Transfer Function Measurement Toolbox

Edgar J. Berdahl and Julius O. Smith III

REALSIMPLE Project*

Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, and the

Department of Electrical Engineering
Stanford University
Stanford, California 94305

Abstract

The Transfer Function Measurement Toolkit is a simple, convenient, and free open-source solution for measuring impulse responses, magnitude spectra, and phase spectra of single-input, single-output (SISO) linear systems. Two measurement methods are explained and demonstrated. The Golay code measurement technique is particularly robust to additive white noise, while the swept sine measurement technique is robust to a weakly nonlinear motor exciting the linear system being measured. The toolbox source code is kept to a minimal length to facilitate further modification by others.

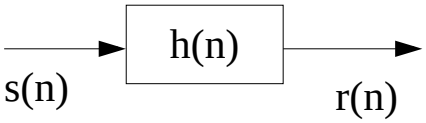

Figure 1: Linear system to be measured

*Work supported by the Wallenberg Global Learning Network

Contents

1 Summary Of Objectives

- To provide a general framework for *characterizing single-input, single-output linear systems.*
- To explain and demonstrate how to measure the *impulse response* of a linear system using *Golay codes.*
- To explain some of the limitations of making measurements using standard sound interfaces.
- To demonstrate how to find the *minimum-phase spectrum* corresponding to a complex spectrum.
- To explain how to measure the impulse response of a system even if the excitation source *motor is weakly nonlinear.* This measurement technique uses a *sine sweep* test signal.

2 Installation

1. Install a sound card, sound interface, or other full-duplex data acquisition card.
2. Consider testing the data acquisition card viewing the soundcard set-up instructions.[1]
3. Install either `MATLAB` or `Octave`.
4. Install `pd`. (Alternatively, you may use other software that is capable of recording a system's output for a given input excitation signal. The software should also be capable of reading and writing WAV files. For example, any multitrack recording software should be fine.)
5. Download `tf_meas.zip`[2] and unzip the contents into a conveniently located local directory.

3 Linear System

Consider the causal, single-input single-output (SISO) system shown in Figure 2. For simplicity, we will take the system to be linear and discrete-time, so that it is characterized by its impulse response $h(n)$ or equivalently its transfer function $H(z)$, which is the z transform of $h(n)$.

$$h(n) \longleftrightarrow H(z) \tag{1}$$

We will assume that both $h(n)$ and $H(z)$ exist so that we can discuss measuring them interchangeably. We will further assume that $h(n)$ has finite length so that we can measure the response to an input signal in a finite amount of time. The goal of this document is to explain how to excite the system with a signal $s(n)$, measure the response $r(n)$, and use $s(n)$ and $r(n)$ to determine $h(n)$ (and equivalently $H(z)$). In particular, it is useful to pick a signal $s(n)$ that contains a large amount of energy so that measurement noise will not significantly corrupt the measurement results.

[1] http://ccrma.stanford.edu/realsimple/soundcard_test/
[2] http://ccrma.stanford.edu/realsimple/imp_meas/tf_meas.zip

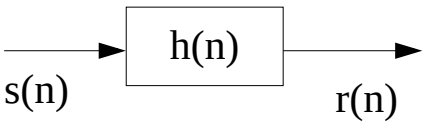

Figure 2: Linear system

4 Limitations Of Sound Interfaces

Sound cards and sound interfaces are not designed for making transfer function measurements. They merely provide a cost-effective solution since almost all computers have sound cards. We demonstrate these weaknesses given measurements made on a PreSonus Firepod sound interface. The output from channel 1 was directly connected to the line input on channel 1, and the sampling rate was $f_S = 44.1$kHz.

1. Sound interfaces attenuate frequencies near DC and near half of the sampling rate $f_S/2$. Figure 3 shows the measured magnitude response of the sound interface. Ideally it would be constant for all frequencies. However, Figure 3 shows that the -3dB point for the DC blocker lies near 5Hz. The anti-aliasing filters cause the magnitude to roll off sharply such that the high frequency -3dB roll-off point is about 20.9kHz.

