \((p, −q, r, −s, t, −u)\).
(iv) \((s, t, u, p, q, r)\).
(v) \((p, q, r, s, t, u)\) \(\ast (1, \cos(2\pi/3), \cos(4\pi/3), \cos(6\pi/3), \cos(8\pi/3), \cos(10\pi/3))\).

(A) \(\frac{1}{3}(P + R + T, Q + S + U)\).
(B) \((P, S)\).
(C) \(\frac{1}{2}(P + S, Q + T, R + U)\).
(D) \((S, T, U, P, Q, R)\).
(E) \((P, R, T)\).
(F) \((T, U, P, Q, R, S)\).
(G) \((P, −Q, R, −S, T, −U)\).
(H) \(3(0, 0, R, 0, T, 0)\).
(I) \((P + S, 0, Q + T, 0, R + U, 0)\).

7.10. Duality relation in matrix form

One of the duality relations for the DFT is \(\mathcal{F}\mathcal{F}f = Nf^\top\). What is the matrix form of this; i.e., what is the matrix \(\mathcal{F}\mathcal{F}\)?

\(^{17}\)From D. Kammler.
7.11. **DFT properties**

(a) Prove the shift theorem for the discrete Fourier transform:

\[ \mathcal{F}(\tau_pf) = \omega^{-p} \mathcal{F}f, \]

where

\[ \tau_pf[n] = f[n - p]. \]

(b) **Replication.** Suppose that the signal \( f = (f[0], f[1], \ldots, f[N - 1]) \) has discrete Fourier transform \( \mathcal{F} \). We create a new signal \( g[n], n = 0, 1, \ldots, 2N - 1 \), with *twice* the number of points defined by

\[
g[n] = \begin{cases} f[n], & n = 0, 1, \ldots, N - 1, \\ \frac{f[n - N]}{2}, & n = N, N + 1, \ldots, 2N - 1. \end{cases}
\]

Find the DFT of \( g \) in terms of \( \mathcal{F} \).

(c) **Simple upsampling.** Again suppose that the signal \( f \), of size \( N \), has discrete Fourier transform \( \mathcal{F} \). We create a new signal \( h \) of size \( 2N, n = 0, 1, \ldots, 2N - 1 \), with twice the number of points, differently, by inserting 0’s among the values \( f[n] \); i.e.,

\[
h[n] = \begin{cases} f[n/2], & n \text{ even}, \\ 0, & n \text{ odd}. \end{cases}
\]

Find the DFT of \( h \) in terms of \( \mathcal{F} \).

(d) **Simple downsampling.** With \( f \) as above, we create another new signal \( g \), with *half* the number of points, \( n = 0, 1, \ldots, N/2 - 1 \) (assume that \( N \) is even) by keeping only the values of \( f[n] \) at even indices; i.e.,

\[
g[n] = f[2n].
\]

Find the DFT of \( g \) in terms of \( \mathcal{F} \).

7.12. **The DFT, upsampling, and linear interpolation**

(a) Let \( y \) be the discrete signal, periodic of order \( M \),

\[
y = \left(1, \frac{1}{2}, 0, \ldots, 0, \frac{1}{2}\right).
\]

Show that its DFT is

\[
\mathcal{Y}[m] = 1 + \cos(2\pi m/M).
\]

(b) Let \( f = (f[0], f[1], \ldots, f[N - 1]) \) be a discrete signal and let \( \mathcal{F} = (\mathcal{F}[0], \mathcal{F}[1], \ldots, \mathcal{F}[N-1]) \) be its DFT. Recall that the upsampled version of \( f \) is the signal \( h \) of order \( 2N \) obtained by inserting zeros between the values of \( f \); i.e.,

\[
h = (f[0], 0, f[1], 0, f[2], \ldots, 0, f[N - 1], 0).
\]

Show that \( \tilde{f} = h \ast y \) is the linearly interpolated version of \( f \):

\[
\left(f[0], \frac{f[0] + f[1]}{2}, f[1], \frac{f[1] + f[2]}{2}, f[2], \ldots, f[N - 1], \frac{f[N - 1] + f[0]}{2}\right).
\]
Problems and Further Results

473

**Hint:** Here we take $M = 2N$ for the period of $y$. Note that

$$y = \delta_0 + \frac{1}{2}\delta_1 + \frac{1}{2}\delta_{2N-1}$$

and remember the effect of convolving with a shifted discrete $\delta$. Line up the $2N$-tuples.

