Week 2, 18 August 2025 (Video Learning)

ADC: Analog-to-Digital Converter

The course is about bridging and connecting audio representation received by our receptors (ear) into digital and discrete format that can be processed by computer.

Converting Analog to Digital domain requires a method called sampling. To do a perfect sampling we must obey the Nyquist Sampling theorem.

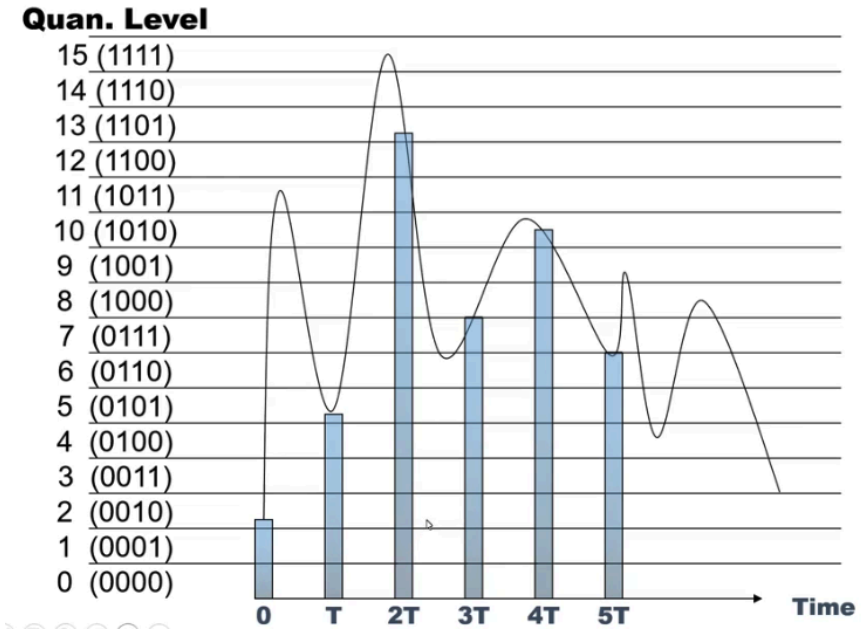
To convert a signal from time domain to frequency domain, we can use a method called Discrete Fourier Transform (DFT)

3. How to tell the difference?

Here's a checklist:

Feature	Time Domain	Frequency Domain
Axis (x-axis)	Time (seconds, ms, samples)	Frequency (Hz, rad/s, or index k)
Notation	$x(t)$ or $x[n]$	$X(f)$ or $X[k]$
Units on y-axis	Amplitude/Signal Value	Magnitude (and phase) of frequency
Typical appearance	Wavy signals, oscillations, random noise	Peaks at certain frequencies, spectrum-like

Sampling



When we have a continuous signal, we do sampling by dividing the value with the interval to produce points that represents a sample. In math, the signal is usually represented with sine function, which is a continuous variable. But when we do sampling, we will have a discrete variable which can be in a type of array or list that stores the discrete numerical value x[k] = [0, T, 2T, 3T, …, 5T] as seen in graph (e.g. x[k] = [0.234, 0.456, 0. 578, …, 0.234]

Nyquist Sampling Rate

One thing to note is that the Nyquist Sampling Rate must be at least 2B, otherise, we will experience the aliasing effect as shown in Lecture 1!

The theorem says:
To perfectly reconstruct a signal without losing information, the sampling rate must be **at least twice the highest frequency** present in the signal.

Mathematically:

$f_s \geq 2B$

B = Bandwidth of the signal / The highest frequency present in the signal.

So if the signal have frequencies up to 5 KHz then the f_s should be 10.000. If we don't obey the Nyquist Sampling theorem, then Aliasing will happen. When it happens, higher frequencies will fold back and appear as lower frequencies in the sampled signal, causing distortion.

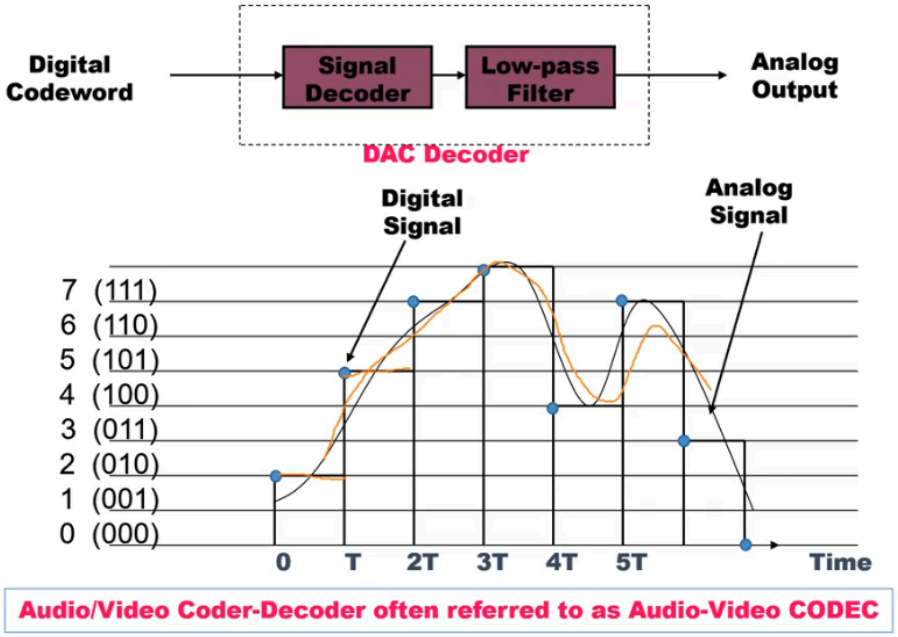
Quantization

Quantization is also one crucial step in sampling. When we sample a signal, we will have signal value at discrete times. But the value can still be a continuous number. And because computer cannot store infinite precision of numbers, we have to round them to fixed level. The rounding is called Quantization.

What is Quantization Interval q?
Imagine the y-axis (amplitude) is divided into equal steps. Each step has a size of quantization interval q. If a sampled value lies between two levels, we round it to the nearest one.

Example:
Let's say we have a 3-bit quantization (so 2^3 = 8 levels, the value ranges from 0-7).
If the sample is 2.36, we quantize it to 2. If it's 2.62, we quantize it to 3. Because it is rounded to nearest level, there is always a difference between the actual signal and the quantized version. This difference is called quantization error or noise. By picking small q (more levels, higher bit-depth) it produces less quantization noise with higher accuracy, but more data. Large q will produce more noise, poorer signal quality, but less data.

DAC: Digital-to-Analog Converter (interpolation)



To convert a digital (discrete values) back to analog (continuous), we use a DAC. The step is as follows:

1. Input: Digital Codeword
    - On the left we can see binary number like 111, 110, 101
    - These are quantized values from the previous analog digital converter
    - Example: 111 = Level 7, 010 = Level 2
2. DAC Decoder → Digital Signal
    - The DAC devocder converts each binary codeword into corresponding amplitude level.
    - The result is a staircase signal shown as the “digital signal” in the plot.
    - Notice how it jumps up and down from on level to another at each sample time 0, T, 2T, 3T, etc.
3. Low-Pass Filter → Analog Signal (Interpolation)
    - That staircase signal is not smooth (it has sharp edges).
    - So to recover the original continuous waveform, we pass it through a low-pass filter.
    - The filter “smooths out” the jumps reconstructing the analog signal (orange curve)
    - This process is sometimes called interpolation because it fills in between the samples
4. Why Low-Pass Filter?
    - The stair-step signal contains not only the original baseband signal but also high-frequencey components
    - A low-pass filter removes those high frequencies, leaving behind only the original analog-like waveform.
    - Without this filter, your audio would sound harsh and robotic.

ADC (Analog-to-Digital Converter): sampling + quantization (continuous → discrete).
DAC (Digital-to-Analog Converter): decoding + interpolation (discrete → continuous).

Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is one of the most important and powerful tools in Digital Signal Processing (DSP).

There is a concept called Decomposition.
Decompose: express (a number or function) as a combination of simpler components.
In DFT, the decomposition produces sine and cosine functions.

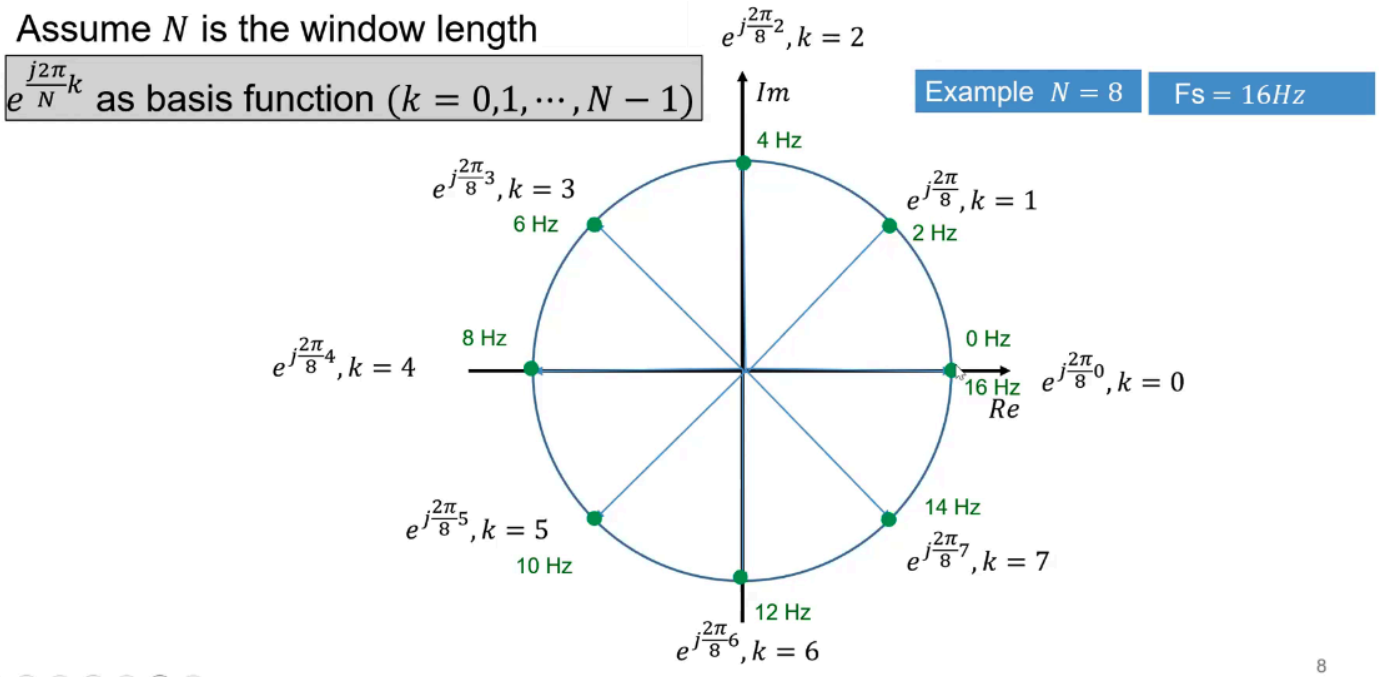
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, 2, \ldots, N-1$$

