The Filter Wizard
issue 13: Buenos Notches! The Filter Wizard versus the vuvuzela
Kendall Castor-Perry

When the insistent drone of massed vuvuzela first imposed itself on the world during televised world Cup matches, I thought "Huh! Well, they'll soon filter *that* out". But soon it became clear that removing this narrow-band noise was presenting challenges to broadcast engineers.

Figure 1 shows an FFT of a 1.024-second chunk of vuvuzela-dominated crowd noise. Pronounced peaks in the spectrum are at frequencies consistent with what's reported elsewhere: fundamental at ~230Hz, with plenty of harmonics. The third harmonic at ~700Hz is the most prominent component. That's consistent with the 'buzzy' quality of the sound.

A single vuvuzela produces a fundamental and harmonics with a narrow occupied bandwidth. Hundreds of them, each played in a unique on-off pattern, will give a broadened spectrum equivalent to a single 'carrier' modulated by a random signal with Gaussian shape. So the elimination problem is more than removing a single tone and its harmonics; we have to suppress several bands of noise.

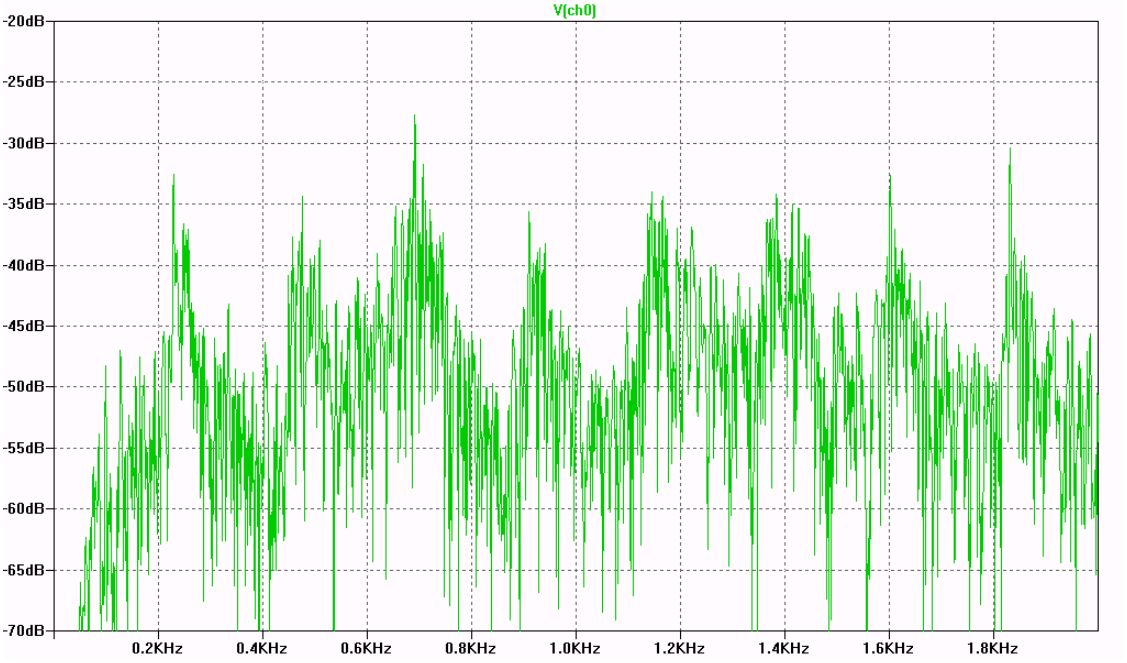

figure 1: FFT of a sample of vuvuzela

The starting choice for notching multiple related frequencies out is often a lowpass comb filter. It's a two-tap FIR filter where you add the input signal to a delayed version of itself. For any input frequency at which the delayed term causes a multiple of 180 degrees phase shift, the input and delayed signals cancel out. A deficiency of this method

here is that the resultant notches only occur at *odd* multiples of the first notch frequency, shown in figure 2.

