Dr. Z's Test CD

Introduction

This PDF document serves as the liner notes (or "booklet") for *Dr. Z's Test CD*, which is really just a collection of 24/96 files[1] in the FLAC format[2] and not a physical disc at all. But back in the day, my collegue Prof. John V. Olson and I used to dump files like these in WAV format onto a CD-R for use at home – such were the halcyon days before "computer audio" was a household thing – hence; for historically sentimental reasons, a Test *CD*. That said, the "disc" was authored for system evaluation and set-up, and is therefore not recommended for an enjoyable listening session. You may think that there are some typical test files missing or notice that there isn't much in the way of traditional musical content – as to the former, a dedicated "burn-in" track is pointless and the intent was listening, not to write tracks for use with an oscilloscope to directly test electronic components (perhaps a *Test CD 2*?);[3] as for the latter, nothing musical is really missing, since that's what your collection is for, isn't it? This ensemble of tracks is primarily designed for testing and evaluation of loudspeaker placement, system setup, room characteristics and overall performance while ensconced in your favorite listening chair. You can use it with headphones, too, but with such devices you would be testing more for your hearing sensitivity or capsule isolation. Headphones are sufficiently unique beasts in their non-flat responses and they don't really interact with the listening room. Bottom line: this is nothing more than a work of computational-audiphiliac Onanism.

I'm trained as a physicist, but do a fair amount of statistical signal processing in my work,[4] particularly on digital records of infrasound generated from natural and anthropogenic sources (and in a previous life, of ground-based magnetometer data from the interaction of the solar wind with Earth's high-latitude magnetic field). But long before that I just really enjoyed listening to recorded music and thinking about the equipment that made that possible. That fascination has remained a strong and continuous thread, making for a lifelong pursuit; this work is therefore but an extension of it: to learn more about the digital aspects of music reproduction, as well as how we perceive that reproduction in real-world listening spaces. There are numerous other very fine Test CDs out there, so why make my own? I tell students that using a "black box" is all well and good – I drive a truck and certainly wouldn't want to build one – but from time to time you get more out of making your own box to better understand what's inside and how it works. In particular, the best way to make sense of data is often to "play" with it (using various digital signal processing tools), so what better playground than a project like this? I get to make the very data I'm about to

[1] Several alternative bit depths and sample rates are used in a few of the tracks to test for such things; all the other tracks were rendered at 24-bit/96 kHz resolution.

[2] Beginning with ver. 1.1, a second "disc" of DSD files in the DSF format was included.

[3] In ver. 1.1.2, some of those electro-analytical type files *were* added at Track [25].

[4] To be honest, I do more managing scientists and translating science to other non-techncial managers than actual science these days, *c'est la vie*.

play with! This particular project all started a few years ago over a beer and conversation... a few years/beers later and this is what I came up with. Perhaps you'll find it interesting and of some use, if not amusement.

Disc 1: Th' FLAC files

Track [1]: Channel ID: right & left...

`01-channelID.flac 0:58` This track begins with an announcement in the right channel, followed by twenty seconds of pink noise; the left channel is silent. Another announcement follows in the left channel, again accompanied by twenty seconds of pink noise; similarly, the right channel is now silent. Pink noise is that found-virtually-everywhere "$1/f$ noise". It should sound somewhat like rushing water, and relatively pleasant. This is due to the decreasing signal energy with frequency and the fact that it is quite common in nature. It should not sound quite so abrasive or annoying as white noise, like the static heard on an un-muted FM tuner when set to a frequency between broadcast stations – this is primarily due to thermal noise in the electronics. The power spectral density (PSD) of pink noise is inversely proportional to frequency. Alternately, each octave contains an equal amount of energy. The mathematical form of the pink noise PSD is given by $S(f) \propto f^{-1}$. In the ambient acoustic environment (e.g., rushing water) it is ubiquitous (and not only acoustically) – a good example is that the PSDs of J. S. Bach's Brandenburg Concertos go roughly as $1/f$. The PSD of white noise is "flat", or constant, independent of frequency. The pink noise generated for this track was scaled to -17 dBFS (decibels relative to full scale).[5] For more information on noise, see Track [8].

This is one track that is fairly useful for an aspect of headphone testing and evaluation. Since it provides signal to only one channel at a time, it serves as a good indicator of both the physical and electrical separation of each ear's capsule, connecting cables and headband. If the cables and headband are properly inert, no vibrational crosstalk between channels ought to occur. There is a common ground shared by the channels in unbalanced headphones, so there may be some degree of electrical crosstalk in such designs. The physical and electrical isolation should be readily apparent and the pink noise clearly defined if your headphones are properly executed.

Track [2]: Stereo Channel Phase ID: in & out...

`02-phaseID.flac 3:01` This track has two sections. The first comprises five subsections of in-phase signals; i.e., the left and right channels contain exactly the same (bit-for-bit)

[5] See the Computation section for the working definitions of dBFS used on this disc.

content ($s_{right}(t) = s_{left}(t)$). After a brief pause of silence, the second section has five similar subsections of signals that are out-of-phase; i.e., the right channel is the opposite of what is in the left channel ($s_{right}(t) = -s_{left}(t)$). Since the signals have no DC component (this would eventually cause damage to the speakers), merely inverting the sign renders the signals out of phase. The in-phase section should sound well-centered and focused between your speakers at all frequencies. The out-of-phase section may be difficult to localize, or sound like it is coming from directly beside you. The former should remain relatively stable at the listening position as you move your head about a bit; the latter may jump wildly, even with small changes in the listening position.Via headphones, the out-of-phase effects are quite striking.

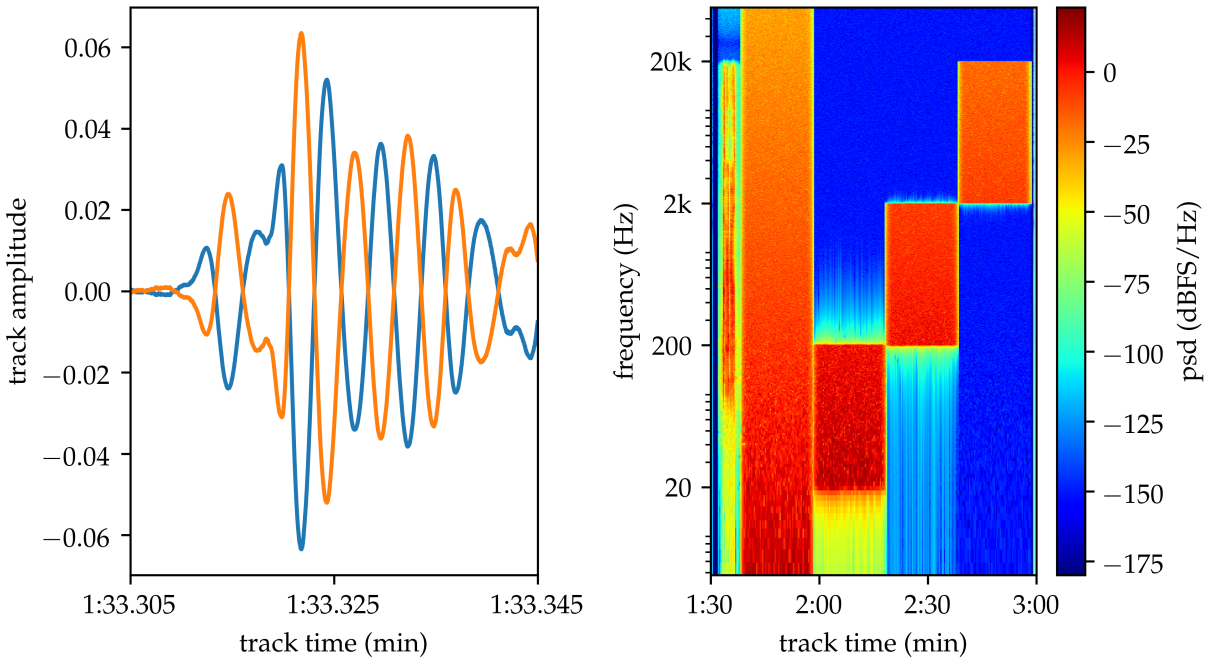

Figure 1: The first few moments of the out-of-phase announcement segment of Track [2] (left panel), depicting the opposite (out of phase) nature of the left (blue) and right (orange) channels. The track amplitude is given arbitrary units, relative to the ± 1 extremal values of typical PCM files (i.e., full scale). The right panel depicts the time evolution of the PSD of all five subsections of the out-of-phase portion of Track [2]: announcement, full-bandwidth pink noise, and the three ascending band-limited blocks.

The subsections begin with an announcement, followed by a segment of full bandwidth -17 dBFS pink noise, then three segments of band-limited pink noise (all of which are 20 s in duration). The band-limited segments were bandpass filtered for the three decades of what nominally covers the typical human hearing response ranges: the bass (20 - 200 Hz), midrange (200 - 2k Hz) and treble (2 - 20 kHz). These particularly chosen band-limited segments are somewhat arbitrary in their

boundaries, but they should help illustrate any particular problems with speaker-listening room interactions or speaker crossover issues.

In Figure 1, the left panel depicts the out-of-phase nature of the first few oscillations of the opening phrase of the announcement at around the 1:33 mark of Track [2]. Note the very small relative amplitude; this is due to zooming in enough to clearly depict individual oscillations at 96 kHz sampling. The left channel is depicted in blue – the right (orange trace) channel is simply the opposite of the left. The right panel depicts the time evolution of frequency content of the five subsections (announcement, full-bandwidth pink noise, three band-limited pink noises) of the out-of-phase portion of Track [2]. This type of plot is a spectrogram and depicts the time evolution of the signal's PSD, and note that the frequency axis is log-scaled, to better match our perception of relative pitches. For a discussion of what the dBFS/Hz scale in the colorbar means in terms of the PSD, see the Computation section. The pink noise is nearly full-spectrum, out to the 48 kHz Nyquist frequency.[6] That the announcement is cut off at about 20 kHz is likely a function of the response of the laptop microphone used to record it, and/or its onboard processing hardware. The three band-limited decades of pink noise are clearly shown.

Track [3]: Chromatic Scales: from C1 to C8 & back again...

`03-chromaScale.flac  3:48` This track also has two symmetric sections. The first comprises several chromatic scales in the left channel, with marker tones appearing in the right channel only at every C-note. The scale rises, and then descends back down. The chromatic scale is somewhat more a musical construct than mere pure tones, since it is something our ears (minds, really) are well accustomed to – deviations from the scale are relatively easy to detect, even for a non-musician. You don't need perfect pitch to hear if there are issues in faithfully reproducing these scales. So problems with your system, listening room and/or speaker placement may be evident and easily identified without anything more complex than the test and evaluation package nestled between your ears. The process is repeated in the second section, but with the channels reversed.

These scales are based on the tuning of $A_4 = 440$ Hz – this is the familiar "concert A" of western music that the oboist is called to play when a symphony orchestra tunes up. These scales cover seven octaves, and so eighty-five of the eighty-eight standard keys of a modern piano, from C_1 (32.703 Hz) to C_8 (4.186 kHz). This arrangement of keys and tuning has been used for the past few centuries and is known as twelve-tone equal temperament, because each note is one twelfth of an octave apart from its nearest neighbors, such that their frequencies are given by

$$f_{n\pm m} = f_n \cdot 2^{\pm \frac{m}{12}} \ .$$

[6] For a any periodically sampled process, the highest frequency resolvable is exactly half of the sample rate. This is known as the Nyquist frequency. So for this disc's $f_s = 96$ kHz, the Nyquist is given by $f_N = f_s/2 = 48$ kHz.

A single scale thus comprises the following twelve notes[7]

$$C_n \;\; C_n\sharp \;\; D_n \;\; D_n\sharp \;\; E_n \;\; F_n \;\; F_n\sharp \;\; G_n \;\; G_n\sharp \;\; A_n \;\; A_n\sharp \;\; B_n \;\; ,$$

leading back to the next octave and C_{n+1} at twice the frequency of C_n.

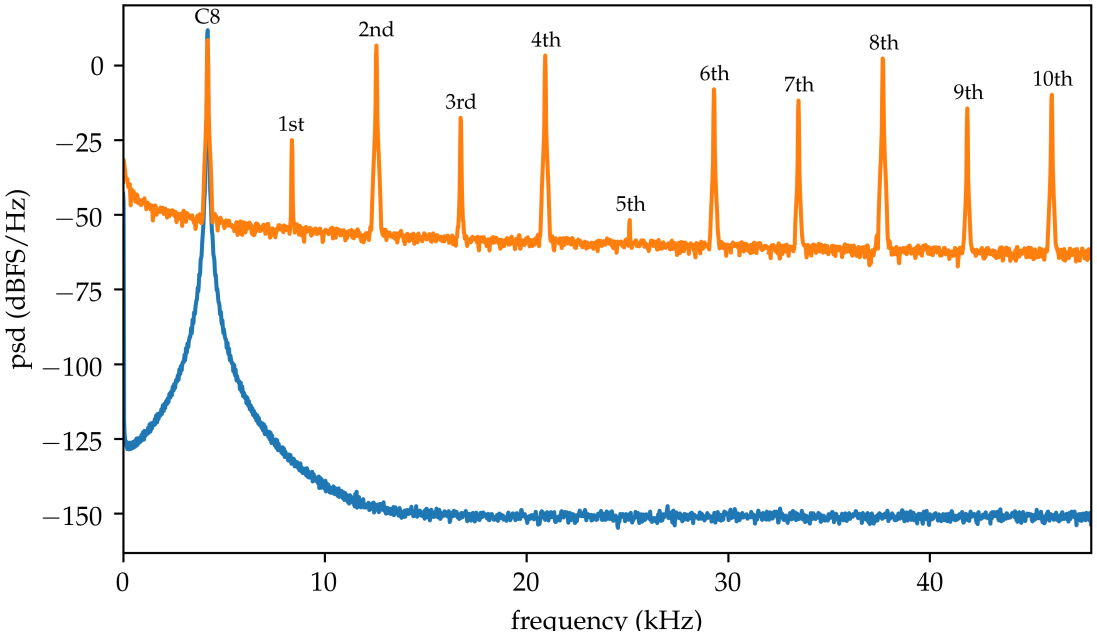

Figure 2: A PSD comparison of the relatively pure C_8 tone at the top of the scale in left channel (blue) at roughly 57 s on Track [3], with the timbre of the doubled C_8 note in the right channel (orange). Harmonics are noted above each multiple of the C_8 fundamental, but note that the nth harmonic is a multiple of the fundamental by $(n+1)$.