For DFT, we take assumption that $x[n]$ is periodic.

$X[n]$ is in time domain

The $e^{-j\frac{2\pi}{N}kn}$ is the DFT Basis function.

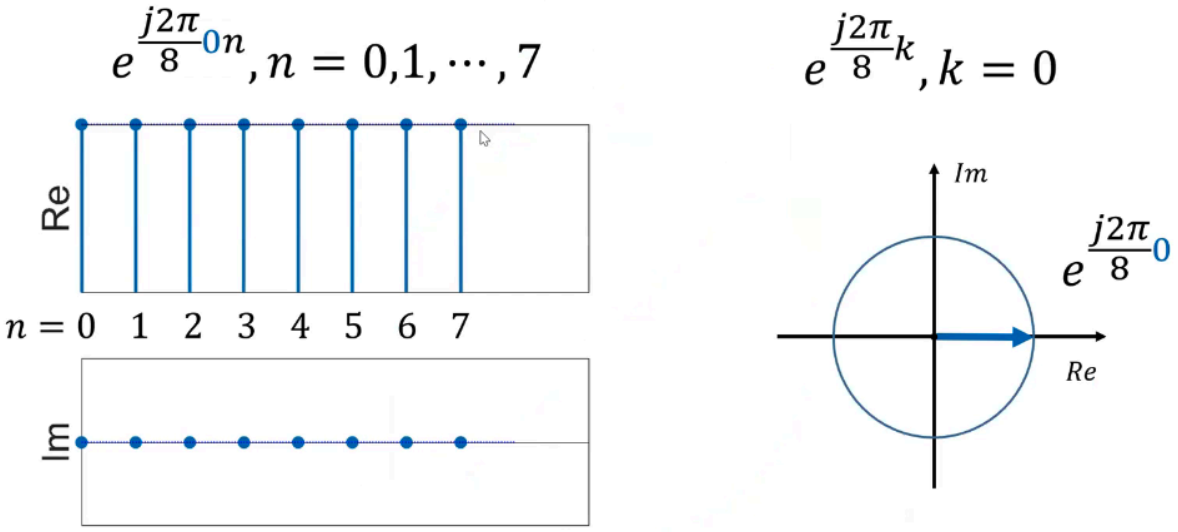
We can use the Euler's formula to express this exponential function into the cosine term and the sine term.
$e^{j\theta} = cos(\theta) + j * sin(\theta)$



So we can use those 8 dots to represent the discrete frequencies. So the DFT basis functions is used to differentiate the real and imaginary part of the signal.

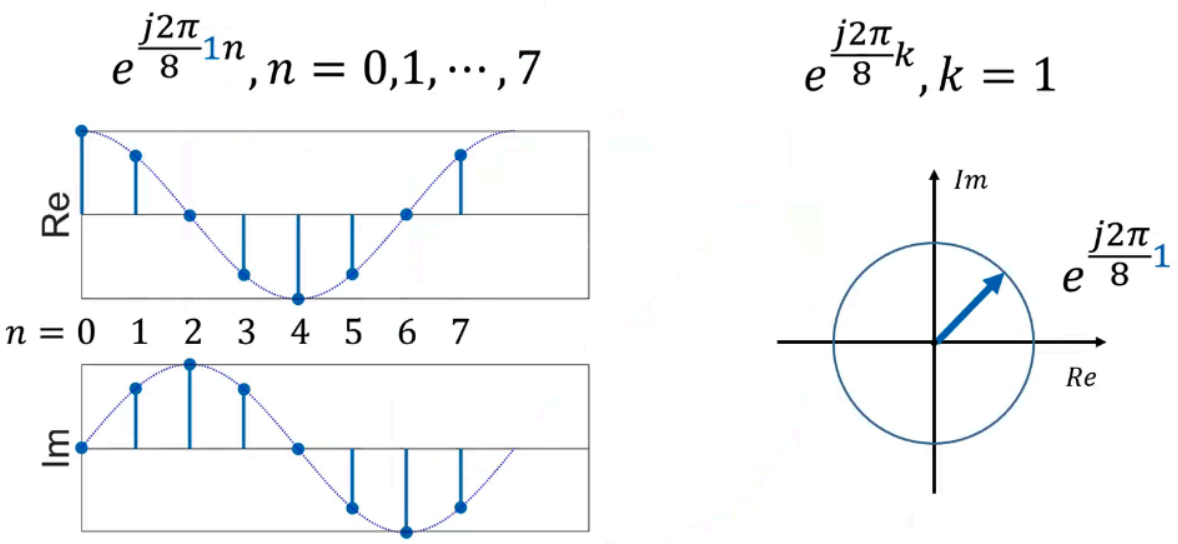
DFT Basis function

For: k=0



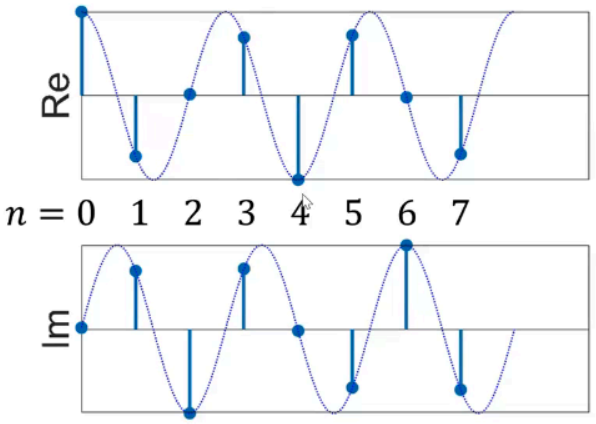
The basis function will always produce real number 1 because everything is on power to 0. The coefficient k couldn't change the direction o the blue arrow. So the number will always show 1 on the real, and 0 on the imaginary.

For: k=1

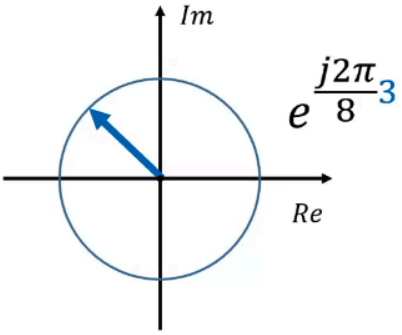


The basis function is k equals one is that the real number (x-axis) shows a cosine function and the imaginary (y-axis) shows a sine function. This reflects that the coefficient will make the blue arrow turn counter-clockwise depends on the coefficient. When n equals 0, the blue arrow will reset to the default position making real =1 and imaginary=0, but when the signal takes 1 step of n, the arrow will have the same position as seen in the image. It moves by 45 degree. This can be seen in the next figures, where if the k increases, then the movement of the arrow will be enlarged,thus showing a higher frequency (the length of the wave gets shortened). Notice that the 45 degree also happen because we divided the sample by 8. When using different value, the signal will also be different.

