Lecture 11

DFT and FFT

11.1 Introduction

In the next set of lectures, we will cover discrete Fourier analysis, ie: decomposition of discrete-space (or discrete-time) signals into complex exponentials. Whereas earlier in the course we considered continuous-space signals that span all of space, next we will consider discrete-space signals that are finite in length. Importantly, this development will be critical for the computational solution of practical problems.

11.2 Definition of the DFT

The DFT of a length-M sequence $f[m]$ (for $m = 0, 1, \ldots, M-1$) is defined as:

$$\hat{f}[k] = \sum_{m=0}^{M-1} f[m] e^{-i2\pi \frac{mk}{M}} \tag{11.1}$$

Importantly, the DFT of a length-M sequence is itself a length-M sequence. Note that this definition resembles that of the continuous-time Fourier Transform reviewed earlier in the course, except the integral over all of $\mathbb{R}$ in the FT is replaced by a finite sum over M elements in the DFT.

Question: How many operations (multiplications) would it take to naively implement Eq. 11.1 above in order to calculate our length-M DFT?

Also, it is often helpful to express the DFT as a matrix-vector operation (where the input and output sequences are expressed as length-M vectors):

$$\hat{\boldsymbol{f}} = \boldsymbol{F}\boldsymbol{f} \tag{11.2}$$

where the $M \times M$ matrix $\boldsymbol{F}$ is called the *DFT matrix* and has entries defined as follows:

$$\boldsymbol{F}_{k,m} = e^{-i2\pi \frac{mk}{M}} \text{ , for } k = 0, \ldots, M-1 \text{ and } m = 0, \ldots, M-1 \tag{11.3}$$

Analogously to the DFT, the inverse DFT (iDFT) is defined as:

$$f[m] = \frac{1}{M} \sum_{k=0}^{M-1} \hat{f}[k] e^{i2\pi \frac{mk}{M}} \tag{11.4}$$

And, not surprinsingly, the iDFT can also be expressed as a matrix-vector operation:

$$\boldsymbol{f} = \frac{1}{M} \boldsymbol{F}^H \hat{\boldsymbol{f}} \tag{11.5}$$

where ^{H} denotes hermitian transpose, and the iDFT matrix $\frac{1}{M}\boldsymbol{F}^H$ happens to also be the matrix inverse of the DFT matrix $\boldsymbol{F}$ described above, ie: $\frac{1}{M}\boldsymbol{F}^H = \boldsymbol{F}^{-1}$.

Importantly, the iDFT matrix can be seen (from the definition in Eq. 11.4) to have entries:

$$\frac{1}{M}\boldsymbol{F}^H_{m,k} = \frac{1}{M} e^{i2\pi \frac{mk}{M}} \text{ , for } k = 0, \ldots, M-1 \text{ and } m = 0, \ldots, M-1 \tag{11.6}$$

or in other words, $\boldsymbol{F}^{-1} = \frac{1}{M}\boldsymbol{F}^H$, ie: the inverse of the DFT matrix $\boldsymbol{F}$ is (aside from the normalization factor $\frac{1}{M}$) its own conjugate transpose denoted by the symbol ^{H}. What this implies is that the DFT matrix $\boldsymbol{F}$ is an $M \times M$ matrix where the column vectors are all orthogonal to each other (such that the inner product of two columns is zero, $\sum_{m=0}^{M-1} \boldsymbol{F}_{m,k}\overline{\boldsymbol{F}_{m,l}} = 0$ for $k \neq l$). This orthogonality property of the DFT matrix comes in handy in a wide variety of scenarios.

11.3 DFT and FFT

As we have seen above, the DFT is a transform, defined by an equation (Eq. 11.1 above). In contrast, the FFT is a fast algorithm (in reality, a family of fast algorithms) for calculating the DFT. In this course, we will not dig deep into the specifics of how the FFT manages to be so fast. Suffice to say that it exploits extensive redundancies in the DFT matrix. Using FFT algorithms, the DFT of a length-N sequence (which naïvely would require on the order of M^2 operations to compute) will require on the order of $M \log M$ operations. *It is hard to overstate the implications of this speedup, which has revolutionized many areas of science and technology.*[1]

11.4 Why is the DFT not Normalized?

As you can see from the definition of the DFT and iDFT, the iDFT includes a factor $\frac{1}{M}$ that is not present in the DFT. This seeming lack of symmetry is needed because the standard definition of the DFT is not normalized. If we were to include a factor $\frac{1}{\sqrt{M}}$ in the DFT, then the same factor would appear in the iDFT, and the iDFT matrix

[1] http://www.uta.edu/faculty/rcli/TopTen/topten.pdf

would be exactly the conjugate transpose (Hermitian) of the DFT matrix. Although this normalized version would arguably be more elegant mathematically, and lead to cleaner expressions, in this course we will use the standard non-normalized definition of DFT and iDFT, in order to be consistent with most texts and mathematical software packages.

11.5 Running DFT (FFT) in Practice

Consider this line of Matlab code: `fk = fftshift(fft(ifftshift(fx)))`, where `fx` is the original spatial domain sequence and `fk` is the DFT. This line of code has three components:

- `ifftshift` swaps the left and right portions of a sequence, placing the central element at the beginning
- `fft` performs a DFT using a fast (FFT) algorithm
- `fftshift` swaps the left and right portions of a sequence, placing the first element in the center

This approach may raise several questions:

1. Why do we need the `ifftshift` and `fftshift` commands? (ie: why not just run `fk = fft(fx)`?) Based on the definition of the DFT, the 'low frequencies' will appear at the beginning and at the end of the resulting sequence: low positive frequencies at the beginning, and low negative frequencies at the end. These `ifftshift` and `fftshift` commands before and after the DFT enable a more visually intuitive version of the DFT, where the low frequencies are in the center of the sequence, and the high frequencies are at the ends. Similarly, these commands lead to implicitly placing the spatial center of the sequence in the center, rather than near the beginning and end of the sequence.
2. What is the difference between `ifftshift` and `fftshift`? For even-length sequences, they have exactly the same effect and are interchangeable. However, for odd-length sequences, `fftshift` will undo the effect of `ifftshift`, whereas `ifftshift` will not undo its own effect.