@2013 Journal of PU, Part: B
Vol.2 No.2, July 2013, pp 41-45
PRESIDENCY UNIVERSITY
ISSN: 2224-7610

An Efficient Method for Frequency Calculation of an Audio Signal

Nazibur Rahman*

ABSTRACT

The audio signal generated by a tuning fork with a unique frequency has been acquired using a sound card. A Fast Fourier Transform (FFT) of the signal is performed with MATLAB and the frequency spectrum is plotted. The frequency spectrum revealed the unknown frequency of the tuning fork. The actual frequency of the tuning fork is compared with the experimental value. The result verifies that this method of calculating unknown frequency is 99.77% accurate.

Keywords: Fast Fourier Transform, sound card, A/D converter, MATLAB®, frequency spectrum, Nyquist frequency.

INTRODUCTION

The frequency of any unknown audio signal can be calculated using Fast Fourier Transform (FFT) [1] in MATLAB. In this paper, the fundamental (lowest) frequency of a tuning fork has been calculated and verified with the actual frequency. To perform this task, a microphone and a sound card is used to collect and convert sound level data to digital form. FFT is performed on the acquired digital data to find the frequency spectrum. The unknown frequency of the tuning fork has been determined from the peak frequency spectrum. FFT is an algorithm to compute the Discrete Fourier Transform (DFT) and its inverse [2]. To compute the DFT of N points in the naive way, using the definition, takes $O(N^2)$ arithmetical operations, while an FFT can compute the same DFT in only $O(N \log N)$ operations. The difference in speed can be enormous, especially for real time long data sets such as sound wave or speech signal where N may be in the thousands or millions [3]. For 1024 samples a straight DFT requires $1024^2 = 1048576$ arithmetic operations. However for the same number of samples an FFT requires $1024 * \log_2(1024) = 10240$ arithmetic operations.

Configuring the Data Acquisition Session

For this experiment, 1 second of sound level data is acquired on one sound card channel. Because the tuning fork vibrates at a nominal frequency between 200 and 3400 Hz , the sound card is configured to its lowest sampling rate of 8000 Hz according to Shannon's sampling theorem [4]. Even at this lowest rate, we did not experience any aliasing effects because the tuning fork will not have significant spectral content above 4000 Hz, which is the Nyquist frequency [5]. Nyquist rate $F_N = 2F_{max}$. After the tuning fork has been set to vibrate and placed it near the microphone, the data acquisition is triggered using a manual trigger. The whole experiment is completed in six steps.

a) Creating a device object for this experiment

* *Fellow IEB, Assistant Professor, Department of Electrical and Computer Engineering Presidency University, Dhaka-1212, Bangladesh.*
E-mail: nrahman@presidency.edu.bd

b) Adding a channel for data input to MATLAB
c) Configuring property values
d) Acquiring the signal data
e) Clearing the memory after the experiment is done
f) Analyzing of the acquired data.

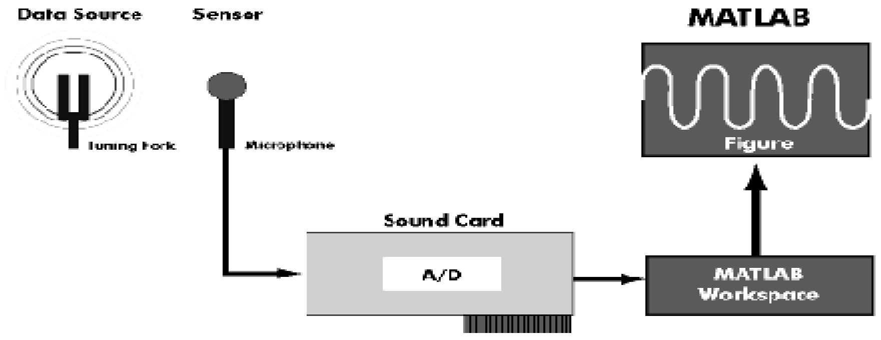

Fig.1: Tuning fork signal input to MATLAB using a sound

a) Creating a device object for this experiment

MATLAB manual states that first a device object has to be created, which is the analog input object Analog Input(AI) for a sound card [6]. The installed adaptors and hardware IDs are found with command ‘daqhwinfo’. The hardware ID for the sound card was ‘winsound’.

AI = analoginput('winsound');

b) Adding a channel for data input to MATLAB

One channel is added to AI object

chan = addchannel(AI,1);

c) Configuring property values

The property values is configured by assigning values to the basic setup properties, and created the variables blocksize and sampling frequency (Fs), which are used for subsequent analysis. The actual sampling rate is retrieved because it might be set by the software to a value that differs from the specified value. The MATLAB program follows.

```
%Pseudo Code for sample rate
duration = 1; %1 second acquisition
set(AI,'SampleRate',8000)
Actual Rate = get(AI,' Sample Rate');
set(AI,' Samples Per Trigger', duration*Actual Rate)
set(AI,' Trigger Type',' Manual')
blocksize = get(AI,' Samples Per Trigger');
Fs = Actual Rate;
```

d) Acquiring the signal data

Acquiring data has been started by the command start AI, which issued a manual trigger, and extracted all data from the sound card. Before trigger is issued, inputting data from the tuning fork into the sound card has begun.

```
start(AI)

trigger(AI)

data = getdata(AI);
```

e) Cleaning up the memory

When the data acquisition is completed, the AI object is removed from memory and from the MATLAB workspace to increase memory.

```
delete(AI)

clear AI
```

f) Analyzing the data

For this experiment, analysis consists of finding the frequency components of the tuning fork and plotting the results. To do so, the function daqdocfft is created. This function has been used to calculate the FFT of data, and requires the values of SampleRate and SamplesPerTrigger as well as data as inputs.

```
[f,mag] = daqdocfft(data,Fs,blocksize);
```

