

CHAPTER 9

Fourier Series and Transforms (Online)

9.7 Discrete Fourier Transforms

A Fourier *series* models a function as a discrete set of numbers (coefficients of a sum); a Fourier *integral* models a function as a continuous function (the transform). In both cases the function you start with is continuous. In this section we will use Fourier analysis on a data set that is discrete to begin with: a set of points, or data, rather than an algebraically defined function.

9.7.1 Explanation: Discrete Fourier Transforms

Throughout this chapter we have used Fourier analysis (series or transforms) to see what oscillations make up a particular function. In practice, though, we rarely measure a function such as $f(x) = e^{-x^2}$. We measure data points.

Consider, for example, the Motivating Exercise (Section 9.1). Observers repeatedly measure the velocity of a star over many years. A star with no planets around it should have a constant velocity, a star with one planet should oscillate like a sine wave, a star with two planets should follow a superposition of two sine waves, and so on. A “discrete Fourier transform” starts with individual velocity measurements and finds the oscillations represented by those points, enabling us to determine the presence of planets.

But Figure 9.1 doesn’t look like a superposition of sine waves, does it? Real oscillations are buried under random noise caused by limitations of the measuring instruments, atmospheric distortion, and other factors. These random fluctuations are often many times greater than the oscillations themselves. Amazingly, the Fourier transform can generally cut through such noise to find the underlying order.

Below we present the formula for a discrete Fourier transform, but first let’s make sure the letters are clear. Take a look at Figure 9.19. We begin with a function $f(x)$ representing N individual points spaced Δx apart. For instance, if we have $f(0)$, $f(20)$, $f(40)$, and $f(60)$, then $\Delta x = 20$ and $N = 4$. (The formula assumes Δx is constant: that is, the points are sampled at evenly spaced intervals.)

We refer to the data points as f_0, f_1, f_2, f_3 , and so on. We use the letter r as an index for this list. In our example above, f_3 would mean $f(60)$; more generally, f_r means $f(r\Delta x)$.

Our goal is to express the data as *<some coefficient>* times an oscillation of a particular frequency, plus *<a different coefficient>*

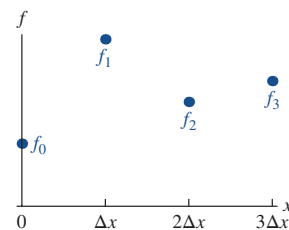


FIGURE 9.19 With four data points $r = 0, 1, 2, 3$.



2 Chapter 9 Fourier Series and Transforms (Online)

times an oscillation of a different frequency, and so on. We refer to these coefficients as \hat{f}_1, \hat{f}_2 and so on, using n as an index into this list. Below, then, we present the formula for turning the f_r list of data points into the \hat{f}_n list of frequency coefficients.

The Formula for a Discrete Fourier Transform

Suppose a function $f(x)$ is known at N discrete points: $f_r = f(r\Delta x)$ for $r = 0, 1, \dots, N - 1$. The “discrete Fourier transform” (DFT)^a of f is a set of points \hat{f}_n with $-N/2 \leq n \leq N/2$ given by the formula:

$$\hat{f}_n = \sum_{r=0}^{N-1} f_r e^{-2\pi i n r / N} \quad (9.7.1)$$

The numbers \hat{f}_n are proportional to the amplitudes of oscillations with frequencies given by:

$$p = \frac{2\pi n}{N\Delta x} \quad (9.7.2)$$

It’s easy to glide over those equations too quickly. To slow down, consider the example of \hat{f}_3 generated from a set with $N = 100$ data points.

- Equation 9.7.1 tells us how to find \hat{f}_3 : it is a sum over all the values in the original data set.

$$\begin{aligned} \hat{f}_3 = & f_0 + f_1 e^{-6\pi i / 100} + f_2 e^{-12\pi i / 100} \\ & + f_3 e^{-18\pi i / 100} + \dots + f_{99} e^{-6(99)\pi i / 100} \end{aligned}$$

Then \hat{f}_4 is a different sum over the entire original data set, and so on. (Such linear combinations can be concisely represented with matrices; see Problem 9.114.)

