An Open-Source Electroacoustic Measurement System

Part 1: Theory, Practicalities & Acoustic Examples

(revised May 20, 2017)

John Vanderkooy and Richard Mann

Overview

These papers describe the design, configuration, software and use of a general-purpose electroacoustic measurement system. The system is based on typical soundcards, of which a few are featured. The software used is GNU Octave, a powerful, free, open-source analysis tool that uses a high-level language which is quite readable and understandable by anyone with a bit of knowledge of mathematics and signals. Matlab prompted the development of Octave, and is an even more responsive but costly commercial development environment. The aim is to place a powerful analysis system in the hands of anyone, encourage high-level electroacoustic measurements and promote their understanding. There are quite a few free or low-cost analysis programs available, so the focus of this offering is to teach the details of how measurements are accomplished, and promote programming by the user. A fair amount of general measurement theory is presented, together with the mathematics and implementation. Programs are given and explained, along with a few examples. Excitation signals can be logarithmic sweeps, sine waves, noise, or even music. The measurement of loudspeakers and other electroacoustic devices are the main concern, but the system is useful to characterize any electronic system. Measurements of rooms or music venues are dealt with to obtain quasi-anechoic responses, harmonic distortion, reverberation times and other room acoustic parameters.

Part 2: Sound Card Setup, System Characterization and more Examples, shows details of the setup program, qualification of the soundcard, signal limits, noise floors, and other interesting measurements.

Readers who are quite familiar with measurement concepts may hold the topics below in abeyance and skip to the section on Software Example Programs.

Introduction

Over the years, electroacoustic measurement systems have slowly evolved from being purely analog to mostly digital. An archetypal purely analog system would be a motor-driven logarithmic sinewave sweep oscillator, coupled to a chart recorder whose pen is driven from a logarithmic potentiometer with fast-acting slip clutches driven from an appropriate detector. These systems are a real marvel, and there are probably some still in use. A later partially digital system was the time-delay spectrometer pioneered by Richard Heyser [1]. This system used a computer to generate a linear sweep with a DAC, and a tracking filter employing an analog multiplier was used to select an appropriate time window, so that quasi-anechoic measurements could be made, removing much of the reverberation and noise. Such a system provided a complex transfer function, and over time the concepts of phase became more commonplace in the audio community. During this period, more professional multi- or dual-channel FFT analyzers became available that analyzed their signals fully digitally, but cost often limited their audio use to research. They were also not trivial to operate and did not serve the general audio community that well, but were pivotal for mechanical vibration studies.

As digital hardware became more affordable, due partly no doubt to the tremendous growth of digital music and signal processing, test systems of all kinds sprang up. Soundcards were readily available and some of them were capable of being used in test systems. Most of these were meant for computer audio playback, and were AC coupled. Many of them were bus-connected devices, so a desktop was necessary and portability was sacrificed. Laptops have soundcards with DACs and ADCs which often have questionable characteristics, and terrible microphones, so except for a few, they are unsuitable for decent performance. Some consumer soundcards are discussed in this paper, including a really cheap unit, a Behringer UCA202/222, an ART USB Dual Pre, and a Focusrite Scarlett 2i2. Many other soundcards work as well. If the operating system recognizes it, it will probably work. A USB soundcard is very convenient, but a PCI or Firewire unit is fine as well.

Important electroacoustic measurements are often made with a real-time spectrum analyzer, or a fractional-octave acoustic analyzer. Such measurements are somewhat qualitative, but they offer real-time interaction with the environment. We will not focus on this, since there are programs available that do this fairly well with normal soundcards.

It is assumed that readers who choose to tackle a measurement system are familiar with electronics and signals. In addition, it helps to be familiar with some mathematics, calculus, and concepts such as DFT, FFT, DAC, ADC, dB, Transfer Function, Nyquist frequency, and arrays of numbers, since these are woven throughout measurement systems. Anyone aspiring to use such a system will learn a lot. Our approach is to leave the data plotted and available in the program workspace environment

for further use, rather than to provide a turnkey system with fixed analysis functions. The paper gives considerable detail on how the signal is analyzed and manipulated, and more programs and measurement functions will be presented on the web.

The Microphone

Many of you will already have a microphone and a soundcard that you might have used for music recording. Any electroacoustic project will require an omnidirectional microphone that has a reasonably flat acoustic response. These are usually fairly cheap, and often have a battery and a cable meant to plug them into the microphone input of a laptop. Cardioid mics are not optimal, since they often have poor bass response. If you have some old equipment like a cassette recorder, an old laptop or even an unwanted telephone, you will likely be able to salvage a small electret microphone. Such mics are usually 2-terminal and will require several volts of bias through a small resistor of typically 4.7Kohm, and an electrolytic coupling capacitor. If the soundcard has at least 20dB of input gain, the dynamic range of the combination will be good, but even if the soundcard has unity input gain, such as an inexpensive Behringer UCA 202/222, acceptable results can be obtained with some care. Many soundcards have analog mic preamps or balanced hi-Z instrument inputs with input level controls. You will need a power amplifier to drive loudspeakers. A few volts is usually sufficient.

The Hardware: DACs and ADCs

In order to understand in detail the nature of a modern measurement system, it helps to clearly understand the nature of the excitation signal from the DAC, how it is altered through the device under test (DUT), and how the ADC samples an analog signal to achieve a measurement. A useful introduction to digital sampling can be found in [2], but trawling the web will bring up some amazing sites. The analog test signal itself is generated by sampling a notional analog continuous-time waveform into a stream of digital numbers, and presenting them to a DAC. In early digital systems, the analog output was held between samples and updated to the new value when the next sample arrived. This 'zero-order-hold' characteristic caused a frequency-response reduction of the system excitation by a factor *sinc(f/fs)*, called an aperture loss. The excitation would be 3.9 dB down at the Nyquist frequency, *fs/2*, and its variation needed correction in the baseband. Modern DACs in soundcards have oversampling internal clocks, and make all the necessary corrections when reconstructing the output at very high rates. An anti-aliasing (AA) filter is necessary to prevent out-of-band signals from falsifying the digital samples acquired by the ADC. Modern oversampling delta-sigma ADCs have a low-order analog filter, feeding a low-bit noise-shaped converter, followed by a steep digital AA decimation filter. They tend to many of the gory details, and the user of the soundcard can then concentrate on the excitation signal and the recorded response.

In what follows, we have used normal soundcards that are AC coupled. Their response usually goes down to 10 Hz or so, and you may be tempted to increase all the coupling capacitors to make the unit respond to much lower frequencies. There are several impediments to this. First, most soundcard ADCs have a digital highpass filter set at about 4 Hz for a sampling frequency of 44100 Hz. This filter frequency scales with fs, so using the card at 11025 Hz will bring it down just below 1 Hz. Some interfaces may allow you to disable the filter. Second, you must change *all* the capacitors in the signal path, including the mic preamp. Modern surface mount components make this difficult.

For some measurements, DC coupling is really necessary, such as infrasound systems and mechanical servos. The best choice then is a truly DC coupled data acquisition card. We have experimented with a National Instruments myDAQ, a relatively inexpensive, student-oriented device, and it can form the basis for a comprehensive system. However, you must now incorporate all the software and drivers to operate the data card. Since DC coupled systems usually use successive-approximation ADCs and DACs which do not oversample, the user must tend to all the issues of AA filtering and removing signal glitches, etc. We have chosen to avoid all those issues!