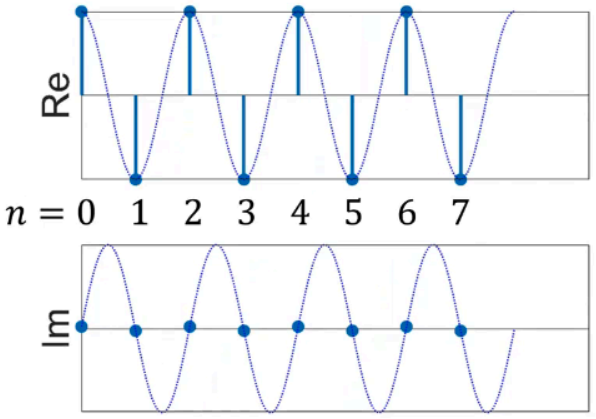
$e^{\frac{j2\pi}{8}3n}, n = 0,1,\cdots,7$



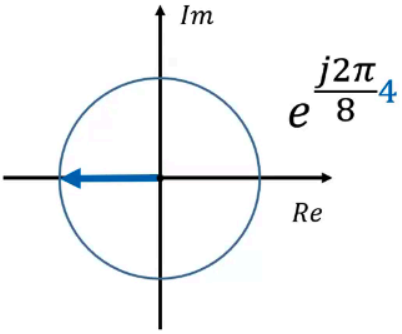
$e^{\frac{j2\pi}{8}k}, k = 3$



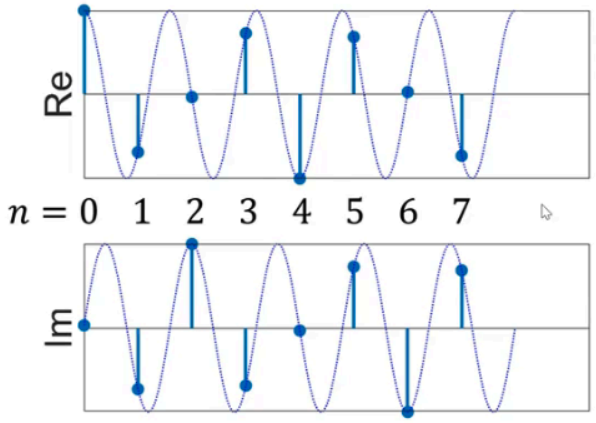
$e^{\frac{j2\pi}{8}4n}, n = 0,1,\cdots,7$



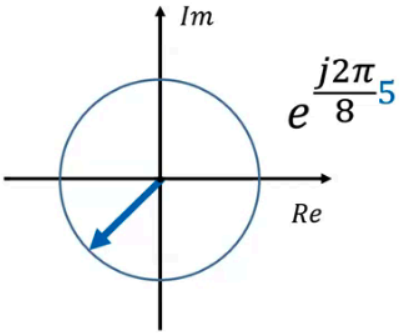
$e^{\frac{j2\pi}{8}k}, k = 4$



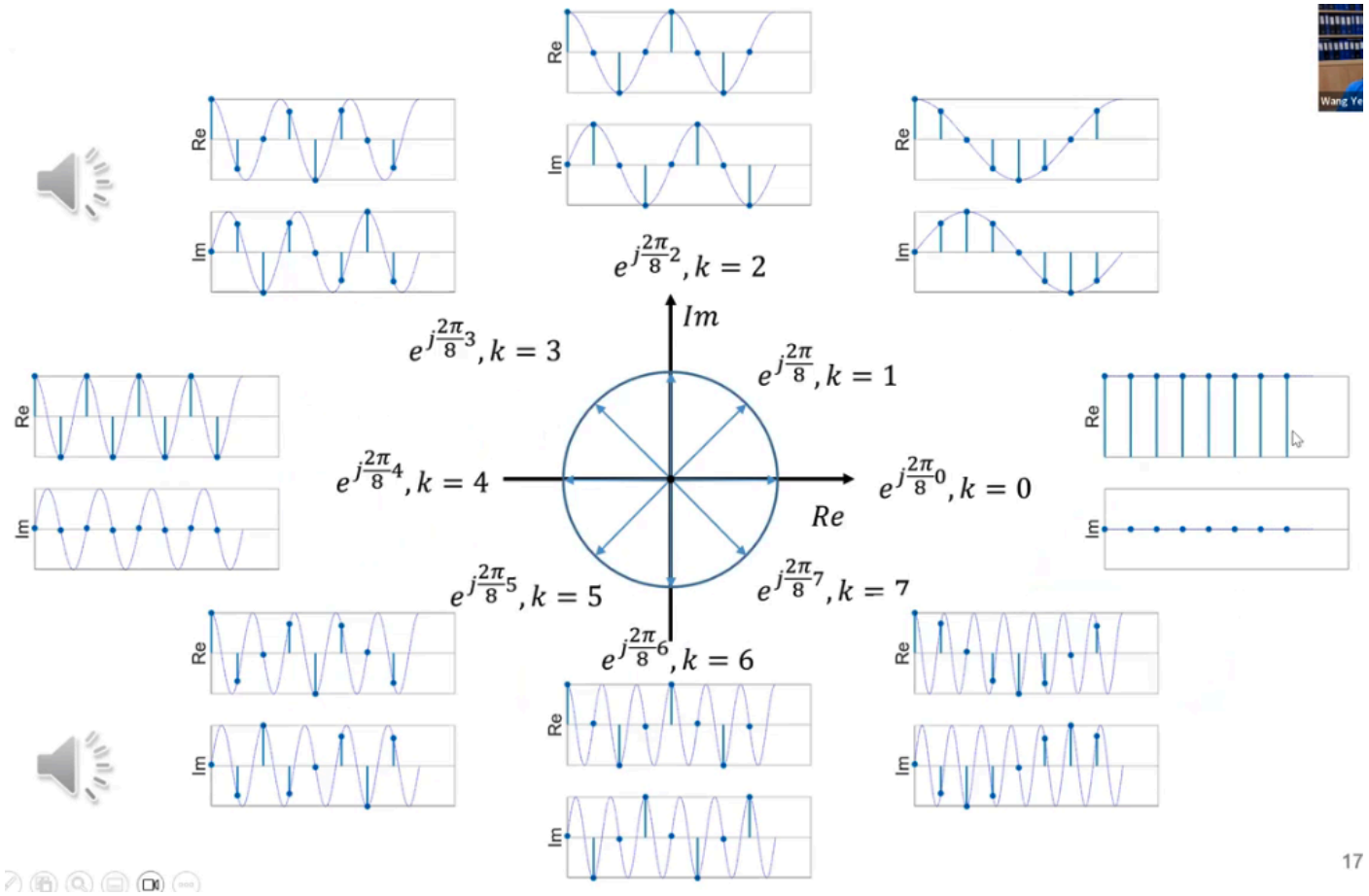
$e^{\frac{j2\pi}{8}5n}, n = 0,1,\cdots,7$



$e^{\frac{j2\pi}{8}k}, k = 5$



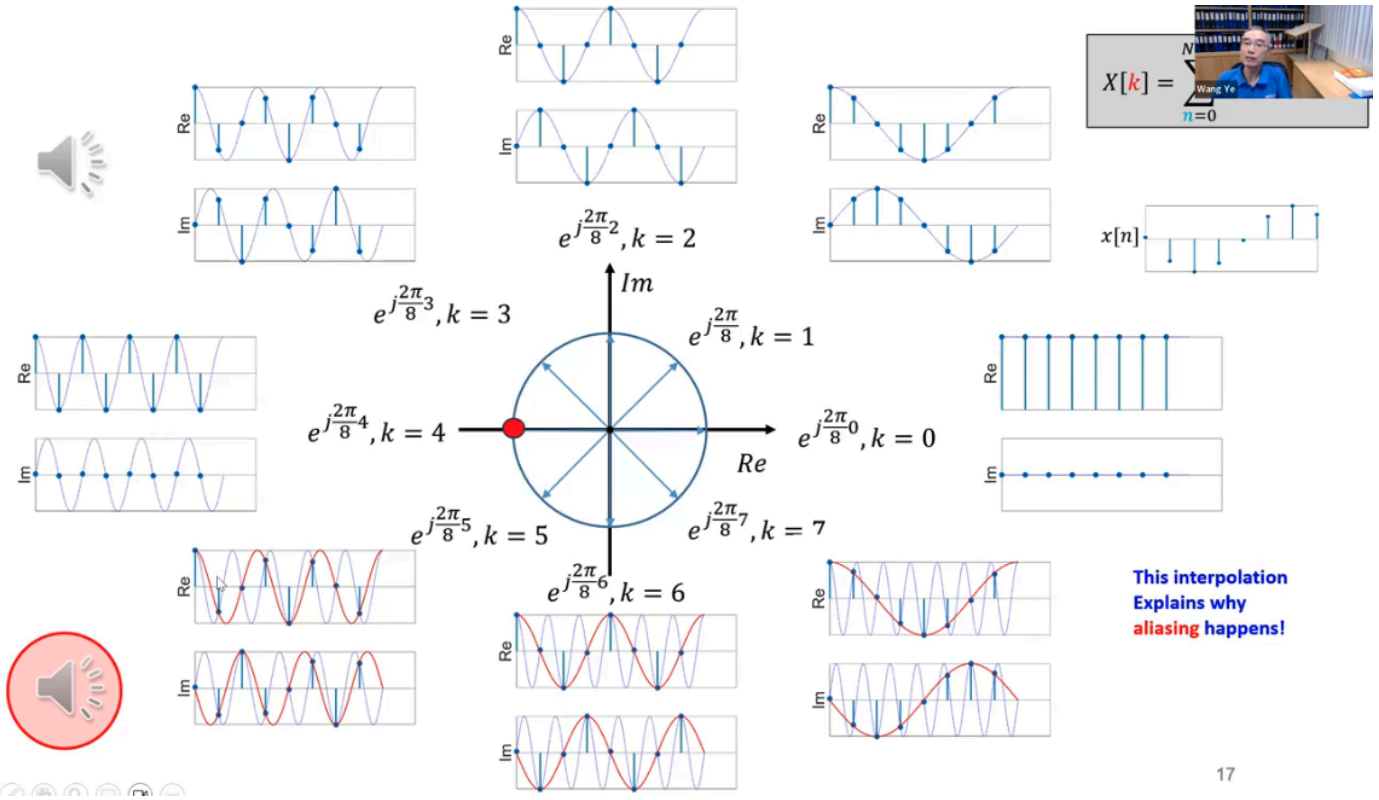
Putting the basis function values all together:



17

Let's say we have x[n], an arbitrary time-domain signal.

If this signal is the exact equivalent signal to k=1 signal, then when we do the dot product, only the k=1 part will have value and the rest of the signals will produce 0.



17

When doing the interpolation, we will see that the discrete point produced when k=5, k=6, and k=7 cannot be used to represent the original signal. Instead of having sound produce in a very frequency (blue line), the interpolation produces a low frequency sound (red line) this is what aliasing is about.