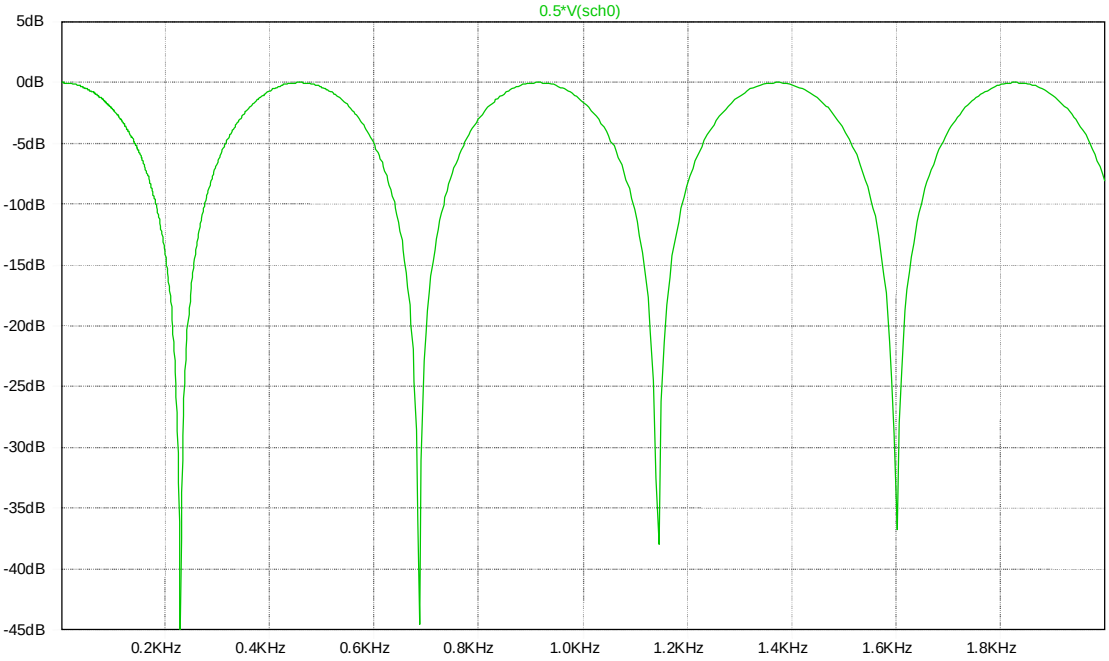

figure 2: lowpass comb filter

In our case, this method leaves the even order harmonic bands unattenuated, and they now stick out like a sore thumb (or should that be a sore throat?), see figure 3.

