Effect Design*

Part 3 Oscillators: Sinusoidal and Pseudonoise

JON DATTORRO, *AES Member*

CCRMA, Stanford University, Stanford, CA, USA

In 1997, Jon Dattorro published articles entitled "Effect Design" that were to appear in the Journal in three parts. Parts 1 and 2 were published in the September and October issues. Part 3 is now being published, unedited. The paper is a tutorial intended to serve as a reference in the field of digital audio effects in the electronic music industry for those who are new to this specialization of digital signal processing. The effects presented are those that are demanded most often, hence they will serve as a good toolbox. The algorithms chosen are of such a fundamental nature that they will find application ubiquitously and often.

7 LOW-FREQUENCY SINUSOIDAL OSCILLATOR[107]

The low-frequency sinusoidal oscillator (LFO) is ubiquitous in effect design. What we seek is high computational efficiency and high signal purity in an algorithmic approach to real-time sinusoid generation. This section is presented more as a cookbook than the others because for the oscillator topologies that we analyze, each has distinct advantages; that is to say, each is useful. We offer the following implementation options:

1) Direct form
2) Coupled form
3) First modified coupled form
4) Second modified coupled form
5) Normalized waveguide.

7.1 Direct-Form Oscillator

The direct-form oscillator is the most efficient option requiring only one multiply, but it is noisy unless truncation error feedback is used [52], [12]. The error feedback can be implemented using only one or two adds, so it is attractive. (We do not consider error feedback here.) The single coefficient γ requires high resolution for very low frequencies of oscillation, however.

The direct-form oscillator was analyzed using state-variable theory,[108] the results of which are shown in Fig. 54. This type of analysis excels at finding the zero input response (ZIR) (see Section 7.6, Appendix 5). The selected output is expressed in terms of the initial conditions of all the memory elements, the state variables. This method of analysis differs considerably from the technique of solving the recursive difference equation for the selected output, in so far as the initial conditions required by the latter are of the output itself. In the case of the direct form, these two methods coincide.

Obviously, by choosing the initial states $y_1[0] = 0$ and $y_2[0] = -\sin\omega$, we get

$$y_1[n] = \sin n\omega\ .$$

There is no quadrature sinusoid at any node. But by choosing, for example, $y_1[0] = 1$ and $y_2[0] = \cos\omega$, we get

$$y_1[n] = \frac{1}{\sin\omega}\left[\sin(\omega + n\omega) - \cos\omega\sin n\omega\right] = \cos n\omega\ .$$

The direct-form oscillator is hyperstable[109] under coefficient quantization, as proven by the equation for the pole locations in the z plane in Fig. 54. Regardless of the quantization of γ, the pole radii are exactly 1 as determined from the pole magnitude. Any instability in the sinusoidal waveform can only be attributed to signal quantization effects, primarily in the form of truncation error in this recursive topology.

One must be cognizant of the relationship between oscillation frequency ω and amplitude when using the dif-

* Manuscript received 1996 March 14; revised 1996 September 14 and 1997 June 28.

[107] Special thanks to Julius Orion Smith for reviewing this section.

[108] Following the type of analysis presented in [53] for the coupled form and the second modified coupled form.

[109] We shall define *hyperstable* in this context to mean stability of oscillation in the face of coefficient quantization. The impact of signal quantization in digital circuits is not included in this definition.

ferent oscillators presented in this section. The equations of Fig. 54 predict that when the frequency is *changed* via γ after the oscillator has been running for some time, the amplitude will deviate, in general. One theoretically overcomes this problem by simultaneously updating the memory elements (the states), but this can be difficult in practice, depending on the particular oscillator topology.

The direct-form oscillator circuit can be derived from one trigonometric identity,[110]

$$\sin[(n+1)\omega] = \cos\omega \sin n\omega + \sin\omega \cos n\omega$$

$$\sin[(n-1)\omega] = \cos\omega \sin n\omega - \sin\omega \cos n\omega$$

where ω is the normalized radian frequency of oscillation ($2\pi fT$) controlled by γ, which should be in q22 format.[111] Summing the two equations yields

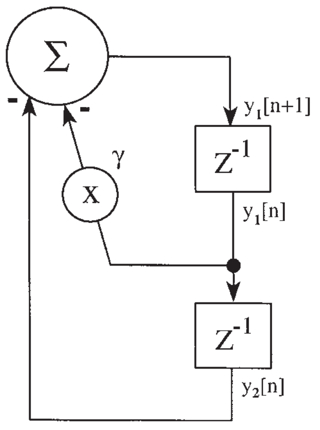
$$\sin[(n+1)\omega] = 2\cos\omega \sin n\omega - \sin[(n-1)\omega] \, .$$

Making the substitution $y_1[n] \leftarrow \sin n\omega$, we get

$$y_1[n+1] = 2\cos\omega \; y_1[n] - y_1[n-1]$$

which is the difference equation for the circuit.[112]

[110] This was pointed out to us by Michael A. Chen.
[111] See Section 9.1, Appendix 7.
[112] This also happens to be the generating recurrence relation of the Chebyshev polynomials, $T_{n+1}(x) - 2xT_n(x) + T_{n-1}(x) = 0$ [54, ch. 28.3, p. 473].



$$\begin{cases} y_1[n+1] = -\gamma\, y_1[n] - y_2[n] \\ y_2[n+1] = y_1[n] \end{cases}$$

$$y_1[n] = \frac{1}{\sin(\omega)} \left(y_1[0] \sin(\omega + n\omega) - y_2[0] \sin(n\omega) \right)$$

$$y_2[n] = \frac{1}{\sin(\omega)} \left(y_1[0] \sin(n\omega) + y_2[0] \sin(\omega - n\omega) \right)$$

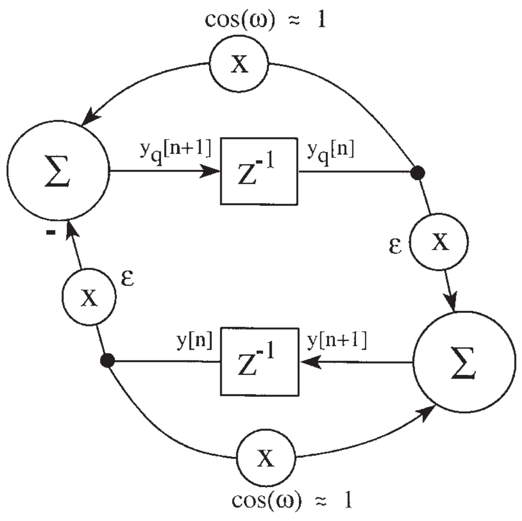
$$\boxed{\gamma = -2\cos(\omega) \quad ; 0 < \omega < \pi} \qquad \text{poles} = \frac{-\gamma}{2} +/- \; j\sqrt{1 - \frac{\gamma^2}{4}}$$

Fig. 54. Direct-form sinusoidal oscillator.

7.2 Coupled-Form Oscillator

The coupled form is an established topology known by several names, including Rader–Gold and normal form. The coupled form is a state-space digital filter structure and is one of the foremost contributions of the branch of linear systems theory known as state-variable analysis. The coupled form has many attributes, including low truncation noise and low coefficient sensitivity, respectively due to signal and coefficient quantization within a finite-precision machine [11, ch. 4.4]. The latter attribute makes tuning easier. Its primary detriment is that four multiplys are required in its unmodified form.

The unmodified coupled form in Fig. 55(a) shows the four multiplys required for oscillation, including two $\cos\omega$ coefficients. Using all four coefficients, the pole



$$\begin{cases} y_q[n+1] = \cos(\omega)\; y_q[n] - \varepsilon\, y[n] \\ y[n+1] = \varepsilon\; y_q[n] + \cos(\omega)\, y[n] \end{cases}$$

$$y_q[n] = y_q[0] \cos(n\omega) - y[0] \sin(n\omega)$$

$$y[n] = y_q[0] \sin(n\omega) + y[0] \cos(n\omega)$$

$$\boxed{\varepsilon = \sin(\omega) \quad ; \omega < \pi} \qquad \text{poles} = \cos(\omega) +/- \; j\,\varepsilon$$

(a)

$$\begin{cases} y_q[n+1] = y_q[n] - \varepsilon\, y[n] \\ y[n+1] = \varepsilon\, y_q[n] + y[n] \end{cases}$$

$$y_q[n] = \left(\frac{1}{\cos(\omega)}\right)^n \left(y_q[0] \cos(n\omega) - y[0] \sin(n\omega) \right)$$

$$y[n] = \left(\frac{1}{\cos(\omega)}\right)^n \left(y_q[0] \sin(n\omega) + y[0] \cos(n\omega) \right)$$

$$\boxed{\varepsilon = \frac{\sin(\omega)}{\cos(\omega)} \quad ; \omega < \pi/2} \qquad \text{poles} = 1 \; +/- \; j\,\varepsilon$$

(b)

Fig. 55. (a) Coupled-form (four-multiplier) sinusoidal oscillator. (b) First modified coupled-form (two-multiplier) low-frequency oscillator.

locations are ideally on the unit circle. It can be deduced from the equations given in Fig. 55(a) that for any initial states ($y[0]$, $y_q[0]$) the outputs at $y[n]$ and $y_q[n]$ yield *quadrature* sinusoids. But when the ideal filter coefficients become quantized to their representation in a finite-precision machine, the pole locations are perturbed in a direction dependent on the polarity of the coefficient quantization error. The impact of this is that the pole radii are no longer exactly 1, so the oscillator amplitude either decays to zero or clips (assuming automatic saturation arithmetic) after some time has passed. We do not relish the four multiply requirement and we recall that the direct form does not suffer from this particular problem. This is because the direct-form pole radii can be fixed by the second-order recursive coefficient [12, Eq. (30)], which is always set precisely to 1 for our present application.

The coupled form has a simpler trigonometric derivation, but uses two identities,

$$\cos[(n+1)\omega] = \cos\omega \cos n\omega - \sin\omega \sin n\omega$$

$$\sin[(n+1)\omega] = \sin\omega \cos n\omega + \cos\omega \sin n\omega \, .$$

Making the substitutions, $y[n] \leftarrow \sin n\omega$ and $y_q[n] \leftarrow \cos n\omega$, we get

$$y_q[n+1] = \cos\omega \;\; y_q[n] - \sin\omega \;\; y[n]$$

$$y[n+1] = \sin\omega \;\; y_q[n] + \cos\omega \;\; y[n] \, .$$

We will make a *modified* coupled-form digital filter behave as an oscillator by placing its poles either slightly beyond or precisely on the unit circle in the z plane, even in the presence of coefficient quantization error. We will see that both of these choices of pole locations can be achieved using only two multiplys, and both have useful purposes.

7.2.1 First Modified Coupled-Form Oscillator

If the coupled-form oscillator is used as an LFO, we make the observation that the two cos ω coefficients are close to 1. If we set them to 1 permanently, we eliminate two multiplys and arrive at the first modified coupled form in Fig. 55(b). Using the first modified coupled form, the quadrature sinusoids at $y[n]$ and $y_q[n]$ will always clip somewhat (assuming saturation arithmetic) because the pole radii are in excess of unity. When the oscillator frequency is very low, the clipping will be slight.

The advantage of this first modified coupled-form implementation is that its output amplitude is at least unity, even after oscillation frequency is abruptly changed. The oscillator never blows up, as the equations in Fig. 55(b) predict, if a saturation nonlinearity is built into the computation units. The detriment to the use of this oscillator is that the frequency of oscillation is practically restricted to a portion of the first quadrant in the z plane.[113] This modified coupled form of the oscillator would be chosen when the criteria for the selection of an LFO do not include sinusoid purity, but full-amplitude stable quadrature signals are a must.

7.2.2 Second Modified Coupled-Form Oscillator

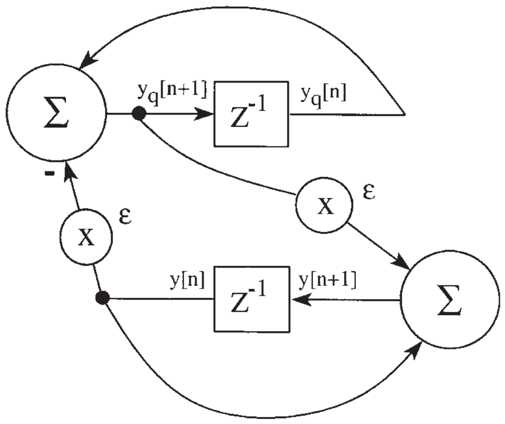
The second modified coupled form, first presented in [53], is shown in Fig. 56. This oscillator produces a high-purity sinusoid at the two outputs $y[n]$ and $y_q[n]$, whose noise floor is low enough to characterize digital-to-analog converters. The two outputs are no longer exactly in quadrature as before,[114] but the oscillator is hyperstable because the pole radii are exactly 1 (even when ε is quantized).[115] The tradeoff for this stability is the linking of amplitude to the frequency of oscillation, as seen in the equations in Fig. 56. By observation of the poles, the frequency of oscillation now spans dc to Nyquist as ε goes from 0 to 2, and so the range of oscillation is not restricted to low frequencies, as in the first modified coupled form.

We have found as predicted, using the second modified coupled form, that when ε (the frequency control) is changed abruptly, the oscillator amplitude will change itself to a new value. This is because we ignore the new initial states $y_q[0]$ and $y[0]$ at the time of the tuning

[113] This restriction is overcome in practice by running the oscillator twice as fast, which means twice the code.

[114] From the equations in Fig. 56 it can be shown trigonometrically for any initial states that the two sinusoids are in a near quadrature relationship, being off by exactly one-half sample at any frequency of oscillation. This was first pointed out to the author by Timothy S. Stilson.

[115] Like the direct form, any instability in the sinusoidal waveform can only be attributed to signal quantization effects, primarily in the form of truncation error in this recursive topology.



$$\begin{cases} y_q[n+1] = y_q[n] - \varepsilon\; y[n] \\ y[n+1] = \varepsilon\; y_q[n+1] + y[n] \end{cases}$$

$$y_q[n] = \frac{1}{\cos(\frac{\omega}{2})} \left(y_q[0] \cos(n\omega - \frac{\omega}{2}) - y[0] \sin(n\omega) \right)$$

$$y[n] = \frac{1}{\cos(\frac{\omega}{2})} \left(y_q[0] \sin(n\omega) + y[0] \cos(n\omega + \frac{\omega}{2}) \right)$$

$$\boxed{\varepsilon = 2\sin(\frac{\omega}{2}) \;\; ; \omega < \pi} \qquad \text{poles} = 1 - \frac{\varepsilon^2}{2} +/- \; j\,\varepsilon\sqrt{1 - \frac{\varepsilon^2}{4}}$$

Fig. 56. Gordon – Smith sinusoidal oscillator, also called second modified coupled-form (two-multiplier) oscillator.

change; that is, no attempt is made to adjust them. We observed these consequent deviations in amplitude and empirically found them to be less than a decibel over a very useful range of oscillation frequency for an LFO. It was never found necessary, in any of our applications, to compensate for these small changes in amplitude. Built-in saturation arithmetic within the computation units will eliminate any possibility of long-term clipping completely.

The ideal digital integrators $1/(1 - z^{-1})$ have never presented a problem in practice because there is zero transmission at $z = 1$ (dc) across each embedded integrator from its input to its output. This comment applies to both modified coupled forms.

7.3 Real-Time Measure of Sinusoid Purity

We made measurements of $\boldsymbol{THD + N}$ for the direct-form and the second modified coupled-form oscillators, each running at a sample rate of 69 818.181 Hz in a real-time hardware development system. For each oscillator, the two parameters are frequency of oscillation and the number of bits in the two's complement signal path. Binary truncation post-accumulation is used to limit the path width. The results are tabulated in Table 8.

The hardware system average noise is roughly −86 dB relative to full scale, which prevented measurements significantly below that level. The readings at −88 dB are probably due to some system signal-transfer anomaly. The term *Flatline* in Table 8 refers to a complete lack of any form of oscillation as viewed on an oscilloscope.

7.3.1 Purity of Direct versus Coupled Form

We now postulate the reason pertaining to the foregone conclusion from the empirical data that the direct-form oscillator is inferior to the second modified coupled form. It is that the latter has truncation error feedback built into its topology. We state this in an equivalent way: each noise transfer function of the second modified coupled form has zeros of transfer, whereas that of the direct form has not. Fig. 57 shows how high-rate truncation noise sources $e[n]$ and $e_q[n]$ are conceptually inserted into a circuit in a linear fashion [12, Eq. (6)]. We have drawn each deterministic noise source into the circuit so that it resides in front of a multiplier, because that is the only location where truncation is demanded by the architecture in contemporary DSP chips.[116]

Each noise source (acting as input) can make its way to either of two outputs for this coupled topology in Fig. 57: $y_q[n]$ or $y[n]$. In each case the noise transfer either picks up a zero at dc, or is multiplied by ε^2, which yields the same outcome at low frequencies ($\omega < \pi/3$). In order for the direct-form oscillator to perform as well, truncation error feedback [12] must be used to introduce a zero into its deterministic noise transfer function, as stated at the outset.

7.3.2 Purity versus Frequency

The data in Table 8 indicate that signal purity is a function of oscillator frequency for both topologies. Binary truncation noise can be modeled like quantization noise [14, ch. 6.9.1, p. 353]. Gray demonstrates [55, ch. 6.3] how the quantization noise of a sampled pure sinusoid is never characteristically white, regardless of its amplitude

[116] Were we instead to place the noise sources following the accumulators, for truncation post-accumulation, we would reach similar conclusions in the analysis of this topology. The ideal digital integrators would not pose a practical problem in this case either, as demonstrated by Table 8.

Table 8. Measurement of oscillator $\boldsymbol{THD + N}$ in decibels.

Number	Direct Form			Second Modified Coupled Form		
of Bits	20 Hz	100 Hz	1000 Hz	20 Hz	100 Hz	1000 Hz
24	−60	−75	−86	−88	−86	−86
23	−58	−75	−86	−88	−86	−86
22	−53	−73	−85	−88	−86	−86
21	−45	−70	−84	−87	−86	−86
20	−37	−64	−84	−87	−86	−86
19	Flatline	−60	−79	−86	−85	−85
18		−58	−74	−85	−84	−85
17		−50	−73	−79	−80	−84
16		−40	−67	−70	−73	−82
15		−25	−61	−67	−68	−78
14		Flatline	−56	−59	−65	−73
13			−54	−47	−61	−68
12			−45	−40	−60	−62
11			−39	−25	−50	−56
10			−37	Flatline	−45	−48
9			−23		−30	−43
8			−19		−21	−42
7			Flatline		Flatline	−35
6						−34
5						−28
4						−24
3						Flatline
2						
1						

or frequency. He explains that the deterministic noise spectrum is discrete, because the signal is sinusoidal,[117] and that the noise spectral components exist at odd harmonics of the sinusoid frequency, including the fundamental. These findings refine the traditional [14] high-rate analysis. Gray's references indicate that these results have been known for decades. The conclusions drawn can be generalized to the present truncation noise situation depicted in Fig. 57:

$$e[n] = \sum_{\substack{m=-\infty \\ m \text{ odd}}}^{\infty} e^{jm\omega n} b_m \tag{55}$$

where ω is the normalized (fundamental) radian frequency of oscillation. Eq. (55) has the form of a continuous-time complex Fourier series [56], then sampled in time, *nT*. The Fourier series coefficients b_m form a conjugate-symmetric set, thus e[*n*] is real valued. The expression for $e_q[n]$ is similar. Each nonuniform b_m will be some function of weighted ordinary Bessel functions of integer order *m* and real argument.

Thus far we have presented all the equations for oscillation in terms of the ZIR. But the noise model in Fig. 57 suggests that the noise sources e[*n*] and $e_q[n]$ are subject to the zero-state response (ZSR) of the circuit. The ZSR of the oscillator is precisely that of an integrator *centered* (in the frequency domain) at the frequency of oscillation.[118] The fundamental frequency of the noise spectrum is the same as the center frequency of the integrator, which is the same as the frequency of oscillation. Any noise energy in the vicinity of the oscillation frequency should cause the oscillator to blow up. This does not happen in practice because of the saturation nonlinearity built into most contemporary DSP chips. The injected noise just causes phase jitter and amplitude perturbations, which decrease signal purity.[119] Signal quantization in a recursive circuit, then, is a secondary form of instability. The primary determinant of stability is pole location, which is why we were concerned with the quantization of filter coefficients.

