Assignment-1

Harsh Kabra - EE18BTECH11019

Download all codes from

https://github.com/harsh006/dsp/codes

and latex file from:

https://github.com/harsh006/dsp

1 Problem

1.1. Let

$$x(n) = \left\{\underset{\uparrow}{1}, 2, 3, 4, 2, 1\right\} \quad (1.1.1)$$

$$y(n) + \frac{1}{2}y(n-1) = x(n) + x(n-2) \quad (1.1.2)$$

1.2. Compute

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \quad (1.2.1)$$

and $H(k)$ using h(n).

1.3. Compute

$$Y(k) = X(k)H(k) \quad (1.3.1)$$

2 Solution

2.1. Impulse response h(n), given difference equation is:

$$h(n) + \frac{1}{2}h(n-1) = \delta(n) + \delta(n-2) \quad (2.1.1)$$

2.2.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, 2..., N-1 \quad (2.2.1)$$

Let $W_N = e^{-j2\pi/N}$
We can express X as Matrix Multiplication of DFT Matrix and x.

$$X = \left[W_N^{ij}\right]_{N\times N} x, \quad i, j = 0, 1, \dots, N-1 \quad (2.2.2)$$

2.3. In the given problem, we have N = 6

$$\implies W_6 = e^{-j2\pi/6} = \frac{1}{2} - \frac{\sqrt{3}}{2}j \quad (2.3.1)$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \end{bmatrix} = \begin{bmatrix} W_6^0 & W_6^0 & W_6^0 & W_6^0 & W_6^0 & W_6^0 \\ W_6^0 & W_6^1 & W_6^2 & W_6^3 & W_6^4 & W_6^5 \\ W_6^0 & W_6^2 & W_6^4 & W_6^6 & W_6^8 & W_6^{10} \\ W_6^0 & W_6^3 & W_6^6 & W_6^9 & W_6^{12} & W_6^{15} \\ W_6^0 & W_6^4 & W_6^8 & W_6^{12} & W_6^{16} & W_6^{20} \\ W_6^0 & W_6^5 & W_6^{10} & W_6^{15} & W_6^{20} & W_6^{25} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 2 \\ 1 \end{bmatrix} \quad (2.3.2)$$

$$\implies \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \end{bmatrix} = \begin{bmatrix} 13 \\ -4 - \sqrt{3}j \\ 1 \\ -1 \\ 1 \\ -4 + \sqrt{3}j \end{bmatrix} \quad (2.3.3)$$

2.4. Similarly, we have
Solving for h(n)
we get,

$$h(n) = \left\{\underset{\uparrow}{1}, -0.5, 1.25, -0.625, 0.3125, -0.15625\right\} \quad (2.4.1)$$

We are interested only in first 6 terms, so h(n)

$$\begin{bmatrix} H(0) \\ H(1) \\ H(2) \\ H(3) \\ H(4) \\ H(5) \end{bmatrix} = \begin{bmatrix} W_6^0 & W_6^0 & W_6^0 & W_6^0 & W_6^0 & W_6^0 \\ W_6^0 & W_6^1 & W_6^2 & W_6^3 & W_6^4 & W_6^5 \\ W_6^0 & W_6^2 & W_6^4 & W_6^6 & W_6^8 & W_6^{10} \\ W_6^0 & W_6^3 & W_6^6 & W_6^9 & W_6^{12} & W_6^{15} \\ W_6^0 & W_6^4 & W_6^8 & W_6^{12} & W_6^{16} & W_6^{20} \\ W_6^0 & W_6^5 & W_6^{10} & W_6^{15} & W_6^{20} & W_6^{25} \end{bmatrix} \begin{bmatrix} 1 \\ -0.5 \\ 1.25 \\ -0.625 \\ 0.3125 \\ -0.15625 \end{bmatrix} \quad (2.4.2)$$

$$\implies \begin{bmatrix} H(0) \\ H(1) \\ H(2) \\ H(3) \\ H(4) \\ H(5) \end{bmatrix} = \begin{bmatrix} 1.28125 \\ 0.51625 - 0.5142j \\ -0.07813 + 1.1096j \\ 3.84375 \\ -0.07183 - 1.1096j \\ 0.51625 + 0.5142j \end{bmatrix} \quad (2.4.3)$$

