ELECTRONOTES 212

Newsletter of the Musical Engineering Group

1016 Hanshaw Road, Ithaca, New York 14850

Volume 22, Number 212 December 2012

CONTENTS OF EN#212

THE SOUND OF A BOUNCING BALL

-by Bernie Hutchins

INTRODUCTION

This is mainly for fun, and it did not work out the way I had hoped. The idea is that we have all heard the sound of a bouncing ball. By this I mean the sounds as the ball strikes the floor. Two things to note: (1) We only hear the sounds of the individual floor strikes (each strike a discrete sound) and (2) it is not periodic in the real case. At first the strikes are loud and widely spaced in time, then they come faster and softer, both the result of dissipative loss of energy. And it's not just balls of course. A flashlight on its side on my desk is setting so that it vibrates in that same general manner, eventually buzzing to silence. We see this every day. And we understand it very well in general terms.

The typical ball bouncing event occurs over many seconds, starting with perhaps a second or so between bounces, and then the ball bounces less high, and strikes at a faster rate. Eventually it does stop completely.

THE MODEL

Here we will take the case of a bouncing ball that is pretty much ideal except each time it strikes the floor, it loses a certain fraction of its energy (probably eventually to heat). The real case in the real world is very similar to this, and every child and probably every dog is familiar with this fascinating event.

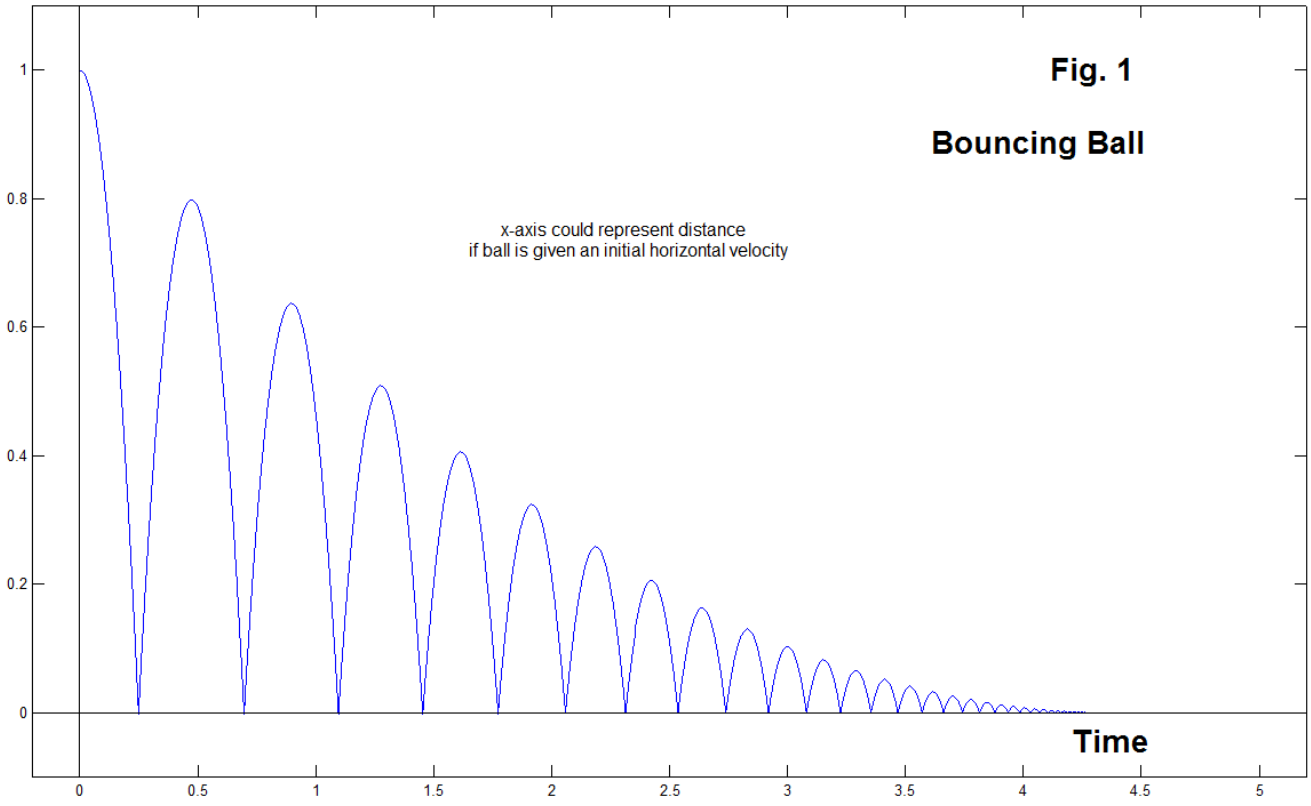


Figure 1 shows a Matlab simulation, a numerical integration starting with just Newton's second law. Here we have assumed that everything is ideal except when the ball bounces off the floor it has only 80% of the energy with which it struck the floor. Note the first full bounce is to a height of 0.8, the second to 0.64, and so on. Here unlike many decaying signals we have studied in the past, the width of the bounces is not constant (periodic except for amplitude changes) but decreases as the amplitude decreases. In fact, the zeros are at 0.2495, 0.6965, 1.0960, 1.4530, 1.7720, 2.0570, 2.3115, 2.5385, 2.7410, 2.9215, 3.0825, 3.2265, 3.3550, 3.4695, 3.5715, 3.6620, 3.7425, 3.8145, 3.8785, 3.9350,…. which is in pretty

good agreement with the theory below. [The simulation interval was 0.0005.] So this looks basically right. Are these sinusoidal lobes? Nope.

THE SIMPLE PHYSICS:

What happens is that we release the ball at a height of h feet (normalized to 1 in Fig. 1). It falls starting with zero velocity, accelerating with rate $\mathrm{g} = 32\ \mathrm{ft/sec^2}$ until such time as it strikes the floor. Upon release, it had a potential energy (relative to the floor) of mgh where m is its mass. At the floor, this has all become kinetic energy and a certain fraction, r, of this is lost during the collision. So it leaves the floor with a smaller velocity (and opposite direction) than it struck it, and will decelerate to zero velocity at a height less than the initial h. At this point, the exact same steps repeat with the new height as a start.

We start with the very first equation of college physics:

$$s = \left(\frac{1}{2}\right) g t^2 \tag{1}$$

where $\mathrm{s} = \mathrm{h}$. Thus t_1, the time at which the ball first strikes the floor is:

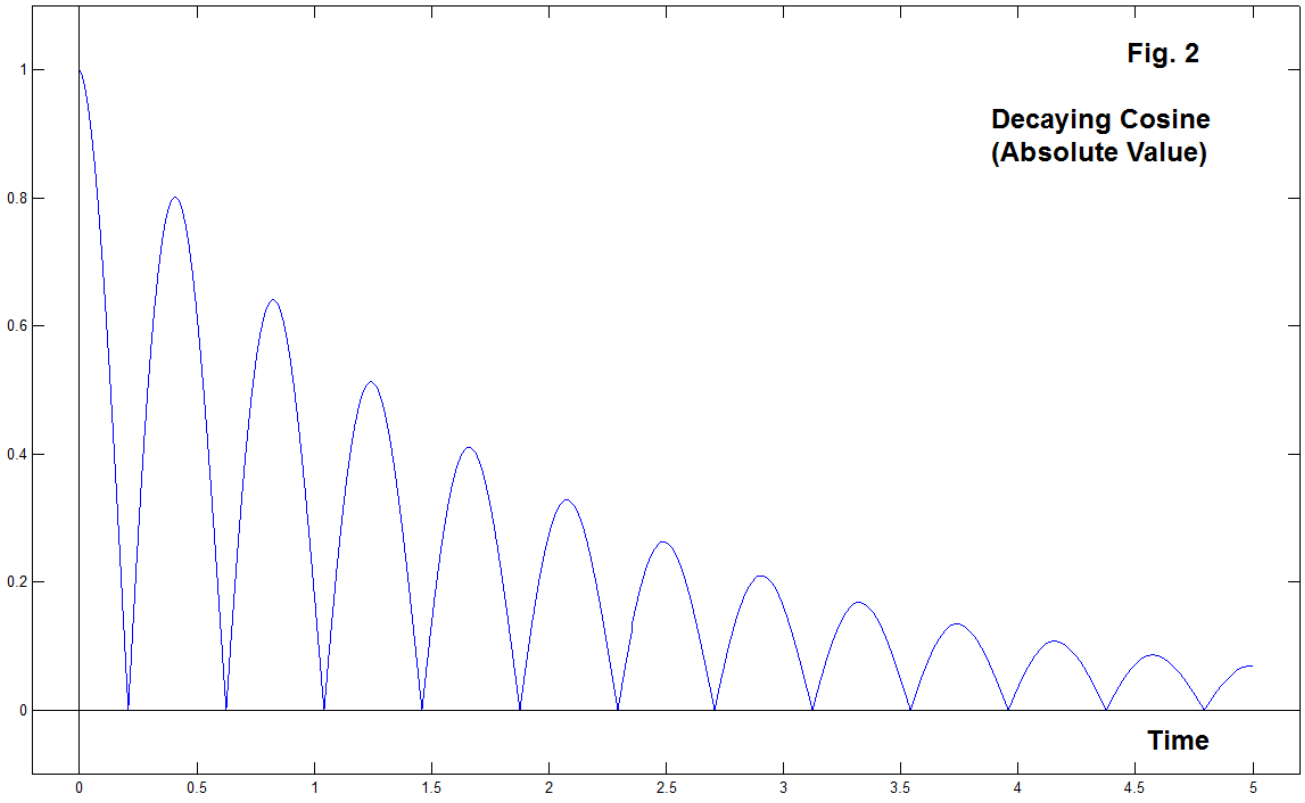
$$t_1 = \sqrt{\left(\frac{2h}{g}\right)} \tag{2}$$

It is convenient to represent the energy during the first bounce as $mgrh$, so the height of the first bounce is rh. The time to fall from the top of the first bounce back to the floor is $(2rh/g)^{1/2}$, which is the same time it took to rise to the height rh, Hence the time of the first bounce is $2(2rh/g)^{1/2}$. The time for the nth bounce (the zeroth bounce, n=0, is represented only by its second half, the initial drop) is:

$$t_n = 2\sqrt{\frac{2r^n h}{g}} \tag{3}$$

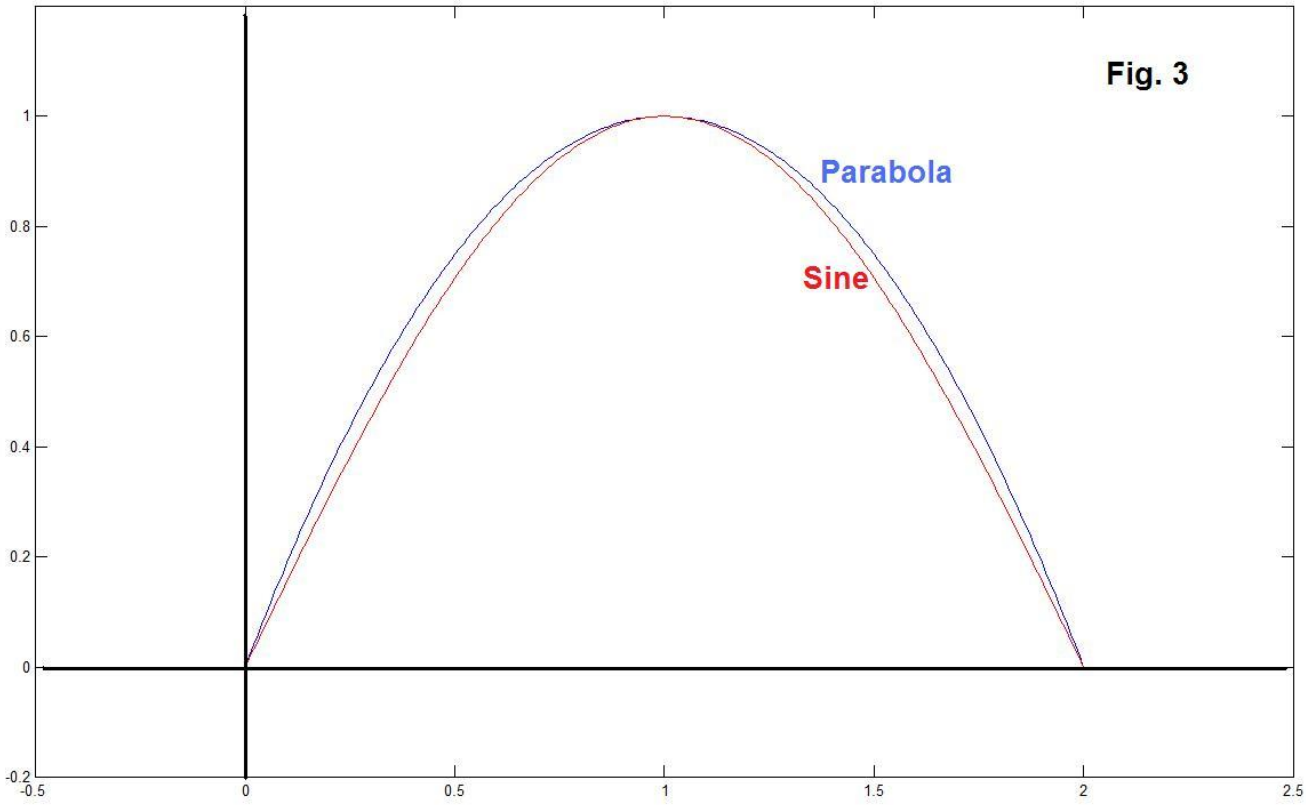
Using equation (3), we compute the widths of the lobes. The first lobe comes out to 1/2, and half of this is really on the negative time side, so we replace it with 1/4. Then we sum the $t_n$for the locations of the zero crossings which give the series: 0.2500, 0.6972, 1.0972, 1.4550, 1.7750, 2.0612, 2.3172, 2.5462, 2.7510, 2.9342, 3.0980, 3.2445, 3.3756, 3.4928, 3.5977, 3.6915, 3.7754, 3.8504, 3.9175, 3.9775. . . . in good agreement with the results of the simulation (this second set of numbers right here are of course better).

So the physics looks good. Note from equation (1) that the lobes are parabolas. So not only are the zeros not equally spaced, but the lobes are not sinusoidal. To better make this point, Fig. 2 shows (the absolute value of) a decaying sinusoidal waveform (actually a cosine of course) manipulated to resemble Fig. 1, but note that the zeros are equally spaced. Just to sharpen the point a bit more, Fig 3 shows two lobe shapes: the parabola and the sinusoidal lobes fit to the points (0,0), (1,1), and (2,0). They are not the same, although the difference is likely minor. If this were a waveshaping, the parabola represents a very small amount of 3rd harmonic distortion (with even less higher odd-harmonic distortions).



So while the shape of the lobes is similar, these are not that closely related due to the tighter spacing of zeros in the case of the bouncing ball. Indeed, we have not really made an issue of the "x-axis" in these graphs. While we understand it to be time in all cases, in Fig. 1, the case of a bouncing ball, it is hard not to think of it as distance. That is, we toss the ball slightly horizontally while releasing it, as we would likely do when playing with a dog. If we just drop it, it goes up and down and we have to think that it is time that is as moving. In the case of the decaying sinusoidal waveform, it is hard to think of the "x-axis" as distance. It represents a harmonic oscillator with damping. Thus it might be a spring with a mass moving on a straight line, with friction. It only moves along a line, and this displacement is almost certainly interpreted as a function of time.

