Name: Xinhe Shan
Class: Math 320
Date: 2016/12/15
Instructor: Zvi Rosen

Fourier Transform and Voice Recognition

1. Introduction

Voice recognition has long been an active field of research due to its great potential in human-machine interactions. Since the invention of computers, we have constantly aimed for better and more direct way of sending computers instructions. From originally having no interface, we have come to a long way and developed graphical user interface and more recently, touch user interface. Looking forward, voice user interface is the next major user interface to come, as it frees up people's hands and eyes. This paper is going to look into one of the signal processing ways—Fourier Transform, examine its specific implementation in MATLAB and demonstrate its application in voice recognition.

2. Fourier Series

Definition 1.1 Fourier Series is a way to represent any periodic function using sine and cosine function [1].

The expression of Fourier Series is the following[2]:

(2.1) $f(t) = a_0 + a_1\cos(\omega_0 t) + b_1\sin(\omega_0 t) + a_2\cos(2\omega_0 t) + b_2\sin(2\omega_0 t) + \cdots + a_m\cos(m\omega_0 t) + b_m\sin(m\omega_0 t)$

where

(2.2) $$a_k = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos(k\omega_0 t)\,dt$$

and

(2.3) $$b_k = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin(k\omega_0 t)\,dt$$

For k = 1, 2, …

and

(2.4) $$A_0 = \frac{1}{T}\int_0^T f(t)dt$$

Example 1.1 Periodic Square Wave

Consider a periodic square wave: $f(x) = \begin{cases} 0, & (\frac{1}{2} + 2n < x < \frac{3}{2} + 2n, n \in Z) \\ 1, & (-\frac{1}{2} + 2n < x < \frac{1}{2} + 2n, n \in Z) \end{cases}$

Because this is an even function, the coefficient of in front of each sine term is 0. Sine function is an odd function and the sum of odd functions is still an odd function. As the function is fitted by more terms of the Fourier series, the deviation between the fit curve and the actual function becomes smaller (Figure.1)[3].

