


Course Title:	Digital Signal Processing
Course Number:	ELE 792
Semester/Year (e.g.F2016):	F2023

Instructor:	Anwar Mirza
--------------------	-------------

<i>Assignment/Lab Number:</i>	3/Project
<i>Assignment/Lab Title:</i>	(1) Frame-Based Signal Processing and FFT Implementations on the TMS320C6713 DSK (2) Media Authentication with Electric Network Frequency Analysis

<i>Submission Date:</i>	December 8, 2023
<i>Due Date:</i>	December 7, 2023

Student LAST Name	Student FIRST Name	Student Number	Section	Signature *
Abdur-Rahman	Taskin	500973553	09	

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report

is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: <http://www.ryerson.ca/senate/current/pol60.pdf>

Introduction

The objective of this project is to explore frame-based processing compared to stream-based processing and to implement frame-based fast fourier transform on an audio signal to analyze its frequency magnitude response and compare it to the reference Electric Network Frequency (ENF). The aim of Lab3 is to implement the triple buffering technique found in frame-based processing to optimize the efficiency of the processing. The goal of Lab3 is to implement an ENF function that can extract the ENF signal from a given audio signal and generate it's max and weighted energies. It is possible to verify the geographical origin of a signal by comparing its ENF to the reference ENF of a location. In a research paper, the ENF of flickering fluorescent lights was analyzed and compared to the ENF captured from the power mains supply and a high correlation was found between them. The technique is to use frame-based processing and perform math operations on each block of data to produce the magnitude frequency response signal. The input signal is segmented into blocks, then each block is windowed and zero-padded before calculating the STFT (Short-Time Fourier Transform). The absolute value magnitude of the frequency response is analyzed along with its max and weighted energies. The max energy is simply the max energy of the calculated magnitude spectrum for each block, while the weighted energy is a weighted average of the energy in each block. The signal processing techniques used in Lab3 will be discussed, followed by the ENF function implementation used in the project. The results showing the comparison of the different methods in Lab3 will be discussed followed by the ENF of an audio signal compared to its reference ENF.

Design Methodology

Lab 3

Stream-based processing vs frame-based processing

Signal processing typically involves processing the input, then calculating the output. Stream-based processing implements this by taking 1 input value at a time and calculating its output. Frame-based processing involves taking multiple input values at once and calculating their output. In Lab3, frame-based processing is used to optimize efficiency in processing the output; instead of calculating the output for each input value, calculating the output for a block of data at once can be much faster. Frame-based processing should be used when processing is computationally heavy and time consuming. If the time to process the output value cannot be performed before the next input value arrives, then the input signal will be corrupted. This constraint can be described in the following equation (1), and is described in **Figure 1** [1].

$$T_{overhead} + T_{processing} \leq T_s \quad (1)$$

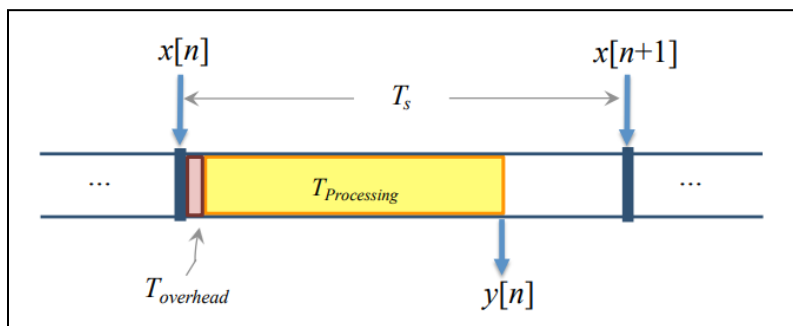


Figure 1: Timeline diagram describing constraint between sampling period and processing time.

A technique called triple buffering is used in frame-based processing to ensure that no input values are lost while processing the output and to improve time efficiency. Triple buffering is a method of processing where there are 3 data buffers; one buffer is used for taking input, another is for processing the output and the last buffer is used for output. Triple buffering used with interrupt-based input/output allows input to be processed while the output is being calculated, and processing begins when a new block is ready. When an input value arrives the interrupt is processed, once it is finished the processing of the output continues.

Windowing and Overlap

Windowing and overlapping is done in signal processing to minimize spectral leakage, which is done in order to get an accurate frequency spectrum. When an input signal is not an integer number of cycles, which causes a discontinuity between the two endpoints. This sharp transition is interpreted by the FFT as high-frequency components that are not actually present in the signal; this phenomenon is called spectral leakage. The solution to this problem is called windowing, which is a mathematical operation applied to a signal to eliminate the large discontinuity at the endpoints. The hamming window in **Figure 2** [2], multiplies the time domain signal by a function which is 1 in the middle and gradually attenuates to near zero at its endpoints. Therefore applying a hamming window to our signal will cause the “envelope” of the signal to look like the hamming window as can be seen in **Figure 3**.

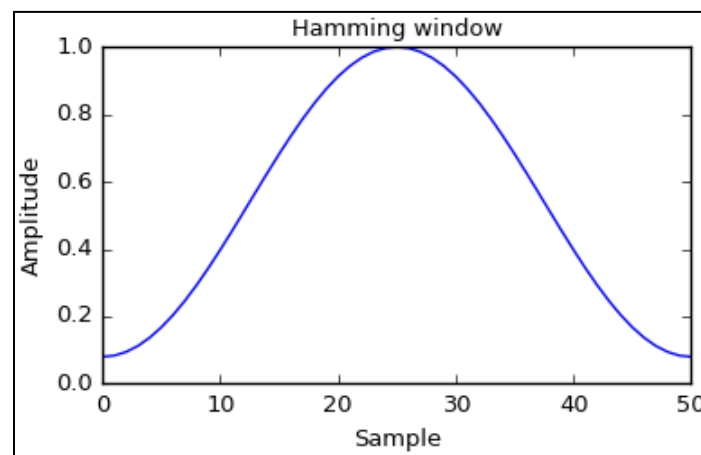


Figure 2: Hamming window function

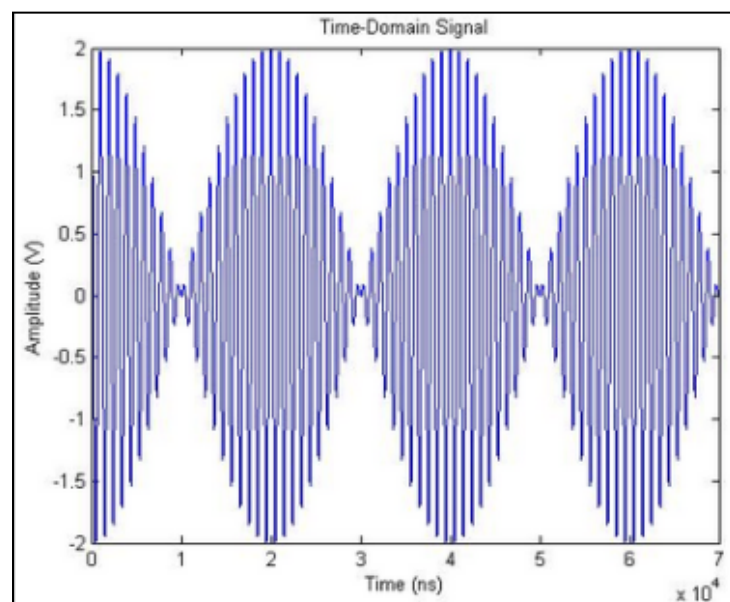


Figure 3: Hamming windowed input signal.

Overlap is then used with a windowing function to ensure that no information is lost from the original signal. The hamming window attenuates the endpoints of the signal close to 0 so any information near the endpoints are minimized in the frequency domain. Overlap in digital signal processing refers to the inclusion of common samples in adjacent signal segments during the processing of consecutive segments. Using overlapping with windowing allows us to retain the original information while minimizing the spectral leakage. The results of windowing and overlapping can be seen in **Figure 4** [3].