(c) In Problem 7.11 you showed that the DFT of $h$ is a replicated form of $F$.

$$H[m] = F[m], \quad m = 0, 1, \ldots, 2N - 1,$$

$$H = \left( F[0], F[1], \ldots, F[N - 1], \frac{F}{E}\right).$$

Assuming this, find the DFT of $\tilde{f}$.

### 7.13. Upsampling by a factor $L$

This is the first of several problems generalizing and working with upsampling and downsampling from Problem 7.11. Upsampling first. Suppose $f$ is a signal of length $N$ and let $L$ be a positive integer. From $f$ we manufacture a signal $g$ of length $LN$ by inserting $L - 1$ zeros after each $f[m]$, $m = 0, \ldots, N - 1$. For example, if $N = 4$ and $L = 5$, then

$$g = (f[0], 0, 0, 0, 0, f[1], 0, 0, 0, 0, f[2], 0, 0, 0, 0, f[3], 0, 0, 0, 0),$$

a signal of length $20 = 4 \cdot 5$. There are 4 blocks of length 5 of the form $f[k], 0, 0, 0, 0$ where $k$ goes from 0 to 3.

More compactly, $g$ is defined by

$$g[m] = \begin{cases} f[m/L], & \text{if } L \text{ divides } m, \\ 0, & \text{otherwise}, \end{cases}$$

for $m = 0, 1, \ldots, LN - 1$. We say that $g$ is obtained from $f$ by upsampling by a factor $L$.

Show that $Fg$ is a concatenation of copies of $Ff$.

This more general upsampling is usually accompanied by some kind of interpolation (e.g., as in Problem 7.12) to replace the inserted zeros by values that more reasonably interpolate the given sample values. A simple scheme is to convolve the upsampled signal with a dinc whose DFT (a $\Pi$) is wide enough to isolate one copy of $Ff$; the result is a digital low-pass filter. There’s a lot that can be done, and thus one enters the world of digital filters, filter banks, etc.

### 7.14. Downsampling by a factor $P$

Here’s a more general take on downsampling. As always, let $f$ be a signal of length $N$, and suppose that $P$ divides $N$, say $N = MP$ for a positive integer $M$. To downsample $f$ by the factor $P$ is to form the signal $g$ of length $N/P = M$ by taking every $P$th sample of $f$. Thus

$$g[m] = f[mP], \quad m = 0, 1, 2, \ldots, M - 1.$$
For example, if $N = 20$ and $P = 4$, then $M = 5$ and
\[ g = (f[0], f[4], f[8], f[12], f[16]). \]

(a) Express $\mathcal{F}g$ as a convolution of a discrete III with $\mathcal{F}f$ (in general, not just for the example). Write the expression for $\mathcal{F}g[m]$ at an index $m$, and show that
\[ \mathcal{F}g[0] = \frac{1}{P} \sum_{k=0}^{P-1} \mathcal{F}f\left[\frac{kN}{P}\right]. \]

(b) What conditions must the spectrum $\mathcal{F}f$ satisfy to avoid overlaps in the spectrum $\mathcal{F}g$ due to the periodizing that you observed in (a)? How could you achieve this prior to downsampling?

Upsampling (together with interpolation) and downsampling are often combined to change the original sampling rate by a fractional amount. This area of digital signal processing is called, naturally, sample-rate conversion. Let me quote Wikipedia:

Sample-rate conversion is the process of changing the sampling rate of a discrete signal to obtain a new discrete representation of the underlying continuous signal. Application areas include image scaling and audio/visual systems, where different sampling rates may be used for engineering, economic, or historical reasons.

For example, Compact Disc Digital Audio and Digital Audio Tape systems use different sampling rates, and American television, European television, and movies all use different frame rates. Sample-rate conversion prevents changes in speed and pitch that would otherwise occur when transferring recorded material between such systems.

7.15. Time domain multiplexing

Let $f$, $g$, $h$ be length $N$ signals with discrete Fourier transforms $\mathcal{F}$, $G$, $H$, respectively.

(a) We combine $f$ and $g$ to form the $2N$-dimensional signal $w$,
\[ w[n] = \begin{cases} f[n/2], & \text{for } n \text{ even}, \\ g[(n-1)/2], & \text{for } n \text{ odd}, \end{cases} \]

where $0 \leq n \leq 2N-1$. Find the discrete Fourier transform $W = \mathcal{F}w$ in terms of $F$ and $G$.