Each note of the scale is based on a pure tone of 0.45 s duration, followed by a 0.2 s silence. To avoid ringing effects (caused by truncating a sinusoid suddenly), each note fades in (the attack) from silence via a cosine taper that is $\frac{1}{37}$ of the note duration. The note plays for a bit (the sustain) and then fades out (the decay) via the same taper, but less quickly (the last two thirds of the note). The marker C-notes of the opposite channel are not pure tones, but instead have a timbre and a pink noise floor, so while they sound in tune with the C-note being played in the opposite channel, higher harmonics are present as well. The level of each note is set, again, to -17 dBFS. The C_1, C_4 "middle C" and C_8 notes are signaled by double notes in the C-note channel. Figure 2 depicts the harmonic structure of timbre in comparison to the pure tones of the C_8 notes in each channel

[7]Each sharp note (the black keys on a piano) could alternately be labeled as the next higher note with a flat; e.g., $F_n\sharp$ is the same frequency as $G_n\flat$ – tho' a dim memory from my childhood trumpet-playin' days tells me that the *musical* interpretations of the two notatonal forms *are* distinct.

at roughly 57 s of Track [3]. Note the timbral extension all the way out to the 48 kHz Nyquist frequency, with harmonics at multiples of the fundamental C_8 note. A table containing all the individual note names, frequencies and start times is provided at the end of this booklet.

Track [4]: Warble Tones I: sweeps...

`04-warbleSweeps.flac  3:54` This track contains a series of sweeping (continuous change in frequency with time, not discrete steps – also known as a chirp) tones that span the traditional, if somewhat unrealized, human hearing response. This response is typically defined to cover the three decades (nearly ten octaves) from 20 to 20k Hz. It is unlikely that your speakers will be capable of reproducing the entire range (especially the lower end), and even more unlikely that your ears will be capable of hearing what can be reproduced (especially the higher end). As the tone's center frequency changes with time, it also oscillates about the instantaneous center – this is known as warbling. All the tones on this track are identical, in-phase stereo channels.

The reason the tone is warbled is to mitigate the effects of room resonances. A 5 Hz warble frequency is used here, as experience shows that this variation does not allow room resonances to fully develop. The stepped tones of Tracks [5]–[7] are also warbled after the same fashion, and for the same reason. That said, these four warbled tracks will allow you to detect room and speaker cabinet resonances, as well as potential issues with speaker crossover handoff between drivers.

The sweeps are broken down into four segments, all depicted in Figure 3. The first is a full-scale rise and fall from 20 to 20k Hz. A descending sweep of the bass decade (200 to 20 Hz) follows. Note where you can no longer hear the tone and you will get an idea of the bass extension in your system; similarly, you will find the limit of your hearing sensitivity at the high frequency end when you can no longer hear the warble tones. The next segment is a rising sweep through the midrange (200 to 2k Hz), and finally a treble sweep (2 to 20 kHz) is presented. Throughout what is audible to you, there should not be any pronounced attenuation or amplification – the perceived volume should remain roughly constant. But, of course, this will occur only in a perfectly executed listening space; the response of real world rooms always have peaks (resonance) and nulls (attenuation) dictated by the geometry of the room and the wavelength (frequency) of the source material. In Figure 3 you will see that the tones extend beyond their nominal decadal boundaries of 20, 200, 2k and 20k Hz. Since the tones are subjected to a two second fade in/out via a cosine taper, this padding allows the tone's power to be exactly the -17 dBFS for the duration between each decadal marker, regardless of the fades (this normalization is applied to Tracks [5]–[[7] as well). Each tone is swept at the rate of 5 s/octave. The spectral width of the warble tone is approximately a factor of $\pm\frac{5}{37}$ about the instantaneous center frequency. The instantaneous center frequency of the sweeping tone (not including the warbling about) is given by

$$f(t) = f_o \cdot 2^{\frac{t-t_o}{\pm T_{oct}}},$$

where f_o is the frequency at time t_o and T_{oct} is the time to sweep one octave. The sign of T_{oct}

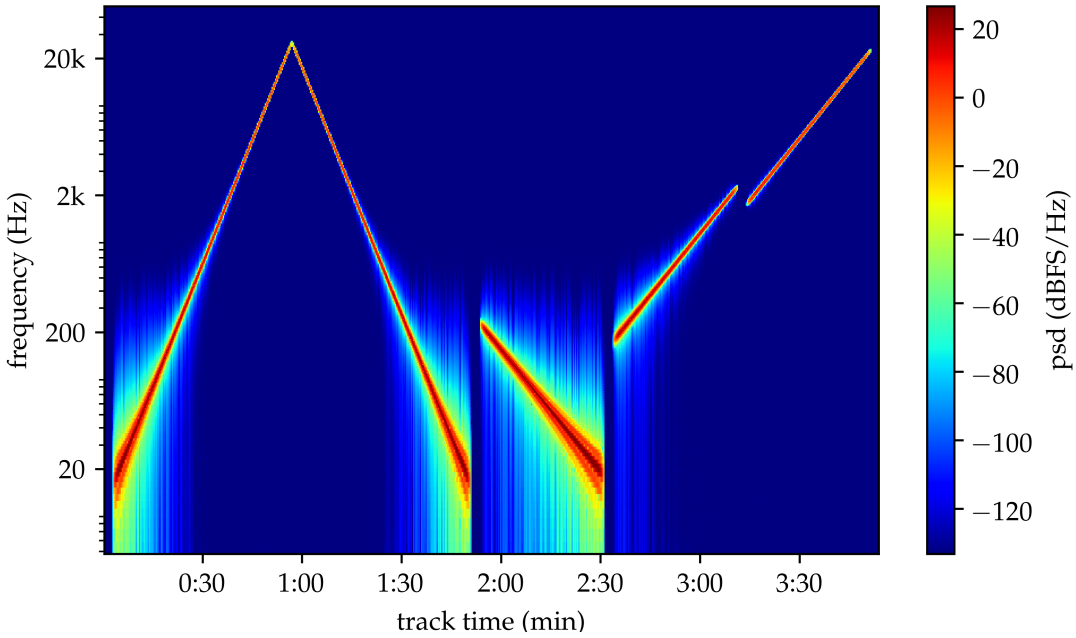

Figure 3: The time evolution the PSD of the series of sweeping warble tones that comprise Track [4]. The first is a full-scale rise and fall from 20 to 20k Hz. The three decades of the standard human hearing response are swept by all of these tones. A descending sweep of the bass decade (200 to 20 Hz) follows, then rising sweeps through each of the midrange (200 to 2k Hz) and treble (2 to 20 kHz) are presented.

determines whether the tone sweeps up or down in frequency. A table containing the time indices of some representative frequencies in each sweep segment is provided at the end of this booklet. This can be useful for assessing problems with room response corrections.

Track [5]: Warble Tones II: bass steps (descending)...

`05-warbleSteps20.flac  2:38` Once you identify a potentially problematic tonal area with Track [4], you can use the next three tracks ([5]-[7]) to further analyze the issue, and hopefully address it. These tracks provide fixed center-frequency warble tones to isolate parts of each decade in the range of human hearing. The first such track ([5]) employs the same philosophy and techniques as Track [4]; however, the continuous sweep (or chirp) is replaced by a series of faded 1/3-octave warbled steps across only the bass decade (200 to 20 Hz; descending, as was for the bass sweep) at steady center frequencies. The center frequencies are roughly given by 201.59,

160, 126.99, 100.79, 80, 63.496, 50.397, 40, 31.748, 25, 198 and 20 Hz. Mathematically, they are given exactly by

$$f_{10-n} = 20 \cdot 2^{\frac{10-n}{3}} \quad n \in [0, 1, \ldots, 9, 10] \ .$$

Each tone persists for 12.7 seconds, followed by a 1.25 s silence. The same 2 s fades on each end of the note are used as in Track [4]. Again, a 5 Hz warble frequency with a width factor of roughly $\pm\frac{5}{37}$ is used here; all the tones on this track are identical, in-phase stereo channels. A table containing each step frequency and start time is provided at the end of this booklet.

Track [6]: Warble Tones III: midrange steps...

`06-warbleSteps200.flac  2:38` This track is quite simply the same as Track [5], apart from the fact that it covers the next decade in ascending fashion, spanning the midrange from 200 to 2k Hz. The center frequencies of the steps are a rising sequence, exactly ten times those in Track [5], or

$$f_n = 200 \cdot 2^{\frac{n}{3}} \ , n \in [0, 1, \ldots, 9, 10] \ .$$

All other parameters are the same as in Track [5]. Figure 4 depicts the ascending tones on Track [6] as the time evolution of the PSDs of each tone. The figure for Track [5] would look similar, but with the time-axis reversed (it was a descending series), and the frequency scale divided by a factor of 10. A table containing each step frequency and start time is provided at the end of this booklet.

Track [7]: Warble Tones IV: treble steps...

`07-warbleSteps2000.flac  2:38` This track is, again, simply the same as Track [6], apart from the fact that it covers the next decade, now ranging across the treble from 2 to 20 kHz. The center frequencies of the tones are a rising sequence, exactly ten times those in Track [6], or

$$f_n = 2000 \cdot 2^{\frac{n}{3}} \ , n \in [0, 1, \ldots, 9, 10] \ .$$

All other parameters are the same as in Track [6]. The figure for Track [7] would look similar to Figure 4, but with the frequency scale multiplied by a factor of 10. A table containing each step frequency and start time is provided at the end of this booklet.

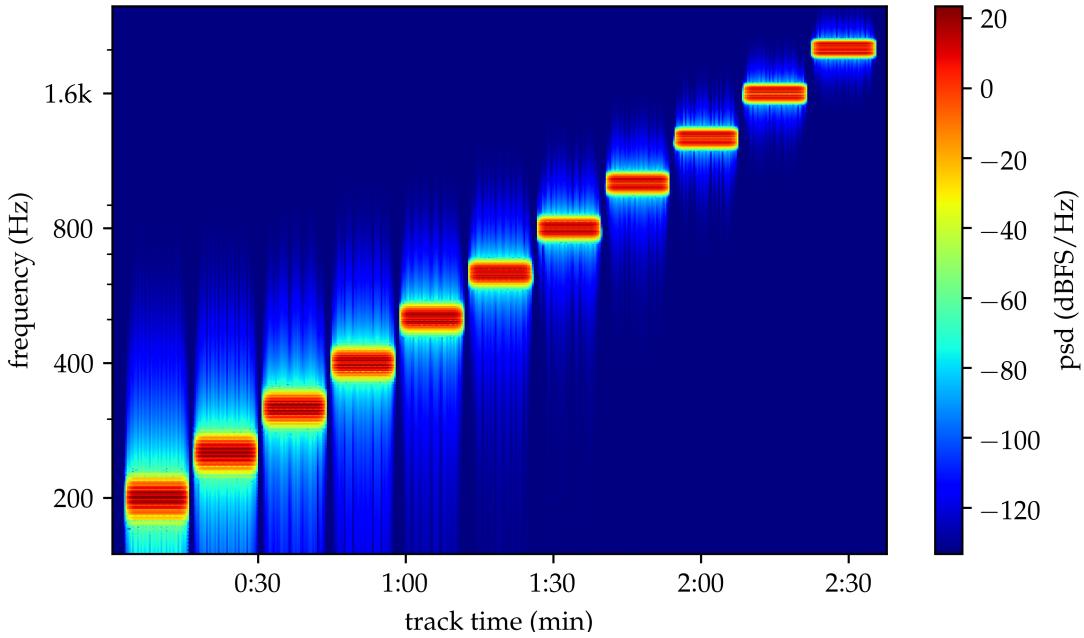

Figure 4: The time evolution the PSD of the series of stepping warble tones that comprise Track [6]. The individual steps (or tones) are spaced at 1/3-octave intervals and span the decade covering the midrange from 200 to 2k Hz. The spectrogram of Tracks [5] and [7] would look similar (see text).

Track [8]: Colored Noise: white, pink and brown...

`08-cNoises.flac  1:12` This track comprises three types of "colored" noise for comparison purposes, each separated by a brief silence. Each channel carries an independent realization of 20 s of a type of noise (i.e., they are not identical or simply out-of-phase – in statistical signal processing parlance, they're uncorrelated). The first sample is white noise (the signal and spectrum shown in blue on Figure 5) – this should not sound so pleasant over time and you will notice a lot of high frequency content, since all frequencies are equally represented in the PSD. The second sample is pink (orange traces, Figure 5) – as discussed for Track [1], it is a pleasant sound.[8] Finally, the third sample is brown (or Brownian[9]) noise (green traces, Figure 5) – this type of noise has a PSD that falls off with twice the slope of the blue trace in the right panel of Figure 5. It is akin to surf noise, with even more emphasis on low frequencies than pink or white.

