Frequency Estimator Design

ECE301 Final Project

Maxwell Riach
Electrical and Computer Engineering
North Carolina State University
Raleigh, USA
msriach@ncsu.edu

Arjun Nair
Electrical and Computer Engineering
North Carolina State University
Raleigh, USA
asnair2@ncsu.edu

Zachary Guess
Electrical and Computer Engineering
North Carolina State University
Raleigh, USA
zjguess@ncsu.edu

***Abstract*—This document highlights the design and simulation of a frequency estimator.The electric network frequency (ENF) in the US has a nominal value of $f0 = 60Hz$ and its instantaneous value fluctuates round the nominal value as a function of time. The ENF signals can be captured by audio or video recordings made in areas where there is electrical activity. This makes the ENF a good criterion for the forensic analysis of a multimedia recording. Using MATLAB and other resources, we have created a program that can accurately capture and calculate several points of info from an ENF signal.**

I. Introduction

The electric network frequency (ENF) in the US has a nominal value of $f0 = 60Hz$ and its instantaneous value fluctuates round the nominal value as a function of time. "Fig. 1" The following plots show some sample ENF signals from various power grids.

The ENF signals can be captured by audio or video recordings made in areas where there is electrical activity. This makes the ENF a good criterion for the forensic analysis of a multimedia recording. With a proper reference database, one can tell when and where a recording was made, and whether the recording has been tampered.

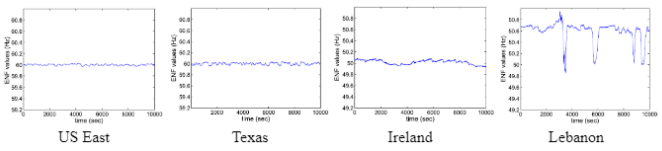

Fig. 1. ENF signals from various power grids.

II. Design of the ENF Signal

Let us formally define the ENF signal as

$$f(t) = f_0 + u(t) \tag{1}$$

where f_0 is the nominal value and $u(t)$ is the random, fluctuating component. Research found that the random signal $u(t)$ can be modeled as an auto regressive (AR) random process. In this project, we assume that its sampled version (with a discretization step size of 0.2 seconds) follows the first-order AR process, namely,

$$u[n] = 0.9u[n-1] + e[n] \tag{2}$$

where the $e[n]$ is a zero-mean white Gaussian noise process with standard deviation $\sigma = 0.05Hz$. Note that the discretization leads to the relationship of $n = t/0.2$, and the discretized ENF signal can still be decomposed into

$$f[n] = f_0 + u[n] \tag{3}$$

A. Simulating ENF Signals

As required by the assignment 5 different discrete ENF signals each 10 minutes long have been plotted in "Fig. 2", with the dynamic range of the vertical axis being limited to around 60 Hz to ensure that the fluctuation can be easily observed. The second part of figure 2 shows an enhanced image of the first 100 seconds of the ENF signals from the first part of the plot. This enhancement allows us to easily see the separate signals and how they fluctuate over time.

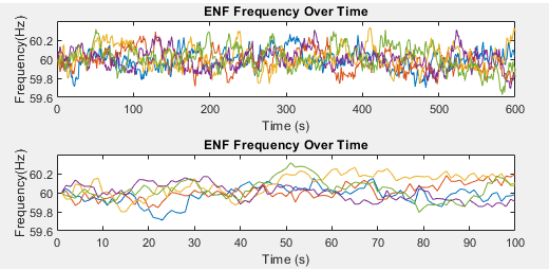

Fig. 2. 5 discrete ENF signals each 10 minutes long

Since the calculated standard deviation of the simulated ENF signals noise is about 0.11 Hz, we will likely see 99.8 percent of our signals within the range of 5.67 and 6.33 Hz. Based on this calculated error, our ENF simulation appears to be generating correctly.

B. Simulating Voltage Signal of Power Supply

Using frequency modulation with a sampling frequency of $f_0 = 150Hz$ a long sinusoidal-like signal $x[n]$ whose frequency changes every 0.2 second is generated for each ENF signal. Using the formula,

$$x[n] = cos(\phi + 2\pi \sum_{l=1}^{n} f[l]/f_s) \quad (4)$$

The resultant Amplitude vs Time of each of the 5 discrete ENF signals can be seen in "Fig. 3".

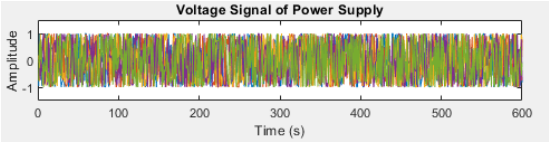

Fig. 3. Amplitude vs Time of each of the 5 discrete ENF signals

In this case phi is a randomly generated initial phase for the frequency.