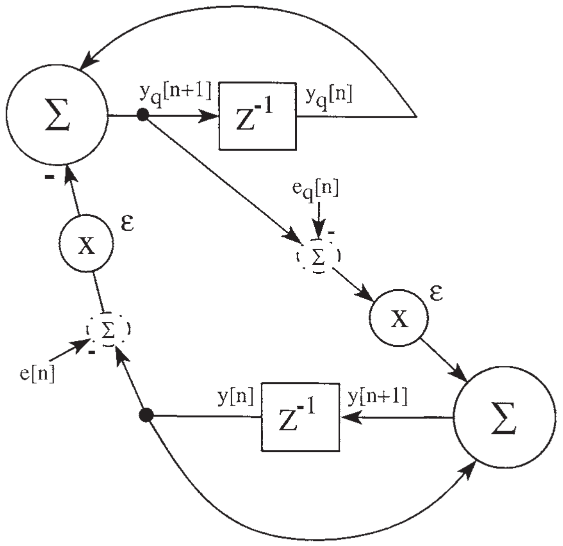


Fig. 57. Second modified coupled-form sinusoidal oscillator showing noise sources.

It appears from Table 8 that the quantity of noise goes up as the center frequency moves down. It is plain that the frequency-shifted integrator becomes more disturbed at low frequencies of oscillation, thus the worsening ***THD + N*** for a given number of bits. We speculate that an explanation of this phenomenon might simply be excessive frequency-domain foldover of the shifted integrator frequency response. Perhaps when the oscillation frequency is low, more harmonics from the left-side noise spectrum (negative frequencies) fall under the shifted integrator that is on the right-side, and vice versa.

7.3.3 Chaotic Behavior

Nonetheless, the direct-form oscillator waveform at 20 Hz is fascinating to view when the signal path bit width is 20 bit (just before Flatline). The oscillation metamorphoses from the sinusoidal to a nearly triangular waveform and back. This chaotic but stable process occurs over periods of real time on the order of minutes. During each epoch, the oscillation frequency can be observed to change by as much as one-half the desired frequency.

It should be pointed out that although we show no data for very low frequencies in Table 8, the first and second modified coupled-form sinusoidal oscillators are routinely called upon to produce frequencies less than 1 Hz (0.1 Hz typical).

7.4 More Recent Developments

Smith and Cook [57] subsequently disclosed a lattice topology for sustained oscillation which is claimed more suitable for VLSI implementation. The oscillator shown in Fig. 58 is called the normalized waveguide oscillator because it was derived as a spinoff from Smith's results in the theory and implementation of digital waveguides.

Although this design recaptures the quadrature relationship of the sinusoids appearing at the circuit outputs, the primary contribution of this topology to the field of oscillator design is that it solves the amplitude deviation problem.[120] It is a remarkable distinction that this sinusoidal oscillator possesses. This normalized waveguide oscillator is designed for instantaneous change in oscillation frequency *without* concomitant change in output amplitude,[121] hence the new notation ω_n showing frequency as a function of the time index. Change in the frequency of oscillation must be performed properly, however. Toward this end, the amplitude coefficient G_n is introduced and

[117] To be more precise, because the analog signal, from which the sampled signal is derived, has a discrete spectrum.
[118] In the interest of avoiding confusion, we stress that we are *not* talking here about the ideal digital integrators embedded in the circuits of Fig. 55, Fig. 56, nor Fig. 57.
[119] For this reason it is recommended to run all the oscillators at full-scale amplitude and to place a volume knob at the output.
[120] It is evident from the equations in Fig. 58 that sinusoid amplitude is linked to the frequency of oscillation. This is similar to the situation for all the oscillators presented, except for the Rader–Gold coupled form.
[121] Because we are dealing with the ZIR, change in frequency of oscillation can be instantaneous in all the oscillators presented. But the Smith–Cook oscillator is the first one to deal effectively with amplitude correction at the instant of the change.

specified to deviate from 1 only at the time of the occurrence of the frequency change,[122] that is, an amplitude compensation factor which is engaged for one sample period. The elegance of the Smith–Cook solution rests in the fact that knowledge of the state time (the precise value of n) at the occurrence is not required, neither is knowledge of the state values. All that is required is knowledge of the previous frequency.

Whenever the frequency of oscillation is changed to a new value, the control G_n is all that is required to maintain constant amplitude at the oscillator circuit output y_1, that is, the memory elements need never be adjusted in the implementation. From Fig. 58,

$$\begin{aligned} y_1[n] &= y_1[0] \cos n\omega_n - y_{2G}[0] \cot\left(\frac{\omega_n}{2}\right) \sin n\omega_n \\ y_2[n] &= y_1[0] \tan\left(\frac{\omega_n}{2}\right) \sin n\omega_n + y_{2G}[0] \cos n\omega_n \, . \end{aligned} \tag{56}$$

For the *analytical* Eq. (56) to remain valid, however, $y_{2G}[0]$ and $y_1[0]$ must be re-evaluated whenever G_n deviates,

$$\begin{aligned} y_1[0] &\leftarrow y_1[n_0] \\ y_{2G}[0] &\leftarrow y_2[n_0]G_{n_0} = y_{2G}[n_0] \end{aligned} \tag{57}$$

where n_0 is the time index at the change; $n_0 \neq 0$. It is important that we interpret Eq. (56) properly so that we may show the technique on paper. We will now demonstrate the validity of these assertions by example.

[122] G_n is not at all involved with frequency tuning. The special case $G_0 = 1$. But G_n can be greater than 1, albeit momentarily, which will necessitate at most q22 arithmetic.

7.4.1 Example

Suppose that at absolute time $n = 0$, we are given $\omega_0 = \omega_{-1}$, $y_1[0] = 1$, and $y_{2G}[0] = 0$. Then while G_n remains static, we expect for $n = 0 \rightarrow n_0$

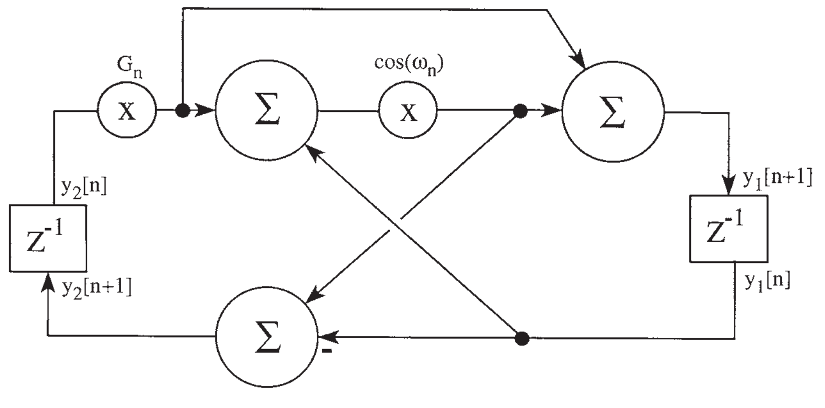
$$\begin{aligned} y_1[n] &= \cos n\omega_0 \\ y_2[n] &= \tan\left(\frac{\omega_0}{2}\right) \sin n\omega_0 \, . \end{aligned} \tag{58}$$

Suppose we suddenly freeze time at $n = n_{0^+}$. At this time we desire a change in the frequency of oscillation, which is effectively to take place at $n = n_0$. We do this by changing the one tuning coefficient from $\cos \omega_0$ to $\cos \omega_{n_0}$ at time n_0, and by letting G_n deviate from 1, but only for one sample period at n_0,

$$G_{n_0} = \frac{\tan(\omega_{n_0}/2)}{\tan(\omega_{n_0-1}/2)} \, . \tag{59}$$

We then have the new initial states from Eqs. (58) and (57),

$$\begin{aligned} \omega_{n_0-1} &= \omega_0 \\ y_1[n_0] &= \cos(n_0\omega_{n_0-1}) \\ y_{2G}[n_0] &= \tan\left(\frac{\omega_{n_0}}{2}\right) \sin(n_0\omega_{n_0-1}) \, . \end{aligned} \tag{60}$$



$$\begin{cases} y_1[n+1] = \cos(\omega_n) \; y_1[n] + (\cos(\omega_n)+1) \, G_n \; y_2[n] \\ y_2[n+1] = (\cos(\omega_n)-1) \; y_1[n] + \cos(\omega_n) \; G_n \; y_2[n] \end{cases}$$

$$\begin{aligned} y_1[n] &= y_1[0] \cos(n\,\omega_n) - y_{2G}[0] \cot(\omega_n/2) \sin(n\,\omega_n) \\ y_2[n] &= y_1[0] \tan(\omega_n/2) \sin(n\,\omega_n) + y_{2G}[0] \cos(n\,\omega_n) \end{aligned}$$

$$G_n = \frac{\tan(\omega_n/2)}{\tan(\omega_{n-1}/2)} \quad ; 0 < \omega_n < \pi/2 \qquad \text{poles} = \cos(\omega_n) \; +/- \; j\sqrt{1 - \cos^2(\omega_n)}$$

Fig. 58. Smith–Cook normalized-waveguide sinusoidal oscillator.

Using the new initial states [Eq. (60)], we find from Eq. (56) that

$$\begin{aligned} y_1[n] &= \cos(n_0\omega_{n_0-1})\cos[(n-n_0)\omega_{n_0}] - \sin(n_0\omega_{n_0-1})\sin[(n-n_0)\omega_{n_0}] \\ y_2[n] &= \cos(n_0\omega_{n_0-1})\tan\left(\frac{\omega_{n_0}}{2}\right)\sin[(n-n_0)\omega_{n_0}] + \tan\left(\frac{\omega_{n_0}}{2}\right)\sin(n_0\omega_{n_0-1})\cos[(n-n_0)\omega_{n_0}] \end{aligned} \qquad (61)$$

for $n = n_0 \to \infty$.

Eq. (61) simplifies by trigonometric identity to

$$\begin{aligned} y_1[n] &= \cos[n_0\omega_{n_0-1} + (n-n_0)\omega_{n_0}] \\ y_2[n] &= \tan\left(\frac{\omega_{n_0}}{2}\right)\sin[n_0\omega_{n_0-1} + (n-n_0)\omega_{n_0}] \end{aligned} \qquad (62)$$

for $n = n_0 \to \infty$.

In both outputs the phase is correct in light of the new starting time. Comparing Eqs. (62) and (58), we see that the amplitude of $y_1[n]$ remains as it was, which is the desired result. To change the frequency successfully again and again, the same procedure can be repeated to get similar results. But the amplitude of $y_2[n]$ changes from its original value of $\tan(\omega_0/2)$. So we have not maintained the amplitude of $y_2[n]$, although we have managed to keep the amplitude of $y_1[n]$ constant throughout the process of changing the frequency of oscillation.

It is interesting to note that Smith and Cook originally derived the same result using principles of energy conservation across transformer-coupled waveguides.

7.4.2 Stability

The Smith–Cook oscillator is hyperstable because there is only one tuning coefficient, $\cos \omega_n$. The equation for the poles in Fig. 58 shows that even under coefficient quantization, the pole magnitude is always unity. Like the direct form, however, the single tuning coefficient requires high resolution at very low frequencies of oscillation. Inaccuracy in the representation of G_n when it deviates cannot affect the frequency of oscillation.

7.4.3 Truncation Noise

We will only permit speculation regarding the noise performance of the Smith–Cook oscillator topology as compared to the (Gordon–Smith) second modified coupled form, as we have made no actual measurements of ***THD* + *N*** to bolster any analytical findings.

Recalling our discussion of purity of direct versus coupled form (Section 7.3.1) there the importance of zeros in the deterministic noise transfer function was revealed. We have not been able to devise an implementation that would place a zero into the time-invariant truncation noise transfer (amplitude coefficient G_n set to 1) while maintaining freedom from amplitude deviation across a change in oscillation frequency.[123] Hence we speculate that the noise performance of the Smith–Cook sinusoidal oscillator is not as good as that of the Gordon–Smith.

[123] This topic is worthy of further research, however. One complicating circumstance is the potentially large disparity in amplitude between the two outputs, as shown by Eq. (56).

7.5 Miscellany

Some other papers relevant to the field of oscillator design are [58], [59].

7.6 Appendix 5: Derivation of Oscillator Equation

We demonstrate the derivation of the oscillator equations for one case only—the first modified coupled-form oscillator. This analysis is adapted from [53], where the derivation of the coupled form and the second modified coupled form is shown.

$$\begin{cases} y_q[n+1] = y_q[n] - \varepsilon\, y[n] \\ y[n+1] = \varepsilon\, y_q[n] + y[n]\,. \end{cases}$$

These state equations describe the ZIR of the circuit in Fig. 55(b), which is duplicated here in Fig. 59. From the state equations we read the matrix

$$\boldsymbol{A} = \begin{bmatrix} 1 & -\varepsilon \\ \varepsilon & 1 \end{bmatrix}.$$

If we define the vector

$$\boldsymbol{y}_n = \begin{bmatrix} y_q[n] \\ y[n] \end{bmatrix},$$

then we may write $\boldsymbol{y}_{n+1} = \boldsymbol{A}\boldsymbol{y}_n$. This state-variable description [11] has the solution

$$\boldsymbol{y}_n = \boldsymbol{A}^n \boldsymbol{y}_0\,, \qquad \text{for all time } n \ge 0$$

where $\boldsymbol{A}^n$ is the state transition matrix and $\boldsymbol{y}_0$ is the vector of initial states, that is, $\boldsymbol{y}_n$ at $n = 0$.

If we can find the eigenvectors and eigenvalues of $\boldsymbol{A}$,

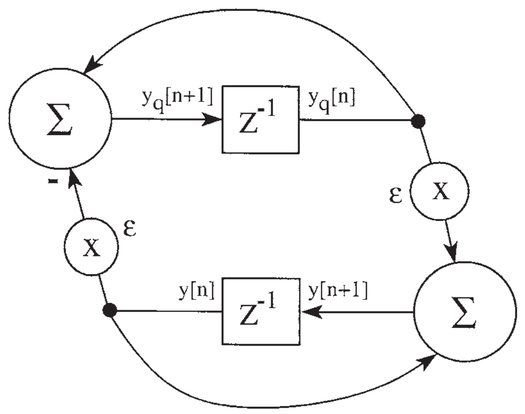


Fig. 59. First modified coupled-form sinusoidal oscillator.

then we may write equivalently

$$\boldsymbol{y}_n = \boldsymbol{T}\,\Lambda^n \boldsymbol{T}^{-1}\boldsymbol{y}_0$$

where $\boldsymbol{T}$ holds the conjugate eigenvectors (given by *Mathematica*) in its columns, and Λ holds the corresponding eigenvalues along its diagonal. Specifically,

$$\Lambda = \begin{bmatrix} 1+\mathrm{j}\varepsilon & 0 \\ 0 & 1-\mathrm{j}\varepsilon \end{bmatrix} \triangleq \begin{bmatrix} \lambda & 0 \\ 0 & \lambda^* \end{bmatrix}$$

$$\boldsymbol{T} = \begin{bmatrix} \mathrm{j} & -\mathrm{j} \\ 1 & 1 \end{bmatrix}$$

$$\boldsymbol{T}^{-1} = \frac{1}{2}\begin{bmatrix} -\mathrm{j} & 1 \\ \mathrm{j} & 1 \end{bmatrix}.$$

By constructing the diagonal matrix Λ, the exponentiation no longer requires matrix multiplication. Hence it is easier to acquire a solution analytically for large n. The eigenvalues are the poles of the system under study. λ^* denotes the conjugate eigenvalue. If we define each eigenvalue in polar form,

$$\lambda \triangleq |\lambda| e^{\mathrm{j}\omega}$$

then we can make the identifications from Λ

$$\Rightarrow \begin{cases} 1 = |\lambda| \cos\omega \\ \varepsilon = |\lambda| \sin\omega . \end{cases}$$

From these we conclude

$$\varepsilon = \frac{\sin\omega}{\cos\omega}$$

$$|\lambda| = \frac{1}{\cos\omega}, \qquad \omega < \frac{\pi}{2}$$

where the actual oscillator coefficient ε is expressed in terms of the desired normalized radian frequency of oscillation ω. It follows that

$$\boldsymbol{T}\Lambda^n\boldsymbol{T}^{-1} = \frac{1}{\cos^n\omega}\begin{bmatrix} \cos n\omega & -\sin n\omega \\ \sin n\omega & \cos n\omega \end{bmatrix}.$$

8 SONICALLY PLEASANT NOISE GENERATION

8.1 Sonic Musing[124]

The author presently owns about five gadgets for sonic noise generation. These devices are analog and emit electronically amplified sound through a loudspeaker, except for one, which consists of an encased fan driven by an induction motor. The fan-type unit was first purchased from Marpac Corporation in 1977. Their analog noise generators, which amplify and filter transistor noise, were found to be quite suitable substitutes for the earlier electromechanical version. The noise generators are used when sleeping, concentrating, or to mask unwanted ambient disturbances.

Several years ago the same company announced a new "improved" digital sonic noise generator. Not only is the noise generation digital and algorithmic, but the front panel has been computerized. After living with the unit for a while, the author concluded that the noise was not pleasant, so back it went to the factory for repair. Discussion with a Marpac engineer revealed that the device was operating within normal parameters and was not in need of repair. He confided that he too liked the analog units better. The author no longer uses that particular digital device because, in the author's opinion, it sounds bad. Notwithstanding, the Marpac company claims growing sales of the digital unit and has since released yet newer models.

When evaluating various algorithms for noise generation, we use our ears. Our purpose for transcribing these events is to convey the knowledge that digital methods for noise generation are *not* inherently bad, but some algorithms sound better than others. We believe that the fundamental algorithm to be presented herein sounds as good as any analog method of noise generation; that is our ultimate criterion for choosing it.

The technique that we prefer to create pleasant audio noise is called the maximal-length pseudorandom noise (PN) sequence [60], [61].[125] Since the PN sequence is generated by a digital circuit called the PN generator, it is a completely deterministic binary sequence {0, 1}. The PN generator circuit comprises a solitary binary register (see Fig. 60) as would be found in almost every microprocessor. Because the recursive circuit has no perturbing input (only initial states) and is marginally stable, the binary PN sequence is periodic. The PN sequence is not truly noise because it is not the *realization* of any random process; hence the prefix "pseudo." Still, we prefer to characterize the binary output by adapting concepts from the field of statistical signal processing [62]. Thus our characterization of the PN sequence will mainly consist of calculations of sample average (or sample mean), sample variance, circular (or cyclic) sample autocovariance,[126] and power spectrum, all over one period. As much as possible, we apply the language of DSP to this characterization. Hence we adopt a simple definition of power spectrum, which is the magnitude squared of the discrete Fourier transform (DFT) of the PN sequence over one period, normalized by the number of samples per period M. This definition of power spectrum is consistent with the engineer's notion of the average power of a signal [31], [14, app. A].

Even though the PN sequence generated by the digital circuit is completely deterministic, we still refer to the emanating pseudorandom binary sequence as "noise" because it is incoherent (but perhaps soothing) to our ears. We analyze two distinct ways of observing (rather, listen-

[124] Thanks go to Robert M. Gray for a careful review.

[125] Often referred to as a maximal-length PN sequence in the literature.

[126] Autocovariance is like autocorrelation, but having the mean removed prior to its calculation. The prefix "sample" indicates that the calculation is performed upon one realization of a noise process, rather than an expectation over a noise probability distribution. The realization comprises the PN sequence in our case. Circular autocovariance is the autocovariance of a periodic waveform.

ing to) that binary sequence. The first method, called the *single-bit* method, is what we also call the traditional method, that is, the observation of a single bit from the PN generator register. Analysis and implementation of the traditional single-bit output of the PN generator are well described in the literature but not always in the language of DSP [60], [63], [64]. The second method of observing the binary sequence, called the *multibit* method, is the simultaneous observation of multiple bits from the generator register. Analysis of the multibit (word) output from the PN generator is touched upon in [65], where it is noted that both the single- and multibit sequences have the same period M. This multibit method is critically dependent on the assignment of weights to (the format of) the individual bits constituting the word output. We discuss a few distinct formats that are readily available within most microprocessors, including two's complement and unsigned format, and we analyze the impact of each. Regardless of the particular format chosen, the multibit output is perfectly uniform in its distribution of amplitude over one period when the number of bits in the word output is the same as the generator *word length*.

