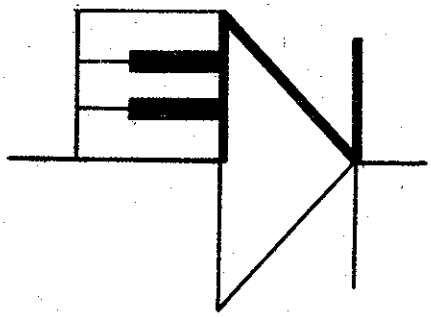


ELECTRONOTES 204

Newsletter of the Musical Engineering Group

1016 Hanshaw Rd. Ithaca, NY 14850

Volume 21, No. 204 August 2004

GROUP ANNOUNCEMENTS

Contents of EN#204

READER'S QUESTION

One of the many pleasures of teaching occurs when a former student remembers you and gets back in touch to ask a question. In this case, it was a student I had more than 20 years ago. Such questions often tend to be somewhat apologetic (I should have been paying more attention to what you told us!). But in all candor, it must be admitted that instructors, too, learn more as time passes, and 20 year old explanations probably weren't all that good to begin with. I hope I understand it better now. Anyway, I will try.

Q:

I thought I had a pretty good understanding of sampling theory, but recently I was trying to explain (to people who do not have a good technical background) how a sinewave could be recovered from just three samples (more than two) within a cycle. I guess now I am not sure!

A:

Actually I think it is a good sign that you find this difficult to understand. It's not simple to answer until the exact question is posed and qualified. In the simplest terms, as most students of DSP learn, we have the "sampling theorem" that says that if you take samples at a rate greater than twice the highest frequency in the signal, you can recover the signal from its samples. In this sense, your case of a sinewave sampled three times per cycle seems to qualify just fine. But now we have to ask you more questions. For example, do you have at least three samples for each and every cycle, or are there just three samples total? Did someone tell you it was a sinewave that you are trying to recover? All these things make a difference.

In order to get to the correct question, which is probably actually something more like "How does a CD player work?" we need to go down some other roads first. Let's begin by supposing that we have three and only three samples total, and someone told us that these belong to a sinusoidal waveform (additional information beyond the three samples). Can we get the sinusoidal waveform back? Yes we can. Suppose we are given the three samples:

$$\begin{aligned} x(0) &= 0 \\ x(1) &= 1/2 \\ x(2) &= \sqrt{3}/2 \end{aligned} \qquad (1)$$

In this case, we probably only need to look at these samples for a few seconds before we notice that they are the samples of a sinewave of unit amplitude, taken at 0°, at 30°, and at 60°. This was easy, and the corresponding sinewave is plotted along with the three given samples, in Fig. 1a. Note that this sinewave, like all sinusoidal waveforms, is assumed to extend to infinity in both directions (we plot just three full cycles in Fig. 1a, and only show the three samples we were given). In Fig. 1b, we show 37 samples corresponding to three cycles plus one extra sample. Note that there are 12 samples per cycle here, and they are dense enough that we do not find it too difficult to suppose that these samples, taken by themselves, do in fact represent a sine wave. We show the actual sinewave with light dotting so the reader can better ignore it and concentrate on the samples themselves. The frequency here is 1/12 of the sampling frequency.

As a first modification of our problem, assume that now we are required to use three samples, and that they must be three samples that correspond to exactly one cycle. This means that we chose a frequency that is exactly 1/3 of the sampling frequency. The choice would be as in Fig. 2a with

$$\begin{aligned} x(0) &= 0 \\ x(1) &= \sqrt{3}/2 \\ x(2) &= -\sqrt{3}/2 \end{aligned} \qquad (2)$$

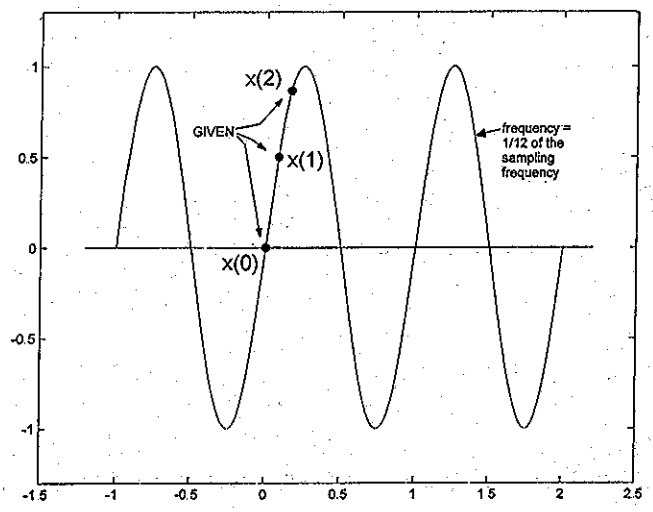


Fig. 1a Three samples, 0, 1/2, $\sqrt{3}/2$, correspond to a sinewave of frequency 1/12.

