Automatic Feedback Detection and Elimination System

by

Robert M. Brown Jr.

Submitted in Partial Fulfillment of the Requirements in EE4012

April 23, 1998

Mississippi State, Mississippi

Department of Electrical and Computer Engineering

Special thanks to Dr. Joe Picone and everyone at ISIP for their advice, assistance, and patience in the development of this project.

TABLE OF CONTENTS

LIST OF FIGURES

Abstract

The main focus of this project is to develop a software based system to detect and remove acoustic feedback from a digital audio source. This program will process an audio file or a digital audio data stream in real-time, detect acoustical feedback, and filter the feedback out of the signal. The program will be executable from a command line or through its Graphical User Interface. The acoustical feedback will be detected with the use of peak detection in the frequency domain and removed with notch filters in the time domain.

The technology developed will be applicable to any computer based digital audio system to improve source quality, available gain from the system, protect equipment from unnecessary strain, and protect hearing. This system will be able to be used in a compatible computer based system where the user is recording or processing audio files that contain or have the potential to contain feedback.

I. Introduction

Background

The presence of acoustic feedback in sound amplification systems can cause problems from mild irritation to damage of equipment to the damage of a person's hearing. Feedback is a loud ringing or howling that occurs when the sound leaving a speaker is picked up by a microphone or other source and is re-amplified, Figure 1. In general, this cycle repeats until the system reaches its maximum volume or something is done to break this loop of sound regeneration. [1] Every acoustic system has distinct regions of resonant frequencies, or frequencies that are prone to feedback. These frequencies are characteristic of the room the system is in, as well as each portion of the system: the speakers, the amplifiers, microphones, and other components.

The most basic way of solving the problem of acoustic feedback is to prevent it from occurring in the first place. A system that is set up and maintained correctly can all but eliminate acoustic feedback. Proper speaker and microphone placement, proper equalization, and optimal system gain can all contribute to producing a stable system. Unfortunately, in real-life application not all of these components can be made ideal, thus causing potentials for feedback. Changes in room characteristics, positions of microphones, and characteristics of other equipment that happen during the duration of a performance or the production of sound can create potential for feedback. These rapid system changes call for a more advanced system of monitoring and removing feedback from a system than having a person perform these tasks in real time.

Different Solutions

There are multiple ways of solving any engineering problem. Finding one solution that best fits the particular situation is a greater challenge. There are several different solutions to detecting and removing acoustical feedback that have already been developed. Along with these solutions, several products have been brought to market that claim to solve the problem of acoustical feedback. These products range from $400 to $1000.

An adapting delay comb filtering device, developed by Bernard A. Hutchins and Walter H. Ku, acquires, nulls, and tracks an unwanted periodic interfering signal that is corrupting a desired signal. [2] Phase-modulation principles have also been applied in solving this problem. [3] Some solutions implement phase-locked loops to provide a detection tool for detecting acoustic feedback in a signal. [4] Other methods of detecting feedback are to look at the frequency domain and determine peaks or deviation between points in the frequency domain. [5]

Several products on the market that perform these tasks are made by Sabine, Inc., Roland Corporation U.S., and Peavey Electronics Corporation. Roland makes the AF-70 Anti-Feedback DI. [6] Peavey Mentor Feedback Eliminator product line boasts several

products to detect and remove feedback. [7] Sabine, Inc. has several products in their line of FBX Feedback Exterminators. [8, 9]

Proposed Solution

The proposed solution to implement is a software package written in C++ and TCL/TK to detect and remove feedback from a digital audio source. This solution processes a headerless data file, detects acoustical feedback, and filters the feedback out of the signal. The detection phase is a two-part process: a Fast Fourier Transform (FFT) and a frequency domain peak detection algorithm. The elimination phase consists of designing a second order notch filter at the detected frequency to remove the specific range of frequencies containing the acoustical feedback.

The final product is able to process a headerless data file, detect acoustical feedback, and filter this feedback from the signal. Several aspects of the software are user definable to enable the user to optimize the software to the users specific problem. The user can set the sensitivity of the detection phase by adjusting the threshold above which peaks are filtered. The resolution of the FFT, thus the resolution of the detection phase, allows the user to determine the necessary trade-offs between run time the resolution of detection. Other user definable characteristics of the software are frame duration and window duration.

II. Theory

What is Acoustical Feedback?