Because the source of our noise is known and hence deterministic, the pseudonoise power spectrum is defined as the discrete Fourier transform of the sample autocorrelation of the periodic PN sequence [62], [66, ch. 7.6], [67]. The pseudonoise created using the traditional single-bit method is spectrally white because the (circular) sample autocovariance of a maximal-length single-bit PN sequence is a Kronecker delta, periodic in M [60]; that is, the single-bit PN process is uncorrelated. We find that this property of the (sample) autocovariance remains approximately intact for an unsigned multibit output, and that the multibit sequence remains maximal in length. So the pseudonoise power spectrum created using a multibit output is nearly white (see Fig. 66) because the autocovariance of the resulting maximal-length PN sequence is spiky, as we shall see (see Fig. 67).

White noise is easily filtered to create noise for audio having spectral color. Filtering white noise by means of a low-pass filter having a cutoff frequency of approximately 400 Hz, for example, yields a very soothing rumble. Such narrow-band noise might then be transposed (shifted) in the frequency domain; achieved through multiplication (ring modulation) in the time domain by a sinusoid of the desired transposition frequency. This simple and pleasant effect is reminiscent of gurgling brooks.[127]

It is certainly reasonable to digitally filter the spectrally white single-bit PN sequence to make multibit audio noise having a desired color. Filtering the single-bit sequence with no forethought yields arbitrary distribution of amplitude.[128] What we seek here are specific filters which yield a perfectly uniform amplitude distribution over one period M of the filtered sequence while simultaneously shaping the white spectrum of the single-bit PN sequence. We make no attempt to enumerate all the appropriate filter types, but we do investigate a few designs in detail which incorporate the finite impulse response (FIR) filter. FIR filtering turns out to be the proper interpretation of the multibit output from the PN generator. Beyond these few goals, we make no further attempt to investigate colored noise generation.[129]

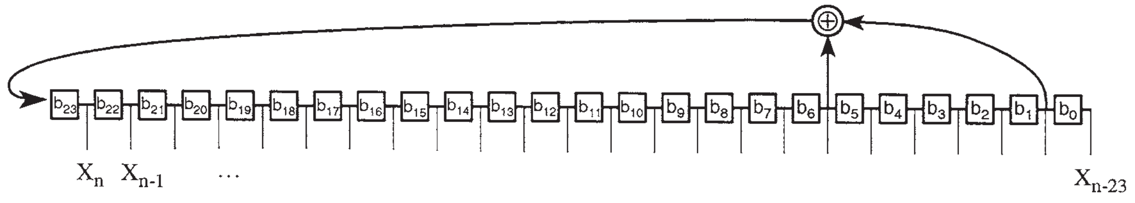
8.2 Recursive PN Circuit

We present two example PN generator circuits for audio in Fig. 60, each showing all the individual bits b_i of one 24-bit register such as would be found inside a typical DSP chip [1], [49]. The symbol $\oplus$ is the notation for eXclusive – OR logic or, equivalent, modulo 2 binary addition. At each sample period T the logic is performed on the

[127] Say that f_0 is our desired transposition frequency. Then a simple way to enhance our gurgling brook would be to transpose two independent noise generators; one of them to $f_0 + f_\varepsilon$ the other to $f_0 - f_\varepsilon$ where f_ε is only a few hertz. By so doing we dismantle the spectral symmetry of the noise that would otherwise exist about f_0.

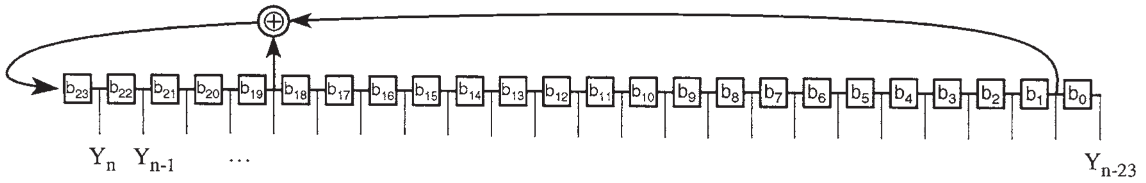
[128] The central limit theorem [56], [61], [62], [66]–[68] predicts that under certain conditions, recursive filtering of a signal (having pretty much any statistical distribution of amplitude) will tend to make the amplitude distribution of the filtered output Gaussian.

[129] The term "pink noise" refers to a random process having a power spectrum that falls as 3 dB per octave, thus its power Eq. (37) over any octave interval is the same. Often employed in the audio field because it is subjectively spectrally white, its power spectrum is proportional to $1/f$ in the linear frequency variable f. More algorithms and computer programs for colored noise generation and stochastic processes can be found in [69].



b23 b22 b21 b20 b19 b18 b17 b16 b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
⊕
Xn Xn-1 …
Xn-23

(a)



b23 b22 b21 b20 b19 b18 b17 b16 b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
⊕
Yn Yn-1 …
Yn-23

(b)

Fig. 60. Two maximal-length PN generators. *word length* = 23 bit. (a) Generator equation: $X_{n+1} = b_{23}[n + 1] = b_6[n] \oplus b_1[n]$. (b) Its reciprocal: $Y_{n+1} = b_{23}[n + 1] = b_{19}[n] \oplus b_1[n]$. For K-bit-wide output, take K unsigned MSBs; $K \in 1, 2, \ldots, 24$.

selected bits and then the whole register is shifted right by 1 bit, accepting the logical result into the most significant bit (MSB). When the PN generator's feedback entry port is at the MSB, we say that the generator is left-justified within the register. For any left-justified PN generator,

$$b_{i-1}[n+1] = b_i[n]\ , \qquad 0 < i \leq 23\ .$$

Each PN generator in Fig. 60 then is fundamentally a single-bit shift register having feedback.

The PN generator output for each example in Fig. 60 is taken to be the succession of 24-bit-wide *unsigned* register values at the rate $1/T$. The MSBs would be selected from each respective example for fewer than 24 bit of desired output from a 24-bit register. Strictly speaking, the proper bit-width output for these examples is 23 MSBs because the generator *word length* is 23 bit. The *word length* of the left-justified generator is determined by the location of the rightmost logic input.

Using the logic shown in Fig. 60(a), a uniformly distributed asymptotically uncorrelated[130] PN sequence of length $M = 2^{23} - 1$ is generated (ignoring bit b_0) comprising unique 23-bit words. In Fig. 60(b) another uncorrelated maximal-length PN sequence is generated, asymptotically uncorrelated to itself. We shall examine only autocorrelation in detail.

As suggested by the circuits in Fig. 60, the unsigned register value 0 cannot be produced in general. To start a given generator, the significant bits of its register are initialized with any positive value. Using a different initial register value only shifts the periodic sequence in time, that is, the same sequence is started at a different phase of its $M = 2^{\textit{word length}} - 1$ long cycle.

Both example circuits in Fig. 60 are derived from *word length* entry 23 in Table 9. The generator equation for Fig.

[130] We adapt the term "asymptotic" [62] here to mean that the circular sample autocovariance of a PN sequence is not a periodic Kronecker delta. Rather it is periodically spiky, decaying to near zero at midcycle. One cycle of such a spiky autocovariance might have a shape like that in Fig. 67.

[131] The time index $[n]$ appears once for each logic equation in Table 9 because all time indices on the right-hand side are the same. The logic equations themselves are not solitary; that is, there exist more generators for a given *word length* that are maximal length and unique in PN sequence. More generator equations can be found in [61, ch. 7.4], [65], [60] and in Appendix 6, Section 8.6.

Table 9. Generator equations for maximal-length PN sequences.[131]

word length	generator	word length	generator
1	$b_0[n+1] = b_0[n]$	33	$b_{32}[n+1] = b_{13} \oplus b_0[n]$
2	$b_1[n+1] = b_1 \oplus b_0[n]$	34	$b_{33}[n+1] = b_{15} \oplus b_{14} \oplus b_1 \oplus b_0[n]$
3	$b_2[n+1] = b_1 \oplus b_0[n]$	35	$b_{34}[n+1] = b_2 \oplus b_0[n]$
4	$b_3[n+1] = b_1 \oplus b_0[n]$	36	$b_{35}[n+1] = b_{11} \oplus b_0[n]$
5	$b_4[n+1] = b_2 \oplus b_0[n]$	37	$b_{36}[n+1] = b_{12} \oplus b_{10} \oplus b_2 \oplus b_0[n]$
6	$b_5[n+1] = b_1 \oplus b_0[n]$	38	$b_{37}[n+1] = b_6 \oplus b_5 \oplus b_1 \oplus b_0[n]$
7	$b_6[n+1] = b_1 \oplus b_0[n]$	39	$b_{38}[n+1] = b_4 \oplus b_0[n]$
8	$b_7[n+1] = b_6 \oplus b_5 \oplus b_1 \oplus b_0[n]$	40	$b_{39}[n+1] = b_{21} \oplus b_{19} \oplus b_2 \oplus b_0[n]$
9	$b_8[n+1] = b_4 \oplus b_0[n]$	41	$b_{40}[n+1] = b_3 \oplus b_0[n]$
10	$b_9[n+1] = b_3 \oplus b_0[n]$	42	$b_{41}[n+1] = b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0[n]$
11	$b_{10}[n+1] = b_2 \oplus b_0[n]$	43	$b_{42}[n+1] = b_6 \oplus b_4 \oplus b_3 \oplus b_0[n]$
12	$b_{11}[n+1] = b_7 \oplus b_4 \oplus b_3 \oplus b_0[n]$	44	$b_{43}[n+1] = b_6 \oplus b_5 \oplus b_2 \oplus b_0[n]$
13	$b_{12}[n+1] = b_4 \oplus b_3 \oplus b_1 \oplus b_0[n]$	45	$b_{44}[n+1] = b_4 \oplus b_3 \oplus b_1 \oplus b_0[n]$
14	$b_{13}[n+1] = b_{12} \oplus b_{11} \oplus b_1 \oplus b_0[n]$	46	$b_{45}[n+1] = b_8 \oplus b_5 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0[n]$
15	$b_{14}[n+1] = b_1 \oplus b_0[n]$	47	$b_{46}[n+1] = b_5 \oplus b_0[n]$
16	$b_{15}[n+1] = b_5 \oplus b_3 \oplus b_2 \oplus b_0[n]$	48	$b_{47}[n+1] = b_7 \oplus b_5 \oplus b_4 \oplus b_2 \oplus b_1 \oplus b_0[n]$
17	$b_{16}[n+1] = b_3 \oplus b_0[n]$	49	$b_{48}[n+1] = b_6 \oplus b_5 \oplus b_4 \oplus b_0[n]$
18	$b_{17}[n+1] = b_7 \oplus b_0[n]$	50	$b_{49}[n+1] = b_4 \oplus b_3 \oplus b_2 \oplus b_0[n]$
19	$b_{18}[n+1] = b_6 \oplus b_5 \oplus b_1 \oplus b_0[n]$	51	$b_{50}[n+1] = b_6 \oplus b_3 \oplus b_1 \oplus b_0[n]$
20	$b_{19}[n+1] = b_3 \oplus b_0[n]$	52	$b_{51}[n+1] = b_3 \oplus b_0[n]$
21	$b_{20}[n+1] = b_2 \oplus b_0[n]$	53	$b_{52}[n+1] = b_6 \oplus b_2 \oplus b_1 \oplus b_0[n]$
22	$b_{21}[n+1] = b_1 \oplus b_0[n]$	54	$b_{53}[n+1] = b_6 \oplus b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_0[n]$
23	$b_{22}[n+1] = b_5 \oplus b_0[n]$	55	$b_{54}[n+1] = b_6 \oplus b_2 \oplus b_1 \oplus b_0[n]$
24	$b_{23}[n+1] = b_4 \oplus b_3 \oplus b_1 \oplus b_0[n]$	56	$b_{55}[n+1] = b_7 \oplus b_4 \oplus b_2 \oplus b_0[n]$
25	$b_{24}[n+1] = b_3 \oplus b_0[n]$	57	$b_{56}[n+1] = b_5 \oplus b_3 \oplus b_2 \oplus b_0[n]$
26	$b_{25}[n+1] = b_8 \oplus b_7 \oplus b_1 \oplus b_0[n]$	58	$b_{57}[n+1] = b_6 \oplus b_5 \oplus b_1 \oplus b_0[n]$
27	$b_{26}[n+1] = b_5 \oplus b_2 \oplus b_1 \oplus b_0[n]$	59	$b_{58}[n+1] = b_6 \oplus b_5 \oplus b_4 \oplus b_3 \oplus b_1 \oplus b_0[n]$
28	$b_{27}[n+1] = b_3 \oplus b_0[n]$	60	$b_{59}[n+1] = b_1 \oplus b_0[n]$
29	$b_{28}[n+1] = b_2 \oplus b_0[n]$	61	$b_{60}[n+1] = b_5 \oplus b_2 \oplus b_1 \oplus b_0[n]$
30	$b_{29}[n+1] = b_{16} \oplus b_{15} \oplus b_1 \oplus b_0[n]$	62	$b_{61}[n+1] = b_6 \oplus b_5 \oplus b_3 \oplus b_0[n]$
31	$b_{30}[n+1] = b_3 \oplus b_0[n]$	63	$b_{62}[n+1] = b_1 \oplus b_0[n]$
32	$b_{31}[n+1] = b_{28} \oplus b_{27} \oplus b_1 \oplus b_0[n]$	64	$b_{63}[n+1] = b_4 \oplus b_3 \oplus b_1 \oplus b_0[n]$

60(a) simply adds 1 to the bit indices of entry 23 to account for the left justification by 1 bit within the 24-bit register.[132] The *reciprocal* PN generator, Fig. 60(b), is derived by reversing the bit indices of entry 23, that is,

$$b_i[n] \rightarrow b_{word\ length-i}[n]$$

then add 1 to the indices for justification. The reciprocal generator produces an asymptotically uncorrelated sequence of the same maximal length [65, Eq. (15)].[133]

Table 9 notation is for a *word length* bit register, but Fig. 60 demonstrates how the implementation is translated to a more suitable left-justified format within a 24-bit register. Hence we can fit any one of the first 24 generator logic equations from Table 9, each of differing *word length*, into a 24-bit register, and so on. The required justification left is (24 − *word length*) bits. Fig. 60 then shows two of all the possible *word length* bit generators (see Section 8.6, Appendix 6) that might occupy a 24-bit register. Keep in mind that *word length* (= 23 in Fig. 60) is a variable selected by the designer.

The number of terms in the generator equation is similarly a design variable. Here is an example having more terms and a 16-bit *word length*:

Nonjustified 16-bit generator	$b_{15}[n+1] = b_5[n] \oplus b_3[n] \oplus b_2[n] \oplus b_0[n]$
Reciprocal generator	$b_{15}[n+1] = b_{16\text{-}2}[n] \oplus b_{16\text{-}3}[n] \oplus b_{16\text{-}5}[n] \oplus b_0[n]$
	$= b_{14}[n] \oplus b_{13}[n] \oplus b_{11}[n] \oplus b_0[n]$
Left-justified 16-bit generator	$b_{23}[n+1] = b_{13}[n] \oplus b_{11}[n] \oplus b_{10}[n] \oplus b_8[n]$
Reciprocal generator	$b_{23}[n+1] = b_{22}[n] \oplus b_{21}[n] \oplus b_{19}[n] \oplus b_8[n]$

On the other hand, Fig. 60 also represents two of many possible logic configurations for a *fixed* 23-bit *word length*. For example, given an 8-bit desired *word length*, we find 12 (but only 12) maximal-length PN sequences using four-term generators.[134] Those unique asymptotically uncorrelated sequences are listed in Table 10 in terms of their generator equations, and were found by brute force examination of the periodicity of all possible sequences. But there are more elegant methods of finding generator equations by formulating the search in terms of representative polynomials. The following *Mathematica* script produces 0 as output when the candidate polynomial would generate a maximal-length PN sequence:

```
wordlength = 8;        (* execution time proportional to wordlength *)
p[x_] = x^wordlength + x^6 + x^5 + x^1 + 1;    (* candidate generator *)
px = Simplify[x^wordlength p[1/x]];
PolynomialMod[PolynomialRemainder[x^(2^wordlength − 1) + 1, px, x], 2]
```

The generator polynomial is related to the Table 9 (or Table 10) *word length* entry as in the following 8-bit *word length* example:

8-bit polynomial	$x^8 + x^6 + x^5 + x + 1 \leftrightarrow b_7[n+1] = b_6[n] \oplus b_5[n] \oplus b_1[n] \oplus b_0[n]$
Reciprocal polynomial	$x^8 + x^7 + x^3 + x^2 + 1 \leftrightarrow b_7[n+1] = b_7[n] \oplus b_3[n] \oplus b_2[n] \oplus b_0[n]$.

We do not pursue polynomial theory any further here. The ardent reader is pointed to [65]. We provide an exhaustive listing of two- and four-term generators for a limited number of useful *word lengths* in Section 8.6, Appendix 6. Table 10 provides an exhaustive list for an 8-bit *word length*. Each generator equation in Table 10 produces a unique PN sequence.

As long as the maximal-length PN sequence is of sufficient duration that its period (in seconds) MT is imperceptible, every generator equation is suitable for audio, and the corresponding PN sequence so synthesized sounds like any other over long duration.

8.3 Single-Bit PN Generator

Before proceeding to the multibit case, we first examine the power spectrum and circular (sample) autocovariance of the single-bit maximal-length PN sequence,

$$X_n \equiv b_{23}[n]$$

where bit b_{23} is shown in Fig. 60. Using the language of DSP, we wish to verify that the circular autocovariance of

[132] We did not choose *word length* entry 24 because we anticipate the need for an extra bit of precision if we wish to properly convert the unsigned register to two's complement bipolar format. We discuss an example of this conversion later.

[133] When multiple uncorrelated noise sources are required, we may try the next largest *word length* from Table 9 that fits the available register width.

[134] The number of terms (operands) in the generator equations is always even. There are no two-term generators in the 8-bit case, as suggested by Table 9. (See Appendix 6, Section 8.6.1.)

the periodic single-bit PN sequence is indeed a periodic Kronecker delta, and we wish to characterize and quantify the parameters of the corresponding white power spectrum $|X_n[k]|^2/M$, namely, to show that the true power spectrum of the single-bit PN sequence is indeed white. $X_n[k]$ is the length-N DFT of the $(N - M)$-zero padded PN sequence X_n

$$X_n[k] = \sum_{n=0}^{N-1} X_n e^{-\mathrm{j}(2\pi k/N)n} .$$

The sequence period M is always odd for maximal-length PN sequences,

$$M = 2^{\textit{word length}} - 1 .$$

As we will explain, the true power spectrum can be found when $N = M$.

The single-bit PN generator topology is the same as that for the circuits of Fig. 60, except that only the MSB is observed at the output. Because the single-bit sequence has only two possible amplitudes {0, 1}, we calculate and then remove the dc component (the mean) of the PN sequence when we view its circular (sample) autocorrelation. We allow the dc component to remain in the calculation of the power spectrum, however. The precise calculation of sample variance and mean proceeds as follows:

$$\begin{aligned}\sigma^2_{X_n} &= \frac{1}{N}\sum_{k=0}^{N-1}\frac{|X_n[k]|^2}{M} - m^2_{X_n} = \frac{1}{M}\sum_{n=0}^{M-1}X_n^2 - \left(\frac{1}{M}\sum_{n=0}^{M-1}X_n\right)^2 \\ &= \frac{1}{N}\sum_{k=1}^{N-1}\frac{|X_n[k]|^2}{M} - m^2_{X_n}\frac{N-M}{N}, \qquad M \le N \\ m_{X_n} &= \frac{X_n[0]}{M} = \frac{1}{M}\sum_{n=0}^{M-1}X_n = \frac{1}{M}\sum_{n=0}^{N-1}X_n, \qquad X_M, X_{M+1}, \ldots, X_{N-1} = 0 .\end{aligned} \tag{63}$$

Eq. (63) follows directly from the standard definitions [62], [66], [67]. The relationship to the frequency domain $X_n[k]$ comes from Parseval's relation for the DFT [14]. The DFT length N is typically (but not necessarily) even because we often calculate the DFT using an FFT algorithm which demands that constraint. That is why Eq. (63) and later Eq. (65) account for a potential disparity between the original sequence length M (which is here odd) and the DFT length N.