System Architecture

There is some flexibility in how we choose to use the DACs and ADCs of a soundcard. We need only one DAC output channel for the system excitation, the second channel is not usually necessary, although it could have a polarity-inverted signal for driving balanced inputs, even though many devices already use balanced outputs. The two ADC input channels need antialiasing protection, and that is already provided by most stereo codecs in the soundcard. A soundcard capable of record-play at 96 or even 192 kHz is useful for a measurement system, since it is capable of measuring the response well beyond audible frequencies, allowing aspects like filter rolloff and out-of-band spuria to be studied. However, you must check if the soundcard actually delivers the increased bandwidth. The Focusrite Scarlett 2i2 2nd-gen works nicely up to 40 kHz with sampling frequencies of 88.2 and 96 kHz. At 192 kHz the transfer function is 7dB down at 80 kHz. In the earlier version of this document it was stated that "at higher sampling rates there are relatively gentle filters that remove much of the higher frequencies, don't offer as much antialiasing protection, and have increased distortion." However, the Focusrite 2i2 is quite good even at 192 kHz, and with reference normalization it can provide accurate measurements to beyond 80 kHz.

For audio we do not recommend 192 kHz for any soundcard. The ADCs and DACs are significantly compromised at these frequencies, and it is sometimes true even at 96kHz. Some may consider this better for high-quality audio, but there is really no evidence for that. A simple 44.1/48 kHz soundcard is fine for normal work. For a measurement system, though, a higher sampling frequency may be useful.

Figure 1 shows a typical block diagram of the system architecture. In this paper we will use *italics* for variables. Ch-1 DAC is fed a digital excitation signal, *sig*, which becomes an analog signal fed both as reference to Ch-1 input channel, and also as input to the DUT. The Ch-1 digitally captured analog reference signal is called *ref*. The DUT may represent a power amplifier driving a loudspeaker, and a microphone with preamp to provide an output signal. The DUT output goes to Ch-2 input and is digitally captured as *dut*.

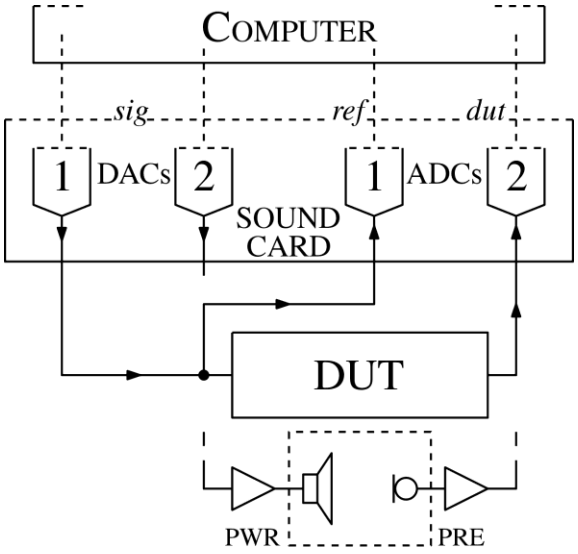

Figure 1. Typical system architecture showing how the computer sends and receives digital data signals *sig*, *ref* and *dut*, representing the excitation signal, a measured signal reference, and the output of the device under test. The DUT may be a power amplifier feeding a loudspeaker in a room, with a measuring microphone and preamp producing an analog output signal. The actual analog signals are never really known by the analysis program, but if the DACs and ADCs are of high quality, we can think of *sig*, *ref* and *dut* as faithful representations of those signals. Many soundcards have adjustable preamps in the *ref* and *dut* input lines.

Our system is focused on measuring both the transfer function and the time domain response, which can be derived in several ways. One method is to compare the measured output signal *dut* with the generated excitation *sig*. The AA-filter characteristic then is included in the net response, and there may be an uncertain delay between the output signal *sig* and the captured, recorded signals *ref* and *dut*. A second method to derive the transfer function or time response of the DUT is to compare the recorded output signal *dut* with the recorded signal *ref*. This avoids any record-play delays and gives excellent normalized results, but since the *ref* signal has very low energy near the Nyquist frequency due to the AA filter, the transfer function will be noisier there. We can surmount that difficulty by ignoring or weighting down that spectral region, since it is beyond the passband of the AA filter. Both methods of obtaining the transfer function are illustrated in the programs to be presented. The time domain or impulse response of the DUT is then obtained by an inverse Discrete Fourier Transform (DFT) of the transfer function.

It is worth reiterating that all the data, *sig*, *ref* and *dut* are actually arrays of digital samples in the memory of the computer. Of course we think of them as analog voltages, derived from good DACs and ADCs, but from the measurement program's point of view, they are digital data, which will need calibration factors to refer to real signals.

Resampling and Operating System Problems

Although we may think that our soundcard is functioning at a particular sampling frequency, channel assignment, and bit width, there are some pitfalls of which you should be aware. In a program, we may set a sampling rate that is different than that set by the operating system. What seems to happen is that the hardware actually obeys the Windows setting, but the data is then resampled to be admitted to the program. We discovered this for several soundcards during tests of loopthrough transfer functions, which usually show an AA filter that has a steep stopband rolloff just slightly below half the sampling

frequency. In one case the soundcard at 44100 had a response that was flat to 20kHz and plummeted down quickly beyond that, as it should. The program demanded 48000 sample rate and the resampling function must have used a software AA filter, resulting in returned data that was down by over 10dB at 20kHz. Such resampling AA filters are usually much sloppier than the hardware AA filters in the codecs. To avoid resampling, we must make sure that the operating system setting is the same as requested in the program. In a measurement system it is worth checking, with a loopthrough, that the AA filter is what it should be. Of course the system setting must be one that is actually supported, and it will be presented as a choice if the system has successfully interrogated the hardware.

The operating system may alter settings, gains and sampling rates without the intervention of the user. For example, the signal inputs are sometimes given a high gain so that low-level microphone signals will be in the fractional volt range. This *microphone boost* can easily overload a soundcard input. Reducing the input signal amplitude might increase the noise level. Microphone boost should be defeated, unless the microphone has no preamp. Digital changes may also be made without the user's consent. We found, for example, that a simple Bearinger UCA202 soundcard, which only has line inputs and outputs, no volume controls nor necessary driver (it is class compliant), had some 30dB of input gain added by Windows7. The ART USB Dual Pre had the same problem, understandably since its chipset is similar and it is also class compliant. Since it was impossible for Windows to change the input gain, it must have simply increased the level by a large factor digitally. In order to use the full ADC range, we had to use control panel to force a record gain of 4 out of 100! This leaves the digital signal with a good dynamic range. Such bizarre behaviour was unexpected, and we must leave debugging specific hardware to the individual user. We have nothing but scorn for such behaviour in operating systems. Why should the system not simply relay the ADC digital output to the application program? Surely an operating system must have options to leave well enough alone! Linux seems to be blameless…

Another common situation we experienced was the resetting of soundcard parameters when we used various cards and systems in our testing. Sometimes the operating system would set the recording device to single channel, which is what might normally be expected since laptops have only one microphone. Simply disconnecting and reconnecting the card may set parameters back to some default. Be vigilant! If a soundcard habitually sets itself back to 44.1 or 48 kHz, it's wise to leave the measurement programs with the same default.

Record-Play Delays

Any particular soundcard has its signature of operation. The best ones will *output a stimulus and simultaneously record the response* on all available channels, but most of them do not come up to that pinnacle of operation. For even the best drivers, USB soundcards will inevitably give delays between output stimulus and recorded response, and they typically will not be an integer sample, nor will they be the same for each instance. We have dealt with this by determining the delay using a crosscorrelation analysis *for each measurement*, and with care even fractional delays could be dealt with. We found that for the playback and record commands of Matlab and Octave, *the recorder starts **before** the player for all normal soundcards*. This means that we do not need to worry about the recorder not capturing the beginning of the stimulus signal, rather, we must ensure that the end of the stimulus play file is also properly recorded. Operating systems also matter. Linux generally seems to have considerably shorter record-play delays than Windows.

In order to calculate the requirements for the data files, let us call *T_sweep* the intended time duration of the sweep. Suppose that the player is delayed and starts a time *T_delay* after the recorder. This means that the record time should be at least *T_sweep+T_delay* in duration, so that the recorder will capture the whole sweep. The stimulus file will have duration *T_sweep* for the sweep, but may need to be zeropadded for some time, else the DAC may continue to output its last value, which may not be zero. We have chosen to make the stimulus file the same total length as the record file, even though the last zeropadded part will only be partially recorded.

