An Algorithm for Extracting the Relative Phase of Harmonics from a Periodic Digital Signal

Joseph A. Yasi*
Department of Physics, University of Illinois at Urbana-Champaign

(Dated: August 7, 2004)

An algorithm is presented for the extraction of the relative phase of the harmonics of a digital audio signal. This extraction is possible through the use of digital filters and nonlinear fits over small time windows. Comparisons of reconstructed phase information with known input functions demonstrates an accuracy of $\pm 0.05^\circ$ for stable audio signals such as a square wave or a sawtooth. The relative phase of the harmonics can then be used to reassemble the waveform, and vary the phase to study the effects of phase on tone quality. This method is also generic enough to be applied to any arbitrary periodic digital signal as long as it is stable enough and the harmonics are not too close to each other in frequency. Possible applications are discussed.

I. BACKGROUND AND INTRODUCTION

The ear's sensitivity to phase information in sounds has been disputed for years. Helmholtz states

> The quality of the musical portion of a compound tone depends solely on the number and relative strength of its partial simple tones, and in no respect on their differences of phase.[1]

For years, this has been taken as fact, and phase information was ignored as inaudible when looking at musical instruments. However, in an appendix to the same book, R. Koenig disagrees, citing his experiments with the wave siren,

> Although the quality of tone principally depends on the number and relative intensity of the harmonic tones, the influence of difference of tone is not by any means so insignificant as to be entirely negligible. We may say, in general terms, that the differences in the number and relative intensity of the harmonic tones compounded produces those differences in the quality of tone which are remarked in musical instruments of different families. But the alteration of phase between these harmonic tones can excite at least such differences of quality of tone as observed in musical instruments of the same family, or in different voices singing the same vowel.[1]

Therefore, there is some evidence from early work that the ear can hear differences in phase between the harmonics.

Recent work on binaural location of audio sources has also provided evidence that the ear is especially sensitive to phase information at frequencies lower than 1500 Hz.[2] The ears encode the phase of the signals at each ear and use this data to measure the interaural time difference at these frequencies to locate sound sources. Without phase information, humans would not be able to locate sound below 500 Hz because the wavelength of sound is much larger than the diameter of the head. Since the wavelength is so long, the head does not show the other ear from a sound source, and the intensity difference between the ears cannot be used for location as it is with higher frequencies. Since the ear uses phase information for location, the brain receives this information, and there is a possibility that it affects the brains perception of tone as well due to the nonlinearities of the ear.[3] The location of sound by phase information was also studied at UIUC, confirming the use of phase for location of low frequency sounds.[4]

The Fast Fourier Transform(FFT) takes a temporal signal and transforms it into the frequency domain, yielding amplitude and phase at each frequency. Since musical signals are nonstationary, the Short Term Fourier Transform(STFT) must be used, since the FFT assumes a stationary signal over the time window. Any modulation over the time window causes phase and amplitude distortion.[5] The accuracy of the phase information decreases, and the widths of the sidebands increase as the size of the time window decreases. Since musical signals require small windows due to their transient components, the FFT is not accurate enough for getting the phase information of the harmonics.

In this paper, the author discusses an algorithm for the extraction of the phase information of the harmonics from an audio signal through the use of digital filters. Since digital filters also cause distortion of the signal, the algorithm uses a Finite Impulse Response(FIR) filter with linear phase distortion over the bandpass region. This phase information can then be used to study the phase relation between the harmonics of different instruments.