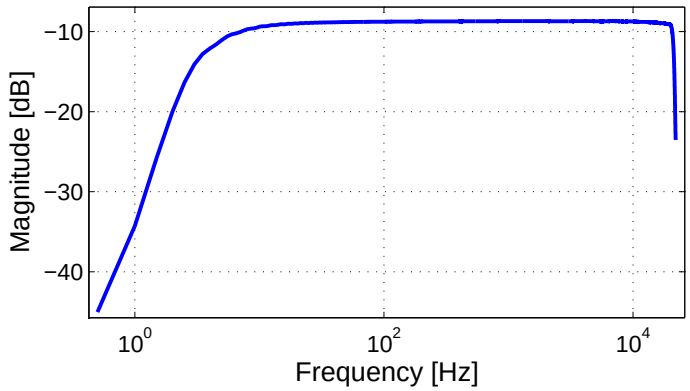

Figure 3: Magnitude response of the PreSonus FirePOD sound interface when the channel 1 output is directly connected to the channel 1 line input

2. The same sound interface filters cause the phase response measurement to be incorrect as well. Although ideally the phase response would be 0 radians for all frequencies, Figure 4 reveals that the phase measurement is distorted in roughly the same regions where the magnitude measurement is distorted.

3. Even when a computer is programmed to pass an input signal flowing into the sound interface input out of the sound interface output as fast as possible, there is a *delay* or *latency*. This delay is typically on the order of tens of milliseconds. Figure 5 shows the impulse response measured on the PreSonus card. The latency is approximately 62ms. The latency could have been decreased some by adjusting software settings in `pd`. The impulse response also rings noticeably due to the anti-aliasing filters.

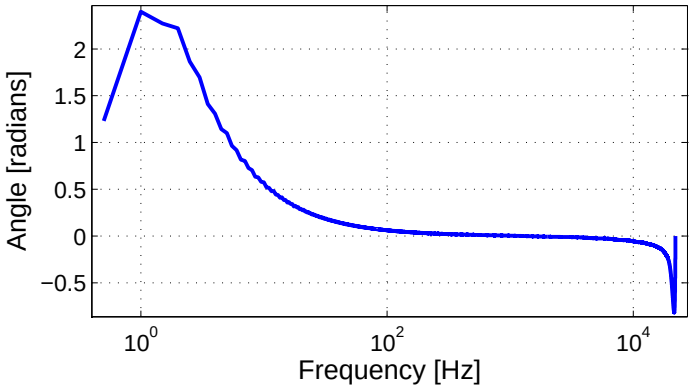

Figure 4: Phase response of the PreSonus FirePOD sound interface when the channel 1 output is directly connected to the channel 1 line input

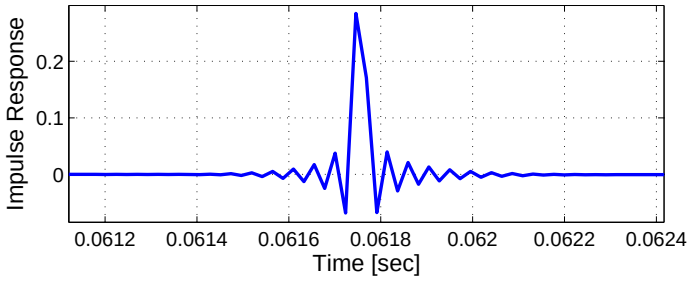

Figure 5: Impulse response of the PreSonus FirePOD sound interface when the channel 1 output is directly connected to the channel 1 line input

For reference, we provide the measured impulse response[3] shown in Figure 5.

5 Minimum Phase Systems

One further consequence of the delay is that determining the phase response of the measured system is more complicated. The delay is responsible for a linear phase term since $\delta(n-k) \longleftrightarrow e^{-j2\pi fk/f_S}$. If the delay is known (or measured), then it may be removed by multiplying the measured spectrum by $e^{j2\pi fk/f_S}$. However, if the system being measured is known to be minimum phase, then *this method*[4] may be applied to find the minimum phase frequency response corresponding to the measured frequency response.

The transfer function measurement toolbox assumes that the system being measured is minimum phase. This is a valid assumption in many cases. For instance, all strictly positive real transfer functions are minimum phase. Dissipative systems are strictly positive real (and therefore minimum phase) if the appropriate quantity is measured and the sensor and motor are collocated. For example, if $s(n)$ controls a motor exerting a force on a dissipative system, and $r(n)$ is the velocity at that same point, the corresponding transfer function will be minimum phase. This holds for other dual variable pairs such as torque and angular velocity, voltage and current, and pressure and fluid flow.