A second requirement relates to the delay required to capture the reverberation of an acoustic measurement. We must add an extra post-data segment to the recording that is larger than the reverberation time *RT60* of the venue. Any errors in the captured data will then be at least 60dB down. Thus together with the sweep duration, the final total duration of the recording should be *T_sweep+T_delay+RT60*. The play file could be made the same length for convenience.

Once the data is recorded and the actual *T_delay* is precisely determined by a crosscorrelation, we should check that the allotted record duration includes the whole stimulus and the reverberant tail. Typical practice may be to double, so that we might post-zeropad the stimulus file by *2T_delay* at each end. This usually captures the reverberation sufficiently also, especially for ascending logsweeps that end at high frequency, since the reverberation time at these higher frequencies is generally lower. In the programs to follow we have simply zeropadded the last quarter of the excitation sweep.

There may be situations for which this is inadequate. In all cases we have to check meticulously how the system behaves. There may also be slight delays *between the channels* of a soundcard. There is no substitute for intelligence in the operation of a measurement system…

Two Channels?

Soundcards typically have two channels, and indeed it may be useful to have two full data analysis channels. However, as in the system architecture discussed earlier, we usually employ the first input channel data, *ref*, solely to determine the record-play delay of the analyzer. Several tactics are possible to get two useful channels. If phase is unimportant, the record delay doesn't need to be known, so this first channel can be used to record the output of a second system or microphone. It is often possible to determine a useable value of the delay from the data in either channel itself anyway, so a separate loopthrough may not be required.

Even when we use one channel as a time reference to remove delays, we must still take into account that the two input channels may not be simultaneously sampled. Soundcards sometimes sample their two channels alternately, giving a ½ sample time delay. When we do a transfer function analysis, this causes a linear phase shift with frequency which amounts to −90 degrees at the Nyquist frequency. This should be compensated if we wish accurate phase determinations.

Many soundcards have a mixing control, which allows direct monitoring so that the output channels are a blend between the analog input signal (labelled '*Preamp*', with no latency) and the DAC output signal (labelled '*Computer*', which has some latency). This may apply to only the headphone output, but often applies to the main outputs as well. *For a measurement system, it should be set to 'Computer'*, since any input signal which bleeds through may cause instability due to the reference loopback, and it falsifies the measurements. If the direct monitor has a switch, turn it off.

A Trick to Avoid Crosscoupling

We noticed in Behringer UCA202 and UCA222 soundcards that there was a significant crosscoupling (−30dB) at low frequencies between the input channels when the two output channels were driven with an in-phase sweep signal. This makes the unit unfit for a measurement system, when used that way. It appears that this is due to inadequate decoupling of the midrail voltage that is shared between ADCs and DACs in the Burr-Brown 2902E codec. To reduce such coupling by 40dB, we would have to increase the decoupling capacitor by a factor of 100, quite impractical. When the output channels are driven with equal antiphase signals, this crosscoupling is very low (−80dB), and the unit is then useful in a measurement system. The DACs draw much of the supply current and seem to be the problem; putting large signals symmetrically into the ADC input channels does not result in crosscoupling. It's possible that this behaviour is unique to the UCA card, but other codec chips may be similar. It is not burdensome to drive stereo outputs in antiphase. It may even be useful for driving balanced inputs, although many soundcards already have balanced outputs on each channel.

The Excitation Signals

In early measurement systems, the excitation was often designed to allow straightforward demodulation to achieve the desired measurement. The stepped fixed-frequency oscillator was later replaced by sine sweeps while fast detectors decoded the DUT output. Different sweeps have been devised for various purposes. We will highlight two main types: the linear sweep, and the exponential or logarithmic sweep. In Part 1 we will use only the logsweep.

A linear sweep changes its frequency linearly with time [1, 3]. If the sweep is defined by $s(t)=sin(\varphi)$, the frequency is $(1/2\pi)\, d\varphi/dt$, and we will set this to kt, where k is the sweep rate in Hz/s. Thus, ignoring integration constants, $\varphi=\pi\, k\, t^2$, so that the sweep becomes

$$s(t)=sin(\pi\, k\, t^2). \qquad (1)$$

Its power spectrum is constant, independent of frequency. The cosine can also be used, but generally the sine is preferred because it starts more gently at zero time. Decoding schemes might be simple average- or rms-responding detectors, coupled to a mechanical chart recorder. A tracking filter using an analog multiplier and lowpass filter could also be used. Such a filter can be manipulated to select the early or direct response, rejecting the reverberation, or it can focus on later parts of the response. The main problem with a linear sweep and tracking filter is that in order to resolve sharp resonances at low frequencies, a very low sweep rate must be used, resulting in excessive test duration [3, 4]. The linear sweep is just one of a more general set of powerlaw sweeps, described by $sin(\pi\, k\, t^{\alpha})$, where α is the sweep index [5]. Powerlaw sweeps with $\alpha>2$ have a spectrum that falls with frequency as $f^{(\alpha-2)/(\alpha-1)}$, whereas a linear sweep has a flat, constant spectrum.

A widely-used sweep which covers fractional changes of frequency in equal time intervals can be implemented by making the frequency exponentially dependent on time, and it is commonly called a logarithmic or logsweep. Suppose we make the

phase $\varphi=exp(t/L)$. The instantaneous frequency will be $(1/2\pi L)\ exp(t/L)$, and this frequency will change by a factor e=2.718 in a time L. If the frequency advances from *f_start* to *f_stop*, the sweep can be written as

$$s(t)=sin(2\pi\, f\_start\ L\ \{exp(t/L)-1\}), \qquad (2)$$

in which $L = T / log(f\_stop/f\_start)$, which defines L in terms of the total chosen sweep time, T. It is easily shown that the power spectrum of the logsweep varies as $1/f$, similar to pink noise, because the sweep spends equal lengths of time in each fractional frequency interval.

A very useful feature of a logarithmic sweep is that the harmonics generated by the DUT are separated by fixed time delays for each harmonic, so they are all *simultaneously* separated temporally in the impulse response [5]. This was recognized by us long ago [6], and others have extensively used it [7, 8, 9]. A harmonic analysis program, well commented, is discussed later and is available on the website.

Maximum-Length Sequences

Let us also discuss the well-known maximum-length sequence, or MLS excitation signal [10]. This binary signal is noise-like and has the highest possible crest factor of 1. Originally it was used because it required only modest computer resources, but that is not relevant today. When an MLS signal is used to excite the system, one period can be taken and crosscorrelated with the evoked DUT response in the time domain, to determine directly the system impulse response, much like the inverse filter approach for logsweeps. However, as with logsweeps, we can also determine the system transfer function using frequency-domain ratio methods, and the programs presented later as well as the soundcard setup program of Part 2 take that approach. There are programs for constructing MLS signals of different orders. MLS methods can be used to determine the system total intermodulation distortion [11] in a way that is not possible by any other technique.

Since the MLS method is a periodic analysis technique, we usually send two consecutive MLS sequences and analyze only the second, allowing the reverberation to build up to steady state for it. The transfer function will have all the reverberation of the system, which might be a loudspeaker in a room, and an inverse DFT will give the complete impulse response. This impulse response can be edited to have only the direct signal, removing most of the acoustic reflections, so that upon Fourier transformation, a quasi-anechoic response is obtained. Of course the same approach applies to logsweeps, or indeed to any excitation and analysis that provides a complete transfer function. We discuss later a program that does a quasi-anechoic measurement with a logsweep.

