A Tour through Sampling Theorem, Discrete Fourier Transform, Orthogonality of Complex Exponentials and Inverse DFT

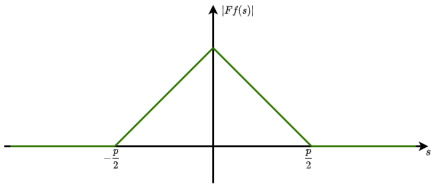
J.R

June 12, 2021

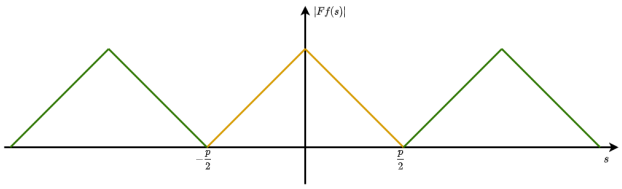
In this note, I wanna sort out the logic of going from Sampling Theorem to DFT, and finally get to Inverse DFT by exploiting the orthogonality of complex exponentials. The expected effect of this note is to help you (and, of course, me) to understand DFT and np.fft package, what exactly it takes in and returns and why. Most of the content is from EE261 lectures, with some extra examples made up by myself.

1 Sampling Theorem

Suppose f is a band-limited signal with bandwidth p, i.e. $\mathcal{F}f(s) = 0 \quad \forall |s| > \frac{p}{2}$. The spectrum of f looks like



we can first convolve $\mathcal{F}f$ with a Shah function Ш_p to get a period version of $\mathcal{F}f$ with period p, then get back the original $\mathcal{F}f$ by multiplying a rectangle function Π_p



This operation can be written as

$$\Pi_p \cdot (\mathcal{F}f * \text{Ш}_p) = \mathcal{F}f \tag{1.1}$$

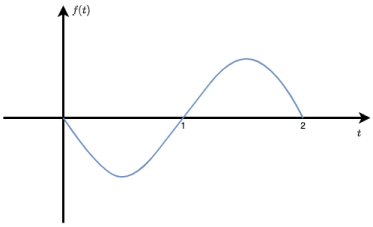
Though this seems to be a trivial operation, it has significant implications. We take the inverse Fourier Transform to both sides. The righthand side becomes $\mathcal{F}^{-1}\mathcal{F}f = f$. By applying the convolution theorem, we can see that the lefthand side becomes

$$
\begin{aligned}
&\mathcal{F}^{-1}\Big[\Pi_p \cdot (\mathcal{F}f * \text{Ш}_p)\Big] \\
=&\mathcal{F}^{-1}\Pi_p * \mathcal{F}^{-1}\Big[\mathcal{F}f * \text{Ш}_p\Big] \\
=&\Big[p\text{sinc}(pt)\Big] * \Big[f \cdot \mathcal{F}^{-1}\text{Ш}_p\Big] \\
=&\Big[p\text{sinc}(pt)\Big] * \Big[f \cdot \frac{1}{p}\text{Ш}_{1/p}\Big] \\
=&\text{sinc}(pt) * \Big[f(t) \cdot \sum_{k=-\infty}^{\infty} \delta_{k/p}(t)\Big] \\
=&\text{sinc}(pt) * \Big[\sum_{k=-\infty}^{\infty} f(\frac{k}{p}) \cdot \delta_{k/p}(t)\Big] \\
=&\sum_{k=-\infty}^{\infty} f(\frac{k}{p})\text{sinc}(pt) * \delta_{k/p}(t) \\
=&\sum_{k=-\infty}^{\infty} f(\frac{k}{p})\text{sinc}(p(t-\frac{k}{p})) \qquad (1.2)
\end{aligned}
$$

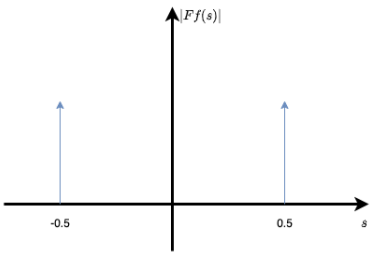
Thus we have $f(t) = \sum_{k=-\infty}^{\infty} f(\frac{k}{p})\text{sinc}(pt-k)$. This formula tell us that, if f is a band-limited signal with bandwidth p, we can interpolate all values of f in terms of discrete samples spaced no more than $\frac{1}{p}$. However, when we sample at a rate lower than p, $\Pi_p \cdot (\mathcal{F}f * \text{Ш}_p)$ does not give back $\mathcal{F}f$ and something will go wrong. Let's see an example.