For systems that are not minimum phase, such as systems involving a transmission delay between the input and output quantities, the phase plotted by the transfer function measurement toolbox is not the system phase response, but rather the minimum phase response corresponding to the measured system phase response.

6 Golay Code Theory

The length L bilevel sequences $a(n)$ and $b(n)$ are *Golay* if and only if the following condition holds, where $\star$ is the autocorrelation operator [1]:

$$a(n) \star a(n) + b(n) \star b(n) = 2L\delta(n) \tag{2}$$

$\delta(n)$ is the Kronecker delta function. Recall that (2) can also be written using $*$, the convolution operator.

$$a(-n) * a(n) + b(-n) * b(n) = 2L\delta(n) \tag{3}$$

Given that $a_L(n)$ and $b_L(n)$ are Golay, it turns out that $a_{2L}(n) = [\ a_L(n) \quad b_L(n)\ ]$ and $b_{2L}(n) = [\ a_L(n) \quad -b_L(n)\ ]$ are also Golay. This means that Golay sequences can be constructed recursively given Golay seed sequences such as $a_2(n) = [\ 1 \quad 1\ ]$ and $b_2(n) = [\ 1 \quad -1\ ]$. See the MATLAB/Octave source code `generate_golay.m`[5] for details. Notice also that the resulting bilevel sequences consist of only 1's and -1's. This means that the signal contains the maximum possible power level given that $|s(n)| \leq 1 \forall n$. This property is helpful in combatting measurement noise.

[3] http://ccrma.stanford.edu/realsimple/imp_meas/direct2ImpResp.wav
[4] http://ccrma.stanford.edu/~jos/filters/Matlab_listing_mps_m_test.html
[5] http://ccrma.stanford.edu/realsimple/imp_meas/generate_golay.m

Let $r_a(n) = a(n) * h(n)$ be the response due to the Golay code input $a(n)$, and let $r_b(n) = b(n) * h(n)$ be the response due to the Golay code input $b(n)$. Due to (2), the impulse response $h(n)$ may be determined as follows:

$$h(n) = \frac{1}{2L}(a(n) \star r_a(n) + b(n) \star r_b(n)) \tag{4}$$

See `golay_response.m`[6] for more details.

7 Golay Code Measurement Procedure

1. Generate the Golay codes `golayA.wav`[7] and `golayB.wav`[8] using `generate_golay.m`[9].
2. Open the `pd` patch `golay.pd`,[10] in `pd`.
3. Ensure that the patch is not in editing mode, and check the "compute audio" box in the main `pd` window.
4. Adjust the "Output Volume" so that when you click on "Record Response to Golay A", the system under test is behaving linearly (i.e. not clipping), but so that the input signal to the sound interface is not too noisy.
5. If there is an input volume on the sound interface, adjust it so that the levels approximately match those shown in Figure 6 when you click on "Record Response to Golay A" and "Record Response to Golay B." If the sound interface has no input volume, then you will need to adjust the "Output Volume" accordingly.
6. Once you are satisfied with the results, click the "Write Responses to Disk" button.
7. `pd` will write the files `RespA.wav` and `RespB.wav` to disk. Rename these files so that the names match the measurement you just made. For instance, you might rename them to `hpfRespA.wav`[11] and `hpfRespB.wav`[12] if they corresponded to the measurement of a high-pass filter.
8. Run `golay_response('hpf')`[13] in `MATLAB` or `Octave` to analyze the measured response. Plots will be created, and the file `hpfImpResp.wav`[14] will be written to disk.

8 High Pass Filter Measurement

The circuit shown in Figure 7 was measured to show how the sound interface non-idealities affect a measurement. V_{IN} was connected to the output of channel 1 of the PreSonus sound interface, and V_{OUT} was connected to the line input of channel 1 of the interface.

