DSP Lab. Week 7
Fourier Transform

Kyuheon Kim
Media Lab. Rm567
kyuheonkim@khu.ac.kr

Last update : September 2, 2019

MOTIVATION

- Synthesize Complicated Signals
  - Musical Notes
    - ✓ Piano uses 3 strings for many notes
    - ✓ Chords: play several notes simultaneously
  - Human Speech
    - ✓ Vowels have dominant frequencies
    - ✓ Application: computer generated speech
  - Can all signals be generated this way?
    - ✓ Sum of sinusoids?

Euler's Formula Reversed

- Solve for cosine (or sine)

$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$$

$$e^{-j\omega t} = \cos(-\omega t) + j\sin(-\omega t)$$

$$e^{-j\omega t} = \cos(\omega t) - j\sin(\omega t)$$

$$e^{j\omega t} + e^{-j\omega t} = 2\cos(\omega t)$$

$$\boxed{\cos(\omega t) = \tfrac{1}{2}(e^{j\omega t} + e^{-j\omega t})}$$

Inverse Euler's Formula

❖ Solve for cosine (or sine)

$$\cos(\omega t) = \frac{1}{2}\left(e^{j\omega t} + e^{-j\omega t}\right)$$

$$\sin(\omega t) = \frac{1}{2j}\left(e^{j\omega t} - e^{-j\omega t}\right)$$

Spectrum Interpretation

❖ **Cosine = sum of 2 complex exponentials:**

$$A\cos(7t) = \frac{A}{2}e^{j7t} + \frac{A}{2}e^{-j7t}$$

One has a positive frequency
The other has negative freq.
Amplitude of each is half as big

Spectrum Interpretation

- **Sine = sum of 2 complex exponentials:**

$$A\sin(7t) = \frac{A}{2j}e^{j7t} - \frac{A}{2j}e^{-j7t}$$

$$= \frac{1}{2}Ae^{-j0.5\pi}e^{j7t} + \frac{1}{2}Ae^{j0.5\pi}e^{-j7t}$$

$$\boxed{\frac{-1}{j} = j = e^{j0.5\pi}}$$

- **Positive freq. has phase = -0.5π**
- **Negative freq. has phase = +0.5π**

Fourier Series

- ANALYSIS
  - Get representation from the signal
  - Works for PERIODIC Signals
- Fourier Series
  - Answer is: an INTEGRAL over one period

$$a_k = \frac{1}{T_0}\int_0^{T_0} x(t)e^{-j\omega_0 k t}dt$$

What if x(t) is not periodic?

- **Sum of Sinusoids?**
  - Non-harmonically related sinusoids
  - Would not be periodic, but would probably be non-zero for all *t.*
- **Fourier transform**
  - gives a "sum" (actually an integral) that involves ALL frequencies
  - can represent signals that are identically zero for negative *t.* !!!!!!!!!

Fourier Transform

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega$$

Fourier Synthesis
(Inverse Transform)

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$$

Fourier Analysis
(Forward Transform)

$$\text{Time - Domain} \Leftrightarrow \text{Frequency - Domain}$$

$$x(t) \Leftrightarrow X(j\omega)$$

Fourier Transform Property

$$x(t)*h(t) \Leftrightarrow H(j\omega)X(j\omega)$$

$$x(t)p(t) \Leftrightarrow \frac{1}{2\pi}X(j\omega)*P(j\omega)$$

$$x(t)e^{j\omega_0 t} \Leftrightarrow X(j(\omega-\omega_0))$$

Differentiation Property

$$\frac{dx(t)}{dt} \Leftrightarrow (j\omega)X(j\omega)$$

Fourier Transform Property

Linearity Property

$$ax_1(t)+bx_2(t) \Leftrightarrow aX_1(j\omega)+bX_2(j\omega)$$

Delay Property

$$x(t-t_d) \Leftrightarrow e^{-j\omega t_d}X(j\omega)$$

Frequency Shifting

$$x(t)e^{j\omega_0 t} \Leftrightarrow X(j(\omega-\omega_0))$$

Scaling

$$x(at) \Leftrightarrow \frac{1}{|a|}X(j(\frac{\omega}{a}))$$

Discrete Fourier Transform(DFT)