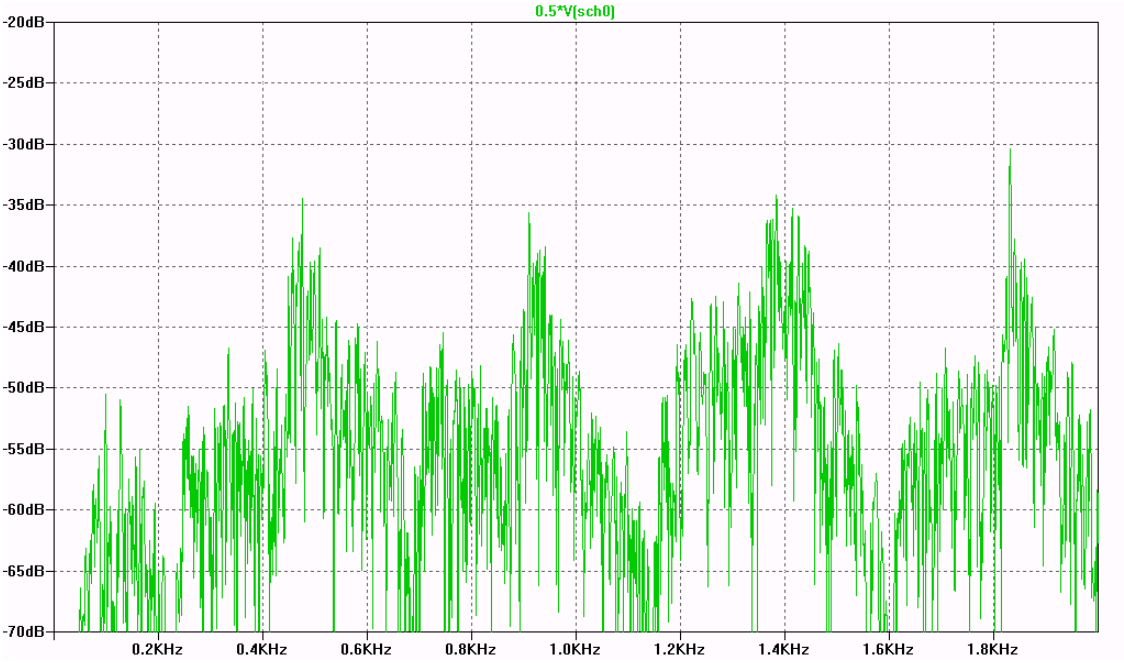

figure 3: lowpass comb only attenuates fundamental and odd harmonics

If you take the *difference* between input and a delayed signal, you get a *highpass* comb filter. If we make the delay equal to the period of the fundamental, we get notches at *every* multiple of the fundamental. We also get a notch at zero frequency, so we'll lose the low-frequency component of the stadium roar.