Because any single-bit maximal-length unipolar PN sequence has $2^{\textit{word length}-1}$ countable ones {1} per period M, the numerical value of the sample variance and mean over one period M can be predetermined exactly via Eq. (63),

$$\begin{aligned}\sigma^2_{X_n} &= \frac{1}{4}\cdot\frac{1-2^{-(\textit{word length}-1)}}{(1-2^{-\textit{word length}})^2} \\ m_{X_n} &= \frac{1}{2}\cdot\frac{1}{1-2^{-\textit{word length}}} .\end{aligned} \tag{64}$$

In other words, for any given period M there appears one more 1 than there appear {0} [65]. The distribution of amplitude is then discrete {0, 1} and nearly uniform. Hence the sample variance and mean asymptotically approach ¼ and ½ respectively, with increasing *word length*.

The proper term for the mean-removed autocorrelation is autocovariance [67].[135] Both the circular autocovariance and the circular autocorrelation can be expressed in terms of the power spectrum of the single-bit PN sequence X_n

$$\begin{aligned}C_{X_n}[l] &= R_{X_n}[l] - m^2_{X_n} \\ C_{X_n}[l] &= \underset{N}{\mathrm{IDFT}}\left\{\frac{|X_n[k]|^2}{M}\right\} - m^2_{X_n} \\ C_{X_n}[0] &= \sigma^2_{X_n}\end{aligned} \tag{65}$$

where $C_{X_n}[l]$ is the circular autocovariance (for two-sided lag index l) of the PN sequence X_n calculated by inverse discrete Fourier transform (IDFT) of the same length N as the DFT $X_n[k]$, $R_{X_n}[l]$ is the circular autocorrelation, and M is the original record length (rather, the period in the case of PN sequences).

Table 10. Four-term 8-bit maximal-length PN sequence generator equations.

word length	generator
8	$b_7[n+1] = b_4[n] \oplus b_3[n] \oplus b_2[n] \oplus b_0[n]$
8	$b_7[n+1] = b_5[n] \oplus b_3[n] \oplus b_1[n] \oplus b_0[n]$
8	$b_7[n+1] = b_5[n] \oplus b_3[n] \oplus b_2[n] \oplus b_0[n]$
8	$b_7[n+1] = b_6[n] \oplus b_3[n] \oplus b_2[n] \oplus b_0[n]$
8	$b_7[n+1] = b_6[n] \oplus b_5[n] \oplus b_1[n] \oplus b_0[n]$
8	$b_7[n+1] = b_6[n] \oplus b_5[n] \oplus b_2[n] \oplus b_0[n]$
8	$b_7[n+1] = b_6[n] \oplus b_5[n] \oplus b_3[n] \oplus b_0[n]$
8	$b_7[n+1] = b_6[n] \oplus b_5[n] \oplus b_4[n] \oplus b_0[n]$
8	$b_7[n+1] = b_7[n] \oplus b_2[n] \oplus b_1[n] \oplus b_0[n]$
8	$b_7[n+1] = b_7[n] \oplus b_3[n] \oplus b_2[n] \oplus b_0[n]$
8	$b_7[n+1] = b_7[n] \oplus b_5[n] \oplus b_3[n] \oplus b_0[n]$
8	$b_7[n+1] = b_7[n] \oplus b_6[n] \oplus b_1[n] \oplus b_0[n]$

[135] We will view the *circular* [14, ch. 11.7.1] as opposed to the linear autocovariance, which is appropriate and justified by the fact that the PN sequence is periodic in M.

8.3.1 Single-Bit PN Sequence Power Spectrum and Autocovariance

Fig. 61 shows the expected power spectrum as it would be calculated by a nonwindowed zero-padded length-N DFT of the single-bit PN sequence. The power spectrum of the original length-M PN sequence is truly discrete in the Fourier sense because the sequence is periodic in M. Since M is not necessarily equal to N, the expected power-spectrum levels would be as given in Fig. 61.

The levels shown in Fig. 61 are termed "expected" because the power spectrum of the single-bit PN sequence calculated using a length-N DFT for $N \neq M$ is generally not so flat. Hence in Fig. 61 each non-dc expected level should be considered to be the average of the calculated levels over the frequency index k. Indeed, from Eq. (63)

$$\sigma^2_{X_n}\frac{N}{N-1} + m^2_{X_n}\frac{N-M}{N-1} = \frac{1}{N-1}\sum_{k=1}^{N-1}\frac{\left|X_n[k]\right|^2}{M} .$$

For the analysis of periodic sequences, the most accurate estimation of the power spectrum would have N set to be the same as M, the period length, as is well known. The true power-spectral levels for the original periodic sequence would be seen by letting

$$N \rightarrow M \qquad \text{and} \qquad \text{indices}\{N/2,\ N/2-1\} \rightarrow (M-1)/2$$

in Fig. 61, that is, were the power spectrum calculated instead using a DFT having length M. When $N = M$, the power-spectral levels shown in Fig. 61 are the *true* (not average) spectral levels. In other words, the single-bit sequence has the exact spectral shape shown in Fig. 61 when measured properly, that is, when $N = M$. For then the power spectrum levels in Fig. 61 converge in the Fourier sense to their true values,

$$\frac{\left|X_n[k]\right|^2}{M} = \begin{cases} \sigma^2_{X_n}\dfrac{M}{M-1}, & k = 1,\ldots,M-1;\ N \longrightarrow M \\ Mm^2_{X_n}, & k = 0 \end{cases} \tag{66}$$

which is constant, hence flat over $k \neq 0 \bmod M$. We forgo plotting an actual power spectrum for a single-bit PN generator because it is identical to the theoretical drawing in Fig. 61 when $N \rightarrow M$. The dc component level [Eq. (63)] at $k = 0$ is insensitive to the choice of N.

It seems that we have gone to much trouble to distinguish between the DFT length N and the original sequence length M when the development would have been much simpler had we made these two lengths equivalent. Although simplicity is always desirable, the present state of the equations should make our findings easier to corroborate as power-of-2 FFT algorithms are commonplace. Because DFT algorithms are more appropriate, we will discontinue this distinction between N and M from here on, that is, we will now let $N = M$.

Because the true power spectrum is white (flat, except for dc), the single-bit circular autocovariance is a periodic Kronecker delta, as postulated. More precisely, from Eq. (65) we may write the (length-M) IDFT,

$$\begin{aligned} C_{X_n}[l] &= \frac{1}{M}\left\{Mm^2_{X_n} + \sum_{k=1}^{M-1}\sigma^2_{X_n}\frac{M}{M-1}e^{j2\pi kl/M}\right\} - m^2_{X_n}, \qquad -\infty < l < \infty \\ &= \frac{\sigma^2_{X_n}}{M-1}\left(M\sum_{q=-\infty}^{\infty}\delta[l-qM] - 1\right) \end{aligned} \tag{67}$$

for the Kronecker delta δ, which becomes periodic in M. Note that Eq. (67) has a small negative offset [60, ch. 5], [65].

8.4 Multibit PN Generator

We now examine the case of the multibit output maximal-length PN sequence generator. Strictly speaking, all amplitude in DSP is discrete, hence all associated distribution of probability must also be. For example, the single-bit PN generator already considered has only two possible amplitudes {0, 1} asymptotically (with increasing *word length*) equal in distribution over one period M of the corresponding uncorrelated PN sequence. In either the single- or the multibit case of a maximal-length PN generator, the distribution of probability is discrete. In this section we therefore think in terms of the more appropriate (discrete) probability mass function (pmf) [62], [67] rather than its continuous amplitude counterpart, called the probability density function. The generator equations given in the various tables each realize a unique asymptotically uncorrelated maximal-length PN sequence having perfectly uniform pmf when applied as in the circuit of Fig. 60 and when a multibit output is observed over one period M. The pmf is uniform because each *word length* bit value produced by the generator is equiprobable over

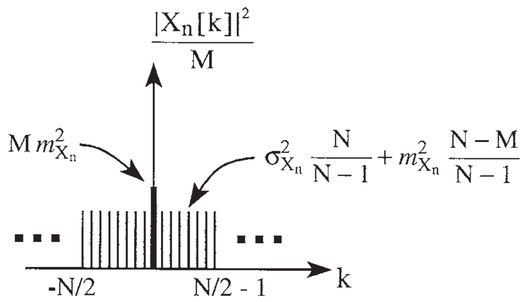


Fig. 61. Expected levels of single-bit PN power spectrum. M—record length; N—length of DFT. Power spectrum is periodic in N as implied by ellipses.

one period M [65]. The sequence is maximal length because every *word length* bit unsigned value (except 0) is generated only once before the sequence repeats. The sequence is noiselike because the successive values are produced in a seemingly incoherent order and the period of repetition MT is long in duration. Using the maximal-length generators given in Table 9, Table 10, or Appendix 6, the sequence length is the same as for the single-bit case: $M = 2^{word\ length} - 1$ samples. This length is the longest possible for a given *word length* since the unsigned register value 0 is precluded.

8.4.1 Digital Filter Interpretation of the Multibit PN Process

In practice it is customary to take the full 24-bit unsigned register as the output. But it is understood that a PN generator nestled within a 24-bit register may be designed to use an abbreviation of that register. The bits to the right of the *word length* MSBs may be masked in that case, but it is easier to accept them as they likely have little audible consequence. Alternately, some implementations may send fewer than *word length* MSBs to ensuing circuitry. When we speak of "uniform probability mass," however, we are implicitly referring here to the distribution of amplitude, over period M, ascribed to the *word length* MSBs that constitute the generator. Regardless of the constraints imposed by a particular implementation, one should always choose the MSBs from the left-justified format that we propose in Fig. 60.

We have yet to discuss exactly how to numerically convert the unsigned register $\underline{X}$ depicted in Fig. 60 to a more suitable word format. When a multibit unipolar (0, 1) (non-negative) PN sequence X is desired, the register MSB must be cleared like in Fig. 62(a)[136] because most contemporary DSP chips default to two's complement format, which is bipolar [70], [1], [71], [49]. But to convert the unsigned register to multi-bit bipolar $(-1, 1)$ format, one must first convert to two's complement unipolar format. After the ensuing operations indicated in Fig. 62(b), the resulting sequence $\tilde{X}$ may then safely be interpreted as two's complement bipolar.

This next point is critical. If one simply casts the unsigned generator register to integer (two's complement) format $\check{X}$ without clearing the MSB, as illustrated in Fig. 62(c), the consequent audio noise power spectrum suffers a deep cut in the low-frequency region.

To prove the foregoing statements, we must understand what happens to the single-bit PN sequence X_n in its conversion to word format. Fig. 63 shows the example of Fig. 60(a), but this time having weights h_i attached to each of the bits, which are then summed in a linear fashion. But the sum Σ in Fig. 63 is simply a depiction of the numerical definition of two's complement format (see Section 9.1.1, Appendix 7, Fig. 70).

$$\left.\begin{array}{l}\begin{cases} h_0 = -1 \\ h_i = 2^{-i} \end{cases} \quad ; \quad \check{X}, \quad \text{two's complement cast q23} \\ \\ \begin{cases} h_0 = -2 \\ h_i = 2^{-i+1} \end{cases} \quad ; \quad \check{X}, \quad \text{two's complement cast q22} \\ \quad\vdots \end{array}\right\} , \qquad i = 1, \ldots, 23 .$$

Having a slightly different set of weights, the sum Σ could also be a depiction of unsigned or two's complement unipolar format,

$$\left.\begin{array}{lll} h_i = 2^{-i-1} ; & \underline{X}, & \text{unsigned q24} \\ h_i = 2^{-i} ; & \underline{X}, & \text{unsigned q23} \\ h_i = 2^{-i+1} ; & \underline{X}, & \text{unsigned q22} \\ \vdots & & \end{array}\right\} , \qquad i = 0, \ldots, 23$$

or

$$\begin{cases} h_0 = 0 \\ h_i = 2^{-i} \end{cases} \quad ; \quad X, \quad \text{two's complement unipolar q23}, \qquad i = 1, \ldots, 23$$

$$\begin{cases} h_0 = 0 \\ h_1 = 0 \\ h_i = 2^{-i+1} \end{cases} \quad ; \quad X, \quad \text{two's complement unipolar q22}, \qquad i = 2, \ldots, 23 . \tag{68}$$

$$\vdots$$

[136] If the unsigned register is logically shifted right 1 bit, then the result is a two's complement unipolar sequence; that is the method we consider. Another way to clear the MSB is simply to mask it. (Some of the succeeding equations will require alteration if the latter method is used.)

Any chosen format is applied to the shift register that constitutes a PN generator. Fig. 63 is then an accurate multibit model of the actual generator circuit in Fig. 60(a). A single-bit PN sequence model would have all $h_i = 0$ except for $h_0 = 1$.

The single-bit causal digital filter depicted in Fig. 63 is of type IIR, direct form II. Each successive bit z^{-1} of that digital filter sees a delayed replica of the single-bit PN sequence X_n. The arrangement of the weight and single-bit PN sequence indices in Fig. 63 suggests convolution in the formation of the multibit output. Indeed, the IIR digital filter can be broken apart into recursive and nonrecursive parts, as in Fig. 64.[137]

[137] Thanks go to Jeffrey Barish for pointing this out, and for reviewing the entire manuscript.

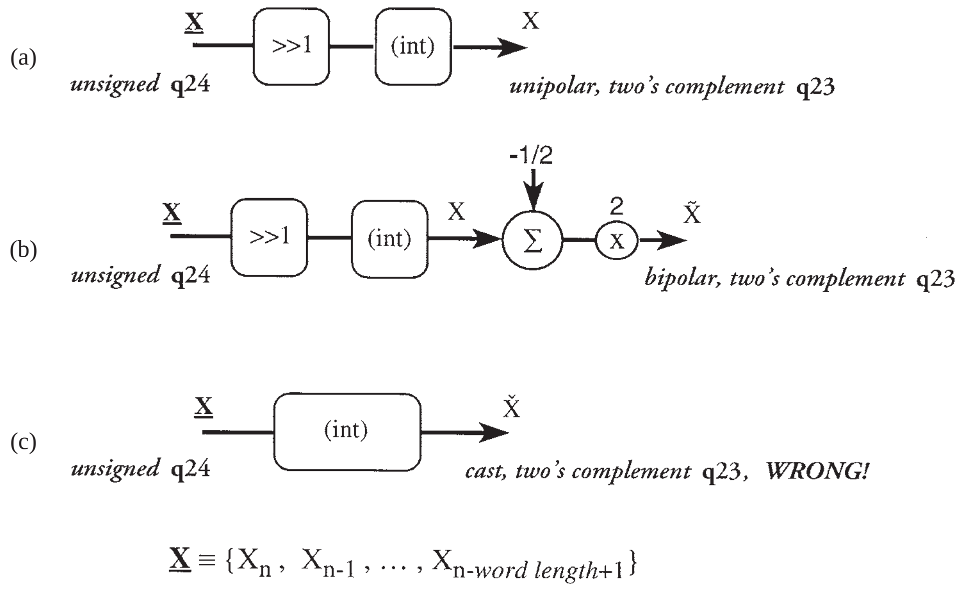

(a)
X
>>1
(int)
X
unsigned q24
unipolar, two's complement q23
(b)
-1/2
2
unsigned q24
bipolar, two's complement q23
(c)
cast, two's complement q23, WRONG!
X ≡ {Xn , Xn-1 , … , Xn-word length+1}

Fig. 62. Right way (a), (b) and wrong way (c) to take register contents. (a) Right way to convert unsigned register $\underline{X}$ to two's complement unipolar format. (b) Right way to convert to two's complement bipolar format. (c) Cast method places a deep cut at dc in audio noise power spectrum. >>1 indicates a logical shift right by 1 bit. (int) means *cast* as in C code.

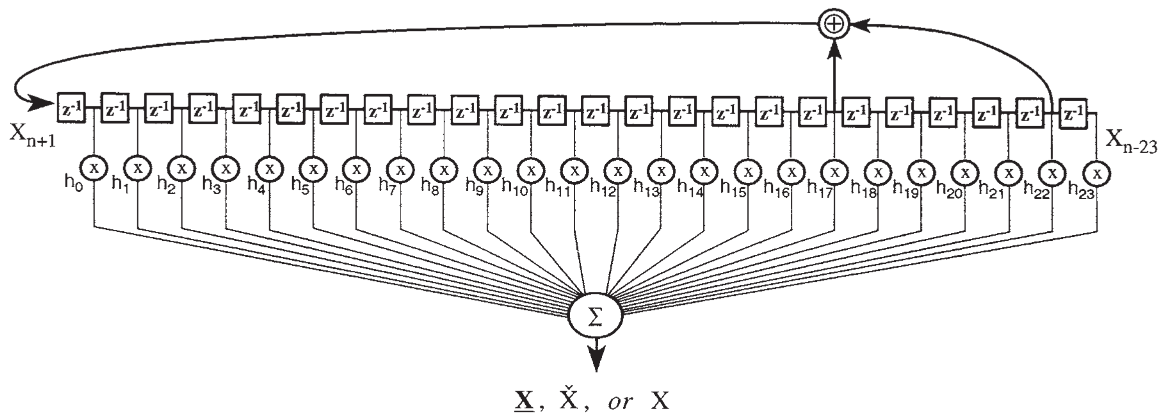


Fig. 63. Interpretation of register conversion to unsigned or two's complement as a digital filtering of single-bit PN sequence X_n.

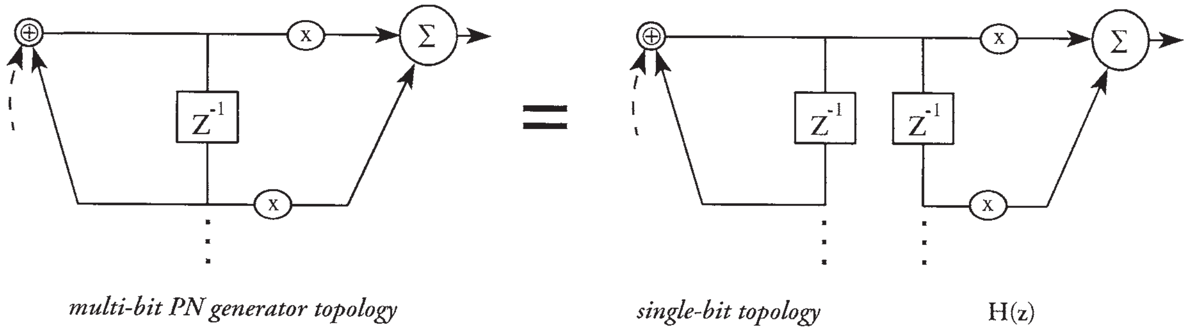


Fig. 64. Decomposition of direct form II into recursive and nonrecursive parts.

The FIR filter $H(z)$ in Fig. 64 is identified as the nonrecursive part. Its z domain transfer function depends on the chosen word format,

$$H(z) = \sum_{i=0}^{23} h_i z^{-i} \tag{69}$$

$$H_{\check{X}}(z) = -1 + \sum_{i=1}^{23} 2^{-i} z^{-i} = \frac{1 - 2^{-24} z^{-24}}{1 - \frac{1}{2} z^{-1}} - 2$$

$$\approx -\frac{1 - z^{-1}}{1 - \frac{1}{2} z^{-1}}, \qquad \text{cast, two's complement q23, } \textit{WRONG}!$$

$$H_{\underline{X}}(z) = \sum_{i=0}^{23} 2^{-i-1} z^{-i} = \frac{1}{2} \cdot \frac{1 - 2^{-24} z^{-24}}{1 - \frac{1}{2} z^{-1}}$$

$$\approx \frac{\frac{1}{2}}{1 - \frac{1}{2} z^{-1}}, \qquad \text{unsigned q24.}$$

$$zH_X(z) = z \sum_{i=1}^{23} 2^{-i} z^{-i} = z \left(\frac{1 - 2^{-24} z^{-24}}{1 - \frac{1}{2} z^{-1}} - 1 \right)$$

$$\approx \frac{\frac{1}{2}}{1 - \frac{1}{2} z^{-1}}, \qquad \text{unipolar, two's complement q23.}$$

The geometric interrelationship of the FIR coefficients allows a more compact z transform [Eq. (69)] that resembles an IIR transfer function. The large register width (24 bit) diminishes the high powers of z^{-1}, yielding the approximate transfers that are less cumbersome. The two's complement transfer function $H_{\check{X}}(z)$ is a consequence of the "wrong" way to perform the format conversion; it places a zero of transmission close to dc (Fig. 65). The others have near-unity transfer at dc, but are not flat (Fig. 66). The two's complement unipolar transfer $H_X(z)$ is premultiplied by z because the right-shift 1-bit in Fig. 62 is equivalent to a unit advance of X_n.