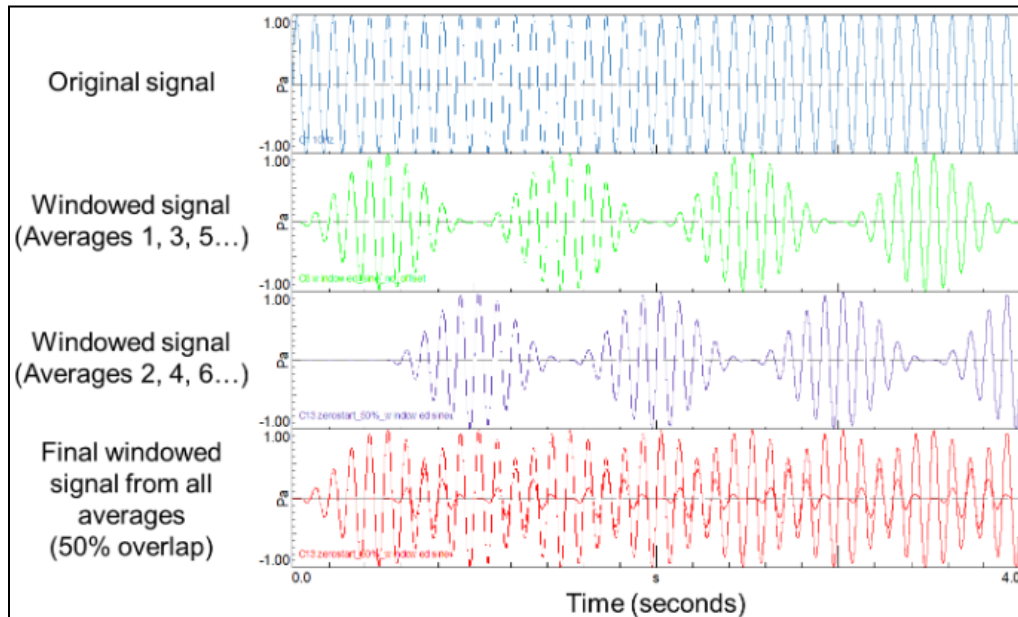


Figure 4: Process of windowing and overlapping an input signal and their results.

FFT vs DFT

The FFT (Fast Fourier Transform) and DFT (Discrete Fourier Transform) are both methods of implementing the Fourier transform for discrete-time signals. The key difference between these two methods is the algorithm they use. The FFT is an implementation of the DFT that is more efficient in both time and memory, this is because the algorithm used by the FFT eliminates redundant calculations that are performed by the DFT and instead it makes use of previously calculated values. Due to this reason the FFT is used in signal processing applications since it is computationally efficient, while using a DFT would be computationally heavy in most real world applications.

Project

Real-time vs offline implementation

Real-time processing of an ENF signal means the signal must be continuously monitored and generated through frame-based processing. While offline implementation allows for batch processing, which can be more computationally efficient. Real-time processing restricts certain techniques that can be applied in offline methods, but it also requires techniques that are only found in real-time scenarios. Some of these are, the use of ENF extraction algorithms, the need for timestamping to maintain records of variations, adaptive filtering to handle noise in real-time scenarios, and more.

ENF function structure

The ENF function can be described as a list of sequential steps given a list of parameters. The section below shows the parameters required followed by the algorithm for the function.

Parameters

1. x - input audio signal.
2. F_s - sampling frequency of the input signal.
3. *BlockSize* - size of each block for frame-based processing.
4. *ZeroPad* - number of 0's to be appended to the end of each block.
5. *Overlap* - percentage of overlap to be applied to each block.
6. *Window* - type of window function to be applied to each block.
7. *Frequency* - the frequency harmonic of the ENF to be generated.

Algorithm

1. Input segmentation
 - 1.1. The *BlockSize*, *Overlap* and the length of x are used to calculate the dimensions of the “index matrix”.
 - 1.2. The “index matrix” is a matrix of the indices of the input signal that the segmented matrix should contain.
 - 1.3. Then the matrix indexing technique in matlab is used to generate the segmented blocks.
2. Windowing
 - 2.1. Applying the windowing function to each block using the *Window* parameter.
3. Zero padding
 - 3.1. Applying *ZeroPad* number of 0s to the end of each block in the segmented matrix.
4. FFT magnitude spectrum
 - 4.1. Applying the `fft()` function to the windowed and zero-padded matrix.
5. Extracting desired frequency range
 - 5.1. Using F_s , length of x and *Frequency* to create an index array to acquire the desired frequency range.
6. Max and weighted energy
 - 6.1. Applying the function for `max()` for max energy of the desired frequencies of the spectrum.
 - 6.2. Executing the math operations required for the weighted energy of the spectrum as seen in the equation (2) below.

$$F_{wf}[m] = \frac{\sum_{k=K_1}^{K_2} F[k] \mathcal{E}[k, m]}{\sum_{k=K_1}^{K_2} \mathcal{E}[k, m]}$$

(2)

Results

Lab 3

Clock Timings

The goal of this lab was to create a function combining the best of FFTr2.c and FFT128.c. The advantage of using FFTr2.c is the use of an efficient optimized floating-point FFT function that was coded in Assembler from Texas Instruments. The advantage of FFT128.c is that it implements the triple buffering technique mentioned previously. The clock timings in the table show that FFT128.c is faster than the regular DFT implementation and FFTr2.c is faster than FFT128.c without triple buffering. When triple buffering was implemented with the use of the optimized floating-point FFT function the clock timings were faster than both.

Table 1: Clock timings of different DFT methods.

Method	Run #1	Run #2	Run #3	Average
DFT	3,147,287	3,147,807	3,147,813	3,147,635.67
FFT128	134,577	134,573	134,568	134,572.67
FFTr2	7,689	7,689	7,689	7,689
Combined	4,421	4,305	4,343	4356.33

Oscilloscope comparison

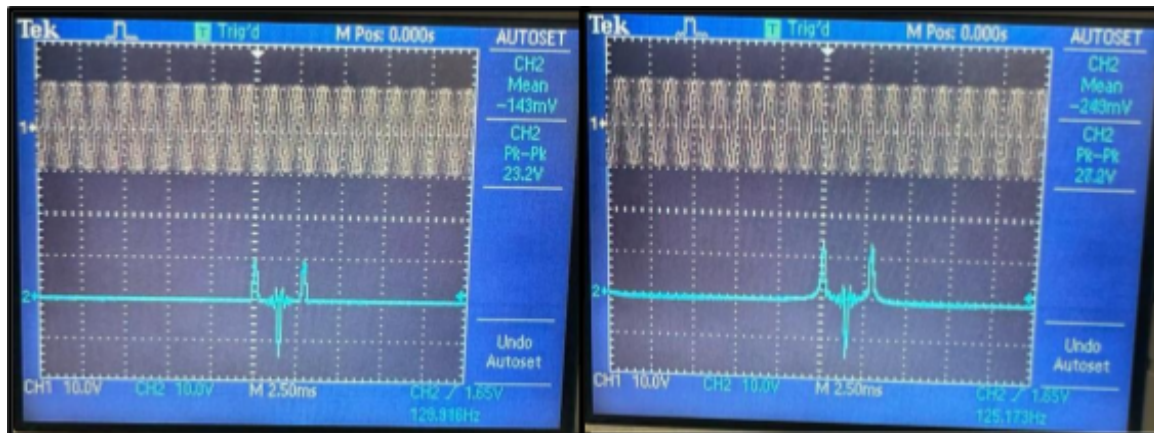


Figure 5: Hamming windowed FFT (left) vs. non windowed FFT (right).

Project

Surface plots

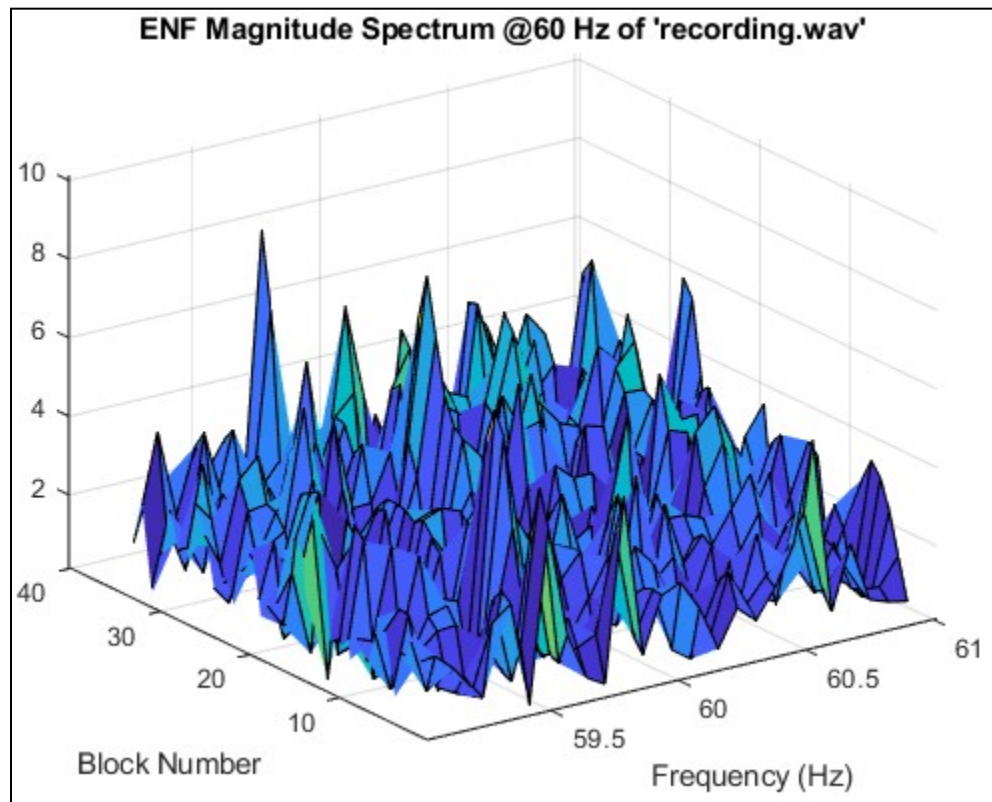


Figure 6: Surface plot of ENF @60Hz harmonic.