[8]Some products incorrectly purport to use "natural white noise" for soothing sounds, but they are actually using pink or brown noise for their content.

[9]A significant contribution to statistical physics was made by Einstein in his theoretical description of molecular Brownian motion.

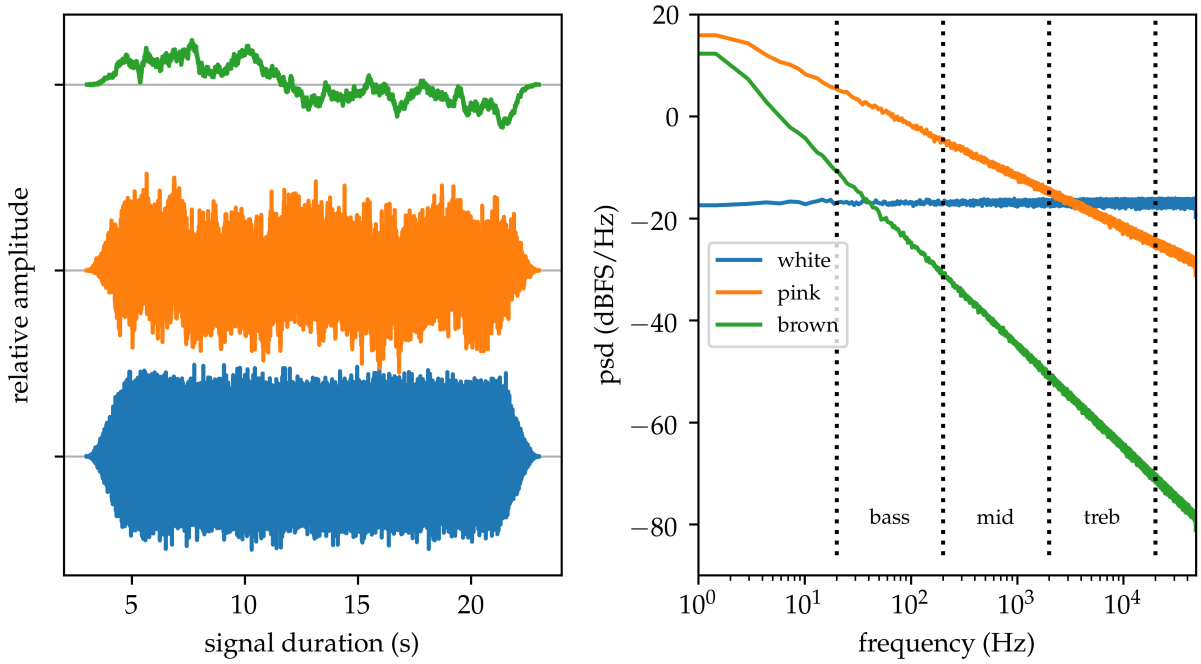

Figure 5: Noise samples from the left channels of Track [8] for white (blue traces), pink (orange traces) and brown (green traces) noise (left panel), and their respective PSDs (right panel). White noise has constant energy per frequency, whereas pink noise falls off at a rate of 3 dB per octave; brown noise at 6 dB per octave. Each signal is rendered at -17 dBFS. Stadia (dotted) are included on the PSD panel to visually separate the infrasonic, bass, midrange, treble and ultrasonic frequency bands. Please note that the irony is not lost on the author that the color of each trace does not correspond to its noisy namesake!

You may perceive that each type of noise is quieter than the preceding example. This is not actually the case, since each of the samples is normalized to be rendered at -17 dBFS they have precisely equal energy (or "volume") content – it's a combination of your speaker's inability to faithfully reproduce the extreme lower octaves and your ear's inability to resolve much of that low frequency content that makes brown noise sound quieter than the others. The PSD for each has the form $S(f) \propto f^{-c}$, where $c \in [0, 1, 2]$, for white, pink and brown, respectively. In Figure 5, stadia lines (dotted) have been added to demarcate the nominal divisions between infrasonic, bass, midrange, treble, and ultrasonic bands. Even in the time series (left panel), the low-frequency emphasis in brown noise is quite evident, and in the PSD, very little brown signal energy is present in the audible range.

Tracks [9]–[16]: Bit Depths & Sample Rates: xx bit/yy kHz...

`tt-bitRatesxxyy.flac 0:ss` These tracks comprise 2 s of pink noise (faded in and out for 0.045 s on each end) along with a synthetic voice (all at -17 dBFS) announcing the bit depth and sample rate[10] at the beginning and end of each track. These short tracks are detailed in the table, below, and are for use in confirming that the correct bit depth and sample rate are being sent to, and output by, your playback software and DAC.[11]

Even though my current DAC[12] supports 32-bit file formats, I opted not to include any for a few reasons, some better than others. Firstly, FLAC decoders only support up to 24-bit PCM – that's easy enough to fix by simply writing to a 32-bit WAV, but I like digital files fully tagged with metadata. A second, if equally weak reason, is that there's effectively no real-world commercial content in 32-bit format, despite the proliferation of 32-bit DAC chipsets. Third, and perhaps the only technical reason, is that even the finest 32-bit chip is going to be effectively limited to 20- or 21-bit resolution (~126 dB s/n ratio) in the real world (see Track [22] text). This is true even with the high precision scientific instruments I use in my low frequency acoustic work – electronic noise floors in analog stages are things we must contend with in the real world, too.

track no.	filename	duration	bit depth	sample rate
[9]	`09-bitRates1644.flac`	`0:24`	16	44.1 kHz
[10]	`10-bitRates1696.flac`	`0:23`	16	96 kHz
[11]	`11-bitRates2444.flac`	`0:24`	24	44.1 kHz
[12]	`12-bitRates2448.flac`	`0:23`	24	48 kHz
[13]	`13-bitRates2488.flac`	`0:24`	24	88.2 kHz
[14]	`14-bitRates2496.flac`	`0:24`	24	96 kHz
[15]	`15-bitRates24176.flac`	`0:26`	24	176.4 kHz
[16]	`16-bitRates24192.flac`	`0:25`	24	192 kHz

Track [17]: Perceptual Pitch I: a sequence of Shepard tones...

`17-shepardTones.flac 1:02` This track was actually authored *after* Track [18], so it might be helpful to read that track's description first. Unlike the chirps of Track [18], this track

[10] You'll notice I didn't construct all possible combinations for the bit depth and sample rate – there's a bias here toward what is actually in my collection. And yes, Michael Fremer did actually send me some 16/96 files ripped from a Romero's LP on his Continuum Caliburn rig! For real world listening I think that 24/96 is a practical limit, at least at the time of this writing – in fact, given files sizes and even taking into account that storage is cheap, from what I've actually been able to *hear*, I think that 24/48 might really be the sweet spot. But that's just *me*.

[11] Writing tracks to DSD (i.e., a `.dsf` file) is beyond my PYTHONic technical ken; however, a solution was found, and so a second "disc" of such files is now provided.

[12] At the time of producing this Test CD I was using the sublime Grace Design m920, which supports up to PCM "32/384" and DSD128 resolutions.

presents a sequence of discrete tones (chords, really) that seem to be ever-increasing in pitch. The tones and mathematical parameters are set up such that they are the same as laid out in Roger Shepard's original 1964 paper.[13] Although he used computer punch cards and output to magnetic tape at 10 kHz sampling, everything else here is the same as in his original experiment – this is just my pure digital, "24/96" homage, normalized to -17 dBFS. Not so useful a track for system evaluation, but where's the harm in a little computer audio intellectual geek-ery?

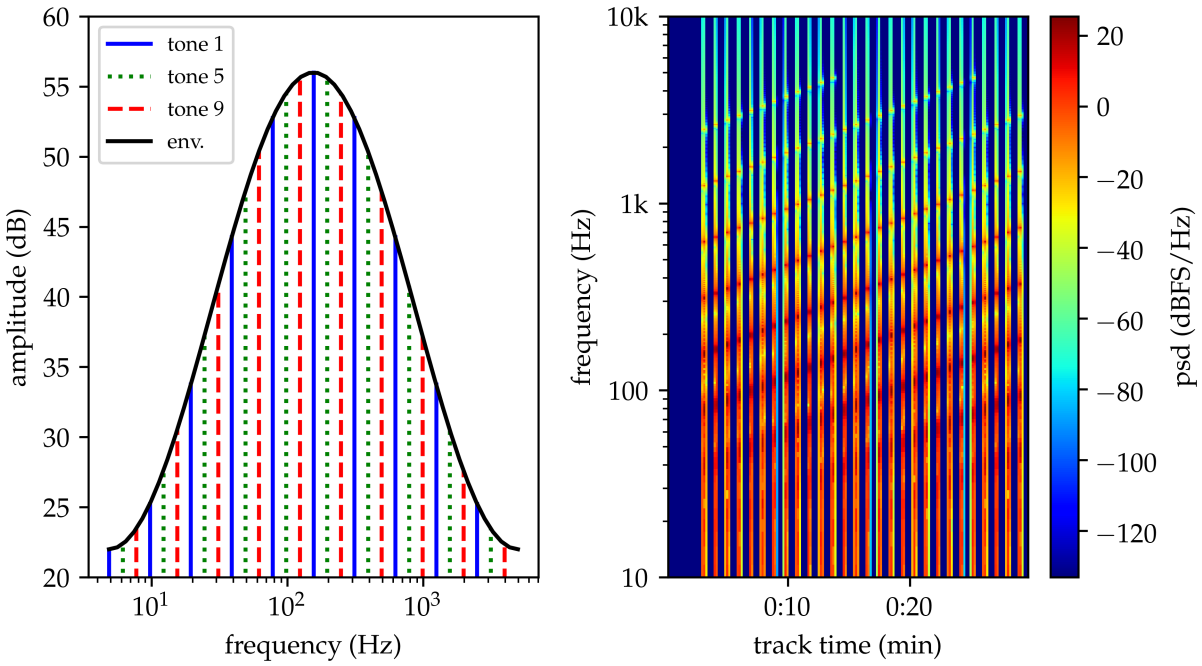

Figure 6: The component structure of selected tones in the sequences of Track [17], and the amplitude envelope applied to each, is depicted in the left panel. The time evolution the PSD of the series of these "Shepard tones" is shown in the right panel. Note that the transition back to the beginning of the sequence at 14.5 s is graphically obvious, but difficult to hear. See text for details of the signal parameters.

Each tone is a 0.12 s chord (note with timbre), spanning the fundamental and nine harmonics by octaves; e.g., $f_o \cdot [1, 2, 4, \ldots, 2^9]$. A 0.84 s gap is placed between each tone, and a smooth (cosine taper) fade in/out over 0.01 s is applied. The sets of tones start at $f_o = 4.863$ Hz and f_o increases eleven times such that one more step (the thirteenth tone) would have a fundamental of $2f_o$. Therein lies the key to the perceptual conundrum – the set of twelve tones is a complete one, and the cycle merely repeats five times. The amplitude of the components of each chord are also multiplied by a cosine taper (in relative terms, 22 dB at the extremes and 56 dB in the middle) in log-frequency. This accomplishes two goals: a) the sum of the amplitudes of each

[13] See `http://dx.doi.org/10.1121/1.1919362`.

chord's components is a constant (i.e., relatively constant loudness with each tone), and b) much like the glissando in Track [18], as one component passes quietly out of hearing at high frequency, a new one is quietly taking its place at the low end. In this way, the thirteenth tone (about 14.5 s into the track) is replaced by the first, and the listener is unaware(?) that the sequence of tones is simply repeating. The right panel of Figure 6 depicts the time evolution of the PSD of Track [17] over its first 30 s. The left panel depicts the the relative amplitudes of the components of selected tones in the sequence. Note how they are all subject to the same amplitude envelope, such that as components enter and leave successive tones, they are not perceived as either "new" or "missing".

Track [18]: Perceptual Pitch II: a Shepard-Risset glissando...

`18-shepRissGliss.flac  4:26` This track was added to the ones from the original version of the Test CD thanks to Prof. Peter A. Delamere mentioning Shepard tones while we were out running on the campus trails. The glissandos of this track are something I had coded some years ago for my physics students, while teaching at North Pole High School. At that same time I was first introduced to John Luther Adams, and a demonstration of that code served as an "incidental interview" of sorts to work with him on his installation, *The Place Where You Go To Listen*, at the University of Alaska Fairbanks' Museum of the North. Unfortunately, somewhere along the line I lost that original code. Fast forward a few years and, now that the Test CD was being rewritten in PYTHON, I thought to construct a new version of the code and that it would also make an interesting test track – thanks, Peter! Again, sounds pretty cool, but perhaps not so useful to the audiophile.

The perceptual conundrum underlying a series of Shepard-Risset chirps is the same as for the discrete tones of Track [17], in that they also seem to perpetually rise in pitch, yet remain in the audible range. It may be done in discrete steps[14] or as a series of chirps as on this track. Such chirps were first produced by Jean-Claude Risset; hence, the Shepard-Risset glissando. The key is that in the midst of a series of these tones, one will be rising into the middle of the range of human hearing and at full volume. Meanwhile, a previously-started tone's pitch is rising out of the range of human hearing as it diminishes in volume; similarly, a new tone will have started infrasonically, and will be increasing in both pitch and volume. The result is a perception of ever-increasing pitch. The impression is not perfect, however. Your mind may occasionally lock onto one tone and then jump quickly across to another, creating a sporadic impression of jumps in pitch.