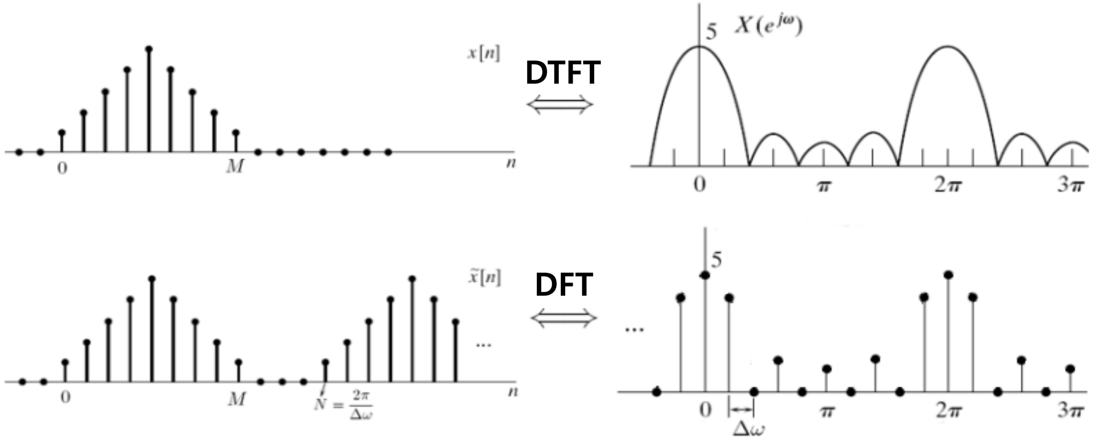
$$\underbrace{X[k]}_{\text{Discrete}} = \sum_{n=0}^{N-1} x[n] e^{-j\left(\frac{2\pi}{N}\right)kn}$$

DFT

$$\underbrace{x[n]}_{\text{Discrete}} = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\left(\frac{2\pi}{N}\right)kn}$$

Inverse DFT

Signal Variable	Transform Variable	Transform
Continuous	Continuous	Fourier Transform (FT)
Sample in Time		
Discrete	Continuous	Discrete-Time Fourier Transform (DTFT)
Sample in Frequency		
Discrete	Discrete	Discrete Fourier Transform (DFT)



DFT Programming

$$X[k] = \boxed{\sum_{n=0}^{N-1}} x[n]\, e^{-\frac{j2\pi kn}{N}}$$

$$k = 0,\ 1,\ .\ .\ .\ ,\ N-1$$

DFT

```
for (int k = 0; k < N; k++) {
    for (int n = 0; n < N; n++) {
        X[k] += x[n] * complex(cos((-2 * PI*k*n) / (double)N), sin((-2 * PI*k*n) / (double)N));
    }
}
```

DFT Programming

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{N}} \longrightarrow \boxed{e^{j\theta} = \cos(\theta) + j\sin(\theta)}$$

$$k = 0,\ 1,\ .\ .\ .\ ,\ N-1$$

DFT

```
for (int k = 0; k < N; k++) {
    for (int n = 0; n < N; n++) {
        X[k] += x[n] * complex(cos((-2 * PI*k*n) / (double)N), sin((-2 * PI*k*n) / (double)N));
    }
}
```

DFT Example1 (Page 336.)

DFT) First we consider the case of a complex exponential with a special frequency.

$$x1[n] = e^{j2\pi\frac{nko}{N}},\ \ for\ n = 0,1,2,3\ \ldots\ldots N-1$$

$\boldsymbol{ko = 200}$ **, N = 1000**

$$X[k] = \sum_{n=0}^{N-1} x1[n]\, e^{-\frac{j2\pi kn}{N}}$$ 을 구하시오

DSP2_Homework1 (전역 범위) main()

```cpp
#include<iostream>
#include<cmath>
#include<fstream>
#include"complex.h"
using namespace std;
#define PI 3.141592

void main()
{
	ofstream OutFile_k,OutFile_mag,OutFile_phase;
	OutFile_k.open("k.txt");
	OutFile_mag.open("mag.txt");
	OutFile_phase.open("phase.txt");

	int N = 1000;
	int k0 = 200;
	complex* x1 = new complex[N];
	for (int n = 0; n < N; n++) {
		x1[n] = complex( cos((2*PI*n*k0)/(double)N), sin((2*PI*n*k0)/(double)N));
	}

	complex* X = new complex[N];
	for (int k = 0; k < N; k++) {
		for (int n = 0; n < N; n++) {
			X[k] += x1[n] * complex( cos((-2*PI*k*n)/(double)N), sin((-2*PI*k*n)/(double)N) );
		}

	}

	for (int k = 0; k < N; k++) {
		OutFile_k << k << endl;
		OutFile_mag << X[k].mag() << endl;
		OutFile_phase << X[k].phase() << endl;
	}

	system("pause");
	return;
}
```

DFT Example1 (Page 336.)