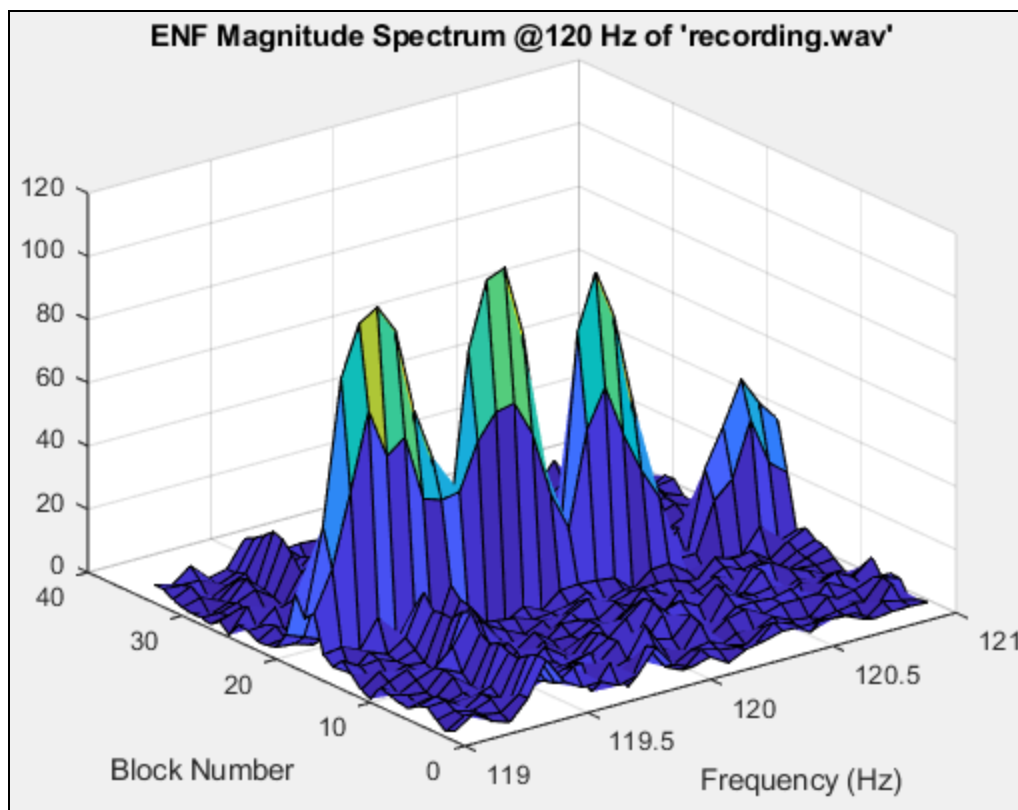


Figure 7: Surface plot of ENF @120Hz harmonic.

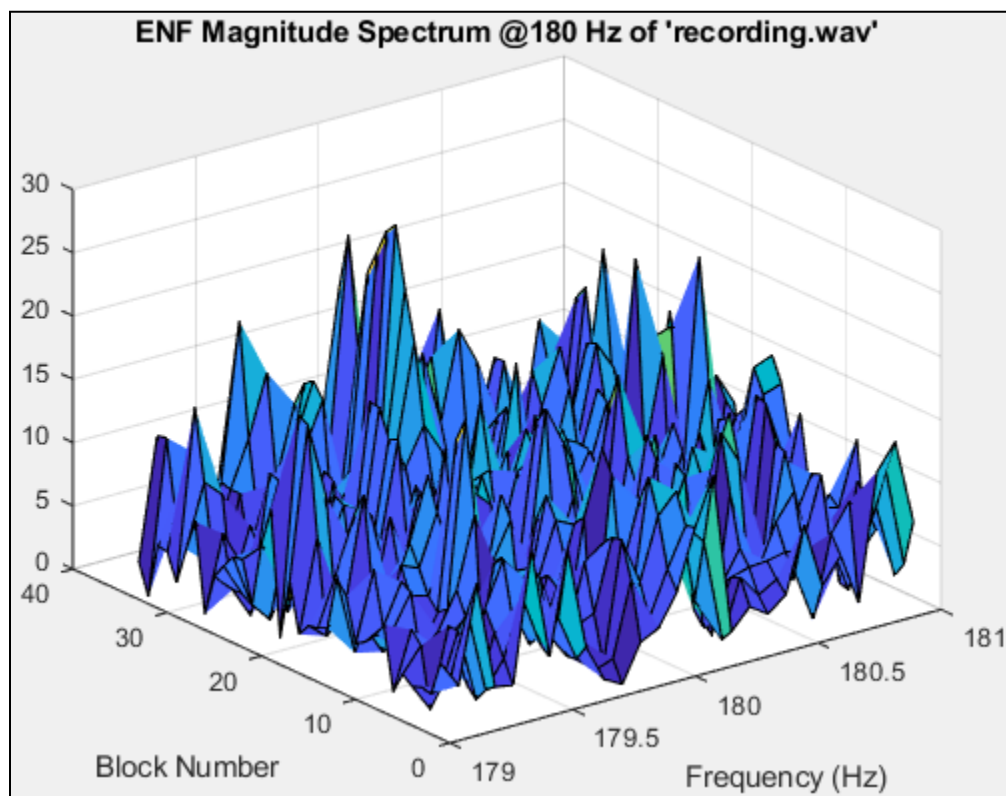


Figure 8: Surface plot of ENF @180Hz harmonic.

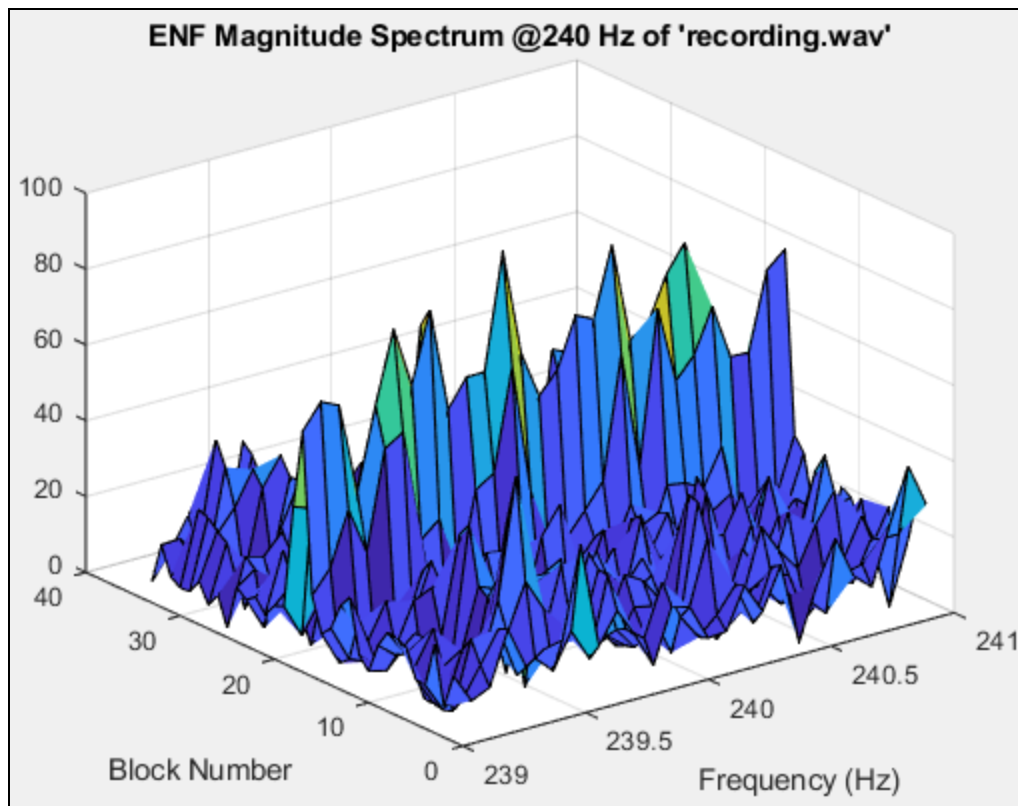


Figure 9: Surface plot of ENF @180Hz harmonic.

