Signal Processing

Winter Vacation Capstone Study

TEAM Kai.Lib

발표자 : 원철황

2020.02.04 (TUE)

Orthogonality

연속시간 푸리에 급수

푸리에 급수가 말하는 것은 임의의 주기 함수는 삼각함수의 합으로 표현될 수 있음을 뜻한다.

Orthogonal Functions (직교 함수)

Fourier Analysis 의 Formula 를 공부할 때, 공식을 단순히 외우는 것 보다는 유도 과정을 따라 가는 것이 추후 이해에 도움이 된다. 푸리에 급수 공식은 가장 먼저 함수의 Orthogonality에서부터 출발해야 하며, 이 개념은 함수를 벡터처럼 다룰 수 있다는 것을 뜻한다. 함수 역시 Inner Product를 정의할 수 있으며 구간 [a,b]에서 내적은 다음과 같이 정의된다.

$$(f_1, f_2) = \int_a^b f_1(x) f_2^*(x) dx$$

Orthogonal Set (직교 집합)

그리고 구간 [a,b] 에서 다음을 만족한다면 실수 함수의 집합은 직교 집합이다.

$$(\phi_m, \phi_n) = \int_a^b \phi_m(x) \phi_n^*(x) dx = 0, \ for\, m \neq n$$

여기서 ϕ_n, n = 1,2,3,... 을 기저함수(basis function)라고 부른다.

Orthogonality

Orthogonal Set이 중요한 이유는 구간 [a, b]에서 정의된 함수를 같은 구간에서 정의된 **기저함수**와 상수를 이용해 선형적으로 분해할 수 있기 때문이다. 즉,

$$f(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) + \cdots \quad = \sum_{n=0}^{\infty} c_n\phi_n(x)$$

함수는 벡터이다. 하나의 함수를 Orthogonal Set의 함수를 이용해 위와 같은 급수로 표현될 수 있다는 뜻은 벡터와 연관 지어 생각해야 한다. 선형대수학에서 **벡터**는 **추상적으로 덧셈과 곱셈 연산이 가능한 대수적 객체**(algebraic objects that can be added and scaled)로 정의한다.

선형대수학에서 벡터에 대하여 성립하는 공리

$$x + y = y + x$$
$$(x + y) + z = x + (y + z)$$
$$0 + x = x + 0 = x$$
$$(-x) + x = x + (-x) = 0$$

< 덧셈 >

$$0x = 0$$
$$1x = x$$
$$(cd)x = c(dx)$$

< 곱셈 >

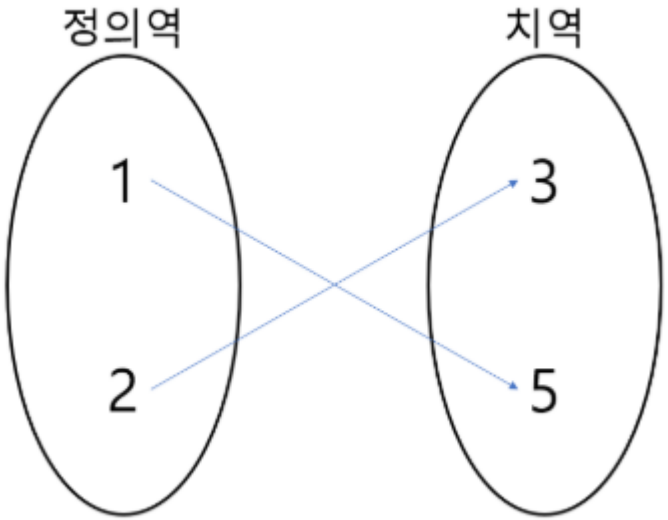
$$c(x + y) = cx + cy$$
$$(c + d)x = cx + dx$$

< 분배 >

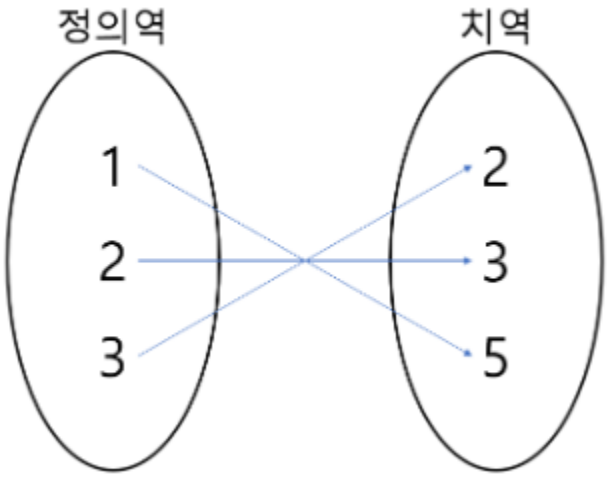
Orthogonality

벡터는 벡터 공간 상의 한 점(point)이다.
또, 벡터는 공간 상의 한 점을 표현하기 위해 사용된다. 마찬가지로 이를 함수에 대해서 확장하면 **기저 벡터**가 기준이 되는 **함수 공간 상의 한 점은 함수 벡터**와 대응된다.

또한 벡터에 대해 논할 때 차원이라는 용어를 많이 사용한다. 그 예시는 다음과 같다.



< 2차원 벡터 (3,5) >



< 3차원 벡터 (5,3,2) >

- 4개의 실수를 나열한 것을 4차원 벡터로 볼 수 있다
- 5개의 실수를 나열한 것을 5차원 벡터로 볼 수 있다
- 일반적으로 실수 함수는 **무한개의 실수를 정의역**으로 하고 **무한개의 함수 값을 치역**으로 하는 무한 차원 벡터이다.

Continuous Time Fourier Series (CTFS)

기저 벡터(basis vector)를 이용한 벡터의 표현

$$(5,3) = 5 \times (1,0) + 3 \times (0,1)$$

이처럼 (1,0) 과 (0,1) 이라는 벡터 쌍을 기저벡터로 삼았을 때, (5,3)을 나타내는 방법을 뜻한다. 이 아이디어를 이용하면, **함수는 무한개의 직교하는 기저벡터**가 필요하게 된다. 그리고 **Continuous Time Fourier Series 에서 삼각함수들은 무한차원 벡터공간의 벡터를 표현해주기 위한 기저 벡터**라고 생각할 수 있다. 따라서 임의의 주기 함수는 다음과 같이 표현 가능한 것이다.

$$f(x) = \sum_{n=0}^{\infty} c_n \phi_n(x)$$

Sine 과 Cosine의 Orthogonality

$$\int_{-\pi}^{\pi} \sin kx \, \cos nx \, dx = 0 \quad \text{always;}$$

$$\int_{-\pi}^{\pi} \sin kx \, \sin nx \, dx = \begin{cases} 0 & \text{if } k \neq n, \\ \pi & \text{if } k = n \neq 0, \\ 0 & \text{if } k = n = 0; \end{cases}$$

$$\int_{-\pi}^{\pi} \cos kx \, \cos nx \, dx = \begin{cases} 0 & \text{if } k \neq n, \\ \pi & \text{if } k = n \neq 0, \\ 2\pi & \text{if } k = n = 0. \end{cases}$$