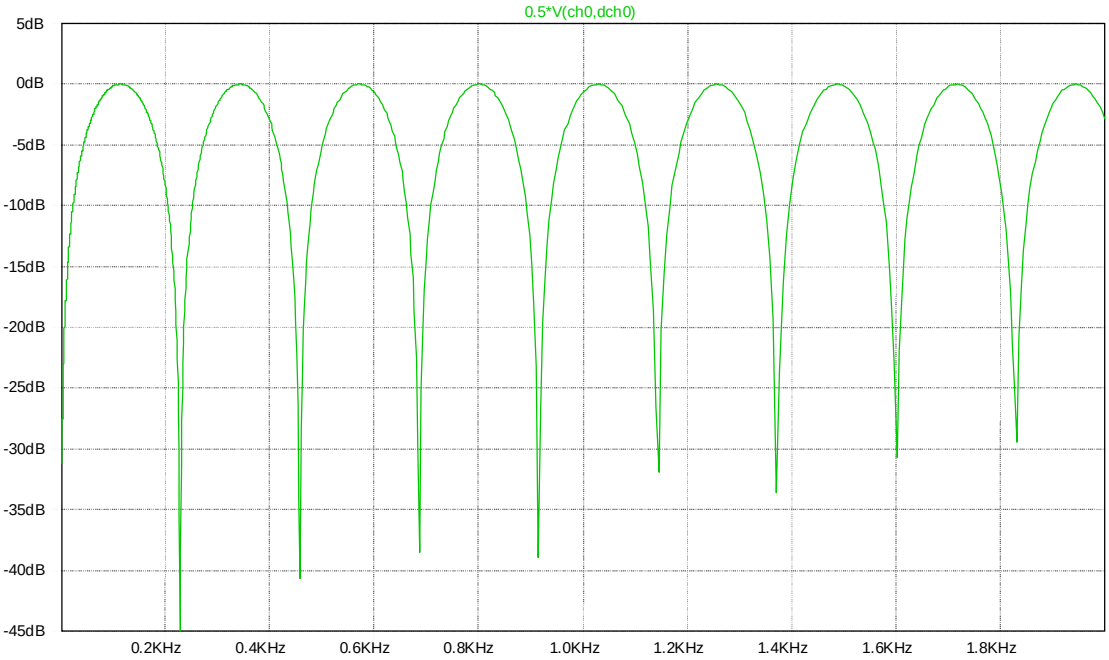

figure 4: highpass notch to remove all components

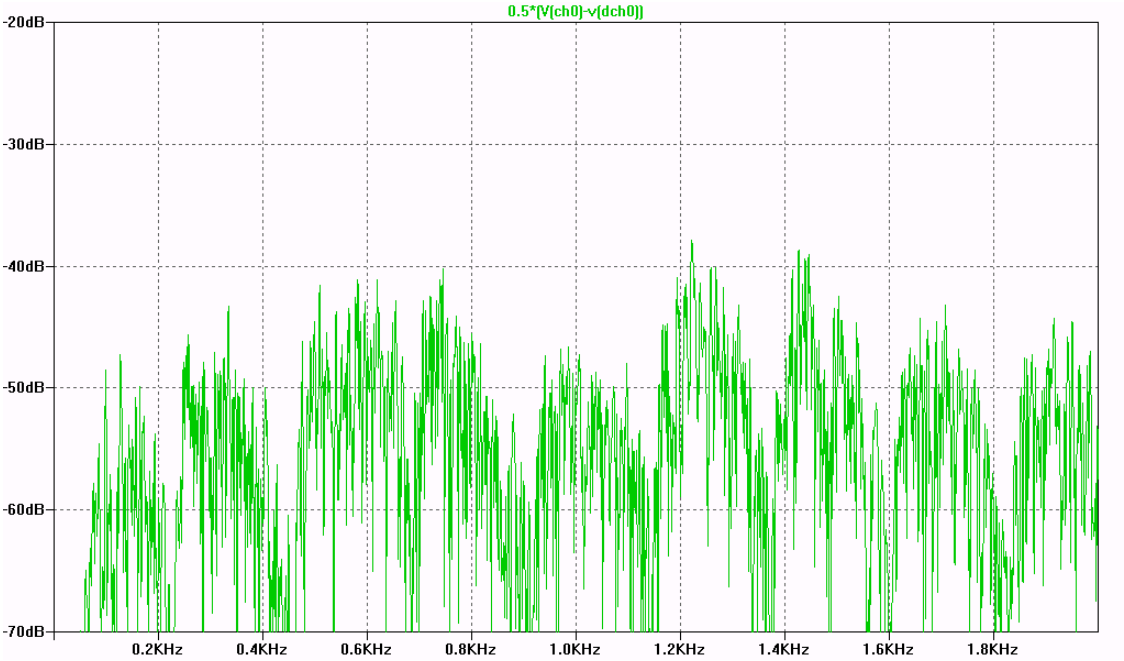

figure 5: spectrum after a highpass comb filter

The resulting sound with the highpass comb is 'lighter' because we lost the bass, but there's still a clear, annoying tonal quality to it. The problem is that near the frequencies that we're trying to remove, these cancellation notches are narrow. They are good at getting rid of single tones. But when the offending signal is broadened, they leave behind residuals that still sound like tones.

Let's look at 'proper' notch filters. Creating notch filters with single stopbands is part and parcel of the toolbox of the filter designer. Such a filter has two separated passbands, with a stopband sitting in the gap between those passbands. The order of the filter determines how "sharp" we can make the notch – note that this is different from how *deep* it is.

We could address the task of devuvuzalation by designing a notch filter for each of the offending frequency bands, estimating attenuation levels and passband widths from our FFT analysis. We'd connect all these filters in series, being careful that the transition bands of the two filters didn't overlap, or we'd not get a very satisfactory response between notches.

Figure 6 shows a simple example, using the simplest possible notch filter, a 2nd order biquad. I put eight of them in series, tuned to 230Hz and the first seven harmonics. Because the width of each bulge in the FFT plot is constant on a linear scale, I've made the Q's of the individual notch transfer functions proportional to centre frequency.