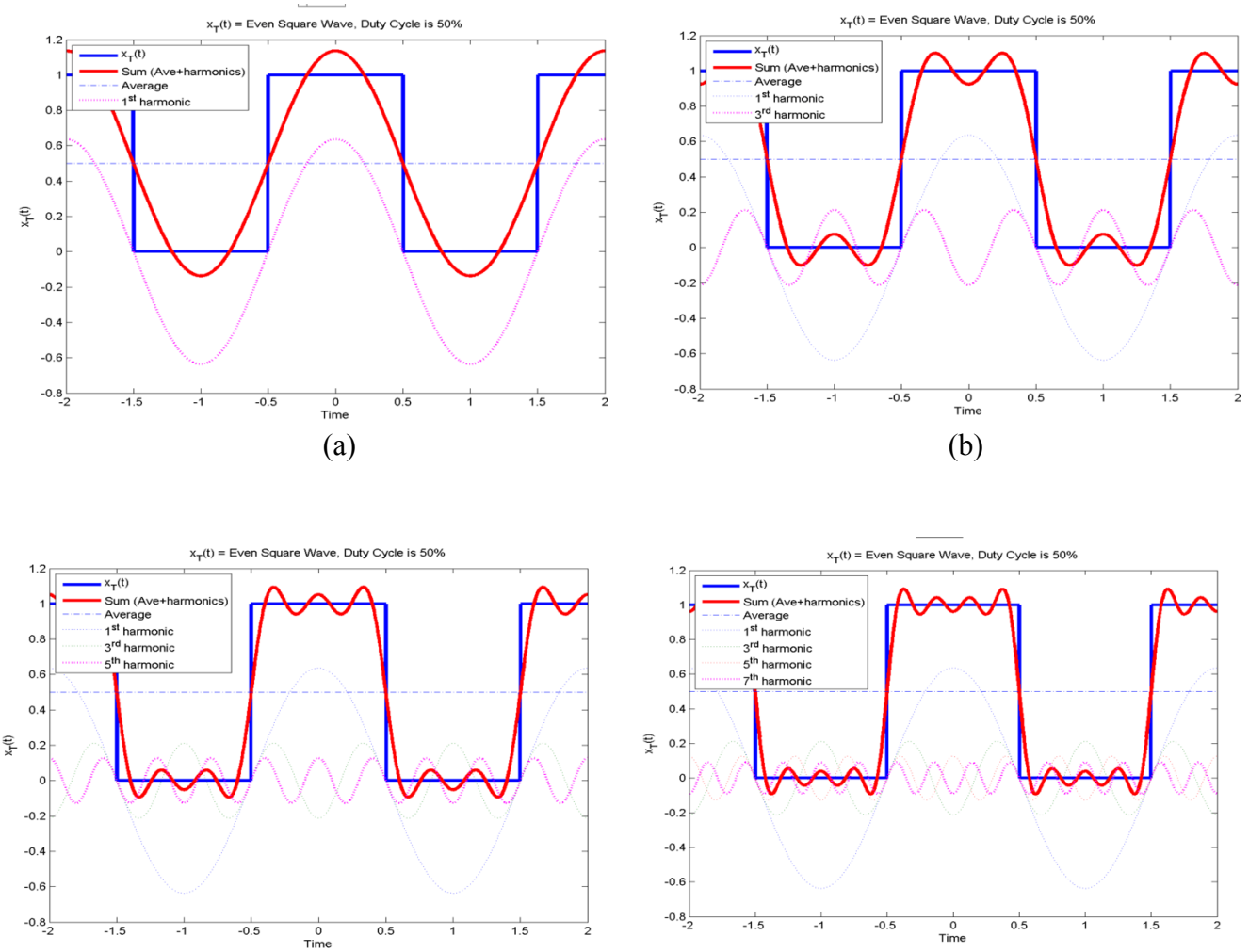


Figure 1. (a) square wave fitted by $a_1\cos(\omega_0 t)$, (b) square wave fitted by $a_1\cos(\omega_0 t) + a_3\cos(3\omega_0 t)$, (c) square wave fitted by $a_1\cos(\omega_0 t) + a_3\cos(3\omega_0 t) + a_5\cos(5\omega_0 t)$, (d) square wave fitted by $a_1\cos(\omega_0 t) + a_3\cos(3\omega_0 t) + a_5\cos(5\omega_0 t) + a_7\cos(7\omega_0 t)$

Proposition 2.1 The Fourier series can be expressed in a complex form[2]:

(2.5) $$f(t) = \sum_{k=-\infty}^{\infty} \tilde{c}_k e^{ik\omega_0 t}$$

where

(2.6) $$\tilde{C}_k = \frac{1}{T}\int_{-T/2}^{T/2} f(t) e^{-ik\omega_0 t} dt$$

Proof:

The Fourier series, equation (1), can be written in the summation form as:

(2.7) $$f(t) = a_0 + \sum_{k=1}^{\infty}[a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t)]$$

Using Euler's equation

(2.8) $$e^{\mathrm{i}\theta} = \cos\theta + \mathrm{i}\sin\theta$$

we get

(2.9) $$\cos\theta = \frac{1}{2}\left(e^{\mathrm{i}\theta} + e^{-\mathrm{i}\theta}\right) \qquad \sin\theta = \frac{1}{2\mathrm{i}}\left(e^{\mathrm{i}\theta} - e^{-\mathrm{i}\theta}\right)$$

Equation (7) becomes:

(2.10) $$f(t) = a_0 + \sum_{k=1}^{\infty}[\frac{1}{2}a_k(e^{ik\omega_0 t} + e^{-ik\omega_0 t}) + \frac{1}{2i}b_k(e^{ik\omega_0 t} - e^{-ik\omega_0 t})]$$

Combine the like terms:

(2.11) $$f(t) = a_0 + \sum_{k=1}^{\infty}[c_k e^{ik\omega_0 t} + c_k^* e^{-ik\omega_0 t}]$$

where

(2.12) $$c_k = \frac{1}{2}(a_k - ib_k)$$

and

(2.13) $$c_k^* = \frac{1}{2}(a_k + ib_k)$$

If we define

(2.14) $$c_k^* = c_{-k} = \frac{1}{2}(a_k + ib_k)$$

and

(2.15) $$a_0 = c_0$$

Equation (11) becomes:

(2.16) $$f(t) = c_0 + \sum_{k=1}^{\infty}[c_k e^{ik\omega_0 t} + c_{-k} e^{-ik\omega_0 t}]$$

and can be further written as:

(2.17) $$f(t) = c_0 + \sum_{k=1}^{\infty}[c_k e^{ik\omega_0 t}] + \sum_{k=-1}^{-\infty}[c_k e^{ik\omega_0 t}]$$

Combining the two sum, equation (17) becomes equation (5). When k = 0, the exponential term $e^{-ik\omega_0 t}$ becomes 1, which satisfies the term C_0.

For $k = 1, 2, 3 \ldots$ we have

(2.18) $$C_k = \frac{1}{2}(a_k - ib_k)$$

Plug in equation (2) and (3)

(2.19) $$C_k = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\,(\cos(k\omega_0 t) - i\sin(k\omega_0 t))\mathrm{dt}$$

which equals to

(2.20) $$C_k = \frac{1}{T}\int_{-T/2}^{T/2} f(t) e^{-ik\omega_0 t}\mathrm{dt}$$

Similarly, for $k = 1, 2, 3 \ldots$ we have

(2.21) $$C_{-k} = \frac{1}{T}\int_{-T/2}^{T/2} f(t) e^{ik\omega_0 t}\mathrm{dt}$$

Therefore, equation (6) holds

$$\tilde{C_k} = \frac{1}{T}\int_{-T/2}^{T/2} f(t) e^{-ik\omega_0 t} dt$$

3. Fourier Transform

Definition 3.1 Fourier Transform is a way to transform a function from its time domain to its frequency domain[4].

The equation for Fourier Transform is the following:

(3.1) $$F(\omega) = \int_{-\infty}^{\infty} f\,(t) e^{i\omega t}\, dt$$

And the equation for the inverse Fourier Transform is:

(3.2) $$f(t)= \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}\,d\omega$$

The definition of Fourier Transform is similar to Fourier series. The only difference is that Fourier Transform does not require the function in the time domain to be periodic.

Proposition 3.1 Fourier Transform can be deduced from Fourier series by simply taking the period T to infinity.