The track contains a series of 62 chirps, each 86 s in duration, with a staggered start of ~2.867 s between them. Each chirp or glissando fades in to full volume from 13.33 to 96 Hz. The chirp is sustained at full volume between 96 and 4.167k Hz, then begins to gradually fade out toward 30 kHz. These parameters are quite different than Shepard's original ones for discrete tones, and were chosen by the author empirically, such that the perceptual effect is maximized (at least to my

[14] Roger N. Shepard's original 1964 paper in the *Journal of the Acoustical Society of America* took this approach (see Track [17]).

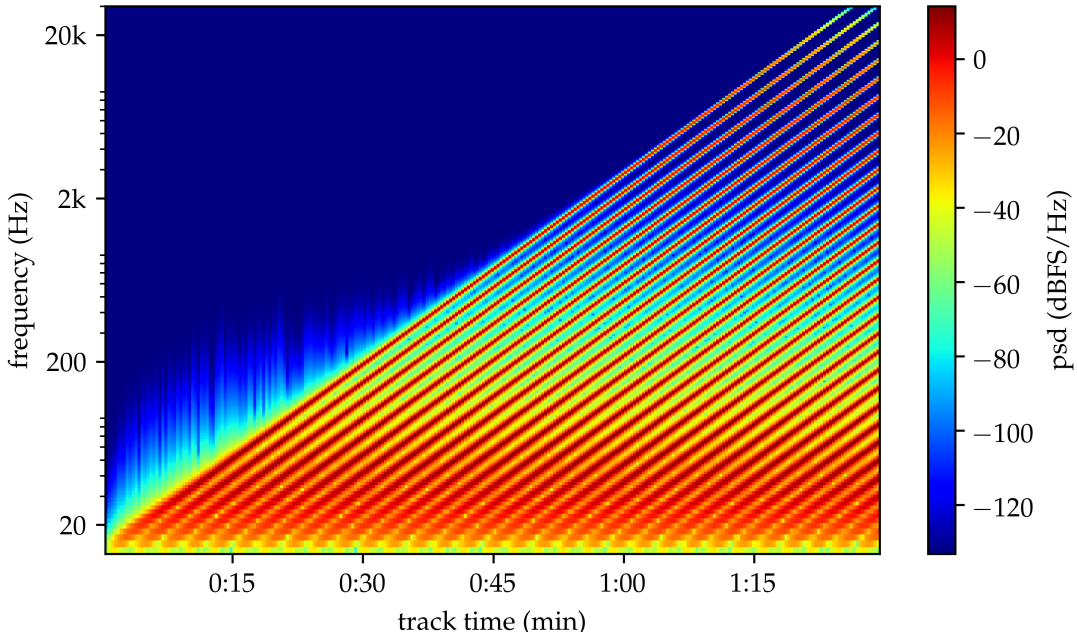

Figure 7: The time evolution the PSD of a series of Shepard-Risset glissandos for the first minute, or so, of Track [18]. Each begins subsonically, rising in amplitude to full volume somewhere in the audible range. Then each tone begins to fade out, while still in the audible range, and terminates ultrasonically. See text for details of the signal parameters.

ears). There's nothing really useful in testing your system with such a track, but it was pretty cool to code up and equally so to listen to! Of course, the overall level is set to -17 dBFS.

Track [19]: Phase Inversion: ±polarity...

`19-phaseInvert.flac  1:06` This track is an excerpt from the opening Aria of the *Goldberg Variations*[15] by Johann Sebastian Bach (BWV 988). Unlike the rest of the Test CD, this 24-bit, 96kHz track is not normalized to -17 dBFS; rather, it's level is left unaltered (roughly -19 dBFS in our RMS definition). As with many tracks on the Test CD, there is a 3 s pad of zeros (silence) applied to both end of the excerpts. First, we have a 28.672 s excerpt of both channels, reproduced

[15]Performed by Kimiko Ishizaka on a Bösendorfer 290 Imperial CEUS piano. The excerpt is from the *Open Goldberg Variations*, which are free to download and share. They are governed by the Creative Commons Zero license (CC0), which means that they are a part of the public domain. A quick point about the right panels of Figure 8: the Bösendorfer 290 Imperial CEUS piano is equipped with 97 keys, adding nine bass keys down to C_0, but such notes are not called for in any of the *Goldberg Variations*.

exactly as in the original FLAC file. After a brief pause, the same excerpt is played, but the polarity is inverted in *both* channels (unlike on Track [2]). Some people claim to be able to discern absolute phase – this track might enable you to decide for yourself.

Track [20]: Phase Distortion: gradually corrupted phase...

`20-phaseDistort.flac  0:34` This track takes the right channel of the same 28.672 s excerpt from Track [19] and leaves those 2,752,512 samples well enough alone (the standard 3 s pad of zeros is appended on each end), at its original -19.5 dBFS level. The left channel is the same as the right in some ways; in others, not. The phase of the left channel is gradually distorted as the track plays. Initially the first segment has no phase distortion, but by a quarter way through the excerpt (roughly the 10 s mark), the left channel has completely randomized phase. Despite this, you'll notice that the musical content of the right (undistorted) channel is still clearly recognizable in the left, despite the distortion. Phase is a tricky, and often ignored aspect of spectral analyses – all the spectra depicted prior to this track destroy the phase information entirely and only depict the magnitude-squared amplitude of the spectral components of any signal. That two tracks have identical spectral amplitudes is not sufficient to make them "identical" – amplitude is only half the story. And even then, the amplitude isn't all that fixed a quantity either!

The length of the excerpt for this track was chosen so that it might be broken up into 1,344 segments (0.014 s each at 96 KHz sampling) of 2^{11} samples – it's merely convenient that it was also used in Track [19]. To effect the phase distortion, each of those segments from the right channel was Fourier transformed[16] and subjected to a phase shift. The phase of each Fourier component of the n^{th} segment was shifted by

$$\phi'_n = \begin{cases} \phi_n + 2\pi\frac{(n-1)u}{366} & \text{for } n \leq 336 \\ 2\pi u & \text{for } n > 336 \end{cases},$$

where ϕ_n and ϕ'_n are the original/distorted phases and u is a uniformly distributed random deviate on the unit interval. Note that since phase is cyclical over the unit circle, once you add a uniformly distributed random shift on that interval, you have effectively completely randomized the phase of the Fourier components of that segment. After this step, the inverse Fourier transform is computed to render a distorted time series from the original.

It is pretty clear that the two channels of Track [20] *sound* different (but not completely so), and indeed the time series shown in the left panel of Figure 8 reflect that. However, comparing the amplitudes of the raw spectral components (by taking the FFT of both channels of each of the 1,344 blocks of 2^{11} samples) we find that they are *identical*. This is why the raw (red) trace in the upper right panel of Figure 8 is valid for both the left and right channels. In fact, they are identical by construction, since only the phases were messed with. Now, phase is a tricky thing – what if

[16] Quickly, since the fast Fourier transform (FFT) was used and each segment length was a power of two.

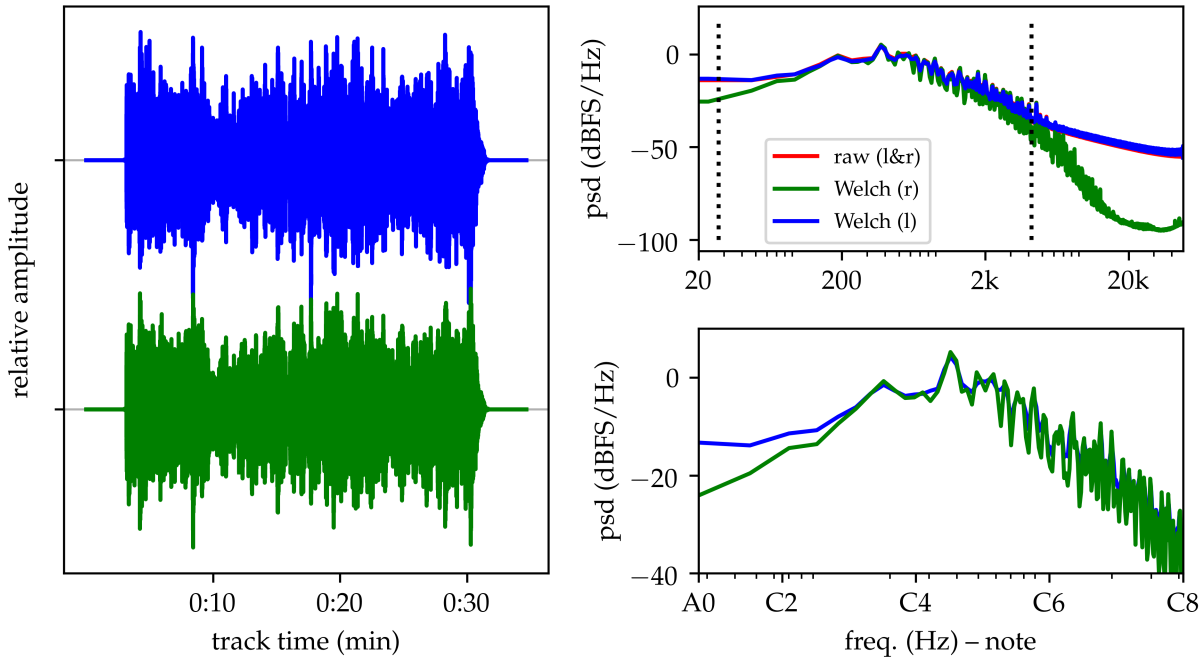

Figure 8: The impact (or lack thereof) of phase distortion on amplitude in Track [20] is depicted by various means. The left panel illustrates the two channels as time series; original (right channel, green) and distorted (left channel, blue) – they appear similar, but not identical. The upper right panel shows the similarity in the spectra of each channel; raw spectra for left and right are shown in red (they *are* identical), more conventionally computed Welch spectra for the right (green) and left (blue) channels, and stadia are placed at the frequency of the A_0 and C_8 extremal keys of a standard piano. The bottom right panel zooms in on the Welch spectra of the upper panel, with various keys indicated at their respective frequency. See text for details and further explanation.

we didn't know that the left channel was prepared in blocks of 2^{11} samples? We'd probably not compute the spectrum the way we did for the red trace. To illustrate this point, the upper and lower right panels of the figure both illustrate the Welch spectra (a commonly used technique) that now uses 2^{12} samples (twice as many) per FFT, slides an overlapping window of that length through the data and averages the results. This technique is used to enhance confidence in the spectral estimates (beyond the scope of this booklet, for sure, but stop by for a beer and I'll regale you) at the expense of spectral resolution – digital time series analysis is a game of optimization and compromise. By analyzing the two channels in this fashion, we note that the left and right channels do *not* have the same amplitude – or do they? The distorted left channel's Welch spectrum is nearly the same as the raw spectra for either channel, but the right channel shows a clear difference at both high and low frequencies. In part, how the spectral amplitudes are estimated effects their values. In fact, estimating the spectrum (*estimating* being the key word) of identical

time series in different fashions yields different spectra. While statistical signal processing is an exact science, its application borders on an art form. It's in the eye of the beholder, eh?

Spectral amplitudes are functions of frequency, not time, but we have a convenient way of visualizing the time evolution of spectral amplitudes as a spectrogram (or time-frequency spectrogram; e.g., Figure 7). Phase information is a function of frequency, too. There are "phasograms" as well, which depict the time evolution of phase; however, the information is often difficult to comprehend – in fact, so much so that it's not worth making a plot of such here. It suffices to describe the phase distortion and visually look at its effect (or lack thereof) in the PSDs.

Track [21]: Bit Resolution I: effective bit depths...

`21-bitResolve.flac  3:44` This track is yet another homage, in this case to a demo Tom Erbe once showed me while we were working together with John Luther Adams. He'd written a neat little GUI that allowed you to change the effective bit depth of a short snippet of music, as it played, and keep the relative signal level (volume) constant. It was pretty cool to hear the differences in real time; moreover, it was amazing how recognizable the piece was even when the effective bit depth was incredibly low. In this track you'll hear both channels of the same 28.672 s excerpt from Track [19] (with 3 s silences padding the segments), seven times. First up, the unaltered, 24-bit/96 kHz version is presented.[17] The next five iterations are rendered at roughly the same dBFS level, but at ever decreasing bit depths: [16, 8, 4, 2, 1]. In the seventh and final segment, the bit depth decreases with time (by one bit each $\sim$0.68267 s) from the original 24-bit depth down to 4-bit resolution; then increases all the way back up again. This scheme works out since the excerpt divides up nicely into 42 blocks of 2^{16} samples each. It is interesting to note how recognizable the tune is, even at remarkably low (1-bit!) resolution; similarly, how difficult (impossible?) it is to tell the difference between 24- and 16-bit resolutions. Figure 9 shows an example of the discretization errors introduced by reducing the bit-depth of each segment by depicting a portion of the full-resolution excerpt and three of the degraded representations.

It is important to note that even the 24-bit portion (depicted in blue on Figure 9) comprises stepwise, discrete samples – it's not really a curve. You just cannot discern it on the scale of the plot since there are more than 16.7 million (precisely 2^{24}) levels available to represent the signal content. In fact, even the 8-bit representation (with only 2^8 or 256 levels available) is not easily differentiable from the original on the scale of this plot, so it's not shown.

Reducing the effective bit depth, while maintaining a constant dBFS level, is simply a matter of bit masking. It is the same effect as rounding a binary number down. To illustrate, let's look at a hypothetical 8-bit word representing a single sample: `10111010`. The same sample in effective

[17] The entire track is rendered at 96 kHz sampling – in fact, it's all actually still a 24-bit/96 kHz FLAC file. But by limiting what levels are available, we effectively reduce the bit depths to render the desired effects. Like Tracks [19]-[20], this one has *not* been normalized to -17 dBFS; it's original level was left unaltered.

