ELL319: Digital Signal Processing, II Semester 2020-21

Lab Experiment 4

DFT, IDFT and FFT

April 12, 2021

1 Aim

In this experiment, you will learn to compute the N point Discrete Fourier Transform, and Inverse Discrete Fourier Transform for a sequence and to plot magnitude and phase spectrum. Using DFT and IDFT, you will then perform linear and circular convolution of two sequences. You will also learn about Fast Fourier Transform and its usefulness in finding the frequency spectrum of signals.

2 Theory

2.1 Discrete Fourier Transform

The Discrete Fourier Transform is used to find the amplitude and phase spectrum of a discrete time sequence $x[n]$ of length N:

$$X(k) = DFT(x[n]) := \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \quad \forall k \in \{0, 1, ...N-1\} \tag{1}$$

An intuitive explanation of how equation 1 finds the frequency component of a signal can be found by breaking the $e^{-j2\pi kn/N}$ term into $cos(2\pi nk/N) - jsin(2\pi kn/N)$. Using this, and rearranging, we get:

$$X(k) = \sum_{n=0}^{N-1} x[n] \cdot cos(2\pi kn/N) - j\sum_{n=0}^{N-1} x[n] \cdot sin(2\pi kn/N) \tag{2}$$

The real part of $X(k)$ can be seen as the correlation between x and a cosine of frequency k. The imaginary part can similarly be seen as the correlation between x and a sine of frequency k. Correlation between two signals, by its very definition, can be seen as a metric of the similarity between the two signals. Two sine waves of same frequency will have a large correlation, while waves of different frequencies will have a lower correlation. So, a signal x having the frequency k in its spectrum, will have a high correlation with a cosine or a sine signal of that frequency, and the magnitude of the real or complex part of $X(k)$ will be large. Since we mainly care about the frequencies present in the signal, we can look at the magnitude $|X(k)|^2 = Re(X(k))^2 + Im(X(k))^2$ to tell us which frequencies

are present in the signal.
To get the actual frequenies (in Hertz) in the signal, we need to know the sampling frequency f_s used to sample $x[n]$. Frequencies are given by:

$$f = f_s \cdot k/N \tag{3}$$

2.2 Inverse DFT

Given a frequency spectrum $X(k)$, IDFT is the inverse of the operation given in equation 1:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X(k) \cdot e^{j2\pi kn/N} \quad \forall n \in 0, 1, ...N-1 \tag{4}$$

This yields the sample domain signal $x[n]$ back.

2.3 Convolution of Two Signals using DFT and IDFT

An interesting, and quite useful fact about Fourier Transform is the property that convolution in time (or sample) domain is equivalent to multiplication in frequency domain. Which means that given two signals $x[n]$ and $h[n]$, the signal $y[n] = x[n] * h[n]$ has the Fourier transform $Y(k) = X(k) \cdot H(k)$, $X(k)$ and $H(k)$ being the fourier transforms of $x[n]$ and $h[n]$ respectively.
Computing the convolution of two signals $x[n]$ of length L and $h[n]$ of length K, can be achieved by doing the following:

- Make both signals of size $N = L + K - 1$, by appending 0's at the end.
- Compute N-point DFT of $x[n]$ and $h[n]$
- Compute $Y(k) = X(k) \cdot H(k)$ where $\cdot$ is element-wise multiplication
- Compute the N point IDFT of $Y(k)$ to get $y[n]$

To understand why this method is useful, take a look at the following simple system:

$$\begin{tikzcd}
x[n] \arrow[r] & \boxed{H_1(k)} \arrow[r, "y_1{[n]}"] & \boxed{H_2(k)} \arrow[r, "y_2{[n]}"] & \cdots \arrow[r, "y_{m-1}{[n]}"] & \boxed{H_m(k)} \arrow[r] & y[n]
\end{tikzcd}$$

Here, $H_i(k)$ is the Impulse response of the i^{th} system, and its output $y_i[n] = y_{i-1}[n] * h_i[n]$, where $*$ is convolution in sample domain. Computing $y_1[n], y_2[n] \ldots y_{m-1}[n]$ is too tedious. However, using the property about Fourier Transform, we can easily find the output of the complete system using $y[n] = \text{IDFT}(X(k) \cdot \prod_{i=1}^{m} H_i(k))$

2.4 Fast Fourier Transform

A typical implementation of DFT is quadratic in the size of the signal, i.e. time complexity of DFT is $O(n^2)$. This is very slow for large signals, like audio recordings, which can have hundreds of thousands, or even millions of samples. To circumvent this issue, a class of algorithms, collectively called Fast Fourier Transforms, were developed by researchers in the 20^th century. These algorithms typically use the divide and conquer strategy to compute Fourier Transforms of smaller and smaller segments of signal, and combine them to compute the overall Fourier transform of the complete signal.
The simplest, and most common algorithm used for FFT is the radix-2 Decimation In