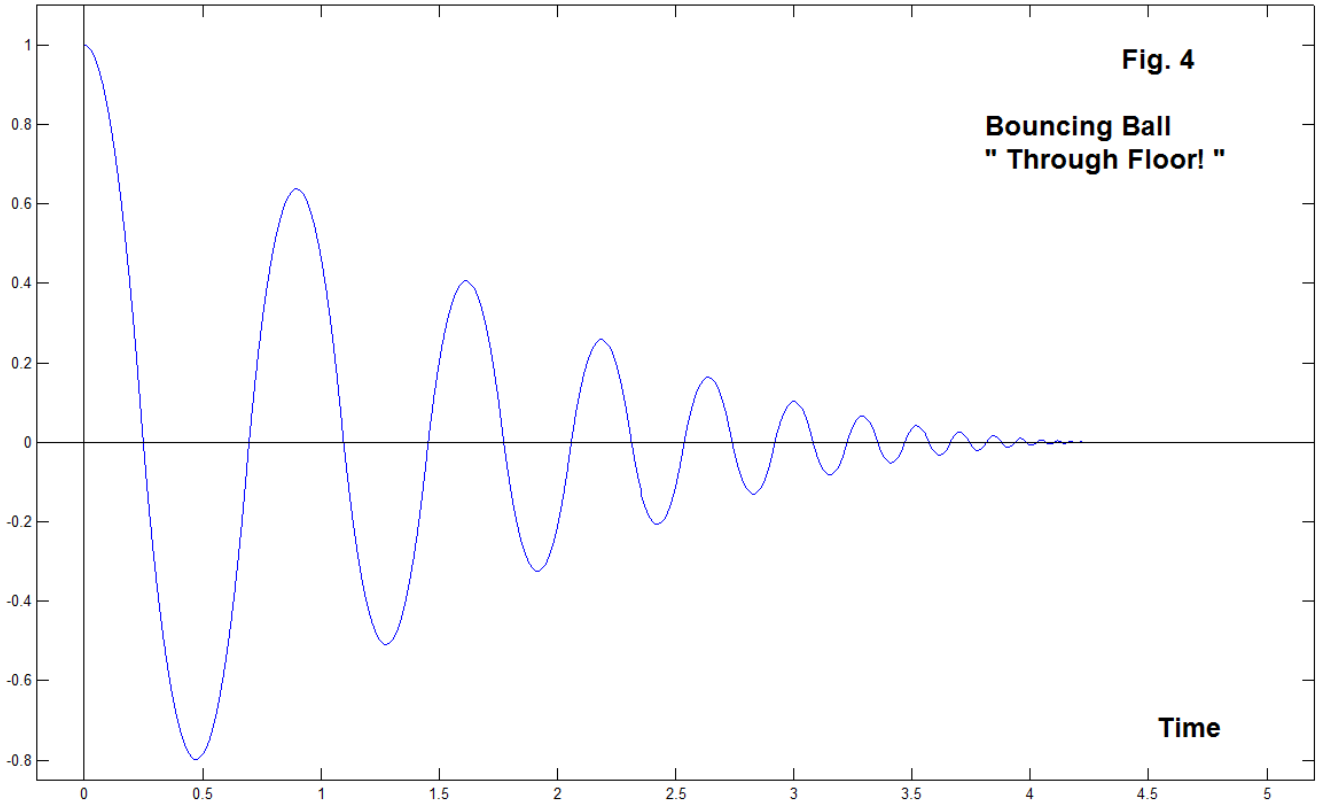
There is the additional difficulty that a harmonic oscillator does not “bounce” when it strikes the zero of the vertical axis. Thus the absolute value interpretation of the harmonic oscillator is a difficult physical concept (mechanically at least). On the other hand, the ball clearly does bounce so as to naturally remain positive.



BOUNCING “THROUGHT THE FLOOR” – AND SPECTRAL ASPECTS

Our initial question was what the bouncing ball sounds like. Here we assume that the waveform is presented at a rate that should be audible directly, and not as a series of discrete strikes. While we don’t anticipate anything much harmonically (Fig. 3), and the effect of the absolute value is similar (Fig. 1 and Fig 2), if we do listen to the two, they are vastly different, for an obvious reason: the frequency seems to be changing, rising as time progresses in the case of the bouncing ball. The decaying sinusoidal waveform is a ”pluck” or “ping” or “bongo” while the bouncing ball is a “chirp” or “whoop” type of sound. As is often the case, the exact effects depends on the spectral content that is present within an appropriate time window for the human ear. To get some idea of this, we of course try the FFT (Fast Fourier Transform or DFT).

Before looking at the spectral considerations, we will find it useful to consider the bizarre concept of the ball "bouncing through the floor." Another way to look at it is to suppose that we "Un-Absolute-Value" the bouncing ball. Every other instance of a bouncing lobe is allowed to go negative (Fig. 4 – compare to Fig. 2). Why not – we are making this all up after all. If you have been staring at plots of sinusoidal waveforms for many many years, the plot of Fig. 4 does in fact look "over rounded" as would be suggested by Fig. 3. Our goal at this point is to simply consider the ringing waveform of Fig. 4 as a new type of resonator to investigate.



The four plots in Fig. 5 show the damped sinusoidal (damped harmonic oscillator) in blue and the parabolic lobes of the bouncing ball (in red) including bipolar and absolute-value versions of both. The absolute-value plots are shown by the solid lines. The bipolar plots are shown as dotted lines and are overlapped by the solid lines in half the cases in an obvious way. As mentioned, the "through-floor" bouncing ball is a strange inclusion in our study, but mathematically perfectly acceptable.

The generation of the "through floor" case, and incidentally, the generation of all the graphs is shown as the program ***BounceBall.m*** included below.