Daqdocfft outputs the frequency and magnitude of data, which is then ploted in MATLAB.

```
function [f,mag] = daqdocfft(data,Fs,blocksize);

%    [F,MAG]=DAQDOCFFT(X,FS,BLOCKSIZE) calculates the FFT of audio signal X

%    using sampling frequency FS and the SamplesPerTrigger

%    provided in BLOCKSIZE

%Pseudo Code for calculating the FFT

xfft = abs(fft(data));

% Avoid taking the log of 0.

index = find(xfft == 0);

xfft(index) = 1e-17;

% Finding the magnitude in dB.

mag = 20*log10(xfft); mag = mag(1:floor(blocksize/2));

f = (0:length(mag)-1)*Fs/blocksize;

f = f(:);
```

RESULTS AND DISCUSSION

The results are plotted using the following MATLAB program.

```
plot(f,mag)
grid on
ylabel('Magnitude (dB)')
xlabel('Frequency (Hz)')
title('Frequency Components of Tuning Fork')
```

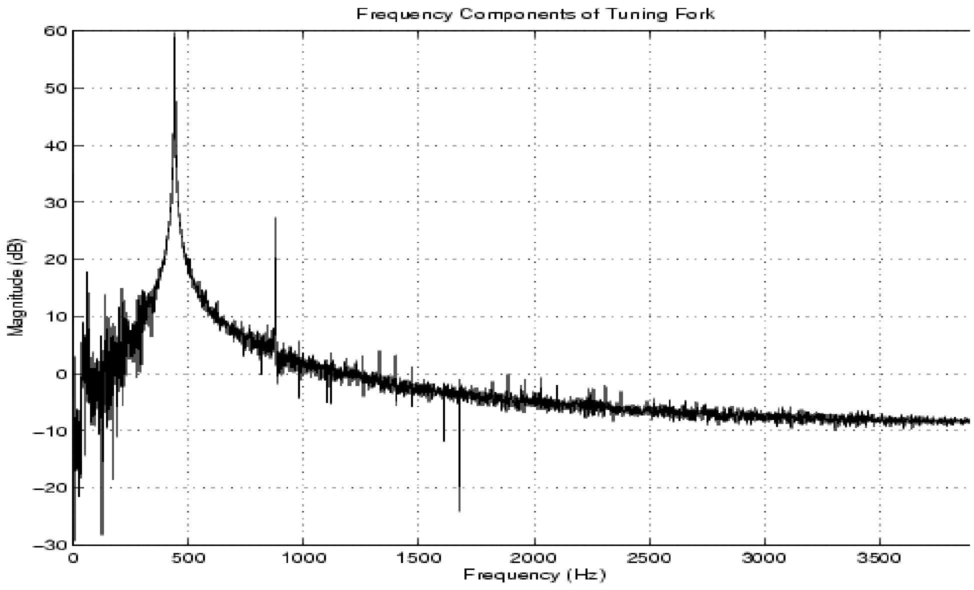

Fig.2: Frequency spectrum of the audio signal

The Figure 2 shows the fundamental frequency around 440 Hz and the first overtone around 880 Hz. A simple way to find actual fundamental frequency using MATLAB command is

```
[ymax,maxindex]= max(mag);

maxindex

maxindex =

  441
```

The answer is 441 Hz. The plot shows the tuning fork frequency is 440 Hz.

% error = $\frac{(441-440)x100}{440}$ = 0.23 %

The result confirms that any unknown audio frequency can be determined by MATLAB with a microphone and a sound card. The input audio signal can be further analyzed and processed using Digital filters for suitable application.

CONCLUSION

The analog audio signal of a tuning fork is acquired by a microphone and a sound card using MATLAB®. This signal is further analyzed in frequency domain by plotting the frequency spectrum of the time domain audio signal. The frequency spectrum shows the fundamental frequency of the audio signal as a peak in the plot. This method can be further extended to determine multiple frequencies in a signal. Fast Fourier Transform is a very efficient method of calculating the frequency spectrum of a time domain signal.

REFERENCES

[1] http://en.wikipedia.org/wiki/Fast_Fourier_transform accessed on 10 June, 2013.
[2] J. G. Proakis, and D. G. Manolakis, "Efficient Computation of the DFT: Fast Fourier Transform Algorithms," in *Digital Signal Processing: Principles, Algorithms and Applications*, 3rd ed., New Jersey: Prentice-Hall,1995, pp. 511-535.
[3] R. A. Roberts and C. T. Mullis, *Digital Signal Processing*, Boston: Addison-Wesley, 1987. pp. 212- 230.
[4] R. G. Lyons, *Understanding Digital Signal Processing*, 2nd ed., New Jersey: Prentice Hall, 2004, pp. 123-150.
[5] C. Schuler and M. Chugani, *Digital Signal Processing*: *A Hands-on Approach*, 1st ed., Noida, India, Tata McGraw-Hill, 2005, pp. 11-41.
[6] *MATLAB 7.5 user manual*, The MathWorks Inc., Natick, MA, 2007.