eg 1. Pure sine function

$f(t) = \sin(2\pi 0.5t)$ has frequency 0.5, and f has Fourier Transform $\mathcal{F}f(s) = \frac{1}{2i}(\delta_{0.5}(s) - \delta_{-0.5}(s))$. The graph of f and $\mathcal{F}f$ looks like



(a)

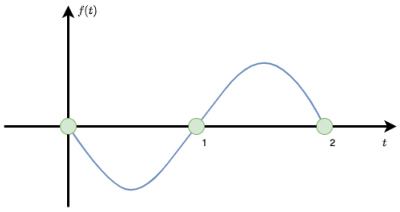


(b)

Note that $\mathcal{F}f(s) = 0$ for s strictly larger than 0.5.

The bandwidth of f is anything larger than 1. When we interpolate f using (1.2) with sampling rate p, what we are really doing (1.1). However, when we sample at a rate lower than the bandwidth, the equality in (1.1) does not hold anymore.

For example, when we interpolate f with a sampling rate exactly 1, the sample points are shown below



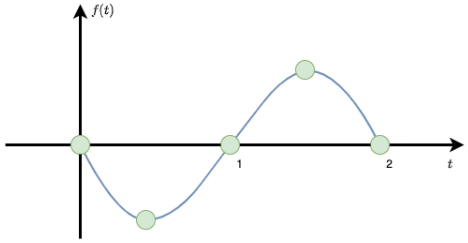
we have

$$\begin{aligned}\mathcal{F}f * \text{Ш}_1 &= \frac{1}{2i}(\delta_{0.5} - \delta_{-0.5}) * \sum_{k=-\infty}^{\infty} \delta_k \\ &= \frac{1}{2i} \sum_{k=-\infty}^{\infty} (\delta_{k+0.5} - \delta_{k-0.5}) \\ \Pi_1 \cdot (\mathcal{F}f * \text{Ш}_1) &= \frac{1}{2i}(\delta_{0.5} - \delta_{-0.5} - \delta_{0.5} + \delta_{-0.5}) = 0\end{aligned}$$

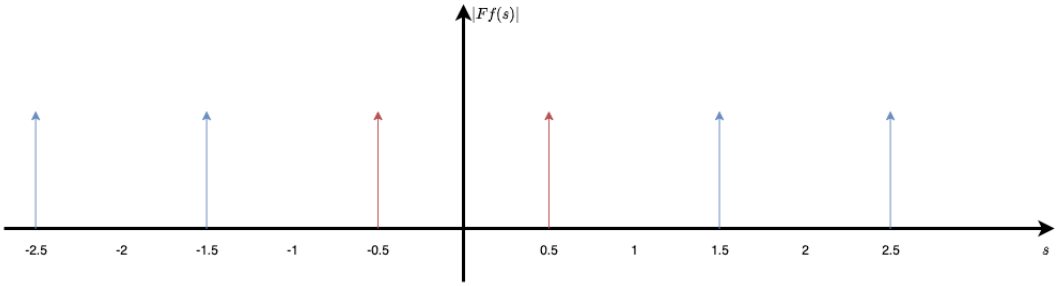
Then take the Fourier Transform, $\mathcal{F}^{-1}0 = 0$. We get a constant function 0 because we were sampling at too low a rate so that those sample points completely missed the oscillations between samples. Just like we see a resting fan when the fan is spinning at a certain speed.

Though the constant function is not the original sine function, it agrees with the sine function at the sample points, and that's called a "alias" of f.