- Equation 9.7.2 tells us what \hat{f}_3 represents: it tells us how much of the original data set oscillates with frequency $6\pi / (100\Delta x)$. Note that the denominator $N\Delta x$ is the entire span of the original data set which means that $2\pi / (N\Delta x)$, $4\pi / (N\Delta x)$, $6\pi / (N\Delta x)$, etc. are all the frequencies that fit perfectly into that domain.

As you can see, calculating N -values for your N data points in this way requires summing N^2 terms. But discrete Fourier transforms are so important that decades of research have gone into algorithms that can quickly calculate the DFT of a list with billions of input points. The need to do so arises more often than you might think, in applications ranging from image manipulation to electronic signal processing.

Interpreting a Discrete Fourier Transform

As with so many of these formulas, Equation 9.7.1 hides a number of subtleties.

- The DFT formula assumes that the function you are modeling is periodic, and that your data represent a full period. For instance (as we mentioned above) if you start with 100 data points with spacing 1, the terms in the DFT represent a period of 100, a period of 100/2, a period of 100/3, and so on; all of them start over at precisely 100. If you *inverse* transform your DFT you will recreate the original 100 data points and then reproduce them again and again.

^aThe acronym “DFT” is widely used for Discrete Fourier Transform, but unfortunately it has a number of other uses. The most problematic for our readers may be “Density Functional Theory,” a model of many-body quantum systems used by physical chemists. There is probably less danger of confusion with the UK’s “Department For Transport.”



- Equation 9.7.1 makes no mention of Δx so the DFT depends on the values of your data points but not on how often they were sampled. If you sample 100 points over the course of a year, \hat{f}_3 tells you how much your data oscillate with a period of a third of a year. If you sample the same 100 points in a minute you will get the same \hat{f}_3 , this time representing a period of 20 seconds.
- Strictly speaking \hat{f}_n is not the amplitude of an oscillation; \hat{f}_n/N is the amplitude. That matters for an inverse discrete Fourier transform, but for most purposes we only care about the *relative* amplitudes of the oscillations so we ignore that detail.
- We said above that Equation 9.7.1 assumes its input (the f_r numbers) are periodic, repeating the same N numbers over and over. The output (the \hat{f}_n numbers) will *also* be periodic with period N . For instance if you put in 100 data points you will find that $\hat{f}_{101} = \hat{f}_1$ and $\hat{f}_{102} = \hat{f}_2$ and so on. In most cases the frequencies of interest are given by $-N/2 \leq n \leq N/2$.⁴ A negative n just means a negative exponent in the complex exponential, but n and $-n$ both represent oscillations with the same period, just as they do in regular Fourier series with complex exponentials.
- Just as with a regular Fourier transform, if the input data are real then \hat{f}_{-n} will always be the complex conjugate of \hat{f}_n . Since DFTs are generally used to model measured data, which is by definition real, you can generally get away with only calculating about half of the amplitudes. (In our example above you could calculate \hat{f}_0 through \hat{f}_{50} the hard way and then you would know \hat{f}_{-50} through \hat{f}_{-1} .)

EXAMPLE

Discrete Fourier Transform

Problem:

Find a discrete Fourier transform of the list $f(0) = 1, f(3) = 4, f(6) = 2, f(9) = 3$.

Solution:

We plug $f_0 = 1, f_1 = 4, f_2 = 2, f_3 = 3$, and $N = 4$ into Equation 9.7.1.

$$\begin{aligned}\hat{f}_0 &= 1 + 4 + 2 + 3 = 10 \\ \hat{f}_1 &= 1 + 4e^{-2\pi i/4} + 2e^{-2\pi i(2)/4} + 3e^{-2\pi i(3)/4} = 1 + 4(-i) + 2(-1) + 3i = -1 - i \\ \hat{f}_2 &= 1 + 4e^{-2\pi i(2)/4} + 2e^{-2\pi i(2)(2)/4} + 3e^{-2\pi i(3)(2)/4} = 1 + 4(-1) + 2 + 3(-1) = -4\end{aligned}$$

It's worth going through that exercise to see the patterns in the formula. You can see the r -values counting 0, 1, 2, 3 as you move across the terms in any line, and you can see the n -values counting 0, 1, 2 as you count down the lines.