*Electronic address: `yasij@rpi.edu`; Also At: Department of Physics, Rensselaer Polytechnic Institute, Troy, NY

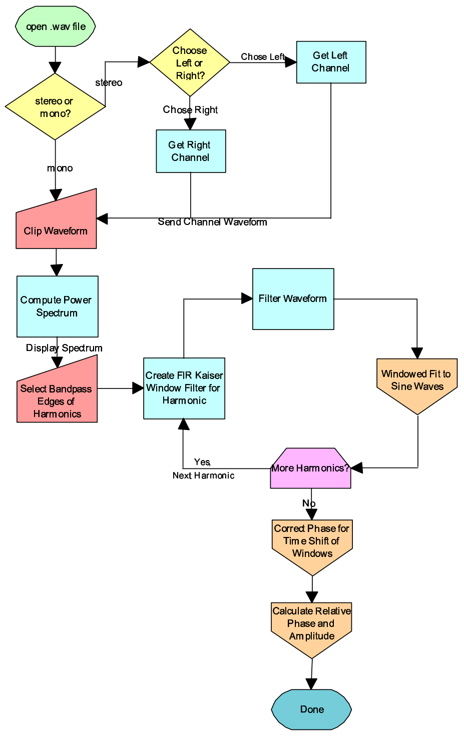


FIG. 1: Basic structure of the algorithm. The digital filter is run on the entire waveform, instead of each window to minimize the effects of distortion at the edges of the time window. A FIR filter is used because it has linear phase response.

II. TECHNICAL DESCRIPTION

The algorithm was developed in Mathworks MATLAB[9]v6.5.1 Release 13 on Microsoft Windows XP Professional, using the Signal Processing, Control System, and Optimization Toolboxes. Also, the Time-Frequency Toolbox[10]is great for generating 3D spectrograms of the signal. This tool is inherently good for signal analysis because of it's native use of matrices and extensive signal processing functions.

To find the phase evolution of the harmonics of the waveform over time, approximate frequency bands of the harmonics must be known. To approximate the frequencies of the harmonics, the algorithm calculates the power spectrum of the waveform using Welch's method.[6] The user then must choose acceptable bands for the harmonics based on this power spectrum to construct digital bandpass filters for each harmonic. Since the main interest is the phase of the harmonics, a filter with linear phase response over the bandpass region is ideal. A Kaiser Window based Finite Impulse Response(FIR) filter has linear phase response over the entire spectrum

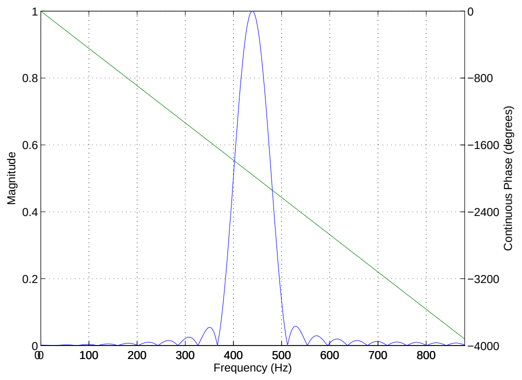


FIG. 2: Frequency and Phase response of a typical Kaiser Window based FIR filter created by the algorithm. This is a band pass filter around 440Hz with ripple. Notice the phase response is linear, and the passband is narrow with relatively low ripple.

(Fig. 2), and it performs well over a narrow bandpass, making it the filter of choice.

Using each filter, the harmonics are separated from the waveform into their own arrays. To find how phase, frequency and amplitude evolve over time, each harmonic is divided into 1024-sample rectangular windows with 50% overlap(See Fig. 3). Amplitude, frequency and phase of each window is found using a nonlinear least-squares fit to a sine function.

$$A_{fit} \sin\left(2\pi f_{fit} + \phi_{fit}\right) \tag{1}$$

Where A_{fit} is the fit amplitude, f_{fit} is the fit frequency, and ϕ_{fit} is the fit phase. The built-in MATLAB `'fminsearch'` function is used to find the best fit using a simplex direct search method.[7] The fit phase and amplitude are then corrected for the phase and amplitude distortion of the filter from the filter response curve by dividing by the amplitude distortion coefficient at the fit frequency, and subtracting the phase distortion phase shift. Phase is normalized to between $-\pi$ and π, and amplitude is normalized to be positive.