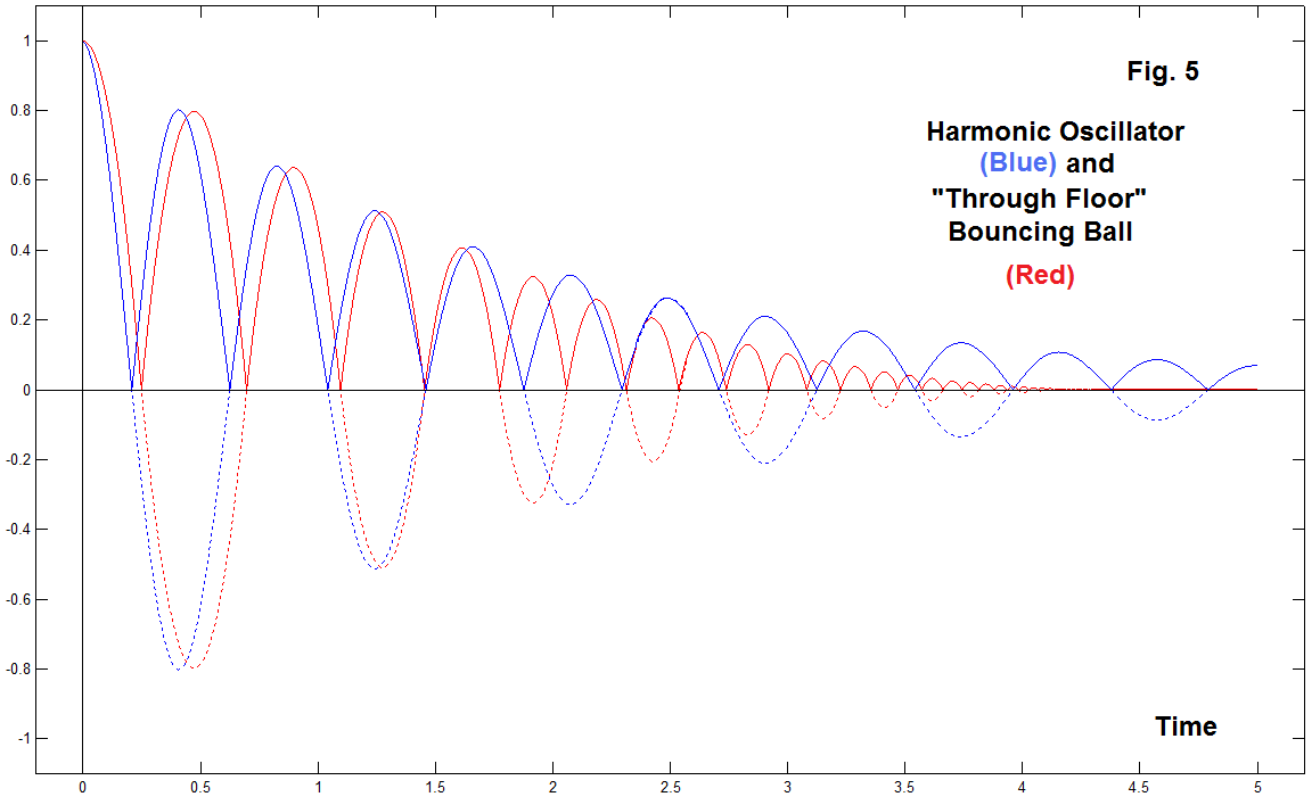


Fig. 5
Harmonic Oscillator
(Blue) and
"Through Floor"
Bouncing Ball
(Red)
Time

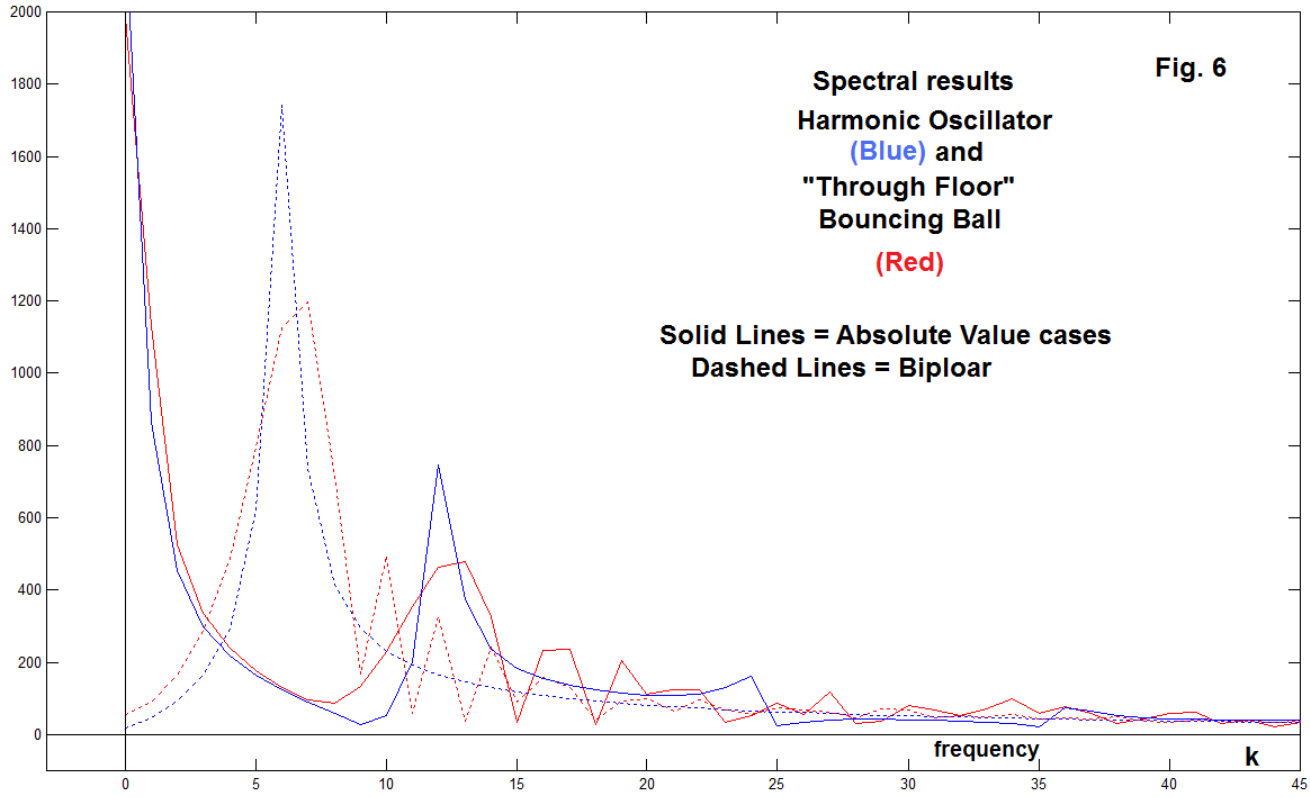


Fig. 6
Spectral results
Harmonic Oscillator
(Blue) and
"Through Floor"
Bouncing Ball
(Red)
Solid Lines = Absolute Value cases
Dashed Lines = Biploar
frequency
k

In Fig. 6, using the same code for the plotted lines as we had in Fig. 5, we show the corresponding spectral information. Here we took FFT's of all 10000 time points of the waveforms and plot only the first 45 frequency points (magnitude FFTs).

Here are the things we need to note about Fig. 6. First, we dispense with any concern for the large dc values for the two absolute value cases (solid curves) which are due to the large dc bias of course. Secondly, we note the there are peaks in the absolute value cases somewhere around k=12. For the two bipolar cases (dotted curves) the peaking is at a frequency of k more like 6, half that value. Recall that k=6 would correspond to something that "wiggled" about 6 times total in the time record, and k=12 to something that wiggled about 12 times total, and we see that Fig. 6 is consistent with Fig. 5, most easily understood from the damped sinusoidal case (blue curves), but generally evident. Thus, the absolute value cases have frequency peaks at about twice the frequency of the bipolar. [This is exactly the same observation and for the same reason that a full-wave rectifier in a 60 Hz power supply shown a ripple frequency of 120 Hz.]

The final observation is that the spectrum for the bipolar cases, the red dotted curve (bouncing ball) is wider than the blue one (harmonic oscillator), and this we understand in terms the bouncing ball having a sweep of frequency (a broader spectrum in general). It is this wider spectrum, which was expected, relative to the standard ringing "plink" of the damped harmonic oscillator (resonator) that we hoped would lead to interesting new sounds. This remains to be demonstrated.

There are several issues here which readers are invited to explore. One is the relative difficulty of constructing the waveform of the bouncing ball. We can do it by simulation and by the physics formulas of course. But it is not clear that there is a "device" that does this, as there is in the case of a harmonic resonator. A more subtle problem perhaps is that issue alluded to above concerning the spectral window – what does the ear hear? Here, after all, the "frequency" is changing. If we "play" a waveform too slowly, in general a low-frequency portion (early) may thump while the high-frequency portion (late) will whistle, as the tone dies away. So the trick is to get the whole tone squeezed into a length such that the ear hears it as "one thing" (perhaps 30 to 300 ms). This is not a unique instance of having signal events need to be manipulated to achieve an overall aural impression. This is familiar to all persons who have endeavored to produced individual percussive sounds.

As of this writing, the results are hit-and-miss. But we said at the beginning we were mainly interested in having fun.