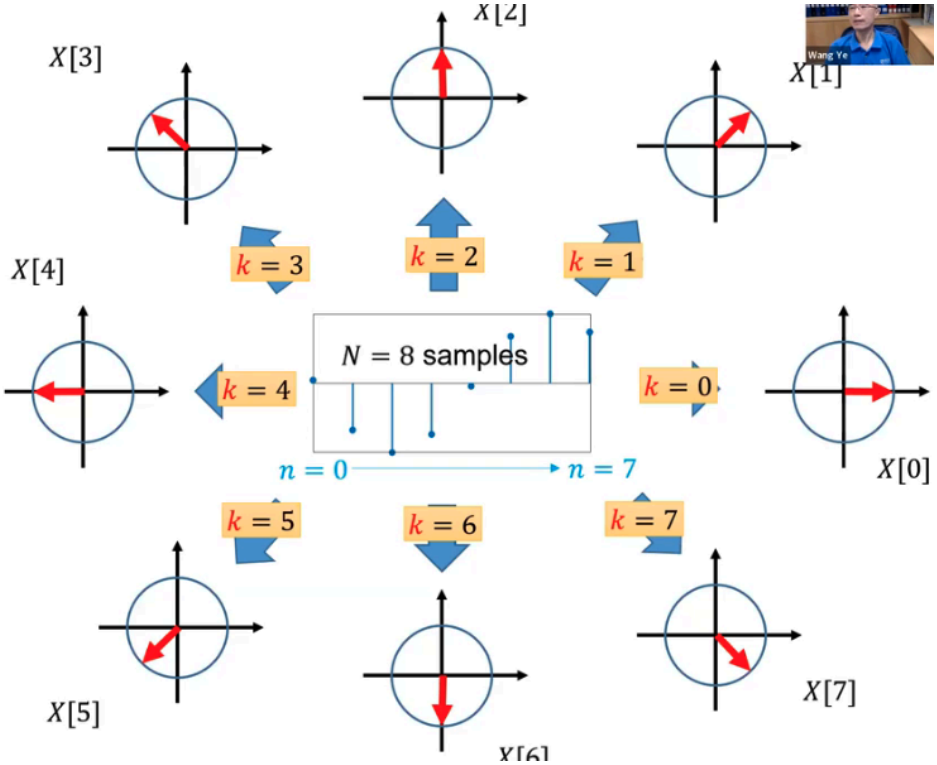
If we look more detail into the red line and compare it to the counterpart. k=1 to k=7, k=2 to k=6, and k=3 to k=5. Then we will see a symmetric property on the real, and inverted symmetric on the imaginary part.

This phenomena explains that when we have a high frequency signal higher across half the nyquist sampling rate, this sample will be mirrored back to the lower frequency range. When we produce the sound that experience the aliasing, whenever the audio is trying to hit a high frequency notes, it will be replace with the mirrored signal which is a low frequency note.

The key insight

DFT is a form of pattern matching between x[n] and each basis function. It computes the similarity between x[n] and each basis function.

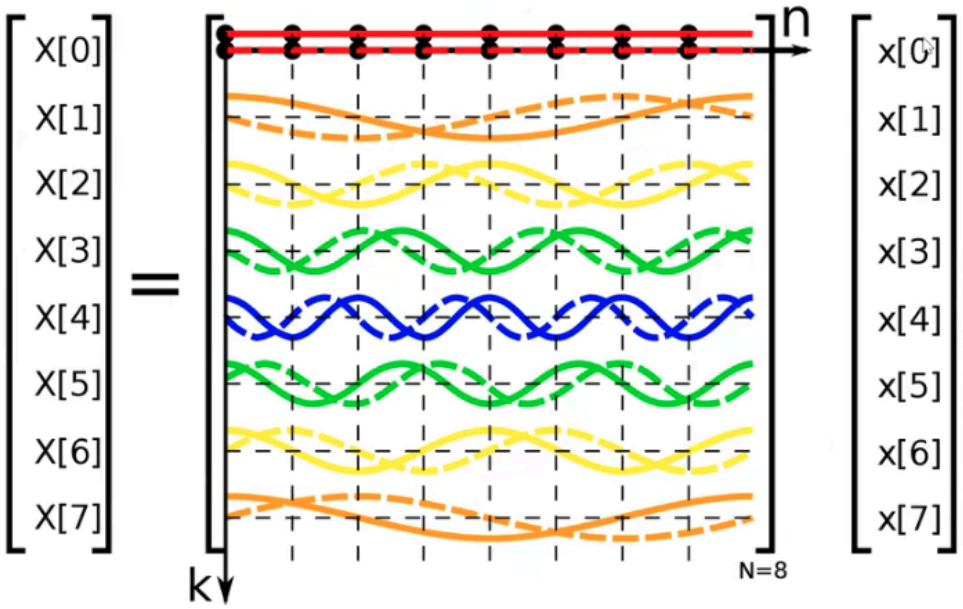
The purpose of DFT is to determine the frequency components of an unknown signal x[n].
visualization:



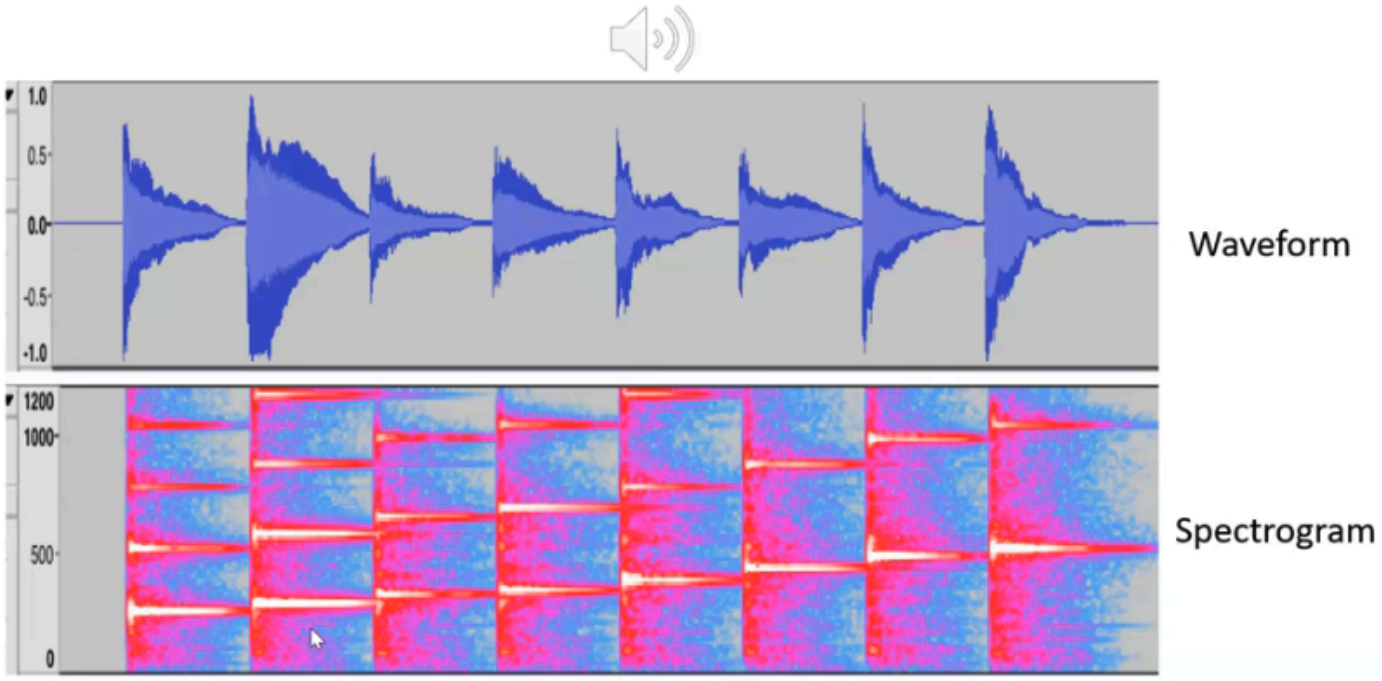
Or with matrix representation (not accurately):

DFT Formula: $X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}$

A Matrix representation



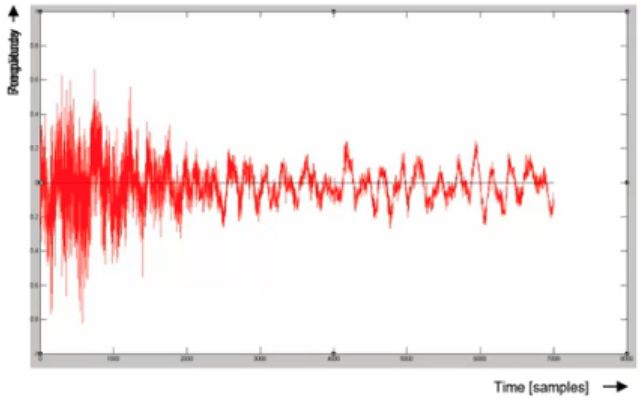
Audio representation in time and frequency domain



The waveform is an audio representation in time domain. Before we playback the audio, we have no clue what the signal is, it could be music, footsteps, or something else. But when we convert it to a time-frequency domain using a spectrogram with the help of DFT, then we can understand that it looks more like a music signal.