We could similarly calculate \hat{f}_{-1} , but the fact that the inputs were all real guarantees that $\hat{f}_{-1} = \hat{f}_1^* = -1 + i$. (You can do the calculation directly and verify this.) We know for the same reason that $\hat{f}_{-2} = \hat{f}_2^*$, and we know from periodicity that $\hat{f}_{-2} = \hat{f}_2$. The only way those can both be true is if \hat{f}_2 is real, which we see it is. With N real inputs, $\hat{f}_{N/2}$ will always be real (assuming N is even). Can you see why the same argument guaranteed that \hat{f}_0 would be real as well?

So the DFT, from $n = -N/2$ to $N/2$, is $(-4, -1 + i, 10, -1 - i, -4)$.

Once you have gone through the process once or twice, there's no great value in repeating it. DFTs are meant for extremely large data sets and are performed by computers.

⁴Technically $\hat{f}_{-N/2} = \hat{f}_{N/2}$ so you only need to calculate values from $n = -N/2 + 1$ to $N/2$.



4 Chapter 9 Fourier Series and Transforms (Online)

Question: What does $\hat{f}_2 = -4$ represent?

Answer:

From Equation 9.7.2 it represents the amplitude associated with the frequency $2\pi(2)/(4 \times 3) = \pi/3$.

Inverse Discrete Fourier Transforms

When we take the Fourier series of a function we get a list of coefficients c_n . If we multiply each of those coefficients by $e^{\pi i n x / L}$ and add them, we recover the original function. In a similar way, a discrete Fourier transform takes as input the value of a function at certain discrete points and gives as output the coefficients \hat{f}_n , and the “inverse discrete Fourier transform” calculates the original values f_r from those coefficients.

Inverse Discrete Fourier Transform

The inverse discrete Fourier transform of a set of points \hat{f}_n is

$$f_r = \frac{1}{N} \sum_{n=-N/2+1}^{N/2} \hat{f}_n e^{2\pi i n r / N} \quad (9.7.3)$$

(The sum can go over any N consecutive values of n , and it is often written from 0 to $N - 1$, but we prefer this way because it focuses on the most physically meaningful values. Clearly for odd N you would have to adjust the limits of the sum.)

Recall that the x -value of each f_r is $r\Delta x$. If we plug $r = x/\Delta x$ into Equation 9.7.3 the exponential becomes $e^{2\pi i n x / (N\Delta x)}$, which is why we said above that each mode \hat{f}_n has frequency $2\pi n / (N\Delta x)$. We can also see from this formula that \hat{f}_n is not technically the coefficient of the oscillation; \hat{f}_n / N is.

Remember that the DFT coefficients \hat{f}_n are periodic; they are N numbers that repeat forever. You can generate an inverse DFT from any N such numbers in a row and get a function that perfectly replicates the original points you used to create your DFT. Between those points, however, the behavior may surprise you unless you use the range specified in Equation 9.7.3. See Problem 9.107.

With the DFT and inverse DFT formulas you can translate a data set into “Fourier space” and then recover the original data. That may not seem very useful, but for many applications we can transform a set of data into Fourier space, manipulate it in some way, and then do an inverse transform to get back to regular space. This is illustrated in the example below.

EXAMPLE

Smoothing the Rough Edges

Problem:

Copy an image from the Web and eliminate all high frequency oscillations from it.

Solution:

We chose an image of Rodin’s sculpture “The Thinker.” We imported this photo into Mathematica, which stored it as a 3072×2048 grid of numbers. We then asked



Mathematica to take a discrete Fourier transform of that grid of numbers. (We haven't given you the formula for a multivariate discrete Fourier transform, but Mathematica's built-in DFT routines knew just what to do.) The result was a 3072×2048 grid of complex numbers representing the DFT. An inverse Fourier transform of this grid reproduced the original picture exactly.