[6] http://ccrma.stanford.edu/realsimple/imp_meas/golay_response.m
[7] http://ccrma.stanford.edu/realsimple/imp_meas/golayA.wav
[8] http://ccrma.stanford.edu/realsimple/imp_meas/golayB.wav
[9] http://ccrma.stanford.edu/realsimple/imp_meas/generate_golay.m
[10] http://ccrma.stanford.edu/realsimple/imp_meas/golay.pd
[11] http://ccrma.stanford.edu/realsimple/imp_meas/hpfRespA.wav
[12] http://ccrma.stanford.edu/realsimple/imp_meas/hpfRespB.wav
[13] http://ccrma.stanford.edu/realsimple/imp_meas/golay_response.m
[14] http://ccrma.stanford.edu/realsimple/imp_meas/hpfImpResp.wav

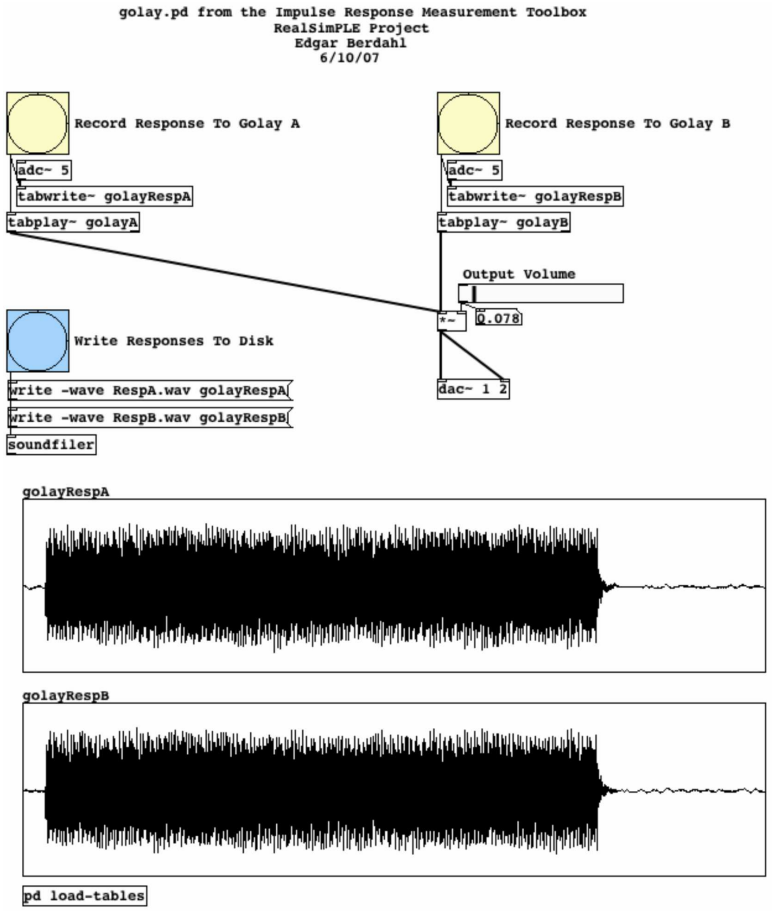

Figure 6: `golay.pd` after making a measurement with an appropriate input level

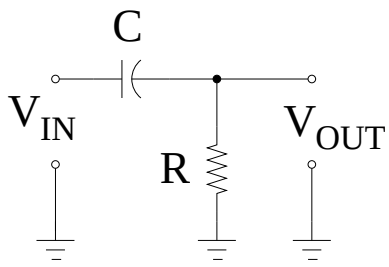

Figure 7: High pass filter electrical circuit

The analog transfer function $H(f)$ can be determined analytically using the voltage divider rule:

$$H(f) = \frac{V_{OUT}(f)}{V_{IN}(f)} = \frac{R}{R + \frac{1}{j2\pi fC}} = \frac{j2\pi fRC}{j2\pi fRC + 1} \tag{5}$$

In this case, $R = 1\text{k}\Omega$ and $C = 0.47\mu\text{F}$, so the -3dB point is about $f_{3dB} = \frac{1}{2\pi RC} \approx 340\text{Hz}$. Figure 8 and Figure 9 show that the frequency response is accurately measured in the range of about 10Hz to about $9f_S/20$.