Feedback is a loud ringing or howling that occurs when the sound leaving a speaker is picked up by a microphone or other source and is re-amplified, Figure 1. This cycle repeats until the system reaches its maximum volume or something is done to break this loop of sound regeneration. [1] For a system to be stable, it is necessary that at any frequency for which the overall phase shift is an integral multiple of $2 \cdot \pi$ radians the loop gain must be less than one. Any time this criterion is violated the system will begin to oscillate and feedback will occur at the frequencies for which the loop gain exceeds one. [4]

Three different frequency domain plots of sound are presented to aid understanding of what sounds in the frequency domain look like. The first plot is music, Figure 2, the second is feedback, Figure 3, and the third is music and feedback, Figure 4. It can be seen that there is a definite difference in the three different frequency domain plots. The plot of music is fairly dense with not much wide variation in frequency. The plot of feedback has definite peaks with great variation in the signal. The plot of both feedback and music shows a combination of the previous two plots. In this plot there is

still the denser features of the music in addition to the spikes of feedback periodically interspersed.

Radix-4 Fast Fourier Transform

The Radix-4 Fast Fourier Transform (RAD-4 FFT) algorithm was chosen for its speed in computation. This algorithm decomposes the N-point Discrete Fourier Transform (DFT), $X(k) = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi kn}{N}}$, into sets of two and four-point DFT's.[10] Efficiency is increased due to the computation in a Radix-4 butterfly involving fewer complex multiplications than the Radix-2 butterfly.[11]

An important consideration that was taken was the fact that the resolution of the detection phase is dependent on the sample frequency and the number of FFT points used by the equation $\text{Resolution} = \frac{\text{Sample Frequency}}{\text{Number FFT Points}}$. As an example, if the sampling frequency of the file is $44100Hz$ (a standard sampling rate for audio formats), and a 1024 point FFT is used the resolution will be $43.07Hz$, $\text{Resolution} = \frac{44100Hz}{1024} = 43.07Hz$. Unfortunately, the more points used in computing the FFT the computation time is increased thus increasing the program execution time. The fact that resolution is dependent on both the sampling frequency and the number of FFT points used causes a trade off between these characteristics and total processing time.

Time Domain Windowing

Rectangular time-domain windowing was used to provide smoothing between each frame of data and to increase the resolution in the detection phase. If windowing were not implemented, discontinuities would occur between frames that would cause spikes between each frame causing unwanted distortion. Each window is centered around a particular frame and overlapping the previous and what will be the next window, this overlap provides the smoothing between frames. Figure 5 demonstrates this using a five sample window and a three sample frame.

Detection Phase

The detection phase of the software consists of iterating through each point in the FFT and determining which point is the maximum or which points are above the designated threshold. The FFT was performed on each window of data to help in increase the resolution of the detection phase by allowing more points to be evaluated for

each frame of data processed. The two methods implemented are an algorithm to detect the maximum peak in each frame and to detect all the peaks above a threshold in each frame. The threshold will be determined by *X*dB over the average of the FFT. Each point in the FFT above this threshold will be recorded as a point that is determined to be acoustical feedback. When a peak is detected it corresponds to particular frequency by $\text{Frequency} = \text{FFT Point} \cdot \frac{\text{Sample Frequency}}{\text{Number of FFT Points}}$.

Second Order Notch Filters

A second order notch filter will be implemented to remove the specific frequency containing the feedback. The transfer function of the filter is $H(z) = b_0 \frac{1 - 2\cos\varpi_0 z^{-1} + z^{-2}}{1 - 2r\cos\omega_0 z^{-1} + r^2 z^{-2}}$. [12] A typical spectrum of a second order notch filter with the notch centered about f_o and a smooth pass band can be seen in Figure 6. The filter or set of filters change for each frame of data that is processed. With each new frame of data a new filter or set of filters are computed and applied to each sample in the frame. This allows multiple filters to be implemented on each sample in a frame of data and the characteristics of the filter to change as the characteristics of the signal change.

III. The Solution

The Automatic Feedback Detection and Elimination System was developed in software format using C++ on a UNIX based system, Solaris 6.5, using a Sparc 200MHz processor. The software was written in a modular format for ease of programming, debugging, and implementation. A free-ware program, developed by Dr. Joe Picone, to process a headerless data file using time-domain windowing was used as a base to develop the frame work for this program. [13] These files were modified and tested so that they fit the needs of the Automatic Feedback Detection and Elimination System.

The basic algorithm is divided into two major tasks: detection and elimination, Figure 7. The flow of the software is to input the audio file as *F(t)*, perform a FFT, detect acoustic feedback by detecting the peaks above a threshold in the frequency domain, and finally filter the peak frequencies from the signal.

When a headerless data file is processed, time domain windowing is used to divide the file into frames so that each frame of data is processed independently. This is done to allow changes to be made in the filter design for each frame of data as the characteristics of the audio sample change. Each frame of data is computed and windowed to provide smoothing between the frames, as discussed earlier to prevent spikes between each frame. After the each frame of data is processed, it is outputted to a file so to be played at a later time.

