Lab 4 - Discrete Fourier Transform

4.1 Introduction

This laboratory will introduce the Discrete Fourier Transform (DFT) and the associated sampling and windowing effects.

In previous laboratories, we have used the Discrete-Time Fourier Transform (DTFT) extensively for analyzing signals and linear time-invariant systems.

$$\text{(DTFT)} \qquad X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n} \tag{4.1}$$

$$\text{(inverse DTFT)} \qquad x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X\left(e^{j\omega}\right) e^{j\omega n} d\omega. \tag{4.2}$$

While the DTFT is very useful **analytically**, it usually cannot be exactly evaluated on a computer because (4.1) requires an infinite sum and (4.2) requires the evaluation of an integral.

The discrete Fourier transform (DFT) is a sampled version of the DTFT, hence it is better suited for numerical evaluation on computers.

$$\text{(DFT)} \qquad X_N[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \tag{4.3}$$

$$\text{(inverse DFT)} \qquad x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_N[k] e^{j2\pi kn/N} \tag{4.4}$$

Here $X_N[k]$ is an N point DFT of $x\ [n]$. Note that $X_N[k]$ is a function of a discrete integer k, where k ranges from 0 to $N-1$.

In the following sections, we will study the derivation of the DFT from the DTFT, and several DFT implementations.

4.2 Deriving the DFT from the DTFT

4.2.1 Truncating the Time-domain Signal

The DTFT usually cannot be computed exactly because the sum in (4.1) is infinite. However, the DTFT may be approximately computed by truncating the sum to a finite window. Let $w[n]$ be a rectangular window of length N:

$$w[n] = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases}. \tag{4.5}$$

Then we may define a truncated signal to be

$$x_{\mathrm{tr}}[n] = w[n]x[n] \tag{4.6}$$

The DTFT of $x_{\mathrm{tr}}[n]$ is given by:

$$X_{\mathrm{tr}}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_{\mathrm{tr}}[n]e^{-j\omega n} = \sum_{n=0}^{N-1} x[n]\, e^{-j\omega n} \tag{4.7}$$

We would like to compute $X(e^{j\omega})$, but the truncation window distorts the desired frequency characteristics; $X(e^{j\omega})$ and $X_{\mathrm{tr}}(e^{j\omega})$ are generally not equal. To understand the relation between these two DTFT's, we need to convolve in the frequency domain:

$$X_{\mathrm{tr}}\left(e^{j\omega}\right) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X\left(e^{j\sigma}\right) W(e^{j(\omega-\sigma)})d\sigma \tag{4.8}$$

where $W\left(e^{j\omega}\right)$ is the DTFT of $w[n]$. Eq. (4.8) is the periodic convolution of $X\left(e^{j\omega}\right)$ and $W\left(e^{j\omega}\right)$. Hence the true DTFT, $X\left(e^{j\omega}\right)$, is smoothed via convolution with $W\left(e^{j\omega}\right)$ to produce the truncated DTFT, $X_{\mathrm{tr}}\left(e^{j\omega}\right)$.

We can calculate $W\left(e^{j\omega}\right)$:

$$W\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} w[n]\, e^{-j\omega n} \quad = \sum_{n=0}^{N-1} e^{-j\omega n}$$
$$= \begin{cases} \dfrac{1-e^{-j\omega N}}{1-e^{-j\omega}}, & \text{for } \omega \neq 0, \pm 2\pi, \ldots \\ N, & \text{for } \omega = 0, \pm 2\pi, \ldots \end{cases} \tag{4.9}$$

For $\omega \neq 0, \pm 2\pi, \ldots$, we have:

$$W(e^{j\omega}) = \frac{e^{-j\omega N/2}}{e^{-j\omega/2}} \frac{e^{j\omega N/2} - e^{-j\omega N/2}}{e^{j\omega/2} - e^{-j\omega/2}} = e^{-j\omega(N-1)/2} \frac{sin(\omega N/2)}{sin(\omega/2)} \tag{4.10}$$

Notice that the magnitude of this function is similar to sinc $(\omega N/2)$ except that it is periodic in ω with period 2π.