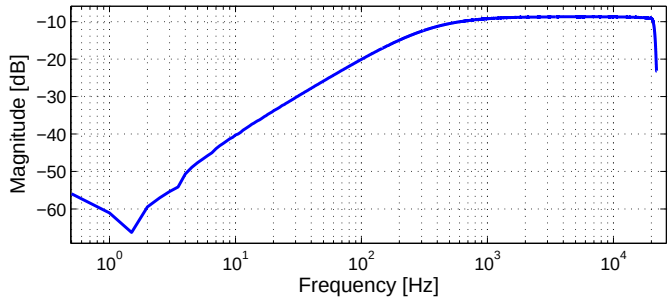

Figure 8: Measured magnitude response of the high pass filter

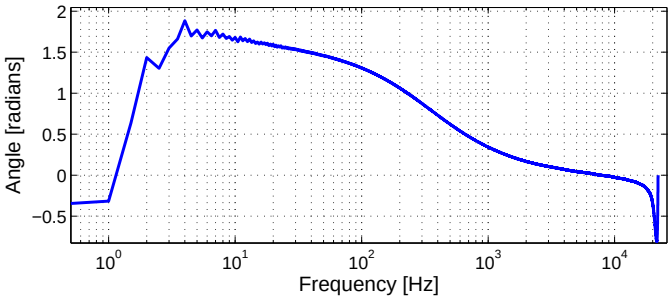

Figure 9: Measured phase response of the high pass filter

The ringing in the measured impulse response distracts from the more subtle characteristics of the ideal high pass filter impulse response. For transfer functions that pass large amounts of energy at high frequencies, it may be more instructive to inspect the frequency domain measurement results.

9 Sine Sweep Measurement Theory

In some cases, it is desirable to relax the power-maximizing constraint $|s(n)| = 1 \forall n$ in favor of obtaining some other desirable measurement system properties. For example, we may care more about the accuracy of the measurement at lower frequencies compared to higher frequencies, so we would like the excitation signal $s(n)$ to contain more energy at lower frequencies [1]. We might also be measuring a mechanical or acoustical system in which the motor controlled by $s(n)$ behaves weakly nonlinearly. If the nonlinearity is memoryless and is NOT preceded by any filtering, then

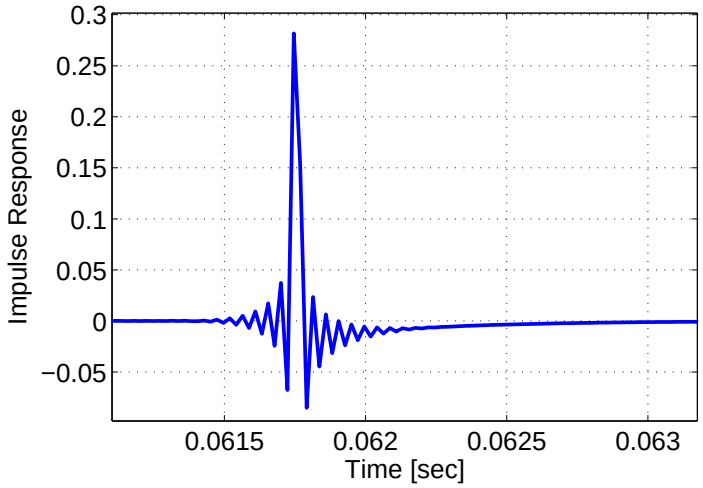

Figure 10: Impulse response

the system to be measured matches the Hammerstein model shown in Figure 11. The goal is to measure $h(n)$, independently of the motor nonlinearity $f(s)$. Performing the measurement is complicated by the fact that superposition no longer holds.

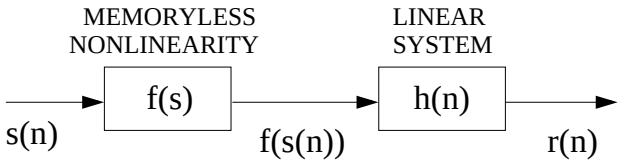

Figure 11: Hammerstein Model

Mathematically, the Hammerstein system behaves as follows:

$$r(n) = (f(s) * h)(n) \tag{6}$$

It turns out that we can obtain both of these desirable measurement system properties by using a new excitation signal $s(n)$. This signal is a sine wave, whose frequency is exponentially increased from ω_1 to ω_2 over T seconds [2].

$$s(n) = \sin[K(e^{-n/Lf_s} - 1)] \tag{7}$$

where $K = \frac{\omega_1 T}{\ln \frac{\omega_2}{\omega_1}}$ and $L = \frac{T}{\ln \frac{\omega_2}{\omega_1}}$. The MATLAB/Octave code `generate_sinesweeps.m`[15] generates the appropriate sine sweep.