Proof:

Plug (6) into (5) and write $\frac{1}{T}$ as $\frac{\omega_0}{2\pi}$, we get:

(3.3) $$f(t) = \sum_{-\infty}^{\infty}\left\{\frac{\omega_0}{2\pi}\int_{-T/2}^{T/2} f(t)e^{-ik\omega_0 t}dt\right\}e^{ik\omega_0 t}$$

As T $\longrightarrow \infty$, ω_0 becomes dω. The $k\omega_0$ on the exponential term becomes ω, as the sum becomes integral

(3.4) $$f(t) = \int_{-\infty}^{\infty}\left\{\frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt\right\}e^{i\omega t}\mathrm{d}\omega$$

The equation in the "{}" become a function of ω after the integral, so (25) becomes:

(3.2) $$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\{f(\omega)\}\,e^{i\omega t}\mathrm{d}\omega$$

and

(3.1) $$f(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

4. Fourier Transform in MATLAB

Definition 4.1 Discrete Time Fourier Transform (DTFT) is Fourier Transform on a sequence in the time domain, instead of on a continuous function[5].

Its expression looks like:

(4.1) $$f(\omega) = \sum_{t=-\infty}^{\infty} x[t]e^{-i\omega t}$$

Because an audio file in essence is a long vector with a fix number of samples in one second, we cannot apply the analytical Fourier Transform on it. DTFT, instead of taking the integral of a

continuous function $f(t)$, it takes the sum of a sequence $x[t]$. This sequence can be viewed as the audio file in the context of digital signal processing.

However, there is one more problem. (26) maps the discrete time series to a continuous function in the frequency domain, which requires an evaluation on every possible ω. This requires infinite amount of calculation and computers cannot do it.

Definition 4.2 Discrete Fourier Transform (DFT) is similar to DTFT, expect that it evaluates Fourier Transform on a finite time domain with finite many frequency point.

The expression of DFT is the following:

(4.2) $$f(\omega) = \sum_{t=0}^{N-1} x[t] e^{-\frac{it2\pi k}{N}}, \quad k = 0,1,2, \dots, N-1$$

Even though DFT only picks finite many frequency point to evaluate Fourier Transform, the result of DFT is quite accurate as shown in Figure 2.

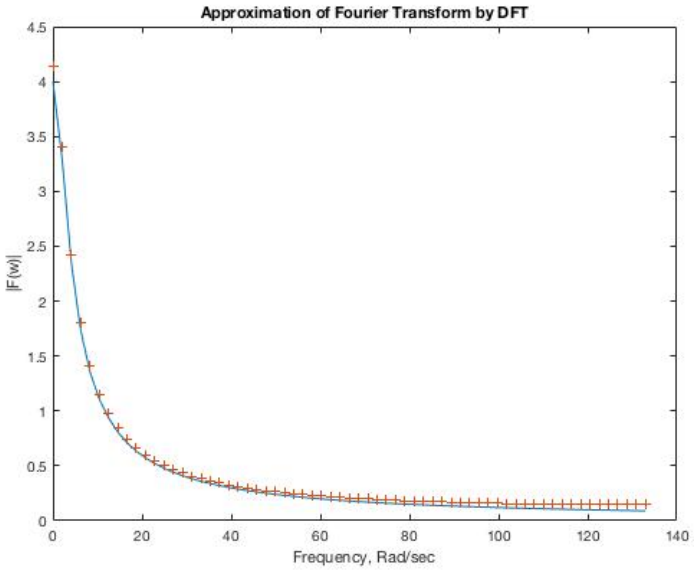


Figure 2. Approximating Fourier Transform of $f(t) = 12e^{-3t}$ by DFT. The blue line is the function in the frequency domain yielded by Fourier Transform and the yellow dots represent the output of DFT.

Proposition 4.1 The complexity of naïve DFT is $O(N^2)$

Proof:

For each t, there is N calculations because k picks value from 0 to N-1. There are N different t. Therefore, a DFT needs N^2 calculations and is of $O(N^2)$.

This extremely big complexity makes naïve DFT unpractical in real life applications, raising the need to have another way of computing DFT.

5. Fast Fourier Transform

Definition 5.1 Fast Fourier Transform is an algorithm that computes DFT with complexity $O(N\log_2 N)$[6].

Proof:

Starting from the same equation for DFT:

$$f(\omega) = \sum_{t=0}^{N-1} x[t] e^{-\frac{it2\pi k}{N}}, \quad k = 0,1,2,\dots,N-1$$

We can divide it into two terms, one for odd index and one for even index:

$$f(\omega) = \sum_{t=0}^{\frac{N}{2}-1} x[2t] \exp\left(-\frac{i2\pi k(2t)}{N}\right) + \sum_{t=0}^{\frac{N}{2}-1} x[2t+1] \exp\left(-\frac{i2\pi k(2t+1)}{N}\right),$$
$$k = 0,1,2,\dots,N-1$$

Move the 2 from the nominator of the exponential to the denominator we have:

$$f(\omega) = \sum_{t=0}^{\frac{N}{2}-1} x[2t] \exp\left(-\frac{i2\pi kt}{N/2}\right) + \sum_{t=0}^{\frac{N}{2}-1} x[2t+1] \exp\left(-\frac{i2\pi k(t+1/2)}{N/2}\right),$$
$$k = 0,1,2,\dots,N-1$$