DFT) First we consider the case of a complex exponential with a special frequency.

$$x1[n] = e^{j2\pi\frac{nko}{N}},\ \ for\ n = 0,1,2,3\ \ldots..N-1$$

$ko = 200$, N = 1000

$$X[k] = \sum_{n=0}^{N-1} x1[n]\, e^{-\frac{j2\pi kn}{N}}$$ 을 구하시오

```
DSP2_Homework1        (전역 범위)        main()

#include<iostream>
#include<cmath>
#include<fstream>
#include"complex.h"
using namespace std;
#define PI 3.141592

void main()
{
    ofstream OutFile_k,OutFile_mag,OutFile_phase;
    OutFile_k.open("k.txt");
    OutFile_mag.open("mag.txt");
    OutFile_phase.open("phase.txt");

    int N = 1000;
    int k0 = 200;
    complex* x1 = new complex[N];
    for (int n = 0; n < N; n++) {
        x1[n] = complex( cos((2*PI*n*k0)/(double)N), sin((2*PI*n*k0)/(double)N));
    }

    complex* X = new complex[N];
    for (int k = 0; k < N; k++) {
        for (int n = 0; n < N; n++) {
            X[k] += x1[n] * complex( cos((-2*PI*k*n)/(double)N), sin((-2*PI*k*n)/(double)N) );
        }


    }

    for (int k = 0; k < N; k++) {
        OutFile_k << k << endl;
        OutFile_mag << X[k].mag() << endl;
        OutFile_phase << X[k].phase() << endl;
    }

    system("pause");
    return;
}
```

DFT Example1 (Page 336.)

DFT) First we consider the case of a complex exponential with a special frequency.

$x1[n] = e^{j2\pi\frac{nko}{N}}, \ for\ n = 0,1,2,3 \ldots\ldots N-1$ $\quad$ $ko = 200$, N = 1000

$$X[k] = \sum_{n=0}^{N-1} x1[n]\, e^{-\frac{j2\pi kn}{N}}$$ 을 구하시오

DSP2_Homework1 (전역 범위) main()

```cpp
#include<iostream>
#include<cmath>
#include<fstream>
#include"complex.h"
using namespace std;
#define PI 3.141592

void main()
{
    ofstream OutFile_k,OutFile_mag,OutFile_phase;
    OutFile_k.open("k.txt");
    OutFile_mag.open("mag.txt");
    OutFile_phase.open("phase.txt");

    int N = 1000;
    int k0 = 200;
    complex* x1 = new complex[N];
    for (int n = 0; n < N; n++) {
        x1[n] = complex( cos((2*PI*n*k0)/(double)N), sin((2*PI*n*k0)/(double)N));
    }

    complex* X = new complex[N];
    for (int k = 0; k < N; k++) {
        for (int n = 0; n < N; n++) {
            X[k] += x1[n] * complex( cos((-2*PI*k*n)/(double)N), sin((-2*PI*k*n)/(double)N) );
        }

    }

    for (int k = 0; k < N; k++) {
        OutFile_k << k << endl;
        OutFile_mag << X[k].mag() << endl;
        OutFile_phase << X[k].phase() << endl;
    }

    system("pause");
    return;
}
```

DFT Example1 (Page 336.)

DFT) First we consider the case of a complex exponential with a special frequency.

x1[n] = $e^{j2\pi\frac{nko}{N}}$, $for\ n = 0,1,2,3 \ldots\ldots N-1$ $ko = 200$, N = 1000

