

Re: randomized white noise = white noise?

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2006-05/msg05038.html>

- *From:* "Tom Bruhns" <k7itm@xxxxxxx>
 - *Date:* 26 May 2006 15:38:00 -0700
-

I'll try to answer two posts at once here...

Note that when a rolloff slope changes from 3dB/octave to 6dB/octave, it's a gradual thing. You can draw lines on a Bode plot that are straight and intersect at a sharp corner, but in practice, the transition is gradual. So the difference in response between just the 3dB/octave and the added 6dB/octave in the top octave will be relatively small. In fact, if you get Scilab and run the simple code I suggested, you can see a plot of the response—easy to plot the response of your filter, or the difference between it and "true pink." And what you will see with your original coefficients is that they cause the high end to drop off faster than "pink" anyway, by just about what you are suggesting with that "6dB slope in the top octave". So you may have that already. In fact, in Scilab (or Matlab) it's easy to make most of what I posted be a routine that will plot the response or whatever, and you just change the coefficients and run the routine to see the effect. Even with a slow processor, it should do the calcs in a second or less, especially if you used fewer points logarithmically spaced, but even if you do that one point per Hz hack I posted, it's still fast.

OK, have a look at the relationships in your filter to see why you get large values pretty often. Notice that b2, for example, remembers over half its previous value, and adds MORE than one times the input to itself. So let's say you have three white samples in a row that are in the range 0.5 to 1.0 (assuming that white runs from -1 to +1). Note that:

$$\begin{aligned} b2(k+1) &= 0.57*b2(k) + 1.05*white(k) \\ &= 0.57* (0.57*b2(k-1) + 1.05*white(k-1)) + 1.05*white(k) \\ &= \\ &= 0.57*(0.57*(0.57*b2(k-2)+1.05*white(k-2))+1.05*white(k-1))+1.05*white(k) \\ &= 0.185*b(k-2)+0.34*white(k-2)+0.60*white(k-1)+1.05*white(k) \end{aligned}$$

If the average value of white = 0 (no DC bias), then on average b(k-2) will be zero. A string of three "whites" in a row at 0.5 or above will then, on average, result in an output from just THAT equation that's greater than 1.

Re: randomized white noise = white noise?

The way around this is to scale "white" down, and don't let it cover the whole range from -1 to +1. How much do you scale it? The absolute worst case would be if white stayed at +1 (or -1) for a long time. For your filter (which technically is three one-pole resonators), you can easily get this value: first you find the value for each of the "b" values, then add them plus 0.1848 for the current "white" value that's in the output. How do you get the value for each "b"? Simple: it will reach steady state when the contribution from the "old" b decays by exactly the amount of the new input. So for b0, $\text{white} * 0.099 = (1 - 0.99765) * b0(\text{steady-state})$. So $b0(\text{steady-state}) = 42.12 * \text{white}$. That is, if the input is locked at 1, b0 will eventually get to 42.12. So an absolutely safe scaling would be to not let "white" get above $1/42.12$. Now in practice, the probability of "white" staying that close to full scale in one direction for long enough to do that is so tiny (and in fact would be zero for any practical pseudorandom number generator) that you can safely make the scaling larger than that. For b2, the safe scaling (by a similar calculation) would be 0.408.

Note that since you are dealing with NOISE here, if you can gracefully handle the overflow, letting it happen once every few thousand samples is likely not going to even be noticeable. I suspect that if you scale the white input by 0.25, you will be fine. Try it and see how that works. Your filter probably does a very decent job of turning the uniform amplitude distribution into a Gaussian amplitude distribution, so the mere fact that you see about 3 or 4 out of 10 that's larger than the range of the WHITE input tells me that the standard deviation is quite close to full scale. Then the scaling I suggested, 0.25, should get you to about one in 10,000 that overflows. Note that you do not have to SHIFT your white; you can just duplicate the top bit into the next two bits (for a signed value), and it will still be white. The bits in a uniform-amplitude-distribution random number should be all uncorrelated, and it doesn't matter which bits you drop.

Cheers,
Tom

.