Looking at the results in ENF for different harmonics 120Hz is the best.

Comparison between Max and Weighted energy

The maximum magnitude does not provide much of an analysis on the ENF signal comparison due to the mathematical formula being applied to the FFT. The weighted energy however provides a much better comparison between the recording and reference signals. Using the weighted average formula we receive a better resolution as we get a wider range of values.

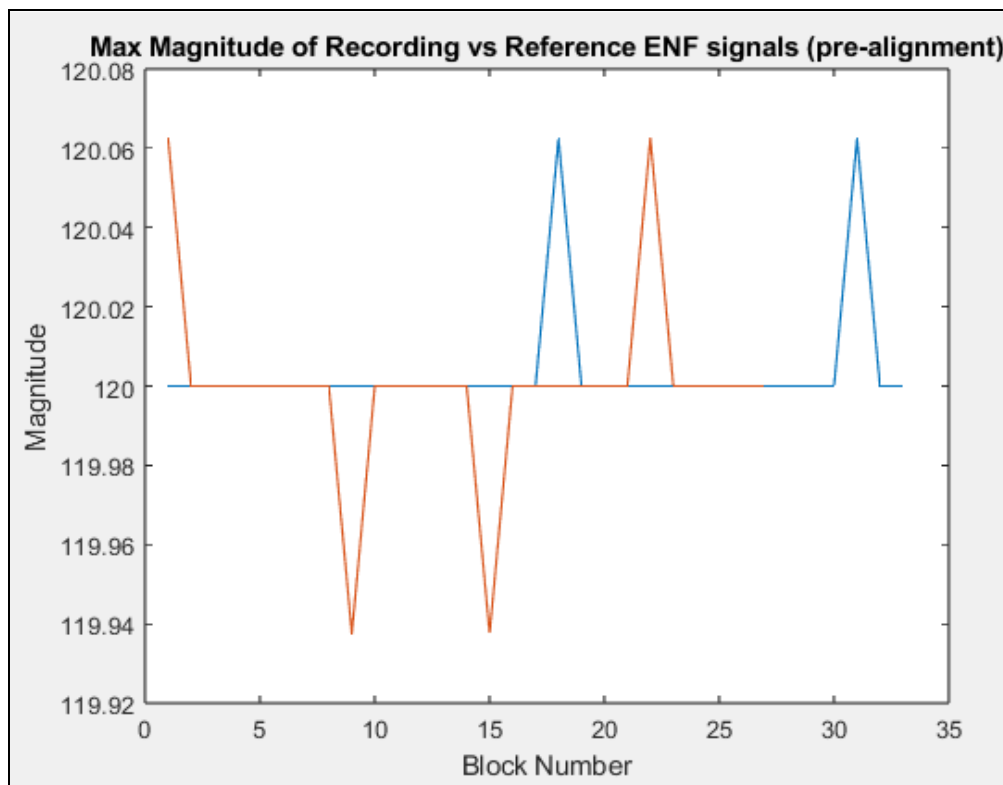


Figure 10: Max magnitude plot of recording and reference signal (pre-alignment).

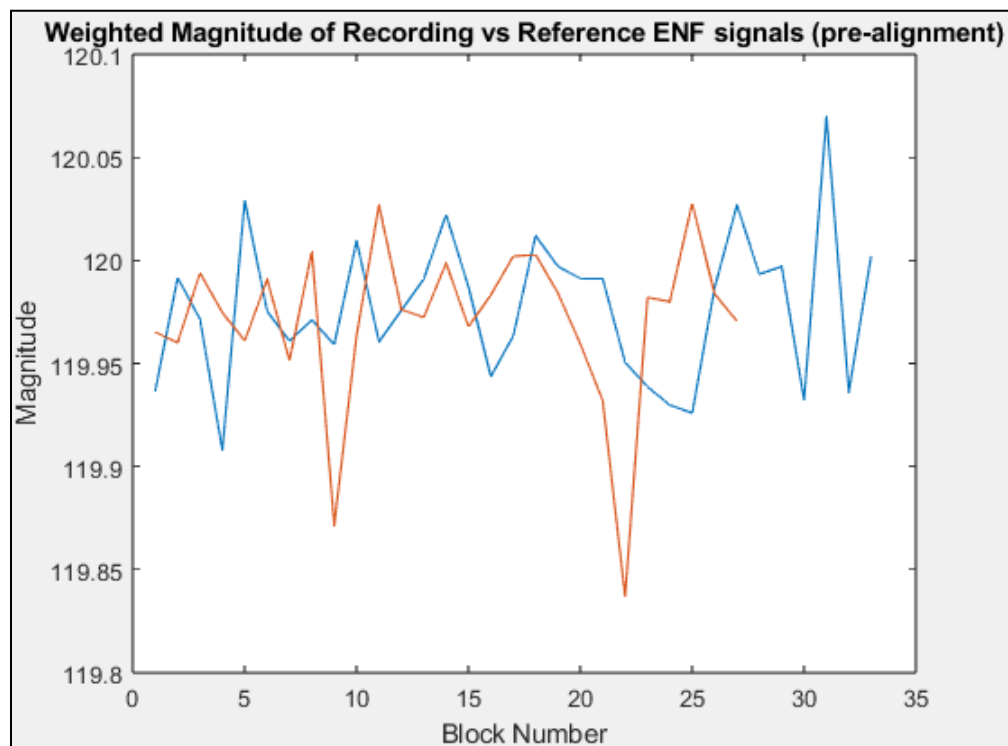


Figure 11: Max magnitude plot of recording and reference signal (pre-alignment).

Recording vs Ground Truth

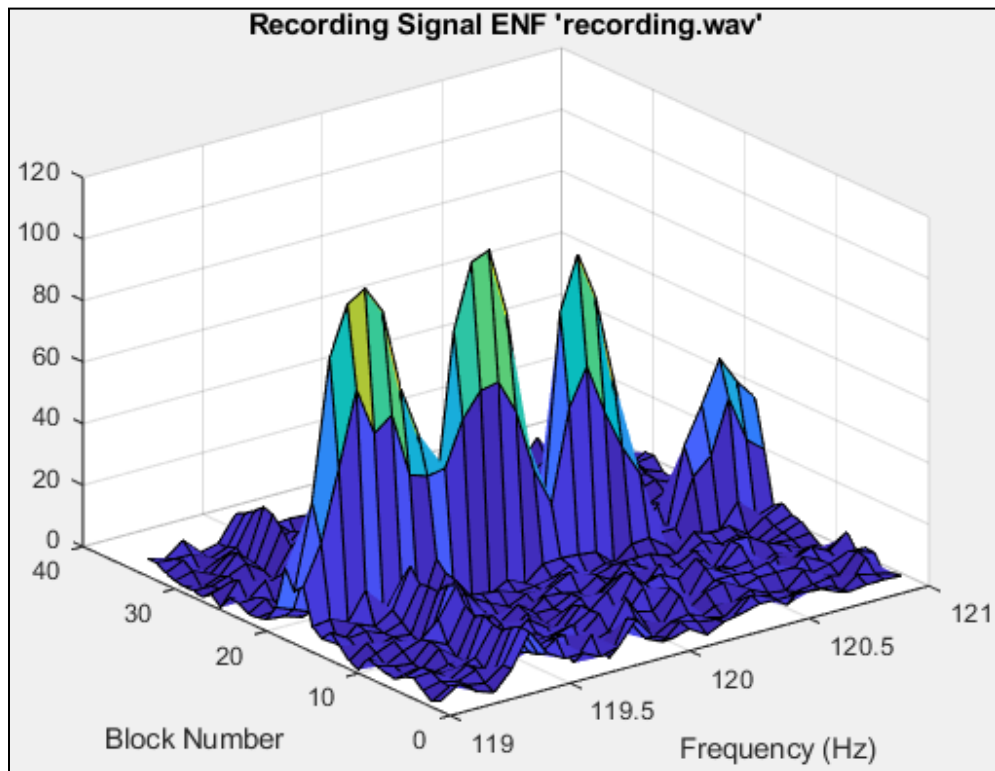


Figure 12: Surface plot of “recording.wav” @120Hz harmonic.