※ 벡터공간 상의 직교하는 기저벡터는 대표적으로 Gram-Schmidt 방법을 이용할 수 있다.

http://www.stumblingrobot.com/2015/08/06/prove-the-orthogonality-relations-for-sine-and-cosine/

Continuous Time Fourier Series (CTFS)

연속시간 푸리에 급수

x(t) = x(t+T) 를 만족하는 임의의 신호 x(t)는 다음을 만족한다.

$$x(t) = \sum_{k=-\infty}^{\infty} a_k exp\left(j\frac{2\pi k}{T}t\right)$$

이는 어떠한 주기함수 x(t)는 특정한 함수 집합에 의해서 decompose 되었다는 것을 뜻한다.

$$a_k = \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} x(t)exp\left(-j\frac{2\pi k}{T}\right)dt$$

이는 그 계수를 결정하는 식으로 해당 index k 가 x(t) 의 성분에 어느 정도 기여도를 가지고 있는지에 관한 Coefficient 라고 할 수 있다.

이제 x(t)의 타당성에 대해서는 Orthogonal Set이라는 것을 증명하면 입증되는 것이다.

$$\{\phi_k(t)|\phi_k(t) = exp\left(j\frac{2\pi k}{T}t\right),\ k = \cdots, -2, -1, 0, 1, 2, \cdots\ on[0, T]\}$$

Continuous Time Fourier Series (CTFS)

집합의 직교성에 관한 증명

$$\{\phi_k(t)|\phi_k(t) = exp(j\frac{2\pi k}{T}t),\ k = \cdots, -2, -, 1, 0, 1, 2, \cdots, \}$$

에 대하여 Orthogonal Set 의 정의에 의해

$$\int_0^T \phi_k(t)\phi_p^*(t)dt$$

$$= \int_0^T exp\left(j\frac{2\pi k}{T}t\right) exp\left(-j\frac{2\pi p}{T}t\right)dt$$

$$= \int_0^T exp\left(j\frac{2\pi(k-p)}{T}t\right)dt$$

$k = p$ 일 때, $\int_0^T 1dt = T$

$k \neq p$ 일 때,

$$\frac{T}{j2\pi(k-p)}\left|exp\left(j\frac{2\pi(k-p)}{T}t\right)\right|_0^T$$

$$= \frac{T}{j2\pi(k-p)}(exp\left(j2\pi(k-p)\right) - 1) \quad = 0$$

그러므로 해당 집합은 직교 집합이다. 따라서 구간 [0,T] 에서 정의되는 함수 x(t)는 decompose 될 수 있다.

※ Fourier Series의 Kernel Function인 $\phi_k(t) = exp\left(j\frac{2\pi k}{T}t\right)$ 은 주기 함수를 수학적으로 다루기 쉬운 Sinusoidal Function으로 분해 가능함을 뜻함.

Continuous Time Fourier Series (CTFS)

Coefficient 결정에 관한 증명

위의 증명으로부터 주기함수 x(t)를 다음과 같이 나타낼 수 있음을 확인했다.

$$x(t) = \sum_{k=-\infty}^{\infty} a_k exp\left(j\frac{2\pi k}{T}t\right)$$

양 변에 Conjugate된 $\phi_p^*(t) = exp(-j\frac{2\pi p}{T}t)$ 를 곱해주고 적분을 취해주면 다음과 같은 식을 얻는다.

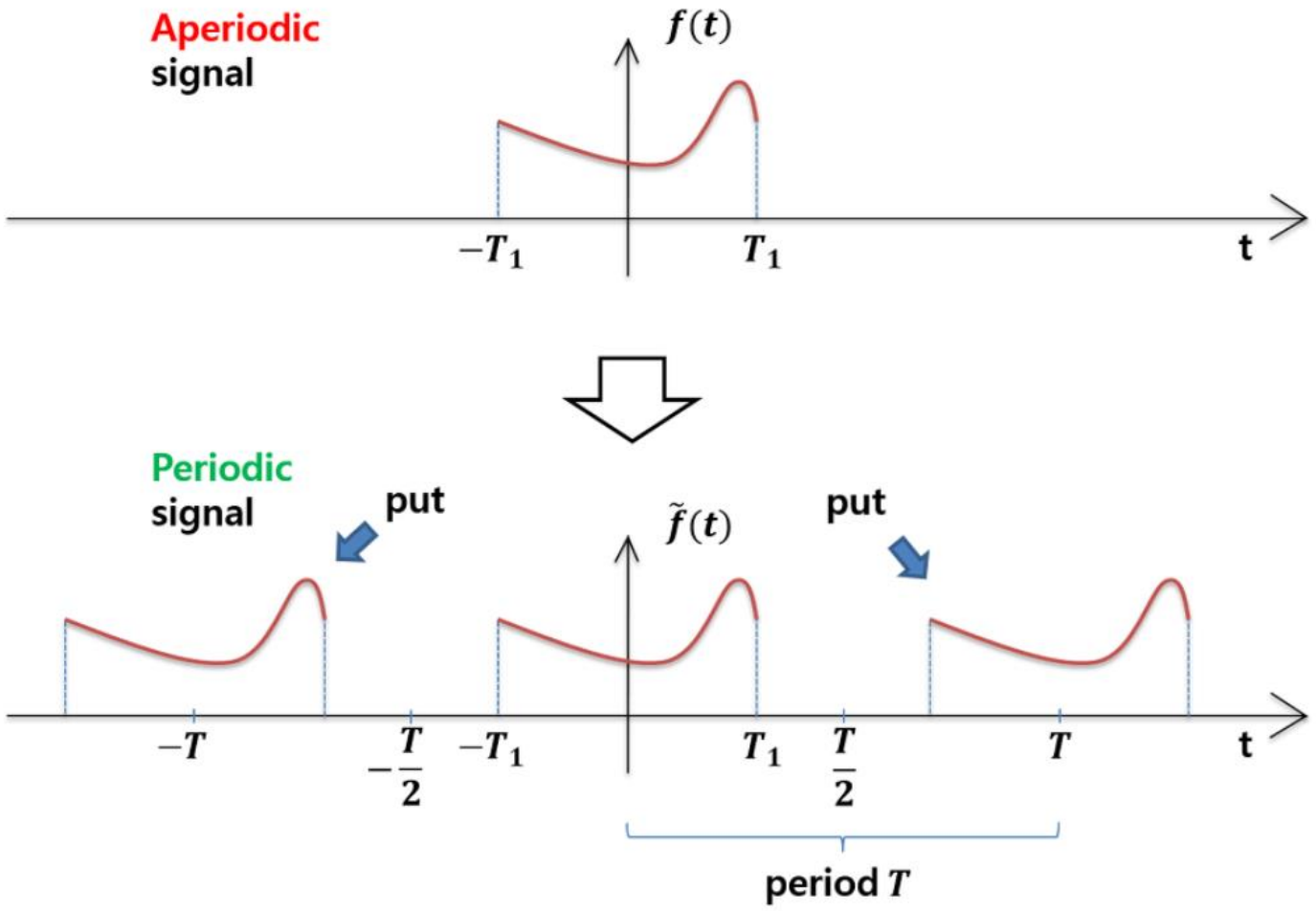
$$\int_0^T x(t)\phi_p^*(t)dt = \int_0^T \sum_{k=-\infty}^{\infty} a_k exp\left(j\frac{2\pi k}{T}t\right) exp\left(-j\frac{2\pi p}{T}t\right)dt \;\; = \sum_{k=-\infty}^{\infty} a_k \int_0^T exp\left(j\frac{2\pi(k-p)}{T}t\right)dt$$