To make the exponential terms of each part the same, we can factor out a constant C_k from the second term:

$$f(\omega) = \sum_{t=0}^{\frac{N}{2}-1} x[2t] \exp\left(-\frac{i2\pi kt}{N/2}\right) + C_k \sum_{t=0}^{\frac{N}{2}-1} x[2t+1] \exp\left(-\frac{i2\pi kt}{N/2}\right),$$
$$k = 0,1,2,\dots,N-1, \qquad where\ C_k = e^{-\frac{i2\pi k}{N}}$$

The exponential term can be expanded using Euler's formula:

$$\exp\left(-\frac{i2\pi kt}{N/2}\right) = \cos\left(-\frac{2\pi kt}{N/2}\right) + i\sin\left(-\frac{2\pi kt}{N/2}\right)$$

Because k goes from 0 to N-1, the cosine and sine terms repeat themselves after k passing N/2 due to the symmetry of cosine and sine function.

$$\cos\left(-\frac{2\pi kt}{N/2}\right) = \cos\left(-\frac{2\pi(N/2+k)t}{N/2}\right)$$

$$\sin\left(-\frac{2\pi kt}{N/2}\right) = \sin\left(-\frac{2\pi(N/2+k)t}{N/2}\right)$$

This property allows the computer to computer k only from 0 to $\frac{N-1}{2}$, effectively reducing the operation by a half. N terms allow such division happen $\log_2 N$ times and this gives us $O(N\log_2 N)$ complexity, making it a much faster algorithm than $O(N^2)$ naïve DFT.

Example 5.1 Fast Fourier Transform of periodic Signals

Fast Fourier Transform decomposes seemingly complicated wave function in its time domain into frequency domain.

For example, the wave function $f(t) = \sin(2\pi 15t) + \sin(2\pi 40t) + \sin(2\pi 10t)$ has the following results:

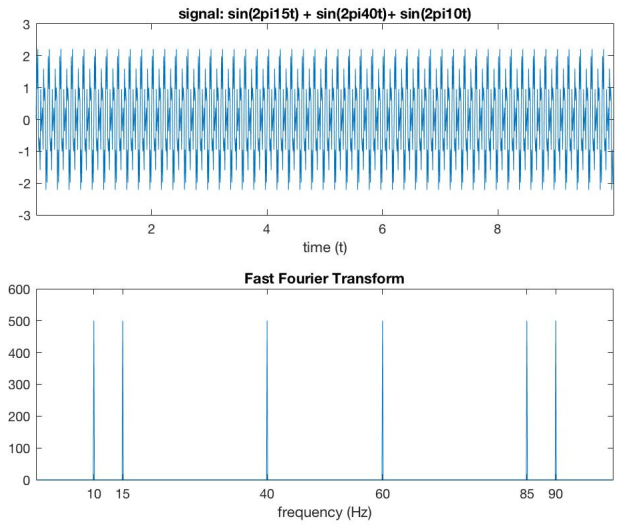


Figure 3. The upper half is the graph of the signal and the lower half is the graph of Fourier Transform

The Fourier Transform accurately extracts the three distinct frequencies: 10 Hz, 15 Hz and 40 Hz. And all three peaks are of the same height because the three sine waves have the same magnitude.

If we add some coefficient before the sine wave, the result of FFT will give peaks of different height

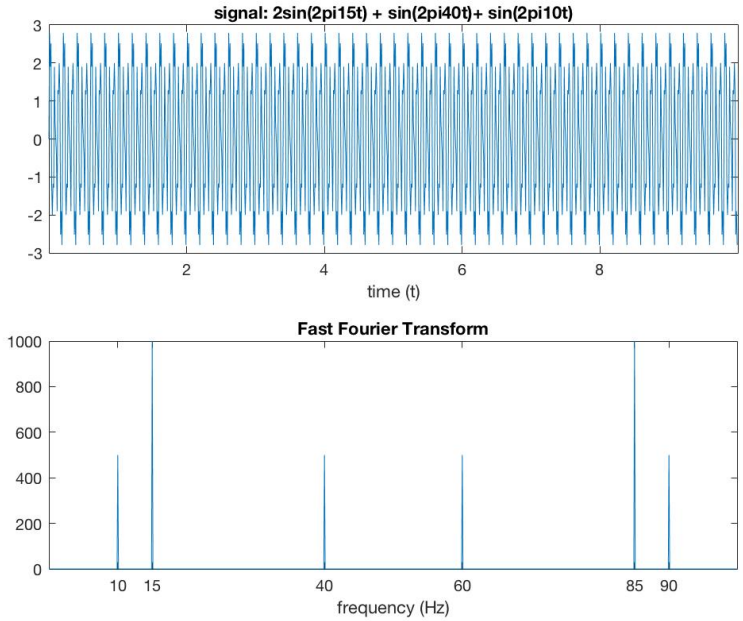


Figure 4. The signal wave is $f(t) = 2\sin(2\pi 15t) + \sin(2\pi 40t) + \sin(2\pi 10t)$ and the peak on 15 Hz and 85 Hz are exactly twice as much as the other peaks.

