

Homework No. 7

Fall 2000

Issued: Wednesday, November 8, 2000**Due:** Wednesday, November 15, 2000Reading in *Signals and Systems: Continuous and Discrete*

Week of 11/13/00 — pp. 363-366

Chapter 4, Section 4-13

Chapter 10, Sections 10-1 through 10-3 and

Sections 10-5 through 10-8

Reading in Proakis (Optional text)

Week of 11/13/00 — Chapter 5, Sections 5-1 through 5-5

Problem 7.1Note: *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:

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

Note that the DFT corresponds to N samples of the discrete-time Fourier transform (DTFT), equally-spaced between 0 and 2π .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 $x[n]$.

$$\text{IDFT : } \quad x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{+j \frac{2\pi kn}{N}} \quad n = 0, \dots, N-1 \quad (2)$$

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 .

¹To 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 `X3`, `X4`, `X7`, `X16`, `X128`.
 - (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 2π 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

$$\text{omega} = 2*\text{pi}*(0:\text{N}-1)/\text{N};$$

- (iii) Plot the magnitudes of the five DFT's of the signal $x[n]$ on the same graph using different linestyles/colors. Use the `stem` command for plotting `X3`, `X4`, `X7`, `X16` and use `plot` to display `X128`.
 - (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 `iX3`, `iX4`, `iX7`, `iX16`, `iX128`. 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 `real(iX3)`, etc.). Plot the 5 inverse transforms. Do you recover your original signal $x[n]$ in all cases?
- (c) Define a vector `W` to be `X4+X4` (the sum of two 4-point DFT's). Now take the inverse transform of `W`, i.e., let `w=ifft(W,4)`. Plot this new signal $w[n]$ using `stem`. Could you have predicted this result? How?
- (d) Recall that for the discrete-time Fourier transform, multiplication in the frequency domain corresponds 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)):

```
Y4=X4.*X4; y4=real(ifft(Y4,4));
Y7=X7.*X7; y7=real(ifft(Y7,7));
Y16=X4.*X4; y16=real(ifft(Y16,16));
Y128=X128.*X128; y128=real(ifft(Y128,128));
```

In other words, you will multiply each DFT by itself and then compute the appropriate inverse transform. The `real` operation is in there so that tiny imaginary parts (which are numerical “noise”) are ignored. Plot each of the resulting sequences using `stem`.

- (b) To think about before Monday: Does multiplying DFT's correspond to a convolution of sequences in the time domain? What would the convolution: $x[n] * x[n]$ look like?