Design and Measurement of a Dipole Microphone (online section)

Mark Williamsen

5/17/09

submitted to Audio Amateur, Inc.
for audioXpress Magazine

This is the online section of the article "Design and Measurement Of a Dipole Microphone," which goes into detail describing the gated measurement technique [A] and software programs used to measure the dipole microphone prototype. For discussion of the microphone's design and construction, please refer to the print section, which can be found in the June 2009 issue of audioXpress.

To carry out the measurements, I used 16 bit samples with no compression, at 44100 samples per second. Almost any sound card should be able to simultaneously play and record two channels in this format. You'll also need a software program to interact with your computer's sound hardware that can play and record at the same time. I chose Audacity <audacity.sourceforge.net> because it's free, and can translate a variety of sound file formats. You'll want to stick with linear PCM encoding, and avoid compressed file formats for signal generation and analysis because they are likely to discard details, especially in the phase response, which you actually need. Software programs that keep digital audio in memory in a compressed format, such as GarageBand, are also ruled out for the same reason.

In my work I only used one of the output channels for test signals, but there are some neat things you can do with two output channels. The second channel could contain synch pulses to trigger a digital oscilloscope, or drive a turntable for polar

measurements. Or if you are using a two-way loudspeaker as your point source (drivers should be coaxial, of course) you can bi-amp it with a stereo amplifier, and then put signals meant for the woofer in one channel and signals meant for the tweeter in the other. This completely avoids the problem of phase cancellation between drivers when using a passive crossover. Of course, you would need to modify the analysis software program to account for two different time delays. I should mention that it's a bad idea to use a cabinet with a tuned port, passive radiator, or multiple woofers as your point source.

In my case, I didn't need to bi-amp the system because I had on hand a single-driver wide-range loudspeaker—the Auratone 5C Super Sound Cube—to use as my point source. The sealed cabinet is 16.5cm wide X 16.5cm high X 13.3cm deep, and contains a single 13.3 cm paper cone driver. Resonance in the cabinet is 138 Hz, so 100 Hz is probably a reasonable lower limit for measurements. High frequency output is usable up to 10 kHz due a small light-weight voice coil with a rigid dust cap. All of my measurements will be tone bursts with a duty cycle of around 1%, so response changes due to voice coil heating won't be a problem. It was very convenient to support the cabinet on a mic stand using an upside-down threaded base I had on hand (*Photo 1*). A larger cabinet would need to be supported on a structure with a minimum of reflecting surfaces, such as wire rack shelving.

You're probably thinking that Auratones don't have a flat frequency response, so how can I use them to measure a microphone's response? While it's true that Auratones don't have a flat frequency response, what they do have is a smooth response, and for these experiments that's all you need. I'll use the omni response of the electret microphone capsules as a reference for comparison. Even that's not really flat above 3 kHz for the capsules I chose, but it's good enough. Readers having a calibrated standard microphone can use that as a reference instead. Since the computer sound hardware can record two channels simultaneously, in many cases you'll be able to record both reference and measured signals at the same time, allowing precise comparison of both magnitude and phase, for instance when comparing the omni and gradient signals.

For polar measurements I set the mike stand supporting the microphone under test on a cheap turntable and turned it by hand 5° after each burst. In this case, all bursts were at the same frequency. Because I was kneeling on the floor while turning the mic stand, it is likely that I became part of the measurement (although, I hope, a small part). For frequency response measurements I left the microphone in a fixed position while the frequency of each burst was varied logarithmically from 100 Hz to 10 kHz. This gives you the luxury of having an equal number of measured data points per octave, a benefit which is not always

available with more sophisticated techniques such as Maximum Length Sequence (MLS) or the Fast Fourier Transform (FFT) [B].

The test signal itself may seem trivial, but there are some subtleties I'd like to bring to your attention. Interested readers can follow along in the source code files tbg.cpp and toneBurst.cpp, skipping over the portion that writes out header information identifying the file format and contents. Polar response and frequency response are really the same measurement, just varying different parameters.

While generating each frequency, I check to see if burst length has fallen beneath a lower limit, which I arbitrarily set to 100 samples. If it has, then I add another cycle to the burst, so it is never less than 2.3msec in length.

Starting at low frequencies, you can only send one cycle, which runs quite a bit over the free-field time limit of 4.7 msec. When you reach 212 Hz, you are within the free-field limit, and by 441 Hz can start adding cycles. By the time you get to 10 kHz, there are 23 cycles in each burst. Each time you add a cycle there is a discontinuity in the measurement, which could appear as an artifact in our results. Luckily, this is a small effect as long as you are near steady-state conditions.

Now that I have the number of cycles for a given burst, I go back and adjust the frequency a little so that the burst can start and end on even samples. Since each burst is always at least 100 samples, this frequency shift is never more than 1%. This is meant to make it easier to implement the matched filter used when analyzing the microphone response. Now you can calculate the sine values needed to populate the wave file, but there is a little problem. If you just launch into a simple sine wave there will discontinuities where the burst begins and ends, when you ask the loudspeaker diaphragm to suddenly go from zero to maximum velocity at the first zero crossing, and from maximum to zero velocity at the last zero crossing (see Figure A) [C]. A simple solution is to replace the sine wave with a raised cosine which is continuous at the start and end, but now you have a DC bump during the burst which will upset the measurement. The final step is to add a harmonic of the raised cosine which is equal in amplitude, but opposite in phase to the fundamental.