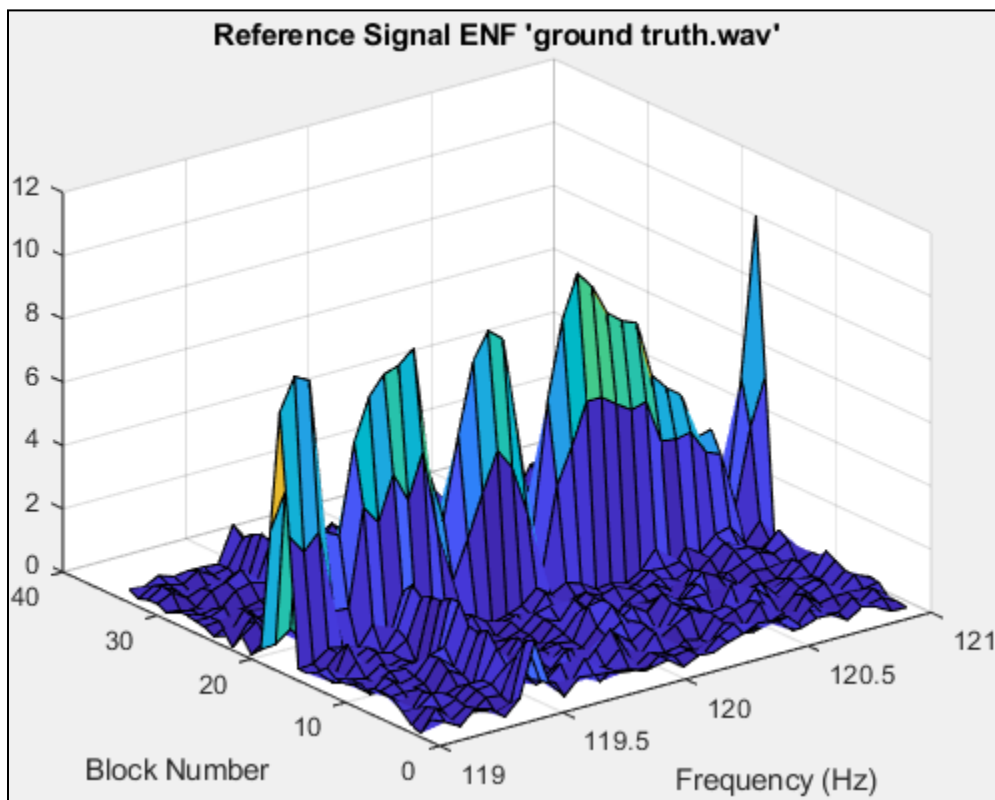


Figure 13: Surface plot of “ground truth.wav” @120Hz harmonic.

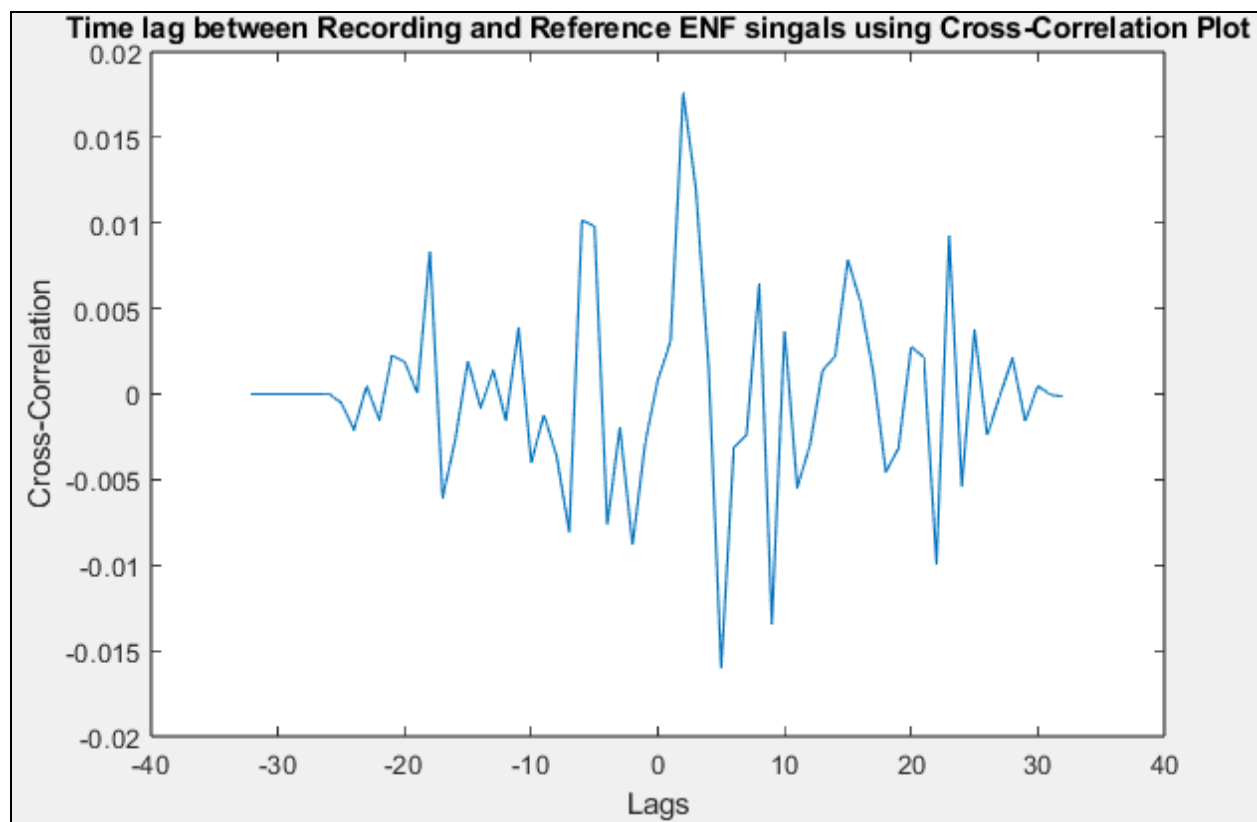


Figure 14: Cross-correlation plot of “recording.wav” and “ground truth.wav” ENF signal @120Hz.