Now let's do something reasonable. f has bandwidth just larger than 1, let's sample with a rate 2. When we sample at a rate 2, the samples are spaced 0.5 from neighbor sample points. The sample points are shown below

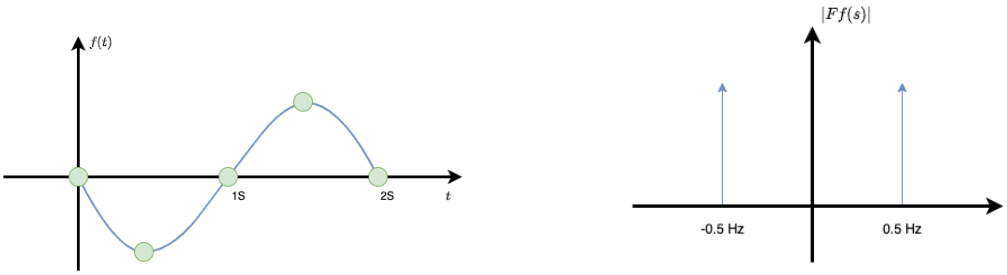


Always remember that when we interpolate from sample points, what we are doing is convolving and multiplying in the spectral domain. When we sample at a rate 2, we are convolving with Ш_2 and multiplying by Π_2, as shown below. $\Pi_2 \cdot (\mathcal{F}f * \text{Ш}_2)$ gives back the original Fourier Transform,



so when we perform the inverse Fourier Transform, we get back f.

For the sake of practicalness, let bring in some units. For example, the unit of $f(t) = \sin(2\pi 0.5t)$ is in seconds, then f has frequency 0.5 Hz, f and $\mathcal{F}f$ looks like



(a) (b)

the bandwidth of f is 1Hz, the samples should be spaced at most $\frac{1}{p} = \frac{1}{1/\text{second}} = 1$ second from neighbors.

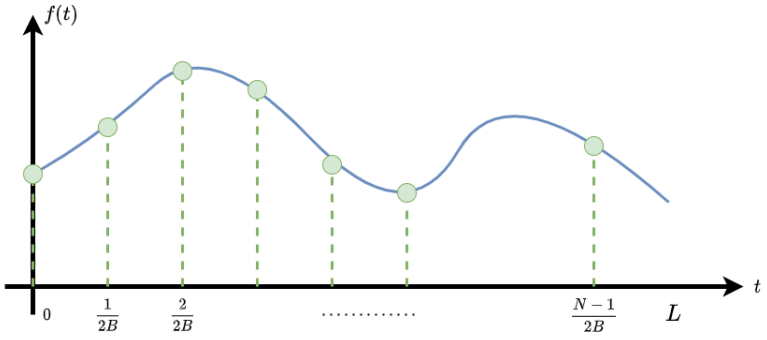
2 Discrete Fourier Transform

Until now, everything is continuous, to make an analogy of Fourier Transform in discrete space, we need to

- Approximate f with a discrete function
- Approximate $\mathcal{F}f$ with a discrete function

Assume function $f(t)$ to be time-limited to $[0, L]$, i.e. $f(t) = 0 \forall t > L$ or $t < 0$, and band-limited to $[0, 2B]$, i.e. $\mathcal{F}f(s) = 0 \forall s > 2B$ or $s < 0$. Here we assume than $\mathcal{F}f$ has non-zero domain $[0, 2B]$ in stead of $[-B, B]$ for the convenience of indexing that will come up later.

According to the sampling theorem, to reasonably approximate a band-limited signal with bandwidth $2B$, we want to take samples at a rate $2B$, so the samples have spacing $\frac{1}{2B}$, giving us a total number of N samples, as shown below.

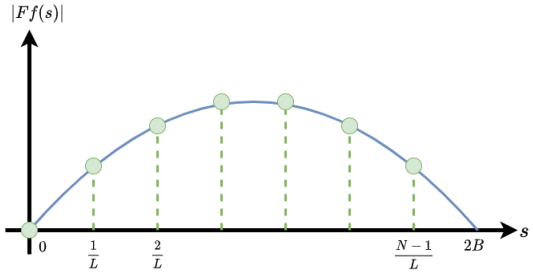


The sample form of f can be written as $f_{sp}(t) = f(t) \sum_{k=0}^{N-1} \delta_{t_k}(t) = \sum_{k=0}^{N-1} f(t_k)\delta_{t_k}(t)$, which is continuous in terms of t, so that we can easily compute its Fourier Transform

$$\mathcal{F}f_{sp}(s) = \sum_{k=0}^{N-1} f(t_k)e^{-2\pi i t_k s} \tag{2.1}$$

, where $t_k = \frac{k}{2B}$.

