1 The Discrete Fourier Transform

These notes describe the Discrete Fourier Transform (DFT), a computational technique especially useful for processing finite length signals in the frequency domain. In Chapter 5, we discussed the Discrete-Time Fourier Transform (DTFT) $X(e^{j\omega})$. The difficulty with using $X(e^{j\omega})$ in a computer is that the frequency ω is a continuous variable, and the computer can only work with discrete values. A natural approach to this difficulty would be to evaluate $X(e^{j\omega})$ every $\Delta\omega$ in frequency. If the values in frequency are close enough together, i.e., $\Delta\omega$ is small enough, it seems reasonable that this should represent all the important information in $X(e^{j\omega})$. This is exactly what the DFT does. However, we will see that we must be careful in applying the DFT, since sometimes this approach can have unexpected consequences.

An important decision in computing the DFT is how many frequencies to use in evaluating $X(e^{j\omega})$? If we use N frequencies, the most straightforward thing to do is to space them evenly in frequency over one period of $X(e^{j\omega})$, say for $0 \leq \omega < 2\pi$. The frequency interval between DFT values is then $\Delta\omega = 2\pi/N$ and the DFT is

$$X[k] = X(e^{j\omega})|_{\omega=(2\pi/N)k}. \tag{1}$$

Substituting the definition of the DTFT into Eq. (1) gives the following

$$\begin{aligned} X[k] &= \left(\sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \right)_{\omega=(2\pi/N)k} . && (2) \\ &= \sum_{n=-\infty}^{\infty} x[n]e^{-j(2\pi/N)kn} && (3) \end{aligned}$$

Given the N-point DFT values $X[k]$ from Eq. (3), the inverse DFT equation can be used to compute the time domain signal for those samples

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j(2\pi/N)kn}. \tag{4}$$

In fact, Eq. (4) and (3) should look familiar. They are almost the same as the discrete-time Fourier series from Chapter 3. This is not an accident. In computing the DFT values $X[k]$ every $\Delta\omega = 2\pi/N$ in frequency, we are in effect sampling $X(e^{j\omega})$. Sampling a signal in one domain means making the signal periodic in the other domain. In Chapter 7, we saw that sampling $x_c(t)$ in time generated a periodic spectrum in frequency. Here, the reverse is true, sampling in frequency will generate a periodic signal in time. The only difference between the DFT equations (Eqs. (3) and (4) above) and the DT Fourier series equations (Eqs. (3.94) and (3.95) in the book) is which equation has the $(1/N)$ term in front. For the DT Fourier series, the $(1/N)$ is in the analysis equation, while for the DFT the $(1/N)$ is in the synthesis equation. This discrepancy is a matter of historical accident and convention, but does not impact the main concepts relating the time and frequency domains here.

The most common application of the DFT is for finite-length signals. It is important to understand this relationship between sampling and periodicity for the case when $x[n]$

is an M-point finite-length signal, i.e., $x[n] = 0$ for $n < 0$ or $n > M-1$. Specifically, we need to understand when $X[k]$ contains all of the important information in the signal. The process of evaluating $X(e^{j\omega})$ every $\Delta\omega$ in frequency is in fact sampling, but in the frequency domain instead of the time domain. Recognizing this allows us to apply our understanding of sampling from Chapter 7 to develop some intuition about this problem. Recall that when we sample a continuous-time signal $x_c(t)$ every T seconds in the time domain, we make copies of its Fourier transform every $2\pi/T$ in frequency. In order to preserve all of the information in $x_c(t)$, we must make T small enough so that the copies of the Fourier transform do not overlap, or alias, in frequency. This also requires the assumption that the Fourier transform of $x(t)$ is bandlimited in frequency.