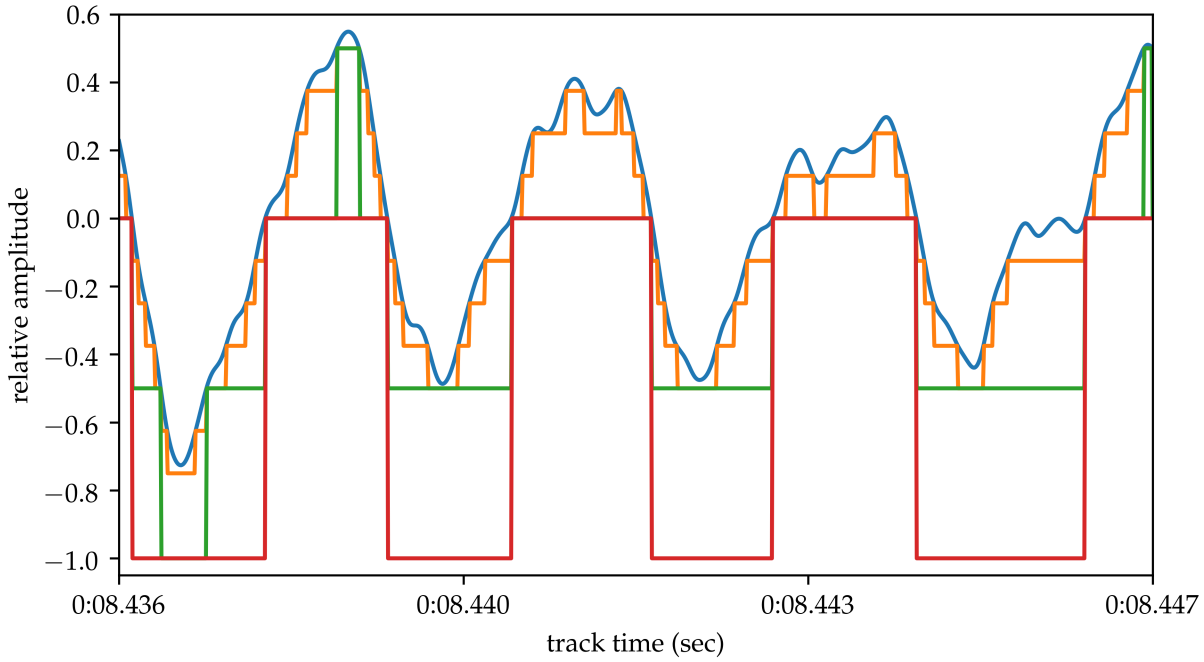

Figure 9: The difference between full and three reduced resolutions. The blue curve depicts 1,000 samples of the original excerpt at its native 24-bit resolution, with the actual track time (in seconds) shown for that portion. This window was selected since it spans the maximum amplitude excursion of the segment (dBFS, in the peak-to-peak sense). The effective bit depths of [4, 2, 1] are shown in [orange, green, red], each superposed on the same time scale. Note that the 24-bit "curve" is similarly discretized, but not visibly so on the scale of this plot. See the text for details.

4-bit resolution would look like: `10110000`. We simply rounded down the binary number representing the sample to the desired least significant bit (LSB) so that there are fewer variable bits available to record the signal (only four in this instance). But since we still have an 8-bit word, the signal level (volume) remains roughly constant.[18] A table containing each effective bit depths and approximate start times is provided at the end of this booklet.

Now for a few comments about the sound. As the bit depths decrease, there's obviously quite a bit of high frequency noise present (especially noticeable at about the 4-bit level) – this "hash" is quantization noise and it stems from the "square wave"-like appearance of the waveforms. Despite this, we can still listen to the music. In fact, even at 1-bit level (with only two levels to represent the signal) not only can we recognize the tune being played, but the tone of the piano as

[18]This operation is done in PYTHON using the bitwise AND operation, as follows: `0b10111010 & 0b11110000`, which gives `0b10110000`. Note that this is equivalent to rounding down to the nearest available bit level, hence the reduced representations in Figure 9 are all lower than the original.

well. Note that the 1-bit (red) trace in Figure 9 isn't quite a uniform square a wave, in that the duty cycle varies a little. This is, in effect, a good segue to the next "disc" containing the DSD files – what we've done is changed our 24-bit pulse code modulation (PCM) waveform into a 1-bit pulse width modulation (PWM, or DSD) waveform; albeit, very poorly.[19]

Track [22]: Bit Resolution II: effective noise floors...

`22-noiseFloor.flac  5:19` This track is indirectly related to Track [21], and directly to the comments at the end of the Tracks [9]-[16] description. It is designed to illustrate that effective noise floors are real-world constraints, notwithstanding their near-ideal representations in the digital domain. The 16-bit depth specified in the (in)famous CD Red Book Standard was selected, in part, because of practical considerations of electronics at the time, and also it provided for an *ample* 96 dB dynamic range (the difference in level between the quietest and loudest possible signals possible within the format).[20] Do we require more than 16 bits for practical listening?

It is a given that low-level signals are distorted most in the digitization process – fewer bits are available to construct an analogue of the acoustic pressure waveform than for a louder signal. So goes the argument then, that the more bits we have available, the better low-level resolution we obtain. *Tru dat!* – but *only* in the digital domain. Once we run that digital data through a DAC, we have to contend with noise in both the electronics and the listening environment.

First, the analogue stages of a DAC (or, for that matter, the ADC on the other end of the musical reproductive cycle) are inherently limited by their electronic components. Even the best digitizers, microphone pre-amps, etc., are limited to 20- or 21-bit effective resolution due to noise in the electrical components and circuits. So while 24- or 32-bit masters might be great for bit-perfect signal processing (filtering, editing, noise-shaping, dithering, etc.) in the digital domain, that level of resolution is unattainable upon listening – once you get to the analogue outs of your DAC, electronic noise governs the effective resolution.

Second, let's consider your hearing, both what's possible and healthy. The generally accepted auditory threshold is $\pm 2 \times 10^{-5}$ Pa at 1 kHz, or a fluctuation about the ambient pressure of roughly only one part in 10 billion! – this is the quietest sound a typical human can perceive and is referred to as 0 dB_{SPL}. On the opposite end, we have the threshold of pain at roughly 130 dB_{SPL} (or someone blowing a trumpet less than two feet from your ear).[21] To cover this range of sound pressure levels,

[19] To protect our speakers, we've taken a further step in preparing this track. Each of the reduced bit depth representations (middle five segments) has been normalized to remove most of the negative DC offset caused by bit masking – Figure 9 shows the signals before this de-trending was done. You'll hear significant start-up transients for the low bit depth segments – rather than ameliorate them with a fade, they are left as-is to demonstrate the sudden transitions from silence in such bit schemes.

[20] There is a rough equivalence that gives 6 dB/bit of information, but this is only an approximation to the proper expression, $\left(20 \log_{10} 2\right)$ dB/bit ≈ 6.02.

[21] We'll avoid using the OSHA limit for exposure of 85 dB_{SPL} as a reference "loud" sound, since folks will correctly

we'd need 22 bits (or 132 dB of dynamic range), but since that's not a multiple of 8 (i.e., a bit depth available on modern chips), we'll say that 24-bit data (144 dB) is what we need for realistic sound reproduction.

Somewhere in between those extremes there's a quiet listening room at 30 dB_{SPL}, or maybe as low as 20 dB_{SPL} – if you live deep in the woods, there's no wind bowing outside and no HVAC or refrigeration compressors running.[22] We'll take 30 dB_{SPL} as a pretty decent listening environment. That's our new threshold of "musical hearing" and all we require now is only 100 dB, or roughly the same as what a 16-bit sample will provide – go figure. The same constraint is placed on a recording at the venue end as well, since most sessions are not conducted in anechoic chambers. Assuming that most musical transients are nowhere near the threshold of pain, then the case of 16-bit data is stronger still. Now, it is true that we can hear things that are "buried in noise", like a particular person's voice at a crowded party (the so-called "cocktail party problem"), so while you can still discern sounds lower than the 30 dB_{SPL} of your quiet room, you cannot make them out all that well, and no amount of extra-bit, low-level resolution is going to help ameliorate that.

But enough pedagogy and on to some listening for yourself. This track revisits the familiar piano excerpt from Tracks [19]-[21]. It is first played back at roughly its original level, although that was set to -19 dBFS here to keep nice, integer-valued levels. With three seconds of silence between renditions, it is played back nine more times, each 12 dB quieter than the previous one, or equivalently, with two bits less resolution (i.e., reduced in volume by a factor of four). Eventually, we end up with a loss of 18 bits of effective resolution by the last rendition. The dBFS levels of all ten segments are given by

$$\text{dBFS} \in [-19, -31, -43, -55, -67, -79, -91, -103, -115, -127] \ .$$

This scheme was chosen since the signal is already at -19 dBFS to start with, or just more than 3 bits of resolution down from full scale. That is, the last example should be audible if a full 21-bit effective resolution is available to our electronics and in our listening room.

Set the volume on your system to a fairly high, but realistic level for solo piano on Track [19], then proceed to this track without adjusting the volume setting. The test here is to determine if you can hear any of the last examples above the noise floor of your electronics and/or listening room. As shown in Figure 10, even the last rendition is clearly above the digital noise floor. Where can you no longer hear the piano playing the piece? A table containing each of the dBFS levels and approximate start times is provided at the end of this booklet.

argue that we need that extra bit depth to cover only extremely brief, dynamic transients to high acoustic levels, not continuous, high-level playback.

[22] Try taking an SPL reading at home, or worse, at work – you'd be surprised how loud it is in the modern world. I know that every time I come home to Alaska off work travel, I marvel at just how quiet my home is (~27.5 dB_{SPL} A-weighted on a rainy morning).

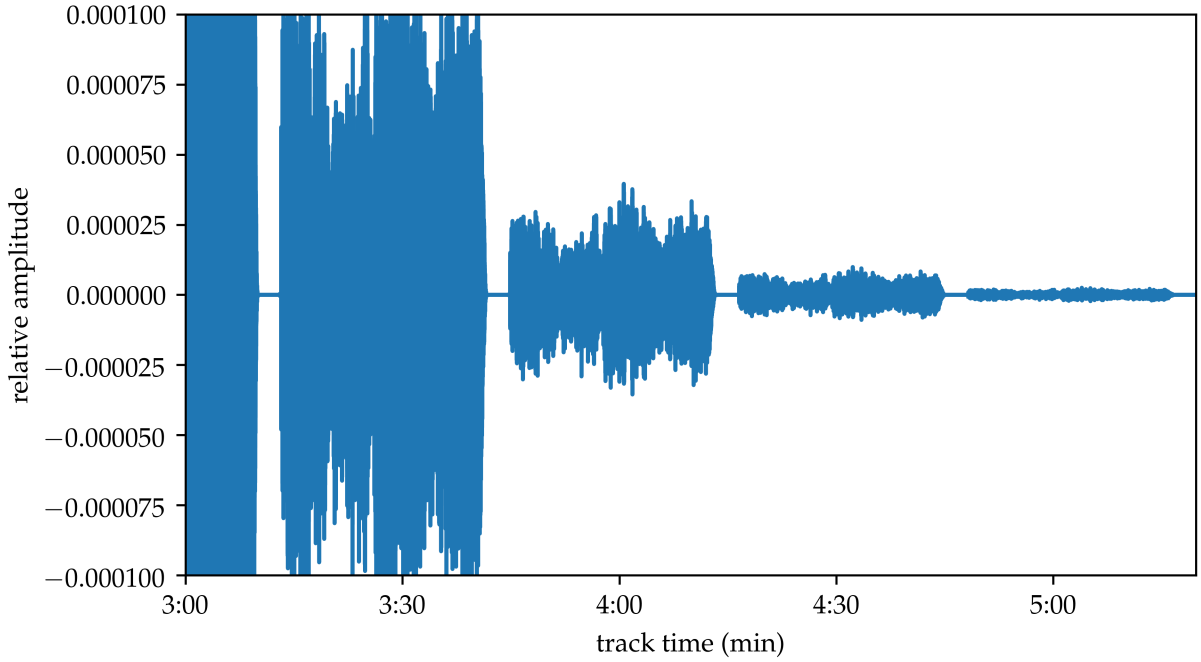

Figure 10: The last few minutes of the left channel of Track [22]. The trace depicts the decreasing volume of each passage. In order to verify that the final passage rises above the digital noise floor (dither), we must zoom in quite a bit on the vertical axis (at full amplitude, the first (-19 dBFS, not shown) passage ranges roughly between ± 0.5, or 4096 times louder than the seventh (-91 dBFS) example that spans the 3:30 mark. Even the tenth segment (-127 dBFS, or 108 dB quieter than the original) is clearly visible in the digital domain. See the text for details.

Track [23]: Test Tone: an industry standard...

`23-testTone.flac  0:20` This track is a long-standing test tone used in the audio industry. It comprises the same 14 s, 1 kHz pure tone (a sine wave) in each channel, faded in/out over the first/last 0.045 s, with 3 s of silence padding on each end. The sine wave is set to ± 0.1 in extremal relative amplitude. This is the same as a -20 $dBFS_{p\text{-}p}$ level – note that this is a different definition than we've been using for all of our previous tracks (see the Computation section for details). A THX-certified system should produce 85 dB_{SPL} with this tone when a voltmeter reads 1 VAC across the speaker terminals. Even without such a system, the tone can be used to ensure normalized volumes for auditioning components ("louder" is often perceived as "better" in listening tests) by ensuring that the output of each device under test is within a 1% of a nominal VAC reading. For example, one DAC under test produces 0.3 VAC with this tone at a volume setting that is appropriate to a particular musical test track. To compare a second DAC, play this tone and adjust the volume until it reads between 0.297 and 0.303 VAC – this should give unbiased results,

at least as far as volume is concerned (provided you've noted each volume level for each DAC). In practice, it's a little more complicated than that, since at frequencies other than 1 Hz the two DACs might vary by more than 1% – still, this one is a standard.[23]

Track [24]: Speaker Polarity: absolute phase...