Time FFT, and it assumes the signal size N to be a power of 2. It then goes on to recursively compute the FFT of the two halves of the signal.
The N point DFT of a discrete time signal is given by:

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \quad \forall k \in \{0, 1, ...N-1\} \tag{5}$$

Radix-2 DIT first computes the DFTs of the even-indexed inputs ($x_{2m} = x_0, x_2, \ldots, x_{N-2}$) and of the odd-indexed inputs ($x_{2m+1} = x_1, x_3, \ldots, x_{N-1}$), and then combines those two results to produce the DFT of the whole sequence. The Radix-2 DIT algorithm rearranges the DFT of the function x_n into two parts: a sum over the even-numbered indices $n = 2m$ and a sum over the odd-numbered indices $n = 2m + 1$:

$$X_k \;=\; \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi j}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi j}{N}(2m+1)k} \tag{6}$$

One can factor a common multiplier $e^{-\frac{2\pi j}{N}k}$ out of the second sum, as shown in the equation below. It is then clear that the two sums are the DFT of the even-indexed part x_{2m} and the DFT of odd-indexed part x_{2m+1} of the function x_n. Denote the DFT of the Even-indexed inputs x_{2m} by E_k and the DFT of the Odd-indexed inputs x_{2m+1} by O_k and we obtain:

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi j}{N/2}mk}}_{\text{DFT of even}-\text{indexed part of } x_n} + e^{-\frac{2\pi j}{N}k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi j}{N/2}mk}}_{\text{DFT of odd}-\text{indexed part of } x_n} = E_k + e^{-\frac{2\pi j}{N}k} O_k. \tag{7}$$

Thanks to the periodicity of the complex exponential, $X_{k+\frac{N}{2}}$ is also obtained from E_k and O_k:

$$\begin{aligned}
X_{k+\frac{N}{2}} &= \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi j}{N/2}m(k+\frac{N}{2})} + e^{-\frac{2\pi j}{N}(k+\frac{N}{2})} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi j}{N/2}m(k+\frac{N}{2})} \\
&= \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi j}{N/2}mk} e^{-2\pi mi} + e^{-\frac{2\pi j}{N}k} e^{-\pi i} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi j}{N/2}mk} e^{-2\pi mi} \\
&= \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi j}{N/2}mk} - e^{-\frac{2\pi j}{N}k} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi j}{N/2}mk} \\
&= E_k - e^{-\frac{2\pi j}{N}k} O_k
\end{aligned} \tag{8}$$

We can rewrite X_k as:

$$\begin{aligned}
X_k &= E_k + e^{-\frac{2\pi j}{N}k} O_k \\
X_{k+\frac{N}{2}} &= E_k - e^{-\frac{2\pi j}{N}k} O_k
\end{aligned} \tag{9}$$

A simple pseudo-code for the above algorithm would look like:

Algorithm 1 FFT

```
1: $X_0, ..., X_{N-1} \leftarrow fft(x, N, 2)$ :            DFT of (x0, x1, x2, ..., x(N-1))
2: $ifN = 1then$ :
3:     $X_0 \leftarrow x_0$                                   trivial size-1 DFT base case
4: $else$ :
5:     $X_0, ..., X_{N/2-1} \leftarrow fft(x, N/2, 2)$        DFT of $(x0, x2, x4, ...)$
6:     $X_{N/2}, ..., X_{N-1} \leftarrow fft(x+1, N/2, 2)$    DFT of $(x1, x3, x5, ...)$
7:     for k = 0 to N/2-1                                combine DFTs of two halves into full DFT
8:         $t \leftarrow X_k$
9:         $X_k \leftarrow t + e^{(-2\pi jk/N)} X_{k+N/2}$
10:        $X_{k+N/2} \leftarrow t - e^{(-2\pi jk/N)} X_{k+N/2}$
11:    end for
12: end if
```

3 Problem Statement

Deadline: 6 May 2021

3.1 Part A: DFT

Write a C function which takes in an array denoting $x[n]$, and returns its complex valued DFT. You may want to use a struct for complex numbers and complex arithmetic, to store data about $x[n]$ and $X[k]$. Plot the frequency magnitude and phase spectra of $X(k)$.

3.2 Part B: IDFT

Write a C function which takes in an array denoting $X(k)$, and returns its complex valued IDFT. Verify its correctness by using function from Previous part (**Part A**) and comparing output, with original signal.

3.3 Part C: Convolution

Using code developed in **Part A** and **Part B**, implement the convolution operation in time domain. Verify using sample $x[n]$ and $h[n]$.

3.4 Part D: FFT

Write a C function which computes the FFT of a sample domain signal $x[n]$.

3.5 Part E: Execution time for

Compare the running time of this FFT function in **Part D** with the DFT function developed in **Part A**.