$X[k] = \sum_{n=0}^{N-1} x1[n]\, e^{-\frac{j2\pi kn}{N}}$ 을 구하시오

DSP2_Homework1   (전역 범위)   main()

```
#include<iostream>
#include<cmath>
#include<fstream>
#include"complex.h"
using namespace std;
#define PI 3.141592

void main()
{
    ofstream OutFile_k,OutFile_mag,OutFile_phase;
    OutFile_k.open("k.txt");
    OutFile_mag.open("mag.txt");
    OutFile_phase.open("phase.txt");

    int N = 1000;
    int k0 = 200;
    complex* x1 = new complex[N];
    for (int n = 0; n < N; n++) {
        x1[n] = complex( cos((2*PI*n*k0)/(double)N), sin((2*PI*n*k0)/(double)N));
    }

    complex* X = new complex[N];
    for (int k = 0; k < N; k++) {
        for (int n = 0; n < N; n++) {
            X[k] += x1[n] * complex( cos((-2*PI*k*n)/(double)N), sin((-2*PI*k*n)/(double)N) );
        }

    }

    for (int k = 0; k < N; k++) {
        OutFile_k << k << endl;
        OutFile_mag << X[k].mag() << endl;
        OutFile_phase << X[k].phase() << endl;
    }

    system("pause");
    return;
}
```

DFT Example1 (Page 336.)

DFT) First we consider the case of a complex exponential with a special frequency.

$$x1[n] = e^{j2\pi\frac{nko}{N}},\ \ for\ n = 0,1,2,3\ \dots\dots N-1$$

$$\boldsymbol{ko} = \mathbf{200}\ ,\ \mathbf{N = 1000}$$

$$X[k] = \sum_{n=0}^{N-1} x1[n]\, e^{-\frac{j2\pi kn}{N}}$$ 을 구하시오

DSP2_Homework1 (전역 범위) main()

```
#include<iostream>
#include<cmath>
#include<fstream>
#include"complex.h"
using namespace std;
#define PI 3.141592

void main()
{
    ofstream OutFile_k,OutFile_mag,OutFile_phase;
    OutFile_k.open("k.txt");
    OutFile_mag.open("mag.txt");
    OutFile_phase.open("phase.txt");

    int N = 1000;
    int k0 = 200;
    complex* x1 = new complex[N];
    for (int n = 0; n < N; n++) {
        x1[n] = complex( cos((2*PI*n*k0)/(double)N), sin((2*PI*n*k0)/(double)N));
    }

    complex* X = new complex[N];
    for (int k = 0; k < N; k++) {
        for (int n = 0; n < N; n++) {
            X[k] += x1[n] * complex( cos((-2*PI*k*n)/(double)N), sin((-2*PI*k*n)/(double)N) );
        }

    }

    for (int k = 0; k < N; k++) {
        OutFile_k << k << endl;
        OutFile_mag << X[k].mag() << endl;
        OutFile_phase << X[k].phase() << endl;
    }

    system("pause");
    return;
}
```