`24-speakerPol.flac  0:51` While this this track was the only one to be written with separate software (and hardware) in mind, it turns out that most of the previous tracks are also very handy with said software; namely, the AUDIOTOOLS app. This track can be used with a microphone and a speaker polarity tester to determine absolute polarity of a loudspeaker driver, or a multimeter for the same purpose with a piece of audio electronics. In particular, it has been validated to work with the aforementioned AUDIOTOOLS,[24] as well as a variety of speakers and microphones (the internal mic of a smart phone will do just fine) – it may (or may not) function with other polarity testers. Before using it to check your speakers, you should first you should ensure that the electronics in the path to the speakers also have the proper phase.

As we heard on Track [2], *relative* phase is easy to discern and can have obvious deleterious effects on realistic audio presentation. But, as observed on Track [19], *absolute* phase is a more difficult aspect of sound to come to terms with. For continuous tones, it just isn't something we can hear – since the phase of such a tone is a spatiotemporal quantity, varying with not only time but also listening position. For transient sounds (impulsive in nature), perhaps some folks can tell the difference, but it's not clear that they're not actually hearing an aspect of the inherent asymmetry in speaker drivers and their enclosures, as seen from front to back. On a laptop, for example, at the transition from positive to negative polarity pulses, you can easily notice a distinct difference in the tones; however, the sensation diminishes after the transition. This effect is likely due to the very small backing volume of the laptop speakers. This track is different from its phase-related predecessors, in that it is certain to assist in the determination of absolute phase, since it doesn't make use of our ears. It can also come in handy for diagnosing issues of relative phase; e.g., a single driver wired oppositely from its stereo counterpart.

The basic requirement for testing absolute polarity is to send a series of pulses with a constant polarity (exclusively positive or negative speaker driver excursion from its equilibrium position; e.g., toward or away from the listener). Electronically, this is easy to verify with a multimeter, but with a speaker you'll need some dedicated software to analyze the acoustic response of a microphone to this track. We'll use both types of pulse (positive and negative) in order to confirm

[23] Perhaps a somewhat more rational approach would be to employ a full-spectrum level check. One such protocol is the recommended practice of SMPTE RP-200, in which a -20 dBFS (in our standard RMS sense) pink noise signal from a single channel is leveled to produce 83 dB_{SPL} (C-weighted, slow). You can use your speakers and the single channel -17 dBFS pink noise segments from Track 1 to achieve this: a reading 84 dB_{SPL} on your meter set to C-weighting/slow is the equivalent.

[24] Ver. 10.*

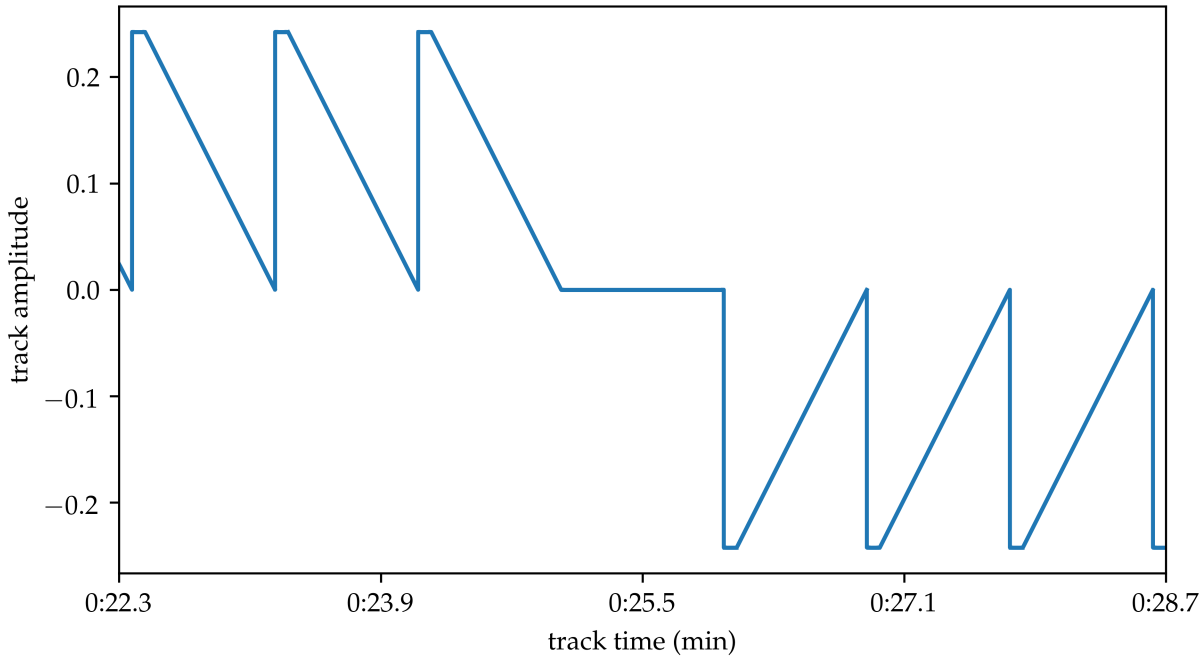

Figure 11: The pulse trains of Track [24] used for testing absolute speaker polarity. Depicted is the portion roughly 25 s into the track, where the transition from positive to negative polarity begins. Note that the tones are scaled to -17 dBFS, well below the ±1 extremal values for safety's sake.

that the polarity reading switches during the track. Most test signals available for this purpose have a square-wave attack, DC sustain and triangular decay. This is not so great for speakers, but this short track will not cause damage at reasonable playback levels. The high frequency content of such sharp transitions is required to assess tweeter phase.

Figure 11 depicts the transition between the two types of pulse trains, approximately 26 s into the track. Individual pulses have a 0.88 duration. A 16 µs cosine-taper attack opens the 0.08 s sustain, which leads to a cosine-tapered fade of 0.8 s. Superposed on the pulse trains is a small-amplitude 135 Hz tone. The entire track is normalized to -17 dBFS, rather than at full scale (or clipped as are some commercial examples), mainly for safety's sake. The first half of the track consists of a series of twenty-five positive pulses; the same number of negative pulses comprise the second half – you should note a distinct change[25] in polarity after roughly 25 s.

[25]It is not uncommon for polarity checker software to take a pulse or two to lock in on the correct polarity, hence a pulse train.

Tracks [25-27]: J-Tests: jitter stimulus at xx bit/yy kHz...

`tt-jTestxxyy.flac 0:36` Each of these tracks comprise a 30 s recreation of the so-called "J-Test" jitter stimulus tone originated by Julian Dunn, padded on each end with 3 s of silence.[26] Dunn formulated a simple test signal to analyze jitter in the output signal from either a physical or numerically-modeled DAC system (cables and all). The J-Test signal comprises two components, both of which are square waves in the digital domain: 1) a relatively high-level component with a period of four samples, and 2) a very low-level component with a period of 192 samples. Upon processing by an ideal DAC, the former is designed to result in a high-amplitude, pure sinusoid of frequency $f_h = f_s/4$ (or a quarter of the sampling rate); the latter, a series of increasingly quiet, low-amplitude, odd-order harmonics of $f_\ell \in [f_s/192, (192/2-1)f_s/192]$. For example, a 16-bit, 44.1 kHz J-Test should produce a strong pure tone at 11.025 kHz, along with a very weak set of tones at 229.6875, 689.0625, 1148.4375, ... 21,820.3125 Hz.

Two of the signal parameters were chosen arbitrarily by Dunn: 1) the ± 0.5, or ~6.02 $dBFS_{p\text{-}p}$ amplitude[27] of the high-level square wave, and 2) and the 192-sample period of the low-level square wave (which was, incidentally, set at the same length as the AES3 channel status block). That the square waves result in theoretical pure tones (sinusoidal waves) is a desired result of the band-limited assumption underlying the sampled data – for finite sampling, square waves result in odd-order harmonics (or modes) limited by the Nyquist frequency, $f_s/2$. A square wave at $f_s/4$ allows only one "harmonic", (the fundamental mode) $f_s/4$, since the 3rd order mode would be $3f_s/4 > f_s/2$, above the Nyquist frequency. So the resultant harmonics expected from an ideal DAC process were selected by design. Moreover, the amplitude of the low-level square wave was specified at the 1-bit level in order to toggle the state of every bit register – the worst possible case for inducing jitter in the data stream. Too, the signal content maximizes predictable coherent patterns in the data stream for subsequent graphical analysis.

Consider a series of 16-bit samples, of amplitude ± 0.5, represented in hexadecimal form for compactness: `C000 C000 4000 4000`. This sequence depicts one period of a square wave at exactly $f_s/4$, regardless of the sample rate. To complete Dunn's J-Test signal, we repeat the previous sequence twenty-three more times, then subtract 1-bit from each of the following 96 samples. One complete, 192-sample period of the combined stimulus signal is then given by the sequence:

`C000 C000 4000 4000` (repeat 23x) `BFFF BFFF 3FFF 3FFF` (repeat 23x).

For these tracks, Dunn's prescription is followed for a 30 s stimulus tone[28] in each of the following resolutions: 16/44, 24/48 and 24/96. The salient characteristics of the 16-bit, 44.1 kHz tone are

[26] A much longer test stimulus might be used in practice, so that the frequency resolution of the analysis might be increased, but the idea is gotten across with this length of a tone.

[27] See the Computation section for the working definitions of dBFS used on this disc.

[28] The 44.1 kHz example is ever so slightly longer than 30 s in order to have an integer number of complete 192-sample sequences.

depicted in the panels of Figure 12. In fact, there's a short course on digital music hiding in that figure alone.

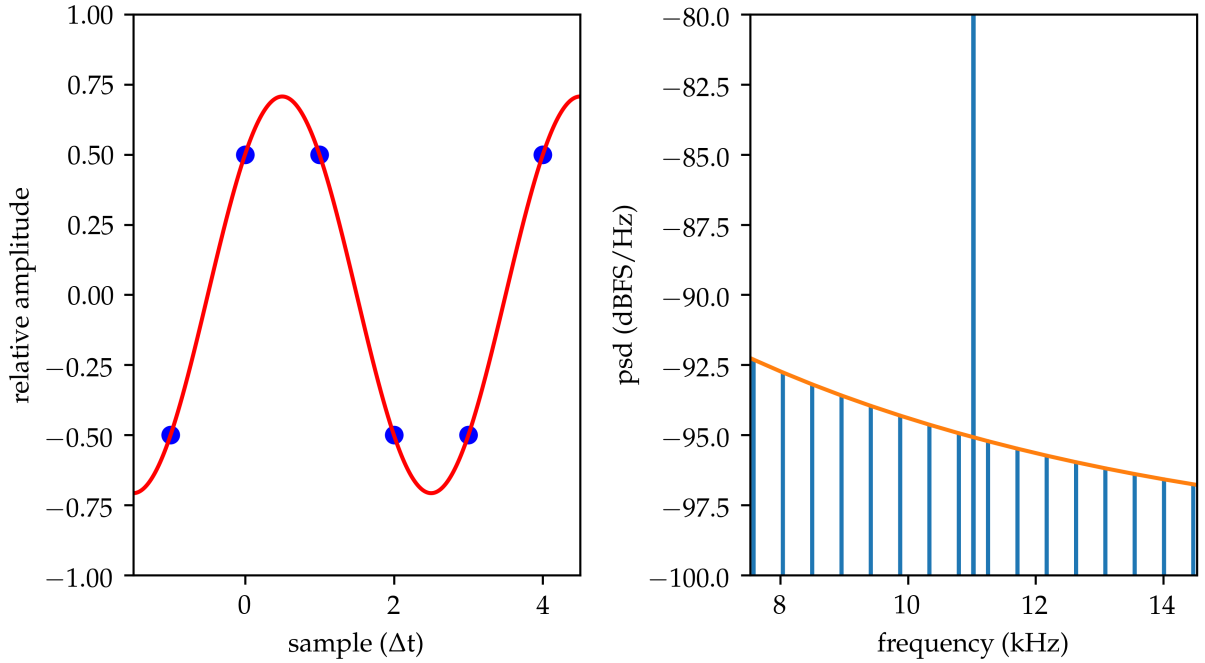

Figure 12: The jitter stimulus tones of Track [25] know as the "J-Test". The left panel shows the actual four-sample square wave at ±0.5 amplitude in blue, with the analogue sinusoid output at ~3.01 $dBFS_{p\text{-}p}$ (note that the amplitude of the continuous fundamental tone is higher than that of the original square wave). In the right panel, we see this high-amplitude tone at exactly 11.025 kHz and the modes within ±3 kHz of it, all in blue. In orange, the predicted amplitude of each is also shown. The vertical scale of the right panel is clipped to show sufficient detail in the low-level modes, so the entire ~28.9 dBFS/Hz central peak is not shown. See text for more detail.