Understanding how the frequency domain sampling in computing the DFT effects the signal requires the mental agility to use duality to its fullest, and exchange time and frequency in the discussion above. Instead of sampling every T seconds, we are sampling every $\Delta\omega$ radians. Instead of sampling in time causing us to make copies of the CT Fourier transform every $2\pi/T$ in frequency, we are sampling in frequency, causing us to make copies of the signal $x[n]$ every $2\pi/\Delta\omega$ in time. Substituting $\Delta\omega = 2\pi/N$ into this equation shows that the copies of $x[n]$ are placed every $2\pi/2\pi/N = N$ samples in time. As expected from duality, sampling in one domain causes the signal to become periodic in the other domain. In order for this operation to preserve all the information in the signal, these copies of $x[n]$ must not overlap, which would be aliasing in time. To avoid this time domain aliasing, the signal $x[n]$ must be finite length, and at most N samples long from $0 \le n \le N-1$. This means that for an M-point signal, $X[k]$ will contain all of the information in the signal so long as the DFT size $N \ge M$. Let's see how this works in some examples.

Example 1: The DFT and Time Domain Aliasing Figure 1 shows a finite-length discrete-time signal $x[n]$with length $M = 12$. For this example, we will look at the time domain effect of computing the DFT with three different sizes. For the first case, we will use $N = 12$ to compute the DFT

$$X_1[k] = X(e^{j\omega})|_{\omega=(2\pi/12)k}.$$

Substituting the values of $X_1[k]$ into Eq. (4) gives the periodic time-domain signal $x_1[n]$ generated by the sampling every $2\pi/12$ in frequency. Figure 2 shows the resulting signal. Comparing Figs. 1 and 2 shows that the copies of $x[n]$ do not overlap, and so $X[k]$ contains all of the information in $x[n]$.

If we only compute $N = 10$ samples of $X(e^{j\omega})$, we expect to see time-domain aliasing in the signal. If we define

$$X_2[k] = X(e^{j\omega})|_{\omega=(2\pi/10)k},$$

then use these values in Eq. (4) to find the inverse DFT, we get a different periodic signal $x_2[n]$, shown in Figure 3. From this plot, it is clear that the expect aliasing has occured, information has been lost, and that we cannot recover $x[n]$ from $x_2[n]$.

Finally, a common technique is to compute more DFT samples than are needed. This technique is known as zero padding, since it is equivalent to adding a string of zeros onto