6. Nyquist Frequency and Aliasing

Theorem 6.1 (Nyquist-Shannon sampling theorem)
The sampling frequency has at least to be twice the frequency of the continuous wave to capture all the information of that wave[7].

Proof [8]

Assume there is a function x(t) in the time domain, and P(t) is an impulse function that samples x(t) at a fixed interval. P(t) by definition is the following:

$$\mathrm{P(t)} = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

And the function after sampling becomes:

$$\mathrm{X_s(t)} = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

Because $\delta(t - nT)$ is 0 everywhere except at nT, we can move x(t) into the sum:

$$\mathrm{X_s(t)} = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT)$$

Next we want to use Fourier Transform to convert $\mathrm{X_s(t)}$ into its frequency domain

$$\mathrm{X_s}(\varpi) = \int_{-\infty}^{\infty} X_s(t) e^{-i2\pi\omega t} dt$$

But this integral is too complicated to compute and we instead evaluate the convolution of $x(nT)\delta(t - nT)$ in the frequency domain. The equivalence of these two computations will be proved later. And the result is:

$$\mathrm{X_s}(\varpi) = \frac{1}{T}\sum_{-\infty}^{\infty} X(\varpi - \mathrm{n}\varpi_\mathrm{s}), where\ \varpi_\mathrm{s} = \frac{2\pi}{T}, the\ sampling\ frequency$$

When ϖ_s greater than 2ϖ we can see that each frequency signal is separate with each other (Figure 5). And one can recover the original signal by taking the inverse Fourier Transform because no information is lost.

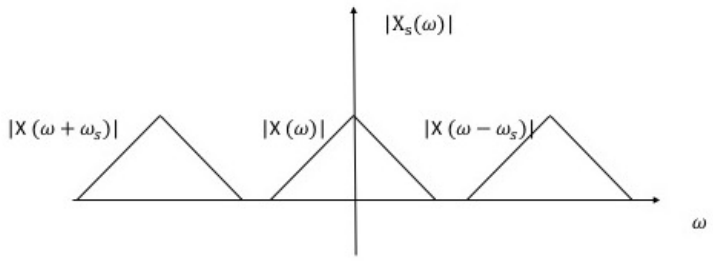


Figure 5. The result of Fourier Transform on the wave after sampling; the sampling frequency ϖ_s is greater than 2ϖ.

When When ϖ_s less than 2ϖ we can see that each frequency signal is overlapping with each other (Figure 6). And the signal in the overlap region is lost.

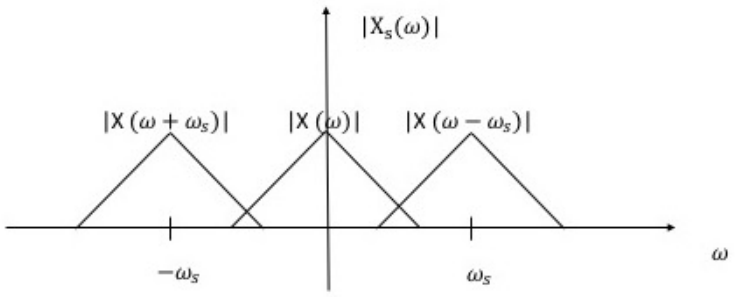


Figure 6. The result of Fourier Transform on the wave after sampling; the sampling frequency ϖ_s is less than 2ϖ.

Example 6.1 Nyquist frequency

The Fourier Transform of sin(2πt) and sin(2π9t) give the same result when the sampling frequency is 10 Hz (Figure 7).

The maximum frequency a 10 Hz sampling could effectively capture is 5 Hz and that's called Nyquist Frequency. Any frequency x Hz above 5 Hz will look the same as the (10 - x) Hz. Therefore, when calculating the magnitude of frequencies yielded by the FFT, one needs to count the magnitude of the aliasing frequency, which is the same as the real frequency. Thus, the real magnitude is always twice what's yielded by the Fast Fourier Transform.