The detection phase is divided into two major portions: the FFT and the peak detection phase. Radix-4 Fast Fourier Transform is performed to find the frequency domain characteristics of the data. As discussed earlier, the FFT and the sampling frequency limit the resolution of the peak detection phase. If the sampling frequency is 44100Hz and the number of points in the FFT is 1024, the resolution is only 43.07Hz. The peak detection phase and filtering phase is limited to 43.07Hz intervals. This causes less discrimination between what is feedback and what is the wanted signal. If feedback was to occur at 35Hz, it would be detected in the range of 0Hz to 43Hz and the whole frequency range would have to be eliminated to remove the feedback at 35Hz. This would cause all of the signal in that range to be eliminated thus severely distorting the wanted signal.

At higher resolutions, the algorithm can more accurately detect remove and remove the portion of signal that contains only feedback without distorting the remaining signal. Unfortunately, to provide a higher resolution a FFT with more points must be used. A FFT containing a larger number of points requires longer computation time thus increasing program run-time. This is something that must be considered when deciding what resolution the user wants the algorithm to perform.

Once the peaks in the frequency domain have been detected, the algorithm moves to the elimination stage. The elimination stage consists of applying a second order notch filter to each sample in the frame of data. This filter is applied in two parts: computing the coefficients and applying the filter. The coefficients for each frame of data are computed for each peak that was detected. These coefficients are stored and used when applying the filter to each sample. With the coefficients computed for each peak, the second order notch filter can be applied. The filter or filters are applied to each sample in the frame of data. This allows a single filter or multiple filters to be applied in the most efficient manner.

After a frame of data is processed, it is written to an output file. The algorithm is repeated for each frame of data in the file. The last frame and window of data are zero padded if the data does not make a complete frame. After processing is done, the output file will contain a version of the original signal with the acoustic feedback remove along with a small portion of silence at the very end of the file due to the zero padding of the last frame.

IV. Testing

The testing of the software was divided into two phases: module testing and spiral testing. During the module testing phase, each part of the software was tested to ensure that it was functioning properly.

Input and Output of a Headerless Data File

The first section to be tested was the input and output of a headerless data file using time domain windowing. The main process in testing this section was to process a headerless data file using the software and compare the input file and the output file to determine if the file was processed with no errors.

The program does zero pad the output to a length of the full size of the last frame. This causes the files to be exactly the same until the very end of the file, where the output file contains a portion of data that is "zero". This can be shown to be consistent and correct by processing an output again using it as the input. The new version of the output file will be identical to the input (the first output) file. This is due to the fact that the file was of a length that produced a number of frames that would completely encompass the data with no new empty spaces that would be zero padded.

Radix-4 Fast Fourier Transform

A program to perform a Radix-4 Fast Fourier Transform was incorporated into the design of the Automatic Feedback Detection and Elimination System. This piece of software was free-ware optimized by The Institute for Signal and Information Processing (ISIP). [14] The bulk of this testing was formatting the data to be inputted to the FFT into type *double* from a type *float* and formatting the output of the FFT from real and imaginary parts to a magnitude of the square root of the sum of the squares: $mag = \sqrt{\mathrm{Re}^2 + \mathrm{Im}^2}$. This section was tested by processing known synthetic data, know sine waves, and plotting the results of the FFT to ensure proper operation.

Peak Detection

The peak detection function processed the output of the FFT to determine the maximum peak or peaks above a threshold set by the user. This was tested with the use of known synthetic data and on real values produced by the FFT. The first set of synthetic data consisted of a simple array of numbers to determine if the function would correctly identify the maximum number in the array. The second set of synthetic data consisted of audio files of known sine waves. This allowed testing on a real audio sample at different frequencies and sampling rates. This allowed testing that first determined that a single peak could be detected along with a number of peaks above a threshold.

Second Order Notch Filter

The second order notch filter was implemented in two parts: computing the coefficients and applying the filter. This allowed for separate testing and implementation of each part. Again, both parts of the filter were tested using sets of synthetic and real data sets. The use of separate functions for computing the coefficients and applying the filter allow the filter to be applied to each sample in a frame while only calculating the coefficients once for each frame of data. Unfortunately, the second order notch filter does not provide a flat enough pass band to prevent significant distortion of the spectrum, Figure 6.

System Optimization

The spiral testing occurred in the system optimization phase of the testing. In this section of testing the optimal values for threshold, filter bandwidth, frame duration, window duration were determined. Bugs in the program as a whole were identified and removed.

V. Results

The final result of this project is a piece of software, Automatic Feedback Detection and Elimination System, and a graphical user interface that operates on a UNIX based system, Solaris 6.5, using a Sparc 200MHz processor that detects and eliminates feedback from a headerless data file. The program can be operated through the command line or via the graphical user interface. The user can process a headerless data file while controlling the sample frequency, the number of FFT points, the window duration, and the frame duration. The program will detect the maximum peaks and filter them using a second order notch filter.