A weakness of the MLS technique is that it is sensitive to time variation, and this makes it less effective for determining the reverberation of a large space [12] due to things like air currents in the room, etc. Several other concerns have surfaced while preparing this paper. It seems that binary MLS signals cause odd behaviour in the sigma-delta devices of normal soundcards, perhaps due to internal overload. This means that we must be very careful and use somewhat lower signal levels, checking meticulously for such pathology. We found that for a number of soundcards, the DAC properties seem to result in noisy transfer functions, possibly related to internal overload of the switched-capacitor digital integrators. If the MLS digital signal level is restricted to be less than 0.5, the errors are insignificant. The ADCs were fine even with full-scale signal levels. A final detail of the analysis is that MLS sequences have a length N=(2^n)−1, and since we must take a discrete Fourier transform of exactly that length, we cannot avail ourselves of the fast algorithms such as the FFT. It's not a problem with modern processors.

Periodic or Single-pass

Most of the processing in measurement systems is done with the discrete Fourier transform, or DFT. It presupposes that both the time and frequency domains are periodic. This means, for example, that when we apply a logsweep or an MLS over the analysis timespan, and analyze the data, we have implicitly assumed that the logsweep or MLS is infinitely repeated. However, in reality we have only sent one instance, so that reverberation may not have built up to the actual level of a true repeated measurement. The implication is clear; if we want to make sure that the reverberation has built up sufficiently in our measurements, one way is to use repeated stimuli, and obtain a steady-state result of adequate length, or alternatively we could post-pend a sufficient zero-stimulus portion to the data so that the reverberation has settled down to below a specific signal level.

In swept measurements, we must capture some signal data even after the sweep has terminated. This will ensure that all the reverberation will be captured. It is customary to start the sweep at low frequencies, and end at high frequencies. This is sensible since low-frequency reverberation times are longer than for high frequency. The resulting impulse response will then be useful for such things as Schroeder plots for room reverberation, or calculation of room acoustic parameters such as

clarity, C50 and C80, or reverberation radius. We have already indicated in an earlier section how the stimulus is zero padded, and the recording is extended to accommodate both record-play delays and reverberation decay.

Fractional Sample Delays

Contrary to a common misunderstanding of digital sampling, *time resolution is not limited by the sampling rate* [13]. Perhaps the desire for some recording engineers to use high sampling rates stems from these misconceptions. These wrong notions still persist, but in a system with proper AA filtering, the time resolution is precise, far below the sampling interval. We outline several situations in which fractional sample delays may matter.

The first is the record-play delay suffered between the DAC excitation and the captured ADC signals. When we do a crosscorrelation to determine this delay, a fractional delay could be extracted, by fitting the peak with a polynomial or spline function. Extracting this delay can also be done by symmetrically taking a number of samples on either side of the peak sample which covers the peak, and calculating the 'centroid sample', p, from the normalized sum,

$$p=\Sigma\, n\, P[n]/\Sigma\, P[n], \tag{3}$$

where $P[n]$ is the power or squared amplitude at sample n. It is imperative that the samples chosen take in most of the energy symmetrically in the peak of the crosscorrelation. The delay in samples, p, will be a real, non-integer number, and allows the frequency f to be normalized to zero delay by multiplying by $exp\{2\,\pi\, j\, f\, p\,/\, fs\}$, where fs is the sampling frequency. The DFT frequency domain must remain conjugate even, so that an inverse DFT will result in a real time domain. The maximum error that occurs, if we simply use the peak sample, is half a sampling interval. This would result in a phase error of up to 90 degrees at the Nyquist frequency. Some subroutines are provided which give fractional-sample delays, both in the time and frequency domains. We do not implement them in the programs at present. There are admittedly many times when such considerations don't matter. It should be pointed out that for a real system like a loudspeaker, the delay may depend on frequency, and choosing it then may not be straightforward.

The second instance necessitating time delay correction is when the soundcard exhibits a time delay *between* the recorded channels. There may be a ½ sample delay between channels due to the practice of alternate sampling. We have noticed for one card that there was a 1-sample time shift, probably due to an indexing error. *It is imperative that the measurement system is given a complete checkout with loopthrough measurements, if accurate phase determinations are required*.

Obtaining the Frequency and Time Responses

In a modern measurement system, the computational cost of a discrete Fourier transform (DFT) is no longer an impediment, and we can simply take the ratio of the DFT spectra of the output and input of the DUT to determine the frequency-domain transfer function. The time domain is then obtained by an inverse DFT, similar to the approach for a dual-channel FFT analyzer. This ratio approach also means that the excitation does not need to have any particular character, but it must be spectrally dense, otherwise the resulting transfer function will be noisy at those frequencies where there is little energy. It is still wise to use arrays whose length is an integer power of 2, since such FFT algorithms are very fast.

The ratio approach can be used whether or not the input to the DUT, *ref*, has been recorded, since the system itself has already prepared the excitation signal. If we wish to use the excitation signal *sig* as the reference, and *dut* is the recorded output of the DUT, we would have

$$TF = \mathrm{DFT}(dut)/\mathrm{DFT}(sig) = DUT/SIG, \tag{4}$$

and it is understood that if phase is important, the appropriate time delay would have been applied to *sig* to align it with *dut*. Note that since the channel 1 *ref* signal was not involved, it could be used to simultaneously measure a second DUT. If on the other hand, the reference, *ref*, has also been recorded, then the frequency-domain transfer function, *TF2*, will be given by

$$TF2 = \mathrm{DFT}(dut)/\mathrm{DFT}(ref) = DUT/REF, \tag{5}$$

where *italic* capital letters are used to denote frequency-domain quantities. If we wish to measure the acoustic transfer function of a loudspeaker, and need to know accurately the acoustic time-delay gap between applying the excitation and the first arrival of the microphone signal, then we must use *TF2* with the recorded *ref*, since *TF* uses *sig* as a reference and it is confounded by the unknown and usually variable record-play delay. If the acoustic delay is not needed, then *TF* may be a better choice.

These two definitions of *TF* each have their strength and weakness. Eq.5, using the recorded reference, has essentially perfect time alignment and amplitude accuracy. The phase response is automatically properly registered. The two channels have no relative time error, and their gains are usually the same within 0.1 dB. The main problem with this method is that the

AA filter will seriously reduce the *REF* response near the Nyquist frequency, so that the transfer function will be noisier there. Of course we may not be interested in response above the AA filter cutoff, so often we simply don't display it. The *TF* of Eq.4 uses the computer data, *sig*, which can have response right up to the Nyquist frequency, if desired. This avoids any problems there, but now we have the responsibility to ascertain the required time delay between *sig* and *dut*. This is done in the program with a crosscorrelation, to the nearest sample.

For either ratio method, the character of the sweep is less important, and the excitation can be chosen to optimize other aspects of the measurement. Wideband music signals could even be used to exercise the DUT more naturally, while determining the response. Our measurement system then becomes reminiscent of a dual-channel FFT analyzer, in which the transfer function of the DUT is derived from the signals at its input and output in normal use. The main difference is that our measurement signals are generally longer and may be zero-padded, with the data acquisition so arranged to capture all of the reverberant as well as the direct response. This will allow proper studies of room acoustics.

Our ratio approach is in the frequency domain, but of course it relates to the time domain by the Fourier transform. A transform does not generate new information; it simply allows information to be viewed in different domains. To calculate the time-domain impulse response, *h*, we have

$$h = \mathrm{IDFT}(TF), \tag{6}$$

where the IDFT operation is the inverse discrete Fourier transform. In order for *h* to be a real time function, the frequency domain transfer function *TF* must be conjugate even. Those readers with some knowledge of complex numbers will notice this in the computer programs. The DFT of a time sequence of N samples (N even) will have N/2+1 frequencies from DC to the Nyquist frequency, fs/2. Excluding DC and Nyquist, the remaining N/2−1 frequencies are complex conjugate mirrors of the first set. We often think of these as negative frequencies, and speak of a double-sided frequency domain, with DC in the middle. If N is odd, there will be no frequency bin at Nyquist. It's a pitiful shame, but since Octave and Matlab do not allow an array index to be zero, the DC frequency index is 1, and for the Nyquist frequency, N/2+1. How much more elegant it would have been if these indices were 0 and N/2!