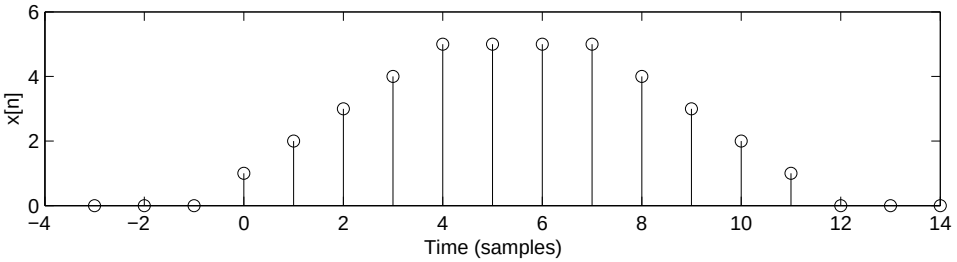

Figure 1: $M = 12$ point finite-length signal $x[n]$ for Example 1

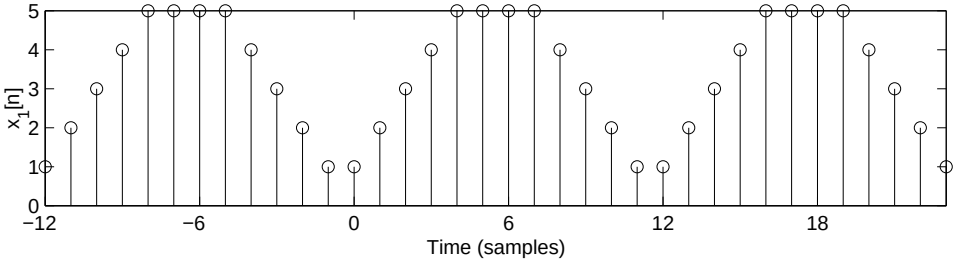

Figure 2: Periodic signal $x_1[n]$ generated by using an $N = 12$ point DFT

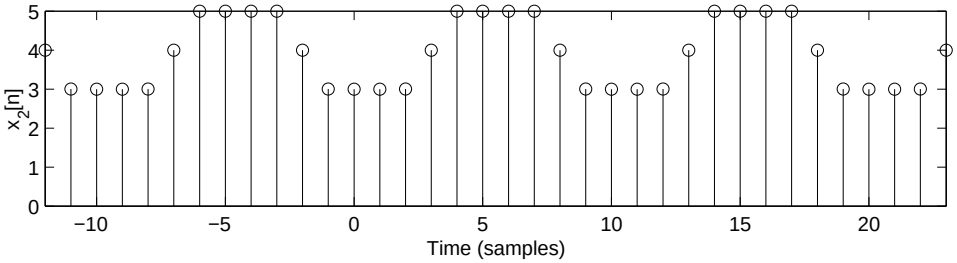

Figure 3: Periodic signal $x_2[n]$ generated by using an $N = 10$ point DFT

the end of $x[n]$, then pretending the signal is longer than it really is. For this example, we can use $N = 15$ to zero pad this signal, defining

$$X_3[k] = X(e^{j\omega})|_{\omega=(2\pi/15)k},$$

then again using Eq. (4) to find the resulting periodic time signal $x_3[n]$. As shown in Figure 4, $x_3[n]$ preserves the original signal $x[n]$, and even has space between the copies of $x[n]$ generated by the frequency domain sampling. In Chapter 7, we saw examples where if we sampled even faster than required by the Nyquist sampling theorem, the spectrum $X(e^{j\omega})$ had gaps with amplitude zero. The $N = 15$ case is the DFT dual of this phenomenom: by taking more samples in frequency than were immediately required, the periodic time signal has gaps between copies.

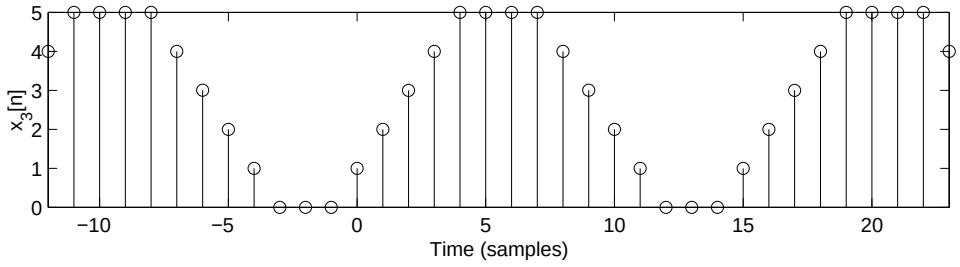

Figure 4: Periodic signal $x_3[n]$ generated by using an $N = 15$ point DFT

A common application of the DFT is to examine the spectrum of a finite length signal. In fact, you have done this many times in Matlab projects this semester, using the Matlab `fft`. The FFT, or Fast Fourier transform, is an efficient way to compute the DFT values, $X[k]$[1]. The Matlab command `X=fft(x,N)` computes the N-point DFT of the finite-length signal $x[n]$. The FFT turns out to be especially efficient when N is a power of 2, i.e., $N = 2^v$ for some integer v. Choosing a large value of N, or zero-padding as we saw in the example above, means that the samples of $X(e^{j\omega})$ in `X` are very close together, and can be plotted as a good approximation to the DTFT. This oversampling is often important to get a good picture of $X(e^{j\omega})$ - and sometimes a value of N large enough to avoid aliasing will provide a misleading picture of $X(e^{j\omega})$. The following example demonstrates why it is often a good idea to use a large N when using the DFT $X[k]$ to approximate the DTFT $X(e^{j\omega})$.

Example 2: Using the DFT to examine the spectrum $X(e^{j\omega})$ Figure 5 shows a rectangular pulse with length $M = 5$. Applying the time shift property of the DTFT to Example 5.3 in the book, the DTFT of this signal can be found as

$$X(e^{j\omega}) = e^{-j2\omega}\frac{\sin(5\omega/2)}{\sin(\omega/2)}, \qquad \text{for } |\omega| \leq \pi. \tag{5}$$

[1] An interesting historical footnote is that one of the modern co-inventors of the FFT, John W. Tukey, grew up in New Bedford, before a successful mathematical career at Princeton and Bell Labs.