The important property of $s(n)$ is that the time delay Δt_N between any sample n_0 and a later point with instantaneous frequency N times larger that the instantaneous frequency at $s(n_0)$ is constant:

$$\Delta t_N = T \frac{\ln(N)}{\ln \frac{\omega_2}{\omega_1}} \tag{8}$$

[15] http://ccrma.stanford.edu/realsimple/imp_meas/generate_sinesweeps.m

This characteristic implies that after inverse filtering the measured response, the signals due to the nonlinear terms in $f(s)$ are located at specific places in the final response signal. Consequently, the linear contribution to the response, which is proportional to $h(n)$ can be separated from the other nonlinear terms. We can thus measure a linear system even if it is being driven by a weakly nonlinear motor.

Because the frequency of $s(n)$ increases exponentially, the system is excited for longer periods of time at lower frequencies. This means that the inverse filter averages measurements at lower frequencies longer, so this measurement technique is better suited to especially low-pass noise sources.

10 Sine Sweep Measurement Procedure

1. Generate the sine sweeps[16] using `generate_sinesweeps.m`[17].

2. Open the `pd` patch `sinesweeps.pd`,[18] in `pd`.

3. Ensure that the patch is not in editing mode, and check the "compute audio" box in the main `pd` window.

4. Adjust the "Output Volume" so that when you click on "Record Response To The Sine Sweeps," the system under test is behaving linearly (i.e. not clipping), but so that the input signal to the sound interface is not too noisy.

5. If there is an input volume on the sound interface, adjust it so that the levels approximately match those shown in Figure 12 when you click on "Record Response To The Sine Sweeps." If the sound interface has no input volume, then you will need to adjust the "Output Volume" accordingly.

6. Once you are satisfied with the results, click the "Write Responses to Disk" button.

7. `pd` will write the file `Resp.wav` to disk. Rename this file so that the name matches the measurement you just made. For instance, you might rename it to `nonlinear2Resp.wav`[19] if it corresponded to the second time you measured the transfer function of a weakly nonlinear system.

8. Run `sinesweeps_response('nonlinear2, 100, 0.4')`[20] in `MATLAB` or `Octave` to analyze the measured response. This means that the inverse filter will be restricted to a dynamic range of 100 (40 dB), which helps avoid exaggerating problems beneath ω_1 and above ω_2, where the excitation signal has little energy. 0.4 refers to the length in seconds of the linear impulse response term to be extracted. Plots will be generated, and the file `nonlinear2ImpResp.wav`[21] will be written to disk.

[16] http://ccrma.stanford.edu/realsimple/imp_meas/sinesweeps.wav
[17] http://ccrma.stanford.edu/realsimple/imp_meas/generate_sinesweeps.m
[18] http://ccrma.stanford.edu/realsimple/imp_meas/sinesweeps.pd
[19] http://ccrma.stanford.edu/realsimple/imp_meas/nonlinear2Resp.wav
[20] http://ccrma.stanford.edu/realsimple/imp_meas/sinesweeps_response.m
[21] http://ccrma.stanford.edu/realsimple/imp_meas/nonlinear2ImpResp.wav