MATLAB PROGRAM USED HERE

```
% BounceBall.m

% TASK 1    Simulate (Fig. 1) and record Zeros
v=0;
g=32;
t=0;
s=1;
dt=0.0005;
N=9999;
r=0.8;
ss=[];
k=1;
for n=0:N
   v=v+g*dt;
   s=s-v*dt;
   ss(n+1)=s;
   if s<0;v=-v*sqrt(r);s=0;zt(k)=n*dt;k=k+1;end
end
zt1to20=zt(1:20)
figure(1)
plot([0:N]*dt,ss)
hold on
plot([-.5 5.5],[0 0],'k')
plot([0 0],[-.2 1.2],'k')
axis([-.2 5.2 -0.1 1.1])
hold off
figure(1)  % Fig. 1 of EN#212

% TASK 1a    Simulate (Fig. 2) and Record Zeros Reverse
v=0;
g=32;
t=0;
s=1;
dt=0.0005;
N=9999;
r=0.8;
ssf=[];
k=1;
d=1;
for n=0:N
   v=v+g*dt;
   s=s-v*dt;
   s=s;
   ssf(n+1)=s*d;
   if s<0;v=-v*sqrt(r);s=0;zt(k)=n*dt;k=k+1;d=-d;end
end
zt1to20=zt(1:20)
figure(2)
plot([0:N]*dt,ssf)
hold on
plot([-.5 5.5],[0 0],'k')
plot([0 0],[-1.2 1.2],'k')
axis([-.2 5.2 -0.85 1.1])
hold off
figure(2)  % Fig. 4 of EN#212
```

```
% TASK 1b  Compute Theoretical Time of Zeros
n=0:20;
t=2*sqrt(2*(0.8.^n)*1/32)
tt=[];
tt(1)=0.25;
for k=2:20
   tt(k)=tt(k-1)+t(k);
end
tt


% TASK 2  Comparison Sine - Fixed Freq (Fig. 3)

t=0:.001:10;
%xx=.8.^(2.4*t).*cos(2*pi*1.2*t);
%xx=.8.^(2.4*t).*cos(2*pi*0.6*t);
xx=.8944.^(2.4*t).*cos(2*pi*0.6*t);


absxx=abs(xx);
figure(3)
plot(t/2,absxx)
axis([0 10 -0.2 1.2])
hold on
plot([-.5 5.5],[0 0],'k')
plot([0 0],[-.2 1.2],'k')
axis([-.2 5.2 -0.1 1.1])
hold off
figure(3)    % Fig. 2 of EN#212

% TASK 3   Comparison Sine Variable Freq.   Fig. 4

r=0.8;
x=[];
for n=1:9999
   f=1/r^(n/250);
   xv(n)=(0.9995.^n).*sin(2*pi*f*n/10000);
   absxv=abs(xv);
end
figure(4)
plot(absxv)
figure(4)    % not displayed in EN#212

% TASK 4  Sine vs. Parabola Lobes Fig. 5

x=0:.01:2;
y=-x.^2 + 2*x;
s=sin(2*pi*x/4);
figure(5)
plot(x,y)
hold on
plot(x,s,'r')
plot([-1 3],[0 0],'k')
plot([0 0],[-.2 1.2],'k')
axis([-.2 2.2 -0.1 1.1])
hold off
figure(5)  % Fig. 3 of EN#212
```

```
% TASK 5: Plot FFTs


SS=abs(fft(ss));
SSF=abs(fft(ssf));
XX=abs(fft(xx));
ABSXX=abs(fft(absxx));

figure(6)
plot([0:9999]/2000,ss(1:10000),'r')
hold on
plot([0:9999]/2000,ssf(1:10000),':r')
plot([0:9999]/2000,xx(1:10000),':b')
plot([0:9999]/2000,absxx(1:10000),'b')
plot([-.5 5.5],[0 0],'k')
plot([0 0],[-1.2 1.2],'k')
axis([-.2 5.2 -1.1 1.1])
hold off
figure(6)  % Fig. 5 of EN#212



figure(7)
plot([0:99],SS(1:100),'r')
hold on
plot([0:99],SSF(1:100),':r')
plot([0:99],XX(1:100),':b')
plot([0:99],ABSXX(1:100),'b')
plot([-3 45],[0 0],'k')
plot([0 0],[-100 2000],'k')
axis([-3 45 -100 2000])
hold off
figure(7)   % Fig. 6 of EN#212
```

DACs and Psuedo-Random Noise Sequences

-by Bernie Hutchins

From time to time I look at the SYNTH-DIY website. My failed attempts to follow it every day are well-represented by an old Yahoo account (not berniehutchins@yahoo.com) that continues to receive all their postings, but which I seldom get a chance to even attempt to clean up. Instead I use their search for anything relating to Electronotes. There is usually nothing new. So I was surprised a couple of days ago to find a series of postings that were not only new, but brand new – same day. Any comments regarding Electronotes are of course of interest to me. Unfortunately I have never been able to post a reply on their site. I guess that's fine, as it is easier and much more productive to do it here in my own newsletter. And this was really interesting. Not the least is seeing what I did and what I wrote 35 years ago. This was with regard to the pseudo-random noise sources of EN#76 [1] (April 1977). Readers to the SYNTH-DIY website were asking about the DAC options there. Let's look at this in detail.

WHAT IS A DAC?

A DAC is a "Digital-to-Analog" converter (DAC or sometimes D/A). In the most general sense, we are converting something this is inherently numerical only, a number or series of numbers, to an electrical signal – a voltage usually. So in this sense, we might have a spider-net of digital logic doing "some math", and we simply connect a wire from somewhere (anywhere) inside, to provide a signal voltage to the external world. Except for the fact that we chose a particular point internal to the digital logic with some care (not wildly) this is in fact precisely what we did in the very early days of thinking about "digital synthesis".

In these early days, if we used op-amps and transistors, we were doing analog synthesis; and if we were using logic IC's, (originally RLT, then TTL, and eventually CMOS), we thought we were doing digital synthesis. A matter of definition perhaps, but we weren't doing digital synthesis in the sense we use the term today – not even close.

For example, we might start with a square wave and divide it down with flip-flops, producing a series of lower octaves. This was not digital. Even the use of those magnificent top-octave generators and flip-flop strings was not digital. In fact, some later analog synthesizers actually used a logic gate (preferably an Exclusive-OR) as a cheap make-do "ring modulator" (instead of a true analog multiplier – quite expensive,) but this was not digital.

But, if we do take the square waves in octaves AND if we consider the notion of adding them up for a staircase-wave (approximating a sawtooth), AND if we note the need to weight these with binary weighting, AND if we claim (correctly) that the flip-flops are counting (an upward staircase), THEN we are doing digital synthesis. Very limited however. But the op-amp summer (with different resistors), converting digital words as output – not just one bit, is properly considered to be a traditional DAC. Hal Chamberlin (EN#39 [2]) presented a device that counted, and even jumped through a sine-wave lookup table, and fed the result to a DAC. This was digital. True digital synthesizers (perhaps as live signal samplers rather than true synthesizers) followed soon enough.

DACs and Noise Generation

So what about the noise generator. Why a DAC anyway. Why not one bit. Indeed, the Pseudo-Random Binary Sequence (PRBS) generator (see Fig. 5 from EN#76 reproduced here) is inherently one bit (and perhaps millions of samples before repeating). If we wanted a noise generator to just produce raw sound for snare-drum synthesis for example, this two level sequence is enough. If on the other hand, we wanted to use the noise source to control pitches (driving a VCO with tones lasting perhaps a second) than two levels (two pitches only) aren't going to do it. And with multiple levels, we might have a preference for output levels that are evenly spaced, for use with exponential VCOs of course.

In the EN#76 article, I used two different DACs (Fig 8 of EN#76 reproduced here). One had equal weighting, and the other had a more conventional binary weighting. Both gave equally spaced levels, one had 256 levels and the other only 8 levels. Let's begin by considering, for one thing, that if we were only interested in getting multiple levels and/or spectral shaping, then analog filters would have been all we needed (as with the analog noise sources). Two of the things we get with the DACs are the hold, and equally spaced levels. A third thing we get is the correlation (or lack thereof) between successive output levels.

It is necessary here to recognize that the use of a conventional (or unconventional) DAC here to process and output the PRBS is not a conventional use of a DAC. A conventional use takes a full-size digital word (as many bits as the converter size), holds it (latch), and converts it to a single analog voltage. It is essentially parallel. On the other hand, the sequence being converted here is a time series. So some mindset change is useful at this point. We are just using existing devices. They do certain jobs for us regardless of their usual employment.

