**NoteSynth**

**Harit Gedia (B18EE017)**

**Ekagra Das (B18CSE014)**

**Kumar Pradumn (B18CSE024)**

**Aditya Kumar Verma (B18CSE004)**

**Ashutosh Sanodia (B18BB007)**

***Abstract***

Various uses of Cooley Tukey algorithm can be seen in today’s lifestyle such as voice recognition,Using Cooley Tukey Algorithm to extract discrete tones for given audio input. Applying filter to remove all the noise and also the duplicate frequencies and then plotting the results giving us the final key versus time graph or data to be sent to the music synthesiser to produce the desired piano tones.

**1. Introduction**

**1.1 Background**

Having learnt piano, we know the difficulties that people face during creations of new rhythms or converting existing rhythms to piano or any other instrument. Thus we decided to work in project called NoteSynth, a simple yet elegant tool that converts audio-signals into their respective piano tunes which, then is played by an audio synthesizer. It's extremely useful for pianists, music directors, and composers as they can play and direct the notes accordingly, or even check the synchronization of their tune with other instruments and lyrics.

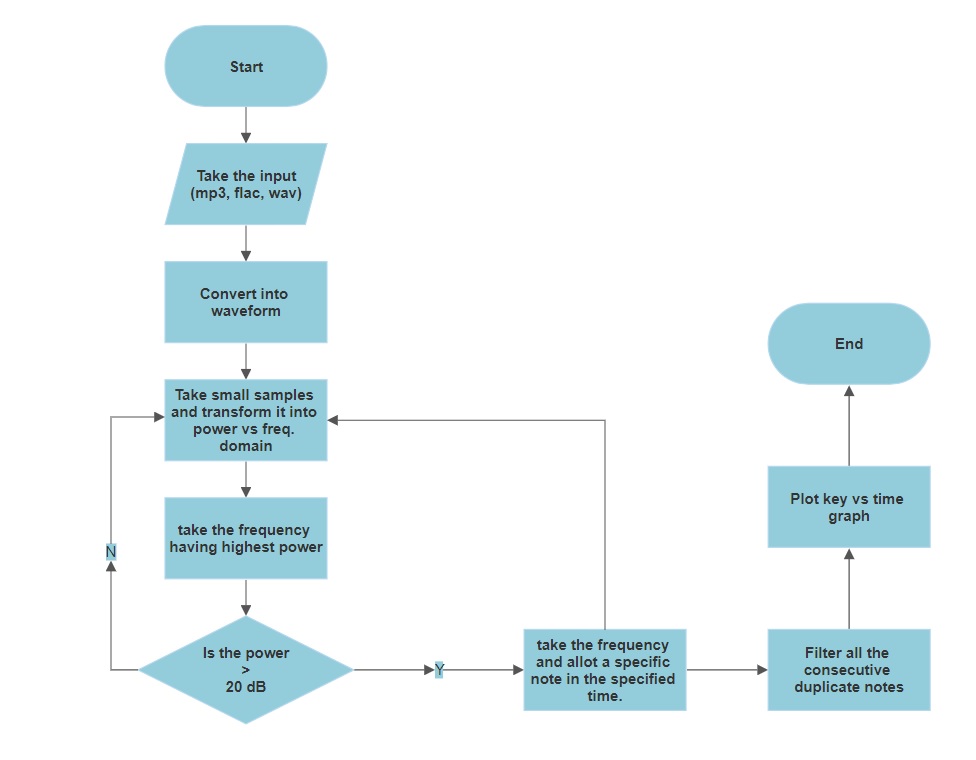
We've chosen to develop this project to demonstrate the usefulness of the Cooley-Tukey Fast Fourier Transformation, one of the most indispensable algorithms in Digital Signal Processing, which will help us convert the time domain input signal to the frequency domain.

**1.2 Problem Statement**

The objective of this project was to convert audio files into a collection of tunes with the help of mathematical concepts and concepts we learned from the course Signals and Systems.