2.5. We can find Y using,

$$Y(k) = X(k)H(k) \quad (2.5.1)$$

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \\ Y(5) \end{bmatrix} = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \end{bmatrix} \times \begin{bmatrix} H(0) \\ H(1) \\ H(2) \\ H(3) \\ H(4) \\ H(5) \end{bmatrix} \tag{2.5.2}$$

$$\implies \begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \\ Y(5) \end{bmatrix} = \begin{bmatrix} 16.65625 \\ -2.95312 + 1.16372j \\ -0.07813 + 1.1096j \\ -3.84375 \\ -0.07813 - 1.1096j \\ -2.95312 - 1.16372j \end{bmatrix} \tag{2.5.3}$$

2.6. The following code computes Y and generates magnitude and phase plots of X, H, Y

https://github.com/harsh006/dsp/tree/master/code/ee18btech11019_dsp1.py

2.7. The following plots are obtained

2.8. Lets now look at one of the property of W_N and how can it be used to reduce time complexity of computation.

$$W_N^2 = W_{N/2} \tag{2.8.1}$$

2.9. F_N is the N-point DFT Matrix.
Using the property of Complex Exponentials we can express F_N in terms of $F_{N/2}$

$$F_N = \begin{bmatrix} I_{N/2} & D_{N/2} \\ I_{N/2} & -D_{N/2} \end{bmatrix} \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} P_N \tag{2.9.1}$$

For N = 6

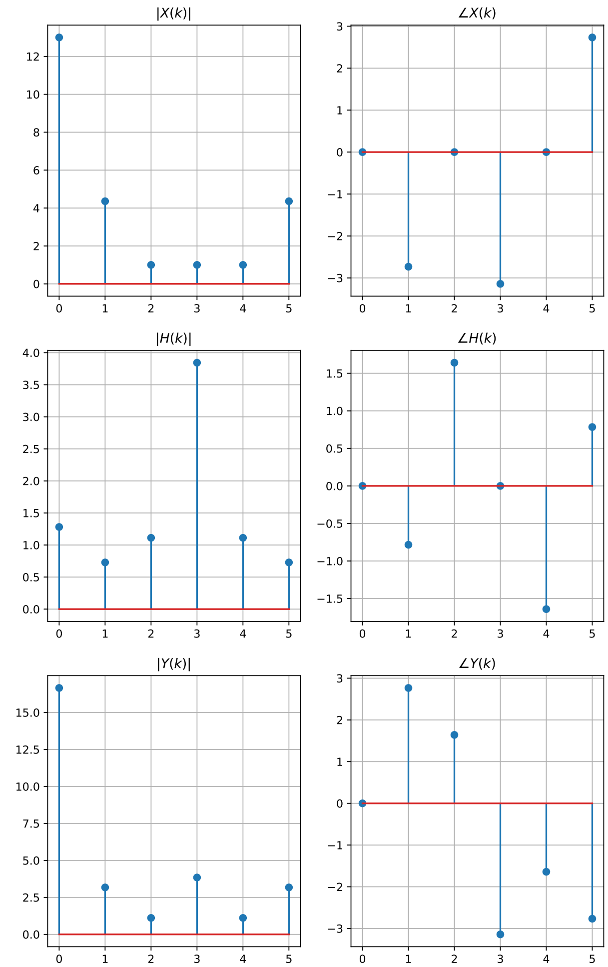
$$\implies F_6 = \begin{bmatrix} I_3 & D_3 \\ I_3 & -D_3 \end{bmatrix} \begin{bmatrix} F_3 & 0 \\ 0 & F_3 \end{bmatrix} P_6 \tag{2.9.2}$$

where I_3 is the 3x3 identity matrix

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9.3}$$

$$D_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & W_3^1 & 0 \\ 0 & 0 & W_3^2 \end{bmatrix} \tag{2.9.4}$$

$$P_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9.5}$$



$$\implies P_6 \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(1) \\ x(3) \\ x(5) \end{bmatrix} \tag{2.9.6}$$

Let

$$\begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \end{bmatrix} = F_{N/2} \begin{bmatrix} x(0) \\ x(2) \\ x(4) \end{bmatrix} \tag{2.9.7}$$

$$\begin{bmatrix} X_2(0) \\ X_2(1) \\ X_2(2) \end{bmatrix} = F_{N/2} \begin{bmatrix} x(1) \\ x(3) \\ x(5) \end{bmatrix} \tag{2.9.8}$$

be the N/2 point DFTs.

