Lab 5
October 17, 2016
Fast Fourier Transform

INSTRUCTIONS:
All lab submissions include a written report and source code in the form of an m-file. The report contains all plots, images, and figures specified within the lab. All figures should be labeled appropriately. Answers to questions given in the lab document should be answered in the written report. ***The written report must be in PDF format***. Submissions are done electronically through **my.ECE**.

1 Background

The FFT was named one of the top ten algorithms of the 20th century. Ideas for a fast algorithm date back to Gauss in the early 19th century. However, credit is often given to Cooley and Tukey, who developed the algorithm in 1965 to detect nuclear tests. It reduces the computational complexity of DFT from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. The FFT has many applications in signal processing as well as solving differential equations such as Maxwell's equations.

2 Derivation of FFT

We learned that a naive implementation of DFT yields an $\mathcal{O}(N^2)$ algorithm where as the fast Fourier transform (FFT) yields an $\mathcal{O}(N \log N)$ algorithm. This is achieved through recursively dividing the input signal $x(n)$ into parts. To see this explicitly, let us first write the definition of the DFT.

$$\text{DFT}_N\{x\} = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N} \tag{1}$$

Assuming that $N = 2^l$ where l is a positive integer greater than 1, (1) can be divided into even and odd parts $x_e(n)$ and $x_o(n)$.

$$x_e(m) = x(2m), \qquad m = 0, \ldots, N/2 - 1 \tag{2}$$
$$x_o(m) = x(2m + 1), \qquad m = 0, \ldots, N/2 - 1 \tag{3}$$

Which enables us to re-write the DFT as

$$\text{DFT}_N\{x\} = \sum_{m=0}^{N/2-1} x_e(m)e^{-i2\pi k(2m)/N} + \sum_{m=0}^{N/2-1} x_o(m)e^{-i2\pi k(2m+1)/N} \tag{4}$$

$$= \sum_{m=0}^{N/2-1} x_e(m)e^{-i2\pi km/(N/2)} + e^{-i2\pi k/N} \sum_{m=0}^{N/2-1} x_o(m)e^{-i2\pi km/(N/2)} \tag{5}$$

$$= \text{DFT}_{N/2}\{x_e\} + e^{-i2\pi k/N}\text{DFT}_{N/2}\{x_o\} \tag{6}$$

The relationship we have found can be expressed as

$$\text{DFT}_N\{x\} = \text{DFT}_{N/2}\{x_e\} + e^{-i2\pi k/N}\text{DFT}_{N/2}\{x_o\} \tag{7}$$

where an N-point DFT has been written as a sum of two $N/2$-point DFT. Hence, the complexity has been reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N^2/2)$. The exponential term outside the DFT is known as a "twiddle factor". Its multiplication with the result of the $\text{DFT}_(N/2)\{x_o\}$ is $\mathcal{O}(N)$ and does not dominate the complexity of the overall operation.

If we apply (7), the overall cost is roughly four $N/4$-point DFTs which results in $\mathcal{O}(N^2/4)$. Since $N = 2^l$, the number of times this division can be performed is $l-1$ times. This results in an $\mathcal{O}(N \log N)$ algorithm. The length of the shortest DFT that can be taken from this algorithm is $N = 2$. A 2-point DFT is easy to compute. It is simply

$$\text{DFT}_2\{x\} = \sum_{n=0}^{1} x(n)e^{-i2\pi kn/N} = \{x(0) + x(1), x(0) - x(1)\} \tag{8}$$

3 Implementation of FFT

Assume that $\mathbf{x}$ is the input signal to the FFT and $\mathbf{X}$ is the output, both of which are length N. The FFT algorithm can be implemented recursively. First, check if $\mathbf{x}$ has a length greater than 2. If so, check whether or not the signal is divisible by 2. If not, perform a 2-point DFT, as specified in (8).

Given that N is greater than 2, check and see that N is evenly divisible by 2. This can be done using **mod**. If it is evenly divisible by 2, then the vector form of (7) can be written as

$$\mathbf{X} = [\mathbf{X}_e, \mathbf{X}_e] + e^{-i2\pi(0:\text{N-}1)/N}. * [\mathbf{X}_o, \mathbf{X}_o] \tag{9}$$

where $\mathbf{X}$ is the output of the FFT and

$$\mathbf{X}_e = \text{DFT}_{N/2}\{\mathbf{x}(1{:}2{:}\text{end} - 1)\} \tag{10}$$

$$\mathbf{X}_o = \text{DFT}_{N/2}\{\mathbf{x}(2{:}2{:}\text{end})\} \tag{11}$$

are DFTs of length $N/2$. Equations (9), (10), and (11) use MATLAB colon notation. If N is not evenly divisible by 2, then one simply takes the DFT of $\mathbf{x}$, which is $\mathcal{O}(N^2)$. However, by this time, $\mathbf{x}$ has already been decimated a few times so the

DFT is being applied to a short signal.

Report Item: Write a function called *myFFT* that uses recursion to implement the FFT. Compare your results with **fft**.