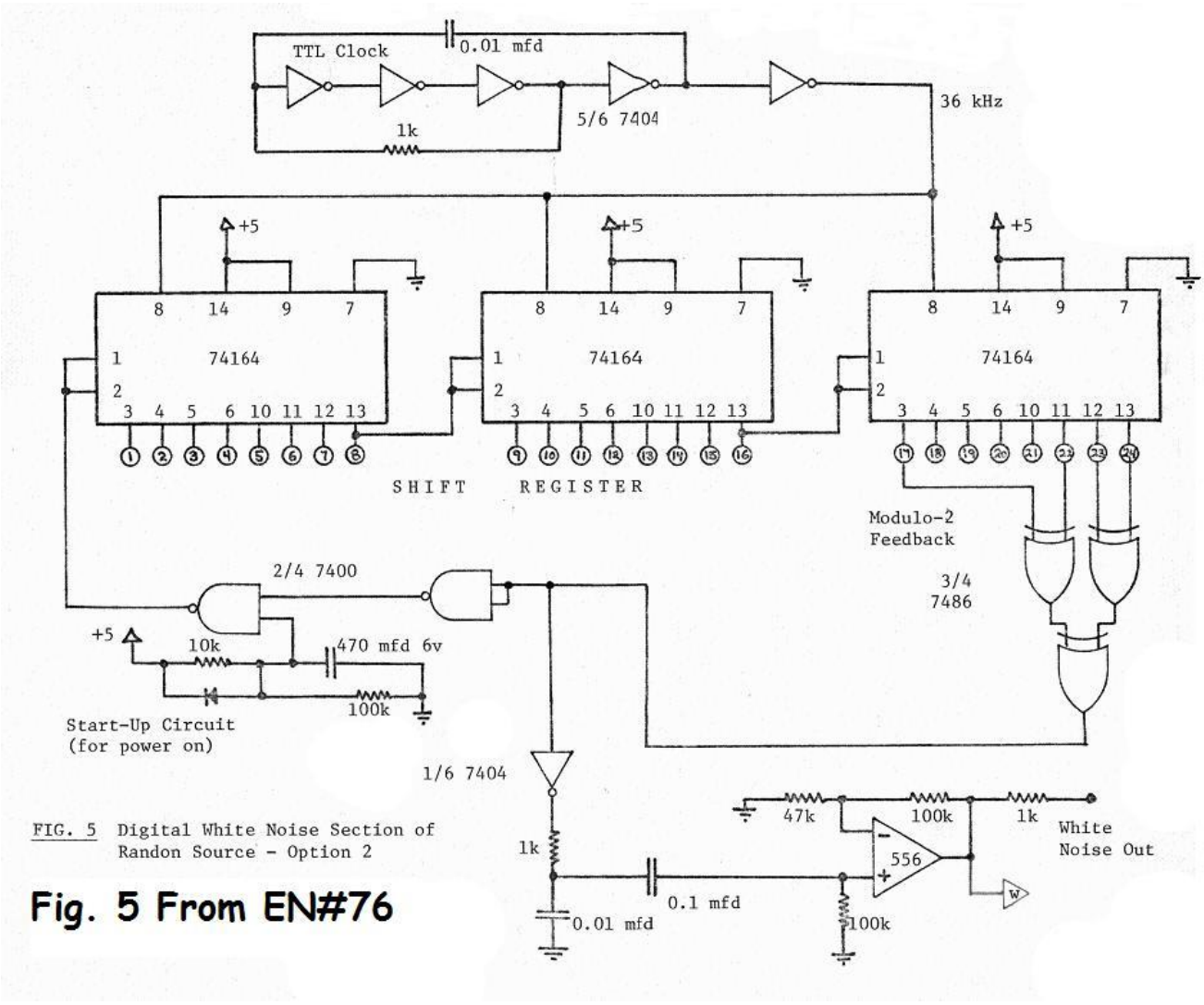


FIG. 5 Digital White Noise Section of Randon Source - Option 2

Fig. 5 From EN#76

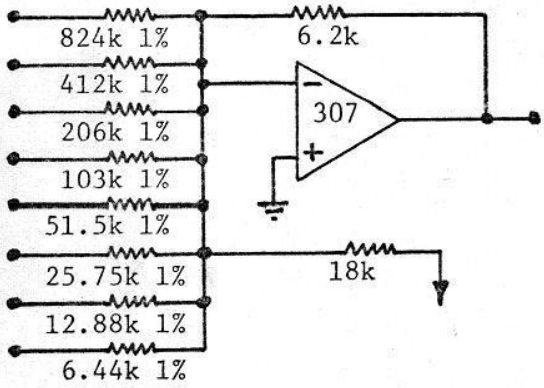


Fig. 8a Standard Binary Weighting
Uniform Distribution

Fig. 8 From EN#76

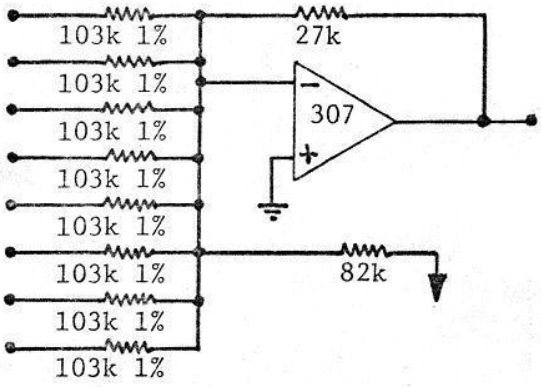


Fig. 8b Uniform Weighting
Gaussian Distribution

PRBS Generators

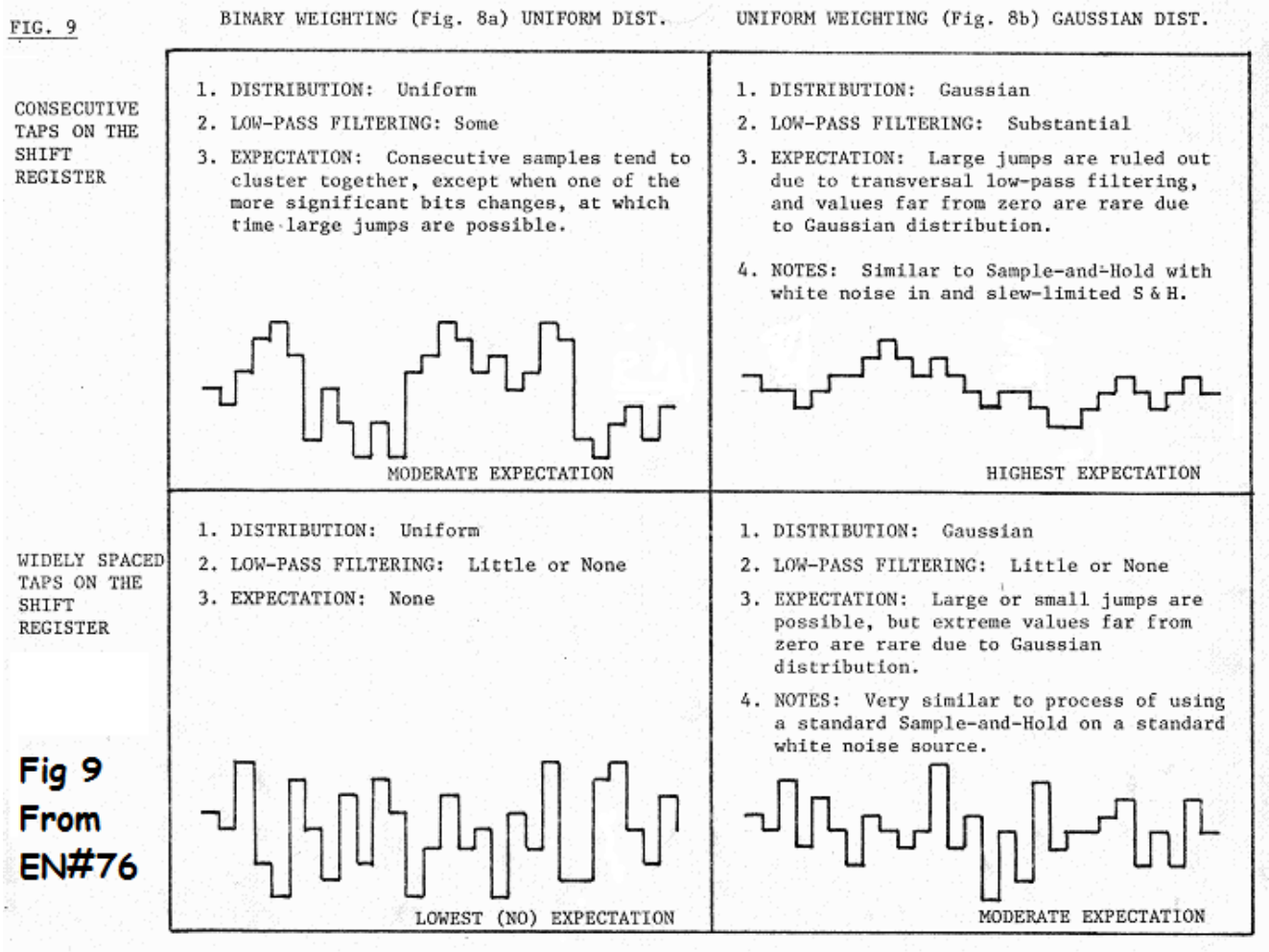
My experience with PRBS sequences and digital noise was actually the result of work I helped with (mostly Yuri Neuvo did it) in connection with a Cornell project [3]. With the PRBS approach (used principally so that we could generated bursts and repeat them exactly) we had TTL running essentially as fast as possible, with very long PRBS sequences. The sequence was fed into a counter, and 1024 consecutive one bit samples were added up. That is, the counter was zeroed, then incremented from zero, one step at a time, each time the sequence was a 1. This is exactly like counting heads in a long series of coin flips. Each noise output sample was the result of 1024 clockings! Thus a 10 MHz clocking on the PRBS was only about a 10 kHz output rate. On average, the count approached something like 512. The distribution of levels was of course "normal" or "Gaussian" (central limits theorem). The DAC in Fig. 8b likewise has a Gaussian distribution, although it is a running average. If we only took every 8^{th} output, we would have a kid-brother of the lab generator. [That "beast" generator could also produce a log-normal distribution, and could produce bursting (similar to the EN#76 probability switches)].

With regard to the Fig. 8b DAC, note that nothing prevents us from having more than 8 levels – we just need more resistors. If we use up all the "taps" on the PRBS generator, nothing prevents us from running the sequence well beyond the end using more shift registers after the feedback taps. But now we need to get to the issue of correlation.

Correlation Between Samples

Suppose we do use equal weighting, perhaps with the 8 resistors I used of perhaps with 256 levels with 248 more shift registers and 248 more resistors (I'm not suggesting this except in theory). If we use successive shift register taps (the first options in EN#76) then each successive output will differ from the previous output by at most one level (or no change at all). Only the new input that walks into the DAC and the one that walked out matter. These two are either the same, or they differ. The levels are thus very highly correlated. In fact, if I made the DAC infinitely long, we have our old friend the "red noise" or "random walk" [4]. [The statistics of this running average probably should be examined.] Musically, if we were using this as a pitch sequencer, it just plays up or down one note at a time. Not very interesting. (We could sample every 8^{th} output to get rid of the correlation, effectively as we had with 1024 sums of the lab generator.)

What happens with the binary weighting? We do get more levels of course. But jumps can be much larger, although there still is correlation. As is mentioned in EN#76, (for Fig. 8a) the distribution is not Gaussian, but uniform.