After the fits, the phase from the later time windows must be corrected to the proper reference phase from the

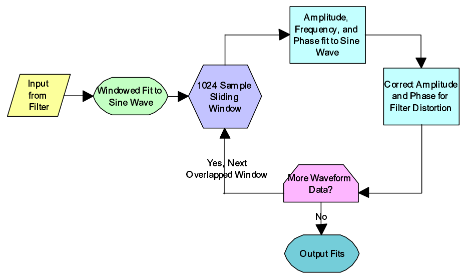
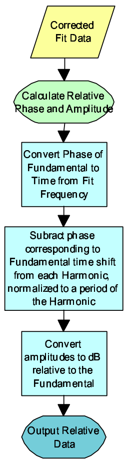

Input from Filter
Windowed Fit to Sine Wave
1024 Sample Sliding Window
Amplitude, Frequency, and Phase fit to Sine Wave
Correct Amplitude and Phase for Filter Distortion
More Waveform Data?
Yes, Next Overlapped Window
No
Output Fits

FIG. 3: Frequency, Phase and Amplitude Fit algorithm.

[9] Available at: http://www.mathworks.com
[10] Available at: http://crttsn.univ-nantes.fr/˜auger/tftb.html



Corrected Fit Data
Calculate Relative Phase and Amplitude
Convert Phase of Fundamental to Time from Fit Frequency
Subract phase corresponding to Fundamental time shift from each Harmonic, normalized to a period of the Harmonic
Convert amplitudes to dB relative to the Fundamental
Output Relative Data

FIG. 4: Relative Phase Calculation algorithm. Relative phase must be calculated in the time domain because equal time delays result in different phase shifts at different frequencies.

beginning of the clip.

$$\phi_k^h\left(f_{adj}^h\right) = \phi_{fit}^h + 2\pi\frac{f_{adj}^h}{F_s}kS_wS_o \tag{2}$$

Where ϕ_k^h is the corrected phase of the harmonic, f_{adj}^h is the adjusted mean frequency of the harmonic, F_s is the sampling rate, k is the time window index, S_w is the number of samples per window, and S_o is the overlap ratio. f_{adj} is calculated by finding a local minimum of the standard deviation of phase over the whole sample using (2), with a start guess at the mean fit frequency for the harmonic. A linear trend is introduced if the frequency is off slightly from the true mean frequency, so this small correction helps adjust for fit artifacts. This constant adjustment is only relevant for the absolute phase of each harmonic. The constant time adjustment correction for each sample cancels itself out when calculating relative phase between the harmonics, but the small adjustment to the mean frequency is useful for the relative phase time calculations.

In order to compare the phase of the harmonics between different sound clips, phase must have a common reference between the different clips. This algorithm calculates the relative phase of the harmonics to the fundamental (See Fig. 4). A time shift of the entire signal results in different phase shifts at different frequencies because a constant time shift is a larger percentage of the period for higher frequencies. Therefore, the phase of each harmonic must be converted to time to calculate relative phase. Then, the time shift of the fundamental can be subtracted from the time shift of each harmonic and converted back into the phase of each harmonic, by

$$\psi_k^h = \phi_k^h - \left(\phi_k^{fund}\frac{f_{adj}^h}{f_{adj}^{fund}} \mod \frac{\phi_k^h}{2\pi f_{adj}^h}\right) \tag{3}$$

Where ψ_k^h is the relative phase of the hth harmonic. The time shift of the fundamental is normalized to a period of the harmonic. The relative phase information is now normalized to a fundamental with zero phase, is easily comparable across different sound clips and intuitively gives the viewer of the information an understanding of what is happening with the phase of the harmonics.

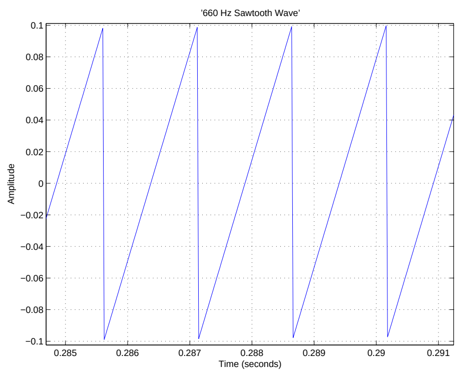


FIG. 5: 660 Hz sawtooth wave.