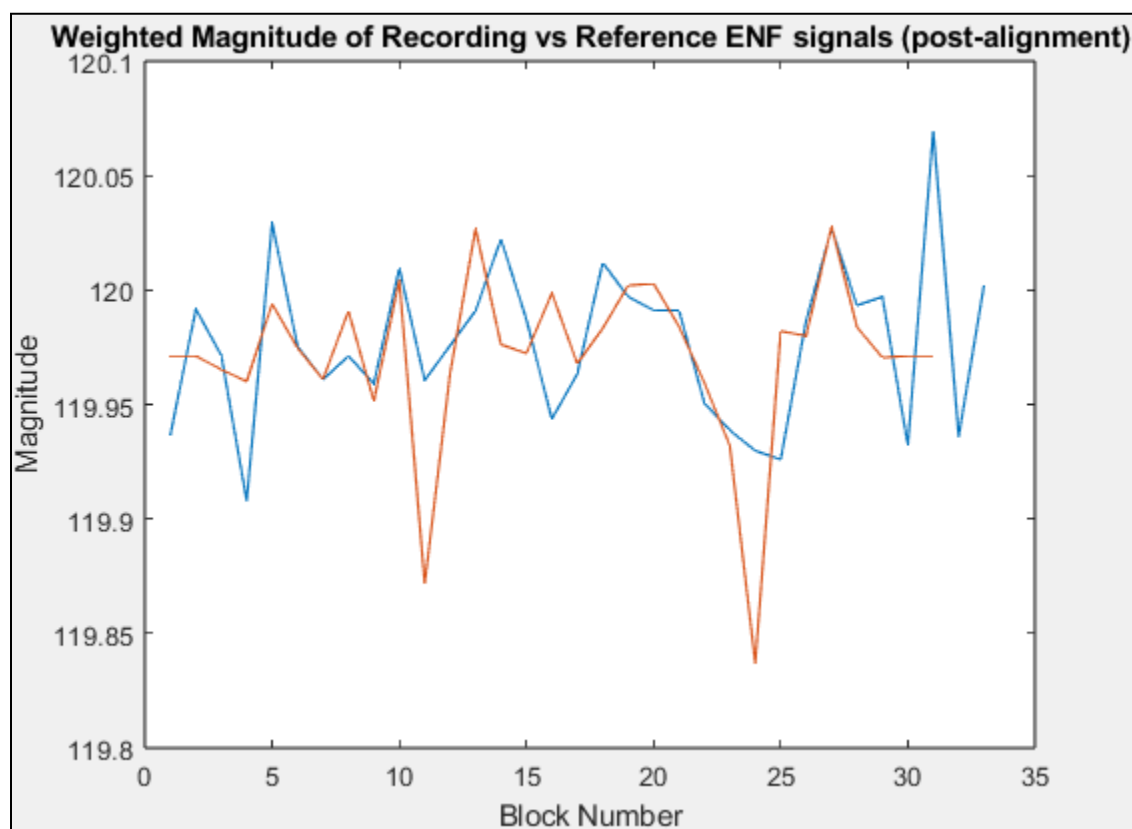


Figure 15: Weighted magnitude plot of recording and reference signal after alignment.

The “recording.wav” signal was found to be 2 delays off from the reference signal “ground truth.wav”. This delay corresponds to 16s of delay from the audio signal.

Recording vs Ground Truth - filtered and down-sampled

Down-sampling and filtered results in a better resolution for the graph and a cleaner plot, as the lower frequency noise gets eliminated from the signal.

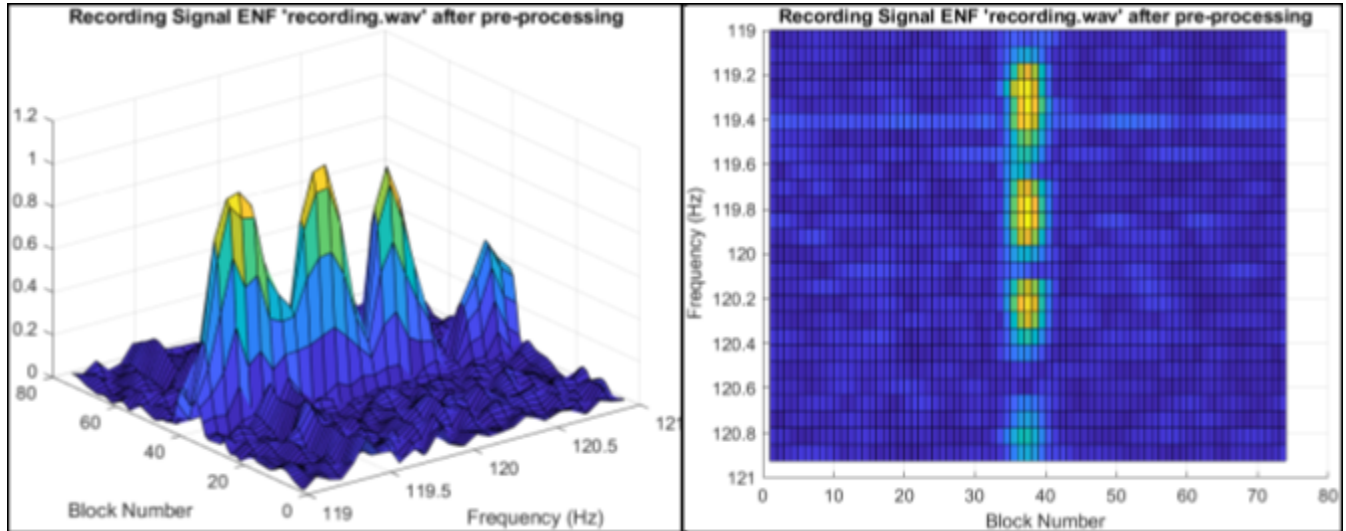


Figure 16: Filtered and down-sampled surface plot of “recording.wav” @120Hz harmonic.

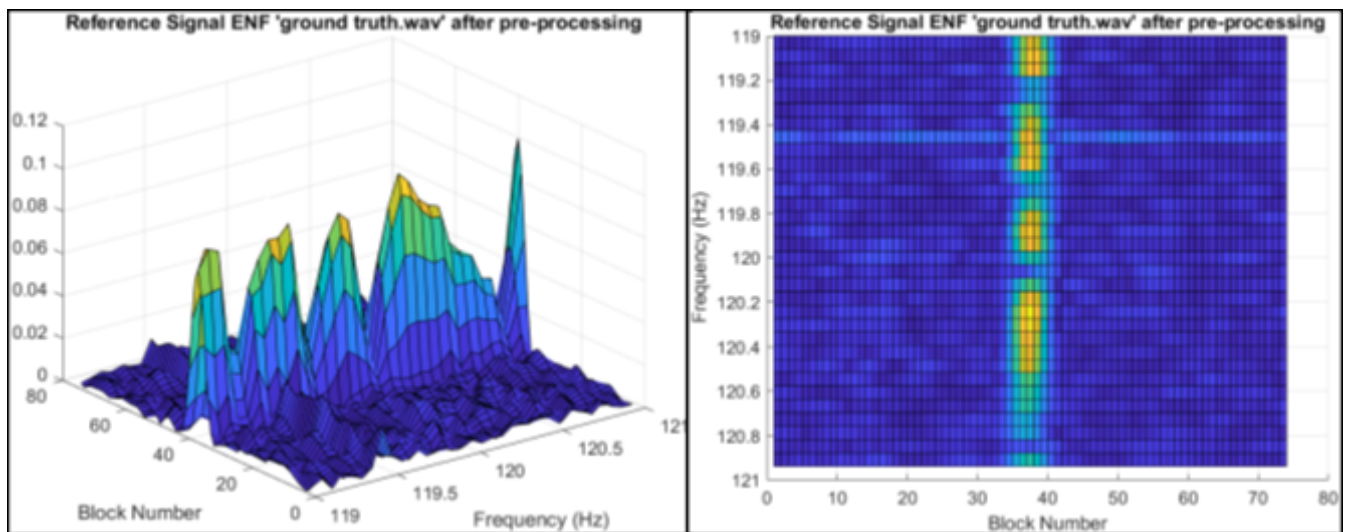


Figure 17: Filtered and down-sampled surface plot of “ground truth.wav” @120Hz harmonic.

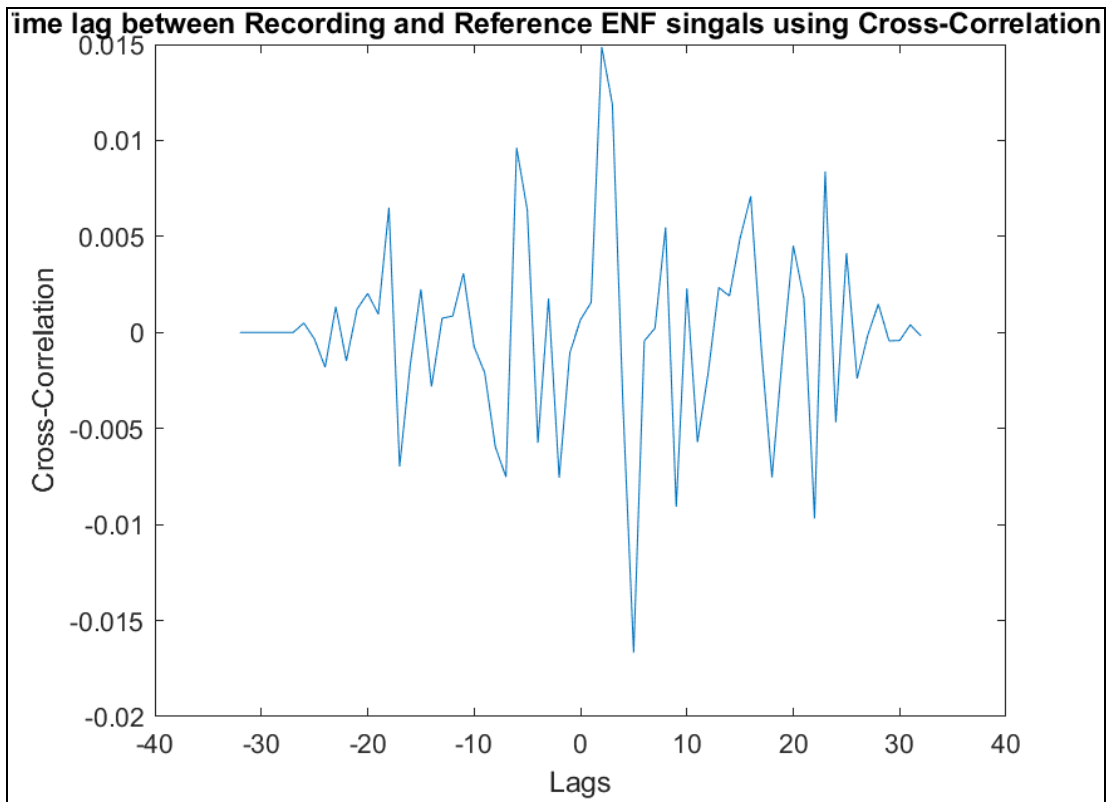


Figure 18: Cross-correlation plot of “recording.wav” and “ground truth.wav” ENF signal @120Hz after down-sampling and filtering.

