Mathematical physics Project Report

Name: Vidhi Tailor	Name: Kunal Singh	Name: Nilay Ghalsasi
UID: 219113	UID: 219114	UID: 219122
Roll no: 08	Roll no: 09	Roll no: 14

Application of Fourier transforms on audio signals

Theory

Introduction:

Periodic function:

The functions which repeat themselves in a finite interval of time are called periodic functions.

The function f(t) is periodic if and only if: f (t + T) = f(t), here T is the period.

We encounter periodic functions on a daily basis, but there are many functions which are non-periodic in nature which can be represented by a sum of periodic functions (sine, cosine). Fourier series helps us represent any function as a sum od sinusoidal functions.

Fourier series:

It is defined as expansion of function or representation of function in the form of sine and cosines.

Mathematically it is given as follows:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right)$$

Here a_0, a_n and b_n are called the Fourier coefficients, they are given by the following formulae:

$$a_0 = \frac{1}{\pi}\int_0^{2\pi} f(s)ds, \quad n = 0,1,2$$

$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(s)\cos(ns)\,ds, \quad n = 0,1,2$$

$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(s)\sin(\mathrm{ns})ds, \quad n = 0,1,2$$

For the above formulae to be valid the following condition must be satisfied by the function. The condition is called the Dirichlet's condition for Fourier series.

f(x) for the interval of (-π, π):

- The function f(x) should be single valued
- f(x) should be bounded
- It should have at most finite no. of maxima and minima
- It should have finite number of discontinuities

Thus, Fourier series is applicable for both continuous and discontinuous functions

Integral transforms:

Integral transform maps a function from its original function space onto another function by performing integration on the original function.

Mathematically: $g(x) = \int_a^b f(t)K(x,t)dt,$

Here a, b and K (x, t) are same for all function pairs of f and g. K (x, t) is called the kernel.

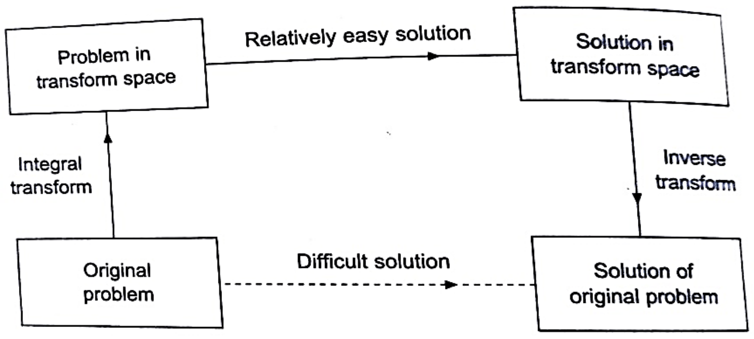
The above equation can be written as follows in operator form: g(x) = φ f(t)

The function g(x) is called the integral transform of f(t) and φ is called the operator. The transform is determined by the specific type of a, b and the kernel (K (x, t)). For the transform to be useful, its inverse must also exist. This just does not mean that we simply have and equation of type, φ^{-1} g(x) = f(t). The inverse should also be calculable (easily).

Why use integral transform?

The answer is as follows:

1. Even though the integral transformations may look difficult to evaluate, they actually make the problem easy. this is shown by the following schematic diagram



The schematic of integral transform is taken from the following book: "Mathematical methods for physicist" by "Arfken, Weber and Harris" seventh edition.

2. Sometimes it is easier to analyse data in different domain and integral transform helps in doing so.

Fourier transform:

Mathematically: $g(\omega) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(t)\mathrm{e}^{\mathrm{i}\omega\mathrm{t}}dt$

Here the ω is assigned to the transformed variable. This is done because while studying signal processing we transform the signal from time domain to one by time domain or frequency domain. This helps us in analysing the signal in a more informed way.

Difference between Fourier series and transform:

The difference between Fourier Series and Transform is that the former is applied on a periodic function whereas the Fourier Transform is applied on a non-periodic function where the magnitude tends to zero as time or space tends to infinity.