This cancels the DC bump, leaving a net average DC value of zero during each burst. Each cycle begins and ends at zero as well, so any number of cycles can be strung together to form a burst of the desired length. The harmonic does get sent to the loudspeaker and reaches the microphone under test, but is removed in analysis by a matched filter, as you shall see. You now have a band limited test signal that can be reasonably reproduced by a single driver loudspeaker over two decades of frequencies, with a

minimum of measurement artifacts. Readers who are interested in absolute phase should note that this test signal allows you to easily check absolute phase of microphones, regardless of the acoustic time delay between speaker and microphone. The generated test signal file now contains a series of harmonically shaped tone bursts separated by silence, occurring at precise intervals. I used a 1 Hz rate for polar response measurements so I would have enough time to rotate the turntable between bursts. For frequency sweep measurements, I found that a 2 Hz rate was slow enough for reverberation in my living room to die down after each burst. You may have to wait longer between bursts if testing in a reverberant environment, such as a basement, garage, or commercial space.

I used an old Sony receiver as the test amplifier, and found that it interfaced nicely with my computer's sound hardware with a simple patch cable. I used ordinary mic stands and adapters to support the loudspeaker and the microphone under test. The threads on the top end of the mic stands are 5/8"-27, which I couldn't find at any hardware store. Luckily, Atlas Sound <www.atlassound.com> has a complete selection of adapters and couplers that match these threads. The prototype microphone that I built has enough gain to connect directly to my computer's line input jack. For purchased microphones, however, I must use an outboard mixer with input connectors and gain stages appropriate to the microphone under test.

To make a measurement, I used Audacity to open the test signal wave file, and checked visually that the file contained harmonically shaped tone bursts. Be sure to select Audio Preferences to enable "Play other tracks while recording new one." If you don't make this setting, Audacity will just connect the inputs to the outputs causing a screaming feedback loop. Play back the test signal first to check levels so you can avoid clipping. The test signal should be clearly audible in the room at all frequencies, without being overwhelming. Of course you'll want to minimize ambient noise, but this isn't as big a problem as you might think since you'll be using a matched filter during analysis. Now rewind the file pointer and click "Record." When the measurement is complete, stop recording and save the result as a wave file. If you keep multiple tracks on screen at the same time, be sure all except the test signal are muted when making a measurement.

Just as the test signal wave file was generated off-line using a command line program, I will now analyze the measured signal wave file off-line using another command line program called "tba.exe." By using the same sample clock in full-duplex mode, you are assured that the stimulus and response files are perfectly synchronized, after you measure and take into account the acoustic time delay from loudspeaker to microphone. My computer's sound hardware has a consistent 1/2 sample delay

between channels when recording (but not when playing back) which I've subtracted from all measurement data shown in this article. This rather large error is not specified or even acknowledged by the vendor, although it's obviously built into their design. The message is that it's up to you to measure and account for anomalies in phase and frequency response of your sound hardware.

Following along again in the source code files tba.cpp and toneBurst.cpp, first you scan through the header information, which you can probably ignore unless something goes wrong. Next comes measurement data from the microphone under test, recorded in time domain. But what you really want is the phase and amplitude response for each burst, in frequency domain. First you have to find each burst in the data. Then you perform a single frequency discrete Fourier transform to obtain phase and amplitude. Formally, this is defined as [D]:

$$F(\omega_p) = \frac{1}{2N} \sum_{k=0}^{2N-1} f(t_k) e^{i\omega_p t_k} \qquad (1)$$

You fix ω_p at the frequency of interest and then sum up the products of a calculated sinusoid and the measured samples within the burst. The result is a complex number which gives amplitude and phase angle in the complex plane. Note that this uses all the data available in the burst, so we can respond more accurately to the desired signal while ignoring noise and interference. Burst duration, number of cycles, and exact frequency are all calculated as before when we generated the test signal file.

Similarly, the interval between bursts is fixed for both stimulus and response files, but now there is a delay consisting of at least two components. There is an acoustic delay due to the distance from source to microphone under test, which stays fixed when measuring frequency response. You can also arrange for this distance to be fixed when measuring polar response by setting up the turntable so that the acoustic center of the microphone is aligned directly above the center of rotation of the turntable. The other delay is due to latency in the playback and recording software, which can vary from one measurement to the next. I used Audacity's built-in zoom capability to measure the total delay on screen for each measurement. It turned out to be around 2000 samples or 45 msec for my system. I found that by using a simple meter stick to set the microphone position with respect to the loudspeaker, I was able to reliably set the acoustic delay within +/- one sample, or about 7 mm. The total delay may then be entered as a command line parameter so the analysis program can find each tone burst in the response signal file. Please realize that if you get the delay time wrong, you are effectively telling the analysis program to look for the bursts in the wrong place which will certainly scramble the results. For frequency sweeps, it's a good idea to look at the phase response to be sure it's not wrapping around, which would indicate an incorrect delay time in analysis.