The deep cut into the magnitude response of the cast-method filter $H_{\check{X}}(z)$ [Fig. 62 (c)] is shown in Fig. 65. A significant range of the audio noise power spectrum is consequently lost. Fig. 66 shows the approximate magnitude response for the transfer function $H_{\underline{X}}(z)$ of unsigned word format or conversion to two's complement unipolar format $zH_X(z)$ as in Fig. 62(a); there is a loss of 9.5 dB at Nyquist. Sonically the transfer of Fig. 66 is preferable for musical purposes.

8.4.2 Multibit PN Sequence Power Spectrum and Autocovariance

The multibit PN generator is simply a digital filtering of the single-bit PN sequence. As is well known for linear

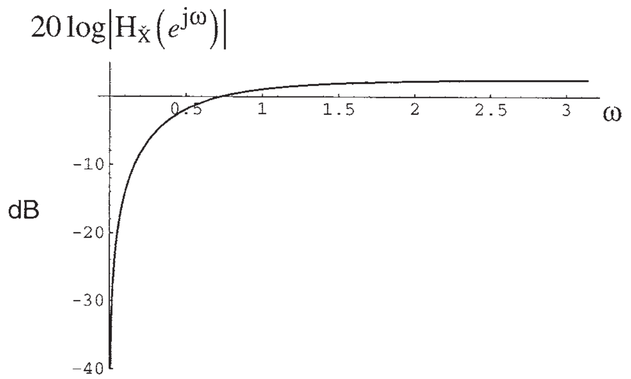


Fig. 65. Approximate magnitude response of two's complement integer-cast filter transfer function.

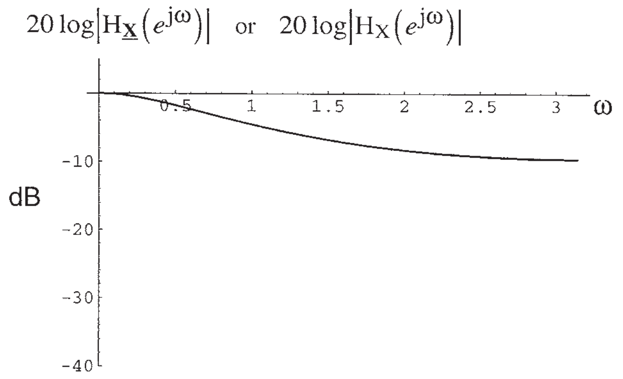


Fig. 66. Approximate magnitude response of unsigned or two's complement unipolar filter transfer function.

systems [62], [68], the expressions for filter output power spectra are simply

$$\begin{aligned}
\frac{\left|\check{X}[k]\right|^2}{M} &= \frac{\left|X_n[k]\right|^2}{M}\left|H_{\check{X}}\left(e^{\mathrm{j}2\pi k/M}\right)\right|^2 \\
\frac{\left|\underline{X}[k]\right|^2}{M} &= \frac{\left|X_n[k]\right|^2}{M}\left|H_{\underline{X}}\left(e^{\mathrm{j}2\pi k/M}\right)\right|^2 \\
\frac{\left|X[k]\right|^2}{M} &= \frac{\left|X_n[k]\right|^2}{M}\left|H_{X}\left(e^{\mathrm{j}2\pi k/M}\right)\right|^2 .
\end{aligned} \tag{70}$$

The numerical value of the sample variance and mean for the multibit maximal-length PN sequence can be predetermined exactly via expectations, simply because the pmf is *precisely* uniform over one period *M*,

$$\left.\begin{array}{lll}
\left.\begin{aligned} \sigma^2_{\underline{X}} &= \sigma^2_X = \frac{1}{12} - \frac{2^{-(word\ length-1)}}{12} \\ m_{\underline{X}} &= m_X = \frac{1}{2} \end{aligned}\right\}, & \underline{X}, X \in (0,1) \\
\left.\begin{aligned} \sigma^2_{\tilde{X}} &= \frac{1}{3} - \frac{2^{-(word\ length-1)}}{3} \\ m_{\tilde{X}} &= 0 \end{aligned}\right\}, & \tilde{X} \in (-1,1) \\
\left.\begin{aligned} \sigma^2_{\check{X}} &= \sigma^2_{\tilde{X}} + \frac{2^{word\ length} - 2}{(2^{word\ length} - 1)^2} \\ m_{\check{X}} &= \frac{1}{1 - 2^{word\ length}} \end{aligned}\right\}, & \check{X} \in [-1,1),\ \check{X} \neq 0
\end{array}\right\}, \quad word\ length > 1 . \tag{71}$$

The multibit unipolar (0, 1) and bipolar (−1, 1) [−1, 1) domains are presumed quantized to *word length* bits. The quantization Δ for both bipolar domains are identical.

With the advent of our digital filter interpretation of the multibit PN sequence, the (complex) general expression for autocovariance in terms of the autocorrelation becomes

$$\begin{aligned}
C[l] &= R[l] - \left|m^2\right| \\
m &= m_{X_n}\sum_{i=0}^{23} h_i = m_{X_n} H(1) \\
C[0] &= \sigma^2 .
\end{aligned}$$

The standard linear systems derivation finds the circular autocovariance *C*[*l*] of the multibit PN sequence via the length-*M* IDFT of Eq. (70). That calculation is expressed in Eq. (72), the multibit generalization of Eq. (65),

$$\begin{aligned}
R_{\check{X}}[l] &= C_{\check{X}}[l] + m^2_{\check{X}} = \underset{M}{\mathrm{IDFT}}\left\{\frac{\left|\check{X}[k]\right|^2}{M}\right\} \\
R_{\underline{X}}[l] &= C_{\underline{X}}[l] + m^2_{\underline{X}} = \underset{M}{\mathrm{IDFT}}\left\{\frac{\left|\underline{X}[k]\right|^2}{M}\right\} \\
R_{X}[l] &= C_{X}[l] + m^2_{X} = \underset{M}{\mathrm{IDFT}}\left\{\frac{\left|X[k]\right|^2}{M}\right\} .
\end{aligned} \tag{72}$$

The autocovariance is thus dependent on the chosen word format. That particular format determines *H*(*z*) as in Eq. (69), namely,

$$\begin{aligned}
C_{\check{X}}[l] &= C[l]\big|_{H(z)\,=\,H_{\check{X}}(z)} \\
C_{\underline{X}}[l] &= C[l]\big|_{H(z)\,=\,H_{\underline{X}}(z)} \\
C_{X}[l] &= C[l]\big|_{H(z)\,=\,H_{X}(z)} .
\end{aligned} \tag{73}$$

The simpler time-domain derivation of circular autocovariance [Eq. (74)] in [62, ch. 6.2] requires the digital filter *H*(*z*) to be causal, and assumes that the single-bit noise process $\{X_n\}$ is weakly stationary and two sided (in time),

$$\begin{aligned}
C[l] &= \frac{\sigma^2_{X_n}}{M-1}\left(M\sum_{q=-\infty}^{\infty}\delta[l - qM] * \sum_{i=0}^{23} h_i h^*_{i-l} - \sum_{i=0}^{23} h_i \sum_{r=0}^{23} h^*_r\right) \\
&= \frac{\sigma^2_{X_n}}{M-1}\left(M\sum_{q=-\infty}^{\infty}\delta[l - qM] * Z^{-1}\left\{H(z)\,H^*\left(\frac{1}{z^*}\right)\right\} - \left|H(1)\right|^2\right)
\end{aligned} \tag{74}$$

where the centered asterisks $*$ denote linear convolution (with precedence), while the asterisk superscripts denote conjugation. The inverse z transform in Eq. (74) is a function of lag l (which extends infinitely in both directions, as before). We note a strong resemblance of Eqs. (74) and (67). The periodic Kronecker delta of Eq. (67) now undergoes convolution in Eq. (74). The trailing 1 in Eq. (67) becomes $|H(1)|^2$.

The correct evaluation of Eq. (74) assumes that the generator *word length* is properly related to the register width, as in Table 11. As stated previously, it is *not* necessary to mask off bits to the right of short *word length* MSBs for musical purposes. Hence the relationships in Table 11 are purely theoretical.

8.4.3 Two's Complement Bipolar Word Format

We now examine the important case of a uniform pmf two's complement bipolar q23 (Section 9.1, Appendix 7) PN sequence as in Fig. 62(b). Given the current trend in DSP chip design which defaults to two's complement format [70], [1], [71], [49], this multibit bipolar case is important because it is probably the most desirable (spectrally and numerically) of the format conversions that we have discussed. The true power spectrum can be determined in two ways. The first way is via frequency-domain interpretation of Fig. 62(b) itself,

$$\frac{\left|\tilde{X}[k]\right|^2}{M} = \left(\frac{\left|X[k]\right|^2}{M} - \frac{\left|\underset{M}{\mathrm{DFT}}\{1/2\}\right|^2}{M}\right)4$$

$$= 4\,\frac{\left|X_n[k]\right|^2}{M}\left|H_X(e^{\,j2\pi k/M})\right|^2 - M\,\delta[k \bmod M] \qquad (75)$$

where δ is the Kronecker delta. The power spectrum $|X_n[k]|^2/M$ of the single-bit PN sequence is shown in Fig. 61. (Recall that the power spectrum converges to its true value [Eq. (66)] when $N = M$.)

The second way to determine the true power spectrum of this multibit sequence follows from the DFT of the circular sample autocorrelation

$$\begin{aligned} R_{\tilde{X}}[l] &= C_{\tilde{X}}[l] + m_{\tilde{X}}^2 \\ C_{\tilde{X}}[l] &= 4\,C_X[l] \\ m_{\tilde{X}} &= 0 \\ \sigma_{\tilde{X}}^2 &= C_{\tilde{X}}[0] \end{aligned} \qquad (76)$$

for $C_X[l]$ as in Eqs. (73) and (74). [The sample variance is given in Eq. (71).] From Eq. (76) we get an expression for the power spectrum that is equivalent to Eq. (75),

$$\frac{\left|\tilde{X}[k]\right|^2}{M} = \underset{M}{\mathrm{DFT}}\{R_{\tilde{X}}[l]\} = \underset{M}{\mathrm{DFT}}\{4C_X[l]\}$$

$$= 4\sigma_{X_n}^2\,\frac{M}{M-1}\left(\left|H_X(e^{\,j2\pi k/M})\right|^2 - \left|H_X(1)\right|^2\delta[k \bmod M]\right). \qquad (77)$$

Proper evaluation of Eqs. (75) and (77) depends on the *word length* being exactly 1 bit less than the register width to accommodate the right-shift 1 bit in Fig. 62(b). Pertinent to our example in Figs. 60 and 63, *word length* is assumed to be 23 bit, as specified in Table 11. Also, the exact expression for $H_X(z)$ from Eq. (69) must be used. Given those caveats, we find that the true multibit bipolar PN sequence power spectrum [Eq. (75) or (77)] is zero at dc, discrete, but otherwise has the envelope shown in Fig. 66.

The plot of autocovariance in Fig. 67 is a numerical evaluation of Eq. (76) via Eqs. (73), (74), (68), and (64). Fig. 67 shows just how spiky the autocovariance of the multibit bipolar PN sequence is. It is significantly diminished for lags outside of $l = \pm 11$, but never goes [Eq. (74)] to precisely 0. The example shown has $(M - 1)/2 =$ 4 194 303, so the significant correlation is relatively brief indeed.

8.4.4 Spectral Equalization

The IIR form of the transfer functions $H(z)$ in Eq. (69) suggests that multibit PN sequence equalization is easy. Introduction of the term

$$H_{\mathrm{EQ}}(z) = (1 - 2z^{-1})\,v\,, \qquad v \text{ constant}$$

into the numerator of $zH_X(z)$ makes that transfer function become asymptotically all pass with increasing register width,

$$\begin{aligned} zH_{X_{\mathrm{EQ}}}(z) &= \left[\,zH_X(z)\,\right]H_{\mathrm{EQ}}(z) \\ &\approx \frac{\tfrac{1}{2} - z^{-1}}{1-(\tfrac{1}{2})z^{-1}}\cdot v\,. \end{aligned}$$

Fig. 68 shows how equalization is actually performed upon a two's complement bipolar PN sequence. Fig. 68 is simply a digital filtering of the output of the circuit in Fig. 62(b).

Once equalization is performed on a multibit PN sequence in this manner, the power spectrum becomes asymptotically white[138] with increasing *word length*, and the circular autocovariance becomes a periodic Kronecker delta. The drawback to equalization is that the resulting multibit sequence no longer has uniform pmf.

The equalized multibit bipolar PN sequence over one period M has sample variance asymptotically equal to the

Table 11. Theoretical relationship of *word length* to 24-bit register width.

$\underline{X}$	Unsigned	*word length* = 24 bit
$\check{X}$	Two's complement cast method	*word length* = 24 bit
X	Two's complement unipolar	*word length* = 23 bit
$\tilde{X}$	Two's complement bipolar	*word length* = 23 bit

[138] This is true assuming that the relationship between *word length* and register width as shown in Table 11 is maintained.

given constant squared v^2, namely, Eq. (78),[139]

$$\left.\begin{aligned}\sigma^2_{\tilde{X}_{EQ}} &= \left(1 + \frac{2^{-(2\,word\,length-1)}}{1 - 2^{-word\,length}}\right) v^2 \\ m_{\tilde{X}_{EQ}} &= 0\end{aligned}\right\}, \quad \tilde{X}_{EQ} \in (-1,1), \quad word\,length > 1. \tag{78}$$

We do not investigate equalization any further here because our multibit PN sequences, as derived previously, suit most of our musical purposes.

8.4.5 Uniform pmf Multibit Realization

Fig. 69 shows a brief realization (PN sequence) of a two's complement bipolar PN process generated by a circuit similar to that in Fig. 60 having format conversion as in Fig. 62(b). This sequence has sample variance approximately equal to 1/3 and mean [Eq. (71)] exactly equal to 0. The plot reveals that maximal-length PN sequence generators are fundamentally relaxation-type oscillators. The arrows in the figure highlight exponential growth and decay, which are fairly obvious in that microscopic view.

[139] The sample variance Eq. (78) is found analytically by solving the autocovariance equation $4C_{X_{EQ}}[l]$ for lag $l = 0$ [substituting the impulse response of the equalized digital filter $H_{X_{EQ}}(z)$ into $4 * $ Eq. (74)]. When Eq. (78) is verified by simulation, it is critical to initialize the old state of $H_{EQ}(z)$. The proper initialization is determined by recognizing that the filter input is periodic.

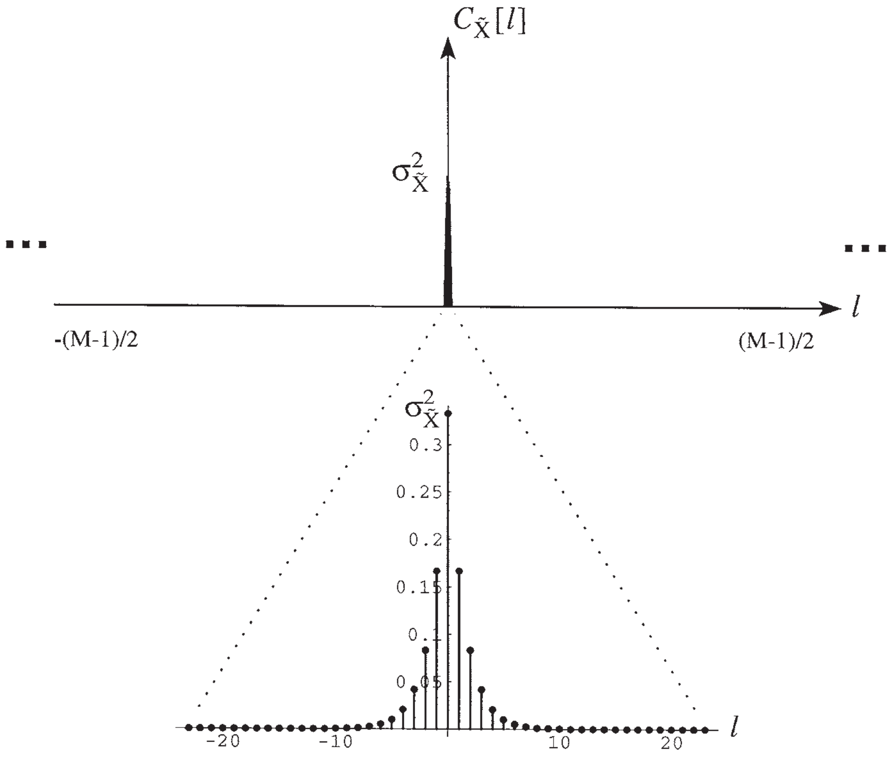


Fig. 67. Autocovariance of two's complement bipolar PN sequence having length $M = 2^{word\ length} - 1$; *word length* = 23.

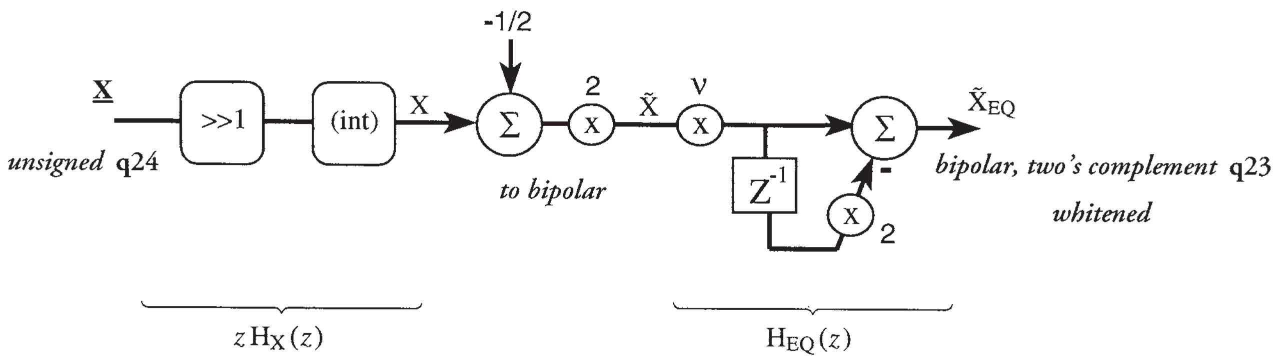


Fig. 68. Right way to convert unsigned register to two's complement bipolar format while equalizing audio noise power spectrum.

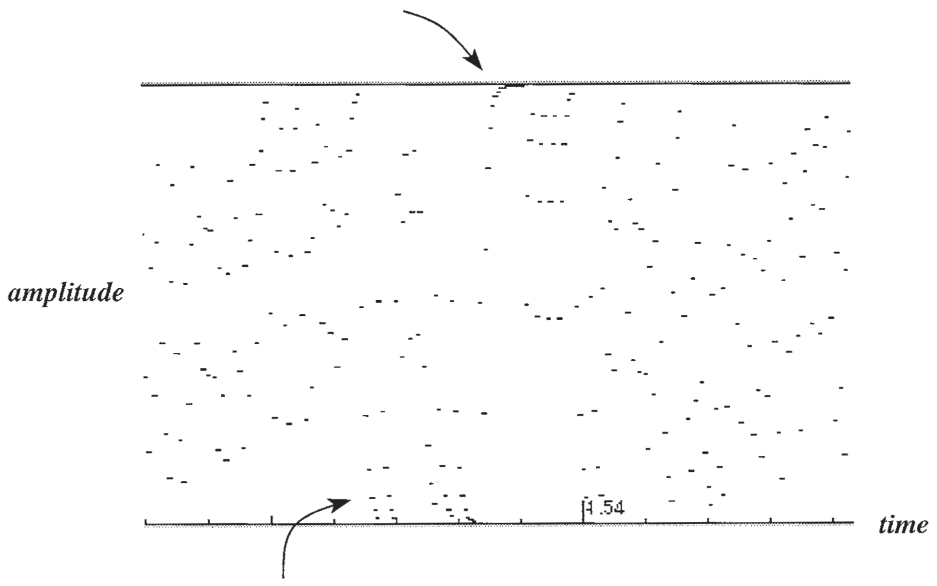


Fig. 69. Bipolar uniform pmf noise realization, exhibiting correlation.