Inverse Fourier transform:

Mathematically: $f(t) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} g(\omega)\mathrm{e}^{-\mathrm{i}\omega\mathrm{t}}dt$

Here as we can clearly see that we transform the function from frequency domain to time domain.

Importance of Fourier transform:

A normal signal is in the format of Amplitude vs time, this does not give us any information about the frequencies, which is important for analysing the data in a better way. Thus, what Fourier transform does is it converts the amplitude vs time plot of data into PSD vs frequency plot.

Gabor transform:

The Fourier transform is the most widely used tool for analysing frequencies of a given waveform, the disadvantage of this is that the time information is lost after the transformation takes place and it gets hard to say when frequency occurred at which time. Decomposing signal in such a manner, where it is decomposed into a number of elementary waveforms that are localised in time and frequency plays an important role in signal processing, and can help reveal important data for analysing non-stationary signals such as music and speech. Gabor with the aim of localizing frequency components of sounds, introduced the "windowed Fourier Transform" also known as Gabor transformation. The windowed function for extracting the local information from Fourier transform of a signal is given by: $g_\alpha(t-b)$, where α = width of the window and b = represents the translation of window over the time domain.

The basic idea is to use the window function to localise Fourier transform and then shift the window to another position, repeating this over the length of the signal. It localises the Fourier transform to t = b.

The Gabor transform is given by: $$(G_b^\alpha f)(\omega) = \int_{-\infty}^{\infty} \left[e^{-i\omega t} f(t)\right] g_\alpha(t-b)dt,$$

If $g_\alpha(t)$, denotes a gaussian function then, $$g_\alpha(t) = \frac{1}{2\sqrt{\pi\alpha}} e^{-t^2/(4\alpha)}, \alpha > 0$$

DFT (discrete Fourier transform):

Discrete Fourier transform is a highly important tool in digital signal processing, with many applications, e.g. DFT is used to calculate a signal's frequency spectrum. There are many signals in which the shape of the time-domain waveform does not give us much information, such as in human speech, changing brightness of stars and planets as they rotate and revolve around each other, in such cases the key information lies in the frequency, phase and amplitude of the sinusoidal component. DFT is used to extract this information.

DFT is a numerical method for performing Fourier transform. In DFT discrete time data is converted into discrete frequency output.

It converts a finite sequence of equally-spaced data points of a function in time domain into a sequence of same length, of equally spaced DTFT (discrete time Fourier transform) data points, which is a complex function of the frequency domain. The sampling interval of DTFT is the inverse of the duration of the input sequence. If the input sequence has all non-zero value of a function, its DTFT is continuous and periodic, and DFT provides discrete samples of one cycle.

Suppose, we take a vector of n input amplitudes, $\{f_0, f_1, f_2, \ldots\ldots, f_{n-2}, f_{n-1}\}$, performing DFT on the vector gives us a set of n frequencies.

DFT is defined as:

$f\,[\,i\,] \;=\; \sum_{n=0}^{N-1} x(\,j\,)\, e^{-k2\pi ij/N}$, x(j) represents the time domain and N is the length of the sequence.

And its inverse, IDFT (inverse discrete Fourier transform) is given by:

$$x\,[\,j\,] \;=\; \sum_{n=0}^{N-1} f(\,i\,)\, e^{-k2\pi ij/N}$$

One way to look at the DFT formula is to set it as a matrix operation. The matrix formulation of DFT is much easier to work with, and has the form , $F = D_N\, x$, here F is the set of resulting frequencies, x is a Nx1 vertical vector representing the time domain.

And the transformation matrix D_N is given by:

$$\mathbf{D}_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \quad , \; where \quad w_N = e^{\frac{-2\pi k}{N}}$$

The process of this matrix multiplication requires n^2 multiplication. For a large set of data, the computation of this matrix become inordinate even for performing the matrix operation using a computer. For such reasons, FFT (fast Fourier transformation) is used for computation of Fourier transformation.

FFT:

An FFT algorithm is used to compute DFT/ IDFT of a sequence. FFT algorithm reduces the computing complexity of DFT from n^2, to nlog(n). For larger number of n, the difference in speed is colossal. FFT algorithm are much more accurate than DFT in the presence of round-off error. In an FFT algorithm the transformations are computed by factorizing the DFT matrix, given above, into a product sparse matrix, i.e. a matrix with mostly '0' elements, doing this reduces the computing complexity. There are many different FFT algorithm based on the factorization on 'n', generally as a power of 2.