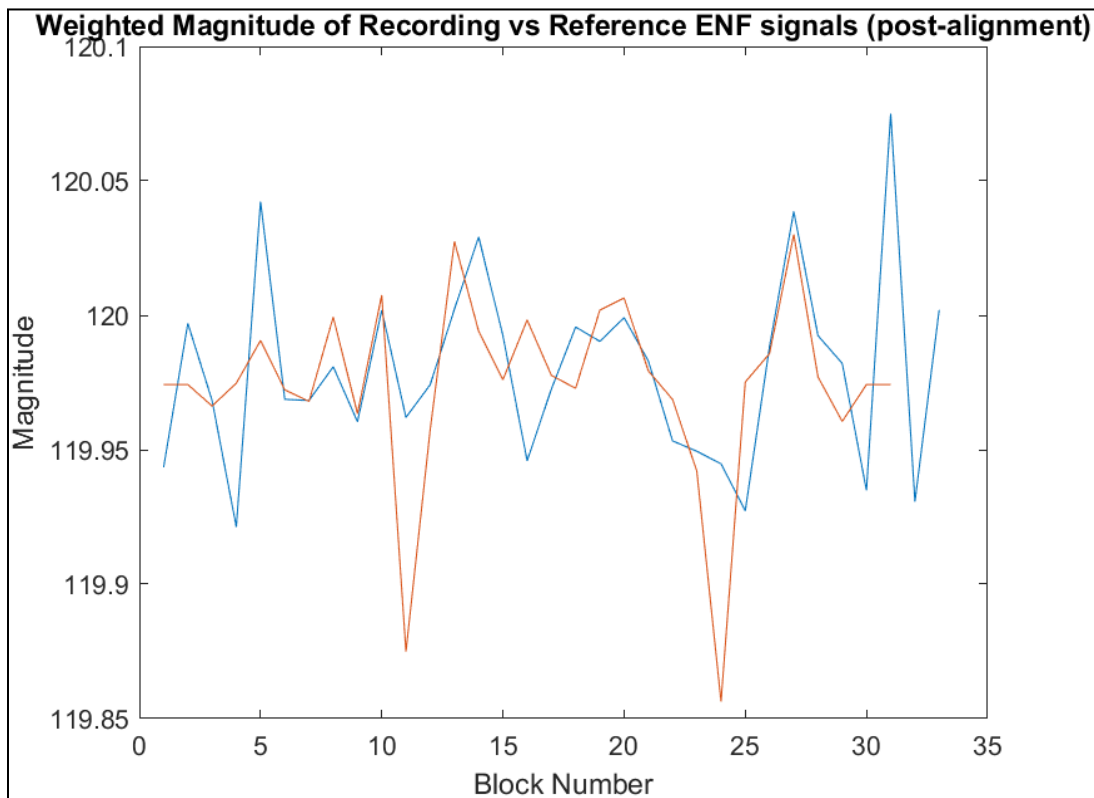


Figure 19: Weighted magnitude plot of recording and reference signal after alignment, down-sampling and filtering.

Recording vs Ground Truth 2 - filtered and down-sampled

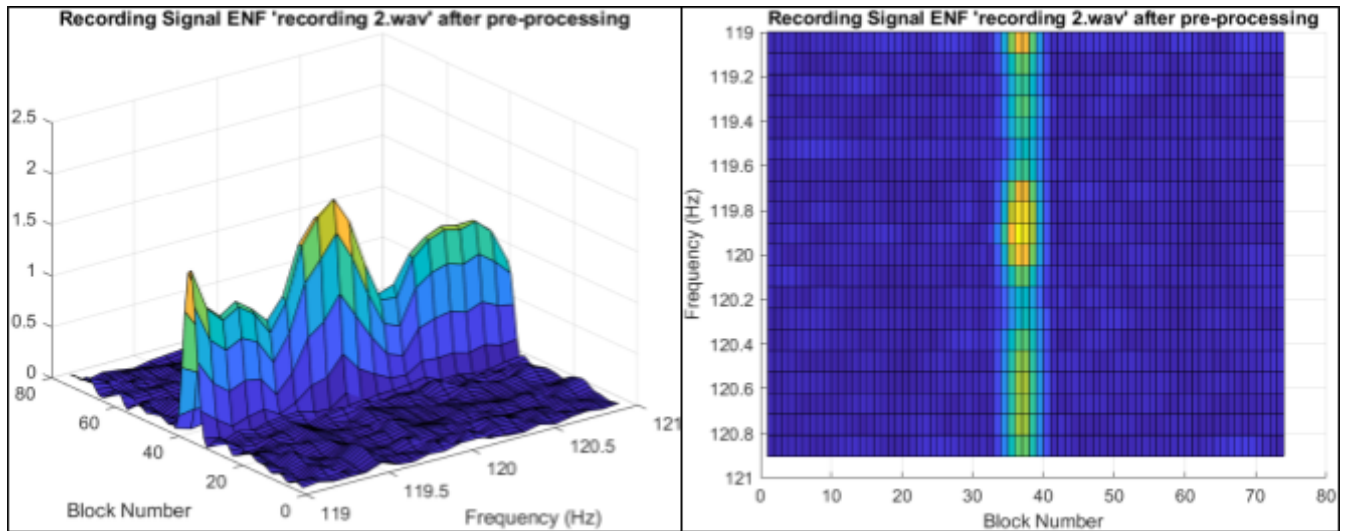


Figure 20: Filtered and down-sampled surface plot of “recording 2.wav” @120Hz harmonic.

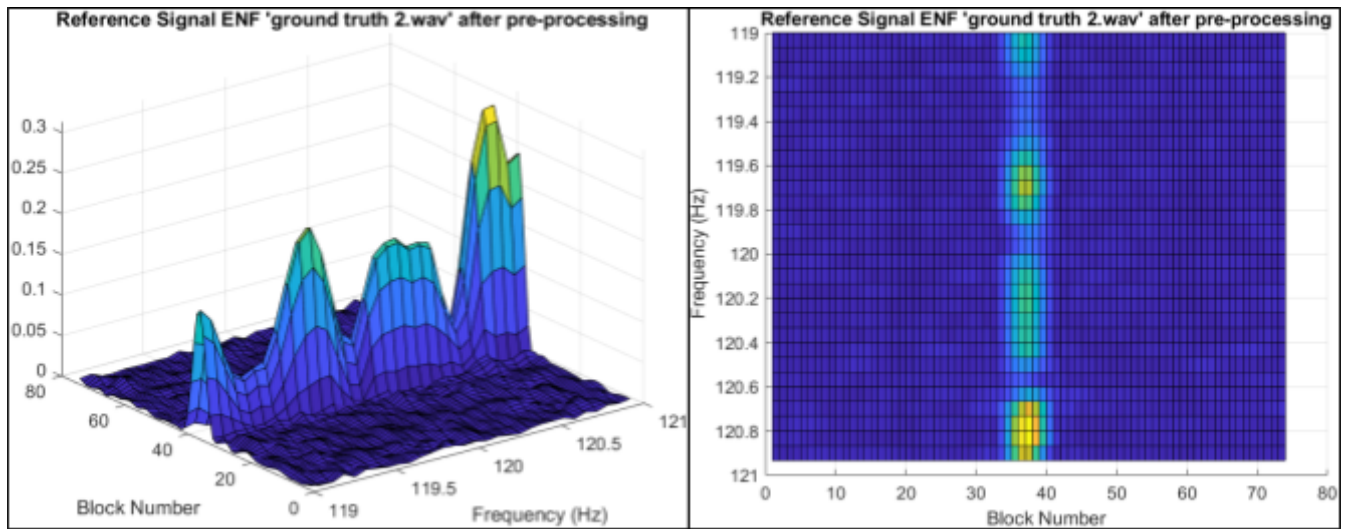


Figure 21: Filtered and down-sampled surface plot of “ground truth 2.wav” @120Hz harmonic.