**1.3 Tools**

We have used MATLAB for the project to take input and process algorithm.

**2. Extraction of piano notes from the input**

1. Taking the input and converting into a waveform file.
2. Run a loop to take small samples of the waveform files for better analysis of the sample,
3. Apply Fast Fourier Transform using the Cooley Tukey Algorithm to the sample to create intensity vs frequency matrix.
4. Take the frequency with the highest power in the particular sample Check whether it’s intensity is greater than acceptable range. If no, then return to the previous step to check the next sample. If yes, then go to the next step
5. Assign the frequency to the specific note and form a table consisting of the time, note and its frequency. If All samples are taken, go to the next step.
6. Remove all the duplicate inputs if they are in consecutive time frames.
7. Print the final table and plot the graph of specific keys vs time.

**2.1 Cooley Tukey Algorithm** [1]

The Cooley-Tukey algorithm is one of the fastest and most common algorithms for fast fourier transform. It breaks the samples to smaller parts and calculates their separate discrete Fourier transforms and hence, reduces the computations done by a big margin.

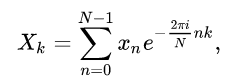
Thus, we can say that it re-expresses theDFT of an arbitrarycomposite size *N* = *N*1*N*2 in terms of *N*1 smaller DFTs of sizes *N*2,recursively, to reduce the computation time to O(*N* log *N*) for highly composite *N* (smooth numbers). Because of the algorithm's importance, specific variants and implementation styles have become known by their own names, as described below. A direct DFT for the whole samples would have taken N2 computations.

Because the Cooley–Tukey algorithm breaks the DFT into smaller DFTs, it can be combined arbitrarily with any other algorithm for the DFT.

**2.1.1 The radix-2 DIT case[2]**

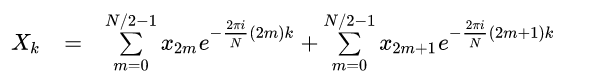
A radix-2 decimation-in-time (DIT) FFT is the simplest and most common form of the Cooley–Tukey algorithm. Radix-2 DIT divides a DFT of size *N* into two interleaved DFTs (hence the name "radix-2") of size *N*/2 with each recursive stage.

The discrete Fourier transform (DFT) is defined by the formula:

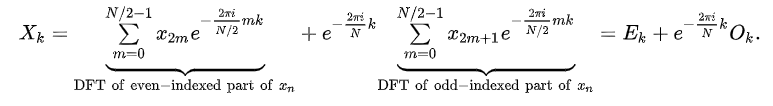


where k is an integer ranging from 0 to N −1.

Radix-2 DIT first computes the DFTs of the even-indexed inputs ( x 2m = x 0 , x 2 , … , x N − 2 ) and of the odd-indexed inputs ( x 2 m + 1 = x 1 , x 3 , … , x N − 1 ), and then combines those two results to produce the DFT of the whole sequence. This idea can then be performedrecursively to reduce the overall runtime to O(*N* log *N*). This simplified form assumes that *N* is a power of two; since the number of sample points *N* can usually be chosen freely by the application (e.g. by changing the sample rate or window, zero-padding, etcetera), this is often not an important restriction.

The radix-2 DIT algorithm rearranges the DFT of the function xn into two parts: a sum over the even-numbered indices n = 2m and a sum over the odd-numbered indices n = 2m + 1:

One can factor a common multiplier out of the second sum, as shown in the equation below. It is clear that the two sums are the DFT of the even-indexed part  and the DFT of odd-indexed part  of the function xn. Denote the DFT of the Even-indexed inputs  by Ek and the DFT of the ***O***dd-indexed inputs by O k and we obtain:

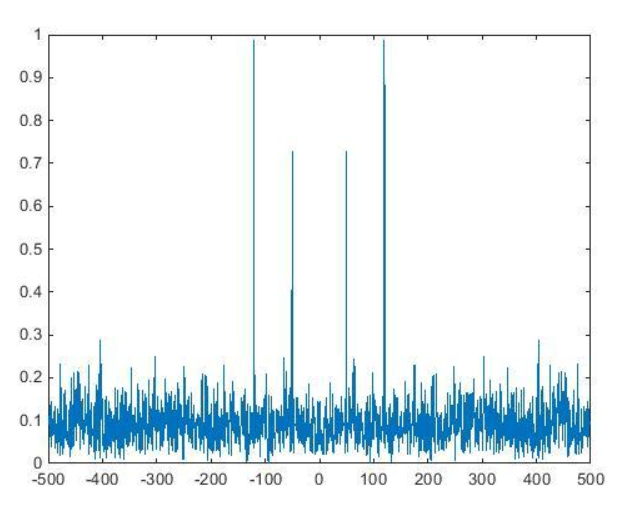


**3. Results and Observations**

**3.1. Observations**

When applied FFT, we get a complex value per bin-frequency whose absolute value gives us the intensity of that particular bin frequency. Also,we get positive as well as negative frequencies and this is the reason why we get a double sided spectrum.

As we get bin frequency in the x-axis, we must multiply the index with nyquist frequency to get the original frequencies.

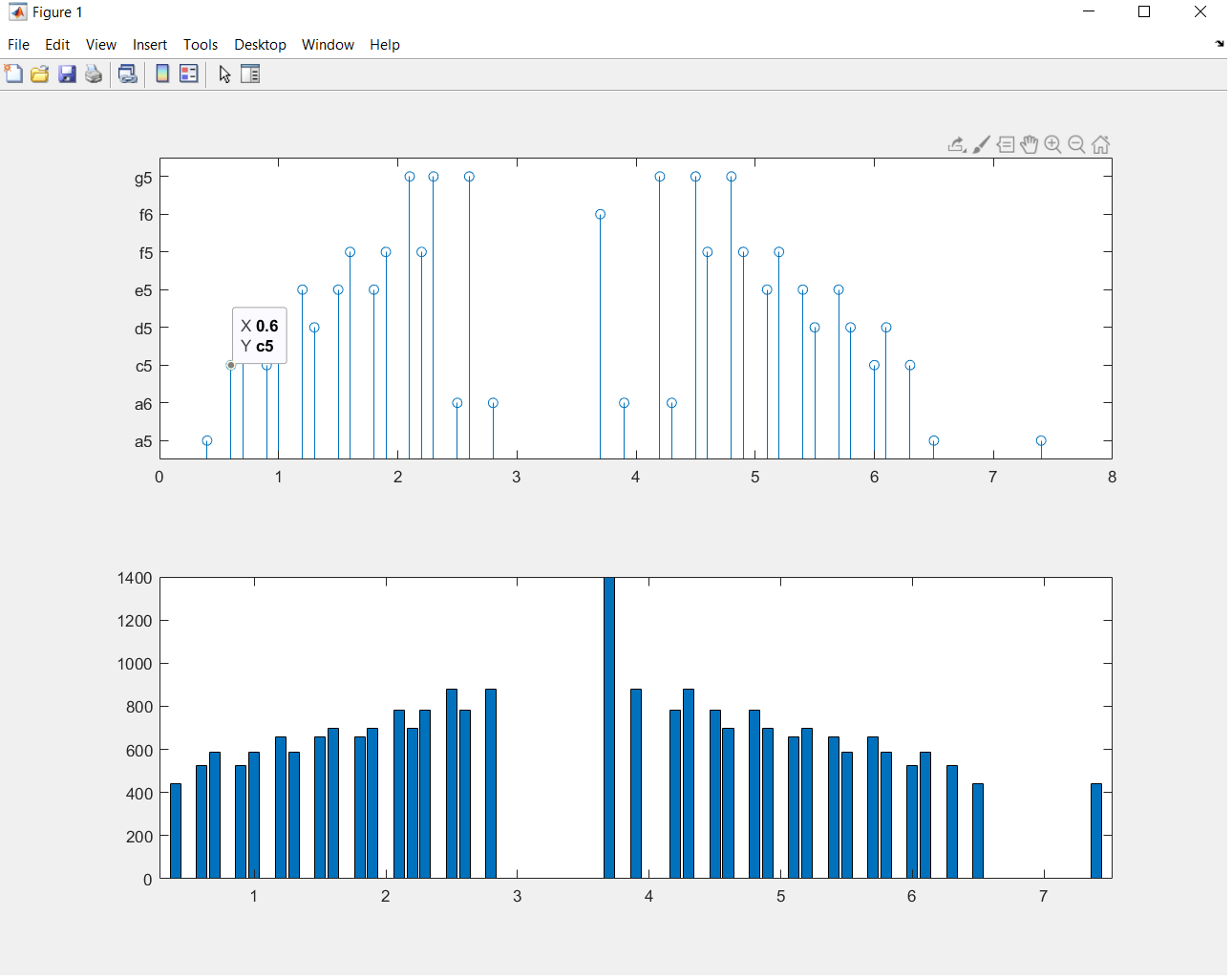
Addition of noise does not make much difference in the output required it must not be loud enough.