This exponential relaxation in time is easily explained by the right-shift by 1 bit that occurs as part of the algorithm on every sample generated. Exponential relaxation in the PN sequence betrays the existence of correlation,[140] that is, the circular autocovariance of this PN sequence would not produce a periodic Kronecker delta (Fig. 67). Hence we may conclude that noise realized in this manner cannot be perfectly spectrally white (Fig. 66). Since exponential growth and decay are common physical occurrences in the real world, this may help explain why these PN generators are sonically acceptable.

8.5 Pseudnoise Conclusions

Further research is required to formulate precise expressions for the distribution of amplitude and the power spectra of functions and combinations of these deterministic PN sequences over one common period, that is, to synthesize other exacting pmfs having known power spectra [72]. We have expended some energy doing so here for the uniform pmf, and we are providing new exhaustive lists of two- and four-term PN generator equations for selected *word lengths* in Appendix 6.

From a musical perspective, the sample variance is handy to help quantify perceived loudness with respect to other sequences having possibly different pmfs. We have learned that the two's complement cast method, the "wrong" way to convert the unsigned generator register to a bipolar format, will destroy low frequencies in the audio noise power spectrum. We have demonstrated the proper way to perform the conversion as well as an easy method for equalizing the power spectrum of the multibit PN sequence.

8.6 Appendix 6: Maximal-Length PN Sequence Generator Equations

8.6.1 Two-Term Exhaustive Generator-Equation Listing

We show only the subscript of the middle term in the two-term generator equations. The last subscript is always 0 while the first (left-hand side of the earlier table equations) is *word length* − 1, that is,

$$b_{\textit{word length}-1}[n+1] = b_{\{\ \}}[n] \oplus b_0[n] \,.$$

```
Two-term 2-bit word length
maximal-length sequence generators
{ 1}
1 sequence found

Two-term 3-bit word length
{ 1} { 2}
2 sequences found

Two-term 4-bit word length
{ 1} { 3}
2 sequences found

Two-term 5-bit word length
{ 2} { 3}
2 sequences found

Two-term 6-bit word length
{ 1} { 5}
2 sequences found

Two-term 7-bit word length
{ 1} { 3} { 4} { 6}
4 sequences found
```

[140] The exponential artifacts cannot be decorrelated by choosing a different logic equation from the tables.

```
Two-term 8-bit word length
0 sequences found

Two-term 9-bit word length
{ 4} { 5}
2 sequences found

Two-term 10-bit word length
{ 3} { 7}
2 sequences found

Two-term 11-bit word length
{ 2} { 9}
2 sequences found

Two-term 12-bit word length
0 sequences found

Two-term 13-bit word length
0 sequences found

Two-term 14-bit word length
0 sequences found

Two-term 15-bit word length
{ 1} { 4} { 7} { 8} {11} {14}
6 sequences found

Two-term 16-bit word length
0 sequences found

Two-term 17-bit word length
{ 3} { 5} { 6} {11} {12} {14}
6 sequences found

Two-term 18-bit word length
{ 7} {11}
2 sequences found

Two-term 19-bit word length
0 sequences found

Two-term 20-bit word length
{ 3} {17}
2 sequences found

Two-term 21-bit word length
{ 2} {19}
2 sequences found

Two-term 22-bit word length
{ 1} {21}
2 sequences found

Two-term 23-bit word length
{ 5} { 9} {14} {18}
4 sequences found

Two-term 24-bit word length
0 sequences found
```

8.6.2 Four-Term Exhaustive Generator-Equation Listing

We show only the subscripts of the three middle terms in the four-term generator equations. The last subscript is always 0 while the first (left-hand side of the earlier table equations) is *word length* − 1, such as

$$b_4[n+1] = \{b_3[n] \oplus b_2[n] \oplus b_1[n]\} \oplus b_0[n]$$

from the first { } entry in the listing that follows.

```
Four-term 4-bit word length  maximal-length sequence generators
0 sequences found

Four-term 5-bit word length
{ 3  2  1} { 4  2  1} { 4  3  1} { 4  3  2}
4 sequences found

Four-term 6-bit word length
{ 4  3  1} { 5  2  1} { 5  3  2} { 5  4  1}
4 sequences found

Four-term 7-bit word length
{ 3  2  1} { 4  3  2} { 5  2  1} { 5  3  1} { 5  4  3} { 6  3  1} { 6  4  1} { 6  4  2}
{ 6  5  2} { 6  5  4}
10 sequences found
```

```
Four-term 8-bit word length
{ 4  3  2} { 5  3  1} { 5  3  2} { 6  3  2} { 6  5  1} { 6  5  2} { 6  5  3} { 6  5  4}
{ 7  2  1} { 7  3  2} { 7  5  3} { 7  6  1}
12 sequences found

Four-term 9-bit word length
{ 4  3  1} { 5  3  2} { 5  4  1} { 6  4  3} { 6  5  3} { 7  2  1} { 7  4  2} { 7  5  1}
{ 7  5  2} { 7  6  4} { 8  4  1} { 8  4  2} { 8  5  1} { 8  5  4} { 8  6  5} { 8  7  2}
16 sequences found

Four-term 10-bit word length
{ 4  3  1} { 5  2  1} { 5  3  2} { 6  5  2} { 7  3  1} { 7  6  2} { 8  3  2} { 8  4  3}
{ 8  5  1} { 8  5  4} { 8  6  1} { 8  7  2} { 8  7  5} { 9  4  1} { 9  4  2} { 9  5  2}
{ 9  6  1} { 9  7  3} { 9  7  6} { 9  8  5}
20 sequences found

Four-term 11-bit word length
{ 4  2  1} { 5  3  1} { 5  3  2} { 6  2  1} { 6  5  1} { 6  5  2} { 6  5  4} { 7  3  2}
{ 7  4  2} { 7  5  3} { 7  5  4} { 7  6  4} { 7  6  5} { 8  3  2} { 8  4  1} { 8  5  2}
{ 8  5  3} { 8  6  2} { 8  6  3} { 8  6  4} { 8  7  1} { 9  2  1} { 9  4  1} { 9  4  2}
{ 9  5  3} { 9  6  3} { 9  6  5} { 9  7  2} { 9  7  4} { 9  8  1} { 9  8  3} { 9  8  4}
{ 9  8  6} {10  3  1} {10  3  2} {10  4  3} {10  6  5} {10  7  2} {10  7  3} {10  8  1}
{10  8  6} {10  9  2} {10  9  5} {10  9  7}
44 sequences found

Four-term 12-bit word length
{ 6  4  1} { 6  5  3} { 7  4  3} { 7  6  4} { 8  2  1} { 8  5  1} { 8  6  5} { 8  7  2}
{ 9  3  2} { 9  7  6} { 9  8  5} {10  2  1} {10  5  4} {10  9  3} {11  7  4} {11  8  6}
{11 10  2} {11 10  4}
18 sequences found

Four-term 13-bit word length
{ 4  3  1} { 5  2  1} { 5  4  2} { 6  4  1} { 6  5  2} { 7  3  1} { 7  3  2} { 7  5  2}
{ 7  6  1} { 7  6  3} { 7  6  5} { 8  3  2} { 8  4  2} { 8  5  3} { 8  6  1} { 8  7  3}
{ 8  7  6} { 9  4  3} { 9  5  2} { 9  6  1} { 9  7  3} { 9  8  2} {10  3  1} {10  4  1}
{10  5  2} {10  5  3} {10  6  4} {10  6  5} {10  7  1} {10  7  6} {10  8  3} {10  8  5}
{10  9  1} {10  9  2} {10  9  4} {11  2  1} {11  4  3} {11  5  1} {11  5  4} {11  6  2}
{11  7  2} {11  8  3} {11  8  4} {11  8  6} {11  8  7} {11  9  1} {11  9  5} {11  9  8}
{11 10  5} {11 10  6} {12  2  1} {12  4  2} {12  4  3} {12  6  3} {12  7  4} {12  7  5}
{12  7  6} {12  8  2} {12  9  3} {12  9  7} {12 10  3} {12 10  6} {12 10  9} {12 11  1}
{12 11  2} {12 11  8}
66 sequences found

Four-term 14-bit word length
{ 5  3  1} { 5  4  3} { 6  4  1} { 7  5  3} { 8  3  2} { 8  4  1} { 8  6  1} { 9  3  2}
{ 9  6  5} { 9  7  2} { 9  8  3} { 9  8  5} {10  3  1} {10  6  1} {11  4  1} {11  4  3}
{11  6  1} {11  6  5} {11  7  1} {11  9  7} {11 10  3} {11 10  9} {12  2  1} {12  5  2}
{12  7  5} {12  9  2} {12 10  1} {12 11  1} {12 11  5} {12 11  6} {13  3  2} {13  4  2}
{13  7  3} {13  8  3} {13  8  4} {13  8  6} {13 10  3} {13 10  6} {13 10  8} {13 11  4}
{13 11  9} {13 12  2}
42 sequences found
```

```
Four-term 15-bit word length
{ 4  2  1} { 5  3  2} { 5  4  2} { 7  2  1} { 7  4  1} { 7  5  2} { 7  6  1} { 8  4  2}
{ 8  5  2} { 8  6  5} { 8  7  2} { 8  7  5} { 9  4  1} { 9  6  2} { 9  7  3} {10  4  3}
{10  5  1} {10  5  4} {10  7  1} {10  7  5} {10  8  5} {10  8  7} {10  9  3} {10  9  4}
{10  9  7} {11  6  3} {11  6  5} {11  7  1} {11  8  2} {11  8  3} {11 10  5} {12  3  1}
{12  4  3} {12  5  2} {12  6  1} {12  6  5} {12  7  3} {12  7  4} {12  8  2} {12  8  3}
{12  8  6} {12  9  1} {12  9  4} {12 11  3} {12 11  5} {13  3  1} {13  5  1} {13  5  2}
{13  6  1} {13  7  3} {13  7  4} {13  8  1} {13  8  7} {13  9  6} {13 10  1} {13 10  2}
{13 10  3} {13 10  7} {13 10  8} {13 11  7} {13 11 10} {13 12 10} {14  2  1} {14  4  1}
{14  5  2} {14  6  3} {14  7  2} {14  8  4} {14  8  5} {14  9  2} {14  9  3} {14  9  8}
{14 10  2} {14 10  5} {14 11  1} {14 11  6} {14 11  8} {14 12  2} {14 12  3} {14 13  1}
{14 13  8} {14 13 11}
82 sequences found

Four-term 16-bit word length
{ 5  3  2} { 5  4  3} { 6  4  1} { 8  7  5} { 9  4  2} { 9  4  3} { 9  5  2} { 9  7  2}
{ 9  7  4} { 9  7  5} {10  5  3} {10  7  1} {10  7  3} {10  7  4} {10  7  6} {10  9  6}
{11  3  2} {11  6  5} {11  9  7} {11  9  8} {11 10  5} {12  3  1} {12  6  1} {12  7  1}
{12  7  2} {12  9  6} {12  9  7} {12 10  3} {13  6  4} {13  8  2} {13  9  6} {13 11  6}
{13 12  7} {13 12 11} {14  8  3} {14  9  1} {14  9  4} {14  9  7} {14 11  7} {14 12  1}
{14 12  7} {14 13  5} {14 13 11} {15  4  1} {15  4  2} {15  7  2} {15  9  4} {15  9  6}
{15 10  4} {15 12  1} {15 12 10} {15 13  4}
52 sequences found

Four-term 17-bit word length
{ 3  2  1} { 5  3  2} { 5  4  1} { 6  4  2} { 6  5  3} { 7  3  2} { 7  4  3} { 7  5  1}
{ 7  6  2} { 8  3  1} { 8  3  2} { 8  4  3} { 8  5  2} { 8  6  5} { 8  7  1} { 8  7  6}
{ 9  3  2} { 9  5  1} { 9  5  3} { 9  5  4} { 9  6  2} { 9  7  1} { 9  7  4} { 9  8  3}
{ 9  8  4} {10  2  1} {10  4  2} {10  4  3} {10  6  1} {10  6  5} {10  7  2} {10  7  6}
{10  9  2} {10  9  5} {11  3  1} {11  3  2} {11  6  4} {11  7  4} {11  8  2} {11  8  6}
{11  9  1} {11  9  6} {11 10  7} {11 10  9} {12  3  2} {12  4  1} {12  4  2} {12  5  3}
{12  5  4} {12  6  2} {12  6  3} {12  7  1} {12  8  1} {12  8  2} {12  8  4} {12  8  7}
{12  9  2} {12 11  4} {12 11  7} {12 11  9} {13  3  1} {13  5  1} {13  5  4} {13  6  5}
{13  7  3} {13  8  1} {13  8  3} {13  8  4} {13  9  4} {13  9  5} {13  9  8} {13 10  3}
{13 10  6} {13 10  8} {13 11  6} {13 12  1} {13 12  2} {13 12  4} {13 12  5} {13 12  8}
{14  5  3} {14  7  3} {14  7  4} {14  9  1} {14  9  4} {14  9  8} {14 10  3} {14 10  4}
{14 11  5} {14 12  1} {14 12  3} {14 12  5} {14 12  8} {14 12 11} {14 13  7} {14 13  9}
{14 13 10} {15  4  1} {15  5  2} {15  5  4} {15  6  2} {15  7  2} {15  8  5} {15  8  7}
{15  9  5} {15  9  6} {15 10  2} {15 10  7} {15 11  2} {15 11  5} {15 11  8} {15 11 10}
{15 12  2} {15 12  9} {15 13  5} {15 13  7} {15 13 11} {15 14  1} {15 14  5} {15 14  6}
{15 14  8} {15 14  9} {15 14 10} {15 14 12} {16  3  1} {16  3  2} {16  5  1} {16  5  3}
{16  5  4} {16  6  1} {16  8  3} {16  8  6} {16  9  4} {16  9  5} {16 10  5} {16 10  8}
{16 10  9} {16 11  1} {16 11  7} {16 12  1} {16 12  4} {16 12  8} {16 12 10} {16 13  2}
{16 13  5} {16 13 12} {16 14  1} {16 14  4} {16 14  6} {16 14  9} {16 15  7} {16 15 14}
152 sequences found

Four-term 18-bit word length
{ 5  2  1} { 6  3  2} { 8  2  1} { 8  7  4} { 9  3  1} { 9  4  1} { 9  6  5} {10  4  3}
{10  5  4} {10  7  3} {10  7  5} {10  8  5} {10  9  3} {11  7  2} {11  8  2} {12  5  2}
{12  5  3} {12  7  3} {12  9  5} {12 10  1} {13  2  1} {13  5  4} {13  6  4} {13  7  5}
{13  9  3} {13  9  6} {13 10  2} {13 10  8} {13 11  5} {13 11  8} {13 12  9} {14  5  1}
{14  5  3} {14  7  2} {14  9  3} {14 11 10} {14 12  5} {14 13  2} {14 13  5} {14 13  8}
```

```
{15  5  2} {15  9  2} {15  9  4} {15  9  5} {15  9  8} {15 11  2} {15 11  6} {15 11  8}
{15 13  4} {15 13  6} {15 14  8} {16  5  4} {16  7  3} {16  8  1} {16  8  5} {16  9  3}
{16 10  7} {16 11  4} {16 11  7} {16 13  3} {16 13  6} {16 15 12} {17  5  1} {17  8  6}
{17 10  2} {17 13  1} {17 13  4} {17 14  9} {17 15  9} {17 16  5} {17 16 10} {17 16 13}
72 sequences found

Four-term 19-bit word length
{ 5  2  1} { 6  2  1} { 6  4  1} { 6  4  3} { 6  5  1} { 7  4  1} { 7  6  5} { 8  6  1}
{ 8  6  5} { 8  7  2} { 8  7  5} { 9  3  1} { 9  4  1} { 9  4  3} { 9  5  2} { 9  5  4}
{ 9  6  4} { 9  7  1} { 9  7  2} { 9  7  5} { 9  8  5} {10  6  4} {10  7  3} {10  7  4}
{10  8  6} {10  9  3} {10  9  4} {10  9  6} {11  4  2} {11  6  1} {11  6  2} {11  6  4}
{11  8  6} {11  9  4} {12 10  3} {12 10  4} {12 10  5} {12 11  1} {13  3  2} {13  4  3}
{13  5  3} {13  8  2} {13  9  1} {13  9  2} {13  9  3} {13  9  4} {13 10  1} {13 10  9}
{13 11  3} {13 11  8} {13 11  9} {13 12  3} {13 12  5} {14  2  1} {14  3  1} {14  5  3}
{14  6  2} {14  7  3} {14  7  5} {14  7  6} {14  9  7} {14 10  3} {14 11  3} {14 11 10}
{14 12  1} {14 12  5} {14 12 10} {14 12 11} {14 13  2} {14 13 11} {14 13 12} {15  5  4}
{15  7  2} {15  9  3} {15  9  7} {15 10  1} {15 10  6} {15 10  8} {15 10  9} {15 12  9}
{15 13  3} {15 13  8} {15 13  9} {15 13 10} {15 14  2} {15 14  4} {15 14 10} {16  4  1}
{16  5  2} {16  6  1} {16  6  3} {16  6  4} {16  7  1} {16  7  6} {16  8  5} {16  8  6}
{16  9  2} {16  9  5} {16  9  7} {16 10  4} {16 10  6} {16 10  9} {16 12  1} {16 12  2}
{16 12  5} {16 12  9} {16 13  3} {16 14  5} {16 14  6} {16 15  6} {16 15 10} {16 15 13}
{17  5  4} {17  6  5} {17  7  3} {17  9  1} {17 10  3} {17 10  6} {17 11  1} {17 11  6}
{17 12  4} {17 12 10} {17 12 11} {17 13  5} {17 13  8} {17 14  1} {17 14  3} {17 14 10}
{17 15  8} {17 16  6} {18  5  1} {18  5  2} {18  6  1} {18  7  3} {18  7  5} {18  8  2}
{18  8  7} {18  9  4} {18  9  6} {18 10  2} {18 10  6} {18 12  3} {18 12 10} {18 13  1}
{18 13  3} {18 13  8} {18 13 11} {18 14  1} {18 14 13} {18 15  3} {18 15 10} {18 15 12}
{18 15 13} {18 16  5} {18 16 10} {18 17  5} {18 17 13} {18 17 14}
158 sequences found

Four-term 20-bit word length
{ 6  4  1} { 6  5  2} { 6  5  3} { 9  5  1} { 9  5  3} { 9  5  4} {10  5  1} {10  7  2}
{10  7  6} {10  8  3} {10  9  2} {11  4  3} {11  5  3} {11  6  1} {11  7  4} {11  9  3}
{11 10  4} {12  6  3} {12  6  5} {12  8  1} {12  8  5} {12  9  2} {12  9  8} {12 11  1}
{12 11  3} {12 11  4} {12 11  7} {12 11  8} {13  7  2} {13  8  4} {13  8  7} {13  9  1}
{13  9  5} {13  9  8} {13 10  5} {13 11  3} {13 12  5} {13 12  7} {14 13  1} {14 13 10}
{15  5  4} {15  7  4} {15  8  7} {15  9  2} {15 10  7} {15 11  7} {15 12  8} {15 14  8}
{16  5  1} {16  6  1} {16  7  3} {16  9  8} {16 10  9} {16 11  3} {16 12  1} {16 12  7}
{16 13  5} {16 13  9} {16 14  3} {16 15  5} {16 15 11} {17  6  3} {17  6  4} {17  9  4}
{17  9  7} {17  9  8} {17 11  2} {17 11  9} {17 12 10} {17 13  4} {17 14  3} {17 14  8}
{17 15  9} {17 15 11} {17 15 14} {17 16  1} {17 16  9} {18  4  1} {18  9  3} {18 11  1}
{18 11  5} {18 11  8} {18 11 10} {18 13  7} {18 13 10} {18 15 14} {19  4  3} {19  7  6}
{19  8  4} {19  9  2} {19  9  8} {19 11  7} {19 12  8} {19 14  4} {19 14  9} {19 15  4}
{19 15 10} {19 15 11} {19 16  2} {19 16 14}
100 sequences found

Four-term 21-bit word length
{ 5  2  1} { 6  5  2} { 7  4  1} { 8  2  1} { 8  6  3} { 8  7  2} { 9  4  3} { 9  5  2}
{ 9  8  3} {10  4  2} {10  4  3} {10  5  1} {10  6  3} {10  7  4} {10  9  4} {11  3  2}
{11  4  1} {11  4  2} {11  6  5} {11  7  1} {11  7  4} {11  8  7} {11 10  5} {12  6  2}
{12  7  6} {12  8  3} {12 10  1} {12 10  3} {12 11  5} {12 11  8} {13  5  2} {13  7  1}
{13  7  3} {13  7  5} {13 10  2} {13 10  3} {13 10  9} {13 12  6} {14  2  1} {14  4  1}
{14  5  3} {14  7  2} {14  8  1} {14  8  6} {14  9  5} {14 11  3} {14 11  4} {14 13  5}
```

```
{14 13 10} {15  3  1} {15  5  3} {15  6  4} {15  7  4} {15  8  2} {15  9  2} {15  9  8}
{15 10  3} {15 10  6} {15 11  6} {15 12  2} {15 13  7} {15 14  2} {15 14  9} {16  2  1}
{16  5  2} {16  7  4} {16  8  7} {16 10  4} {16 10  9} {16 11  2} {16 11 10} {16 12  7}
{16 13  2} {16 14  1} {16 14  2} {16 14  4} {16 14  8} {16 15  1} {16 15  3} {16 15 10}
{17  2  1} {17  4  2} {17  6  2} {17  7  4} {17  7  5} {17  8  3} {17 10  2} {17 10  7}
{17 11  5} {17 12 11} {17 14  4} {17 14  5} {17 14  6} {17 14 10} {17 14 11} {17 15  6}
{18  4  1} {18  6  5} {18  7  3} {18 10  7} {18 11  6} {18 11  8} {18 11  9} {18 12  1}
{18 13  1} {18 13  4} {18 13  9} {18 13 12} {18 14  3} {18 14  8} {18 15 11} {18 15 13}
{18 16  6} {18 16  7} {18 17 11} {18 17 12} {19  2  1} {19  4  1} {19  5  1} {19  7  5}
{19  7  6} {19  8  5} {19  9  6} {19 10  5} {19 11  4} {19 11  8} {19 12  6} {19 13  6}
{19 14  7} {19 14 13} {19 15  4} {19 15  9} {19 16  1} {19 16  5} {19 16  8} {19 16 12}
{19 16 15} {19 17  4} {19 17 10} {19 17 11} {19 18 10} {20  5  2} {20  6  5} {20  7  5}
{20  8  3} {20  9  3} {20 11  9} {20 13  7} {20 14  8} {20 14 10} {20 16  2} {20 16 11}
{20 17  2} {20 17  3} {20 17  7} {20 17 10} {20 17 14} {20 18  6} {20 19  2} {20 19  4}
{20 19  5} {20 19  7} {20 19 13} {20 19 16}
164 sequences found

Four-term 22-bit word length
{ 5  4  3} { 7  6  1} { 8  5  3} { 8  6  5} { 9  5  1} { 9  8  2} { 9  8  4} { 9  8  7}
{10  4  1} {10  7  3} {11  2  1} {11  5  3} {11  7  6} {11  8  1} {12  3  1} {12  7  3}
{12  8  7} {12  9  4} {12  9  6} {12 10  7} {12 11  2} {13  5  3} {13  9  2} {13 10  1}
{13 10  3} {13 10  7} {13 10  9} {13 11  2} {13 11  5} {13 11  6} {13 12  1} {13 12  9}
{14  3  2} {14  5  1} {14  8  7} {14  9  2} {14  9  5} {14  9  7} {15  2  1} {15  7  4}
{15  9  7} {15 10  7} {15 12  7} {15 12  9} {15 12 10} {15 13  7} {15 13  8} {15 14  8}
{15 14 10} {15 14 13} {16  6  3} {16  7  1} {16  9  2} {16 11  9} {16 12  1} {16 12  5}
{16 13 10} {16 15 11} {17  7  3} {17  7  5} {17  8  3} {17  9  5} {17 10  6} {17 11  9}
{17 12  1} {17 13  5} {17 13  8} {17 15  5} {17 16  3} {17 16 14} {18  6  3} {18 11  2}
{18 13 10} {18 14 13} {18 15  7} {18 17  1} {19  3  2} {19  6  2} {19  6  5} {19 11  2}
{19 12  9} {19 14  5} {19 15  5} {19 15 10} {19 15 12} {19 16  4} {19 16  6} {19 17  9}
{19 17 11} {19 17 14} {19 18 17} {20  3  1} {20 11  3} {20 11  4} {20 11  9} {20 11 10}
{20 13  6} {20 13  8} {20 13  9} {20 14 13} {20 15  1} {20 16  3} {20 19  3} {20 19  8}
{21  5  4} {21  7  2} {21  9  1} {21 10  5} {21 10  6} {21 10  9} {21 12  9} {21 13  1}
{21 14 11} {21 15  6} {21 16 15} {21 17  8} {21 17 13} {21 18 12} {21 19  2} {21 19 10}
{21 20  7} {21 20 11}
122 sequences found

Four-term 23-bit word length
{ 5  3  1} { 5  3  2} { 5  4  1} { 6  3  2} { 6  5  2} { 7  2  1} { 7  3  1} { 7  4  3}
{ 7  5  1} { 7  6  2} { 8  6  5} { 8  7  1} { 8  7  3} { 9  5  2} { 9  6  3} { 9  7  4}
{ 9  8  1} { 9  8  6} {10  3  2} {10  5  3} {10  6  4} {10  7  6} {10  8  6} {10  9  4}
{10  9  7} {11  7  3} {11  7  6} {11  8  3} {11  8  5} {11  9  1} {11 10  9} {12  5  1}
{12  5  2} {12  5  4} {12  6  2} {12  6  3} {12  8  3} {12  8  7} {12  9  2} {12 10  2}
{12 11  3} {12 11  5} {12 11  6} {13  4  3} {13  6  1} {13  7  5} {13  8  4} {13  9  1}
{13  9  5} {13  9  8} {13 10  9} {13 11  1} {13 11  3} {13 12  1} {13 12  8} {14  6  4}
{14  6  5} {14  7  4} {14  8  2} {14  8  5} {14  9  2} {14  9  3} {14  9  5} {14  9  6}
{14  9  8} {14 10  6} {14 12  2} {14 12  5} {14 12  8} {14 13  3} {14 13 10} {14 13 12}
{15  2  1} {15  4  1} {15  5  1} {15  6  3} {15  7  1} {15  8  4} {15  8  5} {15  8  7}
{15 11  1} {15 11  3} {15 11  4} {15 11  5} {15 11  8} {15 11  9} {15 11 10} {15 12  8}
{15 13  2} {15 14  5} {15 14  9} {15 14 10} {16  4  3} {16  6  5} {16  7  3} {16  7  5}
{16  7  6} {16  9  1} {16  9  5} {16 11  4} {16 12  2} {16 13  6} {16 14 13} {16 15  1}
{16 15  4} {16 15  8} {16 15 11} {17  3  2} {17  5  3} {17  6  1} {17  7  1} {17  7  6}
{17  8  3} {17  9  4} {17 10  1} {17 10  4} {17 10  7} {17 11  2} {17 11  3} {17 11  5}
```

```
{17 12 11} {17 13  9} {17 14  3} {17 14  9} {17 15  3} {17 15  4} {17 15 13} {17 15 14}
{17 16  6} {17 16  7} {17 16 12} {17 16 13} {18  5  1} {18  8  3} {18  9  2} {18  9  8}
{18 10  5} {18 11  9} {18 12  6} {18 12  8} {18 12 11} {18 13  1} {18 13  5} {18 14  2}
{18 14  3} {18 14  7} {18 14  9} {18 14 10} {18 15  8} {18 15  9} {18 15 12} {18 16  2}
{18 16  7} {18 16 10} {18 17  1} {18 17  7} {18 17  9} {18 17 15} {19  3  1} {19  5  1}
{19  8  6} {19  8  7} {19 10  2} {19 11  1} {19 11  3} {19 12  7} {19 12  8} {19 13  2}
{19 13  6} {19 14  6} {19 14 13} {19 15  1} {19 15  8} {19 15 10} {19 16  3} {19 16  9}
{19 16 14} {19 17  9} {19 17 13} {19 18  1} {19 18 11} {20  3  1} {20  4  1} {20  4  3}
{20  5  3} {20  6  2} {20  7  4} {20  8  3} {20  8  6} {20  9  5} {20  9  6} {20 10  9}
{20 11  3} {20 12  3} {20 12  4} {20 12  6} {20 12  8} {20 12 10} {20 12 11} {20 14  2}
{20 14  9} {20 15  2} {20 15  3} {20 15  5} {20 15  6} {20 15 11} {20 15 12} {20 16  7}
{20 16 12} {20 16 15} {20 17  2} {20 17  8} {20 17 11} {20 17 14} {20 18  3} {20 18  6}
{20 18 13} {20 19  3} {20 19  7} {20 19 10} {20 19 16} {21  2  1} {21  5  2} {21  6  3}
{21  7  5} {21  8  3} {21  9  3} {21  9  5} {21 10  4} {21 10  8} {21 11  2} {21 11  7}
{21 11  9} {21 12  2} {21 12  6} {21 13  4} {21 13 11} {21 14  5} {21 14  9} {21 14 11}
{21 15  9} {21 17  3} {21 17 11} {21 17 16} {21 18  2} {21 18 11} {21 18 14} {21 18 17}
{21 19  1} {21 20  6} {21 20 13} {21 20 17} {21 20 18} {22  2  1} {22  4  2} {22  5  4}
{22  6  5} {22  8  1} {22  8  4} {22  8  7} {22 10  5} {22 11 10} {22 12  4} {22 12  8}
{22 12 10} {22 13  6} {22 14  7} {22 14 10} {22 14 12} {22 15  1} {22 15 14} {22 16  6}
{22 16  8} {22 16 15} {22 17  6} {22 17 10} {22 18  4} {22 18  5} {22 18  8} {22 18 11}
{22 18 16} {22 19  3} {22 19  8} {22 19 18} {22 20  3} {22 20  4} {22 20 16} {22 20 18}
{22 21  1} {22 21  2} {22 21  8} {22 21 16}
292 sequences found

Four-term 24-bit word length
{ 4  3  1} { 7  2  1} { 7  5  4} { 8  5  2} { 9  5  2} { 9  6  4} {10  4  3} {10  6  1}
{10  6  3} {11  5  2} {11  6  2} {11  8  1} {11  9  6} {11  9  8} {11 10  2} {12  3  2}
{13  3  1} {13 10  5} {13 11  5} {14 13  7} {15  3  2} {15  8  6} {15 13  4} {16  3  1}
{16  5  2} {16  7  2} {16  9  7} {16 14  1} {16 15 13} {17  2  1} {17  4  2} {17  8  2}
{17 11 10} {17 14  6} {17 15  8} {17 16  2} {18  5  2} {18 10  7} {18 15  5} {18 15 13}
{18 16  9} {18 17  2} {19  5  4} {19  6  3} {19  9  6} {19 13  3} {19 13  4} {19 13 11}
{19 14 11} {20  9  2} {20 11  5} {20 11  9} {20 18 15} {20 19  1} {20 19  5} {20 19 17}
{21  9  2} {21 11  5} {21 15  1} {21 15  2} {21 18  5} {21 18 14} {21 20 14} {22  5  1}
{22  7  6} {22  8  7} {22  9  3} {22 14 13} {22 15  3} {22 15  4} {22 16  7} {22 17  8}
{22 18 13} {22 19  6} {22 19  8} {22 19 13} {22 19 15} {22 19 16} {22 20  7} {22 21  9}
{22 21 12} {23  5  4} {23  9  3} {23 10  1} {23 10  8} {23 14  1} {23 16 13} {23 18 14}
{23 19  2} {23 21  8} {23 21 11} {23 21 20} {23 22  7} {23 22 17}
94 sequences found
```