여기서 k = p 인 경우만을 확인하므로 Coefficient는 다음과 같이 나타날 수 있다.

$$a_k = \frac{1}{T}\int_0^T x(t)exp\left(-j\frac{2\pi k}{T}dt\right)$$

Continuous Time Fourier Transform (CTFT)

연속시간 푸리에 변환



Continuous Time Fourier Transform (CTFT)

주기가 T인 주기함수는 다음과 같이 나타낼 수 있음을 증명하였다.

$$x(t) = \sum_{k=-\infty}^{\infty} a_k exp\left(j\frac{2\pi k}{T}t\right)$$

where

$$a_k = \frac{1}{T}\int_0^T x(t)exp\left(-j\frac{2\pi k}{T}t\right)dt$$

Series에서 Transform으로 넘어가는 과정에서 필요한 것은 $T \to \infty$ 이다.

자세한 증명 과정은 생략하면 최종적으로 비주기함수에 대한 CTFT는 다음과 같이 나타날 수 있다.

$$x(t) = \int_{-\infty}^{\infty} X_{CTFT}(f)exp\left(j2\pi ft\right)df$$

$$X_{CTFT}(f) = \int_{-\infty}^{\infty} x(t)exp\left(-j2\pi ft\right)dt$$

Discrete Time Fourier Series (DTFS)

이산 시간 영역에서의 직교성 (Orthogonality in discrete time domain)

Continuous Time 에 대해서와 마찬가지로 이산 영역의 어떠한 주기 신호 역시 선형결합으로 나타낼 수 있다는 가정에서 출발한다. 따라서 이산 영역에서의 Basis Vector (기저 벡터)의 Orthogonality를 다음과 같이 정의한다. 기존 Series가 주기 적분에 관한 정의였다면 Discrete 영역에서는 Sample에 해당한다.

$$\sum_{n=0}^{N-1} \phi_k[n]\phi_p^*[n] = 0 \; when \; k \neq p$$

직교성에 관한 증명은 다음과 같이 이루어진다.

$$\left\{ \phi_k[n] | \phi_k[n] = exp\left(j\frac{2\pi k}{N}n\right) \; where \; k = 0, 1, 2, \cdots, N-1 \right\}$$

위의 집합에 대하여 이산 시간 영역에서의 직교성 정의를 적용하면 아래와 같다.

$$\sum_{n=0}^{N-1} \phi_k[n]\phi_p^*[n] \; = \sum_{n=0}^{N-1} exp\left(j\frac{2\pi k}{N}n\right) exp\left(-j\frac{2\pi p}{N}n\right) \; = \sum_{n=0}^{N-1} exp\left(j\frac{2\pi (k-p)}{N}n\right) = \sum_{n=0}^{N-1} \left\{ exp\left(j\frac{2\pi (k-p)}{N}\right)^n \right\}$$

Basis Vector는 Discrete 영역에서 N개만 필요하다 가정한다.

Discrete Time Fourier Series (DTFS)

이산 시간 영역에서의 직교성 (Orthogonality in discrete time domain)

여기서 k = p 인 경우와 k != p 인 경우로 나눌 수 있다.
먼저 k = p인 경우 이전 식은 1을 N번 더한 횟수로 간단히 N으로 나타낼 수 있다.

$$\sum_{n=0}^{N-1} exp\left(j\frac{2\pi n}{N}(0)\right) = N$$

반면 k != p 인 경우 식은 초항이 1이고, 공비가 $exp(j\frac{2\pi(k-p)}{N})$ 인 등비급수의 합으로 나타낼 수 있다.

$$\sum_{n=0}^{N-1}\left\{exp\left(j\frac{2\pi(k-p)}{N}\right)^n\right\} \Rightarrow \frac{1-exp\left(j\frac{2\pi(k-p)}{N}\right)^N}{1-exp\left(j\frac{2\pi(k-p)}{N}\right)} = \frac{1-exp\left(j2\pi(k-p)\right)}{1-exp\left(j\frac{2\pi(k-p)}{N}\right)}$$

이때, k != p 이므로 exp(j * 정수) = 1 이므로 위 식은 0이다.
따라서 앞선 정의에 의해 서로 다른 주기를 가진 Exponential Euler Form은 서로가 독립적이다.

Discrete Time Fourier Series (DTFS)

이산 시간 푸리에 급수 증명

따라서 주기가 N인 이산신호 x[n]에 대하여, Orthogonal Set을 찾았기에, 우리는 각 인덱스에 해당하는 Coefficient 값을 a_k 라 나타내고 다음과 같이 나타낼 수 있다.

$$x[n] = \sum_{k=0}^{N-1} a_k \, exp\left(j\frac{2\pi k}{N}n\right)$$

그리고 이 선형식에 해당하는 a_k 는 다음과 같이 구할 수 있다.

$$a_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n] exp\left(-j\frac{2\pi k}{N}n\right) \; for\; k = 0, 1, \cdots, N-1$$

이는 x[n]에 Conjugate된 Kernel Function을 취한 뒤 Summation을 취하는 과정을 통해 유도 가능하다.

$$\sum_{n=0}^{N-1} x[n]\phi_r^*[n] = \sum_{n=0}^{N-1}\left(\sum_{k=0}^{N-1} a_k \, exp\left(j\frac{2\pi k}{N}n\right)\right)\phi_r^*[n] = \sum_{n=0}^{N-1} a_k \sum_{k=0}^{N-1} \phi_k[n]\phi_r^*[n] = \sum_{n=0}^{N-1} a_k N\delta[k-r] = Na_r$$

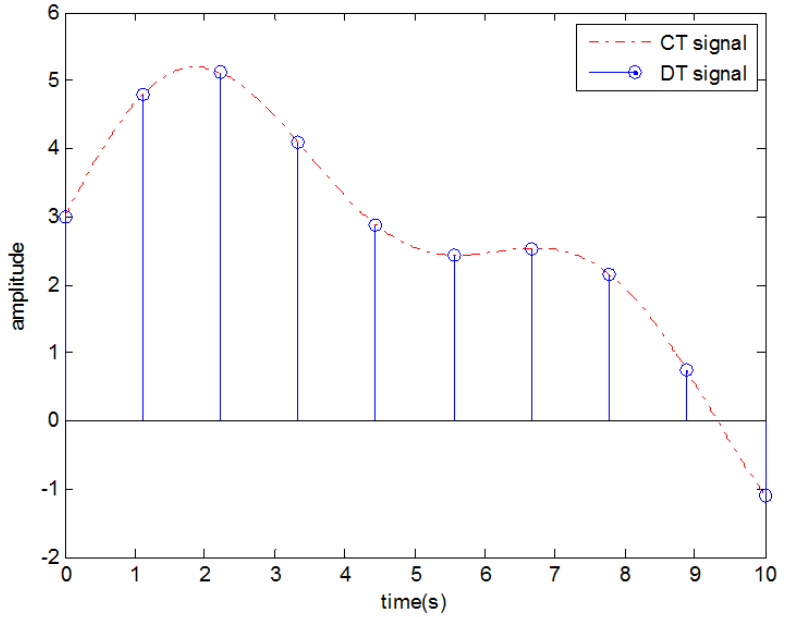
따라서 DTFS의 계수는 위와 같이 나타낼 수 있다.