Then we want to approximate $\mathcal{F}f_{sp}$, which is a continuous function, with a discrete function. Again, we want to take sample under the guide of sampling theorem. Now, do not think f_{sp} as in the time domain and $\mathcal{F}f_{sp}$ in the frequency domain, just think $f_{sp}(t)$ and $\mathcal{F}f_sp(s)$ as normal functions in domain t and s, respectively. Then the Fourier Transform of $\mathcal{F}f$ takes us from domain of s back to the domain of t. Thus, we know that $\mathcal{F}f$ has bandwidth L. That's a tricky change of viewpoint, take some time and think it through. So, $\mathcal{F}f$ has bandwidth L, thus we want take samples spaced $\frac{1}{L}$, the sampling of $\mathcal{F}f$ looks like



Notice that we assume that f is band-limited to $[0, 2B]$, however, the spectrum of the f_{sp} is different than spectrum of f, that is, $\mathcal{F}f$ is not identical to $\mathcal{F}f_{sp}$. Actually, f_{sp} often have larger bandwidth than f, because there are discontinuities at sample points and it takes high frequencies to make jump discontinuities. Here we just simply assume that f_{sp} is also band-limited to $[0, 2B]$. I guess the motivation is, those frequencies outside $[0, 2B]$ did not belongs to f. Low frequencies, which are frequencies of f, come in to meet sample points and high frequencies come in to make jump continuities, so we just pretend they are not there when we approximate the spectrum of f. It's just my guess.

Another thing to notice is that when we sample f, we took $N = 2BL$ samples, here when we take samples spaced $\frac{1}{L}$ apart on an interval $[0, 2B]$, it also give us a total number of N samples.

Now we can write the sampled version of the Fourier Transform of sampled f (take time to get it straight) as

$$\begin{aligned}(\mathcal{F}f_{sp})_{sp}(s) &= \mathcal{F}f_{sp}(s)\sum_{m=0}^{N-1}\delta_{s_m}(s)\\ &= \left(\sum_{k=0}^{N-1} f(t_k)e^{-2\pi i t_k s}\right)\left(\sum_{m=0}^{N-1}\delta_{s_m}(s)\right)\\ &= \sum_{k=0}^{N-1} f(t_k)\left(\sum_{m=0}^{N-1} e^{-2\pi i t_k s_m}\delta_{s_m}(s)\right)\end{aligned}$$

Where $s_m = \frac{m}{L}$ When s is a sample point, erasing all 0 terms, we get the sampled version of the Fourier Transform of sampled version of f, (again, take time and get this straight)

$$(\mathcal{F}f_{sp})_{sp}(s_m) = \sum_{k=0}^{N-1} f(t_k)e^{-2\pi i t_k s_m} \tag{2.2}$$

Here we still has "continuous term" $t_k = \frac{k}{2B}$ and $s_m = \frac{m}{L}$. But $t_k s_m = \frac{km}{2BL} = \frac{km}{N}$ is a "discrete term" since k and m are integers. Now we are ready to declare victory.

Suppose we have a discrete signal that contains N data, $f = [f_0, f_1, ..., f_k]$, we define **Discrete**

Fourier Transform, which gives us another discrete signal $\mathcal{F}f$, as

$$\mathcal{F}f[m] = \sum_{k=0}^{N-1} f[k]e^{-2\pi i \frac{km}{N}} \tag{2.3}$$

which is a "purely discrete" operation.

Though it is okay to only think about this as a well-defined operation on a discrete signal and ignore the L and B stuff, I think it can be helpful to regard the discrete signal as coming from a continuous signal and keep those "approximation steps" in mind.