2.10. By replacing the above results in the equation $X = F_N x$, we get

$$\begin{bmatrix} X(0)\\ X(1)\\ X(2)\\ X(3)\\ X(4)\\ X(5) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & W_6^0 & 0 & 0\\ 0 & 1 & 0 & 0 & W_6^1 & 0\\ 0 & 0 & 1 & 0 & 0 & W_6^2\\ 1 & 0 & 0 & -W_6^0 & 0 & 0\\ 0 & 1 & 0 & 0 & -W_6^1 & 0\\ 0 & 0 & 1 & 0 & 0 & -W_6^2 \end{bmatrix} \begin{bmatrix} X_1(0)\\ X_1(1)\\ X_1(2)\\ X_2(0)\\ X_2(1)\\ X_2(2) \end{bmatrix} \tag{2.10.1}$$

2.11. Using the above method we have broken down an N-point DFT into 2 N/2-point DFTs

$$\begin{bmatrix} X(0)\\ X(1)\\ X(2) \end{bmatrix} = \begin{bmatrix} X_1(0)\\ X_1(1)\\ X_1(2) \end{bmatrix} + \begin{bmatrix} W_6^0 & 0 & 0\\ 0 & W_6^1 & 0\\ 0 & 0 & W_6^2 \end{bmatrix} \begin{bmatrix} X_2(0)\\ X_2(1)\\ X_2(2) \end{bmatrix} \tag{2.11.1}$$

$$\begin{bmatrix} X(3)\\ X(4)\\ X(5) \end{bmatrix} = \begin{bmatrix} X_1(0)\\ X_1(1)\\ X_1(2) \end{bmatrix} - \begin{bmatrix} W_6^0 & 0 & 0\\ 0 & W_6^1 & 0\\ 0 & 0 & W_6^2 \end{bmatrix} \begin{bmatrix} X_2(0)\\ X_2(1)\\ X_2(2) \end{bmatrix} \tag{2.11.2}$$

By doing this recursively,we can reduce our time complexity from O(N^2) to O(NlogN)

Now, say

$$N = 2^3 \tag{2.11.3}$$

Recursively breaking down,

$$F_8 = \begin{bmatrix} I_4 & D_4\\ I_4 & -D_4 \end{bmatrix} \begin{bmatrix} F_4 & 0\\ 0 & F_4 \end{bmatrix} P_8 \tag{2.11.4}$$

$$F_4 = \begin{bmatrix} I_2 & D_2\\ I_2 & -D_2 \end{bmatrix} \begin{bmatrix} F_2 & 0\\ 0 & F_2 \end{bmatrix} P_4 \tag{2.11.5}$$

F_2 is a base case

$$F_2 \begin{bmatrix} x\\ y \end{bmatrix} = \begin{bmatrix} 1 & 1\\ 1 & -1 \end{bmatrix} \begin{bmatrix} x\\ y \end{bmatrix} = \begin{bmatrix} x+y\\ x-y \end{bmatrix} \tag{2.11.6}$$