When setting up your measurement system, it is important to be able to measure background noise and reverberation. This is easily done by waiting until the end of the interval between bursts, and then making a duplicate measurement during a time when the room should be quiet, just before the next burst. With a half second or more of delay between bursts, I found that the noise floor in my living room was 30 dB below the signal at 100 Hz, improving to 50 dB or better above 1kHz without averaging. To improve the signal-to-noise ratio for low frequency measurements, I sent each tone burst multiple times, and then took an average. The desired signal will add coherently, while random noise will add incoherently, pushing down the noise floor by a factor of the square root of N, where N is the number of bursts in the average. I had good results with $N = 10$, improving the signal to noise ratio by a factor of 3.16, or 10 dB. You can take the average in time domain, or in frequency domain as shown in the source code. Of course, you should use averaging only when necessary, because it will increase test time by a factor of N. These features are already implemented in tbg.exe and tba.exe, but you'll have to be sure that the command line arguments match, between signal source and response files.

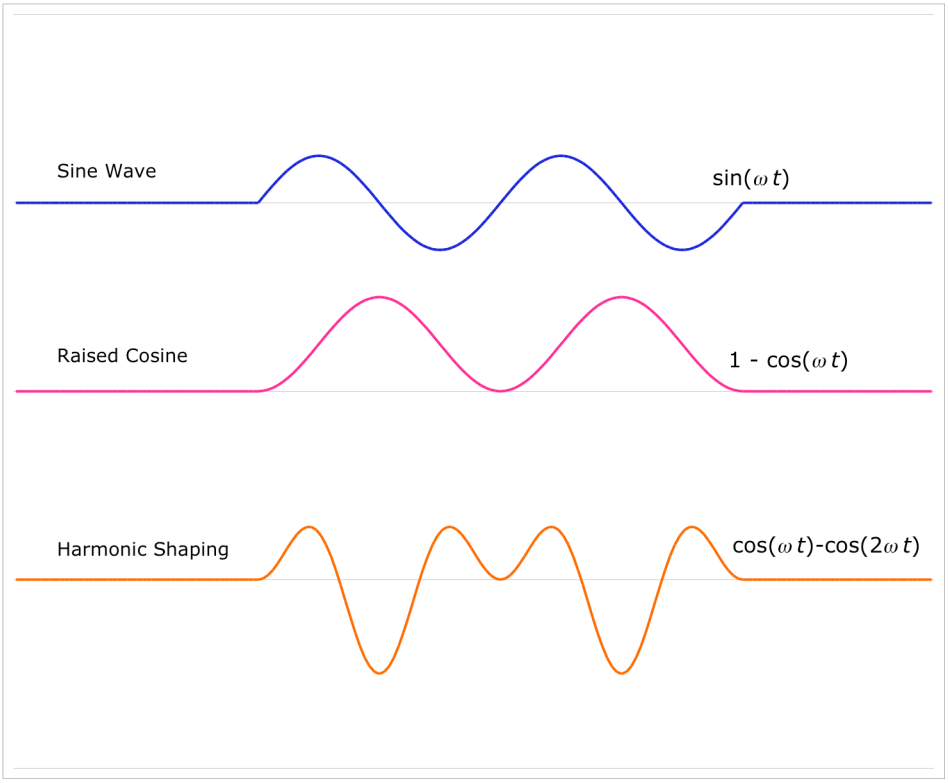

Figure A. Simple sine tone burst shown above has discontinuous slope at the start and end. Raised cosine has continuous slope, but gives a DC bump during each tone burst. Adding in the second harmonic with equal amplitude and opposite polarity corrects the DC bump, and gives a continuous band-limited function that can be repeated as necessary to fill a tone burst. Matched filter removes the harmonic during analysis.

Photo 1. Auratone 5C Super Sound Cube loudspeaker mounted on top of a tripod-style mic stand. Dipole microphone prototype is supported by a mic stand sitting on a cheap turntable. Speaker and microphone are positioned 1 m apart, mid-way between floor and ceiling, near the middle of the room.

References

[A] Henning Møller and Carsten Thomsen, "Electroacoustic free-field measurements in ordinary rooms——using gating techniques," *Brüel & Kjær Application Note 17-196* (1975).

[B] J.A. D'Appolito, "Testing Loudspeakers: Which Measurements Matter," *audioXpress,* Vol. 39, No. 9, pp. 17-25 (Sept. 2008).

[C] S.H. Linkwitz, "Shaped Tone-Burst Testing," *Journal of the Audio Engineering Society,* Vol. 28, No. 4, pp. 250-258 (April 1980).

[D] G.B. Arfken and H.J. Weber, *Mathematical Methods for Physicists, Fifth Edition* (Academic Press, San Diego, 2001).

About the author

Mr. Williamsen is a graduate student in Physics at the University of Wisconsin—Milwaukee. He is active in Milwaukee's busy music festival scene, and has recorded several album projects for local musicians. More info available at his home page, <my.execpc.com/~williamm/>.