C. Implement Frequency Estimator

To analyze our sinusoidal and ENF functions we are tasked to create a periodogram-based frequency estimator. Using our estimator, we find the estimated power spectral density (PSD) of our signals by applying a fast Fourier transform (FFT) then square magnitude the found output. Note that the PSD is a real, non-negative function of frequency. It is important to compare the frequency estimator with the original function built within MATLAB to ensure the accuracy of the PSD created. Comparing across different frequencies as well was ensured in order to create a higher degree of accuracy whilst creating the frequency estimator. Note that since testing this estimator on a 10-minute long recording, each time it is important to only consider a small window of the signal. Throughout testing of this function, it was vital to use test signals to compare between Matlab integrated function of creating a periodogram and the function created to ensure the accuracy behind out function.

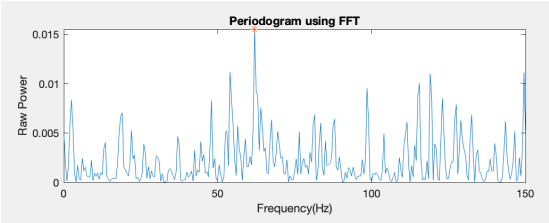

Fig. 4. Periodogram using FFT Example 1

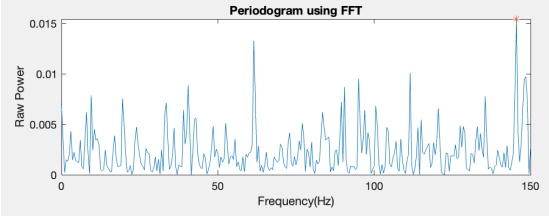

Fig. 5. Periodogram using FFT Example 2

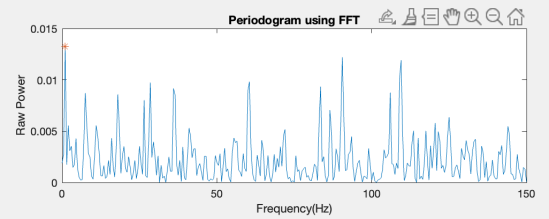

Fig. 6. Periodogram using FFT Example 3

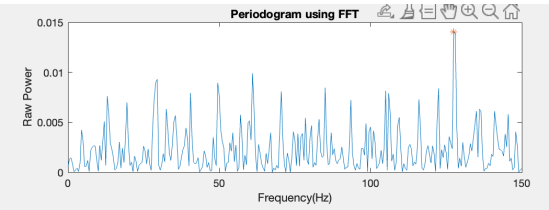

Fig. 7. Periodogram using FFT Example 4

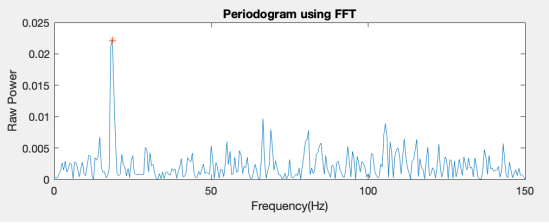

Fig. 8. Periodogram using FFT Example 5

D. Frequency Interpolation

Since the nfft parameter can only be an integer multiple of the frequency resolution, we created a frequency interpolation process. This process sharpens the estimated frequencies found in our Frequency Estimator section. We not only consider the frequency of the largest PSD value, but also the PSD value to the left and right of the largest. Considering the left and right PSD will give the interpolation a better smoothness to show the largest PSD value.

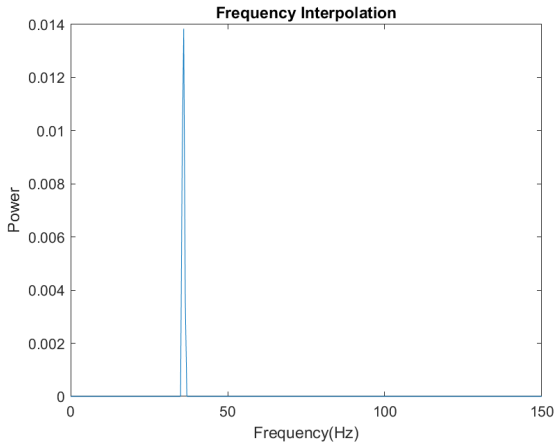

Fig. 9. Example of Frequency Interpolation

E. Performance of Estimator Under Noisy Scenarios

Using a Signal-Noise-Ratio given by the formula

$$SNR = 10 * log_{10}(A^2/\sigma) \tag{5}$$

where A^2 represents the amplitude of the sinusoidal-like signal $x[n]$, and σ^2 is the variance of the additive white Gaussian noise $v[n]$. $v[n]$ can be represented as follows by rearranging the previous equation from Eq. 5.

$$v[n] = 10^{-SNR/10} * x[n]^2 \tag{6}$$

Plotting $y[n]$ using the equation

$$y[n] = x[n] + v[n] \tag{7}$$

with varying Signal-Noise-Ratio's from -40 to 40 in 10 step increments respectively. The resulting waveform can be seen in Fig. 9.