출력

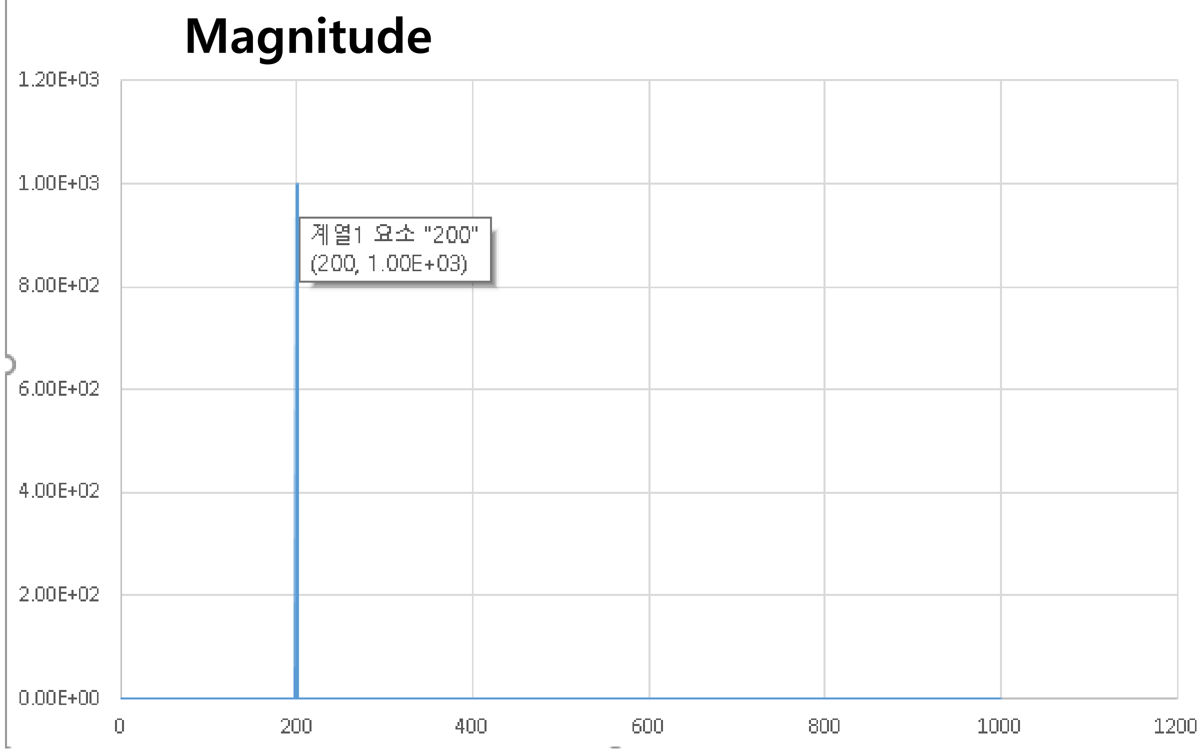
엑셀 그래프

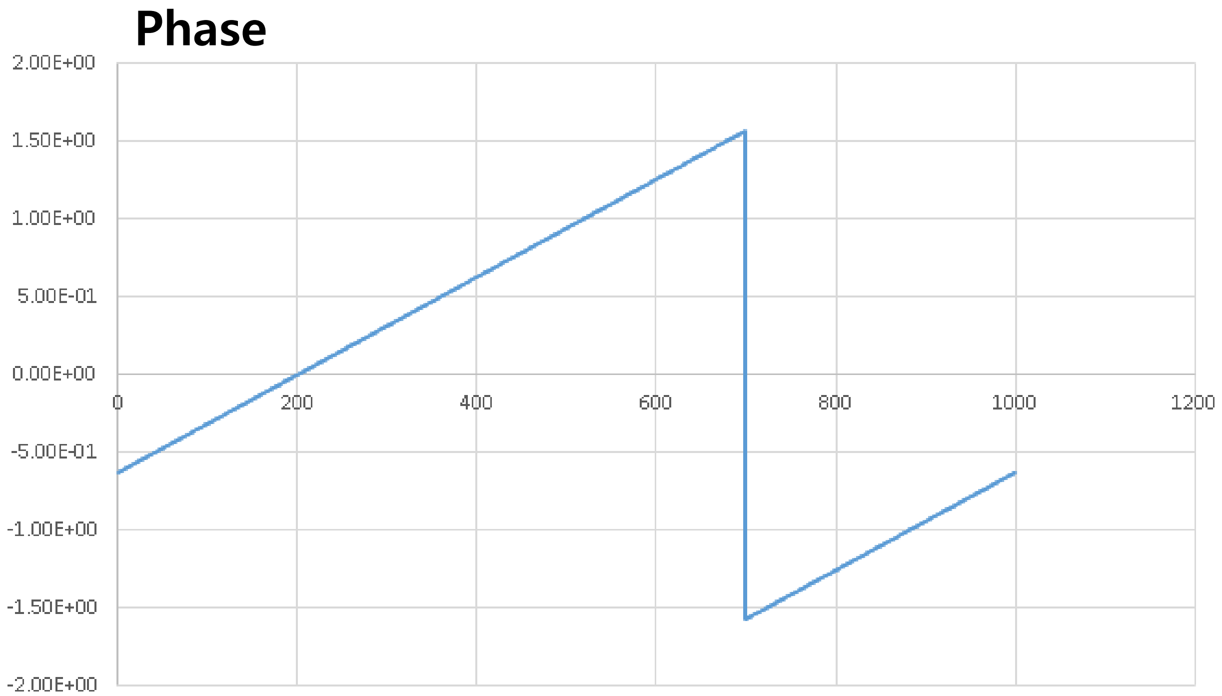
DFT Example1 (Page 336.)