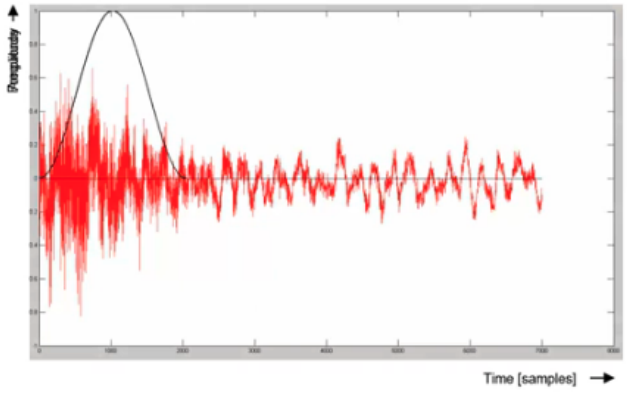
How is a spectrogram constructed?

Let’s say we have this signal which is a time domain signal.

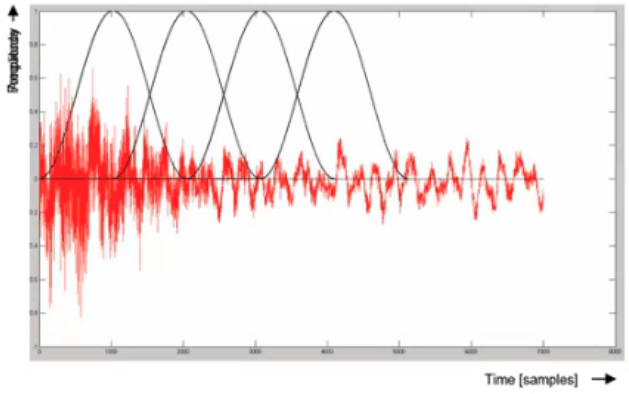


DFT

When we apply a window one of after another, the time shifted from the left to right

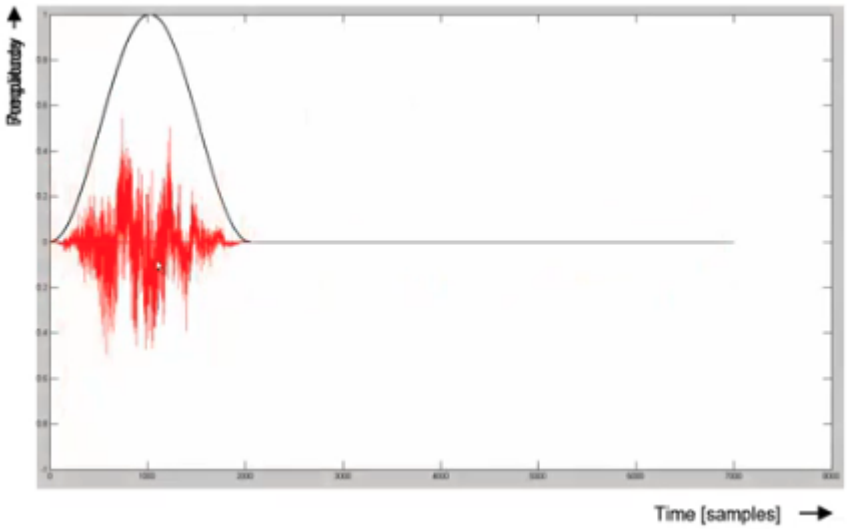


DFT



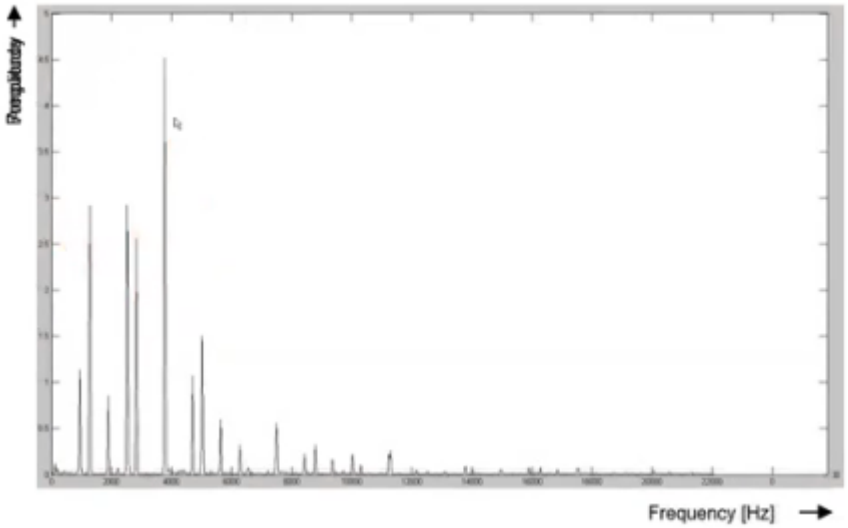
DFT

We’ll get this piecewise signal.



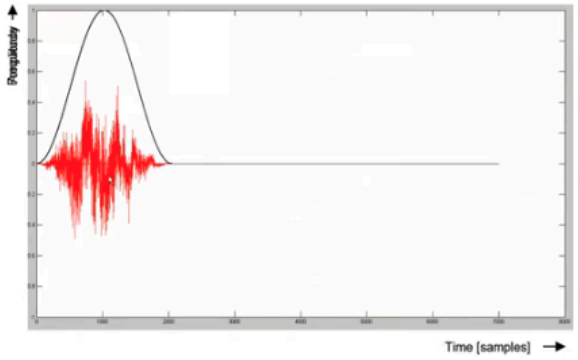
DFT

With that signal, we perform DFT which produce something like this



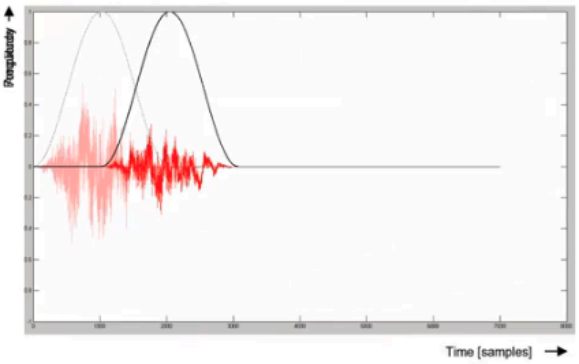
DFT

We then rotate the graph like this



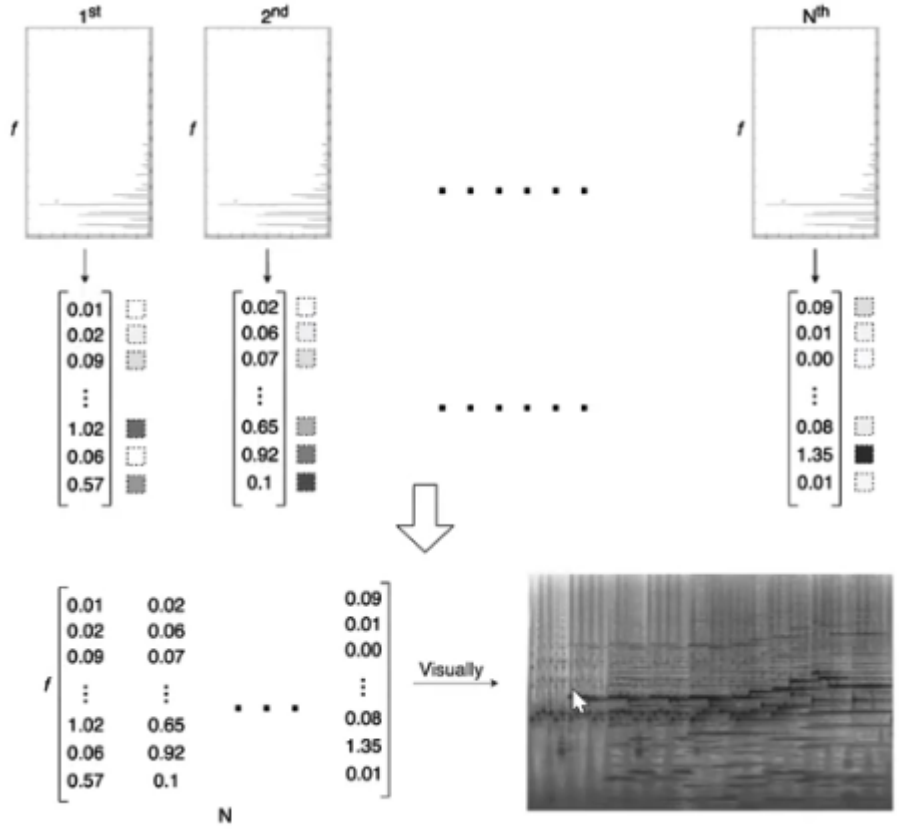
DFT

And then we do it for the rest of the signal



DFT

After getting all the DFT graph, we can concatenate them into something like this



DFT with audio

DFT gives us a complex number for each frequency bin X[k].

For audio we usually care about the log-magnitude spectrum:

$$|X[k]| = \sqrt{Re(X[k])^2 + Im(X[k])^2}$$

$$|X[k]|_{dB} = 20\log_{10}(|X[k]| + \epsilon)$$

|X[k]| is magnitude spectrum
|X[k]|dB is log-magnitude spectrum in dB

X[k] = complex numbers corresponding to frequency bin k
Re = Real part
Im = Imaginary part
∈ = very small number; prevents log(0)

Phase φ[k] are much less commonly-used

$$\varphi[k] = -\tan^{-1}\left(\frac{Im(X[k])}{Re(X[k])}\right)$$

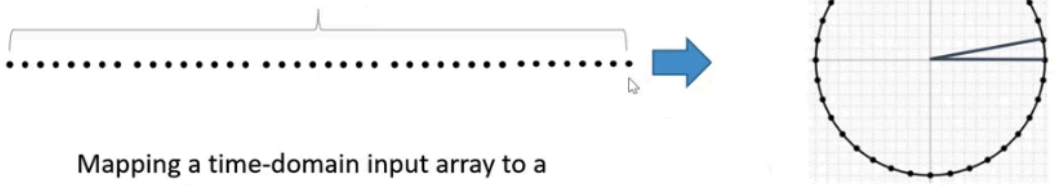
Trade-off of temporal & frequency resolution

DFT gives us information about frequency bins.