To record audio data to test the algorithm, an HP-33120A function generator was connected to a LabPC+ DAQ card with scalable gain connected to a PC through GPIB. Also, musical instruments were recorded using a Peavey PVM45 Microphone connected to a Marantz PMD670 digital recorder. The program was tested on a 2.8 GHz Intel Pentium-4 with 1024 MiB of RAM and a 1.6 GHz Intel Pentium-M with 512 MiB of RAM.

III. RESULTS AND DISCUSSION

Now that the algorithm is giving relative phase data, it needs to be tested for accuracy. Several test waveforms are used to verify this accuracy. Fig. 5 shows a clip of a 660 Hz sawtooth wave. The sawtooth wave is defined by,

$$f(t) = \left\{ \begin{array}{cc} 132t, & -\frac{1}{1320} < t < \frac{1}{1320} \\ f(t+\frac{1}{660}), & \forall t \end{array} \right. \tag{4}$$

Where t is in seconds. By a simple Fourier Series, the sawtooth wave has Fourier coefficients,

$$a_k = 660\int_{-\frac{1}{1320}}^{\frac{1}{1320}} f(t)\cos\left(2\pi 660kt\right)\, dt = 0 \tag{5}$$

$$b_k = 660\int_{-\frac{1}{1320}}^{\frac{1}{1320}} f(t)\sin\left(2\pi 660kt\right)\, dt = \frac{(-1)^{k+1}}{10\pi k} \tag{6}$$

These coefficients yield,

$$A_k = \sqrt{a_k^2 + b_k^2} = \frac{1}{10\pi k} \tag{7}$$

$$\phi_k = \cot^{-1}\left(\frac{b_k}{a_k}\right) = \left\{ \begin{array}{ll} 0, & k\ odd \\ \pi, & k\ even \end{array} \right. \tag{8}$$

Where A_k is the amplitude of the $(k-1)th$ harmonic, and ϕ_k is the sin function phase of the $(k-1)th$ harmonic (The fundamental $(k=1)$ is the $0th$ harmonic, and $k=2$ represents the $1st$ harmonic).

Harmonic	ϕ_r	σ_ϕ	f_{fit}	σ_f
Fund	0.00°	0.00	660.0 Hz	1.77×10^{-3}
1st	180°	9.66×10^{-3}	1320 Hz	3.80×10^{-3}
2nd	0.00554°	2.58×10^{-2}	1980 Hz	5.27×10^{-3}
3rd	180°	1.65×10^{-2}	2640 Hz	6.97×10^{-3}
4th	0.000334°	1.97×10^{-2}	3300 Hz	8.51×10^{-3}
5th	180°	2.57×10^{-2}	3960 Hz	1.01×10^{-2}

TABLE I: Relative phase fit data for 660 Hz sawtooth wave. σ_ϕ is the standard deviation of the phase. ϕ_r is relative phase, and it is defined relative to the fundamental, giving the fundamental, by definition, 0 phase. f_{fit} is the mean fit frequency found by the algorithm. σ_f is the standard deviation of the frequency, showing an error of ±.01 Hz.

Fig. 6 shows the output of the algorithm for the 660 Hz sawtooth wave. The algorithm agrees very strongly with the mathematical result, with error only ±0.026°, as shown in Table I. This result is very promising, and shows that the algorithm can extract phase information from stable signals with great precision.