DFT) First we consider the case of a complex exponential with a special frequency.

x1[n] = $e^{j2\pi\frac{nko}{N}}$, $for\ n = 0,1,2,3 \ldots\ldots N-1$ $\quad$ $\boldsymbol{ko} = \mathbf{200}$ **, N = 1000**

$$X[k] = \sum_{n=0}^{N-1} x1[n]\, e^{-\frac{j2\pi kn}{N}}$$ 을 구하시오





IDFT Programming

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot e^{2\pi i k n / N} \quad , \quad n = [0, N-1]$$

IDFT

```
for (int n = 0; n < N; n++) {
    for (int k = 0; k < N; k++) {
        x_[n] += X[k] * complex(cos((2 * PI*k*n) / (double)N), sin((2 * PI*k*n) / (double)N));
    }
    x_[n] = x_[n] / (double)N;
}
```

IDFT Programming

$$x_n = \frac{1}{N}\boxed{\sum_{k=0}^{N-1}} X_k \cdot e^{2\pi ikn/N} \quad , \quad \boxed{n = [0, N-1]}$$

IDFT

```
for (int n = 0; n < N; n++) {
    for (int k = 0; k < N; k++) {
        x_[n] += X[k] * complex(cos((2 * PI*k*n) / (double)N), sin((2 * PI*k*n) / (double)N));
    }
    x_[n] = x_[n] / (double)N;
}
```

IDFT Programming

$$x_n = \boxed{\frac{1}{N}} \sum_{k=0}^{N-1} X_k \cdot e^{2\pi ikn/N} \quad , \quad n = [0, N-1]$$

$$\boxed{e^{j\theta} = \cos(\theta) + j\sin(\theta)}$$

IDFT

```
for (int n = 0; n < N; n++) {
    for (int k = 0; k < N; k++) {
        x_[n] += X[k] * complex(cos((2 * PI*k*n) / (double)N), sin((2 * PI*k*n) / (double)N));
    }
    x_[n] = x_[n] / (double)N;
}
```

DFT Example1 (Page 336.)

IDFT) x[n] = $\cos(0.4\pi n)$, $for\ n = 0,1,2,3\ \ldots\ldots N-1$

신호 x[n]을 DFT – IDFT한 x_[n]과 x[n] 그래프 비교하기

N = 1000

```cpp
#include<iostream>
#include<fstream>
#include"complex.h"
using namespace std;

#define PI 3.141592

void main()
{
    int N = 1000;
    int k0 = 200;

    complex* x = new complex[N];
    for (int n = 0; n < N; n++) {
        x[n] = complex(cos((2 * PI*k0*n) / (double)N),0.0);
    }

    //DFT
    complex* X = new complex[N];
    for (int k = 0; k < N; k++) {
        for (int n = 0; n < N; n++) {
            X[k] += x[n] * complex(cos((-2 * PI*k*n) / (double)N), sin((-2 * PI*k*n) / (double)N));
        }
    }

    //IDFT
    complex* x_ = new complex[N];
    for (int n = 0; n < N; n++) {
        for (int k = 0; k < N; k++) {
            x_[n] += X[k] * complex(cos((2 * PI*k*n) / (double)N), sin((2 * PI*k*n) / (double)N));
        }
        x_[n] = x_[n] / (double)N;
    }

    ofstream OutFile_n, OutFile_xn, OutFile_xn_;
    OutFile_n.open("n.txt"); OutFile_xn.open("x[n].txt");OutFile_xn_.open("x_[n].txt");

    for (int n = 0; n < 20; n++) {
        OutFile_n << n << endl;
        OutFile_xn << x[n].re << endl;
        OutFile_xn_ << x_[n].re << endl;
    }



    system("pause");
    return;
}
```

DFT

IDFT

DFT Example1 (Page 336.)

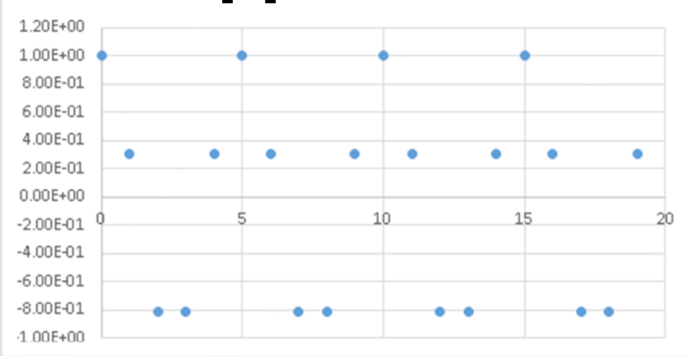
IDFT) x[n] = $\cos(0.4\pi n)\,,\ for\ n = 0,1,2,3\ldots\ldots N-1$