Several files were used in the final testing phase of the software. A 1500Hz sine wave, a sample of music, and a sample of music with acoustical feedback present. Each of these samples were process and the resulting samples were evaluated to determine the effect of the Automatic Feedback Detection and Elimination System software. To evaluate these samples the frequency domain of each was plotted as shown in Figure 8 through Figure 13. In Figure 8, a 1500Hz sine wave was plotted, and the processed version is shown in Figure 9. It can be seen that by processing the sample the peak at 1500Hz was removed. Unfortunately, the filter also amplified the noise in the sine wave. The second file processed was a sample of John Coltrane's Giant Steps, 1960. Figure 10 and Figure 11 show plots of the sample before and after being processed. The first one show the majority of the signal in lower end of the spectrum and a portion of high frequency noise. The processed version show a portion of the peck in the lower end of the spectrum filtered and the majority of the noise distorted throughout the spectrum. The final two plots, Figure 12 and Figure 13, show the frequency spectrums of Giant

Steps with feedback. In Figure 12, the feedback can be seen around 2500Hz and again at 21000Hz along with the music and noise similar to Figure 10. The final plot, Figure 13, shows a significant portion of the feedback removed in the region of 2500Hz.

These three cases help to show how the Automatic Feedback Detection and Elimination System effects on different types of signals. It is evident that the filter introduces significant distortion to the signal. This is caused by the gain of the filter not being 0dB across the pass band. This problem could be solved by using a better filtering algorithm. The filter is effective enough to successfully remove a portion of the feedback from a signal. In the grand spectrum, the system was implemented as planned. As most research projects continue through their life cycle problems are found. The most significant problem encountered was the second order notch filter. Fortunately, the current implementation showed that it was possible to detect and eliminate feedback using the planned system.

VI. Future Work

Future work could better optimization of the system, providing multiple versions of the software to perform on different computer platforms, real-time operation, and firmware realization. Further optimization of the software could be implemented to provide better detection and filtering along with faster program run-time. A more advanced detection algorithm could be implemented. A possibility for this could be considering not only the peaks in frequency but also consider the deviation between peaks. Advanced filtering techniques could be applied to provide more accurate filters. Higher order notch filters could provide better correlation between center frequencies, attenuation, roll-off, bandwidth, stop-band ripple, pass-band ripple, and a flatter pass band.

Continued development of this software could provide multiple versions of the program to operate on different platforms. It would be possible to provide versions to run on the different types of processors (Sparc vs. Intel Pentium) in the UNIX environment along with version to operate on Windows95, WindowsNT, and Macintosh environments.

Further development could be done to provide real-time operation in these different environments. Proper inputs and outputs must be developed, which would allow the software to operate using the computer's microphones and speakers instead of processing a file. It would also be necessary to optimize the algorithms to provide faster operation time, which would minimize the system delay.

The final problem to be conquered would be to implement this system in firmware as a product that would be usable independent of a computer. A system such as this could be used in any type of sound system as an independent unit or possibly integrated into an equalizer or a mixing board.

Appendix A

References

[1] Doran Oster, "The Story of Feedback." Sabine® Adaptive Audio, 1997. http://www.sabineinc.com/frames/home.html 1998.

[2] Bernard A. Hutchins, Jr., Walter H. Ju, "An Adapting Delay comb Filter for the Restoration of Audio Signals Badly corrupted with a Periodic Signal of Slowly Changing Frequency." Journal of the Audio Engineering Society, vol. 30, No. 1/2, 1982 January/February.

[3] L. N. Mishin, "A Method for Increasing the Stability of Sound Amplification Systems." Scientific-Research Institute of medical Instrumentation and Equipment, Moscow, 1957.

[4] Eugene T. Patronis, Jr., "Electronic Detection of Acoustic Feedback and Automatic Sound System Gain Control." Journal of the Audio Engineering Society, vol. 25, 1992.

[5] Michael P. Lewis, Timothy J. Tucker, Doran M. Oster, *Patent 5245665 : Method and Apparatus for Adaptive Audio Resonant Frequency Filtering.* Sabine Musical Manufacturing Company, Inc. September. 14, 1993.

[6] Roland Corporation U.S., http://www.rolandus.com/index.html 1998.

[7] Peavey Electronics Corporation, http://www.peavey.com/ 1998.

[8] Sabine, Inc., http://www.sabineinc.com/frames/home.html 1998.

[9] M. P. Lewis, T. J. Tucker, D. M. Oster, *Method and apparatus for adaptive audio resonant frequency filtering.* Patent Number: 5245665, Sabine Musical Manufacturing Company, Inc. June 12, 1991, http://www.patents.ibm.com/details?patent_number=5245665 1998.