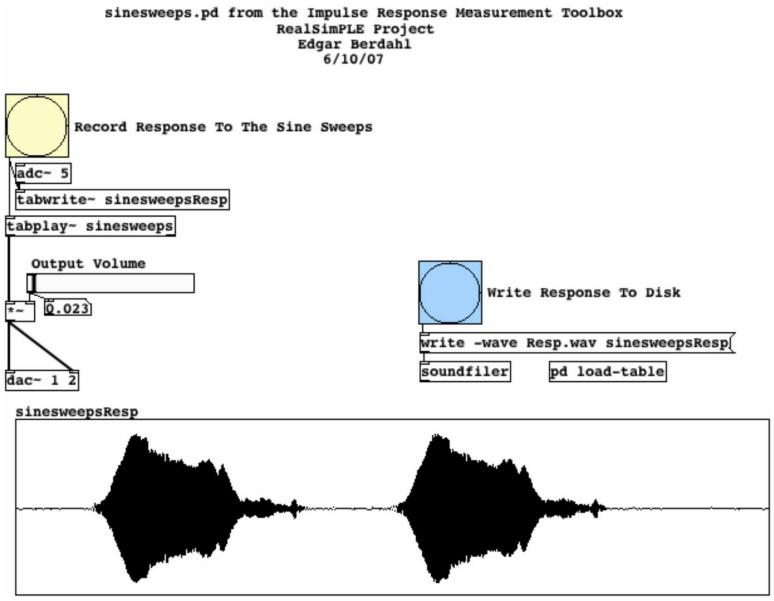

Figure 12: `sinesweeps.pd` after making a measurement with an appropriate input level

11 Sine Sweep Measurement of a Weakly Nonlinear Loudspeaker Driver

To exeraggerate the nonlinearity of a loudspeaker, we cut the cone of a mishandled driver as shown in Figure 13. We monitored the sound pressure several centimeters in front of the dustcap using an Audio Technica AT4049a microphone, which has a flat magnitude response to within 3dB from 100Hz to 5kHz. The output from channel 1 of the PreSonus sound interface was connected to the speaker via a power amplifier, and the microphone was connected to the microphone input of channel 1 on the sound interface. The following results are typical of sine sweep measurements with a weakly nonlinear motor.

Inverse filtering the measured response results in Figure 14, which is a plot of `nonlinear2ImpResp.wav`[22]. The linear contribution corresponds to the spike at the beginning, while the weakly nonlinear terms are clustered closer to the end of the response.

The main linear contribution is cut out and plotted in Figure 15. The measurement was not made in an anechoic chamber, so there is a reflection about 15ms after the main impact.

The nonlinear terms are shown magnified in Figure 16. The lower order nonlinear terms toward the right have larger magnitude but overlap less in time (see Figure 16). Note that (8) implies that the overlapping could be reduced by increasing the total length T of the sweep excitation signal.

The magnitude and phase responses corresponding to the linear impulse response term from Figure 15 are shown in Figure 17 and Figure 18 in *blue*. For comparison, another sine sweep measurement was made at a lower level so that the speaker behaved approximately linearly. Decreasing the level also resulted in more noise and even some systematic error, as is evidenced by the *red* curves in Figure 17 and Figure 18. This comparison demonstrates that making measurements at larger levels can reduce the effects of noise, while nonlinear motor effects can be overcome with the