We then set about 99% of the numbers in the DFT to zero, leaving only the low frequency terms. An inverse Fourier transform of the resulting frequencies produced a softened version of the original. Finally, we set over 99.99% of the numbers in the DFT to zero. An inverse transform of the resulting set was still very recognizable, but blurry.



Image reproduced from 100% of the data in Fourier space, image reproduced from 1% of the data, and image reproduced from 0.006% of the data

This technique can be used to deliberately soften images, but it also provides powerful data compression. You can store pictures in a fraction of the space required for an uncompressed image.

The implications for downloading pictures (in a Web browser for instance) are also important. If you download a picture as pixels the user sees the top row of pixels appear at first, filling in toward the bottom over time. But if you download the same information as frequencies the user sees the entire image almost instantly, sharpening to full resolution over time.

Where Did Those Formulas Come From?

Consider a regular Fourier series for a function with period $2L$: $f(x) = \sum c_n e^{\pi i n x / L}$ where the coefficients are given by

$$c_n = \frac{1}{2L} \int_0^{2L} f(x) e^{-\pi i n x / L} dx$$

If we know the function $f(x)$ we can find the coefficients c_n by integrating, analytically or numerically. If we only know $f(x)$ at N evenly spaced intervals $x = r\Delta x$ then we must approximate this integral with a sum.

$$c_n \approx \frac{1}{2L} \sum_{r=0}^{N-1} f(r\Delta x) e^{-\pi i n r \Delta x / L} \Delta x$$



6 Chapter 9 Fourier Series and Transforms (Online)

The spacing Δx is related to the total period $2L$ and the number of sample points N by $\Delta x = 2L/N$. Substituting this in gives

$$c_n \approx \frac{1}{N} \sum_{r=0}^{N-1} f(r\Delta x) e^{-2\pi i n r / N}$$

We could have left the formula like this, but historically the terms in a DFT are not defined to make $\hat{f}_n \approx c_n$, but rather $\hat{f}_n \approx Nc_n$, which thus eliminates the $1/N$ from the formula, giving us Equation 9.7.1. A similar calculation can be used to derive Equation 9.7.3 for the inverse discrete Fourier transform.

A Few Practical Notes

While discrete Fourier transforms have been around for centuries, their use didn't become common until Cooley and Tukey published a DFT algorithm that is in many cases millions of times faster than straightforwardly applying Equation 9.7.1.⁵ That algorithm is now known as a "Fast Fourier Transform." One of us (GF) runs simulations of the early universe using hundreds of processors to take FFTs of datasets with over 60 billion points, and such large transforms are common in many areas of science and engineering.

As a practical matter, the FFT is most efficient when the number of data points N is a power of 2. If you are working with any other size data set it is usually worth padding the end with extra zeroes to make N a power of 2.

It's also important to remember that a DFT gives us oscillations at the frequencies $p = 2\pi/(N\Delta x)$, $4\pi/(N\Delta x)$... $\pi/\Delta x$. That highest frequency $p = \pi/\Delta x$ is called the "Nyquist frequency," and it sets a limit on how high a frequency we can find from our data. The fact that the Nyquist frequency is inversely proportional to Δx makes sense: if we sample once per second, we won't find an oscillation that happens ten times per second! If a function oscillates much faster than our sampling rate, our discrete Fourier transform will be subject to "aliasing." The power that is actually in modes above the Nyquist frequency will appear incorrectly in the lower frequency modes, causing them to come out higher than they should. As a practical matter you should either know ahead of time what range of oscillation frequencies your data is likely to have or figure it out by trial and error, and make sure you sample frequently enough that your Nyquist frequency is higher than the highest significant frequency of your physical system.