[10] J. Picone, A. Ganapathiraju, M. Balducci, and J. Hamaker, *Benchmarking of FFT Algorithms*, Department of Electrical and Computer Engineering, Mississippi State University, Mississippi.

[11] C.S. Burrus and T.W. Parks, *DFT/FFT and convolution Algorithms: Theory and Implementation*, John Wiley and Sons, New York, NY, USA, 1985.

[12] John G. Proakis, Dimitris G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications.* Third Edition, Prentice-Hall, Inc. 1996.

[13] J. Picone, *Computer Assignment 1*. Department of Electrical and Computer Engineering, Mississippi State University, Mississippi. http://WWW.ISIP.MsState.Edu/resources/courses/ece_4773/tools/1994/solution_01/ 1998.

[14] A. Ganapathiraju, *Benchmarking of FFT Algorithms*. Department of Electrical and Computer Engineering, Mississippi State University, Mississippi. http://WWW.ISIP.MsState.Edu/resources/technology/software/1997/parallel_dsp/ 1998.

Appendix B

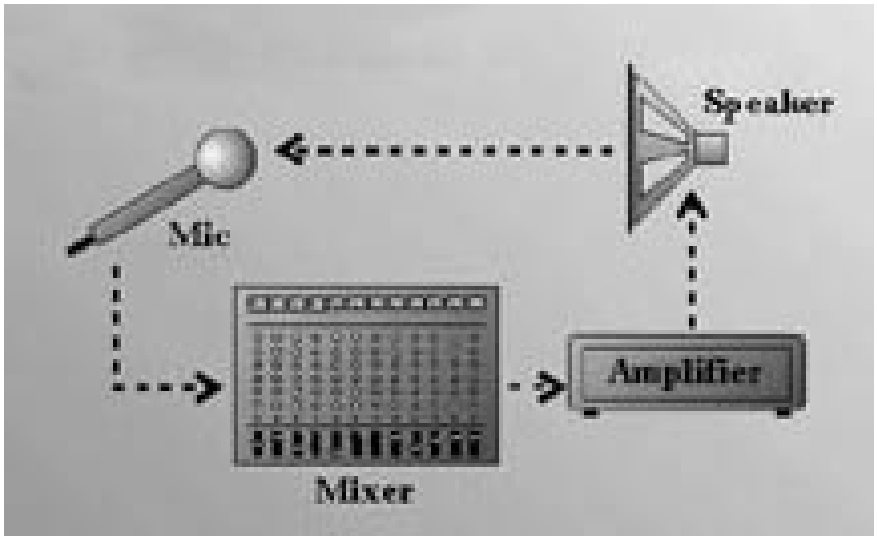

Figure 1: Feedback Path.