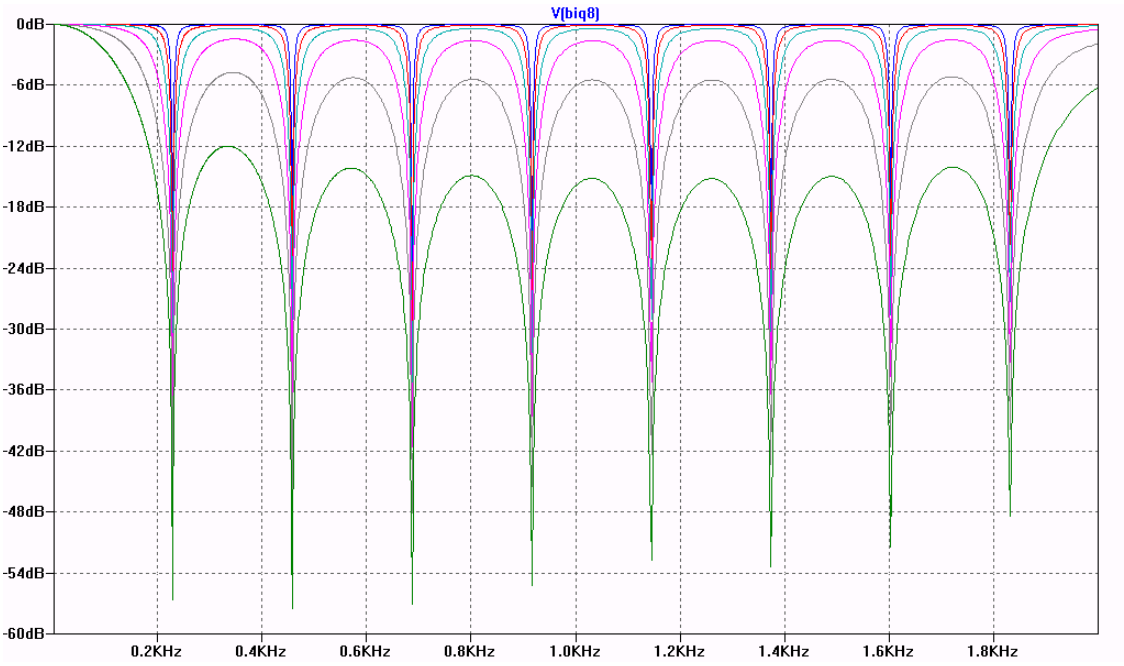

figure 6: 2nd order notch filters in series, for various Qs

The plots show the response with the Q of the fundamental's notch stepped down from 20 to 0.625 in factors of 2. The high Q responses are even narrower than the comb filter responses, while the low Q notches lose us a lot of the (presumably valuable) spectrum between the harmonics. Sure enough, when the high Q value is used, the narrow notches are not enough to remove a significant amount of the energy in the broad bands around the harmonic, while the low Q values suppress the signal between the notches too much.

Now, forgive me for my laziness, but I decided not to design eight 'proper' notch filters, each with nice wide stopbands and sharp transitions to the passband, to fix this problem. Certainly, by devoting say an eighth order transfer function to each frequency, we could deliver an impressive filter. But the resulting function would be 64^{th} order per channel, and would be a handful to implement on any reasonable signal processor. And I really want this filter to be able to run (in stereo) on the processor I'm currently working with, the Digital Filter Block in Cypress's new PSoC3/5 family. I reckon I can afford about twenty biquads per channel, so I'll restrict myself to that.

Of course, instead of viewing this as a problem of creating multiple *notches* to *remove* the interference, we could consider this as a requirement for multiple *bandpass* filters to *pick out* the useful bits in between. This doesn't look any easier if we limit ourselves to standard filter design techniques. But there's a class of filters called n-path filters that could be pressed into service. While sampling of a sort is involved with these filters, they can be built in the analogue domain as well. Decades ago, I used them to create stable narrow-band analogue bandpass filters. But – and I'm sorry to have whetted your appetite again – I'm not going to design one of those here either. They merit a whole article on their own.

Let's look at the problem on the psychoacoustic side. Do we actually need to eliminate the offending frequencies completely? Auditory masking theory suggests that to two tones (or tone-like bands of noise) can only be discriminated if they are different enough in either frequency separation or amplitude ratio. A corollary of this is that if we can 'flatten' the spectrum of the signal (also called 'whitening'), we can't discern a tonal quality to the content of any frequency band.

To do this, we run the signal through a filter whose magnitude response is the reciprocal of the spectrum of the signal itself. In practice, we need to approximate this, to make a realizable filter. Doing it with an FIR filter is feasible, but in my chosen processor I only have 128 taps available, and I can tell that this is way too few to do such a low frequency filter at audio sample rates. One very effective way of doing it is with an IIR-based audio equalizer, either graphic, paragraphic or full parametric. A paragraphic equalizer has bands at fixed frequencies but whose Q can be adjusted to suit the response. Now, it just so happens that for a project at Cypress, I designed an algorithm that forces a paragraphic equalizer's response to pass exactly through a list of frequency response points.