It is also possible to determine the impulse response directly from the time domain data. An inverse Filter approach [7, 9] computes *h* in the time domain. Although these approaches appear to be quite different, the two methods are really the same [5].

Characterizing Your Soundcard

In our programs a variable *sig_frac* is a dimensionless number between full-scale values −1 and +1, scaling the digital samples sent out to the DAC. We call it D_O in what follows below. Let us characterize the ADCs and DACs of our soundcard with sensitivities S_{ADC} (voltage input for full-scale digital acquisition) and S_{DAC} (voltage output for full-scale digital requisition), both with units V/#. Typically these sensitivities are about 1.2 volts peak for digital full-scale, but the input channel ADCs may also have preamps with adjustable gain. The SoundCardSetup.m program can create a full-scale output tone, V_{MAX}, which when measured with a calibrated voltmeter, defines S_{DAC}. If the meter measures RMS voltage, the peak voltage is larger by √2. Thus $S_{DAC}=\sqrt{2}\times V_{MAX}(\mathrm{rms})$, as long as the output level control has been set to maximum It should be left there, and the output level should be changed by altering D_O.

To obtain the ADC calibration, S_{ADC}, we can employ a loopthrough measurement. If D_O is the digital output level, the analog output voltage (and thus the looped ADC input voltage) is $D_O\times S_{DAC}$, and so the measured digital ADC input level, D_I, will be $D_O\times S_{DAC}/S_{ADC}$. The ratio of the digital signals gives

$$D_I/D_O = S_{DAC}/S_{ADC}. \tag{7}$$

Since we have measured S_{DAC} already, the measured digital loop gain or transfer function amplitude determines S_{ADC}. Entering proper values for the two sensitivities will ensure a calibration loopthrough gain of unity. This completes the characterization of the soundcard. If there is a preamp potentiometer or switch, you will need to ascertain the precise factor for any gain settings which you plan to use. To make the D_I and D_O digital data apply to actual electronic or acoustic data, the sensitivities defined above will turn them into real voltages, and the relationship between these will need calibration factors, to be discussed presently.

You will have to measure your own soundcard, but we have measured a few popular ones for Windows 7 using the unbalanced output (half of the output if it's balanced). For the Bearinger UCA202/222, $S_{DAC}=-1.22$V/#, and $S_{ADC}=-1.44$V/#. For the ART USB Dual Pre, $S_{DAC}=+1.05$V/# and $S_{ADC}=+1.49$V/#, using TRS inputs with potentiometer set to minimum. We tried 5 or 6 Dual Pre's and the values can differ by 10% between them. The soundcards were measured under Windows 7 with the recording gain set to 4, which is the recommendation in the brochure and suggested on the web. At a record gain of

3, the ADC can saturate before reaching digital full scale, not a recommended setting. A Focusrite Scarlett 2i2 had S_{DAC}=+1.59V/#, and S_{ADC}=+1.16V/# for the line inputs at 12:00 o'clock pot setting. Minimum gain is −23dB and maximum gain is +27dB, for a total gain range of 50dB. The instrument input has 9dB more gain than the line input setting, and the XLR mic input has 10dB gain above that. This superb soundcard was earlier characterized as having an inverting output. That is not true, both its ADCs and DACs are properly noninverting.

It's worth pointing out that even if a measurement system displays noninverting loopthrough behaviour, it is possible that both the ADCs and DACs are polarity inverting. The Bearinger UCA202/222 falls into this category! Do designers not concern themselves with polarity issues at all? Polarity effects are subtle, but can be heard on speech signals. By connecting a +1.5 volt battery signal to the ADC just after it has started recording, you can easily check the polarity of the digital response. To correct the soundcard behaviour for the measurement system, we can assign negative S values to those channels that are inverting, as we have done above.

Calibration

This section is very specific to an accurate calibration of a measurement system. Although it can be skipped over for now, you may be more receptive to read it when you have a working system.

There are *two major categories* of calibration used in electroacoustic measuring systems. A ***level-independent calibration*** would make the measurement independent of input level, by dividing the measured output by the chosen input, which we call a transfer function. Different signal levels would give the same measured transfer-function response, except for effects of noise, nonlinearity or amplitude compression. This is usually what would be reported in a product review, for example. For systems that are essentially linear, a level-independent calibration seems appropriate. If the two channels of a soundcard are used to measure both the input and output of the DUT, then this form of calibration is most natural, and we will favour it in our programs.

The second category is ***level-dependent calibration***, in which the intended measurement must reflect the actual SPL or output at which the unit is operated. Now there is no transfer function; the measurement consists of the spectrum of the measured *dut* signal, with a different calibration factor. If we were to run a series of measurements on a loudspeaker at different levels, the plots would increase with level to show a parallel family of curves. This is useful if the net SPL or the response change versus level is the intended aim of the measurement, and loudspeakers can fall into this group. It might also be used to measure the gain and response of an amplifier when driven into nonlinearity, using stepped input levels with a constant calibration factor. The type of calibration that you use may say something about how you think…

Consider a loudspeaker measurement using the former, *level-independent* calibration. We need to include the characteristics of the actual loop components that are used, such as the gain of the power amplifier, the mic sensitivity, and the mic preamp gain. Let us call H_{SPKR} the desired response of the loudspeaker in Pa/V (voltage applied to the actual loudspeaker terminals), and S_{MIC} the sensitivity of the microphone in V/Pa. Note that $D_O \times S_{DAC}$ is the output voltage of the soundcard, which is also the reference channel input and the input voltage to the DUT, and that $D_I \times S_{ADC}$ is the measured DUT output voltage (but is *input* to the ADC). We thus have

$$D_I \times S_{ADC} = D_O \times S_{DAC} \times G_{PWR\_AMP}[V/V] \times H_{SPKR}[Pa/V] \times S_{MIC}[V/Pa] \times G_{MIC\_PREAMP}[V/V]. \quad (8)$$

Therefore the *level-independent* Loudspeaker Response, H_{SPKR}, in Pa/V, that we are trying to measure becomes

$$H_{SPKR} = \{1/(G_{PWR\_AMP} \times S_{MIC} \times G_{MIC\_PREAMP})\} \times (D_I \times S_{ADC}/D_O \times S_{DAC}). \quad (9)$$

This immediately tells us that to measure the response, we need to multiply the input-output voltage ratio by the calibration factor of Eq.9 in curly brackets.

We commonly want the sound pressure level in dB relative to p_0=20 μPa, which is the reference for 0 dB SPL. The *level-independent* sound pressure level SPL_I for *1 volt input to the loudspeaker* is

$$SPL_I = 20 \times log10[\{1/(p_0 \times G_{PWR\_AMP} \times S_{MIC} \times G_{MIC\_PREAMP})\} \times (D_I \times S_{ADC}/D_O \times S_{DAC})]. \quad (10)$$

The calibration factor (which does not include the logarithmic factor), again in curly brackets, is to be multiplied by the *level-independent*, measured input-output voltage gain, $D_I \times S_{ADC}/D_O \times S_{DAC}$, allowing the transfer function plots to reflect the proper SPL levels for the loudspeaker. As an example, for a typical power amplifier gain of 30, a mic sensitivity of 12mV/Pa (0.012V/Pa), and a mic preamp gain of 40dB (100), the calibration factor that gives the loudspeaker SPL response *for 1V of input into the loudspeaker* is CF=1/(0.00002×30×0.012×100)=1388.9, and this will add 62.85dB to the vertical log axis of the SPL plots.