Hint: Consider constructing $w$ from upsampled versions of $f$ and $g$.

(b) How does your answer to part (a) simplify if $g = 0$? Explain why this makes sense given what you know about replication and upsampling.

(c) Suppose we defined $w[n]$ for $0 < n < 3N - 1$ by
\[ w[n] = \begin{cases} f[n/3], & \text{for } n = 0 \mod 3, \\ g[(n-1)/3], & \text{for } n = 1 \mod 3, \\ h[(n-2)/3], & \text{for } n = 2 \mod 3. \end{cases} \]

Describe how you expect $W$ to depend on $F$, $G$, and $H$. You do not need to do any calculations for this part; you may generalize from part (a). To keep everything simple and concrete, set $N = 4$ for this part.
7.16. Summation rule of the DFT
Suppose \( f \) is an \( mN \)-periodic function and \( g \) is defined as
\[
g[n] = \sum_{l=0}^{m-1} f[n + lN], \quad n = 0, 1, \ldots, N - 1.
\]
(a) Show that \( g[n] \) is periodic of period \( N \).
(b) Suppose \( f \) has the DFT \( \mathcal{F}f \). What is \( \mathcal{F}g \)?
(c) Suppose \( f = (a, b, c, d, e, f) \) has the DFT \( \mathcal{F}f = (A, B, C, D, E, F) \). What is the DFT of \( g = (a + c + e, b + d + f) \)?

7.17. Discrete correlation
Let \( f \) and \( g \) be discrete, real signals, each periodic of period \( N \). Define their correlation by the formula
\[
(f \star g)[m] = \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f[n]g[n + m].
\]
(a) Show that
\[
\mathcal{F}(f \star g) = \mathcal{F}f \mathcal{F}g.
\]
Use the reversed signal to write the correlation in terms of convolution.
(b) Give an upper bound for \( (f \star g)[0] \).

7.18. DFT’s frequency response and spectral leakage
Compute the DFT of the discrete signal
\[
f[k] = e^{2\pi ik/N};
\]
i.e., find a closed form expression for \( \mathcal{F}f \). This is the frequency response of the DFT.
Qualitatively, how is the spectrum different when \( x \) is an integer and when \( x \) is not an integer? Plot the magnitude of the frequency response for \( x = 2.5 \) and \( N = 8 \). Does the plot agree with what you expect?

7.19. Leakage et al., again\(^\text{18} \)
Consider the periodic signal \( f(t) = \cos(2\pi \frac{5}{32} t) \). The signal and its samples are shown below.

\(^{18}\)From J. Gill.
7. Discrete Fourier Transform

(a) Sketch the continuous Fourier transform of \( f(t) \).
(b) Sketch the DFT of the 32-point sample sequence \( \{ f(0), f(1), \ldots, f(31) \} \).
(c) Sketch the DFT of the 16-point sample sequence \( \{ f(0), f(2), \ldots, f(30) \} \).
(d) The DFTs sketched in parts (b) and (c) should be consistent with the continuous Fourier transform of \( f(t) \). Explain why this is expected. In parts (a) and (b) the sampling rates (1 and 1/2, respectively) are larger than the Nyquist rate, 5/16. This means that there will be no aliasing in the sampled signal, so the DFT of the sampled signal will look like the frequency response of \( f \).
(e) The DFT of the first 16 samples \( \{ f(0), f(1), \ldots, f(15) \} \) has the following plot (the DFT is purely imaginary).

This DFT is qualitatively different from the DFTs of parts (b) and (c). Explain why the above DFT does not agree with the continuous Fourier transform of \( f(t) \).