핵심은 Continuous Time Domain에서 주기 T가 Discrete Time Domain 에서 N으로 변화한 것. 그리고 무한한 실수 index t에서 n으로 바뀐 것이다. 여기서 달라진 것 하나는 **Orthogonal Set**의 **요소 개수**가 **Continuous** 에서는 **무한**했으나 **Discrete Time Domain**에서는 0 ~ N-1로 유한하다는 것이다.

Discrete Time Fourier Transform (DTFT)

이산 시간 푸리에 변환

DTFT를 유도하는 과정은 CTFS에서 CTFT를 유도하는 과정과 거의 흡사하다. 함수의 Orthogonality를 이용해서 Decompose한다는 것 역시 유사하다. 그러나 Discrete Time Domain의 특징 때문에 유도 과정 마지막 부분에 주의할 점이 있다. 먼저 Discrete Time Domain의 특징에 대해 설명한다.

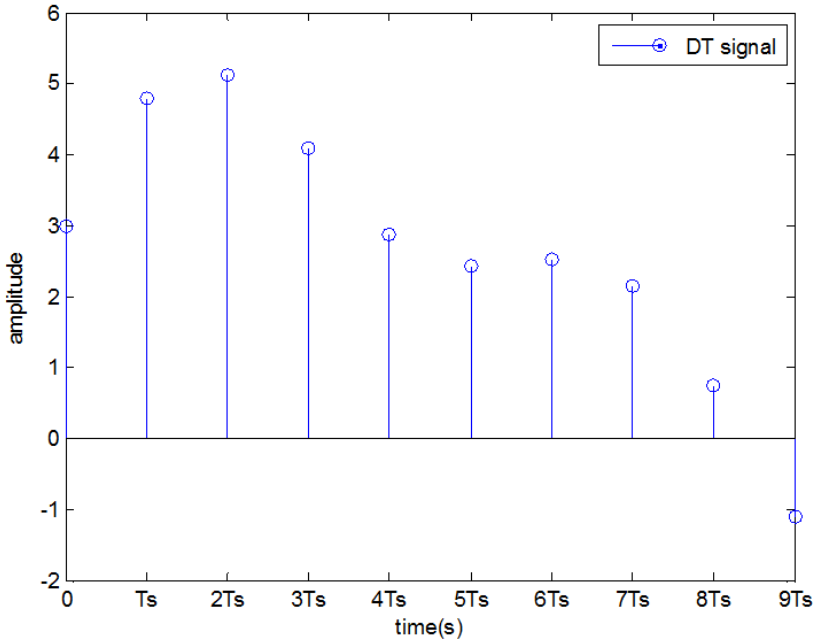


먼저 Discrete Time Signal은 다음과 같이 CT Signal을 Sampling 하는 것에서부터 출발할 수 있다. 즉, 위의 그림에서 빨간 점선으로 표시된 Signal을 주기 Ts (Sampling Time) 로 Sampling 하는 것이다.

Discrete Time Fourier Transform (DTFT)

이산 시간 푸리에 변환

그리고 이를 DT signal만 표현하면 다음과 같이 나타낼 수 있다.



이 때, Ts는 10초를 10칸으로 나누기 때문에 10s를 9등분 한 지점마다 존재하며, 1s의 값을 가지지 않는다. 그렇기에 수식적으로 Sampling 된 Continuous Time Signal은 다음과 같이 표현될 수 있다.

$$x(nTs)$$

Discrete Time Fourier Transform (DTFT)

이산 시간 푸리에 변환

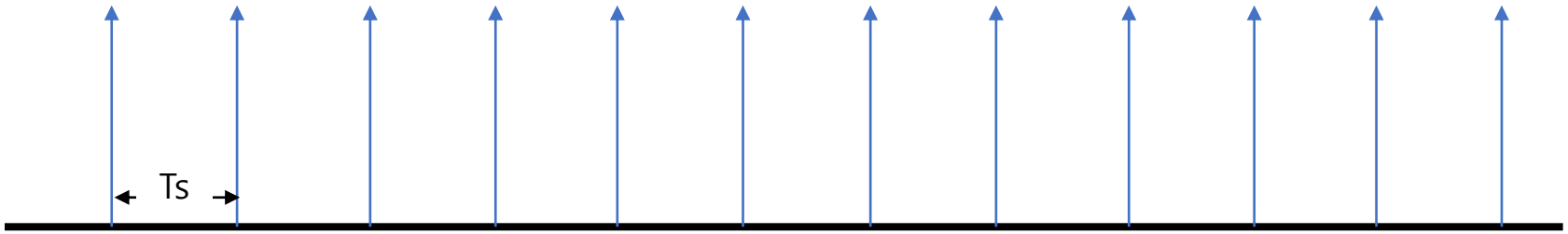
그렇기 때문에 Sampling 시킨 Continuous Time Signal은 최소 주기가 결정된다는 특징을 가지게 된다.

(Ts는 CT Signal을 Discrete하게 관측하는 시간 간격이며, 같은 값이 반복되는 가장 짧은 시간은 Ts마다 반복되며 그 다음은 2Ts 마다 반복이다.)

즉, Continuous Time Domain에서 표현한 Discrete Time Signal은 Ts (Sampling Time or Number) 을 최소 주기로 갖는다. 다른 말로 하면 CT Domain에서 표현한 주파수 f 는 최대 주파수 f_s = 1 / Ts 를 갖게 된다.

$$0 \leq f \leq \frac{1}{T_s} \text{ 또는 } 0 \leq f \leq f_s$$

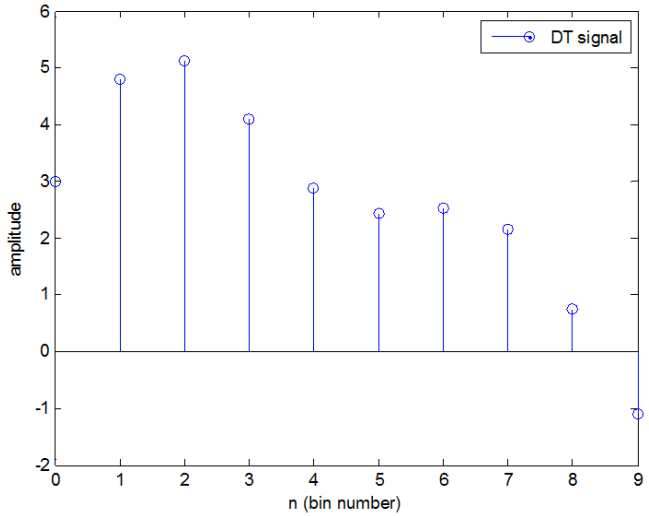
즉, CT Domain에서 표현한 DT Signal은 위의 범위 내에서 나타낼 수 있다. 그렇다면 아래 Ts 의 간격을 하나의 bin 이라 하면 그래프는 어떻게 되는가.