For loudspeakers, a common specification is the SPL output (usually at 1m on the tweeter axis) *for a loudspeaker voltage input of 2.83V*, representing 1 watt into a nominal 8Ω. The calibration factor becomes *CF*=2.83/(0.00002×30×0.012×100). The measuring program plots the transfer function with about 10dB of positive headroom, having a total range of typically 80dB. When a *CF* is entered, the plots will automatically adapt to scale to the nearest 10dB or so. You can modify the axis command for the relevant plot if you wish a different view. Note that with *level-independent* calibration, the SPL response *does not increase* with input level. The input level is chosen to give normal operation and reduce the effects of noise.

The second category of *level-dependent* calibration *leaves out the terms that relate to the output side factors of the soundcard*, G_{PWR_AMP}, S_{DAC}, and D_O. . The measurement program would then use only the DFT of the *dut* signal, which of course is still proportional to the output level. Now there is no transfer function to calculate; the measurement simply gauges the strength of the *dut* signal. The equivalent to Eq.10 gives the level-dependent SPL_D as

$$SPL_D = 20 \times log10[\{1/(p_0 \times S_{MIC} \times G_{MIC\_PREAMP})\} \times (D_I \times S_{ADC})], \quad (11)$$

which *does depend on the excitation level* to the DUT, since the response is proportional to the system excitation, D_I. The *level-dependent* calibration factor is shown in curly brackets in Eq.11. Note that we have left out the power amplifier gain as well, since the focus is now on just the output SPL for the loudspeaker, given the appropriate input. This form of calibration is useful for the measurement program when we arrange to provide a series of voltage inputs to the DUT, given by stepped values of $S_{DAC} \times D_O \times G_{PWR\_AMP}$. These voltage inputs to the DUT are usually intended to have flat spectra, such as a logsweep or an MLS, but they may have spectral character such as power testing noise.

There are two approaches to calibrating your own microphone. If you have access to a calibrated microphone, then measuring the response of a good loudspeaker with each microphone will allow you to calculate a correction curve as a secondary calibration. The other approach is to use a calibrator, which will determine the microphone output for a known acoustic level. Calibrators typically give a 1 kHz tone at 94 dB SPL, which represents 1 pascal rms, or 114 dB SPL, which is 10 pascal rms. Although the calibrator measures the mic at only one frequency, small omnidirectional microphones are usually quite flat, making the responses versus frequency meaningful. The SoundCardSetup.m program of Part 2 can be used for this purpose. It is important to realize that the soundcard level controls must be left undisturbed to maintain the calibration during a measurement.

Soundcard Preamp Gain Controls

Many soundcards have pesky input preamp gain controls. These small knobs often cover a gain range of 40dB or more, and not very uniformly at that. As an example, consider the ART USB Dual Pre unit, an inexpensive but quite capable 2-channel soundcard. Its gain controls are reverse log potentiometers that change the gain of a balanced mic preamp having an instrumentation amplifier configuration. Most soundcards are probably the same. There are only two connections to the 10kΩ preamp potentiometer, whose resistance R determines the gain, G_{MIC_PREAMP} by

$$G_{MIC\_PREAMP} = [2*R_{FDBK}/(R_0+R)]+1, \quad (12)$$

where R_{FDBK} are the feedback resistors of the input opamps (~5kΩ for the USB Dual Pre) and R_0 is the internal lower limiting gain resistor (~50Ω). The nominal minimum gain is ~2, and the maximum is ~200, for a 40dB range. For potentiometer resistance values of 9950, 1831, 476, 111 and 0 ohms, we would have gains of 0, +10, +20, +30, and +40 dB. We chose to use a centre-off 3-position toggle switch *to parallel the 10k potentiometer* with resistors of 2240, infinity (centre-off), and 500 ohms, giving gains of +10, 0, and +20 dB at minimum pot setting. The maximum gain for each switch position is still +40dB. The *ref* channel may be left unmodified, but calibrated gains greatly benefit the *dut* channel.

Nonlinearity

It is inevitable that sufficient signal excitation will cause some nonlinearity. Depending on the type of measurement to be performed, one measurement technique may seem to be better than another, but they may all give different results if the excitations are different. A logarithmic sweep allows the simultaneous separation of the harmonic distortion in the time domain. This is because the instantaneous frequency of the sweep changes by the same relative amount in any particular time interval, independent of frequency. One of the programs, Logsweep1hd.m, computes the transfer function along with the spectra of the second and third harmonics. A totally different approach is used in the soundcard setup program, which among others can use a fixed-frequency signal and does an FFT to show harmonic distortion for an overdriven device. Intermodulation distortion may be addressed in later programs.

Software Example Programs

A number of programs are given on the website, and in due course more may appear. Part 2 introduces the setup program with its many features. We here give some details of **Logsweep1quasi.m**, which is intended to measure a quasi-anechoic loudspeaker transfer function in a normal room. Since there are a lot of reflections from objects in the room, the program allows the measured impulse response to be edited to remove the reflections, so that the response closely approximates that which would occur in a real anechoic chamber. It is presented with commented lines that will help the understanding of each operation. Basic concepts have already been presented above. The program is fairly verbose in its plotting of intermediate data, which can be commented out later when you get the hang of things. This helps to visualize the various views of the data as the program wends its way through, (0) preliminaries and settings, (1) calculating the logsweep, (2) data-gathering, (3) determining record/play delay, (4) calculating a transfer function with room reflections, (5) getting the previous TF and comparing them, (6) calculating the associated impulse response, (7) editing it to remove reflections, and finally (8) obtaining the quasi-anechoic transfer function response. The first 5 or 6 steps are essentially the same in each of the logsweep programs. The user is encouraged to run the program to see how the steps are accomplished.

We illustrate the program in Figure 2, which shows a screen shot of four consecutive labelled plots produced during operation. Top left shows the computer-generated logsweep and a windowed version which tapers down the excitation at both low and high frequencies. Top right shows the looped recorded *ref* signal in blue, and the red *dut* response from the microphone near the loudspeaker. The lower left shows the crosscorrelation between the excitation *sig* and the recorded *ref*, indicating a record-play delay of about 0.3 seconds. The final lower right plot shows the transfer function derived from the DFT of *dut* divided by the DFT of an unwindowed version of *sig*. Room reflections cause the response to display many detailed resonances.