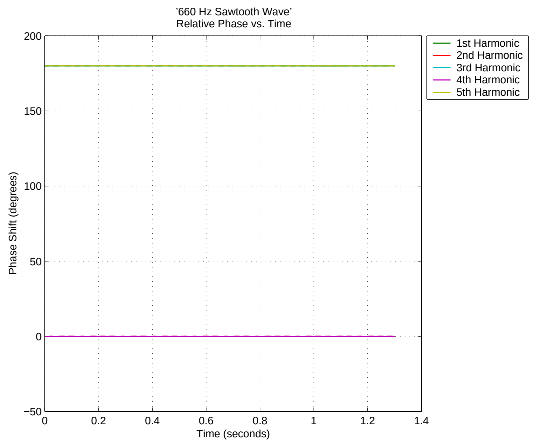
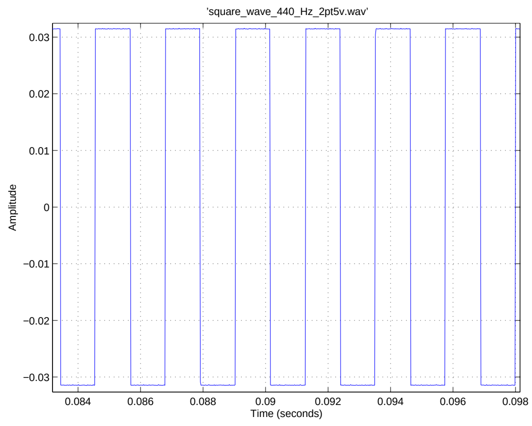


FIG. 6: Relative phase of 660 Hz sawtooth wave.

To further test the algorithm, a 440 Hz square wave from an HP-33120A function generator was recorded using a LabPC+ DAQ card. Fig. 7 shows the recorded waveform. A 440 Hz square wave is defined by,

$$g(t) = \begin{cases} -\frac{1}{10}, & -\frac{1}{880} < t < 0 \\ \frac{1}{10}, & 0 < t < \frac{1}{880} \\ g(t+\frac{1}{440}), & \forall t \end{cases} \tag{9}$$



h

FIG. 7: 440 Hz Square Wave recorded with a LabPC+ DAQ card.

Harmonic	ϕ_r	σ_ϕ	f_{fit}	σ_f
Fund	0.00°	0.00	446.3 Hz	1.37×10^{-3}
1st	−0.0240°	1.59×10^{-2}	1339 Hz	4.11×10^{-2}
2nd	−0.0276°	2.14×10^{-2}	2231 Hz	6.83×10^{-2}
3rd	−0.0479°	3.39×10^{-2}	3124 Hz	8.32×10^{-2}
4th	−0.0604°	4.14×10^{-2}	4016 Hz	9.23×10^{-2}
5th	−0.0622°	4.86×10^{-2}	4909 Hz	1.31×10^{-1}

TABLE II: Relative phase fit data for 440 Hz recorded square wave. σ_ϕ is the standard deviation of the phase. The harmonics are the nth odd harmonics of the square wave because a square wave does not contain even harmonics. ϕ_r is relative phase, and it is defined relative to the fundamental, giving the fundamental, by definition, 0 phase. f_{fit} is the mean fit frequency found by the algorithm. σ_f is the standard deviation of the frequency, showing an error of ±.14 Hz.

Where t is in seconds. With Fourier coefficients,

$$a_k = 440 \int_{-\frac{1}{880}}^{\frac{1}{880}} g(t)\cos(2\pi 440 kt)\, dt = 0 \tag{10}$$

$$b_k = 440 \int_{-\frac{1}{880}}^{\frac{1}{880}} g(t)\sin(2\pi 440 kt)\, dt = \frac{1-(-1)^k}{10\pi k} \tag{11}$$

$$= \begin{cases} \frac{1}{5\pi k}, & k\ odd \\ 0, & k\ even \end{cases} \tag{12}$$

These coefficients yield,

$$A_k = \sqrt{a_k^2 + b_k^2} \tag{13}$$

$$= \begin{cases} \frac{1}{5\pi k}, & k\ odd \\ 0, & k\ even \end{cases} \tag{14}$$

$$\phi_k = \cot^{-1}\left(\frac{b_k}{a_k}\right) = 0 \tag{15}$$

Where A_k is the amplitude of the $(k-1)th$ harmonic, and ϕ_k is the sin function phase of the $(k-1)th$ harmonic.