N = 2048, 44100 Hz signal → each bins is 44100/2048 = 21 Hz wide; each buffer is 46.4 ms long (this is the period)

The larger the window size (temporal of the time resolution), the smaller the width of frequency bins(frequency resolution)

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



Mapping a time-domain input array to a complex frequency-domain representation

We represent the time domain signal using the sample in the domain (the dots of N samples). The longer the N is, the lower the time resolution. But when we convert this time-domain signal into frequency domain using DFT formula it will show the right hand side (the circle).

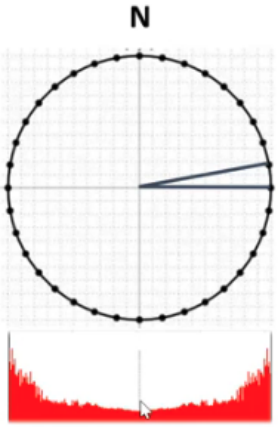
The implication is that the N is an important parameter. Parameter N decides how many slices will be made for the DFT calculation result. It manages the window size.

If the N is small:

- We only look at a short window, the changes between each time frame will be more visible (good time resolution)
- But for frequency resolution, it will cause the out circle to be sparse because the number of grid is very small.

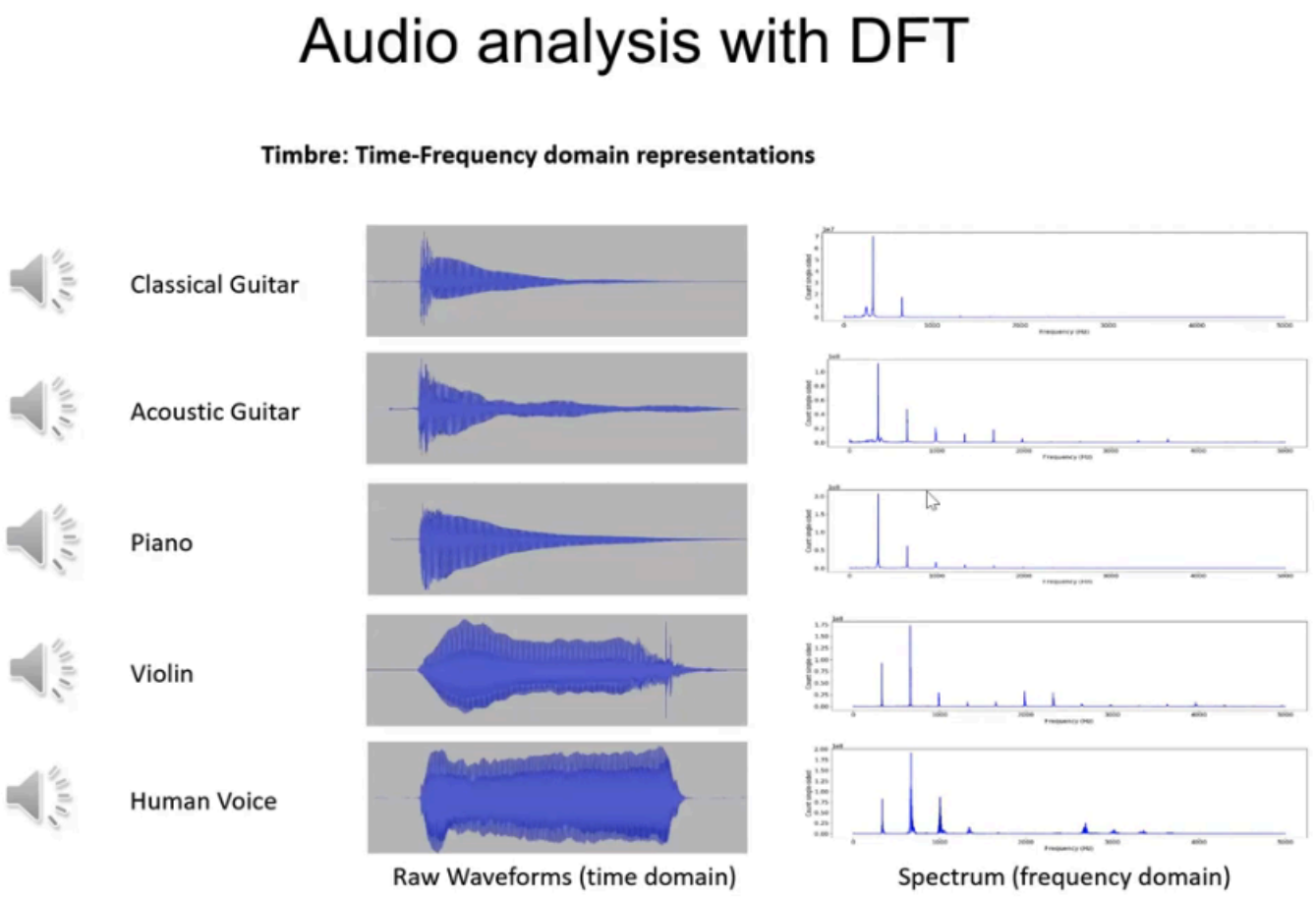
If the N is large:

- We have many sample which cause the DFT to give more frequency points leading to a better separation of the frequencies.
- But since we are looking at a longer window, we lose the ability to see fast changes in time (worst time resolution)



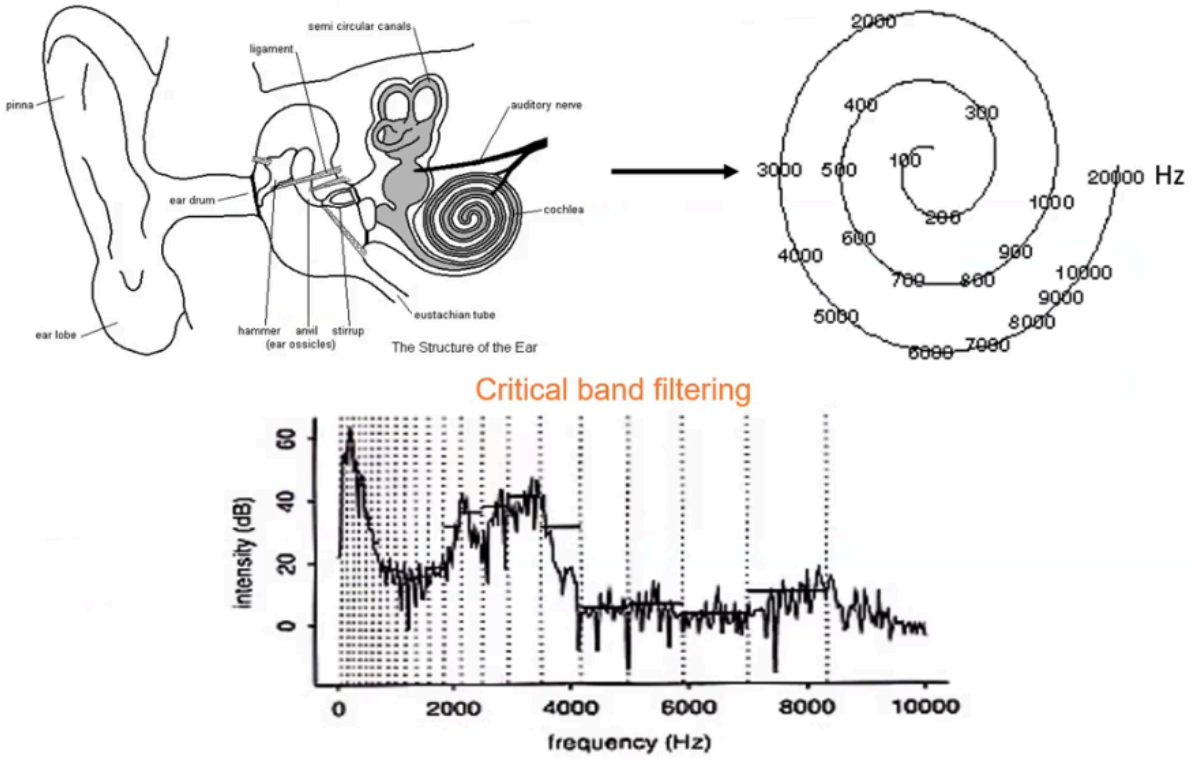
After getting the result of DFT, we can see that because of the symmetry property, we only need to compute half of the spectrum.