The toggling of the least significant bit (LSB) is designed as a torture test for the DAC chain – it is truly a worst case scenario, as it requires every bit register to change state at $f_s/192$). The predictable[29] harmonics provide a framework in which to graphically analyze the results of a physical or simulated test of the tone. Too, the strong quarter sample rate tone provides its own test of the system. First, the LSB toggle in the low-frequency square wave produces a series of precise, low-level harmonics spread across the entire Nyquist-limited bandwidth of the analogue output. There should be no extraneous signal between the harmonics shown in Figure 12. Of course, a real-world system will exhibit such extraneous content, but so long as it is below the orange curve, it is effectively noise at less that the LSB level; and thus, relatively (if not completely)

[29] A square wave of T samples and amplitude A yields $m = T/4$ discrete modes, where $m \in [1, T/2 - 1]$, each with a Fourier amplitude given by $\frac{A \sin(m\pi/2)}{T \sin(m\pi/T)}$.

innocuous. The central peak, at $f_s/4$, should be clearly resolved, with no shoulder width – it is, after all, a theoretically pure tone; the width of the reproduced central peak is another indicator of the jitter-resistance of the DAC system.

Analyzing this stimulus tone requires some particular attention to detail, as the data must be windowed in a particular way in order not to introduce spurious tones. The reason for this is that the resultant tones are constructed independent of sample rate and so will fall at precisely predictable Fourier frequencies. The window should be rectangular (tapered windows will introduce shoulder frequencies to each tone) and of a length that is an integer multiple of 192. This will ensure that the basis tones are all represented correctly.

Tracks [28-30]: Impulse Responses: impulses at xx bit/yy kHz...

`tt-jTestxxyy.flac 0:36` Each of these tracks comprise a series of impulses, or single-sample data spikes at 0 $dBFS_{p\text{-}p}$, to stimulate a system and reveal its impulse response. A typical use with digital content is to study the response of a DAC filter implementation. The track is almost entirely zeros (digital silence) except for eight samples! The usual 3 s pads of silence are present on each end, and each pulse is followed by 5 s of silence until the next pulse. First, the left channel exhibits a positive (+1 sample) then a negative (-1 sample) spike; this is then repeated in the right channel; finally, in both channels simultaneously.

Many years ago an inadvertent rendition of this track actually lead me to changing out the speakers in my system! I'd long been using a pair of Bose 901 Series V and thought they were perfectly fine – in fact, I thoroughly enjoyed listening to music through them. Then one evening in graduate school, I was listening to a CD-R I'd ripped from an LP of the Guarneri Quartet belting out Beethoven's Op. 133 "Große Fuge". My wife wasn't too appreciative of the music at 12:45 am and said it sounded like I was listening to a recording of "four alley cats fighting,"[30] so I switched to a pair of Sennheiser HD580 headphones. Every minute or so I was disturbed to hear a distinct "ping" in the music stream, but I was tired and soon fell asleep. The next day, before my wife got home from work, I isolated one of the annoying pings via the A-B repeat function of my Denon CD player at the time. While it was clearly audible in the headphones, it was completely absent in the Bose reproduction!

Once back in the lab at the university, I read the samples into MATLAB and found that there were a series of, somewhat regularly spaced, one- to three-sample dropouts (sample values of 0) in the data stream.[31] While the inverse of what's intentionally created on this track, it has the same effect of sending an impulse to the playback system. The CD player's DAC produced the

[30] About four years later we saw the Julliard Quartet perform it live in Fairbanks, Alaska and she *loved* it – "It has to be seen to be heard properly!" she exclaimed.

[31] Turns out that a few CD-Rs that a luddite grad school buddy had ripped on a Windows PC were the cause of the 0-valued samples, a consequence of buffer under-runs. Linux rips were, for the record, bit perfect!

impulse response, including a lot of high frequency content in the "ping", which the Sennheisers were easily capable of reproducing. But the array of 4 inch drivers (the output signal having been processed by the Series V Active Equalizer required of those speakers), on the other hand, were not. This was a clear example of how Bose relied on psychoacoustics to get the brain to fill in some missing high frequency information. When I demonstrated the effect to my wife, she said that she, too, could now not not hear what was missing. A few months later, we had a pair of Magenpan 2.7QRs in the listening room!

Disc 2: Th' DSD files

When I first started this project, writing tracks to a file in the DSD format wasn't something I knew how to do. While it appears that `sox` now has a development fork that allows for DSD conversion to more popular formats (e.g., WAV and FLAC), it's still not clear to me if it will *write* DSD files. No PYTHON package that I've come across supports this audio file format either. Enter the TASCAM Hi-Res Editor (ver. 1.02), which allowed me to convert between a PYTHON-authored WAV format to DSD. So, in this roundabout fashion, I now have DSD capability in my ken.

The DSD tracks on Disc 2 were each first written as a WAV (`.wav`) file and then converted "by hand" to DSD (as `.dsf` files) via the TASCAM editor.[32] Since there is no easy way to make equivalency statements between PCM resolution (bit depths/sample rates) and DSD rates, I've chosen a reasonable approximation here in constructing the DSD files. The DSD64 example was converted from a WAV file with native 24-bit/88.2 kHz resolution; similarly, DSD128 from 24-bit/176.4 kHz WAV. Note that these particular WAV resolutions were also selected since the `xx` in DSD`xx` indicates the sample rate multiplier over the Red Book (CD) specified 44.1 kHz.

You'll note that not all of the FLAC tracks of Disc 1 are rendered in the DSF file format. In principle, the TASCAM editor could simply be used to replicate[33] them all – that said, I've only included a few examples I've found useful as a DSD test disc.

Tracks [1]–[2]: DSD Rates: DSD`xx`, `yy` MHz...

`tt`-bitRatesDSD`xx_yy`M.dsf `m:ss` These two tracks are similar to Tracks [9]-[16] on Disc 1 – they're for testing whether or not your DAC correctly identifies and decodes the DSD files in their native format. It is also possible to test whether or not your playback software will pass native DSD to your DAC, pre-process the data and pass it via DoP (DSD over PCM), or perform a full conversion to PCM for non-DSD capable DACs.

[32] Since these tracks are dead simple to construct and there is a non-trivial (read *PITA*) "hand-rolled" aspect to them (unlike the FLAC side of the house which is done with a single makefile), they are not rebuilt each time the package is updated. Therefore, you may notice that the DSD dates and version numbers lag a bit behind their FLAC counterparts.

[33] As of ver. 1.02 the editor has the capability of outputting DSD64, DSD128 and DSD512.

track no.	filename	duration	DSD level
[1]	01-bitRatesDSD64_2.8M.dsf	0:37	64 (2.82 MHz)
[2]	02-bitRatesDSD128_5.6M.dsf	0:41	128 (5.64 MHz)

Each track is constructed using the same recipe as Tracks [9]-[16], but the synthetic voice announces both the DSD rate[34] and the bit depth/sample rate of the original WAV source at the beginning and end of each track. These tracks are detailed in the table, above, and can be used to confirm how your playback software and DAC treat a DSD stream.

Track [3]: DSD Test Tone: DSD64, 2.8 MHz...

03-jTestDSD64_2.8M.dsf 0:20 This track is prepared exactly as is Track [23] on the FLAC disc; however, it is sourced from a 24-bit, 88.2 kHz WAV file and converted to DSD via the TASCAM editor.

Track [4]: DSD J-Test: DSD64, 2.8 MHz...

03-jTestDSD64_2.8M.dsf 0:36 This track is prepared exactly as are Tracks [25]-[27] on the FLAC disc; however, it is sourced from a 24-bit, 88.2 kHz WAV file and converted to DSD via the TASCAM editor.

Track [5]: DSD Impulse: DSD64, 2.8 MHz...

04-impulseDSD64_2.8M.dsf 0:36 This track is prepared exactly as are Tracks [28]-[30] on the FLAC disc; however, it is sourced from a 24-bit, 88.2 kHz WAV file and converted to DSD via the TASCAM editor.

[34] Once more, perhaps you'll notice I didn't construct all possible DSD rates – there's a bias here toward what my equipment will actually handle; which is to say, only DSD64 and DSD128. Frankly, the former sounds pretty good on some albums to me.

Computation

Synthetic waveforms were created using PYTHON[35] 3.7.1 and several digital signal processing methods written by the author in a package known as `WATCtools`. Waveform analysis, in both the the time- and frequency-domain, to ensure that the tracks are doing what they are supposed to be doing, was also performed using various methods written by the author. These were supported by the brilliant (and ubiquitous) NUMPY 1.15.4 and SCIPY 1.1.0 packages. All tracks were written to either FLAC or WAV (the latter to prepare for the DSD tracks) files using the SOUNDFILE 0.10.2 package and tagged via the command line tools in the `flac` 1.3.2 distribution. All of the files were synthesized or recorded at 24-bit resolution, sampled at 96 kHz, unless otherwise noted. Tagging and cover art for the FLAC files were done with a bit of `tsch` scripting, and those were copied by hand to the DSD files using `kid3` 3.6.2 . Audacity 2.1.1 was used to record my voice at some point, via the internal microphone of a 2015 15″ Retina MacBook Pro (why the hell did they hide that thing, anyway?). The synthetic voice was generated by the OS X command line `say` tool, using the voice *Karen*.

Most of the signals present in this collection are normalized to the -17 dBFS level,[36] so there shouldn't be any significant chance of damage to your speakers when reproduced at realistic listening levels. That said, it's important to note which definition for dBFS we're working with here. Since we're using noise for the evaluation of loudspeakers and rooms, the energy content of the signal is more important that the amplitude. This lends itself to the root-mean-square (RMS) definition of dBFS, rather than the peak-to-peak definition of the Audio Engineering Society's Standard AES17-1998, or $\text{dBFS}_{\text{p-p}}$. So what's used in all of these files is the RMS definition, such that a square wave ranging the full scale of ± 1 would give 0 dBFS. A monochromatic sinusoid in this definition can only rise to roughly -3 dBFS without clipping[37] (i.e., it is full scale). A 0 dBFS signal in our definition would be quite loud in playback and likely be clipped (i.e., exceed ± 1 in amplitude). By comparison, a -17 dBFS pure sinusoid in our definition will range between $\pm\sqrt{2} \cdot 10^{\frac{-17}{20}} \approx \pm 0.2$ and so they are realistically quiet. Noise, on the other hand, is only statistically "constrained" in a range, but in practice the signals present on the Test CD are not clipped.[38] Incidentally, some DACs may actually clip a peak-to-peak 0 dBFS signal (in *either* deinfition), depending upon how carefully the analog conversion section is built. The amplitude span of ± 1

[35] This version of the Test CD should be considered to have superceded that of ver. 1.05 and previous editions, which were written in proprietary software (MATLAB). This edition represents a shift to open source software for the entire authoring and production run. Tracks 1-16 on this newer, PYTHON edition are essentially identical to the originals.

[36] Tracks based on the *Open Goldberg Variations*, [19]–[21], were left unnormalized; however, their content was verified to be < -19 dBFS in the RMS definition. Track [22] was set to -19 dBFS, as explained in the description. Only Track [23] was prepared according to the AES standard, at -20 $\text{dBFS}_{\text{p-p}}$. See individual track notes for other deviations from the -17 dBFS standard.

[37] A pure sinusoid of unit amplitude would be exactly $-10\log_{10}(2)$ dBFS in the RMS definition.

[38] In fact, -17 dBFS was chosen because it is highly unlikely that a representative ensemble of colored noise will clip in a few minutes time at 96 kHz sampling. That each of the tracks were not clipped was verified during the production run via a simple PYTHON method written by the author; i.e., the production code checks each file to ensure it is within the peak-to-peak specification of ± 1, or 0 dBFS in the AES definition.

for full scale is in arbitrary, dimensionless units and is commonly used to scale the time series for FLAC and WAV files in software packages like Audacity. All the actual AES17-1998 $dBFS_{p\text{-}p}$ levels for each track and channel are provided in the following table.

	$dBFS_{p\text{-}p}$			$dBFS_{p\text{-}p}$			$dBFS_{p\text{-}p}$	
trk no.	L ch.	/ R ch.	trk no.	L ch.	/ R ch.	trk no.	L ch.	/ R ch.
1	-1.7	-1.7	13	-2.9	-2.9	25	-6.0	-6.0
2	-0.7	-0.7	14	-2.9	-2.9	26	-6.0	-6.0
3	-8.5	-8.5	15	-3.0	-3.0	27	-6.0	-6.0
4	-13.7	-13.7	16	-3.2	-3.2	28	0.0	0.0
5	-13.0	-13.0	17	-2.3	-2.3	29	0.0	0.0
6	-13.0	-13.0	18	-2.4	-2.4	30	0.0	0.0
7	-13.0	-13.0	19	-4.0	-4.0	D2 1	-1.8	-1.8
8	-2.3	-2.3	20	-1.9	-1.9	D2 2	-2.0	-2.0
9	-2.9	-2.9	21	-2.5	-2.5	D2 3	-20.0	-20.0
10	-2.8	-2.8	22	-4.1	-4.1	D2 4	-6.0	-6.0
11	-2.8	-2.8	23	-20.0	-20.0	D2 5	0.0	0.0
12	-2.8	-2.8	24	-12.3	-12.3			

You can see there's quite some difference from our fiducial -17 $dBFS_{rms}$. Even a single extremal sample (near ± 1) will govern the $dBFS_{p\text{-}p}$ value. So in using our dBFS definition and level, we've sacrificed a few bits of potential resolution from our 24/96 source files, but we've ensured that clipping is not present. Bottom line, the AES definition is a good one to prevent clipping, while our definition is more meaningful in terms of the volume level of complex signals (i.e., musical passages). You can always check for clipping after setting a reasonable average (i.e., rms) signal level – and we do!