James Cooley and John Tukey invented the modern generic FFT algorithm in 1965, which is applicable when n is a composite and not necessary a power of 2.

Any FFT method computes the same results in nlog(n) operations, where log is of base 2. FFT algorithms generally fall into two categories: 1) decimated in time and 2) decimated in frequency.

1) Decimated in time:

It re-arranges the DFT equation, $f\,[\,i\,] \;=\; \sum_{n=0}^{N-1} x(\,j\,)\,e^{-k2\pi ij/N}$, into two parts: a sum over the even-numbered time indices j = [0, 2, 4, ..., N – 2] and sum over odd-numbered indices j = [1, 3, 5, ..., N - 1], such as:

$$f[\,i\,] \quad = \sum_{n=0}^{N-1} x(j)e^{-k2\pi ij/N}$$

$$= \sum_{n=0}^{N/2-1} x(2\,j\,)\,e^{-k2\pi i(2j)/N} \;+\; f\,[\,i\,] \;=\; \sum_{n=0}^{N/2-1} x(2\,j\;+1\,)\,e^{-k2\pi i(2j+1)/N}$$

$$= DFT_{N/2}[x(0), x(2), \ldots\ldots, x(N-2)] + W_N^i DFT_{N/2}[x(1), x(3), \ldots\ldots, x(N-1)] \quad \ldots\ldots\text{(A)}$$

, where $W_N^i \;=\; e^{-k2\pi i/N}$ called the twiddling factor.

Thus, DFT of output, $f[\,i\,]$, can be computed as sum of output of two DFTs of length N/2.

2) Decimated in frequency:

It rearranges the DFT equation into two parts: computation of the even-numbered discrete indices $f[\,i\,]$, for

i = [0, 2, 4, ..., N – 2] or $f(2r)$, given as:

$$f(2r) \;=\; \sum_{n=0}^{N-1} x(j)\,W_N^{2rj} \;=\; DFT_{N/2}\,[\,x(j) \;+\; x(j+N/2)]$$

And computation of odd-numbered indices i = [1, 3, 5, ..., N-1] or $f(2r + 1)$, given as:

$$f(2r + 1) = \sum_{n=0}^{N-1} x(j)\, W_N^{(2r+1)j} = DFT_{N/2}\,[\,(x(j) - x(j + N/2))W_N^j]$$

Thus, both the even-numbered and odd-numbered indices of frequency output $f[\,i\,]$ can be each computed by DFTs of length N/2.

Statement of problem

Analyzing a given audio signal by taking its Fourier transform using FFT and plotting a Spectrogram using Gabor transform.

Details of the methods of study

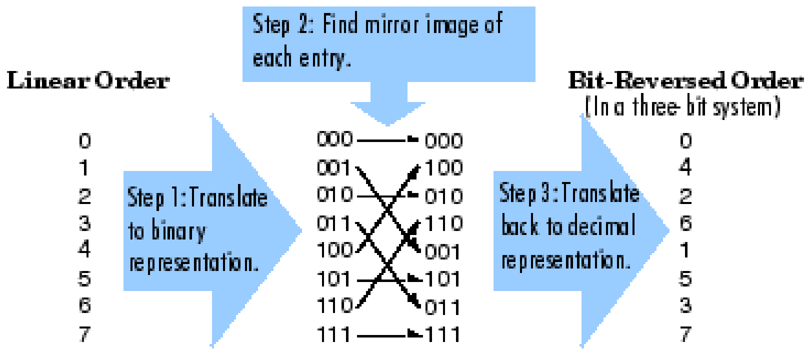
FFT ALGORITHM:

In Cooley-Tukey FFT algorithm, the input elements are first rearranged in a bit-reversed order.

Two number are said to be bit-reversed order of each other, if the binary representation of one is the mirror image of the other. E.g. 1 (001) is bit-reversed order of 4(100).