7.20. A discrete rectangle function for an interval

Here’s a variation of the discrete rectangle function and its DFT: again we want a signal that’s 1 over a block of consecutive indices (which we think of as a discrete version of an interval) and 0 off that block. Take a set \( \mathcal{I} \) of at most \( N \) consecutive integers and let

\[
\Pi_{\mathcal{I}} = \sum_{k \in \mathcal{I}} \delta_k .
\]

Show that

\[
\mathcal{F} \Pi_{\mathcal{I}}[m] = \begin{cases} 
|\mathcal{I}|, & m \equiv 0 \mod N, \\
\omega^{-m \mathcal{I}_{\text{mid}}} \frac{\sin\left(\frac{\pi m |\mathcal{I}|}{N}\right)}{\sin\left(\frac{\pi}{N}\right)}, & m \not\equiv 0 \mod N,
\end{cases}
\]

where \( |\mathcal{I}| \) is the number of elements in \( \mathcal{I} \) and

\[
\mathcal{I}_{\text{mid}} = \frac{\alpha + \beta}{2} = \min \mathcal{I} + \max \mathcal{I}/2 .
\]

This is supposed to suggest the midpoint of \( \mathcal{I} \), except that \( \mathcal{I}_{\text{mid}} \) need not be an integer. Nevertheless, \( \mathcal{I}_{\text{mid}} \) is a perfectly fine number and makes sense in the formula.

7.21. A discrete uncertainty principle\(^{19}\)

In the chapter on convolution, we proved the uncertainty principle for continuous-time signals, the main idea being that time width and frequency width are inversely related. A similar relationship holds for discrete signals,

\(^{19}\)From A. Siripuram.
as we will see in this problem. Let \( f \) be an \( N \)-length vector and let \( F = \mathcal{F}f \) be its DFT. Let \( \phi(f) \) indicate the number of nonzero coordinates in \( f \), and similarly let \( \phi(\mathcal{F}f) \) be the number of nonzero coordinates in \( F = \mathcal{F}f \). Then the discrete uncertainty principle states that

\[
\phi(f)\phi(\mathcal{F}f) \geq N.
\]

In other words, the time width (\( \phi(f) \)) and frequency width (\( \phi(\mathcal{F}f) \)) are inversely related.

Below is an outline of the proof of this uncertainty principle. Complete the proof by briefly justifying the statements (a)–(f) in the proof.

\textit{Proof.} Let the vector \( s \) denote the sign of \( f \):

\[
s[n] := \begin{cases} 
1 & \text{if } f[n] > 0, \\
-1 & \text{if } f[n] < 0, \\
0 & \text{if } f[n] = 0,
\end{cases}
\]

and let \((f, s)\) denote the dot product of the vectors \( f \) and \( s \). Assume that \( f \neq 0 \). Defining \( s[n] \) gives us the advantage of expressing \( |f[n]| \) as \( |f[n]| = f[n]s[n] \).

Now

\[
\begin{align*}
\max_m |F[m]| \quad & \overset{(a)}{=} \sum_{n=0}^{N-1} |f[n]| \\
& \overset{(b)}{=} (f, s) \\
& \overset{(c)}{\leq} (f, f)^{1/2}(s, s)^{1/2} \\
& \overset{(d)}{=} \frac{1}{\sqrt{N}}(F, F)^{1/2}(s, s)^{1/2} \\
& \overset{(e)}{=} \frac{1}{\sqrt{N}}(F, F)^{1/2}\phi(f)^{1/2} \\
& \overset{(f)}{\leq} \frac{\max_m |F[m]|}{\sqrt{N}}\phi(F)^{1/2}\phi(f)^{1/2}.
\end{align*}
\]

Canceling out \( \max_m |F[m]| \) on both sides, we are left with the desired inequality

\[
\phi(f)\phi(F) \geq N.
\]

7.22. \textit{Radix-3 FFT}

Let \( f[n] \) be a 9-point sequence. Show that the order-9 DFT, \( F[m] = (\mathcal{F}f)[m] \), can be expressed in terms of the order-3 DFTs of the following 3-point sequences:

\[
\begin{align*}
\mathcal{F}A &= (f[0], f[3], f[6]), \\
\mathcal{F}B &= (f[1], f[4], f[7]), \\
\mathcal{F}C &= (f[2], f[5], f[8]),
\end{align*}
\]
as

\[ F[m] = E_A[m] + \omega^{-m}E_B[m] + \omega^{-2m}E_C[m], \]

where \( \omega = e^{2\pi i/9} \) and \( m = 0, 1, \ldots, 8 \). (To make sense of this expression you
need to use the fact that \( E_A[m], E_B[m], \) and \( E_C[m] \) are periodic of period 3.)

This result is the first step in establishing a “radix-3” FFT algorithm.