9 GENERAL APPENDIXES

9.1 Appendix 7: q Format for DSP Chips

9.1.1 Scientific and Binary-Radix Notation

These notations are of the form

$$\text{mantissa} \times \text{radix}^{\text{exponent}}$$

Scientific notation has radix 10, while binary-radix notation has radix 2. The exponent is conventionally an integer.

The scientific notation of floating-point constants as in, for example, 7e3, is mathematically equivalent to 7×10^3. Binary-radix notation [70] is similar. For example, 7q3 is defined as mathematically equivalent to 7×2^3 (where q stands for *quanta*). This notation comes in handy when we want to specify the location of the binary point in a fixed-point number which is to be assigned as the contents of some register, that is, to specify the register format.

Fig. 70 shows the two most commonly used formats of a 24-bit register used for holding a coefficient in a digital filtering application. In Fig. 70(a) the range of the coefficient[141] is $[-1., 1.)$ for q23, whereas in Fig. 70(b) it is $[-2., 2.)$ for q22. Here q23 is the maximum binary-radix locator in 24-bit two's complement fixed point.

One way to look at binary-radix notation, then, is as a

[141] The notation [ ) means that the positive extreme cannot be reached exactly.

conversion from floating-point representation, *floating.expression*, to fixed-point representation. For example,

SOME_REGISTER = *floating.expression* q22

This declares the binary point in the 24-bit register to be fixed at 2 bit away from the MSB (or 22 bit away from the LSB). This in turn implies that *floating.expression* is known to have a magnitude less than 2.

Generally speaking, any numerical expression followed with binary-radix notation will be multiplied by 2 raised to the indicated power. But we should instead think of binary-radix notation as a mnemonic tool that provides the *location* (the q format) of the binary point within some fixed-point register, as in the examples of Fig. 70.

9.1.2 Fixed-Point Arithmetic within a DSP Chip

Next we turn to the subject of numerical computation within the executing hardware numerical function units. The rules of fixed-point arithmetic are easy when the q format is employed.

Rule 1—Addition and Subtraction: To add or subtract two fixed-point numbers, their respective binary points must have the same location, that is, the same q.

If they do not have the same q, then a shifter must be used to first bring them into alignment.

Rule 2—Multiplication: In two's complement, when multiplying a 24-bit number having qN with another 24-bit number having qM, the product is a 48-bit number having q$(N + M + 1)$ format. The 24 MSBs of the product have q$(N + M + 1 - 24)$ format.

The extra 1 in rule 2 arises from the common practice of designing the signed multiplier such that there is a built-in permanent shift-left 1-bit of the product to remove the extra sign bit [70], [1], [71], [49].[142] Most often only the MSBs of an accumulation of products constitute some desired result. So, for example, suppose we multiply a q0 signal at 24 bit (typically 16 bit of signal left-justified into a 24-bit word) by a q23 coefficient at 24 bit. The result is a 48-bit product in q24. Now, if we truncate the least significant 24 bit storing only the MSBs, we end up with a 24-bit result in q0. In this case, the q of the result is the same as that of the signal.

[142] The LSB of the 48-bit product should be 0 as a result of the permanent shift left. Typical DSP chips have hardware multiply units that are *not* designed to multiply integers, that is, they are designed to execute fixed-point multiplys.

9.1.3 Numerical Precision

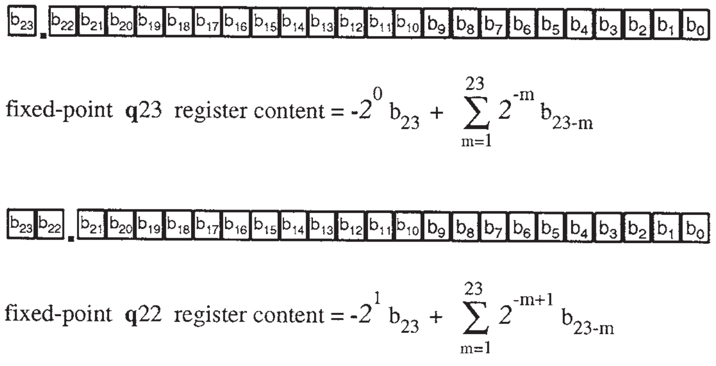
If the programmer is scrupulous, it is not too difficult to implement block floating-point arithmetic. Block floating-point arithmetic is a numerical implementation of an algorithm using a fixed-point DSP chip where the binary-radix point of a block (some collection or group) of operands is fixed, but fixed only over some intermediate portion of a longer computation. A different block (or the same block computed at a different phase of the program) may have a different binary point location. The block floating-point technique finds use such as applied to fast Fourier transform (FFT) [73], [74] or amplitude compression algorithms implemented on fixed-point processors having barrel shifters. When the fundamental word length of a fixed-point processor is large enough (at least 24 bit), the need for block floating point diminishes and fixed-point computations may suffice. An alternative to block floating point is double-precision arithmetic [12].

9.2 Appendix 8: Truncation Mathematics for DSP Chips

9.2.1 Math Function Definitions: Magnitude Truncation, Rounding, and Truncation

```
int result;
double value;
/* magnitude truncation */
result = floor(fabs(value)) * SIGN(value);
/* rounding */
result = floor(fabs(value) + 0.5) * SIGN(value);
/* truncation */
result = floor(value);
```

Shown are the C-program floating-point definitions of the three truncation functions. The truncation functions (magnitude truncation, rounding, truncation) operate on floating-point values typically greater than $|1.0|$, producing integer results. Magnitude truncation and rounding are


b23 . b22 b21 b20 b19 b18 b17 b16 b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
fixed-point q23 register content = $-2^{0}\, b_{23} + \sum_{m=1}^{23} 2^{-m}\, b_{23-m}$
b23 b22 . b21 b20 b19 b18 b17 b16 b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
fixed-point q22 register content = $-2^{1}\, b_{23} + \sum_{m=1}^{23} 2^{-m+1}\, b_{23-m}$
Fig. 70. Two's complement fixed-point format examples.

symmetrical functions. On many machines it is helpful to keep in mind that the (int) or (long) C-cast of a floating-point number actually performs a magnitude truncation. The SIGN( ) macro is defined to return {1., 0., −1.} for strictly positive, zero, and negative arguments, respectively. Recall that the floor(x) function, defining simple truncation, returns the largest integer not greater than x; truncation simply throws away or masks off the fractional part.

9.2.2 Real-Time Compute Statistics

It would be interesting to analyze the statistical impact of each signal-truncation function type when used in real-time computation, such as in a DSP chip program executing some digital filter algorithm. For that application, the meaning of the truncation functions pertains to the handling of what is considered to be the fractional part (the LSBs to the right of the radix point) of some binary word. A specific example might be the conversion from a double-precision (48-bit) result to single precision (24 bit). The most notable outcome is that the use of magnitude truncation introduces noise into the resultant signal, which is statistically 6 dB higher in power than that produced by either rounding or truncation. Although rounding and truncation produce equivalent (to each other) ac noise power, truncation, which introduces a negative dc offset of one-half quantum, is not a zero-mean process. These statistical results can be explained by considering the distribution of the magnitude and sign of the quantization errors due to each truncation function [11, ch. 11.2].

Alternately, we may represent the action of each truncation function in terms of the instantaneous change to some two's complement number, as illustrated in Fig. 71, assuming a nonzero fractional part. None of the truncation functions can change a positive number into a negative number, and vice versa. Iconic Fig. 71 indicates that truncation will increase the magnitude of negative numbers only, whereas rounding can increase the magnitude of all numbers. Further, truncation *always* increases the magnitude of negative numbers, but rounding does not always increase magnitude (explaining the bidirectional arrows). On the other hand, magnitude truncation *always* decreases the magnitude of both positive and negative numbers.

The truncation function simply called "truncation" has no point of symmetry; it causes the same direction of change everywhere in the continuum. This lack of symmetry accounts for the statistical dc offset. Of the three truncation functions, both rounding and truncation have the propensity for producing results that exceed the magnitude of their arguments. This idiosyncrasy explains one of the causes of limit-cycle tones produced in direct form and lattice digital filter topologies [11, ch. 11.5]. Magnitude truncation does not share this characteristic and can sometimes remedy limit-cycle oscillation [9]. (Check out Section 1.3.4.)

9.2.3 Real-Time Compute Implementation

We wish to know how the truncation functions are each implemented in the binary domain. For the sake of illustration, let us assume that we are given 24-bit two's complement binary numbers in q8 format defined in hexadecimal ($) as follows:

```
val1 = $007FFF   // q8 format means $007F.FF = 127.99609375

val2 = $FF8001   // q8 format means $FF80.01 = −127.99609375
```

1) *Binary Magnitude Truncation*: In the binary domain, the corresponding operation to magnitude truncation is,

```
Pseudocode: if(val < 0) val + = $0.FFF . . . ;     /* binary magnitude truncation */
            val = binary truncation(val);          /* discard bits to right of binary point */
```

For the two values, after binary magnitude truncation, val1 = $007F00 and val2 = $FF8100. This is how magnitude truncation might be programmed in a real-time computation within a DSP chip. In decimal, we are adding 0.999 . . . to val when it is negative. We conditionally add $0.FFF . . . to val, rather than $1.0, to avoid bumping up a perfect negative integer. This implies that if *any* of the fractional bits of val are nonzero when it is negative, then magnitude truncation will increase val.