Audio Analaysis with DFT



The five sounds above creating sound with the same pitch (note highness) but it creates a different Timbre. We can then see that also the spectrum is very different. The pitch is identical, but the harmonic components are different

Fourier Transform in Our Ear - a First-order Approximation

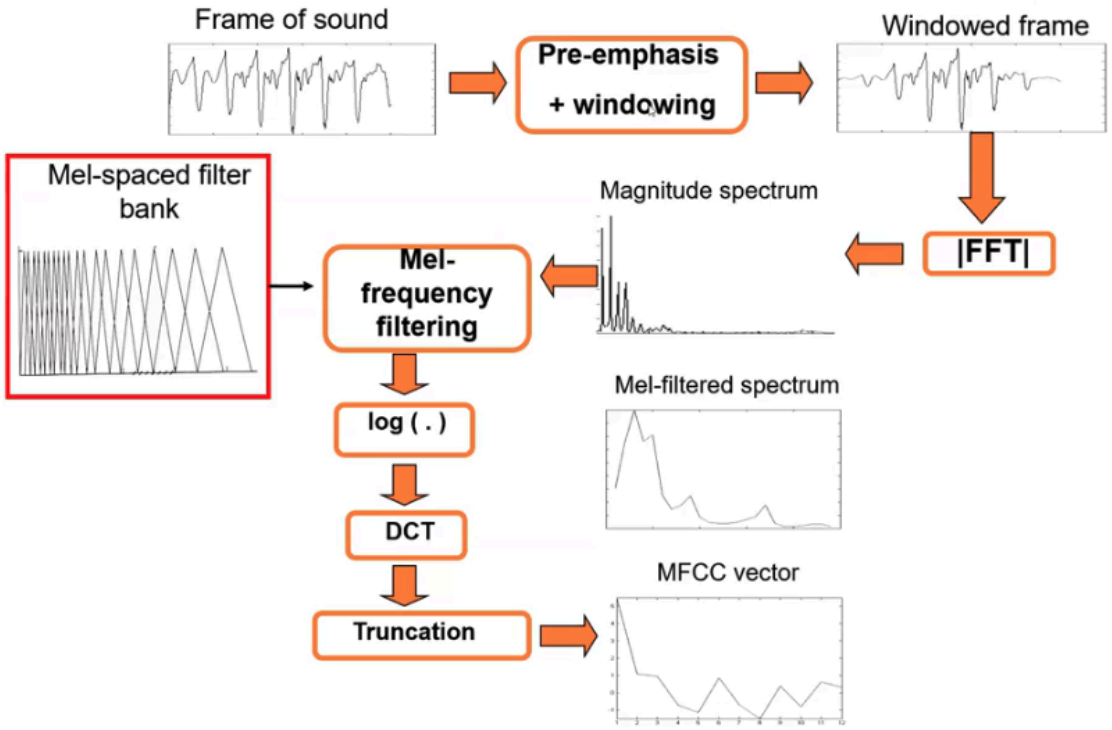


Humans also have a fourier transform in their ear but it is only for a first-order approximation. DFT is a linear transform, but our ear is a nonlinear transform. If we look into the basilar membrane of human anatomy, the position for each frequency is not on the same interval. The part for higher frequency is very small, different from the lower frequency. Which is why humans cannot easily hear sounds lower than 60 and higher than 20000. This concept is called critical band filtering, The benefit is that it has increasing width in the frequencies.

The implementation of this concept is used by the following method

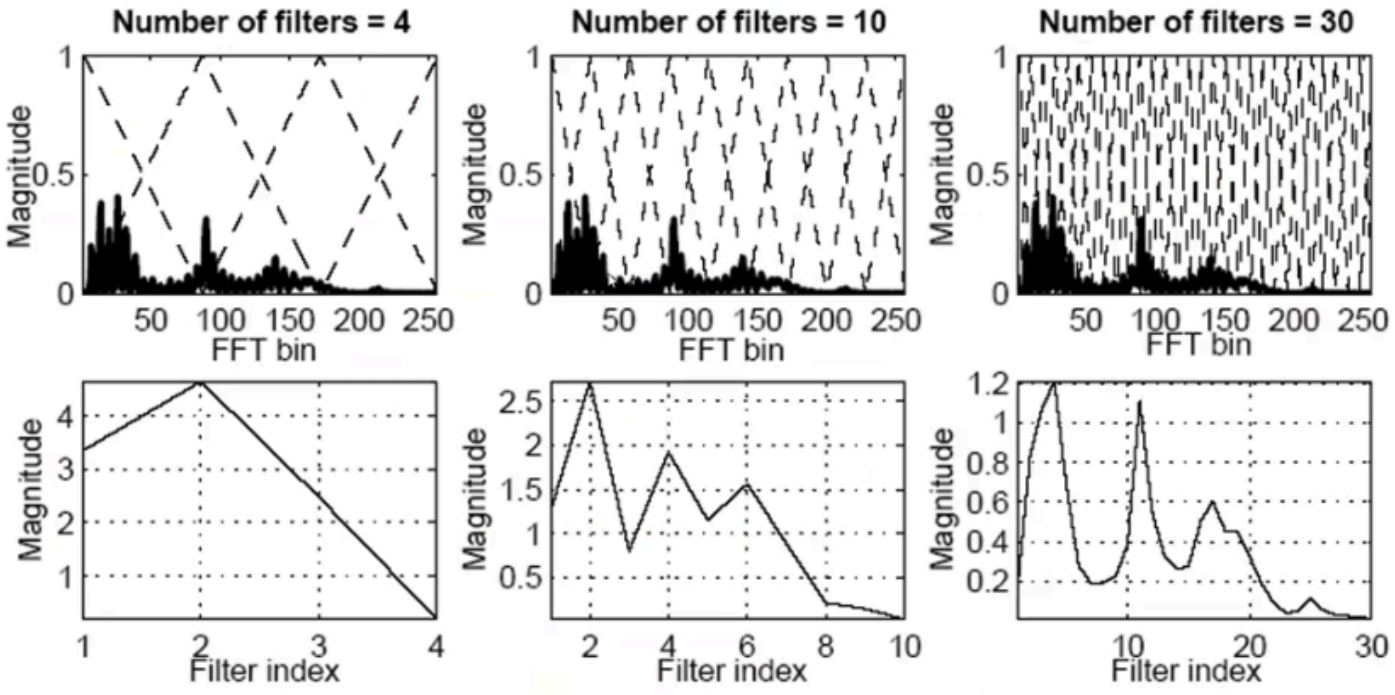
Mel-Frequency Cepstral Coefficients (MFCC) - a perceptual domain audio feature

How do we compute MFCC?



How do we compute the Mel-Frequency Filtering?

Mel-Frequency Filtering produces the mel-spaced filter bank from the input of magnitude spectrum.

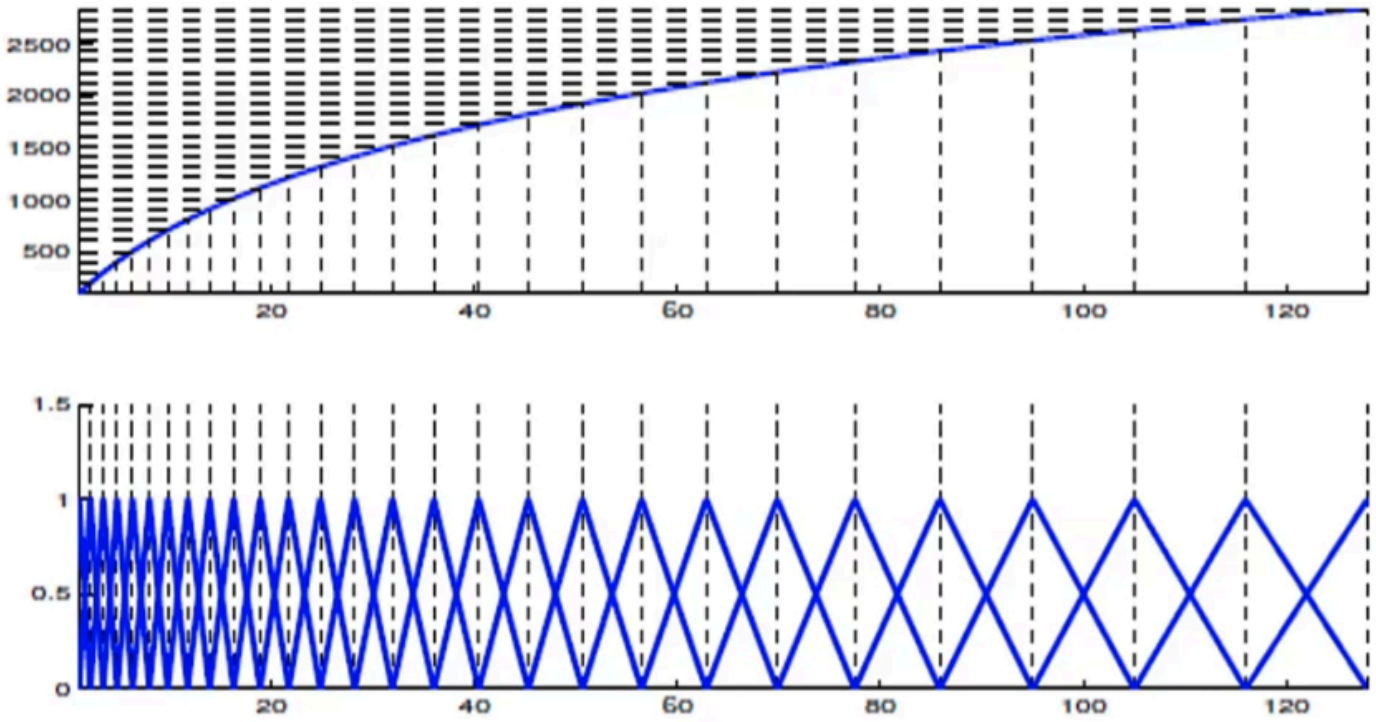


Each filter computes a weighted sum of the FFT magnitude within that frequency band