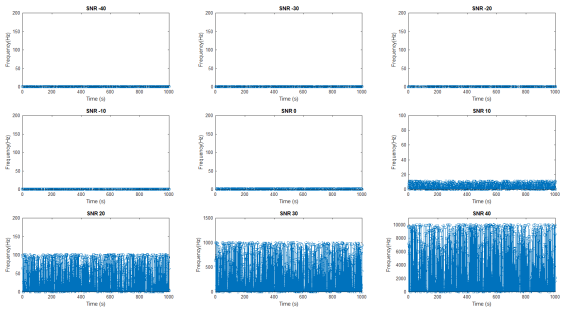

Fig. 10. Signal y[n] with varying Signal-Noise-Ratio's

Using Mean Squared Error (MSE) to analyze the accuracy of the frequency estimates it can be seen that with lower Signal-Noise-Ratio's the MSE would be closer to 0 (i.e. more accurate). However, as the SNR increases the MSE increases as well and would be closer to 1.0, highlighting the inaccuracy as the SNR increases.

F. Discussion of findings on frequency estimation when parameters are changed

When creating a periodogram-based frequency estimator, there are multiple parameters that can affect how a Power Spectral Density (PSD) function can affect how said frequencies are estimated. Just to re-iterate exactly the technique being used, is that there is a moving window where the Fast Fourier Transform (FFT) is computed in each said window, where the PSD is then computed as an average of the FFTs over all windows. To be specific, there are three main parameters that are important for creating a periodogram based around specific input signals. The first is the window length, whereas the second is the percentage window overlap to no overlap, and lastly the number of FFT points. Furthermore, we can even introduce the opportunity to utilize difference windowing functions however the Hanning window is the most widely used as it has good frequency resolution and reduced spectral leakage.

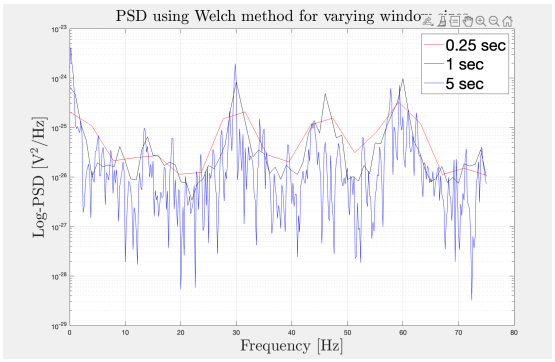

Fig. 11. PSD using Welch method for varying window sizes

The first parameter to go over is window length, where we will be demonstrating how the data is changed with a Matlab function. The number of segments the entire data will be divided into is determined by win and noverlap. We will set the overlaps into 3 different window sizes corresponding to 0.25, 1 and 5 seconds. It is fairly obvious that the smaller window size will increase the total number of windows the data is divided by. This will help smooth out the Power Spectral Display frequency estimates because the random noise will be averaged out. However the downside for the smaller size is that the resolution is compromised as the distance between two points is increased and results in a lower resolution.The ground-truth frequency point is evident from the figure above where we can see that even when the window size is approximately 0.25 seconds with the smooth PSD. Conversely, by increasing the win to 1 second, we have improved the frequency and thus can get a narrower lobe centered around the frequency peak. While the total number of windows has now been decreased due to the increased window length, it is still large enough to cancel out most of the random noise. Lastly, when we analyze when the window is switched to 5 seconds, we can clearly see how the frequency resolution gets a lot sharper by also much noisier as the effects of the noise is not canceled out.

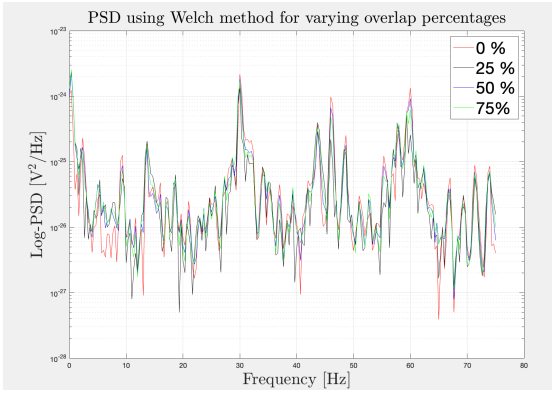

Fig. 12. PSD using Welch method for varying overlap percentages

The second parameter to delve into is how the overlap percentage can affect the Power Spectral Density. For us to compare the affects of the different percentages, it is important

to keep the window size at a consistent size so for the figure above the window is set to 3 seconds. 3 seconds allows the sample to be semi rigid with noise while not compromising the accuracy with an adequate enough resolution. Above we have 4 different varying percentages of overlap ranging from 0, 25, 50, and 75 percent. Looking at the 0 percent overlap, we can see that the PSD is relatively more noisy and bumpy compared to the 50 percent overlap. By increasing the overlap percentages (for a given window size), we increase the total number of window which in turn helps in averaging the effects of noise. However, increasing the overlap percent from 90 to 99, may not help due to the high correlation between the window samples and thus averaging will not cancel the effect of noise.