The fit frequencies and phases for the fundamental and the first 5 odd harmonics are shown in Table II. The fit frequencies are all high, but that is likely due to systematic error, since the deviation of the frequency calculated by the program is very small, and the harmonics all lie on the proper multiples of 446 Hz. The relative phase fits are all within $0.07°$ of $0°$, the mathematical result for a square wave. No even harmonics were present in the waveform.

One of the key advantages of this algorithm is that it can be run on any arbitrary quasi-periodic digital signal, and it runs at a reasonable speed. It only takes a couple of minutes on a 1.6 GHz Intel Pentium-M with 512 MiB of RAM (a relatively standard modern laptop) to analyze the first six harmonics of a 1.5 second clip at 44.1 kHz. The results are also much more accurate than just taking the FFT of the signal. It pinpoints the frequency and phase of each harmonic, showing how they evolve over time. The main improvement that this work gives over previous works on the study of musical signals is the extraction of the phase information. Most prior work has only looked at the relative amplitude between the harmonics, not the phase.

Since this algorithm works with any arbitrary digital signal, it's applicability is not confined to musical signals. It could also be used to analyze harmonic data from the tip vibration in Atomic Force Microscopy.[8] The phase information could then be used to study the harmonic distortion caused by the material. It also has a possible application in voice recognition, if it is modified to better track transient signals by looking at the harmonics at each time window. The phase information could prove to be a valuable voice fingerprint. Many other physical phenomena involve periodic harmonics, giving this algorithm limitless applications.

The algorithm is limited to rather stable signals. If there are two notes near each other in frequency, it has a hard time splitting them apart because the filters cannot isolate the two frequencies. For example, when the program was run on a Tibetan Bowl that has two fundamentals that beat against each other, it would flip between the two frequencies, and could not track them individually. The phase data is not accurate if the frequency varies too much over the time interval because the time subtraction when calculating relative phase is based off the mean fit frequency. It is fine for steady notes if the transient effects of the attack and decay are clipped off the signal.

IV. CONCLUSIONS

An algorithm has been developed to extract the relative phase of the harmonics of musical instruments using digital filters and a least-squares fitting algorithm. Phase and frequency evolution over time information is reliably extracted from a digital signal if the digital signal is relatively stable, and the harmonics are sufficiently far apart. On stable mathematical signals, phase is found to $\pm 0.05°$, which is much more accurate than needed for studying musical instruments because the ear is not sensitive enough to detect phase differences that small, nor is phase that stable on musical instruments.

With the assertion of the accuracy of the algorithm, new opportunities are opened to study musical instruments. The phase change of the harmonics for different notes on the same instrument can be compared to look at resonance shifts. This can then be compared against frequency response resonance data for the instrument. Also, the algorithm can be further improved to track changes in the harmonics over time, yielding an improvement for varying signals.

The algorithm can also be used to quantify the effects of phase on hearing and tone quality. The harmonic information can be used to synthesize the waveform from the harmonics. Relative phase between different harmonics can be varied to determine impact on a listener's perception of tone quality. The effect of phase on tone quality can then be used to improve sound synthesis and give insights into physical synthesis by showing the phase response of the different notes.

Acknowledgments

The author is grateful to Prof. Steve Errede for advising him on this project. Many thanks to Jack Boparai for running the lab and providing the necessary software and equipment for this project. The author would also like to acknowledge Toni Pitts and Sarim Murtaza for their hard work running the REU program. Thank you to Ma'ayan Bresler for recording Oboe, and to Laura Book for recording Viola to analyze with the program. The author is also greatful to the National Science Foundation for supporting and funding this REU program. This material is based upon work supported in part by the National Science Foundation under Grant No. PHY-0243675. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[1] H. L. F. Helmholtz, *On the Sensations of Tone* (Longmans, London, 1885; reprint Dover, 1954), pp. 126,537, 2nd ed., transl. A. J. Ellis.

[2] W. M. Hartmann, Physics Today **52** (1999), URL `http://www.aip.org/pt/vol-52/iss-11/locsound.html`.