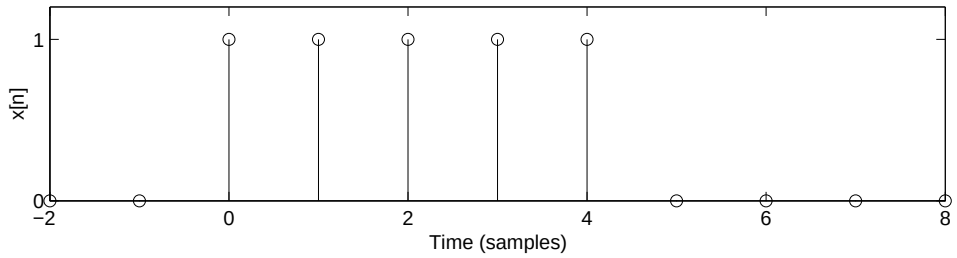

Figure 5: 5-point rectangular pulse $x[n]$ for Example 2

From our discussion of sampling above, we know that if the DFT has $N \geq 5$, there will be no time domain aliasing, and the samples $X[k]$ will preserve all of the information in $x[n]$. However, if $N = 5$, the values of $X[k]$ may not give a good sense of the underlying DT sinc for $X(e^{j\omega})$. Figure 6 shows the values obtained for $X[k]$ when $N = 5$ in Eq. (3).

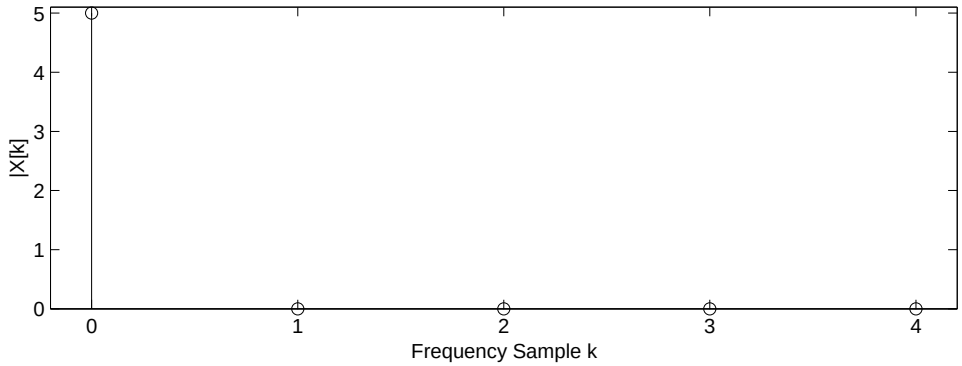

Figure 6: $N = 5$ point DFT for the signal in Fig. 5

At first glance, we might think this signal has an impulse in its spectrum at $k = 0$, and is zero for all the other frequencies. This would be incorrect, though. When $N = 5$, the DFT samples $X(e^{j\omega})$ right on the zero-crossings of $\sin((5\omega/2)/\sin(\omega/2)$. These crossing occur when $\sin(5\omega/2) = 0$, which is when $5\omega/2 = m\pi$ for a non-zero integer m. This gives us zeros in $X(e^{j\omega})$ for $\omega = 2\pi m/5$. (When $\omega = 0$, both numerator and denominator are zero, and we must use L'Hopital's rule to figure out that $X(e^{j\omega})|_{\omega=0} = 5$.) Figure 7 plots the DFT samples $X[k]$ overlaid with the DTFT $X(e^{j\omega})$, demonstrating how this happens. If we keep in mind the periodic copies of $x[n]$ caused by the sampling, this result also makes sense. If we repeat $x[n]$ every 5 samples, the resulting periodic signal will be a constant, which will have only one nonzero discrete-time Fourier series coefficient, at $k = 0$.

