**Bit Depth:** A PCM signal is a sequence of digital audio samples containing the data providing the necessary information to reconstruct the original analog signal. Each sample represents the amplitude of the signal at a specific point in time, and the samples are uniformly spaced in time. The amplitude is the only information explicitly stored in the sample, and it is typically stored as either an integer or a floating point number, encoded as a binary number with a fixed number of digits: the sample's *bit depth.*

The resolution indicates the number of discrete values that can be represented over the range of analog values. So, higher the bit rate, higher the resolution of each sample (amplitude). E.g. if audio is recorded in a bit rate of 16, it means each sample can have one of 2^16 integer values!

**FT vs. DFT vs. FFT**

According to Fourier analysis, any physical signal can be decomposed into a number of discrete frequencies, or a spectrum of frequencies over a continuous range.

Fourier Transform (FT), as known as Continuous-Time Fourier Transform, converts any signal from time domain to frequency domain. If we apply FT to a sine wave, it will show just one line in the spectrogram representing the frequency of the sine wave.

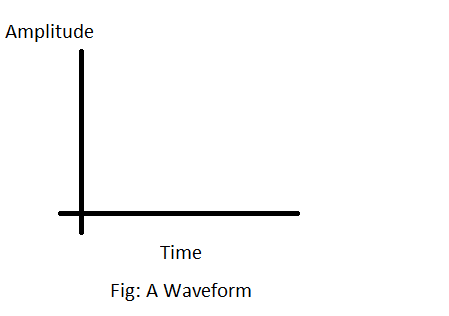
Discrete Fourier Transform (DFT) is the discrete version of the Fourier Transform (FT) that transforms a signal (or discrete sequence) of finite length from the time domain representation to its representation in the frequency domain.

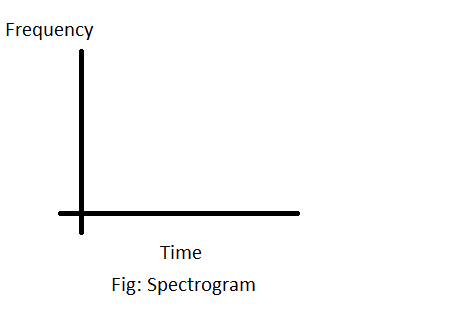
Whereas, Fast Fourier Transforms (FFT) are any efficient algorithms for calculating the DFT. FFTs uses lower number of steps and is useful for large sequences.

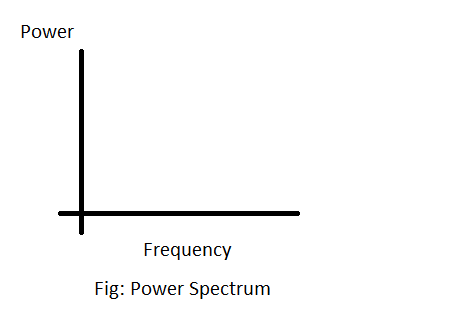
**Waveforms vs. Spectrograms**

We get Spectrogram by applying discrete Fourier Transform (DFT, or FFT) to a Waveform. We basically represent signal into frequency domain from time domain.

Power Spectrum and Power Spectral Density are the same thing. They are used for analyzing stationary signals (which last for infinite time). In our case, each audio frame is a stationary signal





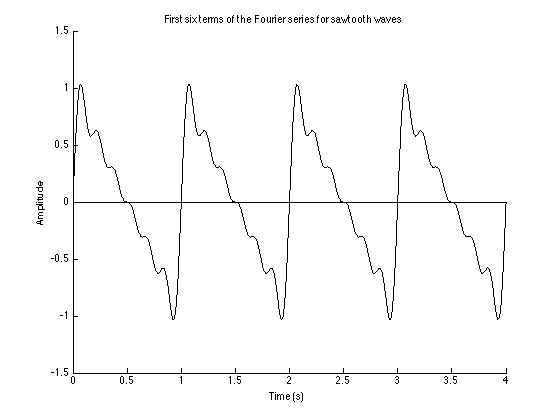
****

**Why do we do framing of audio signal?**

Because we want to apply spectral analysis on audio signals, e.g. FFT. Fourier transform works on stationary (periodic) signals. Audio is a non-stationary (non-periodic) signal. Hence, to apply spectral domain analysis to audio signals we break the audio signals into smaller frames. Each frame can then be considered as a stationary (periodic) signal.