9.7.2 Problems: Discrete Fourier Transforms

9.99 Walk-Through: Discrete Fourier Transforms. You've measured the following data points for a function $f(x)$: $f(0) = 2$, $f(2) = 3$, $f(4) = -6$, $f(6) = 0$.

- Use Equation 9.7.1 to calculate \hat{f}_0 , \hat{f}_1 , and \hat{f}_2 .
- Find \hat{f}_{-1} without using Equation 9.7.1. *This should take no more than 20 seconds.*
- What are \hat{f}_{-2} and \hat{f}_3 ? *Again, more than 20 seconds means you're doing it wrong.*

(d) What frequencies p are represented by the terms \hat{f}_{-1} , \hat{f}_0 , \hat{f}_1 , and \hat{f}_2 ?

For Problems 9.100–9.105 find the discrete Fourier transform of the given list of numbers, going from $n = -N/2$ to $N/2$. Give the smallest magnitude non-zero frequency and the largest frequency (the Nyquist frequency) represented by the DFT.

9.100 $f(0) = 1$, $f(5) = 2$

⁵Cooley, James W.; Tukey, John W. (1965). "An Algorithm for the machine calculation of complex Fourier series". *Math. Comput.* 19: 297–301. The algorithm had been independently discovered numerous times before Cooley and Tukey, beginning with Gauss in 1805. For a discussion of the FFT algorithm itself see "Numerical Recipes" by Press, et. al.



9.7 | Discrete Fourier Transforms 7

9.101 $f(0) = 10, f(6) = -1, f(1.2) = 0, f(1.8) = -7$


9.102 $f(0) = 1, f(2) = 2, f(4) = -2, f(6) = -1$

9.103 $f(0) = 5, f(1) = 4, f(2) = 3, f(3) = 2$


9.104 $f(0) = 5 + i, f(2) = i, f(4) = 3, f(6) = 1 - i$

9.105 $f(0) = 2, f(4) = 4, f(8) = 0, f(12) = -6,$
 $f(16) = 0, f(20) = -4$

9.106 Calculate the DFT of $f(0) = 5, f(5) = -5,$
 $f(10) = 5, f(15) = -5$. Explain why your results
make sense. In other words, how could you
have predicted most of the values you found
without having to calculate anything?

9.107  Consider the points $f(0) = 2, f(3) = 7,$
 $f(6) = -2, f(9) = 1, f(12) = 4, f(15) = -3,$
 $f(18) = 2, f(21) = 2$.

- Calculate the DFT coefficients \hat{f}_n
for $0 \leq n \leq 7$.
- On one plot show the real and imaginary
parts of $(1/8) \sum_{n=0}^7 \hat{f}_n e^{2\pi i n x / 12}$. Show the
original 8 points f_r on the same plot.
- Repeat Part (b) for $-3 \leq n \leq 4$. *Hint:* you
can easily get all the \hat{f}_n -values you need
from the ones you already calculated.
- On one plot show the original points
 f_r and the real and imaginary parts of
 $(1/8)[(1/2)(\hat{f}_{-4} e^{-8\pi i n x / 12} + \hat{f}_4 e^{8\pi i n x / 12} +$
 $\sum_{n=-3}^3 \hat{f}_n e^{2\pi i n x / 12})]$.
- You should have found that all three func-
tions you just defined go through all of
the original data points f_r , but behave
differently in between them. Looking at
your plots, which of these functions is
likely to be the most useful for model-
ing your original function, and why?


9.108  Have a computer generate a table of
256 points $f(x)$ where x goes from 0 to 30. At
each point let $f(x) = 2 \sin(2x) + 2 \sin(5x) + R$,
where R is a random number (different at
each point) from -4 to 4. Plot the data points
on a graph of $f(x)$ vs. x . Then take a discrete
Fourier transform of the data and plot the
results on a graph of $\hat{f}(p)$ vs. p . Explain why
the results appear the way they do.

9.109 Use $x = r\Delta x, p = n\Delta p = 2\pi n / (N\Delta x)$, and
 $\hat{f}_n = Nc_n$ to rewrite Equation 9.7.3 so it looks
like the usual expression for a Fourier series.
A normal Fourier series has infinitely many
terms, but this one will only have N . All
the others can be taken to be zero.