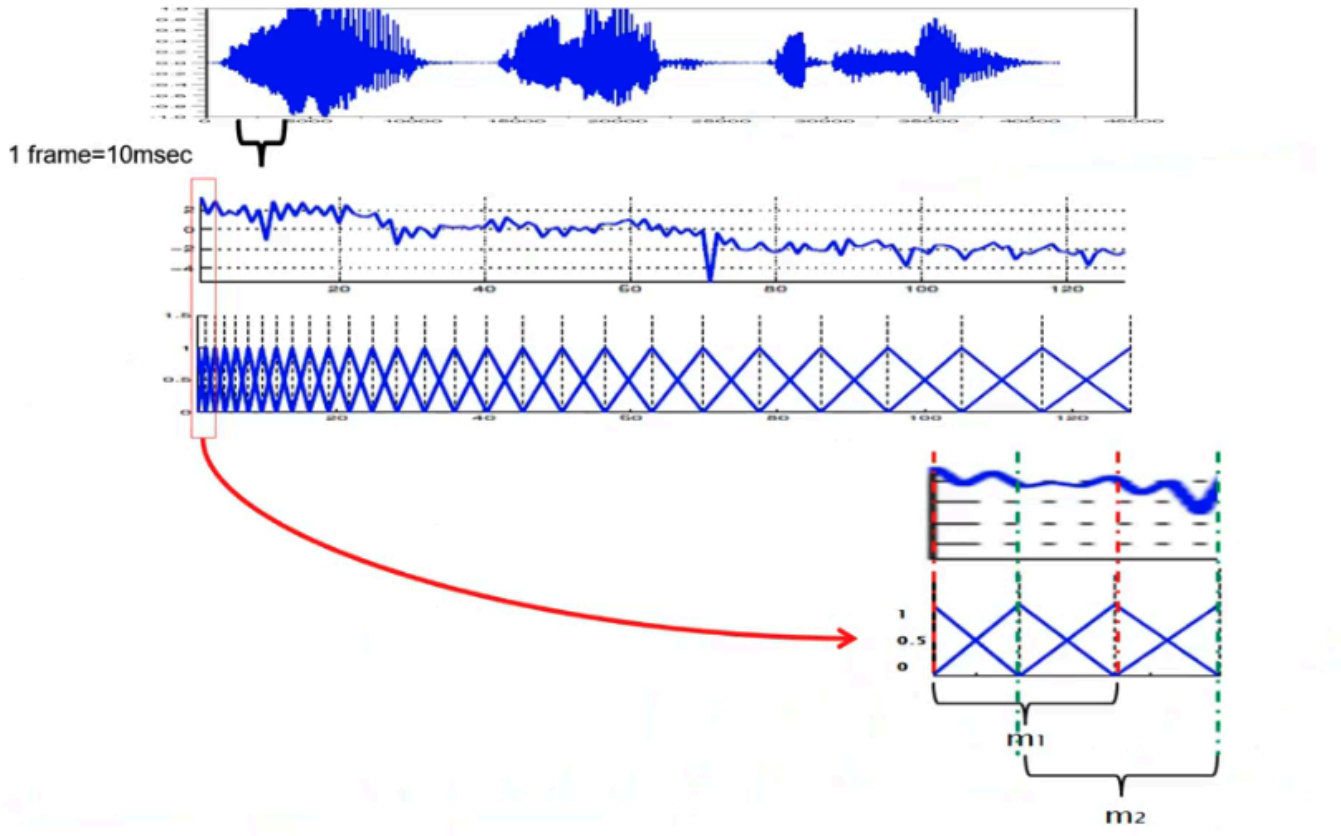
The freq graph is the DFT output. When we choose different number of filters, then the 4 filters is applied by doing point wise multiplication of this dft coefficients and the coefficient of the filter, which result in the triangle. When we increase the number, the MFCC is going to approximate the contour of the DFT.

Mel-Frequency Filterbank:

Mel Scale: $Mel(f) = 1127 \log(1+f/700)$

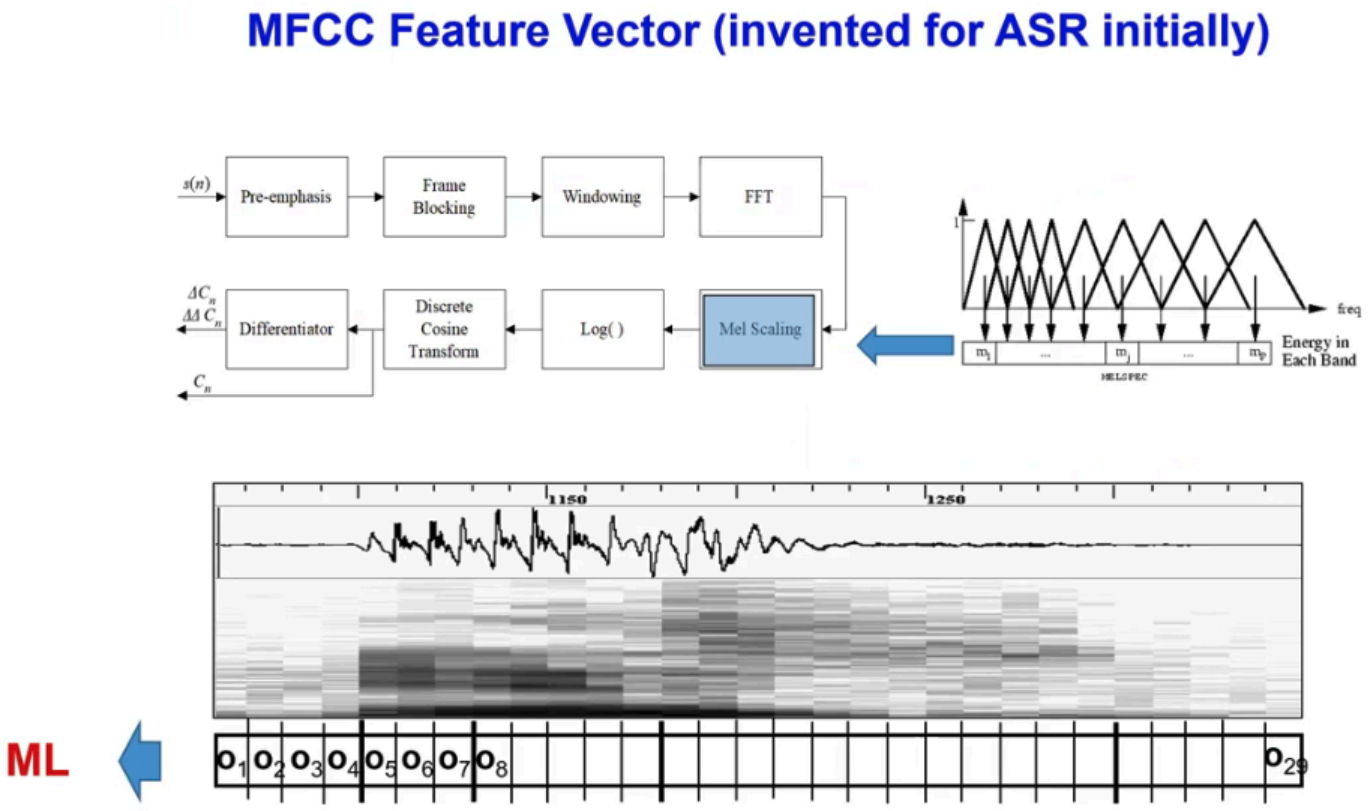


Mel-Frequency Filterbank:

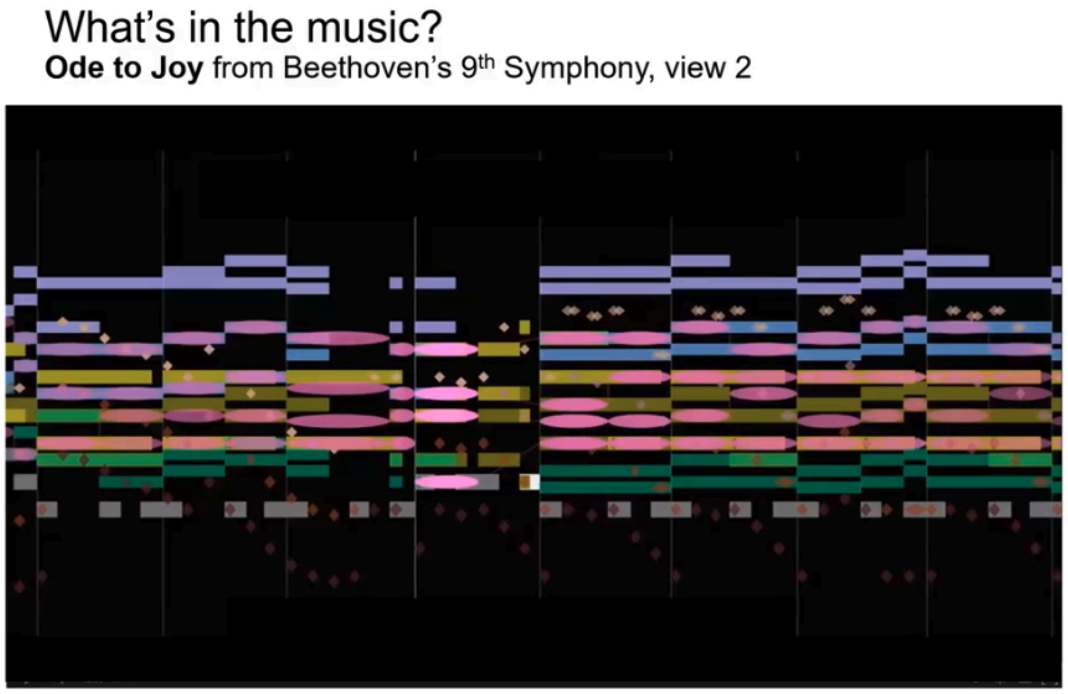


Those are the step in computing the mel frequency spectrum to get the coefficient for each m1, m2 and so on.

MFCC Feature Vector



Here we can see how we have applied this MFCC feature vector for automatic speech recognition application. We can see the original time domain signal that is converted to MFCC. we represent the coefficient with the symbol of o1, o2, o3, o4 and so on. It will direct the feed to machine learning modules to do the speech recognition.



http://www.youtube.com/watch?v=ljGMhDSSGFU&t=12m55s

This is a good representation, that can be used as project idea.