[3] S. P. Lipshitz, M. Pocock, and J. Vanderkooy, J. Audio Eng. Soc. **30**, 580 (1982).

[4] M. Gilson (2000), 398EMI Class at UIUC, URL `http://wug.physics.uiuc.edu/courses/phys398emi/398emi_student_projects_fall00.html`.

[5] P. Masri and P. A. Bateman, in *Colloquium on "Audio Engineering"* (Institude of Electrical Engineers, London, 1995), 1995/089.

[6] P. D. Welch, IEEE Trans. Audio Electroacoustics **AU-15**, 70 (1967), from MATLAB instruction guide.

[7] J. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, SIAM Journal of Optimization **9**, 112 (1998), from MATLAB instruction guide.

[8] U. Dürig, New J. of Phys. **2**, 5.1 (2000).

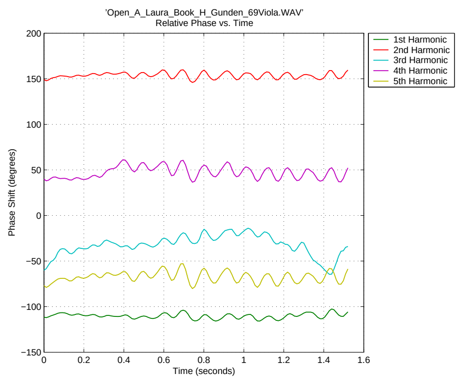


FIG. 8: Relative Phase of the harmonics of an Open A on a 1969 H. Gunden Viola played by Laura Book.

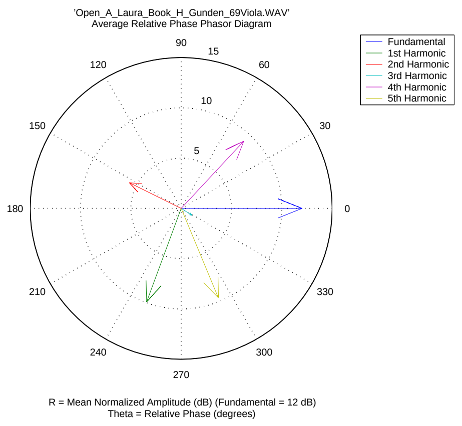


FIG. 9: Phasor plot of the harmonics of an Open A on a 1969 H. Gunden Viola played by Laura Book.

APPENDIX: MUSICAL INSTRUMENT DATA

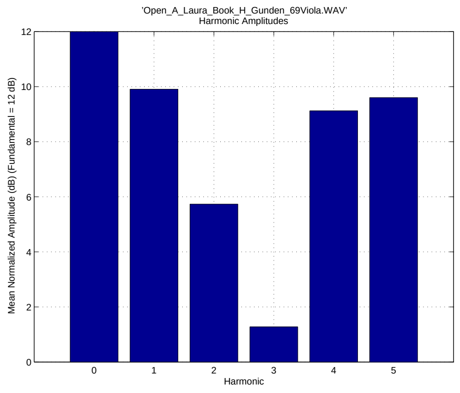


FIG. 10: Relative amplitude plot of the harmonics of an Open A on a 1969 H. Gunden Viola played by Laura Book.

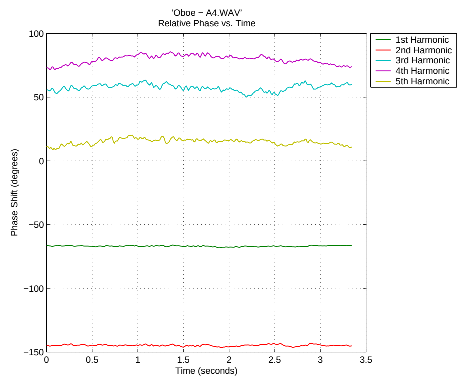


FIG. 11: Relative Phase of the harmonics of an A440 on a Yamaha Oboe played by Ma'ayan Bresler.