2.12. Let x(n) = $\left\{ \underset{\uparrow}{1}, 2, 3, 4, 2, 1, 2, 3 \right\}$

2.13. Solving 8-point FFTs into 4-point FFTs

$$\begin{bmatrix} X(0)\\ X(1)\\ X(2)\\ X(3) \end{bmatrix} = \begin{bmatrix} X_1(0)\\ X_1(1)\\ X_1(2)\\ X_1(3) \end{bmatrix} + \begin{bmatrix} W_8^0 & 0 & 0 & 0\\ 0 & W_8^1 & 0 & 0\\ 0 & 0 & W_8^2 & 0\\ 0 & 0 & 0 & W_8^3 \end{bmatrix} \begin{bmatrix} X_2(0)\\ X_2(1)\\ X_2(2)\\ X_2(3) \end{bmatrix} \tag{2.13.1}$$

$$\begin{bmatrix} X(4)\\ X(5)\\ X(6)\\ X(7) \end{bmatrix} = \begin{bmatrix} X_1(0)\\ X_1(1)\\ X_1(2)\\ X_1(3) \end{bmatrix} - \begin{bmatrix} W_8^0 & 0 & 0 & 0\\ 0 & W_8^1 & 0 & 0\\ 0 & 0 & W_8^2 & 0\\ 0 & 0 & 0 & W_8^3 \end{bmatrix} \begin{bmatrix} X_2(0)\\ X_2(1)\\ X_2(2)\\ X_2(3) \end{bmatrix} \tag{2.13.2}$$

4-point FFTs into 2-point FFTs

$$\begin{bmatrix} X_1(0)\\ X_1(1) \end{bmatrix} = \begin{bmatrix} X_3(0)\\ X_3(1) \end{bmatrix} + \begin{bmatrix} W_4^0 & 0\\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_4(0)\\ X_4(1) \end{bmatrix} \tag{2.13.3}$$

$$\begin{bmatrix} X_1(2)\\ X_1(3) \end{bmatrix} = \begin{bmatrix} X_3(0)\\ X_3(1) \end{bmatrix} - \begin{bmatrix} W_4^0 & 0\\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_4(0)\\ X_4(1) \end{bmatrix} \tag{2.13.4}$$

$$\begin{bmatrix} X_2(0)\\ X_2(1) \end{bmatrix} = \begin{bmatrix} X_5(0)\\ X_5(1) \end{bmatrix} + \begin{bmatrix} W_4^0 & 0\\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_6(0)\\ X_6(1) \end{bmatrix} \tag{2.13.5}$$

$$\begin{bmatrix} X_2(2)\\ X_2(3) \end{bmatrix} = \begin{bmatrix} X_5(0)\\ X_5(1) \end{bmatrix} - \begin{bmatrix} W_4^0 & 0\\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_6(0)\\ X_6(1) \end{bmatrix} \tag{2.13.6}$$

$$P_8 \begin{bmatrix} x(0)\\ x(1)\\ x(2)\\ x(3)\\ x(4)\\ x(5)\\ x(6)\\ x(7) \end{bmatrix} = \begin{bmatrix} x(0)\\ x(2)\\ x(4)\\ x(6)\\ x(1)\\ x(3)\\ x(5)\\ x(7) \end{bmatrix} \tag{2.13.7}$$

$$P_4 \begin{bmatrix} x(0)\\ x(2)\\ x(4)\\ x(6) \end{bmatrix} = \begin{bmatrix} x(0)\\ x(4)\\ x(2)\\ x(6) \end{bmatrix} \tag{2.13.8}$$

$$P_4 \begin{bmatrix} x(1)\\ x(3)\\ x(5)\\ x(7) \end{bmatrix} = \begin{bmatrix} x(1)\\ x(5)\\ x(3)\\ x(7) \end{bmatrix} \tag{2.13.9}$$

Therefore,

$$\begin{bmatrix} X_3(0)\\ X_3(1) \end{bmatrix} = F_2 \begin{bmatrix} x(0)\\ x(4) \end{bmatrix} \tag{2.13.10}$$

$$\begin{bmatrix} X_4(0)\\ X_4(1) \end{bmatrix} = F_2 \begin{bmatrix} x(2)\\ x(6) \end{bmatrix} \tag{2.13.11}$$

$$\begin{bmatrix} X_5(0)\\ X_5(1) \end{bmatrix} = F_2 \begin{bmatrix} x(1)\\ x(5) \end{bmatrix} \tag{2.13.12}$$