Another thing to notice is the relationship between the spacing in time domain and spacing in frequency domain. In time domain, we took N samples in interval $[0, L]$, the spacing Δt between 2 neighbor samples is $\frac{1}{2B}$. In frequency domain, we took N samples in interval $[0, 2B]$, the spacing Δs between 2 neighbor samples is $\frac{1}{L}$. Then we have $\Delta s \Delta t = \frac{1}{2BL} = \frac{1}{N}$. This reveals the reciprocal relationship, which is the relationship often shows up in the context of Fourier Transform, between Δs and Δt. Also, it tells us that once any two of Δs, Δt and N are fixed, than the other one is determined by the fixed two, and that is what np.fft.fftfreq does.

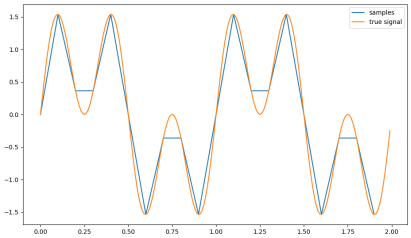
eg2. np.fft.fft

Suppose we have a signal $f(t) = sin(2\pi t) + sin(2\pi 3t)$ that lasts for 2 seconds, where t is in seconds. From the formula we know that f has frequencies 1Hz and 3Hz. But let's suppose that we do not have the formula for the signal and the only thing we know about it is that it has no components that's higher than 5Hz, let's get back the formula with Discrete Fourier Transform.

First, since we know f is band-limited to [-5Hz,5Hz], we have its bandwidth $2B = 10Hz$. According to the sampling theorem, spacing between samples should at most be $\frac{1}{10Hz} = 0.1s$, giving us a total number of 20 samples. So let's generate the sampled signal.

```
import numpy as np
fs = [1, 3]          # frequencies of sin wave
B = 5               # prior knowledge of bandwidth
duration = 2    # 2 seconds
bandwidth = 2 * B
sample_spacing = 1 / bandwidth
N = int(duration / sample_spacing)
xs_cont = np.arange(0, duration, 0.01)
ys_cont = np.zeros_like(xs_cont)
for f in fs:
        ys += np.sin(2 * np.pi * f * xs)
                ys_cont += np.sin(2 * np.pi * f * xs_cont)
plt.plot(xs, ys, label='samples')
plt.plot(xs_cont, ys_cont, label='true signal')
plt.legend()
plt.show()
```

The sampled signal looks like



Now we have the sampled signals, and from now on, we will no longer have the formula of f and we want to recover the formula from the sampled signal.

First let's take a look at np.fft.fftfreq. Remember that when we fix Δt and N, Δs is then determined by Δt and N. Here we have $L = 2s$, the spacing in frequency domain is $\frac{1}{L} = \frac{1}{2s} = 0.5Hz$. Since we take N samples in both time and frequency domain, samples spaced $0.5Hz$ apart gives us the frequencies [-5Hz, -4.5Hz, ... , 4Hz, 4.5Hz], making the interval symmetric. The np.fft.fftfreq just calculate the sample points in frequency domain for us, but in a different order. And np.fft.fft returns the "Fourier Coefficients" corresponds to each frequency returned by np.fft.fftfreq. You will see why I added quote on "Fourier Coefficients" later. The code below show the sampled frequencies and the corresponding coefficient if the coefficient havs magnitude larger than 1e-8.

```
Ff = np.fft.fft(ys)
freqs = np.fft.fftfreq(N, sample_spacing)
for s, F in zip(freqs, Ff):
                    if np.abs(F) > 1e-8:
                    print(s, F)
```

We get frequency-coefficient pairs

1.0	(-4.715509157353898e-15-9.999999999999995j)
3.0	(-8.418070490254451e-17-10.000000000000005j)
-3.0	(-4.829976635421454e-16+10.000000000000005j)
-1.0	(-4.905945511304625e-15+9.999999999999995j)