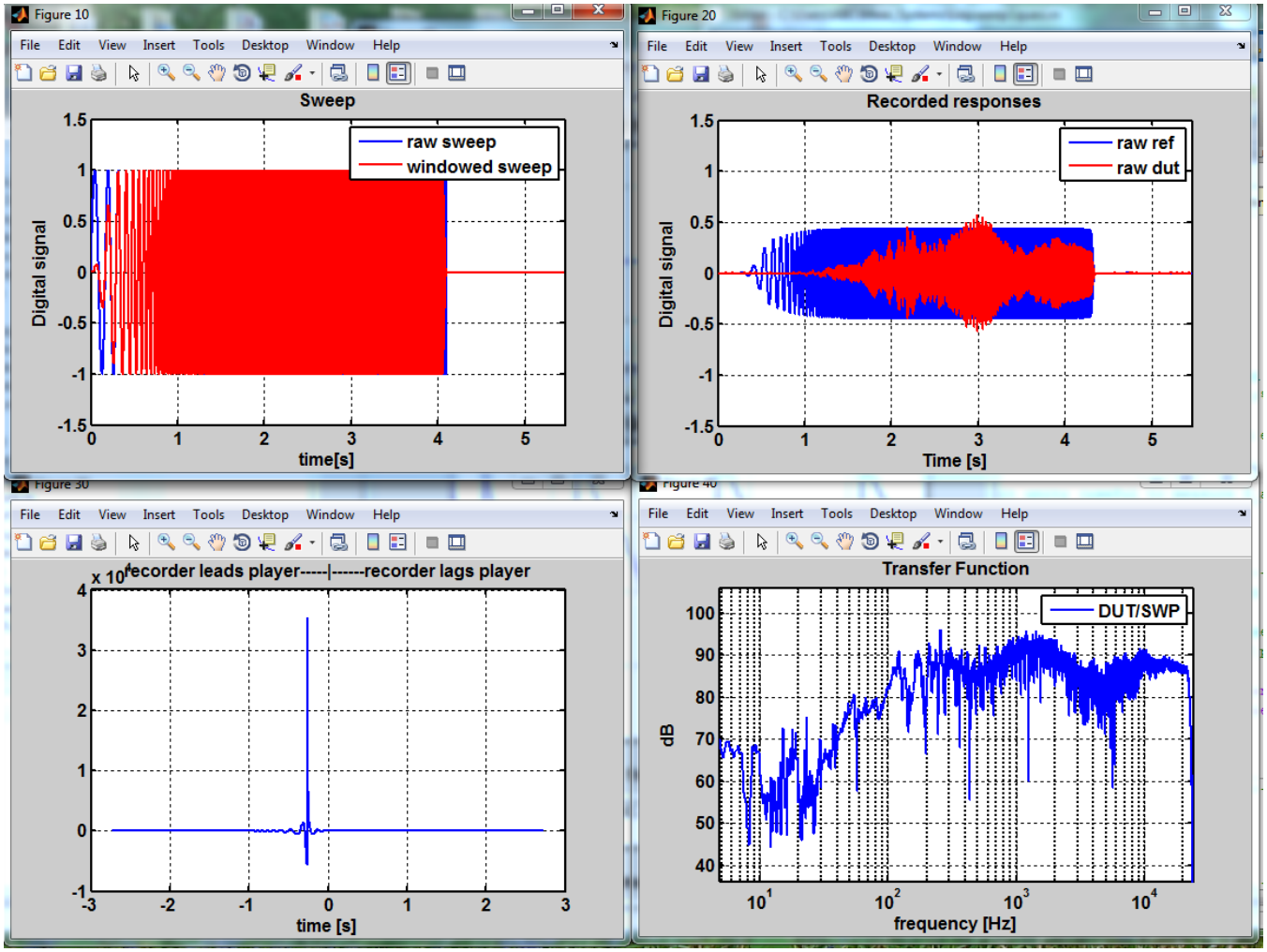

Figure 2. Screen shot of four plots from program operation. Top left: computer generated logsweep. Top right: recorded *ref* and *dut* digital signals. Lower left: crosscorrelation of *sig* and *ref*. Lower right: frequency response of the loudspeaker in a

room. The very ragged response is due to a myriad of reflections from room surfaces. Increased damping reduces the effect of reflections at higher frequencies.

Some tips on running the programs are in order. If the length of the recording file and sampling frequency are not appropriate, a multiple message "INSUFFICIENT TIME CLEARANCE" will be sent to the command window. It means that the record-play delay exceeds the allowed time clearance, which is printed onscreen soon after the program starts. You must then either reduce the sampling frequency, or increase the index defining the power-of-2 size of the files. By changing the constants in the first part of the programs, such as *sig_frac*, you can also prevent overload and modify the environment.

Quasi-anechoic measurements have a number of limitations. The loudspeaker is normally placed on a stool about halfway between ceiling and floor, with the microphone on the measuring axis as close as practical, perhaps 1m away. Even closer distances of 50 or 40cm are not unreasonable. This reduces the amplitude of the reflections relative to the direct sound. If the speaker is placed at height $h/2$ above the floor in a room of height h, and the microphone is a distance d away, the first reflections come in a time $(\sqrt{(h^2+d^2)}-d)/c$ after the direct sound, where c is the speed of sound. For a typical 8-foot ceiling (2.44m), 50 cm distance gives a reflection-free time, τ, of about 5.8 msec. If we truncate the impulse response after a time τ, the response will be smeared and not reliable below frequency $1/\tau$, or 170 Hz in our example. However, we have indeed removed the room reflections! If the impulse response is not near zero at the truncation point, the discontinuity may compromise the results at higher frequencies a bit. To avoid this, the data could be tapered down using a "window" function. We will leave such enhancements to the reader, and introduce the next program plot, which requires user interaction.

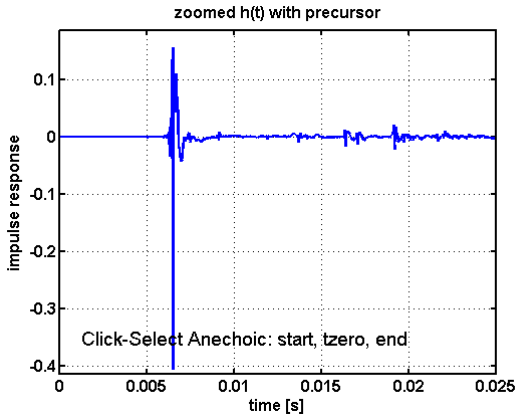

Figure 3. A zoomed portion of the room impulse response, showing some strong reflections from room boundaries. There are often some acausal wiggles before the impulse at "t=0", and room reflections come in about 7 ms after the main impulse. The user must click on the plot at three different axis times to finalize the quasi-anechoic calculation.

The program pauses to display a zoomed portion of the loudspeaker impulse response, figure 3, and requires three clicks from the mouse. The plot may display some acausal wiggles that result from the method of computing the impulse response and/or the AA filter. These should be included in the impulse response, wrapped in periodic time. Three time values are selected on the plot: the first should be such as to capture all the acausal wiggles, the second should be *time zero* where the impulse response rises sharply, and the third must be chosen so that room reflections are removed beyond it. The program then recomputes the *reflection-free* transfer function and displays both the phase and magnitude quasi-anechoic responses. Figure 4 shows the transfer function magnitude.

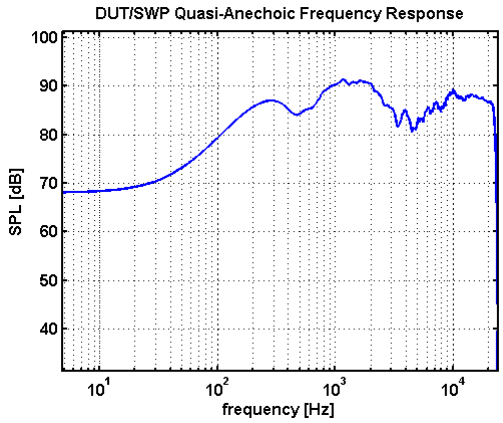

Figure 4. The quasi-anechoic response of the loudspeaker with the reflections suppressed. The resolution of these data is about 150 Hz, and frequencies below 200 Hz are not trustworthy. Above 200 Hz the response would be very much the same if taken in a real anechoic chamber.

Truncation of the impulse response smears the transfer function and misrepresents the lowest frequencies, which have not rung down after 7 ms. What can we do to retrieve these data? If we were to allow a much longer segment of the impulse response to be analyzed, these frequencies would be included, but along with their confounding room reflections. In order to obtain a fairly good "anechoic" measurement at the lowest frequencies, we can use a nearfield technique, placing the microphone near the woofer dustcap. This will reduce most room reflections to insignificance without the need to truncate the impulse response, but it may falsify that response for several reasons. Firstly, a port or auxiliary bass radiator may also be part of the loudspeaker configuration. Sometimes we can do two nearfield measurements and sum these responses to produce a tolerable overall response, but the addition needs to take into account their relative sizes. Secondly, the cabinet itself displays a diffractive signature that will not be properly captured until we are several box dimensions from it. Nonetheless, it is often possible to get meaningful nearfield results and "stitch" them onto those measured quasi-anechoically.