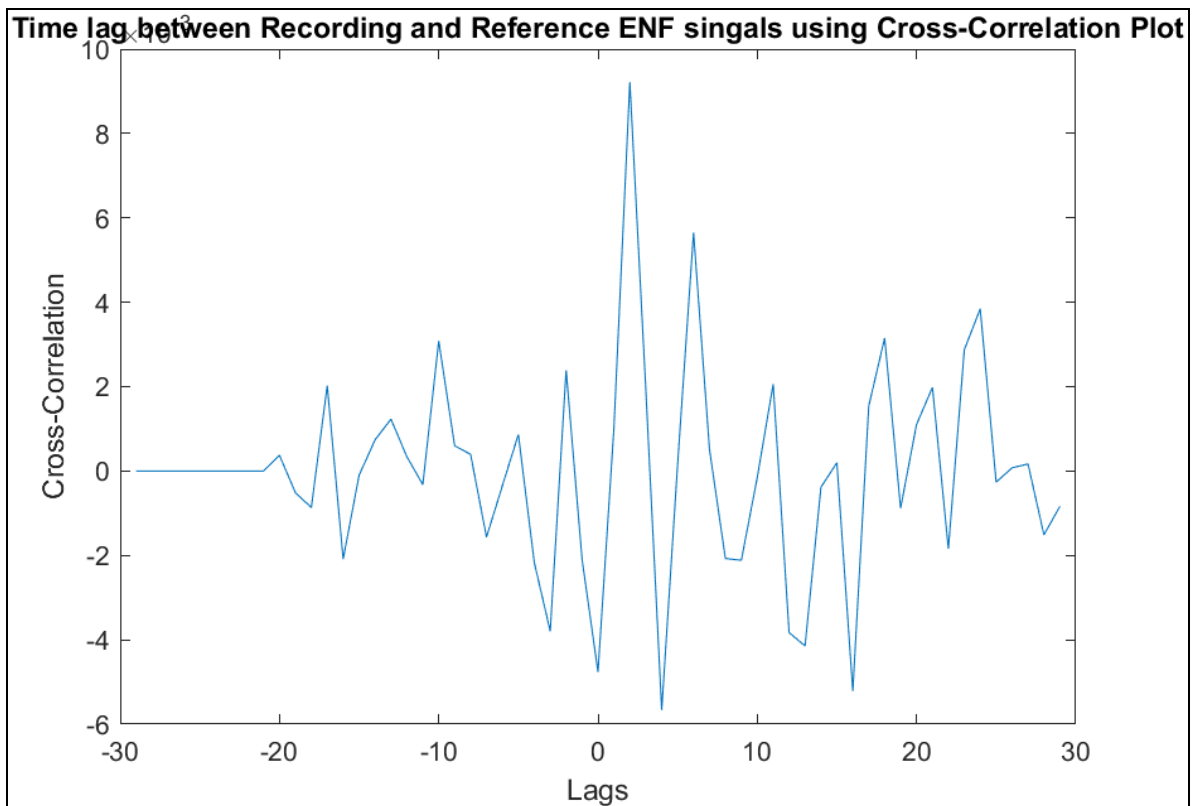


Figure 22: Cross-correlation plot of “recording 2.wav” and “ground truth 2.wav” ENF signal @120Hz after down-sampling and filtering.

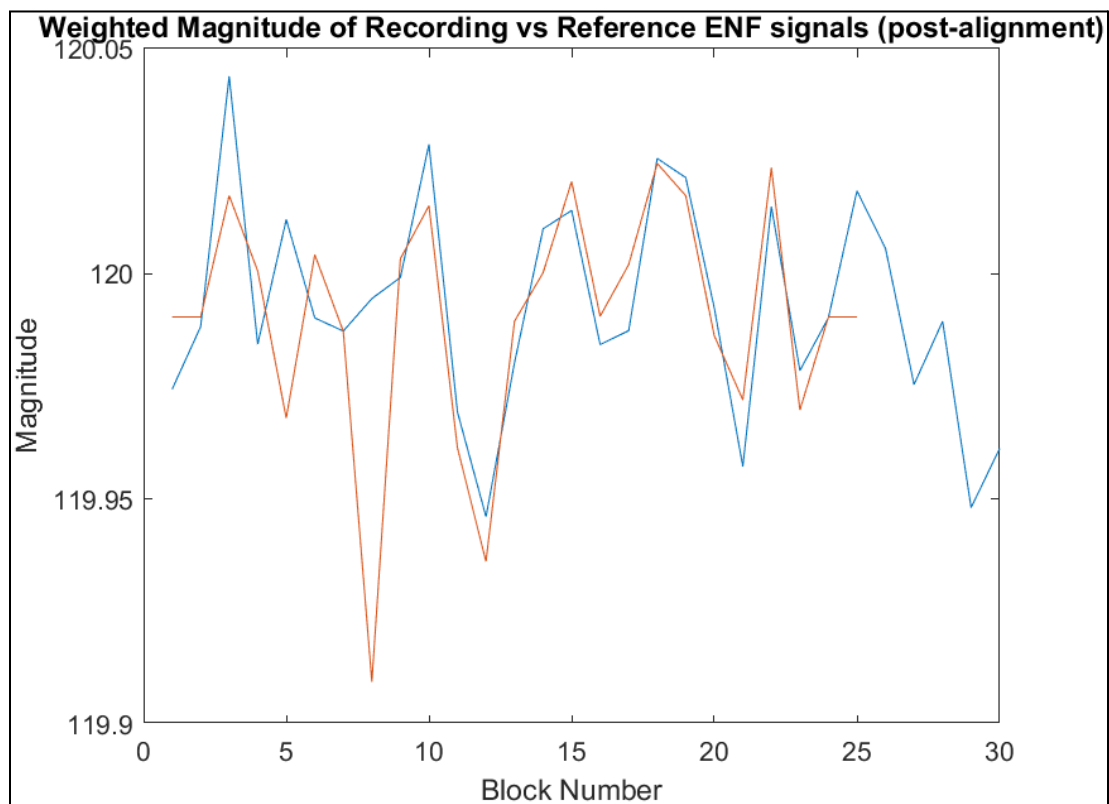


Figure 23: Weighted magnitude plot of recording and reference signal after alignment, down-sampling and filtering.

Conclusion

In Lab 3 triple buffering was combined with Texas Instruments' (TI) optimized FFT function to create a program that generates an FFT given an input signal. This program was faster than the triple buffering program implemented in FFT128.c and the TI's optimized FFT function program implemented in FFTr2.c. The result of which confirmed the effectiveness of frame-based processing in computation heavy signal processing applications. For the project it was found that the weighted average magnitude plot was the most effective in comparison between an audio signal's ENF and its reference signal's ENF. It was also found that the audio signal was 16s delayed from its reference signal using the cross-correlation plot. The results in the project and Lab3 demonstrate how the frame-based processing is useful in computational heavy processing such as analyzing the media authentication using ENF analysis.

References

- [1] "Lab 3 FFT PresentationSlides 2018". Ryerson University. [Accessed: December 07, 2023].
- [2] The Scipy community**. (2017, June 21). scipy.signal.hamming - SciPy v0.19.1 Reference Guide. [https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.signal.hamming.html#:~:text=The%20Hamming%20window%20is%20a,normalize%20the%20nearest%20side%20lobe.&text=Number%20of%20points%20in%20the,an%20empty%20array%20is%20returned.&text=When%20True%20\(default\)%2C%20generates,for%20use%20in%20filter%20design.](https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.signal.hamming.html#:~:text=The%20Hamming%20window%20is%20a,normalize%20the%20nearest%20side%20lobe.&text=Number%20of%20points%20in%20the,an%20empty%20array%20is%20returned.&text=When%20True%20(default)%2C%20generates,for%20use%20in%20filter%20design.)
- [3] Macdonald, s. (2022, January 10). *Overlap: What, Why and How to use it*. Siemens DISW. <https://community.sw.siemens.com/s/article/Overlap-What-Why-and-How-to-use-it>