Fast Fourier Transform (FFT)
Its Applications and Mathematical algorithm

**Definition:** The discrete FFT is an algorithm that converts a sampled complex-valued function of time into a sampled complex-valued function of frequency.

**Why FFT:** An FFT computes the DFT and produces exactly the same result as evaluating the DFT definition directly; the only difference is that an FFT is much faster. (In the presence of round-off error, many FFT algorithms are also much more accurate than evaluating the DFT definition directly.

**Applications of the FFT:**

There are various applications of FFT; we will deal with some of these applications here.

1. The FFT algorithm tends to be better suited to analyzing digital audio recordings than for filtering or synthesizing sounds. A common example is when you want to do the software equivalent of a device known as a spectrum analyzer, which electrical engineers use for displaying a graph of the frequency content of an electrical signal.
2. You can use the FFT to determine the frequency of a note played in recorded music, to try to recognize different kinds of birds or insects, etc.
3. The FFT is also useful for things which have nothing to do with audio, such as image processing (using a two-dimensional version of the FFT).
4. The FFT also has scientific/statistical applications, like trying to detect periodic fluctuations in stock prices, animal populations, etc.
5. FFT are also used in analyzing seismographic information to take "sonograms" of the inside of the Earth.
6. I have even read about Fourier methods used to analyze DNA sequences.

**Problem with FFT:** The main problem with using the FFT for processing sounds is that the digital recordings must be broken up into chunks of \( n \) samples, where \( n \) always has to be an integer power of 2. One would first take the FFT of a block, process the FFT output array (i.e. zero out all frequency samples outside a certain range of frequencies), then perform the inverse FFT to get a filtered time-domain signal back. When the audio is broken up into chunks like this and processed with the FFT, the filtered result will have discontinuities which cause a clicking sound in the output at each chunk boundary. For example, if the recording has a sampling rate of 44,100 Hz, and the blocks have a size \( n = 1024 \), then there will be an audible click every \( 1024 / (44,100 \text{ Hz}) = 0.0232 \text{ seconds} \), which is extremely annoying to say the least.

**Example for using FFT with the sound. (For Your Knowledge).**

My testing signal is composed of three Cosine waveform. Their functions and graphics are below:

\[
dl_g.m\_amplitude\*\cos (2*\Pi*dl_g.m\_frequency*i/pow (2, dl_g.m\_sampling)).
\]

\[
dl_g.m\_amplitude/2*\cos( 4*\Pi*dl_g.m\_frequency*i/pow(2,dl_g.m\_sampling) )
\]
By: Zeyad Al-Hamdany
It’s FFT view is like this

From this FFT view, we can know that signal is composed of four waveforms. By the FFT data we can get its original form like this

And its power view like this

This algorithm works exactly as what we think.
Mathematical Algorithm

1. Decimation In Time (DIT-FFT).
2. **Decimation In Frequency (DIF-FFT).**