To arrange the input in bit-revered order:

i) The indices are translated into their binary value.
ii) Find the mirror-image of each binary input and write it beside their original binary value.
iii) Translate it back to decimal value.



Once the input elements are arranged, output transformation is performed, using decimation in time (equation A), called as the Danielson-Lanczos lemma, the basic idea being to break-up a transform of length N into a transform of order N/2.

This procedure can be applied recursively to break up the N/2 length transform equation into N/4 length. If N is a power of 2, the process can be recursively applied till we get a transform equation of length 1. There are $\log_2 N$ stages required for this decomposition. i.e. if the input sequence has 16-point data (N = 16), 4 recursions of lemma is required. It is much easier to work with data set where the original value of N is a power of 2, if the length of the data set is not a power of two, we can pad the data with 0's until the next power of 2 is reached. Once the transform of length 1 is reached, the Fourier transform of length 1, is just the identity operation, that copies its one input value into its one output slot (frequency spectra). The last step is to rearrange the N frequency spectra obtained, in the exact reverse order that the time domain decomposition took place in.

Power spectral density:

It gives the distribution of Power over the entire frequency range.

It is the normalized squared frequency of the DFT of individual elements in audio time series

Working of functions from different Modules used in the project

- **Librosa Module:**

1)librosa.load : it loads the audio signal as audio time series.

2)librosa.stft: Gabor transform is applied on the given data set. The output is a complex matrix.

3)librosa.griffinlim: does magnitude spectrogram inversion. The (absolute value of)output of the stft is given as input here

- **Numpy Module:**

1)numpy.fft.fft : it calculates the one dimensional DFT takes array as input(audio time series in our case)

2) numpy.fft.ifft: calulates inverse DFT of the one dimensional DFT(which we computed from fft)

3)numpy.random.normal : Gives random values from normal gaussian distribution

- **Algorithm to add noise to the audio signal**:

1. Load the audio file in its audio time series using librosa.load
2. Add white noise to the audio time series generated from the random numbers using numpy.random.normal
3. Write a new audio file using wavfile.write

- **Algorithm for Denoising:**

Step 1) First we load the noisy audio signal as audio time series(type:float)

Step 2) Take fast fourier transform of that audio time series using fft command- $\hat{f}$

Step 3) We take the power spectral density of $\hat{f}$. In the plot we can clearly see three large peaks. We take a threshold value and zero out components below it to remove noise.

Step4) After filtering the noise we take the the inverse fft (using numpy.fft.fft) to get back a clean and denoised audio time series.

Step 5) Take the gabor transform(using librosa.stft) to plot spectrogram

Step 6) Use griffinlim(using librosa.griffinlim) to take inverse of the magnitude of spectrogram

Step 7) create a audio file using the data points from the output of griffinlim.

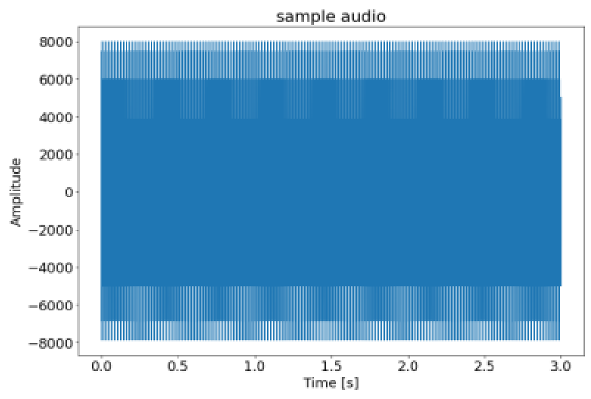
Result

Output of final_submission_audio_generation:

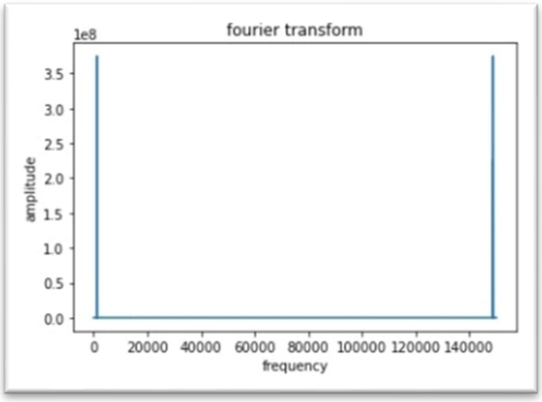
The output is an audio file which we have named as 'submission_audio'. It is of 3 seconds. The file has been added to the drive folder.