Discrete Time Fourier Transform (DTFT)

이산 시간 푸리에 변환

이 특징을 기억하고, 위 그래프를 Time Domain이 아니라 Discrete Time Domain에 표현하면 다음과 같이 나타낼 수 있다.



DT Signal이 CT Domain 에서 표현된 그래프와 다른 점은 x축이 나타내는 변수가 Sampling 시간 간격인 Ts와 Sampling 된 후 순서 간격인 n이다. 이것이 매우 큰 차이를 가져온다. DT Domain 에서는 n의 순서만을 나타내기 때문인데, DT Signal은 Sampling 된 CT Signal과 다음과 같은 관계를 가진다.

$$x[n] = x(nTs)$$

Discrete Time Fourier Transform (DTFT)

이산 시간 푸리에 변환

DT Domain에서의 n은 통상 bin이라고도 불리며 n을 bin number라고 부르기도 한다. 위의 슬라이드에서 CT Domain에서 가장 짧은 주기가 Ts 가 되는 이유는 관측간격이 Ts이고, 가장 그 다음 관측에서 주기성을 띌 때 가장 짧은 주기를 띄는 것이기 때문이다.

해당 개념을 DT Domain에서 사용하면 n 에 관해서는 bin frequency는 다음과 같은 제한 값을 가진다.

$$0 \leq f \leq 1$$

여기서 f는 Time Frequency 가 아닌 Discrete Time Domain에서의 Bin Frequency이다. 주기성으로 따지면 몇 Bin 마다 주기성이 반복되는가, 에 대한 frequency라고 할 수 있다.

이 개념들을 가지고 Discrete Time Fourier Transform을 유도.역시 같은 개념으로 N (Discrete Domain 에서 주기) 을 무한하게 보내 유도 가능.

x[n] 의 수식에 a_k를 대입하고 lim을 무한으로 취한다.

$$\lim_{N\to\infty} x[n] = \lim_{N\to\infty} \sum_{k=N_1}^{N_2} \left(\frac{1}{N} \sum_{n=N_1}^{N_2} x[n] exp\left(-j\frac{2\pi n}{N}k\right) \right) exp\left(j\frac{2\pi k}{N}n\right)$$

이 때, N2와 N1은 N2 – N1 + 1 = N 을 만족하는 정수이다.

Discrete Time Fourier Transform (DTFT)

이산 시간 푸리에 변환

식을 재정렬하여 1/N 을 맨 오른쪽으로 옮기면 다음과 같다.

$$\lim_{N\to\infty} x[n] = \lim_{N\to\infty} \sum_{k=N_1}^{N_2} \left(\sum_{n=N_1}^{N_2} x[n] exp\left(-j\frac{2\pi n}{N}k\right)\right) exp\left(j\frac{2\pi k}{N}n\right)\frac{1}{N}$$

주기 신호 특성 상, 주기 신호는 어떤 위치의 점에서 시작하던 주기만 유지해주면 신호의 형태는 동일하게 유지된다. 따라서 위의 N1 ~ N2 까지 형태는 같다고 할 수 있다. 또한 N이 무한히 커지면서 다음과 같이 나타낼 수 있다.

$$\frac{1}{N} \to df$$

먼저 1/N 은 df 로 나타낼 수 있다.

$$\frac{k}{N} \to f$$

또한 k/N 은 f 로 나타낼 수 있다.
이 때, df 에 곱해진 임의의 k 값은 Index를 나타내며, 주파수 영역에서 어떤 주파수 index를 나타내는지를 보여준다. 그리고 N1 은 음의 무한, N2는 양의 무한으로 발산함을 고려하여 **괄호 안의** 식을 정리하면 다음과 같이 나타낼 수 있다.

$$\sum_{n=-\infty}^{\infty} x[n] exp(-j2\pi fn) = X_{DTFT}(f)$$

Discrete Time Fourier Transform (DTFT)

이산 시간 푸리에 변환

앞에서는 DT Fourier Transform을 정의했다. 계속해서 이를 x[n] 식에 대입하면 다음과 같이 나타낼 수 있다.

$$\lim_{N\to\infty}\sum_{k=N_1}^{N_2} X_{DTFT}(f)exp\left(j\frac{2\pi k}{N}n\right)\frac{1}{N}$$

정적분의 정의를 이용하면 다음과 같이 나타낼 수 있다.

$$\int_{-\infty}^{\infty} X_{DTFT}(f)exp\left(j2\pi fn\right)df$$

이때, f 가 가질 수 있는 범위는 [0,1] 이므로 이는 최종적으로 다음과 같이 나타낼 수 있다.

$$\int_{0}^{1} X_{DTFT}(f)exp\left(j2\pi fn\right)df = x[n]$$

최종적으로 DTFT를 나타낼 때 다음과 같이 나타낼 수 있다.

$$x[n] = \int_{-0.5}^{0.5} X_{DTFT}(f)exp\left(j2\pi fn\right)df$$

$$X_{DTFT}(f) = \sum_{n=-\infty}^{\infty} x[n]exp\left(-j2\pi fn\right)$$

Discrete Fourier Transform (DFT)

이산 푸리에 변환 (Discrete Fourier Transform)

먼저 **Analog Signal**을 시간 영역에서 **Sampling** 하여 **Digital Signal**로 변환시키고, Sampling 된 Digital Signal을 다시 Analog Signal로 원상 복구 시키기 위한 이론을 **Sampling Theory**라고 한다. 이는 **Nyquist**가 정리했다고 하여 **Nyquist Sampling Theory** 라고도 한다. (Bandlimit 된 신호의 2배 이상의 주파수로 Sampling 해야 이론상 원상복구가 가능하다) 이들은 **Time Domain의 Sampling**을 뜻한다. 그렇다면 **Frequency Domain을 Sampling** 해야 하는 이유는? 답은 **모든 Digital System은 Discrete하며 Finite하기 때문**이다.

즉, Discrete한 간격으로 Sampling된 CT Signal을 사용하려 해도 이 신호들이 유한한 Frequency Domain Spectrum을 가지지 않으면 이는 **"Actual Digital System"**에서 활용 불가능하다는 것이다.

컴퓨터의 Clock은 아무리 빨라도 무한한 영역까지 나타낼 수 없다. 따라서 무한한 Hz까지 분석할 수 없으므로 어떠한 Sampling 된 신호를 다시 한 번 Discrete하게 나누어야 한다.