At certain times where no voice input is there, the maximum amplitude is occupied by the frequencies of the noise which needs to be filtered.

When the same frequencies occupies maximum intensity again in the next consecutive samples, the output piano note is recurred which is not acceptable. Thus consecutive repetition was required to be removed.

**3.2.Results**

Plot and script of piano notes versus time depicting the piano tones to be played at its respective time.



**4. Conclusions and Limitations**

**4.1.Key Finding:->**

1.> Cooley Tukey algorithms compute the DFT of an array of size N in O(N log(N)) time which is significantly faster as compared to the direct algorithm which is O(N2).

2.> Because of the algorithm taking only the frequency with maximum amplitude the noise need not be filtered. Thus saving memory space and time for processing. As this takes a single input, the accuracy of the output tone matching with the input voice is increased by a sufficient margin. Besides human input, it can take any other voice input in the given range.

3.> This algorithm is approx. 90% accurate in the case of high pitched and clear voices (like flute) and 40-70% accurate in the case of voice input depending on the recording instrument or the person’s ability.

**4.2.Limitations:->**

1.>As our algorithm takes the frequency with the highest amplitude, it may happen that it fluctuates between two nearby node ranges thus giving an output that may differ from the desired output.

2.>It may happen that we get piano notes of noise if noise passes our noise filter i,e any sound sample with sound amplitude greater than 20db and frequency lower than 5933 may be allowed to pass the filter provided that its frequency has the greatest amplitude in the sample.

3.>This algorithm detects only single voice input, so if the input is coming sources, the only one with higher amplitude will be recorded.

4.> Due to miscellaneous factors like person’s ability etc. can play a role in distorted output.

**4.3 Future Scope->**

1.>Besides piano tunes, several other instruments like drums, violin, guitar, etc. can be synthesized by the human voice or any other voice input.

2.>Multiple inputs from different sources can be taken as input to create a better musical experience.

**References**

[1].Cooley, James W.; Tukey, John W. (1965). "An algorithm for the machine calculation of complex Fourier series". [*Math. Comput.*](https://en.wikipedia.org/wiki/Mathematics_of_Computation) **19** (90): 297–301. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.2307/2003354](https://doi.org/10.2307%2F2003354). [JSTOR](https://en.wikipedia.org/wiki/JSTOR) [2003354](https://www.jstor.org/stable/2003354).

[2].Danielson, G. C., and C. Lanczos, "Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids," *J. Franklin Inst.* **233**, 365–380 and 435–452 (1942).