9.110 Plug Equation 9.7.1 into Equation 9.7.3
and verify that it does simplify to f_r . At
some point in the calculation you should

encounter a term of the form $\sum_{j=-N/2+1}^{N/2} e^{2\pi k i j / N}$,
where k is an integer. This sum equals
zero unless $k = 0$. (You can easily figure
out what it equals when $k = 0$.)

9.111 Use Equation 9.7.1 to show that $\hat{f}_{n+N} = \hat{f}_n$.
Formally a DFT is periodic in frequency
space, but in practice that really means
all of the frequencies outside the range
 $|p| \leq \pi / \Delta x$ are meaningless.

9.112  Consider the function $f(x) = e^{\cos x}$.

- Have a computer generate a table of
200 values of $f(x)$ ranging from $x = 0$ to
 $x = 99.5$. Plot these values on a graph of f
vs. x . (Just plot the values in your table,
not the curve for the function.)
- Take a discrete Fourier transform of your
list of values. What frequency is associated
with the n^{th} term in that list? Plot the mod-
uli of the values of the discrete Fourier
transform from $n = 1$ to $n = 100$, with
the horizontal axis showing the associ-
ated frequencies. (Leave out $n = 0$, which
just represents the constant term.)
- What frequency shows the highest peak
in the DFT? What does that tell you
about the original function? (*Hint:*
you could have predicted this peak
before doing the calculations.)
- What frequency shows the second highest
peak in the DFT? What does that tell you
about the original function? (You proba-
bly couldn't have predicted this one.)

9.113  Consider the list
 $f_r = (1, 2, 3, 4, 3, 2, 1, 2, 3, 4, 3, 2)$.


- Plot these data points, using the
 x -values $1, 2, 3, 4, 5, 6, \dots$
- Take a discrete Fourier transform of
the list, then take an inverse discrete
Fourier transform, and then plot the
resulting points. This plot should look
identical to the previous one.
- Take a discrete Fourier transform of f_r ,
set $\hat{f}_0 = 0$ in the resulting list, then take
its inverse discrete Fourier transform
and plot the results. How does this look
different from your original plot?
- Take a discrete Fourier transform of f_r ,
set $\hat{f}_2 = \hat{f}_{-2} = 0$, take an inverse discrete
Fourier transform, and plot the results.
How does this plot look different from
your original plot? *Hint:* some computers
store the results in the order $\hat{f}_0, \hat{f}_{N-1},$ so
the component \hat{f}_{-2} may be stored as \hat{f}_{N-2} .



8 Chapter 9 Fourier Series and Transforms (Online)

9.114 (This problem requires linear algebra.)

- (a) Starting from Equation 9.7.1 write a 2×2 matrix for converting a pair of data points f_r into a DFT \hat{f}_n . (Remember that n and r go from 0 to 1.)
- (b) Starting from Equation 9.7.3 write a 2×2 matrix for converting \hat{f}_n into f_r .
- (c) Prove that the two matrices you just wrote are inverses. What does this tell you about the two transformations?
- (d) Write a matrix to find \hat{f}_0, \hat{f}_1 and \hat{f}_2 from three data points f_0, f_1, f_2 .

9.115  Look through the Example “Smoothing the Rough Edges” on Page 4. Repeat this process for another image. Choose an image, import it into a computer algebra program,

take a discrete Fourier transform of the image data, set the high frequency components to zero, and inverse transform it to get a new image. You may need to do some trial and error to figure out exactly which components to eliminate. Be aware that when you eliminate Fourier coefficients the inverse discrete Fourier transform will usually give you complex numbers. The easiest way to deal with this is to just take the real part after you take the IDFT. *Hint:* If you use a color image then each pixel will most likely be a list of three numbers instead of a single number. You can still do the same thing. Each component in Fourier space will be a list of three numbers and you can set them all equal to zero for the high frequency modes.