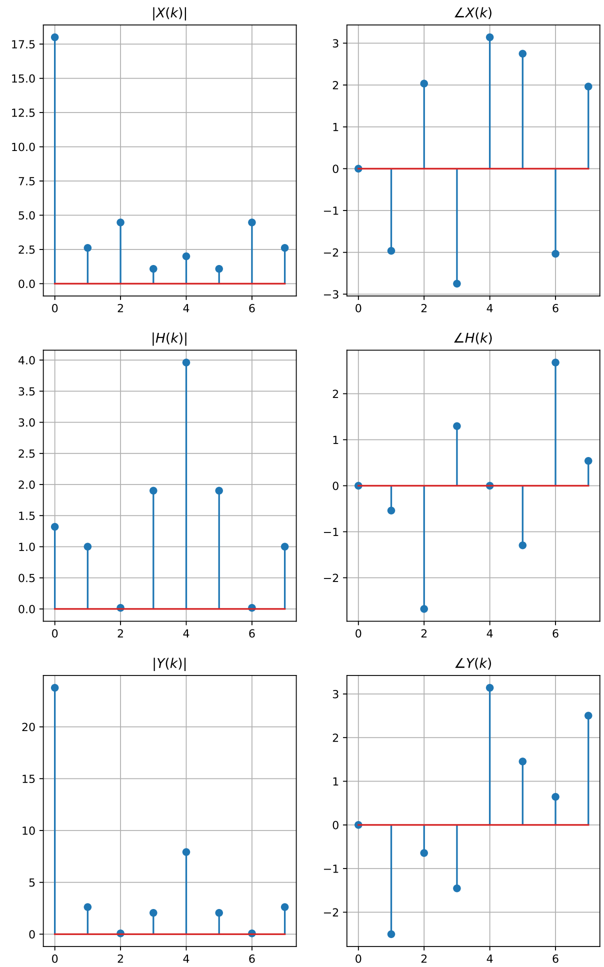
$$\begin{bmatrix} X_6(0)\\ X_6(1) \end{bmatrix} = F_2 \begin{bmatrix} x(3)\\ x(7) \end{bmatrix} \tag{2.13.13}$$

Now, backtraking gives us final X.

2.14. The below code computes Y, for above mentioned x(n) and generates magnitude and phase plots of X, H, Y

```
https://github.com/harsh006/dsp/tree/master/
    code/ee18btech11019_dsp2.py
```

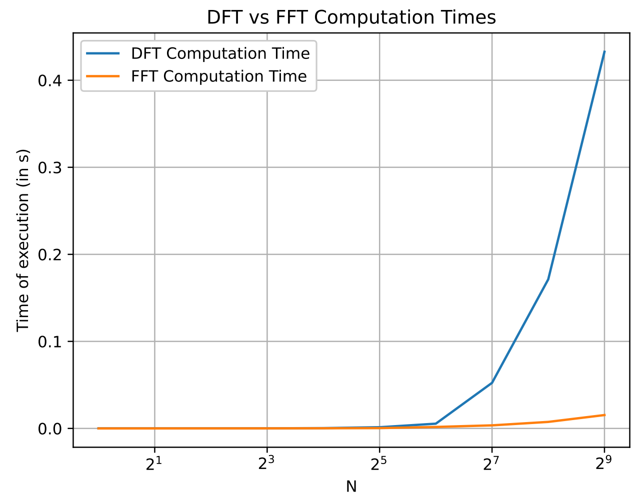
2.15. The following plots are obtained



2.16. The following code COMPARES computation times for N-point DFTs and N-point FFTs of the form N = 2^n

https://github.com/harsh006/dsp/tree/master/code/ee18btech11019_dsp3.py

2.17. The following plot is obtained We can see that



the computation time for DFT computation rises exponentially as we increase N in powers of 2.
But FFT is much faster for larger values

2.18. Because, In FFT - N-point FFT is broken down recursively into 2 N/2-point FFTs recursively. Additionally O(N) operation of Vector multiplication is performed on the N/2 point FFTs. Solving this recurrence gives $O(NlogN)$ time complexity.

2.19. The following is the C program to compute and print the FFT

https://github.com/harsh006/dsp/tree/master/code/ee18btech11019_fft.c