Figure 7 shows a smoothed FFT plot of the vuvuzela noise. This was sampled at 115Hz intervals, and inverted to give the required frequency response. Normally an audio paragraphic equalizer would have bands centred on a logarithmic scale, but clearly we'll

be better off with linearly-spaced bands here. I used an equalizer with 20 frequency bands spaced at 115Hz intervals (I think 17 would have been enough). The compensation algorithm calculates both the gain and the Q value of the filter sections. Figure 8 shows the resultant response of the equalizer.

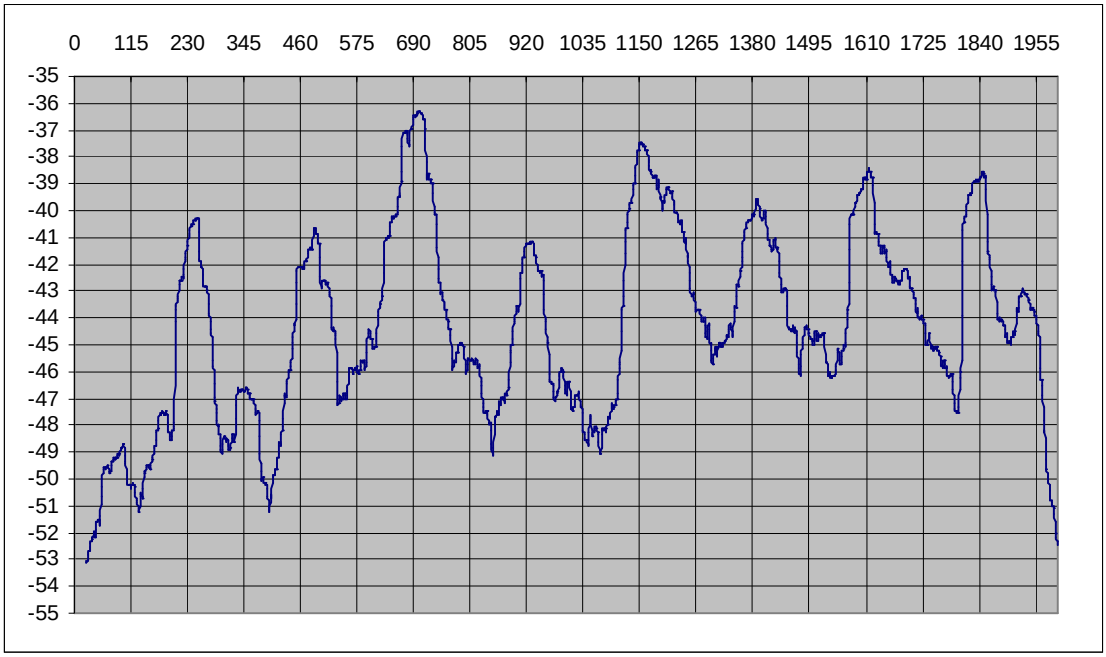

figure 7: smoothed FFT plot really shows up the vuvuzela bands

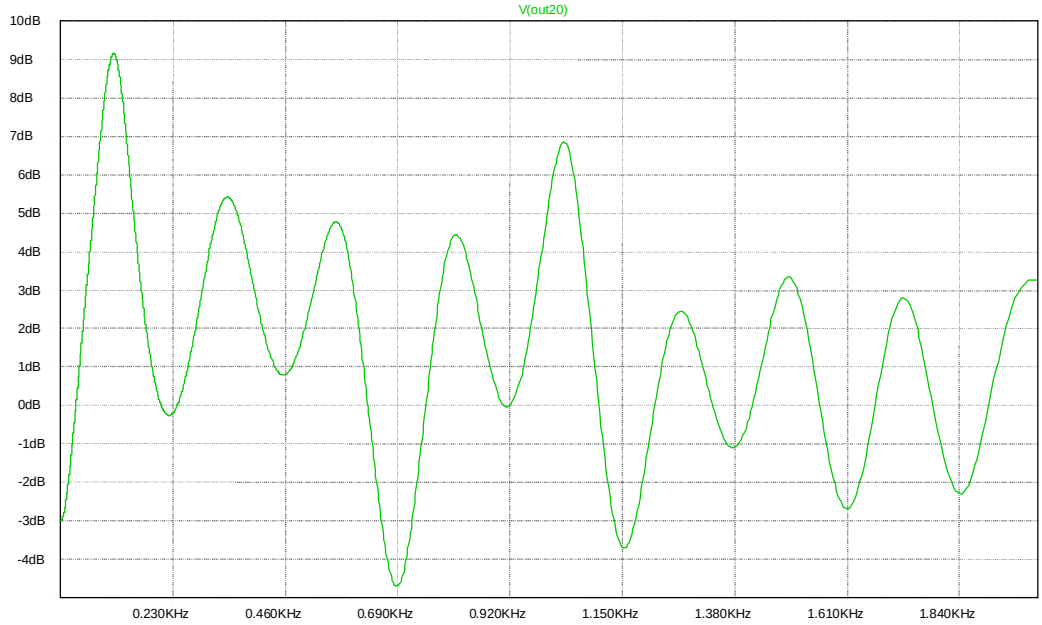

figure 8: the response of the paragraphic set up to invert figure 7