전체 신호 길이가 N인 이산 신호 x[n]에 대하여, f 영역을 N개로 Sampling하고 f -> (2 * pi * k) / N 대입.

$$X[k] = \sum_{n=0}^{N-1} x[n] exp\left(-j\frac{2\pi k}{N}n\right)$$

이에 직교성을 이용해 inverse DFT를 유도하면 다음과 같다.

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] exp\left(j\frac{2\pi k}{N}n\right)$$

DFT vs DTFT

DFT 와 DTFT의 차이

신호를 DTFT하여 주파수 영역에 나타내면 다음과 같이 파란 선으로 나타낼 수 있다. 주목할 점은 **Digital System**에서 **Fourier Analysis**를 할 때, 연속된 모든 실수 값의 주파수를 사용할 수 **없다**. Clock 역시 Discrete하기 때문이다. 따라서 DFT를 이용해 해당 주파수 영역에서의 값 만을 이용해 각 bin마다 x[n] 값을 나타내도록 한다.

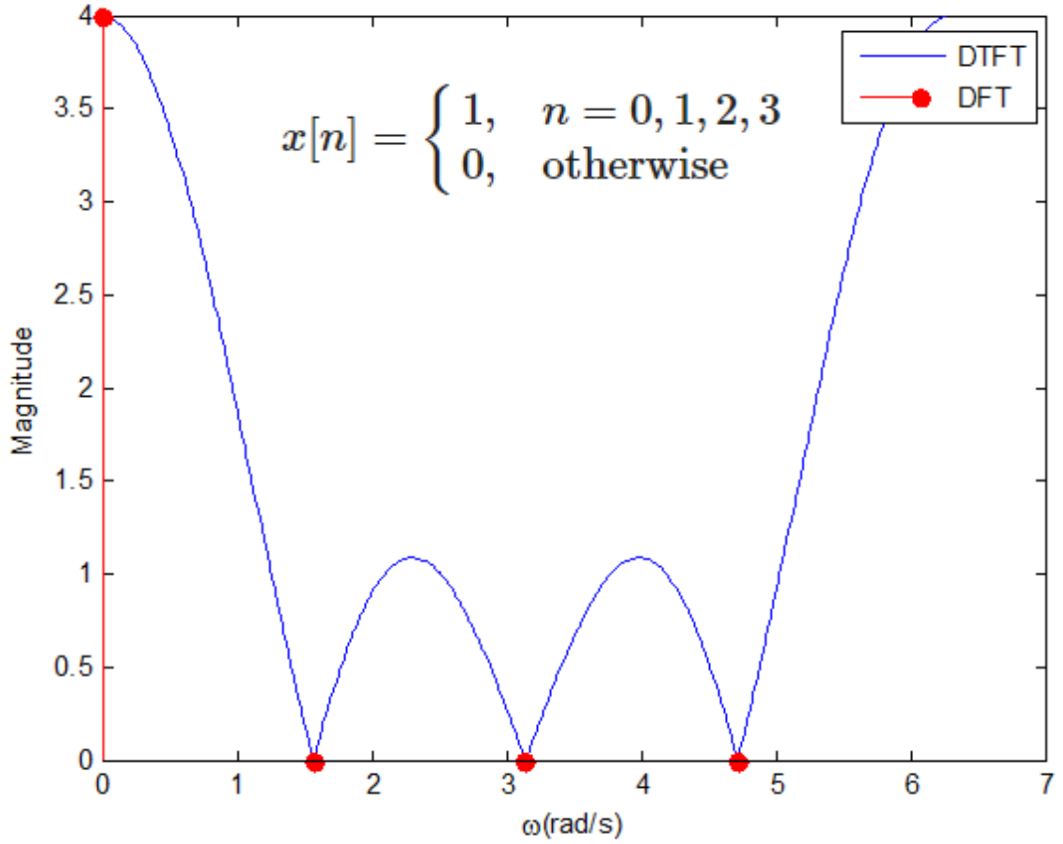
$$exp\left(-j3\pi f\right)\frac{\sin(4\pi f)}{\sin(\pi f)}$$

sampling by f = k/4

$$X[k]=\sum_{n=0}^{3}1\times exp\left(-j\frac{2\pi k}{4}n\right)$$

$$1+exp\left(-j\frac{\pi}{2}k\right)+exp\left(-j\pi k\right)+exp\left(-j\frac{3\pi}{2}k\right)$$

: k = 0,1,2,3 에서 같은 값을 가진다.



※ 변수 **N_FFT** 에서 쓰이는 값이 여기서의 **N(Sampling 된 개수)** 와 관련이 크다고 할 수 있다. 실제적으로 가질 수 있는 **N** 값은 **Clock**에 따라 다를 것으로 예상됨.

Fast Fourier Transform (FFT)

고속 푸리에 변환 (Fast Fourier Transform)

고속 푸리에 변환은 이산 푸리에 변환(DFT)과 그 역변환(IDFT)을 빠르게 수행하는 효율적인 알고리즘이다. FFT는 디지털신호처리에서 편미분 방정식의 근을 구하는 알고리즘에 이르기까지 많은 분야에서 사용된다. 식을 정의에 따라 계산하면 O(n^2)의 연산이 필요하지만, FFT를 이용하면 O(n log n)의 연산만으로 가능하다.

$X[k] = \sum_{n=0}^{N-1} x[n] exp\left(-j\frac{2\pi k}{N}n\right)$ 식이 다음에 대응한다 할 때 $f_j = \sum_{k=0}^{n-1} x_k W^{jk} \qquad j = 0, \ldots, n-1$

Fourier Transform 된 값은 다음과 같이 나타낼 수 있다. 이를 지수의 짝홀을 기준으로 나누면 다음과 같이 나타낼 수 있다.

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$ 에서 행과 열을 바꿔준다. $$\begin{bmatrix} W^0 & W^0 & W^0W^0 & W^0W^0 \\ W^0 & W^2 & W^1W^0 & W^1W^2 \\ W^0 & W^0 & W^2W^0 & W^2W^0 \\ W^0 & W^2 & W^3W^0 & W^3W^2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^1 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^3 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} W^0 & W^0 \\ W^0 & W^2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \end{bmatrix} \\ \begin{bmatrix} W^0 & W^0 \\ W^0 & W^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \end{bmatrix}$$: 결론적으로 FFT는 DFT 연산을 빠르게 진행하기 위한 연산이다.

Discrete Cosine Transform (DCT)

이산 코사인 변환

기존에 이산 푸리에 변환과 유사한 변환이며 수식적으로는 길이가 2배이고 실수 값을 가지는 짝함수 (우함수)에 DFT 연산을 수행하는 것과 동일하다. 가장 널리 쓰이는 것이 type-2 DCT이며, 이 역변환이 type-3 DCT이다.

Relationship between DCT and FFT

DCT (Discrete Cosine Transform) is actually a ***cut-down*** version of the FFT:

- Only the **real** part of FFT
- Computationally simpler than FFT
- DCT -- Effective for Multimedia Compression
- DCT **MUCH** more commonly used.