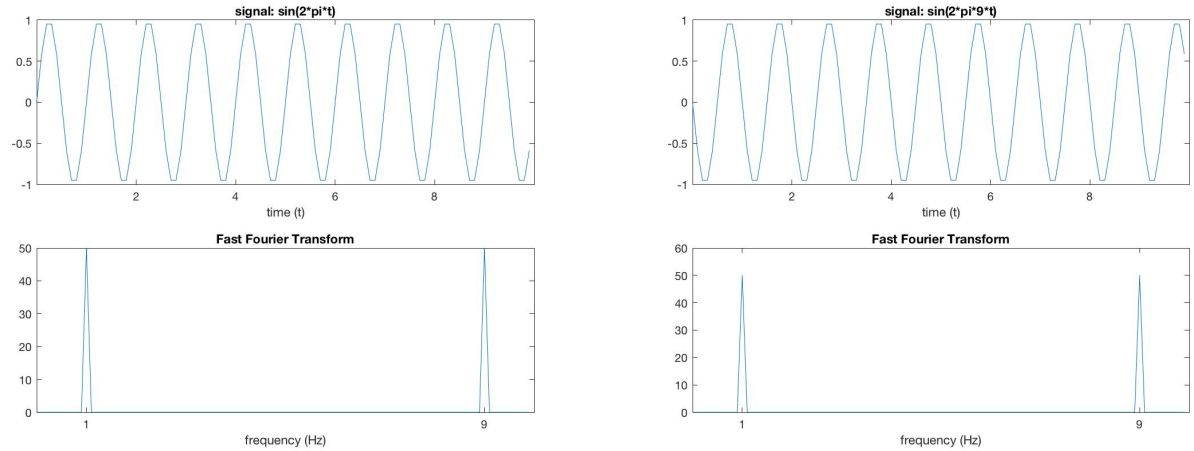


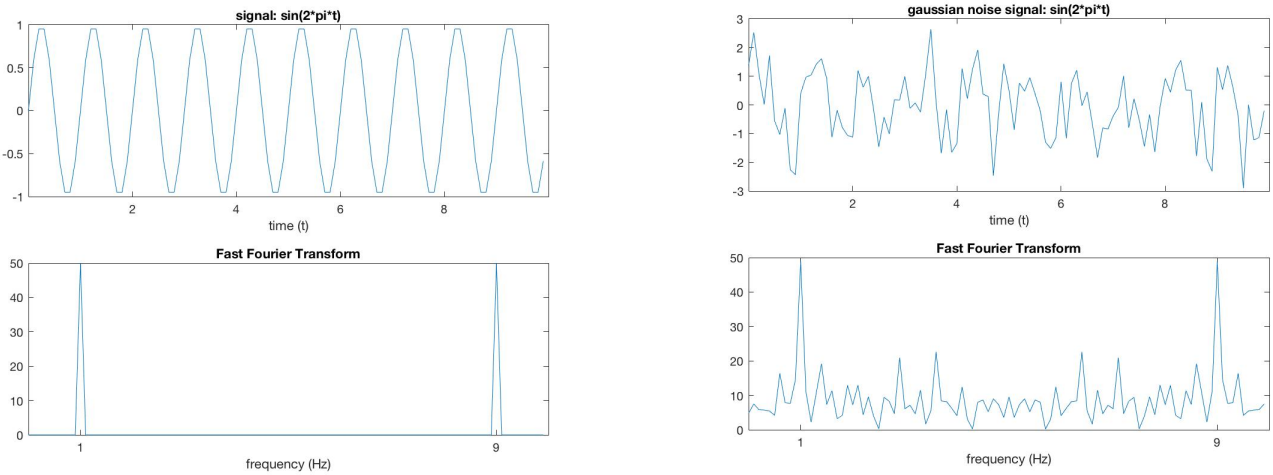
Figure 7. Example of aliasing. The signal wave and FFT result of sin(2πt) and sin(2π9t) appear the same on the figure but are different in reality.

7. Voice Recognition

The Fourier Transform is useful for signal processing because it's robust to noise and good at extracting the feature of the sound.

Example 7.1 Fast Fourier Transform of periodic Signals with noise

For example, when adding Gaussian noise to sin(2πt), the original wave becomes unidentifiable (Figure 8). The periodicity and amplitude change drastically. However, looking at the Fourier Transform diagram, the two frequency peak at 1 Hz and 9 Hz are salient.



(a) (b)

Figure 8. (a) original sin(2πt) wave (b) sin(2πt) with Gaussian noise added.

To prove that Fourier Transform is resistant to noise in general, uniformly distributed noise is added. And the result appears to be more robust than when Gaussian noise is added, because the Fourier Transform peaks are still visible when the noise amplitude is twice the signal amplitude (Figure 9).