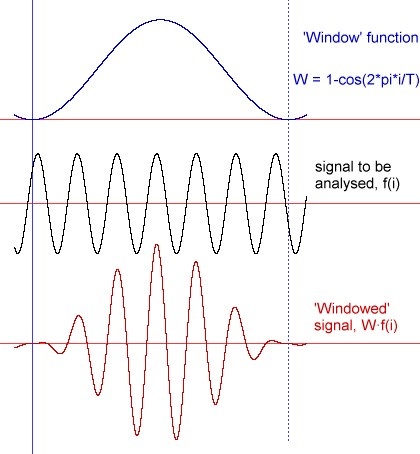
**Why do we do windowing of audio signal?**

Fourier transform treats signals as periodic in nature. That is when DFT/FFT is applied on to a signal the FFT assumes the chunk of signal will repeat itself infinitely (like sinusoids).



For example the audio chunk is assumed to repeat itself like this so there are sharp transitions which will create spurious high frequency noises which will also result in clicks when you listen to the audio which in itself is a separate topic called OLA/OLS (overlapping windows) to get perfect reconstruction of audio.

So in order to minimize this effect or to avoid sharp transitions a windowing function is multiplied with the audio to make the tail end of the audio in that chunk to fall back to zero which will eventually ensure that no high frequency noises exist.



**Number of FFT (N\_FFT):** It is the number of samples on which the short-term Fourier Transform is applied per frame. It is usually equal to the window size, but if window size is smaller, zero-padding is used until window size is equal to N\_FFT. \*That’s why I kept both numbers the same for my thesis!

**Window Size/Frame Length:** The size of each frame of audio (frame is sometimes called analysis **window**!). Standard is 25 ms. Window size unit is time (ms) and frame length has no unit since it’s the number of samples! Frame length is always a power of 2 (like 512, 1024, etc).

e.g. for a sampling rate of 16kHz (i.e. 16000 samples per second),

1000 ms contains 16000 samples

1 ms contains 16000/1000=16 samples

25 ms contains 16\*25=400 samples

**Step Size/Hop Length:** The number of samples between successive frames (the number of samples common between adjacent overlapping frames). In general, more overlap will give more analysis points and therefore smoother results across time, but the computational expense is proportionately greater.

There is a constraint for choosing hop size. This [*constant overlap-add*](http://www.dsprelated.com/dspbooks/sasp/Mathematical_Definition_STFT.html#19930) ([COLA](http://www.dsprelated.com/dspbooks/sasp/Mathematical_Definition_STFT.html#19930)) constraint ensures that the successive frames will overlap in time in such a way that all data are weighted equally. According to this constraint, a Hann or Hamming window can use any hop size of the form R=(M/2)/k. For the Kaiser window, in contrast, there is no perfect hop size other than R=1, where R is the hop size, M is the number of samples in the window, N is the n\_fft (FFT size) and k is a positive integer.

Step size unit is time (ms) and hop length has no unit since it’s the number of samples! Standard step size is 10 ms. Like N\_FFT, hop length Is also a power of 2.

e.g. for a sampling rate of 16kHz (i.e. 16000 samples per second),

1000 ms contains 16000 samples

1 ms contains 16000/1000=16 samples

10 ms contains 16\*10=160 samples

Here, 10 ms is the step size and 160 samples is the hop length.

**NOTE:** In Librosa, the default values for frame\_length and hop\_length are 2048 and 512, respectively. The hop length is one-forth the frame length!

**Window Function in Librosa (used for my Thesis!)**

Librosa uses “Hann” window function by default. Hann is best since it prevents spectral leakage. \*That’s why I used Hann for my thesis.

**Librosa.core.load**

When loading an audio using this method in Librosa, it only stores the audio amplitudes in a numpy array. It does NOT perform framing or windowing in this step! Framing, windowing, and FFT happens when **librosa.feature** class is used to extract a specific audio feature!

What we can change when loading audio using Librosa:

SR, converting signal to mono, offet (start time for audio read), duration of audio to process, data type (default ‘float32’), and resample type (default ‘Kaiser\_best’).

**Dr. Viswanathan said in an email on 27th June, 2020:**

“For speech processing, we typically use 20 ms frame length and 10 ms frame rate (or 10 ms hop length as you call it). In other words, the analysis is done over 20 ms frames at a rate of once every 10 ms.

Unless you are analyzing audio data involving high frequency signal content, the sampling rate used for speech is typically 10 kHZ; some speech recognition systems have used a 16 kHz sampling rate. Unless you are dealing with music signals, use of 48 kHz sampling rate is an overkill and it will increase the computational load. If the original speech data was sampled at 48 kHz for some reason. then one should consider downsampling to 16 or 10 kHz.”

**Bin:** Range of numbers/data, i.e. separated sections.