Another program, **Logsweep1rt.m,** calculates room reverberation time, RT, using a different microphone placement than above. There is a larger distance to the microphone, thus making the reverberation more prominent. Its data-gathering part is the same as Logsweep1quasi.m, and the program then calculates an acoustic room parameter, clarity, from the unfiltered impulse response. C50 (or C80), is the ratio of the early impulse response energy before 50 (or 80) milliseconds, compared to the remaining late energy, expressed in decibels. For our close microphone distance, C50 is about 40dB, but in a real concert hall with the microphone well back, it would be closer to 0dB. The impulse response with all the room reflections is used to compute the reverberation time. The program does a reverse integration of the square of the impulse response, which is the energy decay. If $h[n]$ are the samples of the impulse response, the decay of the energy, $D[n]$, in the room is given by the wonderfully-compact Octave/Matlab statement,

$$D[n] = \text{flip}(\text{cumsum}(\text{flip}(h^2[n]))); \qquad (13)$$

As the program nears completion, the user is prompted to enter the centre frequency of the octave band that is to be analyzed (you might have to position the cursor in the command window for this), and then the decay of the reverberant energy is plotted. After clicking with the mouse on two points of the plot where the decay is fairly straight, the reverberation time is calculated. Although the decay of the energy is usually much less than 60dB, the reverberation time is always scaled to represent a full decay of 60dB. Note that the final plot is shifted in position on the computer screen from the others. You can use that code to position plots to your own taste.

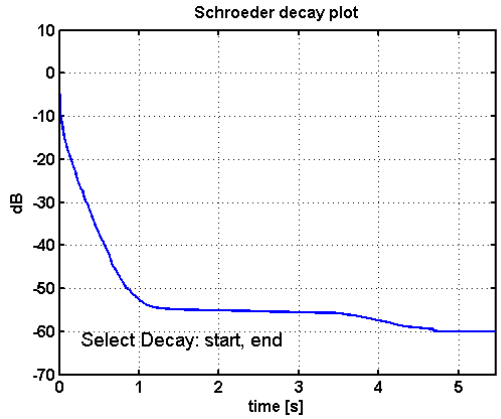

Figure 5. A plot of the room energy decay versus time, filtered in a 1-octave band centered at 1 kHz. The first part of the decay gives a reverberation time RT of about 1.2 seconds. The curvature of the plot is due to an open doorway to a very reverberant hall, resulting in the changing slope. The late decay and the horizontal portion of the plot are due to noise during the measurement. The program cuts off decay below−60dB.

A final program, **Logsweep1hd.m**, computes the linear transfer function of a system together with spectra of the second and third harmonic distortion. It is intended for loudspeakers, and typically the microphone should be placed very close to the speaker, to minimize the reverberation of the room, while capturing all the relevant loudspeaker distortion details. The logsweep has a clean sinusoidal waveform, and the distortion will be solely harmonic, being captured by the microphone together with the linear response. Since the logsweep covers each octave in the same period, the second harmonic will be visible in the recovered impulse response as a relatively clean pulse, *advanced* by the time it took for the sweep to cover a factor of two (1 octave) in frequency. The advance occurs because when the logsweep is at frequency *f*, the second harmonic is already at frequency 2*f*, so its impulse will be recovered early. Similarly the third and higher harmonics will also be separated out early at the respective times that the logsweep took to cover that frequency interval. Going back to the original design for the logsweep, Eq.2, we can show that the n^{th} harmonic will appear at negative times [5], (wrapped around from *t=0*) given by

$$\tau_n = -L\ log(n). \qquad (14)$$

The harmonic impulse responses must be extricated from the total impulse. Each one must include the acausal wiggles that are caused by the AA filter and/or measurement artifacts, and the clean part that follows, without including bits from the other harmonics. The higher harmonic impulses are shorter and therefore more difficult to extract due to the compressed time scale from Eq.14 as *n* increases. The program is tricky, but careful reading of the lines and comments will guide the reader through. Each harmonic spectrum, derived from a DFT of the corresponding impulse response, is plotted with frequency axis *f/n*, since each value of the harmonic spectrum at frequency *nf* should be plotted at frequency *f* which generated that harmonic. That just shifts each harmonic spectrum along the logarithmic frequency axis by the appropriate amount.

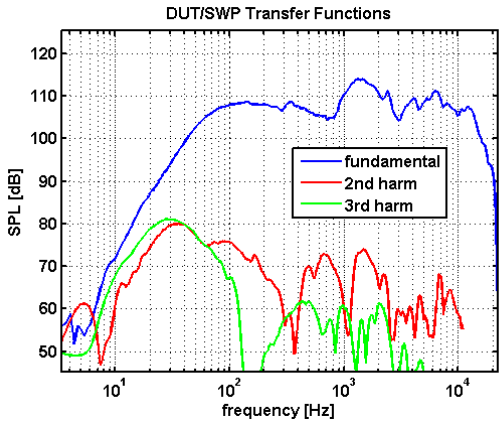

Figure 6. Showing the linear and distortion responses of a Tannoy dual-concentric loudspeaker driven with 5Vrms. The second and third harmonic spectra are derived from separated spikes in the impulse response of the system. A logarithmic sweep allows such separation in the time domain. The harmonic distortion is particularly strong at the lowest frequencies where the cone excursion becomes large. The data has been 1/10-octave filtered.

The reader is urged to modify the programs to personalize the plots, change the parameters, and add new twists of their own. We hope to submit more programs to the website in future, and will try to be responsive to queries and suggestions.

Conclusion

Many soundcards can be made the heart of a reasonably capable general purpose electroacoustic measurement system. Data can be further manipulated in software to make the analysis as comprehensive as is desired. The hope is that this open-source approach will come into more common use so that a community of aficionados can share data and produce further enhancements. The paper has shown some general measurement system theory, and presented programs for several types of measurement. Part 2 has a description of the very important setup program, and shows more features of typical measurements.

Acknowledgements

We thank Sean Thomson of the Steyning Research Establishment of Bowers and Wilkins for many discussions relating to measurement systems, and as a joint author on related aspects [5]. Our colleague Kevin Krauel has been involved in numerous deliberations during the progress of this work.

References

[1] R. C. Heyser, “Acoustical Measurements by Time Delay Spectrometry”, *JAES* Vol. 15, pp.370-382, October 1967.

[2] J. Vanderkooy, “A Digital Domain Listening Test for High Resolution”, presented at 129th AES Convention, San Francisco, CA, USA, Nov. 4-7, 2010. Paper 8203.

[3] J. Vanderkooy, “Another Approach to Time-Delay Spectrometry”, *JAES* Vol. 34, pp.523-538, July 1986.

[4] H. Biering and O. Z. Pedersen, "System Analysis and Time Delay Spectrometry," Pt. I, Bruel & Kjaer Tech. Rev., no. 1, pp. 3-51 (1983); Pt. II, ibid., no. 2, pp. 3-50 (1983).

[5] John Vanderkooy and Sean.P.I. Thomson, “Harmonic Distortion Measurement for Nonlinear System Identification”, presented at 140th AES Convention, Paris France, June 4-7, 2016. Paper 9497.

[6] S. Norcross and J. Vanderkooy, “A Survey of the Effects of Nonlinearity on Various Types of Transfer-Function Measurements”, presented at 99th AES Convention, New York, NY, Oct. 6-9, 1995. Paper 4137.

[7] A. Farina, “Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique”, presented at 108th AES Convention, Paris, France, Feb. 19-22, 2000. Paper 5093.

[8] A. Novak, Ph.D dissertation, *Identification of Nonlinear Systems in Acoustics*, Université du Maine, France, 2009.

[9] A. Novak, L. Simon, F. Kadlec and P. Lotton, “Nonlinear System Identification Using Exponential Swept-Sine Signal”, *IEEE Trans. Instrum. & Meas*. Vol. 59, pp. 2220-2229, 2010.

[10] D.Rife and J. Vanderkooy, "Transfer-Function Measurements with Maximum-Length Sequences", *JAES* Vol. 37, pp. 419-444 (1989 June)

[11] J. Vanderkooy, “Aspects of MLS Measuring Systems”, *JAES* Vol. 42, pp.219-231, April 1994.

[12] S. Müller and P. Massarani, “Transfer-Function Measurements with Sweeps”, *JAES* Vol. 49, pp.443-471, June 2001. A longer version of this paper is also available.