What distribution and what correlation do we want? Likely if we are talking about raw material random sound, it is the spectrum rather than the distribution and correlation (although related to spectrum) that matters. If we are talking about using a random sequence (held for the duration of individual notes), we are talking about musical melody – essentially.

First of all, for melody a Gaussian rather than a uniform distribution seems more familiar. Watch a piano player and see how much time he/she spends near the middle of the keyboard. Famously Voss and Clarke [5,6] found that with regard to correlation we wanted not white noise (too many large jumps) nor red noise (called also Brown noise) which was too correlated (just like playing scales), but something in between. Their suggestion was a pink noise, 3db roll-off, or 1/f noise with regard to power.

Spacing the Taps:

The PRBS generator In EN#76 (April of 1977) clearly predated Voss and Clarke (1978) by just a bit, so breaking up correlation (or even that it might be desirable to do so) was here approached not through filtering but through the spacing of taps. That is, instead of tapping consecutive stages of the PRBS-generating shift register, we put in some spaces (limited here to using 8 of the possible 24). This means that any one sample on the shift register can become "invisible" as it finds there is no output resistor at its current location. Thus it is conceivable that all or most all of taps could be experiencing 1's, and one clock later, these would step into non-tapped stages, and be replaced by 0's. This was why that option is suggested.

Related to this history and terminology of that era is the fact that I called the correlation "expectation" which is a different term if not a different notion. Also, the somewhat antiquated term "transversal filter" is used here (Finite Impulse Response, FIR, would be used today). The term transversal filter is exactly correct here however, and was common for sequential sums from delayed versions of the same signal.

Filtering

Here we have noted that the DAC was receiving serial rather than parallel input data and as such, something more like filtering was going on. Today this is easy to analyze and interesting to consider. Thus we can consider a FIR filter with tap weight the same as the DAC weights. The length 8 DAC with equal weighting thus has an impulse response which we will take to be $h_1(n)$ = 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8. This we know well as having a frequency response of a periodic sync. Fig. A shows this as the red curve. With the binary weighting we can take the weights as $h_2(n)$ = 128/256, 64/256, 32/256, 16/256, 8/256, 4/256, 2/256, 1/256. The corresponding frequency response is shown as the green curve in Fig. A. In fact, this looks like the impulse response of a IIR filter with a pole at z=1/2 (but truncated). Such an IIR filter would have exactly these eight values, and then continue forever. With the pole at 1/2, we would have, at z=1, a distance of 1/2 to the pole, and at z=-1 a distance of 3/2 to the pole, so the response would drop to 1/3 at z=-1 (half the sampling frequency), pretty much as we see. Note also that the response does not quite reach 1 at dc, due to the truncation of the impulse response. So much for consecutive samples.