4.2.2 Frequency Sampling

Eq. (4.7) contains a summation over a finite number of terms. However, we can only evaluate (4.7) for a finite set of frequencies, ω. We must sample in the frequency domain to compute the DTFT on a computer. We can pick any set of frequency points at which to evaluate (4.7), but it is particularly useful to uniformly sample ω at N points, in the range $[0,2\pi)$. If we substitute

$$\omega = 2\pi k/N \tag{4.11}$$

for $k = 0,1, \ldots (N-1)$ in (7.7), we find that

$$X_{\text{tr}}(e^{j\omega}) \mid_{\omega=\frac{2\pi k}{N}} = \sum_{n=0}^{N-1} x[n]e^{-j\omega n} \mid_{\omega=\frac{2\pi k}{N}} = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} = X_N[k] \tag{4.12}$$

In short, the DFT values result from sampling the DTFT of the truncated signal.

$$X_N[k] = X_{\text{tr}}(e^{j2\pi k/N}) \tag{4.13}$$

4.2.3 Windowing Effects

Download DTFT.m for the following section.

We will next investigate the effect of windowing when computing the DFT of the signal $x[n] = \cos\left(\frac{\pi}{4}n\right)$ truncated with a window of size $N = 20$.

1. In the same figure, plot the phase and magnitude of $W(e^{j\omega})$, using equations (4.9) and (4.10).
2. Determine an expression for $X(e^{j\omega})$ (the DTFT of the non-truncated signal).
3. Truncate the signal $x[n]$ using a window of size $N = 20$ and then use DTFT.m to compute $X_{\text{tr}}(e^{j\omega})$.
   Make sure that the plot contains a least 512 points.
   HINT: Use the command [X ,w] = DTFT(x,512) .
4. Plot the magnitude of $X_{\text{tr}}(e^{j\omega})$.

INLAB REPORT:

1. Submit the plot of the phase and magnitude of $W(e^{j\omega})$.
2. Submit the analytical expression for $X(e^{j\omega})$.
3. Submit the magnitude plot of $X_{\text{tr}}(e^{j\omega})$.
4. Describe the difference between $\left| X_{\text{tr}}(e^{j\omega})\right|$ and $\left|X(e^{j\omega})\right|$. What is the reason for this difference?
5. Comment on the effects of using a different length of the window $w(n)$.

4.3 The Discrete Fourier Transform

4.3.1 Computing the DFT

We will now develop our own DFT functions to help our understanding of how the DFT comes from the DTFT. Write your own Matlab function to implement the DFT of equation (4.3). Use the syntax

X = DFTsum(x)

where **x** is an N point vector containing the values $x(0), \ldots, x(N-1)$ and **X** is the corresponding DFT. Your routine should implement the DFT exactly as specified by (4.3) using for-loops for n and k, and compute the exponentials as they appear. Note: In Matlab, "j" may be computed with the command

j=sqrt(-1) . If you use $j = \sqrt{-1}$, remember not to use *j* as an index in your for-loop.

Test your routine DFTsum by computing $X_N[k]$ for each of the following cases:

1. $x[n] = \delta[n]$ for $N = 10$.
2. $x[n] = 1$ for $N = 10$.
3. $x[n] = e^{j2\pi n/10}$ for $N = 10$.
4. $x[n] = \cos(2\pi n/10)$ for $N = 10$.

Plot the magnitude of each of the DFT's. In addition, derive simple closed-form analytical expressions for the DFT (not the DTFT!) of each signal.

INLAB REPORT:

1. Submit a listing of your code for **DFTsum**.
2. Submit the magnitude plots.
3. Submit the corresponding analytical expressions.

Write a second Matlab function for computing the inverse DFT of (4.4). Use the Syntax

x = IDFTsum(X)

where **X** is the N point vector containing the DFT and **x** is the corresponding time-domain signal. Use **IDFTsum** to invert each of the DFT's computed in the previous problem. Plot the magnitudes of the inverted DFT's, and verify that those time-domain signals match the original ones. Use **abs(x)** to eliminate any imaginary parts which roundoff error may produce.