If we want a more informative picture of $X(e^{j\omega})$, we need to zero-pad $x[n]$ and compute more samples in frequency. If we increase N to 20, and define those signals as

$$X_2[k] = X(e^{j\omega})|_{\omega=(2\pi/20)k},$$

we obtain the samples shown in Figure 8, again with $X(e^{j\omega})$ overlaid. This plot gives a better sense of the underlying DT sinc function $X(e^{j\omega})$. Making the DFT size N even

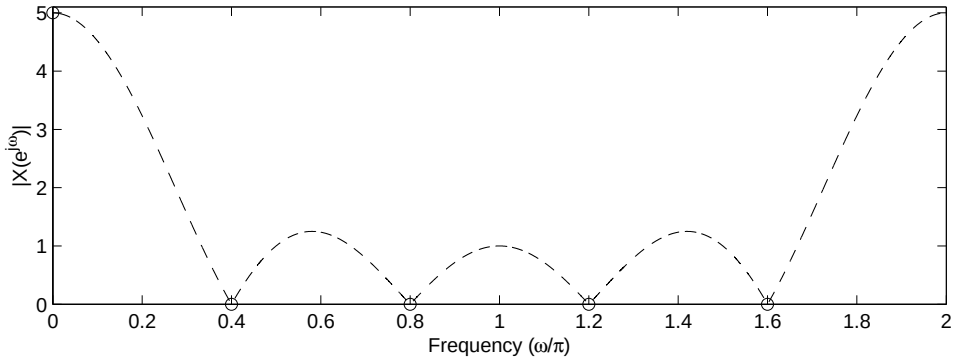

Figure 7: DTFT $X(e^{j\omega})$ with DFT samples $X[k]$ for $N = 5$ overlaid

.

larger will give still better pictures of $X(e^{j\omega})$. In Matlab projects, you found that using $N \geq 100$ will often produce very good approximations to $X(e^{j\omega})$ for relatively short signals like the rectangular pulse in Figure 5. For longer signals, the DFT size must be much larger. This now explains the very large `Nfft` you used for the speech signals in the third part of Matlab II. Those speech signals were over ten thousand samples long, so needed very large DFTs to avoid aliasing and to get a good view of the spectrum $X(e^{j\omega})$.

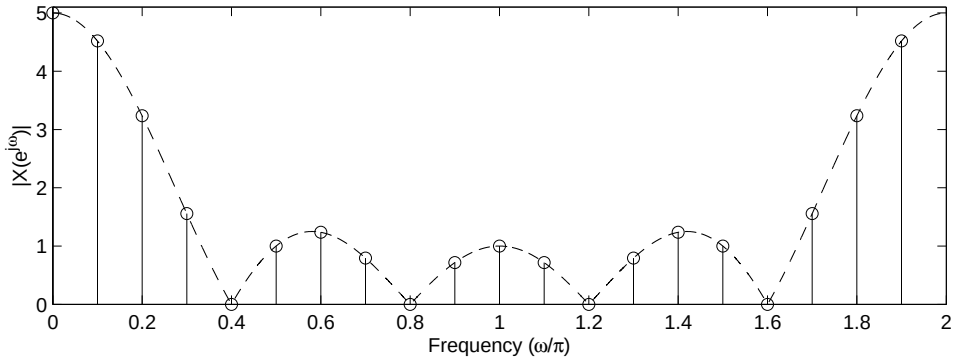

Figure 8: DTFT $X(e^{j\omega})$ with DFT samples $X_2[k]$ for $N = 20$ overlaid

.

In practice there are relatively few truly periodic DT signals of interest. However, there are many applications studying finite-length signals. In fact, every practical experiment or measurement produces a finite-length signal, since we can only record and sample for a finite amount of time. The underlying strategy in all of the examples using the DFT to process finite length signals is to model the finite length signal as though it were one period of a periodic signal with period N. So long as N is larger than the length of the periodic signal, we can always recover the finite-length signal by discarding all of the periods except $0 \leq n \leq N-1$. A good approach to understand what is happening to the signals in the time domain is to follow the three step procedure below:

1. When computing the N-point DFT of a finite-length signal $x[n]$, make a periodic

signal $\tilde{x}[n]$ by adding copies of $x[n]$ every N samples in time.

2. Modify $\tilde{x}[n]$ at each step of the problem or algorithm as appropriate for the operations given.

3. As the final step, truncate the $\tilde{x}[n]$ to obtain the new finite-length signal that results from the inverse DFT.

If we use these steps to remind us of the implied periodicity while working with the DFT, it will help us to explain some strange and unanticipated results that sometimes occur.