위는 FFT와 DCT와의 관계이다. DCT는 기본적으로 영상신호처리에서 많이 사용되는 개념이다. 정보량이 많아 값이 낮은 주파수 성분만을 충실히 취하고 높은 주파수 성분은 값이 작으므로 거의 무시하는 과정을 통해 전체 데이터양을 줄이는 원리이다. 영상 데이터는 변화가 매우 적기 때문에 이 연산을 통해 압축이 가능. 음성신호 역시 DCT를 취하면 정보량을 압축할 수 있다.

Discrete Cosine Transform (DCT)

이산 코사인 변환

정의는 다음과 같다. 두 개의 정수 값 I, j 에 의해 표현된 함수가 주어졌을 때, 2D DCT 변환은 이 함수를 i, j와 동일한 범위를 가지는 정수 u, v로 표현되는 새로운 함수로 변환된다.

$$F(u,v) = \frac{2\,C(u)\,C(v)}{\sqrt{MN}} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \cos\frac{(2i+1)\cdot u\pi}{2M} \cdot \cos\frac{(2j+1)\cdot v\pi}{2N} \cdot f(i,j)$$

이 때, Coefficient는 다음과 같이 구할 수 있다.

$$C(\xi) = \begin{cases} \frac{\sqrt{2}}{2} & if \quad \xi = 0, \\ 1 & otherwise. \end{cases}$$

DCT 연산에 의해 각 값들은 주파수에 따라 나열된다. Index가 높아질 수록 Cos(Pi / 2)에 근접하며 값이 작아지게 된다. 즉, 높은 주파수 값으로 연산이 진행될 수록 값이 작아지는 것을 확인할 수 있다.

Discrete Cosine Transform (DCT)

이산 코사인 변환

예시는 다음과 같다. 저주파 성분에 값이 몰리는 것을 확인할 수 있다.
각각의 인접 Index들 간의 차이(변화 정도)를 Coefficient 값으로 표현했다고 생각된다.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

DCT →

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

8×8 정사각 블록의 DCT 변환 결과

MFCC로의 신호처리 이론 적용

What is the Mel Scale?

Mel Scale은 인간의 귀로 인지된 주파수와 관련된다. 인간은 높은 주파수에서 보다 낮은 주파수에서 작은 변화에 더 민감하다. 이 Mel Scale을 적용함에 따라 인간이 귀로 듣는 주파수와 유사한 Feature를 추출 할 수 있다.

$$M(f) = 1125 \ln(1 + f/700)$$

이는 단순 log 연산을 취해주면 된다.
역 연산은 다음과 같다.

$$M^{-1}(m) = 700(\exp(m/1125) - 1)$$

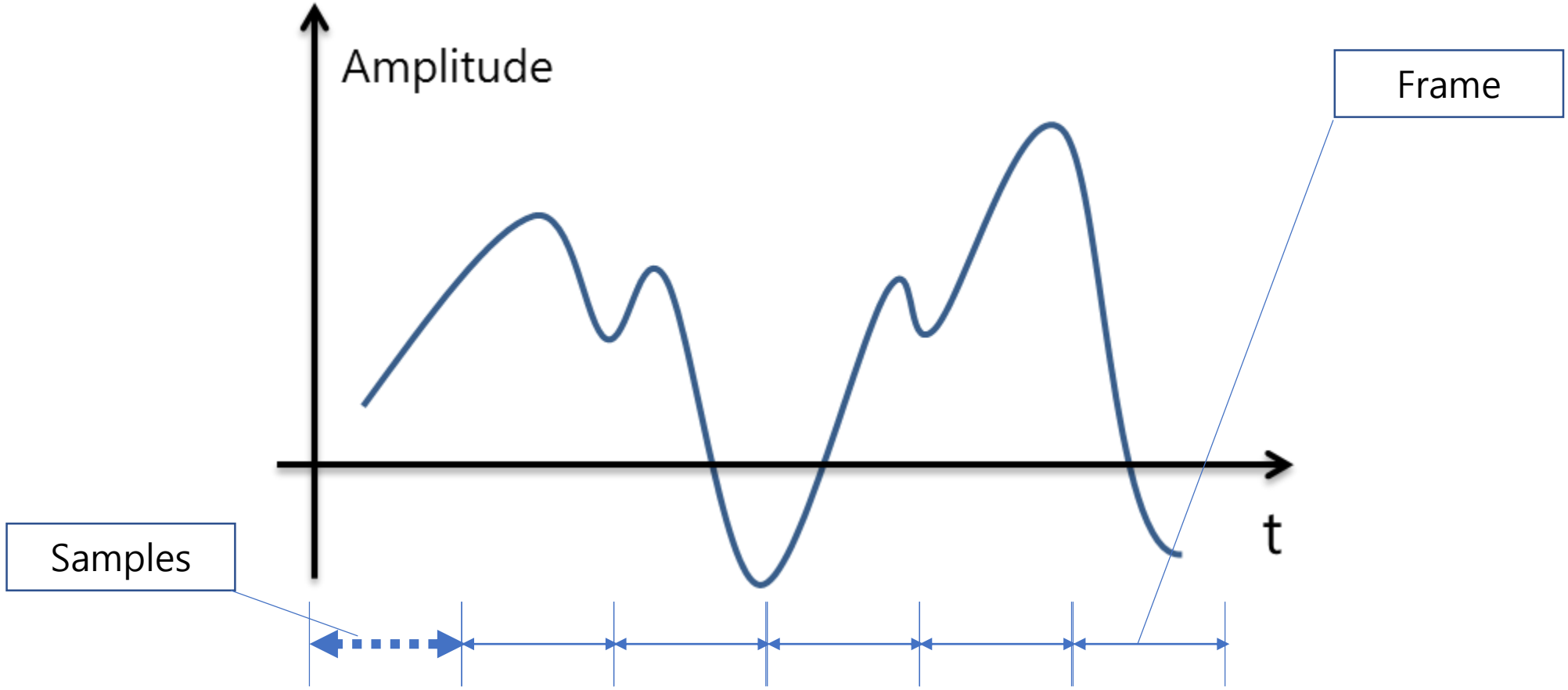
Implementation Steps (적용)

20-40ms frame으로 잘린 speech signal이 있다고 가정. 보통 25ms가 기본이며 이 뜻은 16kHz signal은 0.025 * 16000 = 400 sample 이 존재한다는 것이다. Frame Steps 은 10ms 라고 가정하자. 그러면 0.01 * 16000 = 160 이므로 처음은 0부터 시작하고 그 다음 400개의 sample은 160 sample부터 시작한다. 끝에 다다르면 이는 Zero Padding을 한다.
그 다음은 각 frame에 적용되는 것인데, 12개의 MFCC Coefficient를 각 Frame마다 추출한다.

MFCC로의 신호처리 이론 적용

Examples

Examples of Frame and Samples



MFCC로의 신호처리 이론 적용

Power Spectrum

Time domain에서 signa을 s(n) 이라 하자. i 번 째 frame에 대해 n은 1~400 의 index를 가질 수 있다. 이는 Sample의 개수와 같다. 그리고 Complex DFT (복소수 영역)를 진행할 때 i번째 Frame의 DFT를 구할 수 있고, 이로부터 Power Spectrum을 계산할 수 있다.