INLAB REPORT:

1. Submit the listing of your code for **IDFTsum**.
2. Submit the four time-domain IDFT plots.

4.3.2 Matrix Representation of the DFT

The DFT of (4.3) can be implemented as a matrix-vector product. To see this, consider the equation

$$\boldsymbol{X} = \mathbf{A}\boldsymbol{x} \tag{4.14}$$

where $\mathbf{A}$ is an $N \times N$ matrix, and both $\boldsymbol{X}$ and x are $N \times 1$ column vectors. This matrix product is equivalent to the summation

$$\boldsymbol{X}_k = \sum_{n=1}^{N} \mathbf{A}_{kn}\, \boldsymbol{x}_n \tag{4.15}$$

where $\mathbf{A}_{kn}$ is the matrix element in the k^{th} row and n^{th} column of **A** . By comparing (4.3) and (4.15) we see that for the DFT,

$$\mathbf{A}_{kn} = e^{-j2\pi(k-1)(n-1)/N} \tag{4.16}$$

The $-1'$s are in the exponent because Matlab indices start at 1, not 0. For this section, we need to:

- Write a Matlab function **A = DFTmatrix(N)** that returns the *N* x *N* DFT matrix **A**.
  NOTE: Remember that the symbol ‘ $*$ ’ is used for matrix multiplication in Matlab, and that ‘ **.'** ’ performs a simple transpose on a vector or matrix. An apostrophe without the period is a conjugate transpose.
- Use the matrix $\mathbf{A}$ to compute the DFT of the following signals. Confirm that the results are the

same as in the previous section.

a. $x[n] = \delta[n]$ for $N = 10$
b. $x[n] = 1$ for $N = 10$
c. $x[n] = e^{j2\pi n/N}$ for $N = 10$

INLAB REPORT:

1. Print out the matrix $\mathbf{A}$ for $N = 5$.
2. Hand in the three magnitude plots of the DFT's.
3. How many multiplies are required to compute an N point DFT using the matrix method? (Consider a multiply as the multiplication of either complex or real numbers.)

As with the DFT, the inverse DFT may also be represented as a matrix-vector product.

$$\boldsymbol{x} = \mathbf{B}\boldsymbol{X} \tag{4.18}$$

For this section,

1. Write an analytical expression for the elements of the inverse DFT matrix $\mathbf{B}$, using the form of (4.16).
2. Write a Matlab function **B = IDFTmatrix(N)** that returns the *N* x *N* inverse DFT matrix $\mathbf{B}$.
3. Compute the matrices $\mathbf{A}$ and $\mathbf{B}$ for $N = 5$. Then compute the matrix product $\mathbf{C} = \mathbf{BA}$.

INLAB REPORT:

1. Hand in your analytical expression for the elements of $\mathbf{B}$.
2. Print out the matrix $\mathbf{B}$ for $N = 5$.
3. Print out the elements of $\mathbf{C} = \mathbf{BA}$. What form does $\mathbf{C}$ have? Why does it have this form?

4.3.3 Computation Time Comparison

Read cputime.pdf for help on the **cputime** function.

Although the operations performed by **DFTsum** are mathematically identical to a matrix product, the computation times for these two DFT's in Matlab are quite different. (This is despite the fact that the computational complexity of two procedures is of the same order!) This exercise will underscore why you should try to avoid using **for loops** in Matlab, and wherever possible, try to formulate your computations using matrix/vector products.

To see this, do the following:

1. Compute the signal $x[n] = \cos\left(\frac{2\pi n}{10}\right)$ for $N = 512$.
2. Compute the matrix $\mathbf{A}$ for $N = 512$.
3. Compare the computation time of X = DFTsum(x) with a matrix implementation **X = A*x** by using the **cputime** function before and after the program execution. Do not include the computation of $\mathbf{A}$ in your timing calculations.

INLAB REPORT: Report the CPU time required for each of the two implementations. Which method is faster? Which method requires less storage?