EE20N: Structure and Interpretation of Systems and Signals **Fall 2013**

Lecture 18: October 31

Lecturer: Prof. Kannan Ramchandran *Scribe: Daniel Gerber*

18.1 Announcements

- Midterm 2 next Thursday in class – one cheat sheet (like midterm 1)
- No homework next week

18.2 Agenda

- Geometry of DFT (Matrix View)
- Sampling View of DFT
- Circular Convolution

18.3 Recap

Recall that the equations for the DFT are:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{ik\omega_0 n}, n = 0, 1, ..., N-1$$

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-ik\omega_0 n}, k = 0, 1, ..., N-1$$

where N is the length of the sequence and $\omega_0 = \frac{2\pi}{N}$.

$$\underline{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \xrightarrow{DFT} \underline{X} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

$$\underline{X} = F\underline{x}$$

$$\underline{x} = F^{-1}\underline{X}$$

where $\underline{X}$ and $\underline{x}$ are N-length vectors and F is an $N \times N$ matrix. The 4×4 matrix F is:

$$F = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{pmatrix}$$

where $W = e^{-i\frac{2\pi}{N}}$.

18.4 Matrix view of DFT

We can observe several properties of F. First we note that F is a symmetric matrix. This is easily seen in its structure, as that F is square, and each W term grows with the row index in the same way it grows with column index.

Another important property of F is that it is a nearly unitary matrix. A unitary matrix is defined as a matrix in which the inverse is equal to the conjugate transpose. In other words, for unitary matrix U,

$$U^{-1} = U^H,$$

where H deontes the conjugate transpose. Note that F is nearly unitary. The reason for this is that it is different by a scaling factor of $\frac{1}{N}$, and thus

$$F^{-1} = \frac{1}{N}F^H.$$

However, we generally ignore the scaling factor in our definition of F being a unitary matrix. For example, if we say $G = \frac{1}{\sqrt{N}}F$, we can now say that G is unitary, as that

$$G^{-1} = G^H$$
$$(\frac{1}{\sqrt{N}})^{-1}F^{-1} = \frac{1}{\sqrt{N}}F^H$$
$$F^{-1} = \frac{1}{N}F^H.$$

So in other words, nobody really cares about the $\frac{1}{N}$.

Another neat property of unitary matricies is that their columns and rows are orthogonal. This means that if you take the inner (dot) product of one column with any other column, you end up with zero (same applies for rows). For example, refer to the 4x4 F matrix shown above. The first two columns are

$$[1, 1, 1, 1]$$
$$[1, e^{-i\frac{2\pi}{4}}, e^{-i2\frac{2\pi}{4}}, e^{-i2\frac{2\pi}{4}}].$$

The dot product of these columns gives us

$$(1)(1) + (1)(e^{-i\frac{\pi}{2}}) + (1)(e^{-i\pi}) + (1)(e^{-i\frac{3\pi}{2}}) = 0.$$

A similar result happens for taking the dot product between any two columns or rows. It is also pretty cool how each term in the dot product ends up being a complex number that is $\pi/2$ apart in angle. In general, the terms in the dot product of two columns of an arbitrary F matrix of size $N \times N$ will end up being $2\pi/N$ apart in angle.

Yet another piece of linear algebra intuition: taking the DFT geometrically rotates the coordinate system in N-dimensional space. Generally, multiplying a $N \times 1$ vector by any $N \times N$ matrix will rotate and scale it in N-dimensional space. However, in the case of the F matrix, each entry has a magnitude of 1, and so we only see a rotation.

Rotating the coordinate system can often be very useful in data compression. For example, if our data is the set of 2×1 vectors

$$(4, 3), (7, 8), (19, 22), (20, 20),$$

we can rotate it using the 2×2 F matrix

$$F = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

In the new coordinate system, our data set becomes

$$(7, 1), (15, -1), (41, -3), (40, 0).$$

Since the second entry in each data point is very small, we can approximate it as zero, allowing us to store only the first entry of each data point. This type of transformation happens all the time in data compression.

18.5 Sampling View of DFT

The DFT can be thought of a sampled form of the DTFT. Recall our equations

$$X_d(\omega) = \sum_{n=0}^{N-1} x[n] e^{-i\omega n}$$

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi}{N}nk}.$$

From these equations, we can see that to convert from the DTFT frequency domain representation to the DFT respresentation, we can use

$$X[k] = X_d(\omega)|_{\omega=\frac{2\pi}{N}k, k=0,1,\ldots N-1}$$

18.6 Circular Convolution

Normally if we have an input signal $x[n]$ and pass it through a system with impulse response $h[n]$, we get the output $y[n] = (x * h)[n]$. If we have the frequency domain representation of the input and the frequency response of the system, we can use $Y_d = H_d X_d$. Note that here the frequency domain signals are related to the time domain through the DTFT.

Since multiplication is much simpler than convolution, it is generally the desirable means with which to solve filtering problems. However, computers can't handle the DTFT, and must rely on the DFT instead. We can multiply the DFT components $Y[k] = X[k]H[k]$ and still get the desired frequency domain solution. However, if we wish to solve such a problem in time domain, we must use the circular convolution instead of the normal convolution. In other words, multiplication in DFT frequency domain transforms to circular convolution in time domain. The formula for circular convolution is

$$y[n] = \sum_{m=0}^{N-1} h[m] x[(n-m)\%N],$$

where % represents the modulus operator. More on circular convolution next lecture.