Output final_basic_audio_signal:

1) The graph is an Amplitude vs Time graph. It shows the amplitude of the sound signal at an instant of time.



2) The graph is an Amplitude vs Frequency graph. This is the Fourier transform of the sound file ('submission_audio')

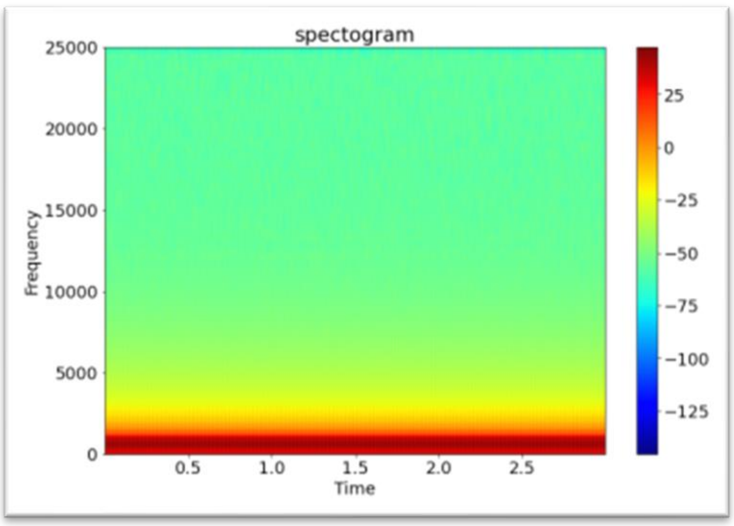


3) We plot a spectrogram for the audio signal. Spectrogram is a way of representing the signal strength, over time at various frequencies that are present in the waveform of the signal. From the graph we can see which frequencies have more or less energy as well as how those energy levels vary with time.

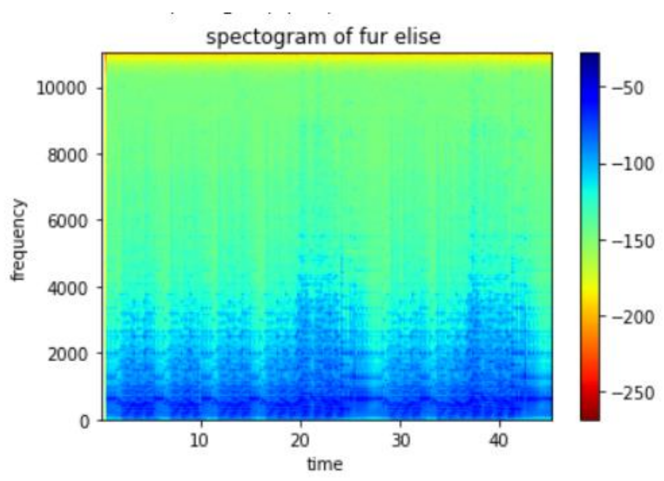
Spectrograms are 2-D plots, with time in the x-axis and frequency in the y-axis, with a 3rd dimension of amplitude (or energy or loudness) being represented by colour.

The spectrogram of the submission audio is shown below. The red colour represents the frequency with the highest strength.

Spectrograms are commonly used to display frequencies of sound waves produced by humans, machinery, animals, jets, whales. They are also widely used in the study of seismology.