N = 1000

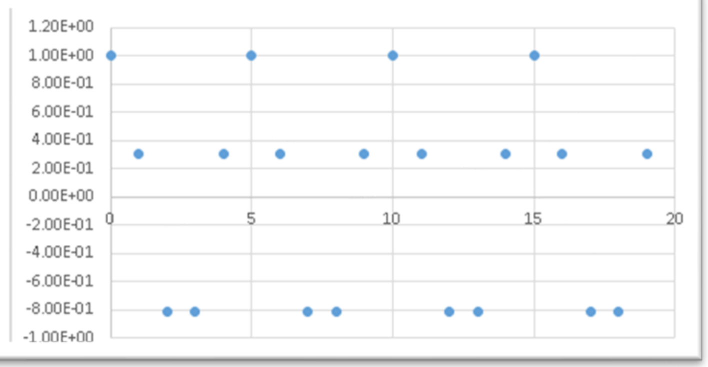
신호 x[n]을 DFT한 후, IDFT하여 x[n] 그래프 비교하기

```
#include<iostream>
#include<fstream>
#include"complex.h"
using namespace std;

#define PI 3.141592

void main()
{
	int N = 1000;
	int k0 = 200;

	complex* x = new complex[N];
	for (int n = 0; n < N; n++) {
		x[n] = complex(cos((2 * PI*k0*n) / (double)N),0.0);
	}

	//DFT
	complex* X = new complex[N];
	for (int k = 0; k < N; k++) {
		for (int n = 0; n < N; n++) {
			X[k] += x[n] * complex(cos((-2 * PI*k*n) / (double)N), sin((-2 * PI*k*n) / (double)N));
		}
	}

	//IDFT
	complex* x_ = new complex[N];
	for (int n = 0; n < N; n++) {
		for (int k = 0; k < N; k++) {
			x_[n] += X[k] * complex(cos((2 * PI*k*n) / (double)N), sin((2 * PI*k*n) / (double)N));
		}
		x_[n] = x_[n] / (double)N;
	}

	ofstream OutFile_n, OutFile_xn, OutFile_xn_;
	OutFile_n.open("n.txt"); OutFile_xn.open("x[n].txt");OutFile_xn_.open("x_[n].txt");

	for (int n = 0; n < 20; n++) {
		OutFile_n << n << endl;
		OutFile_xn << x[n].re << endl;
		OutFile_xn_ << x_[n].re << endl;
	}


	system("pause");
	return;
}
```

x[n]



IDFT 한 x[n]