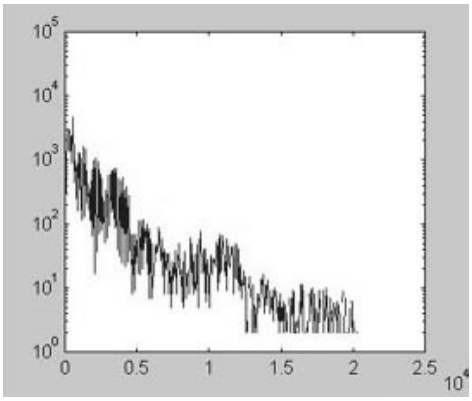

Figure 2: John Coltrane's Giant Steps

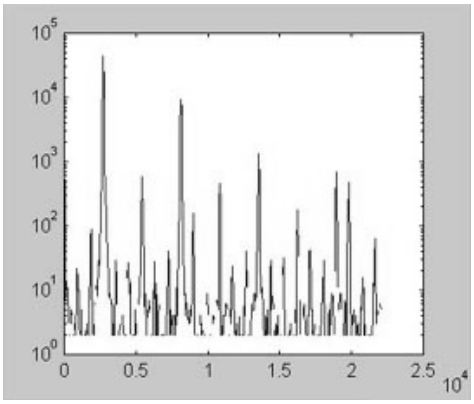

Figure 3: Feedback

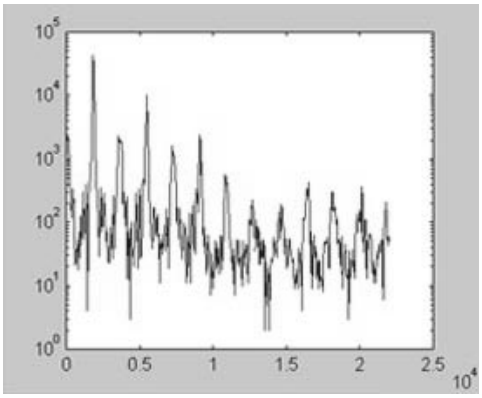

Figure 4: Giant Steps & Feedback

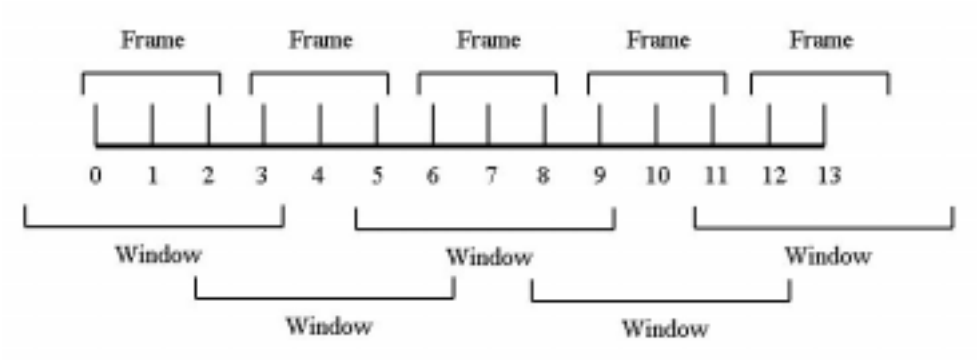

Figure 5: Time Domain Windowing

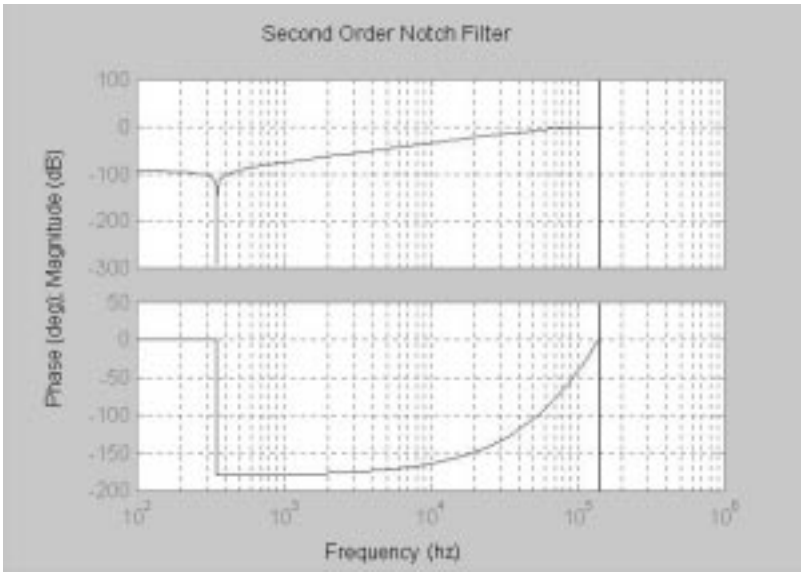

Figure 6: Second Order Notch Filter

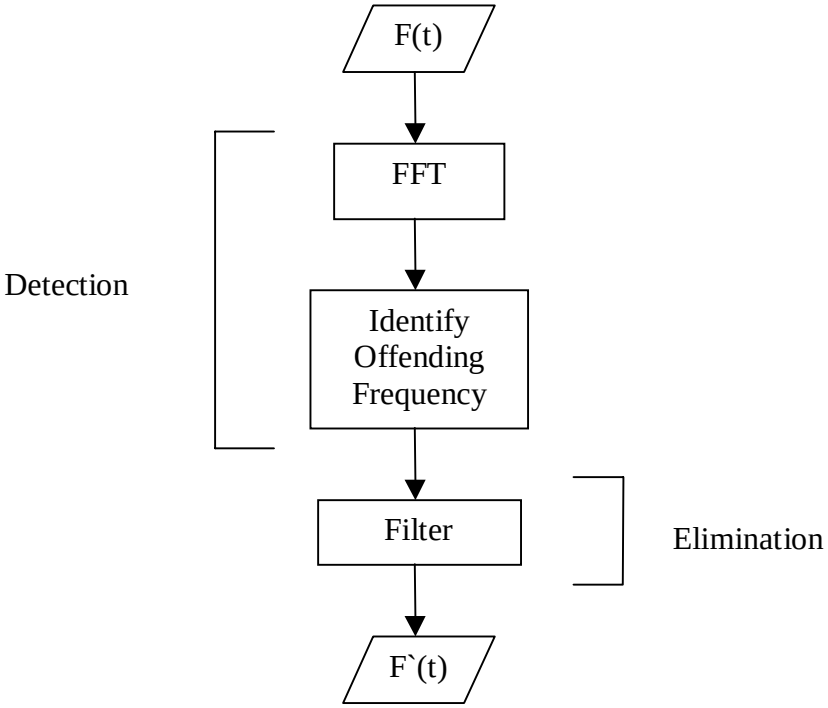

Figure 7: Software Flow Chart.