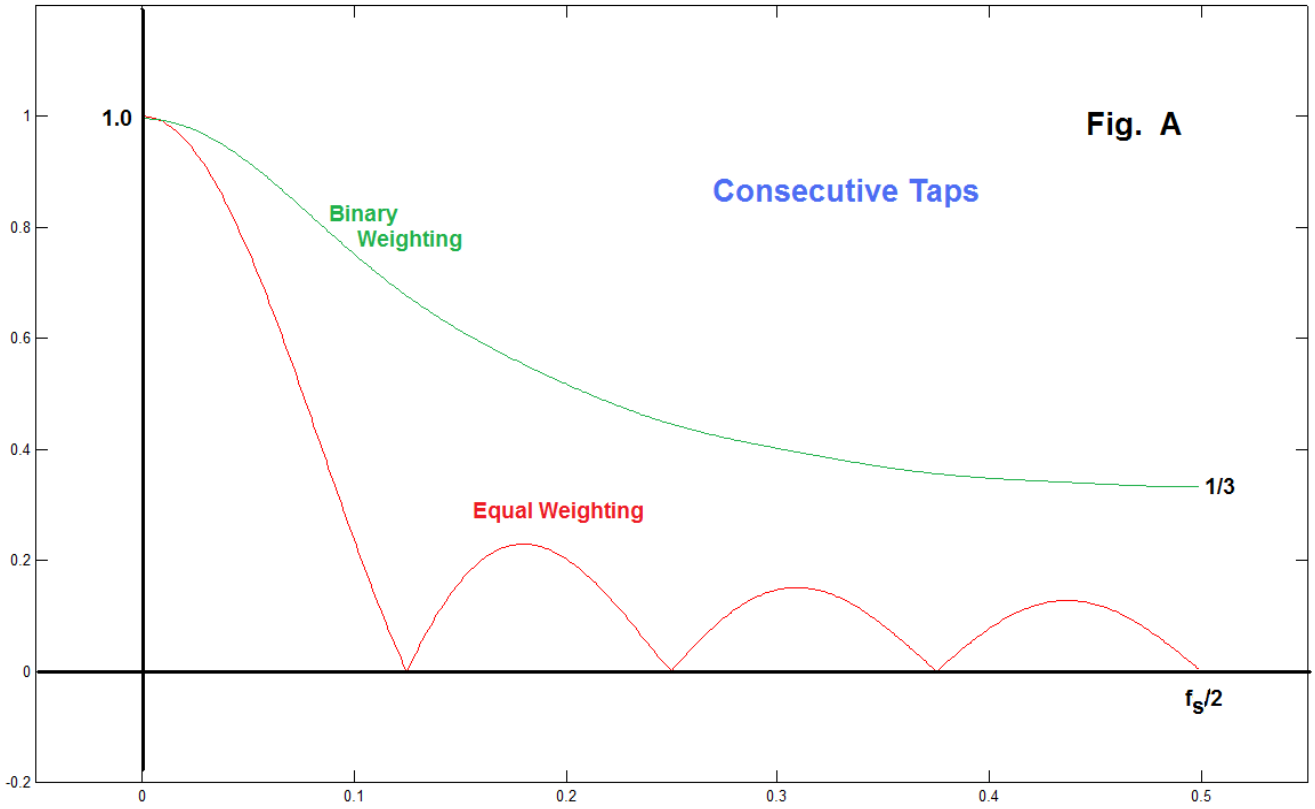


Fig. A
Consecutive Taps
1.0
Binary
Weighting
1/3
Equal Weighting
$f_S/2$
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.1
0.2
0.3
0.4
0.5

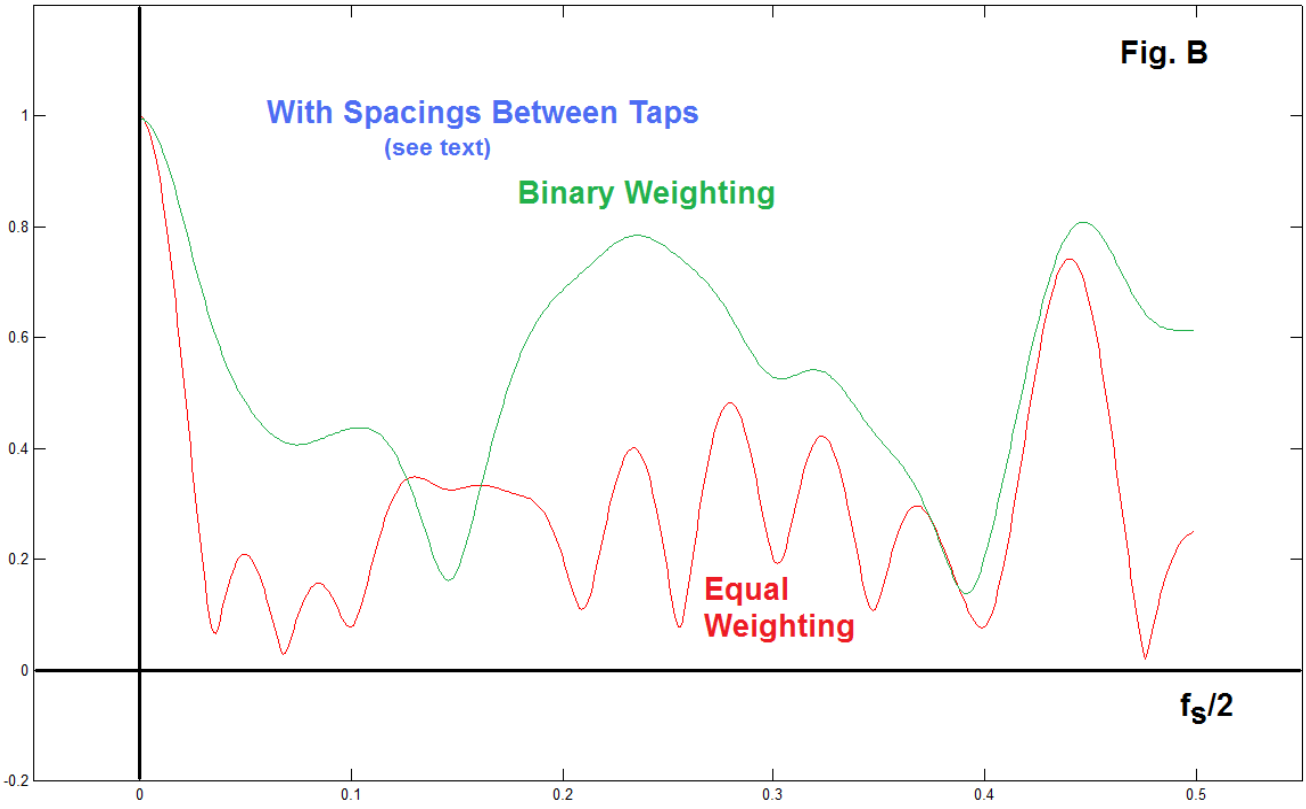


Fig. B
With Spacings Between Taps
(see text)
Binary Weighting
Equal
Weighting
$f_S/2$
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.1
0.2
0.3
0.4
0.5

In Fig. B, we have skipped some tap points. Nothing special about what we chose – just an example to show that things can be quite different. These are the same weights as Fig. A, except for inserted zeros. In Matlab notation:

```
h1=[1   0 0  1 0 0 0  1 0  1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1]/8;

h2=[128 0 0 64 0 0 0 32 0 16 0 0 0 0 8 0 4 0 0 0 0 2 0 0 0 1]/256;
```

The implications of Fig. B are that the spectral shaping would be quite different if we were listening to the noise directly, and the correlation properties would be less severe if we were using melody-producing sequences.

One additional point should perhaps be remade. Whether or not a distribution is uniform or normal (or whatever else) has NOTHING to do with whether or not the spectrum is "white". The spectrum depends on the time autocorrelation, not the amplitude distribution. Because here the outputs were (effectively) filtered, only the original PRBS was actually white. In reality, because all "digital" signals are ultimately held (for a proper output) there is always some roll-off (-4 db at half the sampling frequency).

REFERENCES

[1] Hutchins, B., "The ENS-76 Home-Built Synthesizer System – Part 8, Random Sources" ***Electronotes***, Vol. 9, No. 76, April 1977, pp 3-16

[2] Chamberlin, H., "Fourier Series Waveform Generator", ***Electronotes***, Vol. 5, No. 39, pp 2-5

[3] Neuvo, Y. and W. Ku, "Analysis and Digital Realization of a Pseudorandom Gaussian and Impulsive Noise Source", ***IEEE Trans. on Communications***, Vol. 23, No. 9, Sept 1975, pp 849-858

[4] Hutchins, B., "Fun with Red Noise" Electronotes Application Note No. 384, September 1, 2012

[5] Voss, Richard F. and John Clarke. "'1/f Noise in Music: Music From 1/f Noise." ***J. Acoust. Soc. Am.*** Vol. 63, No. 1 (1978) pp 258-261.

[6] Gardner, Martin. "Mathematical Games", ***Scientific American*** April 1978: 16-32. As he almost always did, Gardner had his own innovative way of doing something - generating sequences with 1/f properties in this case.