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \qquad 1 \le k \le K$$

실질적으로 코드를 구현할 때 Hamming Window 사용여부에 따라 위의 부분을 수정한다. 이때, K는 **length of DFT** 이다. N-point DFT 연산 시 N은 Time Domain에서 Sampling 비중이다. 앞의 예시에서는 400 값을 가진다. 반면 K는 Frequency Domain에서 얼마의 길이로 Sampling을 진행할 것인가를 결정하는 변수.
모델에서 **N_FFT** 변수로 잡아주는 값이 이것이다. (K = **512개로 진행**하고 그 중 **256개 값**을 가져와 사용)

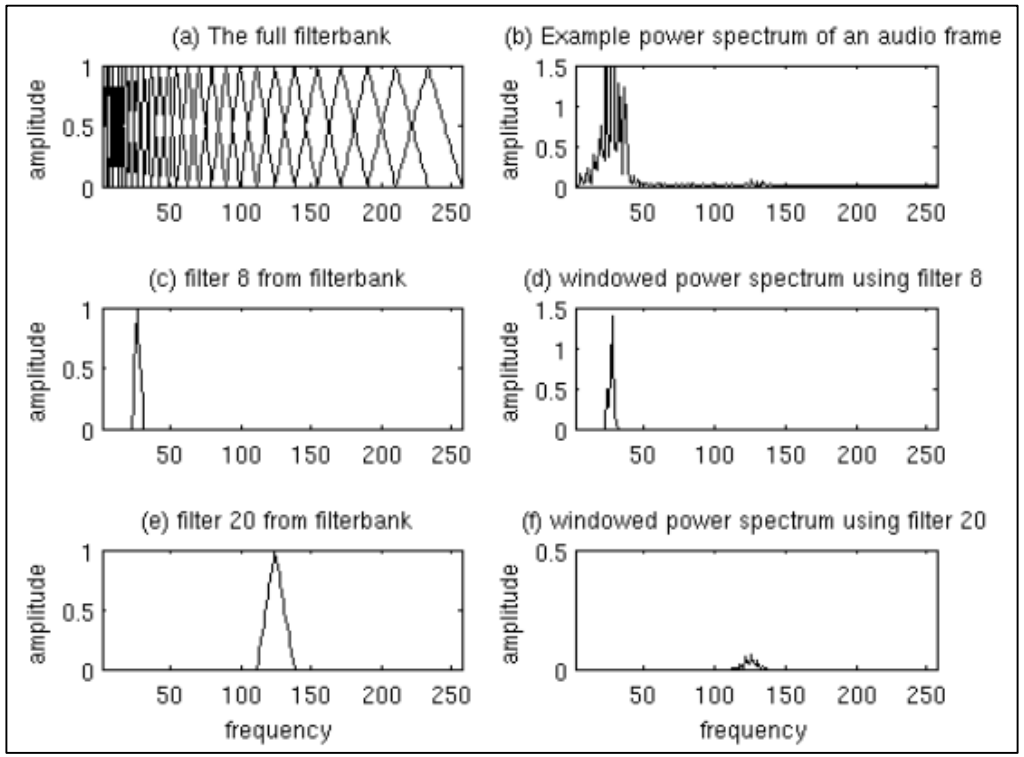
기본적으로 **DTFT** 는 우리가 기본적으로 알고있는 **선형 신호간, Discrete한 영역에서의 Convolution**을 다루기 위해 사용된다. 그러나 **DFT**는 여기서 한 단계 더 나아가 **Computer**에서 신호를 처리하기 위해 만들어졌다. 따라서 K 개의 주파수 영역에서 Sampling 된 각각의 "값"에 대해 제곱(DFT는 Complex이기 때문에 **Conjugate 곱**, 연산이 더 빠르다) 연산이 가능하다.

$$P_i(k) = \frac{1}{N}|S_i(k)|^2$$

MFCC로의 신호처리 이론 적용

Power Spectrum

그리고 각 Frame마다 20~40개의 Triangular Filter를 씌워준다. 이때, Standard는 26 값이다. 따라서 출력되는 Feature Vector는 257개의 열과 26개의 행을 가진다. 여기 까지가 한 Frame에서 나타나는 것이다. 각 Vector들은 대부분 0에 가깝지만 특정 주파수 범위에서는 0이 아닌 값을 보이는데, 이것이 인간이 듣는 Formant 영역이다.



MFCC로의 신호처리 이론 적용

Power Spectrum

마지막으로 26개의 Energy bank들에 대해서 log를 취해준다. 그리고 Discrete Cosine Transform (DCT) 를 취해 Cepstral Coefficient를 얻는다. ASR 영역에서는 낮은 12~13 개의 Coefficient들만 가지고 나머지는 연산에서 제외한다. 이렇게 나온 결과를 MFCC라고 한다.

Mel Filter bank의 계산

In this section the example will use 10 filterbanks because it is easier to display, in reality you would use 26-40 filterbanks.

To get the filterbanks shown in figure 1(a) we first have to choose a lower and upper frequency. Good values are 300Hz for the lower and 8000Hz for the upper frequency. Of course if the speech is sampled at 8000Hz our upper frequency is limited to 4000Hz. Then follow these steps:

1. Using equation 1, convert the upper and lower frequencies to Mels. In our case 300Hz is 401.25 Mels and 8000Hz is 2834.99 Mels.
2. For this example we will do 10 filterbanks, for which we need 12 points. This means we need 10 additional points spaced linearly between 401.25 and 2834.99. This comes out to:

```
m(i) = 401.25, 622.50, 843.75, 1065.00, 1286.25, 1507.50, 1728.74,
       1949.99, 2171.24, 2392.49, 2613.74, 2834.99
```

MFCC로의 신호처리 이론 적용

Mel Filter bank의 계산

3. Now use equation 2 to convert these back to Hertz:

```
h(i) = 300, 517.33, 781.90, 1103.97, 1496.04, 1973.32, 2554.33,
       3261.62, 4122.63, 5170.76, 6446.70, 8000
```

Notice that our start- and end-points are at the frequencies we wanted.

4. We don't have the frequency resolution required to put filters at the exact points calculated above, so we need to round those frequencies to the nearest FFT bin. This process does not affect the accuracy of the features. To convert the frequncies to fft bin numbers we need to know the FFT size and the sample rate,

```
f(i) = floor((nfft+1)*h(i)/samplerate)
```

This results in the following sequence:

```
f(i) =  9, 16,  25,   35,   47,   63,   81,  104,  132, 165,  206,  256
```

We can see that the final filterbank finishes at bin 256, which corresponds to 8kHz with a 512 point FFT size.

5. Now we create our filterbanks. The first filterbank will start at the first point, reach its peak at the second point, then return to zero at the 3rd point. The second filterbank will start at the 2nd point, reach its max at the 3rd, then be zero at the 4th etc. A formula for calculating these is as follows:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

where M is the number of filters we want, and $f()$ is the list of M+2 Mel-spaced frequencies.

MFCC로의 신호처리 이론 적용

Mel Filter bank의 계산