But as it often happens, the given 24-bit binary number may itself constitute the MSBs of a higher precision 48-bit result. It is likely that the 24 LSBs of the higher precision

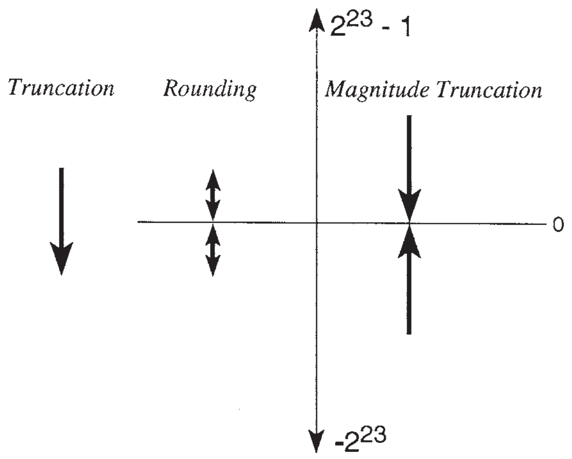


Fig. 71. Direction and relative magnitude of change after quantization of 24-bit two's complement number.

result were nonzero and the situation is such that we are not sure. Statistically it is much better to err on the side of caution if we do not know what those higher precision LSBs were. So in this circumstance, we would amend the pseudocode to conditionally add \$1.0 to val instead of \$0.FFF. . . . Empirically, we have observed much better outcome under this latter assumption.

2) *Binary Rounding*: Curiously, in the case of two's complement binary rounding there is no *conditional* addition. In the binary domain, the corresponding operation to rounding is,

```
Pseudocode: val + = $0.8;                  /* binary rounding */
            val = binary truncation(val);  /* discard bits to right of binary point */
```

For the two given values, after binary rounding, val1 = \$008000 and val2 = \$FF8000. This is how rounding might be programmed in a real-time computation within a DSP chip. In decimal, we are unconditionally adding 0.5 to val, regardless of its sign.[143]

3) *Binary Truncation*: In the binary domain the corresponding operation to truncation is simply,

```
Pseudocode: val = binary truncation(val) ;   /* binary truncation */
                                             /* discard bits to right of binary point */
```

For the two given values, after binary truncation, val1 = \$007F00 and val2 = \$FF8000; the 8 LSBs are masked off, or simply discarded.

One useful fact regarding truncation is that the fractional part (rather, the part that is discarded or masked off) is always positive in sign. This fact could be used to advantage within a digital filtering circuit having truncation error feedback for the purpose of minimizing truncation noise [12].[144]

Fig. 72 shows an example of truncating any 24-bit q8 two's complement number by taking the 16 MSBs and discarding the 8 LSBs. We are interested in the sign of the error e[n] that results from subtracting the truncated number $\hat{y}[n]$ from the full-precision number $y[n]$, that is,

$$y[n] - \hat{y}[n] = \mathrm{e}[n] \geq 0 .$$

Since the b_i can only take on the value 0 or 1, in two's complement, the stated result follows regardless of the sign of $y[n]$. By induction, this result extends to other q, hence other truncation widths.

[143] When comparing rounding results in the floating-point domain with the corresponding results in the binary domain, it is wise *not* to use test values having a fractional part = 0.5 exactly, because this special case produces different results in each domain.

[144] This noise, due to ongoing internal signal quantization error, is also known as roundoff noise [11]. Truncation error feedback is also known to minimize limit-cycle oscillation [18], thus providing an alternative to magnitude truncation as a remedy.

10 ACKNOWLEDGMENT

The author wishes to express his gratitude to Jennie Columba and Antonio for unflagging encouragement and support in the pursuit of knowledge.

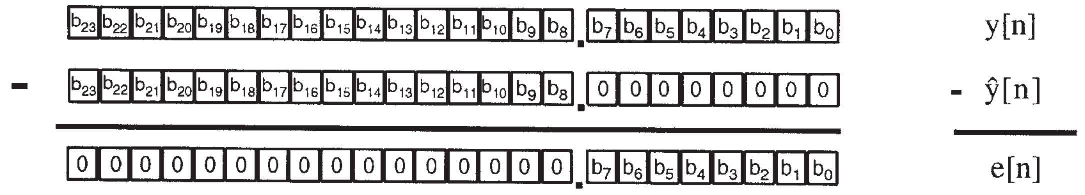
11 REFERENCES

[1] D. C. Andreas, J. Dattorro, and J. W. Mauchly, "Digital Signal Processor for Audio Applications," U.S. patent 5,517,436 (1996 May 14).

[2] W. G. Gardner, "Reverberation Algorithms," in *Applications of Digital Signal Processing to Audio and Acoustics*," M. Kahrs and K. Brandenburg, Eds. (Kluwer Academic, Norwell, MA, 1998).

[3] J. A. Moorer, "About this Reverberation Business," *Computer Music J.*, vol. 3, pp. 13–28 (1979 June), also in *Foundations of Computer Music*, C. Roads and J. Strawn, Eds. (MIT Press, Cambridge, MA, 1988).

[4] D. Griesinger, "Practical Processors and Programs for Digital Reverberation," in *Audio in Digital Times, Proc. Audio Eng. Soc. 7th Int. Conf.* (Toronto, Ont., Canada, 1989 May 14–17), pp. 187–195.



$$\left(-2^{15}\, b_{23} + \sum_{m=1}^{23} 2^{-m+15}\, b_{23-m}\right) - \left(-2^{15}\, b_{23} + \sum_{m=1}^{15} 2^{-m+15}\, b_{23-m}\right) = \sum_{m=16}^{23} 2^{-m+15}\, b_{23-m} \geq 0$$

Fig. 72. Example showing how truncation error is always positive.

[5] B. A. Blesser and K. O. Bader, "Electric Reverberation Apparatus," U.S. patent 4,181,820 (1980 Jan. 1).

[6] D. Griesinger, "Room Impression, Reverberance, and Warmth in Rooms and Halls," presented at the 93rd Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts)*, vol. 40, p. 1064 (1992 Dec.), preprint 3383.

[7] M. Schroeder, "Natural Sounding Artificial Reverberation," *J. Audio Eng. Soc.*, vol. 10, p. 219 (1962 July).

[8] J. M. Jot and A. Chaigne, "Digital Delay Networks for Designing Artificial Reverberators," presented at the 90th Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts)*, vol. 39, p. 383 (1991 May), preprint 3030.

[9] J. O. Smith, "Elimination of Limit Cycles and Overflow Oscillations in Time-Varying Lattice and Ladder Digital Filters," in *Music Applications of Digital Waveguides*, Rep. STAN-M-39, Center for Computer Research in Music and Acoustics (CCRMA), Dept. of Music, Stanford University, Stanford, CA (1987 May).

[10] A. V. Oppenheim, Ed., *Applications of Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1978).

[11] L. B. Jackson, *Digital Filters and Signal Processing*, 3rd ed. (Kluwer Academic, Norwell, MA, 1996).

[12] J. Dattorro, "The Implementation of Recursive Digital Filters for High-Fidelity Audio," *J. Audio Eng. Soc.*, vol. 36, pp. 851–878 (1988 Nov.); Comments, ibid. (*Letters to the Editor*), vol. 37, p. 486 (1989 June); Comments, ibid. (*Letters to the Editor*), vol. 38, pp. 149–151 (1990 Mar.).

[13] P. A. Regalia and S. K. Mitra, "Tunable Digital Frequency Response Equalization Filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, pp. 118–120 (1987 Jan.).

[14] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[15] J. Strawn, Ed., *Digital Audio Signal Processing* (A-R Editions, Madison, WI, 1985).

[16] J. A. Moorer, "The Manifold Joys of Conformal Mapping: Applications to Digital Filtering in the Studio," *J. Audio Eng. Soc.*, vol. 31, pp. 826–841 (1983 Nov.).

[17] D. P. Rossum, "Dynamic Digital IIR Audio Filter and Method which Provides Dynamic Digital Filtering for Audio Signals," U.S. patent 5,170,369 (1992 Dec.).

[18] T. I. Laakso, "Error Feedback for Reduction of Quantization Errors due to Arithmetic Operations in Recursive Digital Filters," D. Tech. Thesis, Rep. 9, Laboratory of Signal Processing and Computer Technology, Helsinki University of Technology, Espoo, Finland (1991).

[19] K. Steiglitz, *A Digital Signal Processing Primer* (Addison-Wesley, Menlo Park, CA, 1996).

[20] *Moog Voltage-Controlled Ladder Filter*, Internet: http://dropmix.xs4all.nl/rick/Emusic/Moog (Rick Jansen) and Internet: http://www-ccrma.stanford.edu/~stilti/papers/moogvcf.pdf

[21] D. Rossum, "Making Digital Filters Sound 'Analog,'" in *Proc. Int. Computer Music Conf.* (San Jose, CA, 1992), pp. 30–33.

[22] "CEM3328 Four Pole Low Pass VCF," Curtis Electromusic Specialties, Los Gatos, CA (1983).

[23] H. Chamberlin, *Musical Applications of Microprocessors* (Hayden, Indianapolis, IN, 1980).

[24] *Mathematica*, version 2, Wolfram Research, Champaign, IL (1994).

[25] B. L. Evans, L. J. Karam, K. A. West, and J. H. McClellan, "Learning Signals and Systems with *Mathematica*," *IEEE Trans. Education*, vol. 36, pp. 72–78 (1993 Feb.).

[26] K. W. Martin and M. T. Sun, "Adaptive Filters Suitable for Real-Time Spectral Analysis," *IEEE Trans. Circuits Sys.*, vol. CAS-33, pp. 218–229 (1986 Feb.); also *IEEE J. Solid-State Circuits*, vol. SC-21, no. 1 (1986 Feb.), pp. 108–119.

[27] M. R. Petraglia, S. K. Mitra, and J. Szczupak, "Adaptive Sinusoid Detection Using IIR Notch Filters and Multirate Techniques," *IEEE Trans. Circuits Sys. II*, vol. 41, pp. 709–717 (1994 Nov.).

[28] T. Kwan and K. Martin, "Adaptive Detection and Enhancement of Multiple Sinusoids Using a Cascade IIR Filter," *IEEE Trans. Circuits Sys.*, vol. 36, pp. 937–947 (1989 July).

[29] P. P. Vaidyanathan, *Multirate Systems and Filter Banks* (Prentice-Hall, Englewood Cliffs, NJ, 1993).

[30] T. I. Laakso, P. S. R. Diniz, I. Hartimo, and T. C. Macdeo, Jr., "Elimination of Zero-Input and Constant-Input Limit Cycles in Single-Quantizer Recursive Filter Structures," *IEEE Trans. Circuits Sys. II*, vol. 39, pp. 638–646 (1992 Sept.).

[31] A. Luthra, "Extension of Parseval's Relation to Nonuniform Sampling," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36 (1988 Dec.), pp. 1909–1911.

[32] R. W. Schafer and L. R. Rabiner, "A Digital Signal Processing Approach to Interpolation," *Proc. IEEE*, vol. 61, pp. 692–702 (1973 June).

[33] F. R. Moore, "Table Lookup Noise for Sinusoidal Digital Oscillators," *Computer Music J.*, vol. 1, pp. 26– 29 (1977 Apr.), also in *Elements of Computer Music* (Prentice-Hall, Englewood Cliffs, NJ, 1990), chap. 3.

[34] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1983).

[35] M. Renfors and T. Saramäki, "Recursive *N*th-Band Digital Filters — Parts I and II," *IEEE Trans. Circuits Sys.*, vol. CAS-34, pp. 24–51 (1987 Jan.).

[36] D. Rossum, "An Analysis of Pitch Shifting Algorithms," presented at the 87th Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts)*, vol. 37, p. 1072 (1989 Dec.), preprint 2843.

[37] R. Adams and T. Kwan, "Theory and VLSI Architectures for Asynchronous Sample-Rate Converters," *J. Audio Eng. Soc.*, vol. 41, pp. 539–555 (1993 July/Aug.).

[38] T. A. Ramstad, "Digital Methods for Conversion between Arbitrary Sampling Frequencies," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, pp. 577–591 (1984 June).

[39] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, “Splitting the Unit Delay — Tools for Fractional Delay Filter Design,” *IEEE Signal Process. Mag.*, vol. 13, pp. 30–60 (1996 Jan.).

[40] J. Dattorro, “The Implementation of Digital Filters for High Fidelity Audio, Part II — FIR,” in *Audio in Digital Times, Proc. Audio Engineering Society 7th Int. Conf.* (Toronto, ON, Canada, 1989 May 14–17), pp. 168–180.

[41] J. C. Dattorro, A. J. Charpentier, and D. C. Andreas, “Decimation Filter as for a Sigma–Delta Analog-to-Digital Converter,” U.S. patent 5,027,306 (1991 June 25).

[42] D. C. Andreas, “VLSI Implementation of a One-Stage 64:1 FIR Decimator,” presented at the 89th Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts)*, vol. 38, p. 872 (1990 Nov.), preprint 2976.

[43] J. O. Smith and B. Friedlander, “Adaptive, Interpolated Time-Delay Estimation,” *IEEE Trans. Aerospace Electron. Sys.*, vol. 21, pp. 180–199 (1985 Mar.).

[44] J. O. Smith III, “Techniques for Digital Filter Design and System Identification with Application to the Violin,” Rep. STAN-M-14, Center for Computer Research in Music and Acoustics (CCRMA), Dept. of Music, Stanford University, Stanford, CA (1983 June).

[45] V. Välimäki, T. I. Laakso, and J. Mackenzie, “Elimination of Transients in Time-Varying Allpass Fractional Delay Filters with Application to Digital Waveguide Modeling,” in *Proc. Int. Computer Music Conf.* (Banff, AB, Canada, 1995), pp. 327–334.

[46] J. O. Smith, “An Allpass Approach to Digital Phasing and Flanging,” Rep. STAN-M-21, Center for Computer Research in Music and Acoustics (CCRMA), Dept. of Music, Stanford University, Stanford, CA (1982 Spring).

[47] W. M. Hartmann, “Flanging and Phasers,” *J. Audio Eng. Soc. (Engineering Reports)*, vol. 26, pp. 439–443 (1978 June).

[48] M. L. Beigel, “A Digital ‘Phase Shifter’ for Musical Applications, Using the Bell Labs (Alles–Fischer) Digital Filter Module,” *J. Audio Eng. Soc. (Engineering Reports)*, vol. 27, pp. 673–676 (1979 Sept.).

[49] *DSP56000/DSP56001 Digital Signal Processor User’s Manual*, rev. 2 (Motorola, DSP Division, Austin, TX, 1990).

[50] F. F. Lee, “Time Compression and Expansion of Speech by the Sampling Method,” *J. Audio Eng. Soc.*, vol. 20, pp. 738–742 (1972 Nov.); also in *Speech Enhancement*, J. S. Lim, Ed. (Prentice-Hall, Englewood Cliffs, NJ, 1983), pp. 286–290.

[51] J. Dattorro, “Using Digital Signal Processor Chips in a Stereo Audio Time Compressor/Expander,” presented at the 83rd Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts)*, vol. 35, p. 1062 (1987 Dec.), preprint 2500.

[52] A. I. Abu-el-Haija and M. M. Al-Ibrahim, “Improving Performance of Digital Sinusoidal Oscillators by Means of Error Feedback Circuits,” *IEEE Trans. Circuits Sys.*, vol. CAS-33, pp. 373–380 (1986 Apr.).

[53] J. W. Gordon and J. O. Smith, “A Sine Generation Algorithm for VLSI Applications,” in *Proc. Int. Computer Music Conf.* (1985), pp. 165–168.

[54] R. W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed. (Dover, Mineola, NY, 1973).

[55] R. M. Gray, *Source Coding Theory* (Kluwer, Norwell, MA, 1990).

[56] R. M. Gray and J. W. Goodman, *Fourier Transforms — An Introduction for Engineers* (Kluwer, Norwell, MA, 1995).

[57] J. O. Smith and P. R. Cook, “The Second-Order Digital Waveguide Oscillator,” in *Proc. Int. Computer Music Conf.* (1992), pp. 150–153.

[58] N. J. Fliege and J. Wintermantel, “Complex Digital Oscillators and FSK Modulators,” *IEEE Trans. Signal Process.*, vol. 40, pp. 333–342 (1992 Feb.).

[59] B. K. Thoen, “Practical Aspects of Digital Sinewave Generation Using a Second-Order Difference Equation,” *IEEE Trans. Circuits Sys.*, vol. CAS-32, pp. 510–511 (1985 May).

[60] S. W. Golomb, Ed., *Digital Communications with Space Applications* (Peninsula Publishing, Los Altos, CA, 1964).

[61] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C*, 2nd ed. (Cambridge University Press, New York, 1992).

[62] R. M. Gray and L. D. Davisson, *An Introduction to Statistical Signal Processing* (Stanford University, Stanford, CA, 1997) Internet: http://www-isl.stanford.edu/~gray/sp.html

[63] J. Borish and J. B. Angell, “An Efficient Algorithm for Measuring the Impulse Response Using Pseudorandom Noise,” *J. Audio Eng. Soc.*, vol. 31, pp. 478–488 (1983 July/Aug.).

[64] D. B. Keele, Jr., “The Design and Use of a Simple Pseudo Random Pink-Noise Generator,” *J. Audio Eng. Soc.*, vol. 21, pp. 33–41 (1973 Jan./Feb.).

[65] F. J. MacWilliams and N. J. A. Sloane, “Pseudo-Random Sequences and Arrays,” *Proc. IEEE*, vol. 64, pp. 1715–1731 (1976 Dec.).

[66] G. R. Cooper and C. D. McGillem, *Probabilistic Methods of Signal and System Analysis*, 2nd ed. (Oxford University Press, Cary, NC, 1986).

[67] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, 2nd ed. (Addison-Wesley, Menlo Park, CA, 1994).

[68] R. N. Bracewell, *The Fourier Transform and Its Applications*, 2nd ed., rev. (McGraw-Hill, New York, 1986).

[69] N. J. Kasdin, “Discrete Simulation of Colored Noise and Stochastic Processes and $1/f^{\alpha}$ Power Law Noise generation,” *Proc. IEEE*, vol. 83, pp. 800–827 (1995 May).

[70] *TMS32010 User’s Guide*, Doc.SPRU001B, Texas Instruments, Digital Signal Processor Products Division, Stafford, TX (1985).

[71] *ADSP-2100 Family User’s Manual*, 3rd ed., Analog Devices, Computer Products Division, Norwood,

MA (1995).

[72] J. G. Proakis, *Digital Communications*, 2nd ed. (McGraw-Hill, New York, 1989).

[73] *ADSP-2100 Family Applications Handbook*, 3rd ed., vol. 1, Analog Devices, Digital Signal Processing Division, Norwood, MA (1989).

[74] S. Kim and W. Sung, "A Floating-Point to Fixed-Point Assembly Program Translator for the TMS 320C25," *IEEE Trans. Circuits Sys. II*, vol. 41, pp. 730–739 (1994 Nov.).

THE AUTHOR

Jon Dattorro is from Providence, RI. He trained as a classical pianist, attended the New England Conservatory of Music where he studied composition and electronic music, and performed as soloist with Myron Romanul and the Boston Symphony Orchestra for Children's Concerts at Symphony Hall. His scores include a ballet and a piano concerto.

Mr. Dattorro received a B.S.E.E. with highest distinction from the University of Rhode Island in 1981, where he was a student of Leland B. Jackson. In 1984 he received an M.S.E.E. from Purdue University, specializing in digital signal processing under S. C. Bass. He is currently working towards a Ph.D. in electrical engineering at Stanford University.

He designed the Lexicon Inc. model 2400 Time Compressor with Charles Bagnaschi and Francis F. Lee in 1986, and he designed most of the audio effects from Ensoniq Corp. between 1987 and 1995. He shares two patents in digital signal processing chip design with David C. Andreas, J. William Mauchly, and Albert J. Charpentier. Personal mentors are Salvatore J. Fransosi, Pozzi Escot, and Chae T. Goh.