For frequencies 1.0, 3.0, -3.0 and -1.0 have coefficient -10i, -10i, 10i, 10i, respectively, with some rounding done by human under the table. Those are exactly the frequencies we wanted. However, if we want to reassemble f from those frequencies and coefficients, we get

$$\begin{aligned}
&-10ie^{2\pi i1t} - 10ie^{2\pi i3t} + 10ie^{-2\pi i1t} + 10ie^{-2\pi i3t} \\
=&10i(e^{-2\pi i1t} - e^{2\pi i1t}) + 10i(e^{-2\pi i3t} - e^{2\pi i3t}) \\
=&10i(-2isin(2\pi 1t)) + 10i(-2isin(2\pi 3t)) \\
=&20(sin(2\pi 1t) + sin(2\pi 3t))
\end{aligned}$$

which is almost right but with an extra factor 20. If you are sensitive enough, you may notice that the extra factor 20 is the number of sample points. We'll get to this later.

3 Orthogonality of Complex Exponentials

In this section, we'll stay in the discrete world and let go all the continuous thing. Now we have the definition of Discrete Fourier Transform

$$F[m] = \sum_{k=0}^{N-1} f[k]e^{-2\pi i\frac{km}{N}} \tag{3.1}$$

We can see the complex exponentials $e^{2\pi i\frac{km}{N}}, k = 0, 1, ..., N-1$ as itself a vector

$$\omega = [1, e^{2\pi i\frac{1}{N}}, e^{2\pi i\frac{2}{N}}, ..., e^{2\pi i\frac{N-1}{N}}]$$

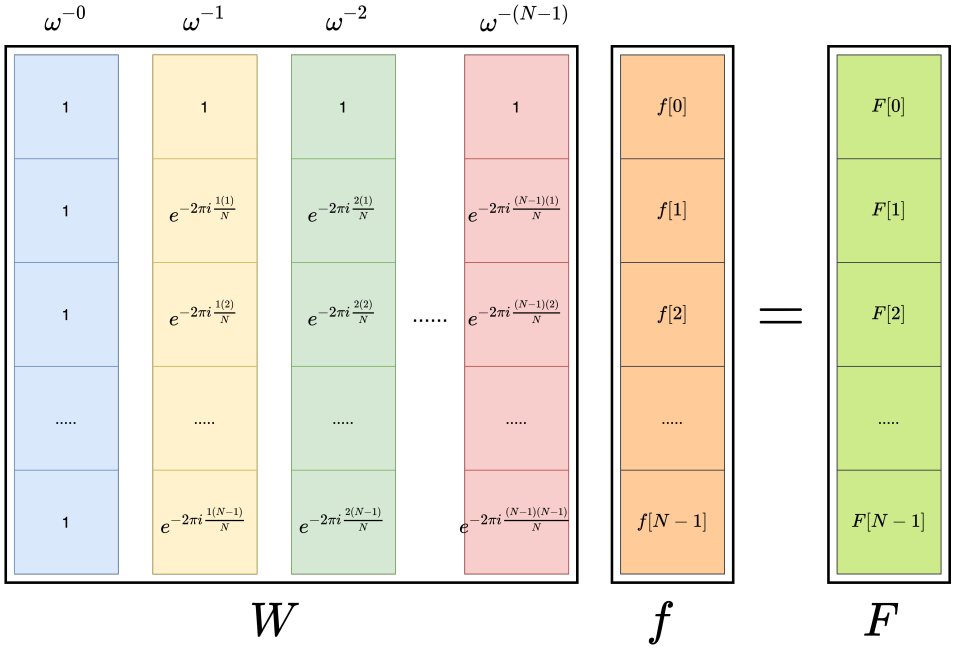
where ω[k] $= e^{2\pi i\frac{k}{N}}$. The power of ω is just each of its element powered.

$$\omega^m = [1, e^{2\pi i\frac{m(1)}{N}}, e^{2\pi i\frac{m(2)}{N}}, ..., e^{2\pi i\frac{m(N-1)}{N}}]$$

Equation (3.1) can be written as

$$F = \sum_{k=0}^{N-1} f[k]\omega^{-k} = Wf \tag{3.2}$$

as illustrated in the following picture, where W is a symmetric complex matrix, columns of which are exponentials of ω.