7.23. A version of the stretch theorem

Does the stretch theorem have an analog for periodic signals \( f \) of length
\( N \)? Since the signal is periodic, we can certainly define \( \sigma_af[n] = f[an] \) for any
integer \( a \) (and we assume \( a \neq 0 \)). But for the Fourier transform \( \mathcal{F}(f[an]) \) we
might think the spectrum scales like \( m/a, \) and \( m/a \) is an integer only when
\( a \) divides \( m \). Any hope? Yes, but we need to know a few more facts about
congruences mod \( N \).

Instead of dividing, as in forming \( m/a, \) we think in terms of a multiplica-
tive inverse, as in forming \( a^{-1}m \) where \( a^{-1} \) is an element in \( \mathbb{Z}_N \) for which
\( a^{-1}a \equiv 1 \pmod{N} \). When does a number \( a \in \mathbb{Z}_N \) have a multiplicative
inverse? It does if \( a \) and \( N \) have no common factors, i.e., if their greatest
common divisor is 1. (Check some examples.) The numbers in \( \mathbb{Z}_N \) that have
a multiplicative inverse are called the units of \( \mathbb{Z}_N \), and the set of all units in
\( \mathbb{Z}_N \) is often denoted by \( \mathbb{Z}_N^* \). For example, \( \mathbb{Z}_7^* = \{1, 3, 5\} \), \( \mathbb{Z}_7^{10} = \{1, 3, 7, 9\} \), and
if \( p \) is any prime, then \( \mathbb{Z}_p^* = \{1, 2, \ldots, p - 1\} \). The second thing we need to
know is that if \( a \) is any unit in \( \mathbb{Z}_N \), then for \( n = 0, 1, \ldots, N - 1 \) the numbers
\( an \) range over all of \( \mathbb{Z}_N \); they may not go in order, but all the numbers in \( \mathbb{Z}_N \)
are hit. You can establish these results yourself or look them up.

Armed with these facts, derive the following discrete version of the stretch
theorem: Let \( a \) be a unit in \( \mathbb{Z}_N \). If \( f[n] \Rightarrow \mathcal{F}[n] \), then \( f[an] \Rightarrow \mathcal{F}[a^{-1}n] \). In
terms of \( \sigma_a \) we can write this without the variable as \( \mathcal{F}(\sigma_a f) = \sigma_{a^{-1}}(\mathcal{F}f) \).
Notice that there’s no \( a^{-1} \) out front on the right as there is in the continuous
stretch theorem.

7.24. What is \( 1 \ast \mathcal{F} \)? What is \( 1 \ast 1^? \), \( 1 \ast a \), where \( a = (a, a, \ldots, a) \)?

7.25. In the chapter we saw that \( \delta_a \ast \delta_b = \delta_{a+b} \). This useful identity is actually a
defining property of convolution if we throw in the algebraic properties that
make convolution like multiplication. To make this precise, let \( \mathcal{D}_N \) be the
collection of discrete signals of period \( N \). This is a (complex) vector space,
because if \( f, g \in \mathcal{D}_N \), so is \( f + g \), and likewise if \( \alpha \) is a complex number, then
\( \alpha f \in \mathcal{D}_N \) whenever \( f \in \mathcal{D}_N \). Now show the following:
There is exactly one operation \( C(f, g), \) operating on pairs of signals in \( \mathcal{D}_N \),
and producing an element of \( \mathcal{D}_N \), with the following properties:

(i) \( C \) is bilinear, meaning

\[
C(f + g, h) = C(f, h) + C(g, h), \quad C(f, g + h) = C(f, g) + C(f, h),
\]

\[
C(\alpha f, g) = \alpha C(f, g), \quad C(f, \alpha g) = \alpha C(f, g).
\]

(ii) \( C(\delta_a, \delta_b) = \delta_{a+b} \).
Problems and Further Results

7.26. **Orthonormal discrete sines**

Consider the discrete functions \( f_n, -\infty < n < \infty \), defined by

\[
f_n[m] = \text{sinc}(m - n), \quad -\infty < m < \infty.
\]

Show that they form an orthonormal family, where we use the inner product of two infinite sequences \( a_n \) and \( b_n \) defined by

\[
\sum_{n=-\infty}^{\infty} a_n \overline{b_n}.
\]

For this, use the results in Problem 6.2.