[22] http://ccrma.stanford.edu/realsimple/imp_meas/nonlinear2ImpResp.wav

Figure 13: Mishandled driver with additional cuts in the cone

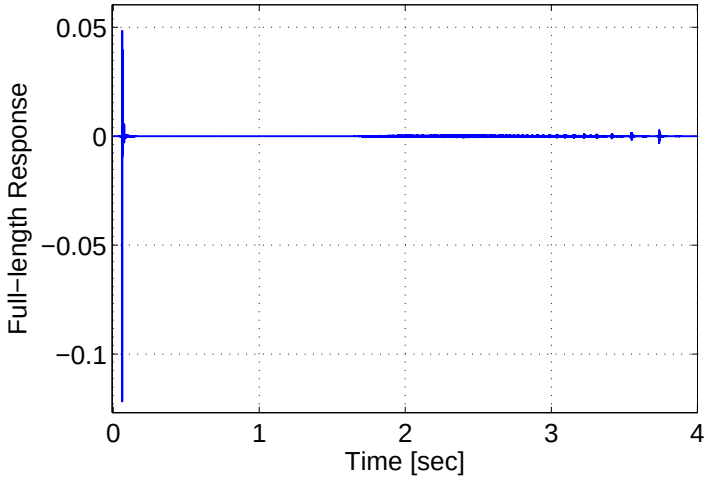

Figure 14: Full-length response

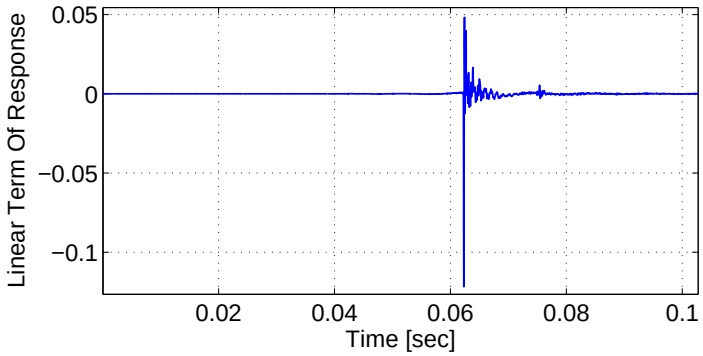

Figure 15: Linear impulse response term

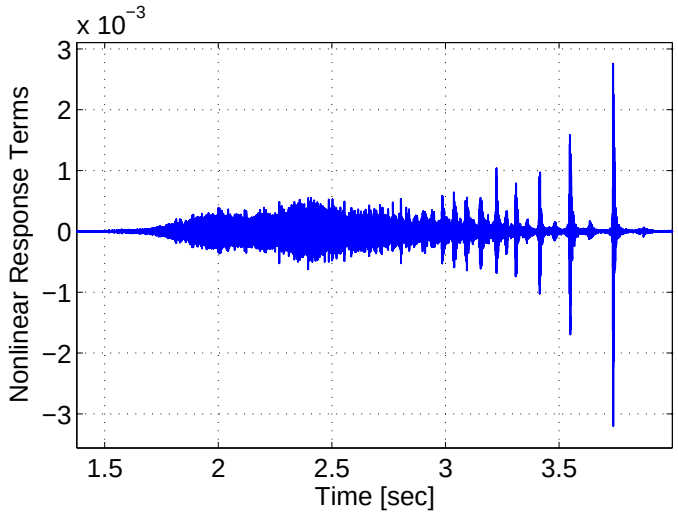

Figure 16: Nonlinear response terms

sine sweep measurement technique.

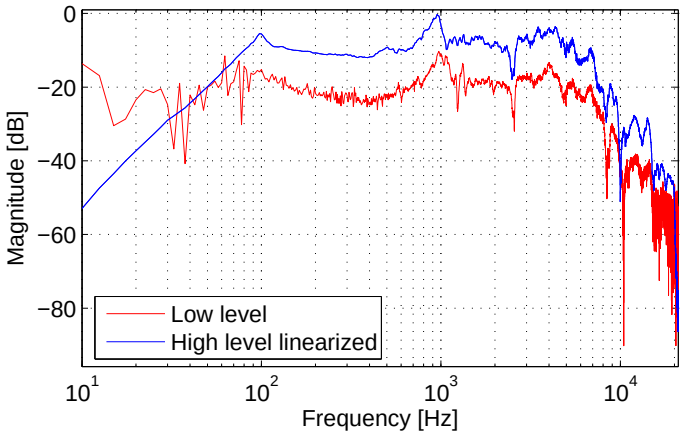

Figure 17: Proportional to magnitude response of $h(n)$

References

[1] J. Abel and D. Berners, *Signal Processing Techniques for Digital Audio Effects*, `http://ccrma.stanford.edu/courses/424/`, 2005.

[2] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," *Audio Engineering Society Convention*, vol. Preprint 5093, Feb. 2000.

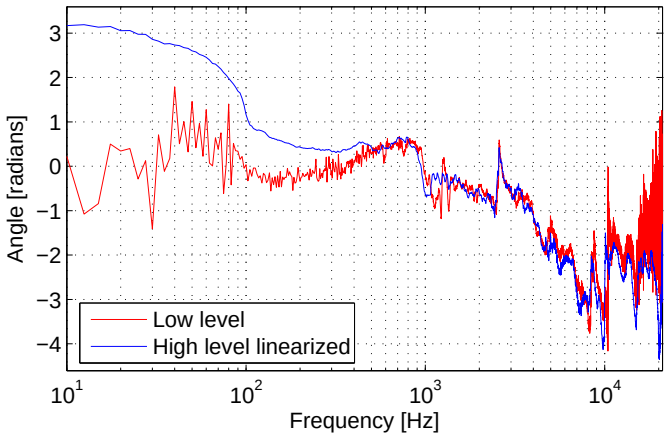

Figure 18: Phase response of $h(n)$