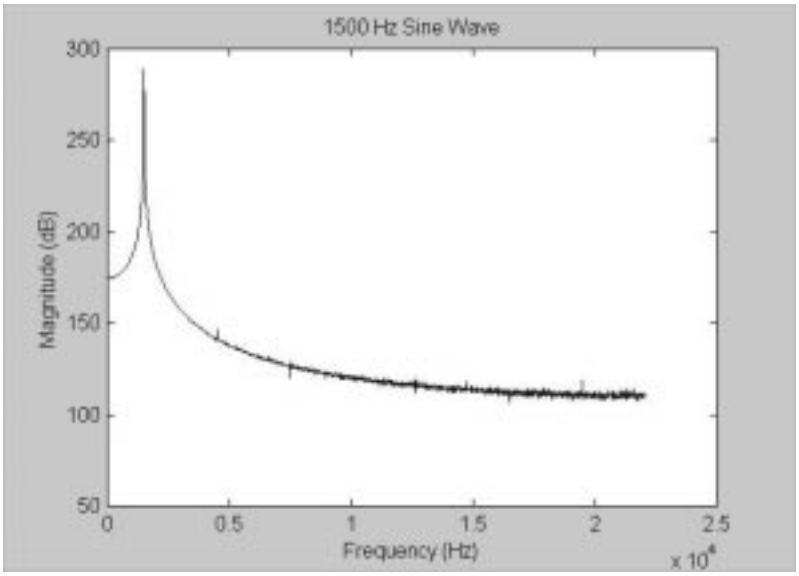

Figure 8: 1500Hz Sine Wave

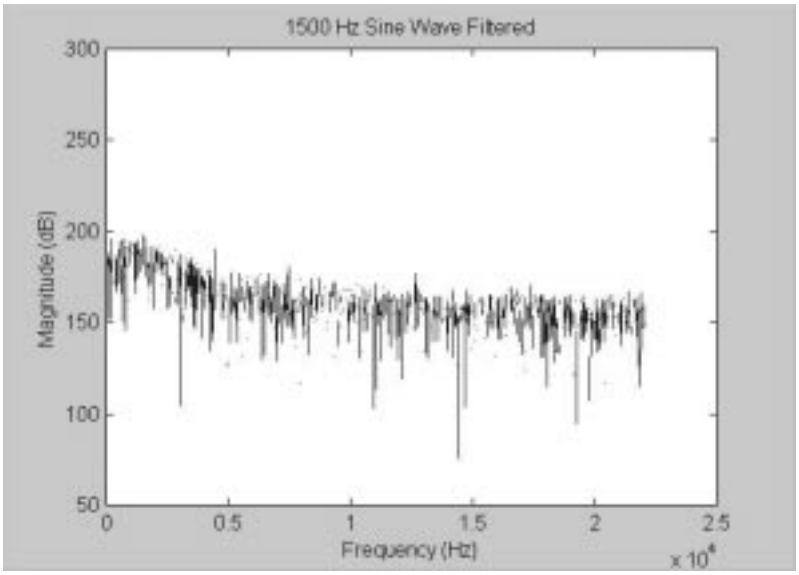

Figure 9: 1500Hz Sine Wave Filtered

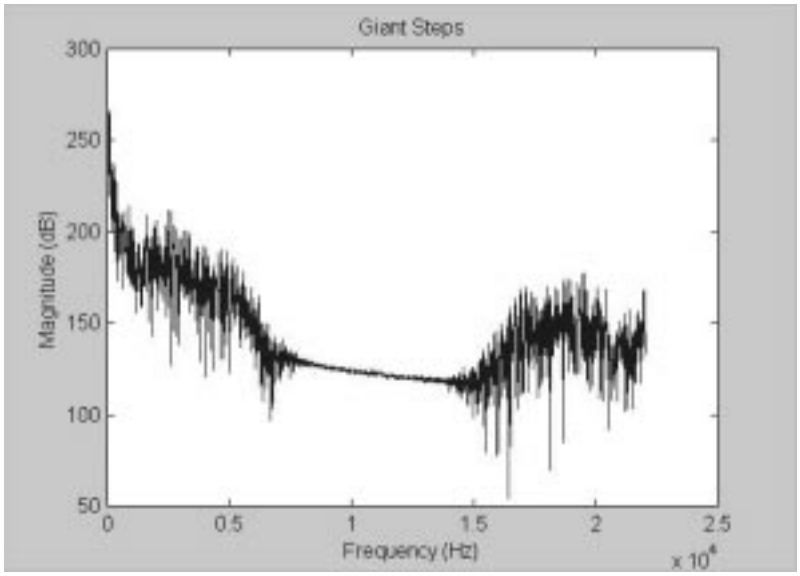

Figure 10: Giant Steps

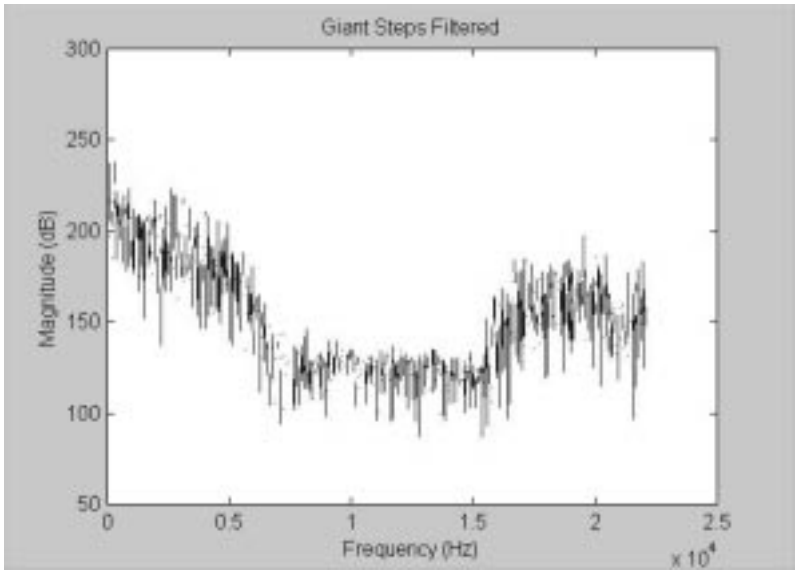