Looking closely at Figure 2, you might wonder how a -17 dBFS C_8 note in Track [3] could rise to nearly +20 dBFS/Hz. For spectral amplitudes in a PSD, the convention is to express them in terms of $units^2/Hz$, where "units" are the dimension of the time series (e.g., in acoustics pressure amplitudes are reported in Pa, so the PSD would have units of Pa^2/Hz). Since our signals are dimensionless (they're merely constrained to ± 1) we use dBFS/Hz to scale our PSDs, whether as spectra or spectrograms. Since this is a dBFS level *per frequency bin* (i.e., a spectral density), it is important to specify what we're referencing this to. We use white noise at 0 dBFS, since its statistics are well known in both the time and frequency domains. Its PSD will have the same amplitude in dBFS/Hz at all frequencies and is easy to calculate. So for all of the spectra and color bars, the level indicates the relative amplitude of the signal in that frequency bin compared to what 0 dBFS white noise would be in the same bin.[39]

[39] You know the nice looking plots you see in the measurements section of hi-fi publication's equipment review, the ones where the spectral peak for a pure tone of frequency f at -20 $dBFS_{p\text{-}p}$ shows up at precisely -20 dBR (dB relative to

The entire Test CD (including documentation, but not the hand-rolled portions required to convert the WAV files to DSD) takes about seven or eight minutes to produce from a single `tcsh` script on a reasonably modern laptop.[40] Please note that the plots in the booklet are generated from the actual FLAC files on the Test CD. Since some of these tracks[41] contain realizations of noisy data or pseudo-random sequences , they will necessarily change each time the code is run. They won't be qualitatively different, but they will be quantitatively so.

Disclaimer & Acknowledgement

This Test CD is presented for entertainment purposes only. The views expressed herein are solely those of the author and are certainly not those of the Wilson Alaska Technical Center, Geophysical Institute, nor the University of Alaska Fairbanks. This work was unsupported and carried out wholly during the author's free time (such as it is) and on his personal equipment. The user assumes all risk for damage to their ears, pets, loved ones, psyche, speakers, headphones and/or audio equipment. For further reading, and to see part of the inspiration for this project, please refer to John Atkinson's wonderful *Stereophile* Test CD 3 (`stereophile.com/features/424`); and please note that he gave me his "thumbs up" to distribute my version. You should also get yourself a copy of the Andrew Smith's AUDIOTOOLS app and a Studio6Digital calibrated microphone (I use their compact *iTestMic2* to great effect) to fully appreciate and make use of the tracks presented here. The author offers this to the audio community wholly without warranty or any promise (hope) of technical support. Have fun with this, and please remember it's all about enjoying the music, *not* just your equipment!

This work is dedicated to Roxy Music, whose *Avalon* was the first CD I had, and is still probably my favorite album.

0 $\mathrm{dBFS_{p\text{-}p}}$)? These are produced with so-called "boxcar" windows of length N, such that they satisfy both $N = N_{fft}$ and $(N_{fft} f)/f_s \in \mathcal{N}$, a natural number. Given this, the spectral components P, of say the Welch method, may be normalized to dBR as $20 \log_{10} \sqrt{2Pf_s/N_{fft}}$.

[40] To be precise, a MacBook Pro (15-inch, 2016), with a 2.6 GHz Intel Core i7, SSD main storage and 16 Gb of memory.

[41] This really only affects Figs. 1-2, 5 & 8 quantitatively, but not qualitatively, since they contain a different realization of colored noise each time the production code is run. Similarly, the table of $\mathrm{dBFS_{p\text{-}p}}$ values will change slightly with each run for any track containing colored noise.

License

Curt A. L. Szuberla
Wilson Alaska Technical Center
Geophysical Institute
University of Alaska Fairbanks

Track [3]: Chromatic Scales

		left channel		*right channel*	
note	***Hz***	***up time***	***down time***	***up time***	***down time***
C1	32.703	00:03.0	01:52.2	01:55.9	03:45.0
C1♮	34.648	00:03.7	01:51.6	01:56.5	03:44.4
D1	36.708	00:04.3	01:50.9	01:57.2	03:43.7
D1♮	38.891	00:05.0	01:50.3	01:57.8	03:43.1
E1	41.203	00:05.6	01:49.6	01:58.5	03:42.4
F1	43.654	00:06.2	01:49.0	01:59.1	03:41.8
F1♮	46.249	00:06.9	01:48.3	01:59.8	03:41.1
G1	48.999	00:07.5	01:47.7	02:00.4	03:40.5
G1♮	51.913	00:08.2	01:47.0	02:01.0	03:39.9
A1	55.000	00:08.9	01:46.4	02:01.7	03:39.2
A1♮	58.270	00:09.5	01:45.7	02:02.4	03:38.5
B1	61.735	00:10.2	01:45.1	02:03.0	03:37.9
C2	65.406	00:10.8	01:44.4	02:03.7	03:37.2
C2♮	69.296	00:11.5	01:43.8	02:04.3	03:36.6
D2	73.416	00:12.1	01:43.1	02:05.0	03:35.9
D2♮	77.782	00:12.8	01:42.5	02:05.6	03:35.3
E2	82.407	00:13.4	01:41.8	02:06.2	03:34.6
F2	87.307	00:14.1	01:41.2	02:06.9	03:34.0
F2♮	92.499	00:14.7	01:40.5	02:07.6	03:33.3
G2	97.999	00:15.3	01:39.8	02:08.2	03:32.7
G2♮	103.826	00:16.0	01:39.2	02:08.9	03:32.0
A2	110.000	00:16.6	01:38.6	02:09.5	03:31.4
A2♮	116.541	00:17.3	01:37.9	02:10.2	03:30.7
B2	123.471	00:18.0	01:37.3	02:10.8	03:30.1
C3	130.813	00:18.6	01:36.6	02:11.4	03:29.4
C3♮	138.591	00:19.3	01:36.0	02:12.1	03:28.8
D3	146.832	00:19.9	01:35.3	02:12.8	03:28.1
D3♮	155.563	00:20.5	01:34.7	02:13.4	03:27.5
E3	164.814	00:21.2	01:34.0	02:14.1	03:26.8
F3	174.614	00:21.9	01:33.3	02:14.7	03:26.2
F3♮	184.997	00:22.5	01:32.7	02:15.3	03:25.5
G3	195.998	00:23.1	01:32.1	02:16.0	03:24.9
G3♮	207.652	00:23.8	01:31.4	02:16.7	03:24.2
A3	220.000	00:24.4	01:30.8	02:17.3	03:23.6
A3♮	233.082	00:25.1	01:30.1	02:17.9	03:22.9
B3	246.942	00:25.8	01:29.5	02:18.6	03:22.3
C4	261.626	00:26.4	01:28.8	02:19.3	03:21.6
C4♮	277.183	00:27.1	01:28.1	02:19.9	03:21.0
D4	293.665	00:27.7	01:27.5	02:20.6	03:20.3
D4♮	311.127	00:28.4	01:26.9	02:21.2	03:19.7

Track [3]: Chromatic Scales

		left channel		*right channel*	
note	***Hz***	***up time***	***down time***	***up time***	***down time***
E4	329.628	00:29.0	01:26.2	02:21.9	03:19.1
F4	349.228	00:29.6	01:25.6	02:22.5	03:18.4
F4♮	369.994	00:30.3	01:24.9	02:23.2	03:17.8
G4	391.995	00:30.9	01:24.3	02:23.8	03:17.1
G4♮	415.305	00:31.6	01:23.6	02:24.4	03:16.4
A4	440.000	00:32.2	01:23.0	02:25.1	03:15.8
A4♮	466.164	00:32.9	01:22.3	02:25.8	03:15.1
B4	493.883	00:33.5	01:21.6	02:26.4	03:14.5
C5	523.251	00:34.2	01:21.0	02:27.1	03:13.8
C5♮	554.365	00:34.8	01:20.3	02:27.7	03:13.2
D5	587.330	00:35.5	01:19.7	02:28.3	03:12.5
D5♮	622.254	00:36.1	01:19.1	02:29.0	03:11.9
E5	659.255	00:36.8	01:18.4	02:29.7	03:11.3
F5	698.456	00:37.5	01:17.7	02:30.3	03:10.6
F5♮	739.989	00:38.1	01:17.1	02:30.9	03:09.9
G5	783.991	00:38.8	01:16.5	02:31.6	03:09.3
G5♮	830.609	00:39.4	01:15.8	02:32.3	03:08.6
A5	880.000	00:40.1	01:15.2	02:32.9	03:08.0
A5♮	932.328	00:40.7	01:14.5	02:33.6	03:07.3
B5	987.767	00:41.3	01:13.9	02:34.2	03:06.7
C6	1046.502	00:42.0	01:13.2	02:34.8	03:06.0
C6♮	1108.731	00:42.7	01:12.5	02:35.5	03:05.4
D6	1174.659	00:43.3	01:11.9	02:36.2	03:04.8
D6♮	1244.508	00:44.0	01:11.2	02:36.8	03:04.1
E6	1318.510	00:44.6	01:10.6	02:37.5	03:03.5
F6	1396.913	00:45.2	01:10.0	02:38.1	03:02.8
F6♮	1479.978	00:45.9	01:09.3	02:38.8	03:02.1
G6	1567.982	00:46.6	01:08.7	02:39.4	03:01.5
G6♮	1661.219	00:47.2	01:08.0	02:40.1	03:00.8
A6	1760.000	00:47.8	01:07.3	02:40.7	03:00.2
A6♮	1864.655	00:48.5	01:06.7	02:41.4	02:59.5
B6	1975.533	00:49.2	01:06.0	02:42.0	02:58.9
C7	2093.005	00:49.8	01:05.4	02:42.7	02:58.2
C7♮	2217.461	00:50.5	01:04.8	02:43.3	02:57.6
D7	2349.318	00:51.1	01:04.1	02:44.0	02:56.9
D7♮	2489.016	00:51.7	01:03.5	02:44.6	02:56.3
E7	2637.020	00:52.4	01:02.8	02:45.2	02:55.6
F7	2793.826	00:53.1	01:02.1	02:45.9	02:55.0
F7♮	2959.955	00:53.7	01:01.5	02:46.6	02:54.4
G7	3135.963	00:54.3	01:00.8	02:47.2	02:53.7

Track [3]: Chromatic Scales

		left channel		*right channel*	
note	***Hz***	***up time***	***down time***	***up time***	***down time***
G7♮	3322.438	00:55.0	01:00.2	02:47.8	02:53.0
A7	3520.000	00:55.7	00:59.6	02:48.5	02:52.4
A7♮	3729.310	00:56.3	00:58.9	02:49.1	02:51.7
B7	3951.066	00:57.0	00:58.2	02:49.8	02:51.1
C8	4186.009	00:57.6	--	02:50.4	--

Track [4]: Warble Tones I

Hz	*time*
20.0	00:05.0
35.6	00:09.2
63.2	00:13.3
112.5	00:17.5
200.0	00:21.6
355.7	00:25.8
632.5	00:29.9
1124.7	00:34.1
2000.0	00:38.2
3556.6	00:42.4
6324.6	00:46.5
11246.8	00:50.7
20000.0	00:54.8

Hz	*time*
20000.0	00:58.8
11246.8	01:03.0
6324.6	01:07.1
3556.6	01:11.3
2000.0	01:15.4
1124.7	01:19.6
632.5	01:23.7
355.7	01:27.9
200.0	01:32.0
112.5	01:36.2
63.2	01:40.4
35.6	01:44.5
20.0	01:48.7

Hz	*time*
200.0	01:55.7
112.5	02:04.0
63.2	02:12.3
35.6	02:20.6
20.0	02:28.9

200.0	02:35.9
355.7	02:44.2
632.5	02:52.5
1124.7	03:00.8
2000.0	03:09.1

2000.0	03:16.1
3556.6	03:24.4
6324.6	03:32.7
11246.8	03:41.0
20000.0	03:49.3

Tracks [5][6][7]: Warble Tones II, III IV

Hz	*Hz*	*Hz*	*time*
201.6	200.0	2000.0	00:03.0
160.0	252.0	2519.8	00:16.9
127.0	317.5	3174.8	00:30.9
100.8	400.0	4000.0	00:44.8
80.0	504.0	5039.7	00:58.8
63.5	635.0	6349.6	01:12.8
50.4	800.0	8000.0	01:26.7
40.0	1007.9	10079.4	01:40.7
31.7	1269.9	12699.2	01:54.6
25.2	1600.0	16000.0	02:08.6
20.0	2015.9	20158.7	02:22.5

Track [21]: Bit Resolution I

eff. bits	*time*
24	0:03.0
16	0:34.7
8	1:06.3
4	1:38.0
2	2:09.7
1	2:41.4

eff. bits	*time*	*time*
24	3:13.0	3:41.0
23	3:13.7	3:40.3
22	3:14.4	3:39.7
21	3:15.1	3:39.0
20	3:15.8	3:38.3
19	3:16.4	3:37.6
18	3:17.1	3:36.9
17	3:17.8	3:36.2
16	3:18.5	3:35.6
15	3:19.2	3:34.9
14	3:19.9	3:34.2
13	3:20.5	3:33.5
12	3:21.2	3:32.8
11	3:21.9	3:32.1
10	3:22.6	3:31.5
9	3:23.3	3:30.8
8	3:24.0	3:30.1
7	3:24.6	3:29.4
6	3:25.3	3:28.7
5	3:26.0	3:28.1
4	3:26.7	3:27.4

Track [22]: Bit Resolution II

dBFS $_{p\text{-}p}$	***time***
-19	0:03.0
-31	0:34.7
-43	1:06.3
-55	1:38.0
-67	2:09.7
-79	2:41.4
-91	3:13.0
-103	3:44.7
-115	4:16.4
-127	4:48.0