7.27. **The discrete time Fourier transform**

In Fourier analysis, different transforms are used to go between the time and frequency domains depending on whether the signal is continuous or discrete in each domain. For example, the DFT is used to transform signals from discrete time to discrete frequency while the Fourier transform transforms a continuous-time signal into a continuous-frequency signal. The **discrete time Fourier transform**, or DTFT, takes a discrete-time signal to a continuous-frequency signal. The DTFT of a signal \( f[n] \) is defined as

\[
F(s) = \mathcal{D}f(s) = \sum_{n=-\infty}^{\infty} f[n] e^{-2\pi isn}, \quad s \in \mathbb{R}.
\]

(a) What is the DTFT of \( \overline{f}[n] \), the complex conjugate of \( f[n] \), in terms of \( F(s) \)?

(b) Find a formula for the inverse DTFT transform \( \mathcal{D}^{-1} \), a transform such that

\[
\mathcal{D}^{-1}(\mathcal{D}f)[n] = f[n].
\]

_Hint:_ \( F(s) = \mathcal{D}f(s) \) is periodic of period 1. Think Fourier coefficients, except with a plus sign in the exponential.

(c) What is \( \mathcal{D}\delta_0 \), the DTFT of the discrete \( \delta \) centered at 0? (Note: This is not the periodic discrete \( \delta \) used in the DFT. This \( \delta_0 \) is just 1 at 0 and 0 elsewhere.)

(d) What is \( \mathcal{D}1 \), the DTFT of the discrete constant function 1? Your answer should not involve any complex exponentials.

(e) Let \( f[n] \) be a discrete-time signal of length \( N \); i.e., \( f[n] \) is identically zero if \( n \) is outside \([0:N-1]\). What is the relationship between the DTFT \( \mathcal{D}f(s) \) and the DFT \( \mathcal{F}f[n] \) of \( f[n] \)?
7.28. Convolution theorem for the discrete time Fourier transform

Let \( f[n] \) and \( g[n] \) be discrete signals defined for all integers \( n = 0, \pm 1, \pm 2, \ldots \) but not assumed to be periodic. Their convolution is the discrete signal

\[
(f * g)[n] = \sum_{m=-\infty}^{\infty} f[n-m]g[m],
\]

also defined for all integers \( n = 0, \pm 1, \pm 2, \ldots \).

Show that the DTFT from the previous problem satisfies

\[
\mathcal{D}(f * g)(s) = \mathcal{D}f(s) \mathcal{D}g(s).
\]

7.29. Solving a quadratic, the hard way

The quadratic formula tells us how to find the roots \( x_0 \) and \( x_1 \) of a quadratic function \( x^2 + bx + c \):

\[
x^2 + bx + c = (x - x_0)(x - x_1) \quad \Rightarrow \quad \{x_0, x_1\} = \frac{1}{2} \left(-b \pm \sqrt{b^2 - 4c}\right).
\]

In this problem, we’ll rederive this formula using the DFT. Throughout parts (a)–(c), assume that the vector \( \mathbf{x} = (x_0, x_1) \) has DFT \( \mathbf{X} = (X_0, X_1) \).

(a) Express \( x_0 \) and \( x_1 \) in terms of \( X_0 \) and \( X_1 \).

(b) Express \( b \) and \( c \) in terms of \( X_0 \) and \( X_1 \).

(c) Combine parts (a) and (b) to express \( x_0 \) and \( x_1 \) in terms of \( b \) and \( c \).

7.30. Solving a cubic, the hard way

In this problem we will find the complex roots of the cubic

\[
f(x) = x^3 - 3\alpha\beta x - (\alpha^3 + \beta^3) = 0.
\]

It can be shown (not required for this problem) that any cubic can be written in this form.

(a) Find a \( 3 \times 3 \) circulant matrix \( A \) such that \( \det(xI - A) = f(x) \):

\[
A = \begin{pmatrix}
a_1 & a_3 & a_2 \\
a_2 & a_1 & a_3 \\
a_3 & a_2 & a_1
\end{pmatrix}, \quad \det(xI - A) = f(x).
\]

**Hint:** Pick \( a_1 = 0 \) and compute \( \det(xI - A) \). You should be able to read off \( a_2, a_3 \) by inspection.

(b) Use (a) to find all the roots of \( f(x) \). Recall that the roots of \( \det(xI - A) \) are the eigenvalues of \( A \).

7.31. The DFT and difference operators

There isn’t a derivative theorem for the DFT because there isn’t a derivative of a discrete signal. Instead one forms differences, and there are some analogies to the continuous case.