Output for the code 'spectogram of fur elise':



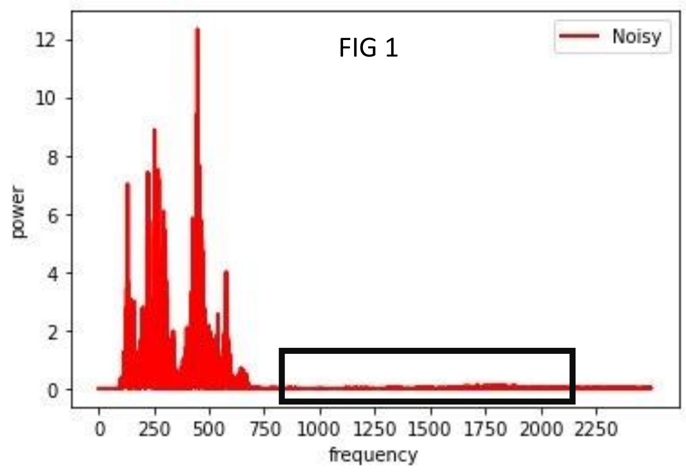
Output of adding white noise to audio:

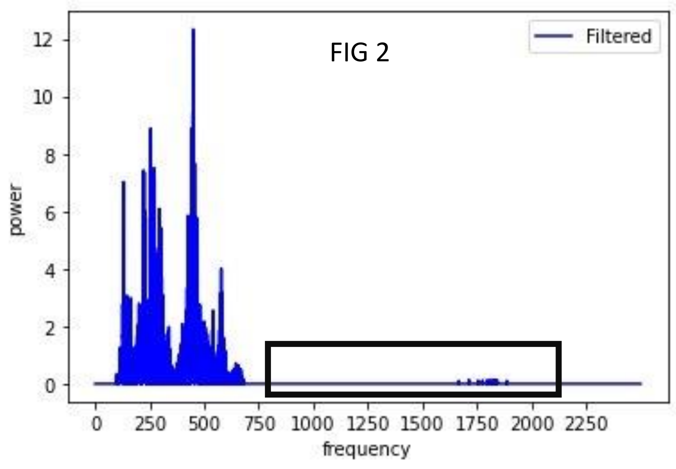
Here we recorded an audio file name 'original audio' and added white noise to its background, 'noiseaudio'. The later is used to demonstrate denoising.

Output of denoising:

As mentioned above, we take the noiseaudio and denoise it with the help of power spectrum density (PSD) plot.

We know that a Gaussian white Noise has a flatter PSD curve. In the PSD vs frequency graph with noise (figure 1) we can see some large peaks and some values with lower PSD that we recognize as our noise and after applying the filter we can see in the filtered signal (figure 2) that those lower PSDs are removed.





Limitations

- There is always uncertainty between the time and the frequency resolution of the window function used in this analysis since it is well known that when the time duration gets larger, the bandwidth becomes smaller. Several ways have been proposed to find the uncertainty bound, and the most common one is the multiple of the standard deviations on time and frequency domain:

$$\sigma_t{}^2 = \frac{\int t^2|x(t)|^2 dt}{\int |x(t)|^2 dt}, \sigma_f{}^2 = \frac{\int f^2|X(f)|^2 df}{\int |X(f)|^2 df}$$

$$\sigma_t \times \sigma_f \geq \frac{1}{4\pi}$$

- The denoised audio is not completely devoid of noise, as many a times the frequency of the noise and the important data (human speech or alert alarm) match and hence the cancellation is not 100%
- Spectrogram to audio conversion also leads to loss of quality as there are truncations of smaller values.

Relevance

- It is related to computational physics component of unit one of mathematical physics
- It is also related to the integral transform topic of unit 2 of mathematical physics

Bibliography

i. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control - J. Nathan Kutz and Steven L. Brunton

ii. https://www.youtube.com/playlist?list=PLMrJAkhIeNNT_Xh3Oy0Y4LTj0Oxo8GqsC

iii. A student's guide to Fourier transforms – J.F. James

iv. Mathematical Methods for Physicists - Arfken Weber and Harris

v. Wavelet transforms and their applications - Lokenath Debnath

Link: https://drive.google.com/drive/folders/1TBevNC3MkTzIWTgrY1SkbI5rQ5VaJHS4?usp=sharing

Credits:

Name	Theory	Coding	Internet survey
Vidhi	DFT, FFT, and FFT algorithm	33%	33%
Kunal	Denoising algorithm, modules, results, limitations	33%	33%
Nilay	Fourier series, Fourier transform, Gabor transform	33%	33%