Digital Signal Processing Ⅱ

8^{th} EXPERIMENT

Report

(8th report of DSP2 course)

Subject	Digital Signal Processing Ⅱ
Professor	Je Hyeong Hong
Submission Date	November 4th, 2021
University	Hanyang University
School	College of Engineering
Department	Department of Computer Science & Engineering
Student ID	**Name**
2019009261	최가온(CHOI GA ON)

Exercises

In this part, there are several exercise questions. Each exercise consists of code and its result. All documents including MATLAB code, result, and this report are uploaded in this website :

https://github.com/Gaon-Choi/ELE3077/tree/main/lab_experiment08

Exercise 1

exercise1-a)

Create a function that compute 'fast fourier transform' and save it as 'my_fft.m; whose input can only be a power of 2.

(Do not use the built-in 'fft' function in MATLAB)

(MATLAB Code) my_fft.m

```
function [y, count] = my_fft(x)
    WN = exp(-2*pi*1j / length(x));
    if length(x) == 2
        W = zeros(2, 2);
        for k = 1:2
            for i = 1:2
                W(k, i) = exp(-2*pi*1j*(k-1)*(i-1)/2);
            end
        end
        y = W*x;
        count = length(x);

    else
        Xe = x(1:2:end);
        Xo = x(2:2:end);
        [ye, count1] = my_fft(Xe);
        [yo, count2] = my_fft(Xo);

        Y = zeros(size(x));
        ye2 = [ye;ye];
        yo2 = [yo;yo];

        for i = 1:length(x)
            Y(i) = ye2(i) + WN^(i-1)*yo2(i);
        end
        y = Y;
        count = count1 + count2 + length(x);
```

```
    end
end
```

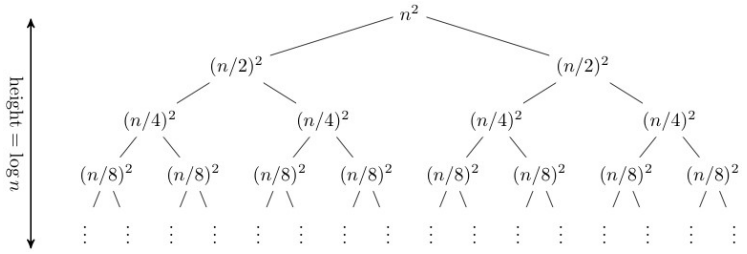
The FFT function is constructed in the form of a recursive function. Therefore, when this function is executed, it is executed in a stacked form like a stack. After that, one by one form the end, is executed and it goes through the process of bringing it together. Since it has this structure, I though of collecting and stacking previous results when implementing "count".

exercise1-b)

Add a 'count' to the 'my_fft' function to check how many times the multiplication takes place.

In other words, modify the my_fft function to [y, count] = my_fft(x).

"count" can be expressed with full binary tree structure, because we implemented with recursive structure when we make 'fft' function, dividing into 2 sub-problems until it reaches the bottom(smallest problem).



source: cs.cornell.edu

It is implemented in my_fft.m, see exercise1_a for detailed information.

exercise1-c)

Generate the sequence ones(64, 1) and calculate the number of multiplications using its fft function. And, check it equals to $n \cdot \log_2 n$. In this case what is the number of multiplications in DFT?

(MATLAB Code) my_fft.m

```
total = 0; N = 64;
x = ones(N, 1);
[X, count] = my_fft(x);

disp("count = " + count);
disp("NlogN = " + N * log2(N));
```

(Results)

Exercise 2

Let $x[n]$ be a sequence

$$x[n] = \cos\left(\frac{\pi}{20}n\right) + 3\cos\left(\frac{\pi}{12}n\right) \quad \text{for } 0 \le n \le 511$$

Generate the sequence X_1 and X_2 that is the FFT of $x[n]$ by using 'my_fft' function and 'fft' function that is MATLAB built-in function. And plot these magnitude and angle respectively.

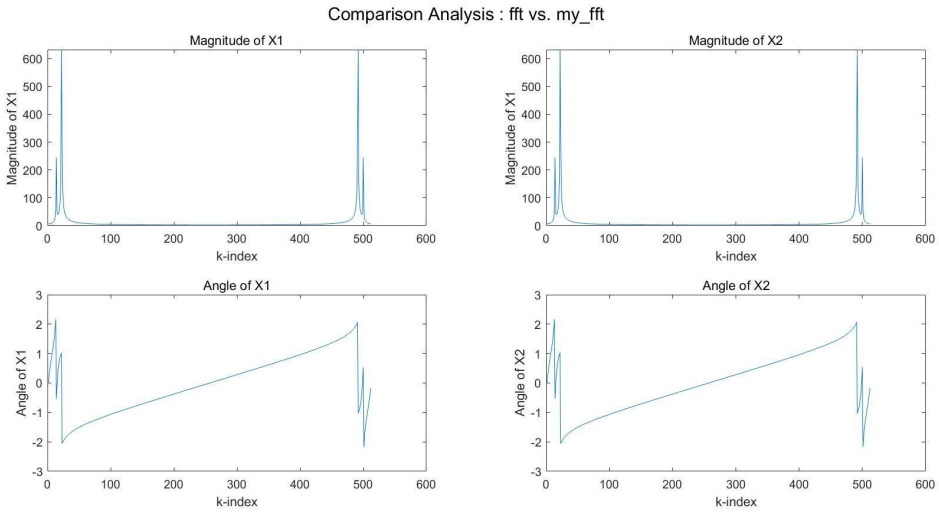
Compare them and check they are almost same.

(HINT: When you use 'my_fft' function, convert the x to the column vector by using x1=x')

(MATLAB Code) my_fft.m

```
n = (0:1:511)';

x = cos((pi/20).*n) + 3 * cos((pi/12).*n);

X1 = my_fft(x);
X2 = fft(x);

magX1 = abs(X1); angX1 = angle(X1);
magX2 = abs(X2); angX2 = angle(X2);

subplot(2, 2, 1);
plot(magX1); title("Magnitude of X1");
xlabel("k-index"); ylabel("Magnitude of X1");

subplot(2, 2, 2);
plot(magX2); title("Magnitude of X2");
xlabel("k-index"); ylabel("Magnitude of X1");

subplot(2, 2, 3);
plot(angX1); title("Angle of X1");
xlabel("k-index"); ylabel("Angle of X1");

subplot(2, 2, 4);
plot(angX2); title("Angle of X2");
xlabel("k-index"); ylabel("Angle of X2");

sgtitle("Comparison Analysis : fft vs. my\_fft");
```

(Results)



According to the result above, we can infer that the implemented 'my_fft' function operates accurately, as same as MATLAB's built-in function 'fft'.