실제 차이

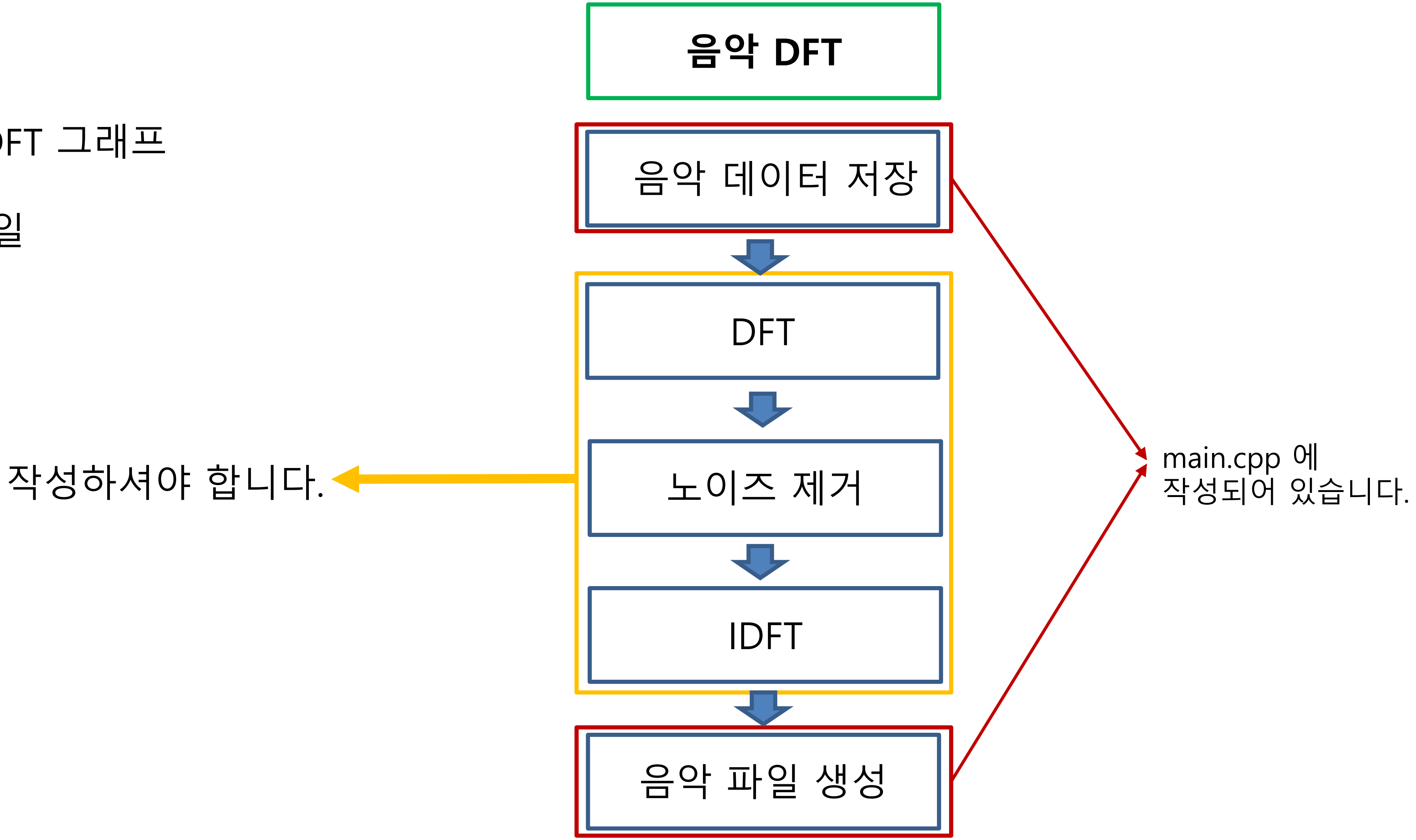
n	x[n]	x_[n]
0	1	0.999967
1	0.309017	0.308985
2	-0.809017	-0.809043
3	-0.809017	-0.809039
4	0.309016	0.308998
5	1	0.999985
6	0.309018	0.309005
7	-0.809016	-0.809028
8	-0.809018	-0.809029
9	0.309015	0.309005

Week 7 assignment

원본 음악 파일을 DFT 그래프, 노이즈가 낀 음악 파일의 DFT 그래프를 각각 출력하라. 노이즈의 주파수를 구하고 노이즈를 제거한 후 IDFT하여 원본 음악과 비슷한 음악 파일을 생성하라.

[보고서 작성 항목]

1) 음악 파일 DFT 그래프
2) 노이즈가 낀 음악 파일 DFT 그래프
3) 노이즈의 주파수
4) 노이즈를 제거한 음악파일



Assignment Rule

"KLAS에 제출할 때 다음 사항을 꼭 지켜주세요"

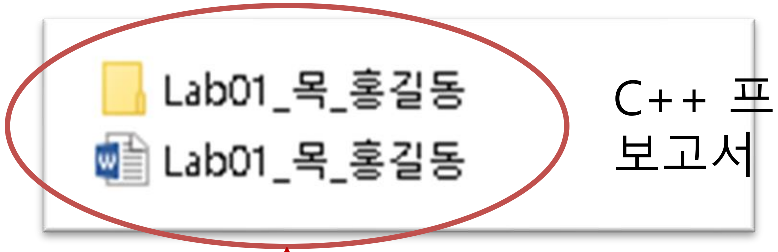
1. 파일명 : "Lab00_요일_대표자이름.zip"
Ex) Lab01_목_홍길동.zip **(압축 툴은 자유롭게 사용)**

2. 제출 파일 **(보고서와 프로그램을 압축해서 제출)**
   - 보고서 파일 (hwp, word): 이름, 학번, 목적, 변수, 알고리즘(순서), 결과 분석, 느낀 점
   - 프로그램

DSP 실험 보고서

과제 번호	Lab01	제출일	2019.09.02
학번/이름	20xxxxxxxx 홍길동 20xxxxxxxx 푸리에		

1. 목적	
2. 변수	
3. 알고리즘	
4. 결과분석	
5. 느낀 점	



C++ 프로그램 파일
보고서

압축하여 제출