Figure 11: Giant Steps Filtered

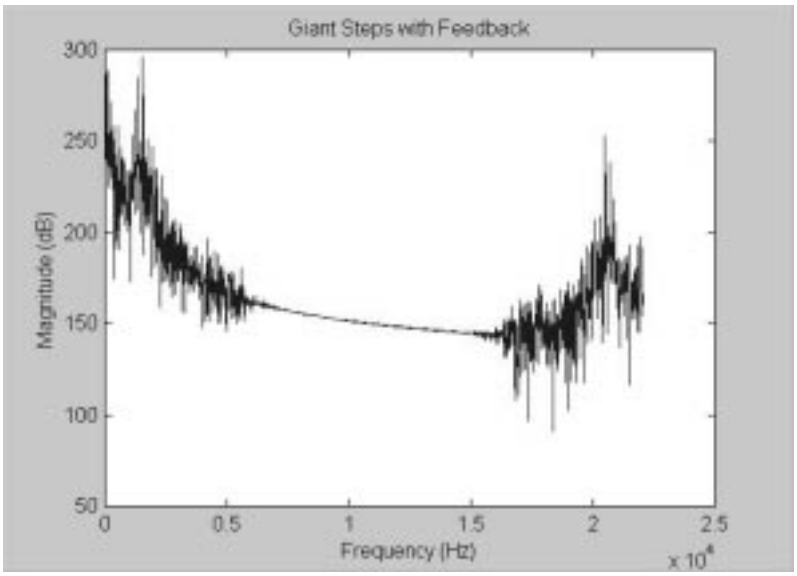

Figure 12: Giant Steps with feedback

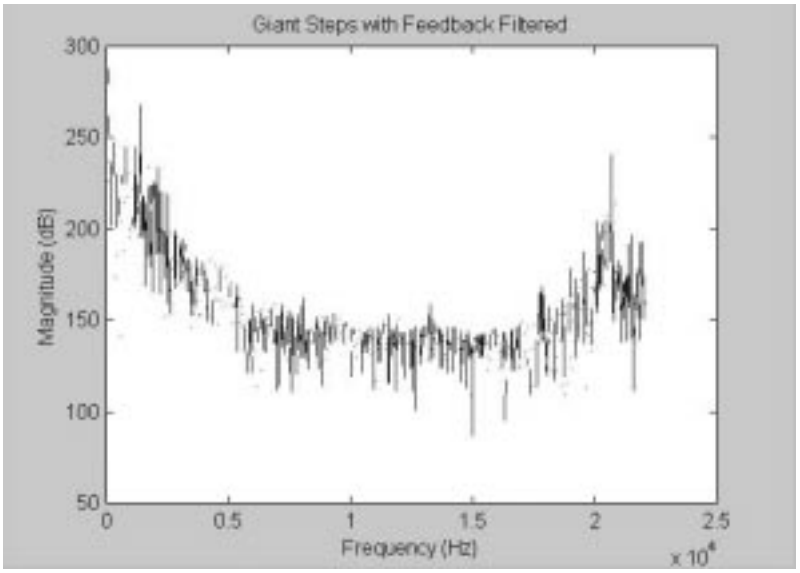

Figure 13: Giant Steps with Feedback Filtered

Appendix C

TCL/TK Code

Automatic Feedback Detection and Elimination System

C++ Code