The equalizer certainly makes the frequency spectrum of the signal more uniform, see figure 9. I scaled the equalizer gain to make the rms value of the output equal to the input. This means having a loss at some vuvuzela frequencies, but a gain at frequencies in between. Of course, the response is only 'correct' for one particular spectrum. We could contemplate making it dynamically adjustable; the response inversion algorithm can run in real time on the PSoC3.

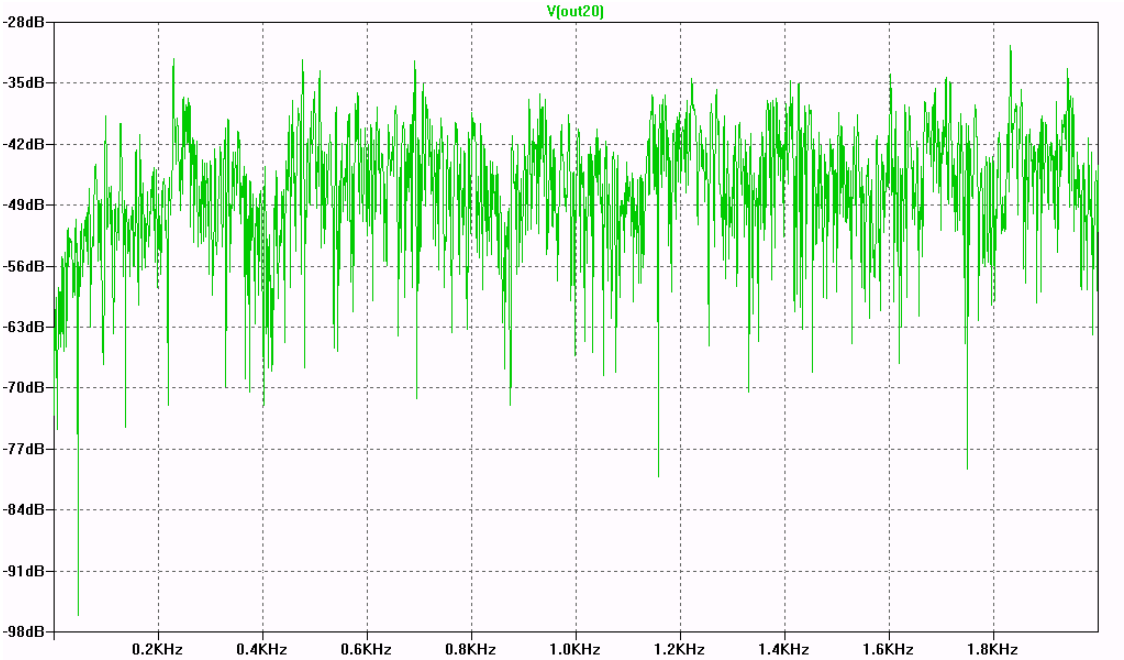

figure 9: equalizer-whitened FFT plot

What does it sound like? Well, not bad, on the test file used. There's a clear sense of stadium background noise, but with a much less persistent tonal quality. Sadly, I won't be able to get the implementation on the PSoC3 development kit done in time for the final, a few days away as I write. Still, happy notching! best - Kendall