Besides symmetry, there is a very important property of the matrix W, that is, let me put it in extra large font, Orthogonality. To study the orthogonality of the matrix, let's study the orthogonality of any two columns, ω^l and ω^k, of W. Take the inner product of ω^l and ω^k

$$\begin{aligned} < \omega^l, \omega^k > &= \sum_{m=0}^{N-1} \omega^l[m] \cdot \overline{\omega^k[m]} \\ &= \sum_{m=0}^{N-1} e^{2\pi i \frac{ml}{N}} \cdot e^{-2\pi i \frac{mk}{N}} \\ &= \sum_{m=0}^{N-1} e^{2\pi i (l-k)\frac{m}{N}} \end{aligned}$$

When $(l-k)\%N = 0$, $(l-k)\frac{m}{N}$ is an integer, so $e^{2\pi i(l-k)\frac{m}{N}} = 1$ and $< \omega^l, \omega^k >= N$. When $(l-k)\%N \neq 0$, we have a geometric series

$$
\begin{aligned}
&\sum_{m=0}^{N-1} e^{2\pi i(l-k)\frac{m}{N}} \\
=&\frac{1-(e^{2\pi i(l-k)\frac{1}{N}})^N}{1-e^{2\pi i(l-k)\frac{1}{N}}} \\
=&\frac{1-e^{2\pi i(l-k)}}{1-e^{2\pi i(l-k)\frac{1}{N}}} \\
=&0
\end{aligned}
$$

Put it together, and since W has only N columns, we get

$$
< \omega^l, \omega^k >= \begin{cases} 0 & l \neq k \\ N & l = k \end{cases} \tag{3.3}
$$

We can see that W is a orthogonal matrix, or say, those complex exponential vectors are "orthogonal" to each other. However, it's not an orthonormal matrix since $< \omega^l, \omega^l >= ||\omega^l||_2^2 = N$ instead of 1. And that's why there was a factor 20 in the previous example where we had $N = 20$ samples.

4 Inverse DFT

Now I have a treat for you. Since we concluded that W is an orthogonal matrix, it means we can easily invert this matrix, reverse the DFT process and get back f from F, leading to inverse DFT.

