Problem 7.1

Note: Matlab plots for this problem are due at the beginning of class on Monday, 11/13!

The purpose of this exercise is to begin exploring properties of the Discrete Fourier Transform (DFT) and the Inverse Discrete Fourier Transform (IDFT). The DFT is defined in Equation 1 below:

\[
DFT : \quad X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{N}} \quad k = 0, \ldots, N - 1
\]  

(1)

Note that the DFT corresponds to \(N\) samples of the discrete-time Fourier transform (DTFT), equally-spaced between 0 and \(2\pi\).\(^1\) Equation 1 defines an \(N\)-point DFT; it requires \(N\) values of the input signal to compute it.

Equation 2 below defines the inverse DFT. It takes \(N\) spectral samples (stored in \(X[k]\)) and inverse transforms them to obtain a time-domain signal \(x[n]\).

\[
IDFT : \quad x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j\frac{2\pi kn}{N}} \quad n = 0, \ldots, N - 1
\]  

(2)

The Matlab functions `fft` and `ifft` implement the DFT and IDFT, respectively. Note that FFT stands for Fast Fourier Transform. This term refers to efficient implementations of the DFT. Both `fft` and `ifft` take 2 arguments: the signal to be transformed (or inverse-transformed) and \(N\), the number of points to use. For example

\[
X4 = \text{fft}(x, 4)
\]

computes the 4-point DFT of the signal stored in the vector \(x\) and returns it in the output vector \(X4\). Similarly, \(y = \text{ifft}(Y, 4)\) computes the IDFT of the vector \(Y\) and stores it in a 4-point vector \(y\).

\(^1\)To see this compare Equation 1 to the definition of the DTFT.
Problem 7.1 – continued

(a) Consider the signal \( x[n] = u[n] - u[n - 4] \) (a 4-point sequence).

(i) Use Matlab to compute the 3-point, 4-point, 7-point, 16-point, and 128-point DFT’s of \( x[n] \).
    Store your results in the vectors \( X_3, X_4, X_7, X_{16}, X_{128} \).

(ii) Compute a vector of frequencies to use for plotting each of the DFT’s. Note that the DFT
    computes samples of the Fourier transform equally-spaced between \( 0 \) and \( 2\pi \) with a sampling
    interval of \( \frac{2\pi}{N} \). Thus the frequency vector for the 3-point DFT is \( \omega = [0 \quad \frac{2\pi}{3} \quad \frac{4\pi}{3}] \).
    It is easy to show that the Matlab command to generate a frequency vector for an \( N \)-point is
    \[ \text{omega} = 2\pi \times (0:N-1)/N; \]

(iii) Plot the magnitudes of the five DFT’s of the signal \( x[n] \) on the same graph using different
    linestyles/colors. Use the \texttt{stem} command for plotting \( X_3, X_4, X_7, X_{16} \) and use \texttt{plot}
    to display \( X_{128} \).

(iv) Repeat part (iii) using the phase of the five DFT’s.

(v) To think about before Monday: Are your results what you expected? Can you calculate the
    DTFT of the signal \( x[n] \) by hand? Would your analytical calculation match what you’ve
    seen with Matlab?

(b) Take the inverse transform of each of your DFT’s and store them in the vectors \( iX_3, iX_4, iX_7,
    iX_{16}, iX_{128} \). Each of these inverse transforms will contain real and imaginary parts. Verify
    that all of the imaginary parts are roughly equal to \( 10^{-16} \) for this example. Since the imaginary
    parts are on the order of Matlab’s numerical precision, you may ignore them (by taking
    \texttt{real}(iX_3), etc.). Plot the 5 inverse transforms. Do you recover your original signal \( x[n] \) in all
    cases?

(c) Define a vector \( \tilde{W} \) to be \( X_4 + X_4 \) (the sum of two 4-point DFT’s). Now take the inverse transform of
    \( \tilde{W} \), i.e., let \( w = \text{ifft}(\tilde{W}, 4) \). Plot this new signal \( w[n] \) using \texttt{stem}. Could you have predicted this
    result? How?

(d) Recall that for the discrete-time Fourier transform, multiplication in the frequency domain corre-
    sponds to convolution in the time domain. In this exercise, we will explore whether multiplication
    of samples of the Fourier transform (i.e., the DFT) will correspond to a convolution operation in
    the time domain.

(i) Compute the following in Matlab (using the DFT’s you calculated for part (a)):
    \[
    Y_4 = X_4 \times X_4; \quad y_4 = \text{real}(\text{ifft}(Y_4, 4)) ; \\
    Y_7 = X_7 \times X_7; \quad y_7 = \text{real}(\text{ifft}(Y_7, 7)) ; \\
    Y_{16} = X_{16} \times X_{16}; \quad y_{16} = \text{real}(\text{ifft}(Y_{16}, 16)) ; \\
    Y_{128} = X_{128} \times X_{128}; \quad y_{128} = \text{real}(\text{ifft}(Y_{128}, 128)) ;
    \]
    In other words, you will multiply each DFT by itself and then compute the appropriate
    inverse transform. The \texttt{real} operation is in there so that tiny imaginary parts (which are
    numerical “noise”) are ignored. Plot each of the resulting sequences using \texttt{stem}.

(b) To think about before Monday: Does multiplying DFT’s correspond to a convolution of se-
    quences in the time domain? What would the convolution: \( x[n] \ast x[n] \) look like?