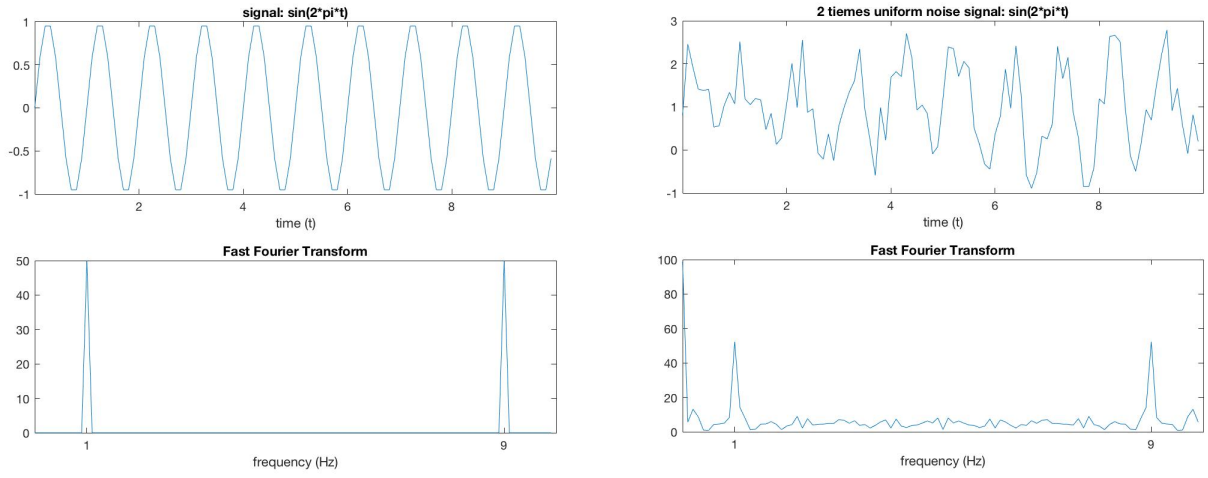
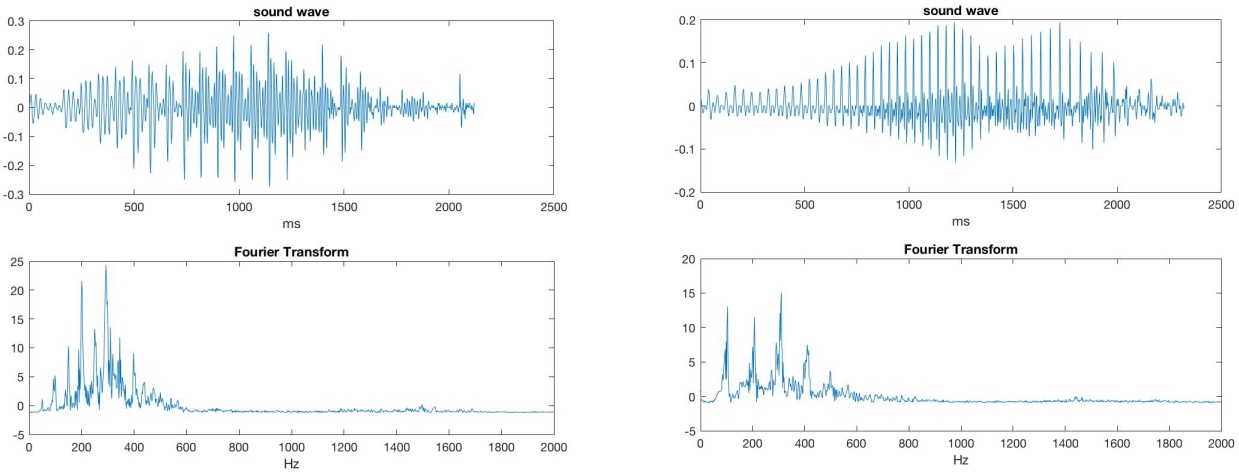


Figure 9. (a) original sin(2πt) wave (b) sin(2πt) with 2x uniform noise added.

Voice Recognition with FFT

We have so far demonstrated that Fourier Transform is capable of decomposing a wave from the time domain into the frequency domain, and is robust to background noise. Since all digital signals are discrete, we need to use DFT to conduct Fourier Transform on the signals. We proved that the DFT has a high fidelity to continuous Fourier Transform. However, the problem of normal DFT is that it has complexity of $O(N^2)$, which is unpractical to implement. FFT, as a fast algorithm of DFT, solves the problem by lowering the complexity to $O(N\log_2 N)$.

Because FFT has above characteristics, it can be used to differentiate different voices. Figure 10 shows the spectrum of two sound wave and FFT of pronunciation by a male and female. It can be seen in the graph that the male's Fourier Transform has two peaks, one around 200 Hz and one around 300 Hz. In contrast, the female's voice has 3 peaks, which are around 100 Hz, 200 Hz and 300 Hz. And the Fourier Transform figure looks much cleaner for the female than the male.



(a) (b)

Figure 10. (a) The sound wave and its Fourier Transform of a male's pronunciation of "one" (b) The sound wave and its Fourier Transform of a female's pronunciation of "one".

8. Comparing FFT spectra

It is shown above that Fourier Transform gives distinguishable results of different wave, and the differences are obvious just by eyes. However, the computer needs a more rigorous way of differentiating two sound waves.

Definition 8.1 Convolution [9]
Convolution is a mathematical calculation that gives the integral of the pointwise multiplication of two functions. Equation 8.1 is the mathematical definition of convolution of two funcitons.

(8.1) $$f(x) * g(x) = \int_{-\infty}^{\infty} f(x)g(y-x)dx$$

Theorem 8.1 Convolution theorem
Convolution theorem states that the Fourier Transform of a convolution is the pointwise dot product of Fourier Transforms.

Proof:

The Fourier Transform of equation 8.1 is:

(8.2) $$\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(x)g(y-x)\right]e^{-i2\pi\omega y}dy$$

and (8.2) can be further written as

(8.3) $$\int_{-\infty}^{\infty} f(x)\left[\int_{-\infty}^{\infty} g(y-x)e^{-i2\pi\omega y}dy\right]dx$$

The inner integral of (8.3) can be written as:

(8.4) $$\int_{-\infty}^{\infty} g(y-x)e^{-i2\pi\omega y}dy = F[g(y-x)] = e^{-i2\pi\omega x}G(\omega)$$

Plug 8.4 into 8.3

(8.5) $$\int_{-\infty}^{\infty} f(x)e^{-i2\pi\omega x}G(\omega)dx = \left[\int_{-\infty}^{\infty} f(x)e^{-i2\pi\omega x}dx\right]\ G(\omega) = \ F(\omega)G(\omega)$$