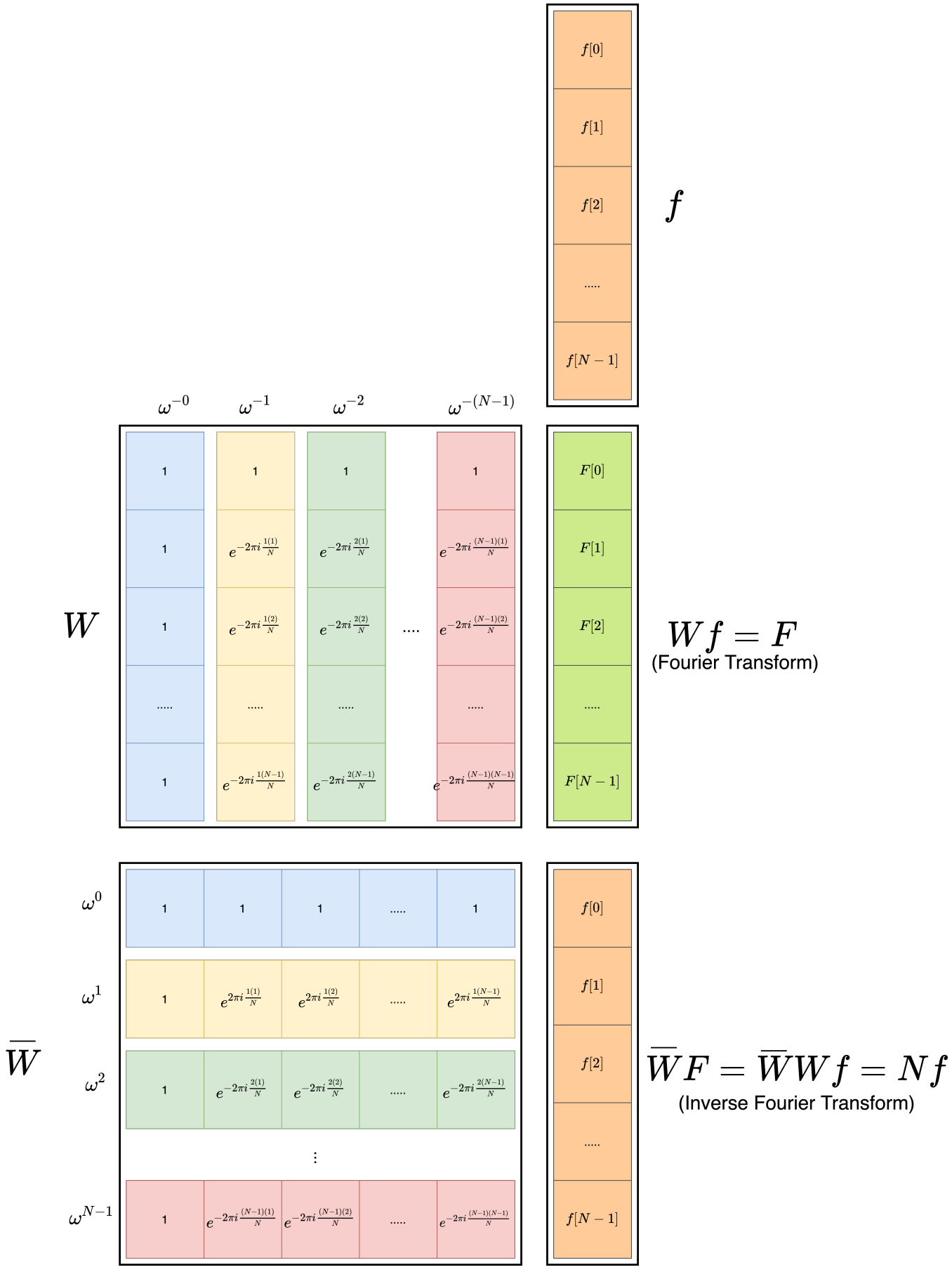
Since W is an orthogonal complex matrix with each column vector having norm $\sqrt{N}$, we can have $\overline{W^T}W = NI$. And because W is a symmetric matrix, we have $\overline{W}W = NI$. DFT written in a matrix multiplication form $F = Wf$ combined with $\overline{W}W = NI$ gives us $\overline{W}F = \overline{W}Wf = Nf$, a multiple of the original discrete signal f.

We can easily write down the formula for inverse DFT in matrix form

$$
f = \frac{1}{N}\overline{W}F \tag{4.1}
$$

as shown in the picture below. Actually, the processes of DFT and Inverse DFT are all about this picture, we don't need to remeber the formula once we have this picture in our head, so I think it's worth a whole page. But for the sake of completeness, let me write down the formula for inverse DFT and wrap up this note.

$$
\begin{aligned}
f &= \frac{1}{N}\overline{W}F \\
&= \frac{1}{N}\sum_{k=0}^{N-1} F[k]\omega^k \qquad (4.2) \\
f[m] &= \frac{1}{N}\sum_{k=0}^{N-1} F[k]e^{2\pi i\frac{mk}{N}} \qquad (4.3)
\end{aligned}
$$


f[0]
f[1]
f[2]
.....
f[N − 1]
f
ω^{−0}
ω^{−1}
ω^{−2}
ω^{−(N−1)}
W
1
e^{−2πi 1(1)/N}
e^{−2πi 2(1)/N}
e^{−2πi (N−1)(1)/N}
e^{−2πi 1(2)/N}
e^{−2πi 2(2)/N}
e^{−2πi (N−1)(2)/N}
e^{−2πi 1(N−1)/N}
e^{−2πi 2(N−1)/N}
e^{−2πi (N−1)(N−1)/N}
F[0]
F[1]
F[2]
F[N − 1]
Wf = F
(Fourier Transform)
W̄
ω^0
ω^1
ω^2
ω^{N−1}
e^{2πi 1(1)/N}
e^{2πi 2(1)/N}
e^{2πi 1(N−1)/N}
e^{2πi 2(N−1)/N}
e^{2πi (N−1)(1)/N}
e^{2πi (N−1)(2)/N}
e^{2πi (N−1)(N−1)/N}
W̄F = W̄Wf = Nf
(Inverse Fourier Transform)