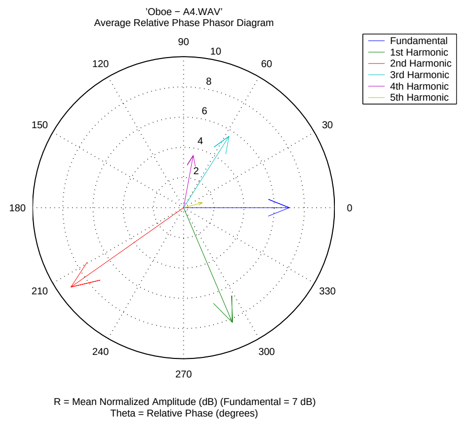


FIG. 12: Phasor plot of the harmonics of an A440 on a Yamaha Oboe played by Ma'ayan Bresler.

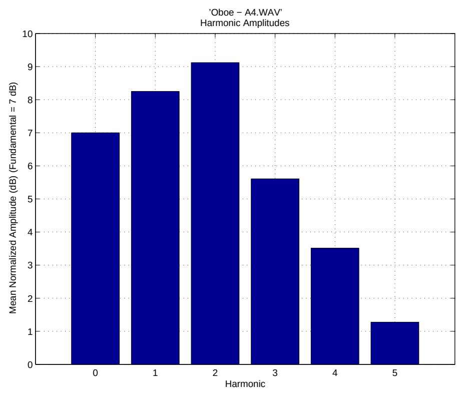


FIG. 13: Relative amplitude plot of the harmonics of an A440 on a Yamaha Oboe played by Ma'ayan Bresler.

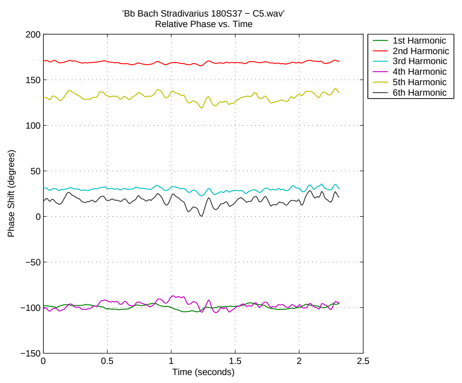


FIG. 14: Relative Phase of the harmonics of a C5(Concert B♭) on a Bach Stradivarius 180S37 Trumpet played by the author.

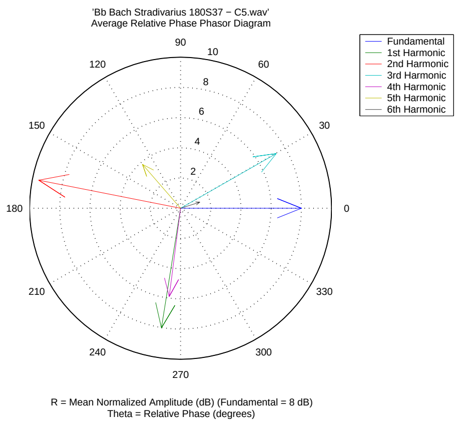


FIG. 15: Phasor plot of the harmonics of a C5(Concert B♭) on a Bach Stradivarius 180S37 Trumpet played by the author.

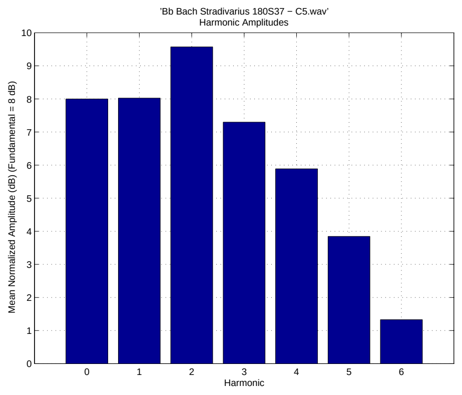


FIG. 16: Relative amplitude plot of the harmonics of a C5 (Concert B♭) on a Bach Stradivarius 180S37 Trumpet played by the author.