---

\[20\]From Raj Bhatnagar.

\[21\]From Aditya Siripuram.
Let \( f \) be a (periodic) discrete signal of length \( N \). Define the backward difference \( \nabla f \) to be the discrete signal
\[
\nabla f[n] = f[n] - f[n-1], \quad n = 0, 1, \ldots, N-1.
\]
Similarly, the forward difference operator is
\[
\Delta f[n] = f[n+1] - f[n], \quad n = 0, 1, \ldots, N-1.
\]
In both cases note how periodicity enters; e.g., \( \nabla f[0] = f[0] - f[-1] = f[0] - f[N-1] \).
(a) For practice, using the definition of the DFT show that
\[
\mathcal{F}(\nabla f)[n] = (1 - e^{-2\pi in/N})\mathcal{F}f[n], \quad n = 0, 1, 2, \ldots, N-1.
\]
(b) We can also get this result another way, and an analogous result for \( \Delta f \), via convolution. Express \( \nabla f \) and \( \Delta f \) in terms of convolution, and show that
\[
\mathcal{F}(\Delta f)[n] = (e^{2\pi in/N} - 1)\mathcal{F}f[n].
\]
Is it true that \( \nabla \Delta f = \Delta \nabla f \)?
Finally, express \( \nabla f \) and \( \Delta f \) in terms of each other via convolution.
One can define higher-order differences (like higher-order derivatives) recursively. For example,
\[
\begin{align*}
\nabla^{(0)} f[n] &= f[n] \quad \text{(just the original signal — no differences)}, \\
\nabla^{(1)} f[n] &= f[n] - f[n-1] \quad \text{(the first difference, as above)}, \\
\nabla^{(2)} f[n] &= \nabla(\nabla f)[n] = \nabla f[n] - \nabla f[n-1] = f[n] - 2f[n-1] + f[n-2], \\
\nabla^{(3)} f[n] &= \nabla(\nabla^{(2)} f)[n] = \nabla^{(2)} f[n] - \nabla^{(2)} f[n-1] = \cdots,
\end{align*}
\]
and so on. Similarly for \( \Delta^{(m)} f \). (Again, these are all defined because of periodicity.)
(c) Express \( \nabla^{(m)} f \) and \( \Delta^{(m)} f \) in terms of convolution and find their DFTs in terms of \( \mathcal{F}f \).
(d) Clearly, if \( f \) is a constant signal, then all differences are identically 0; i.e.,
\[
\nabla^{(m)} f[n] = 0 \quad \text{and} \quad \Delta^{(m)} f[n] = 0 \quad \text{for all} \ n \quad \text{and for any} \ m.
\]
Conversely, suppose there exists an \( m_0 \) such that
\[
\nabla^{(m_0)} f[n] = 0, \quad \text{or} \quad \Delta^{(m_0)} f[n] = 0, \quad n = 0, 1, 2, \ldots, N-1.
\]
Show that \( f \) is a constant signal, and say what the constant is. (Hint: The DFT of \( \nabla^{(m_0)} f \) or \( \Delta^{(m_0)} f \) is zero. Use your result from part (b).)

7.32. A discrete form of Wirtinger’s inequality

Glance back at Problem 1.26 on Wirtinger’s inequality; it’s an inequality between the integrals of a periodic function \( f(t) \) and its derivative \( f'(t) \).
There’s a discrete version of the inequality involving difference operators; see the previous problem.
Let $f[n]$, $n = 0, \ldots, N - 1$, be a discrete, periodic function of length $N$ with
\[\sum_{k=0}^{N-1} f[k] = 0.\]

In terms of the DFT this condition means
\[\mathcal{F}f[0] = \sum_{k=0}^{N-1} f[k] = 0.\]

(You’ll have to see how this comes in.)

The problem is to establish the inequality
\[\sum_{k=0}^{N-1} |f[k]|^2 \leq \frac{1}{4 \sin^2(\pi/N)} \sum_{k=0}^{N-1} |\Delta f[k]|^2.\]

This sure looks like some kind of application of Parseval’s identity. It is, twice in fact — with some work.

Start with $\|\mathcal{F}(\Delta f)\|^2 = N \|\Delta f\|^2$, from Parseval, and show $|\mathcal{F}(\Delta f)[k]|^2 = 4 \sin^2(\pi k/N)$. 

7. Discrete Fourier Transform