Thus, we proved

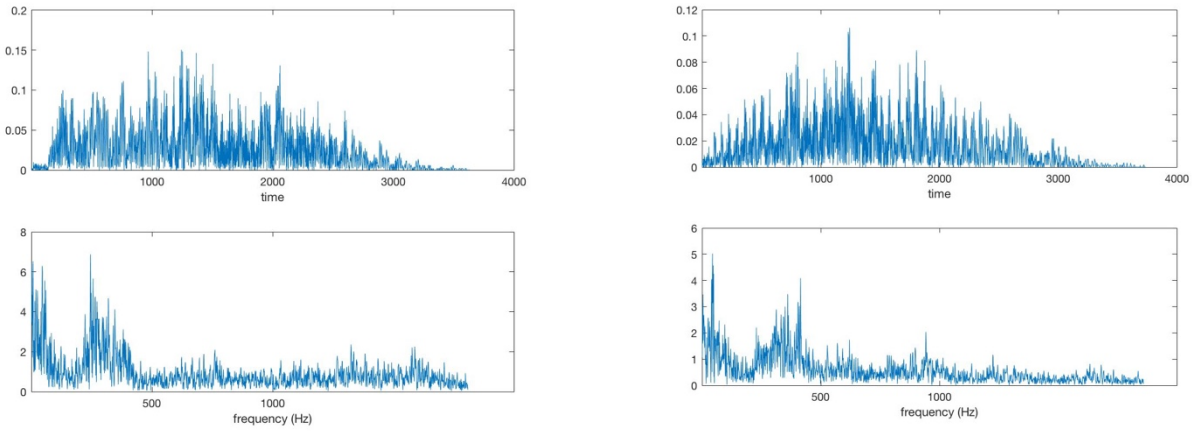
$$f(x) * g(x) = F(\omega)G(\omega)$$

In the MATLAB program, the input sound is compared with a male's and a female's pronunciation using the dot product between two Fourier Transform of the signals. In fact, this is how conv() function implemented in MATLAB and the reason comes down to the speed of computation. Convolution has $O(n^2)$ complexity while the dot product of two Fourier Transform has complexity O(n log n). The reduction of complexity is because Fourier Transform is computed using Fast Fourier Transform in MATLAB, so I will skip the proof.

9. Discussion

While Fourier Transform is useful in distinguishing the difference between two waves, it failed to reach a high accuracy in the experiment of differentiating difference pronunciations. This is one of the main reasons that this research project stops at differentiating male and female voices. One of the potential reasons for this low accuracy is that human voice has a narrow range, thereby has similar FFT spectra. For example, figure 11 shows the Fourier Transform spectra of the audio of "two" and "three". Even though these two words sound different, they appear quite similar on the Fourier Transform graph. In practice, the algorithm cannot differentiate between these two words.

The algorithm also doesn't work smoothly when the input is continuous speech, as it has an hard time separating each word in a sentence.



(a) (b)

Figure 11. (a) The audio of the word "two". The upper diagram shows the wave and the lower diagram shows the Fourier Transform. (b) The audio of the word "three". The upper diagram shows the wave and the lower diagram shows the Fourier Transform.

10. Conclusion

This research paper talks about Fourier Series, Fourier Transform, Discrete Fourier Transform and Fast Fourier Transform from a mathematical perspective. It also demonstrates the application of Fourier Transform in voice recognition.

Fourier Series is a nice way to decompose any periodic function, while Fourier Transform differs slightly and is able to be applied on aperiodic function. The implementation of Fourier Transform on discrete signals is through Fast Fourier Transform. This algorithm uses divide and conquer, effectively reducing the complexity of Discrete Fourier Transform from $O(n^2)$ to $O(n \log n)$.

FFT can be used to transform sound signals from its time domain to its frequency domain. However, one caveat is that the sampling frequency has to be twice bigger than the maximum frequency in the input signal. This threshold frequency is called Nyquist Frequency. Finally, using the convolution theorem, the convolution of two sound waves is replaced by the dot product of their Fourier Transform. While FFT still has a few drawbacks when coming to distinguishing different words' pronunciations, its low computation complexity and ability to shift function domains make it a strong candidate in voice recognition algorithms.

Reference

[1] "Fourier Series." Wikipedia. Wikimedia Foundation, n.d. Web. 15 Dec. 2016.
[2] Chapra, Steven C. Applied Numerical Methods with MATLAB for Engineers and Scientists. Boston: McGraw-Hill Higher Education, 2008. Print.
[3] College, Erik Cheever Swarthmore. "Fourier Series Examples." Fourier Series Examples. N.p., n.d. Web. 15 Dec. 2016.
[4] "Fourier Transform." Wikipedia. Wikimedia Foundation, n.d. Web. 15 Dec. 2016.
[5] " Discrete-time_Fourier_transform." Wikipedia. Wikimedia Foundation, n.d. Web. 15 Dec. 2016.
[6] "Discrete_Fourier_transform." Wikipedia. Wikimedia Foundation, n.d. Web. 15 Dec. 2016.
[7] " Nyquist–Shannon_sampling_theorem." Wikipedia. Wikimedia Foundation, n.d. Web. 15 Dec. 2016.
[8] Wang, Ruye. "Sampling Theorem." Sampling Theorem. N.p., 28 Jan. 2012. Web. 15 Dec. 2016.
[9] " Convolution." Wikipedia. Wikimedia Foundation, n.d. Web. 15 Dec. 2016.