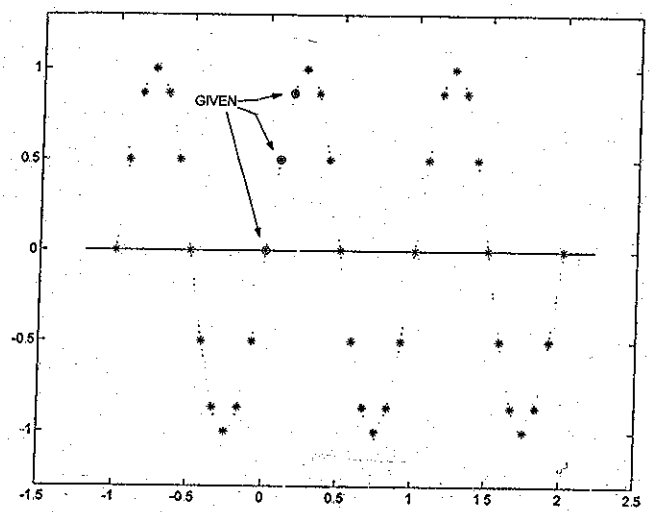


Fig. 1b Three full cycles of a sinewave of frequency 1/12, 12 samples per cycle.

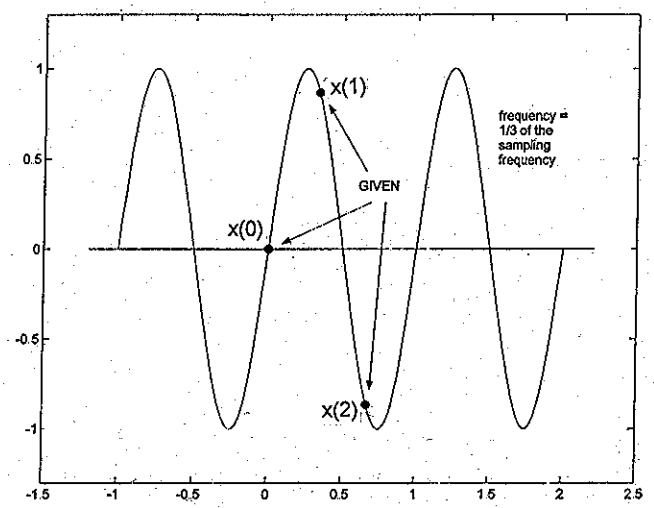


Fig. 2a Here we have three samples that represent exactly one cycle.

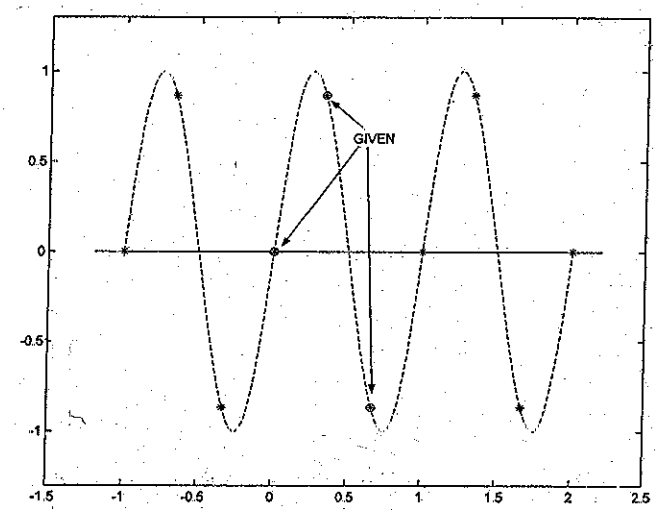


Fig. 2b With only three samples per cycle, it is less obvious that we can recover the sinewave.

We see that the three samples do correspond to a sine wave (again, we show more samples in Fig. 2b). While we note that, according to the sampling theorem, we have enough samples to recover the sinewave, visually we are less convinced of recovery here, relative to Fig. 1b. Often it is this problem as suggested in Fig. 2b (How can we possibly get a sinewave back from that mess?) that we are asked to explain.

[Now before we get carried away, sampling theory says that there are actually more (an infinite number) of other sine waves that will fit these points. Fig. 3 shows another sinewave that has a frequency exactly four times larger than the first one. Of course, this is an aliasing situation - there are fewer than two samples per cycle (fewer than one actually). We remind ourselves of aliasing just in case we are being given a trick question about uniqueness.]

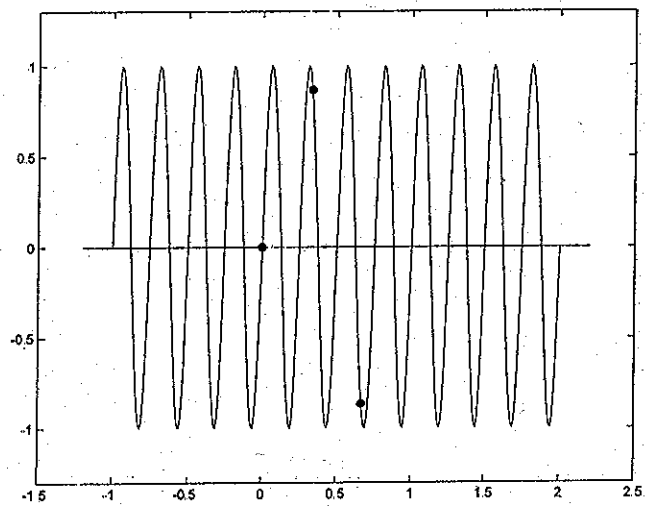


Fig. 3 Here a frequency of 4/3 also fits the given points – an example (of an infinite number of cases) of aliasing.

Less we get carried away even further, recall that we found the sinusoidal waveform that went through the given points by inspection (by guessing). Since we can't expect to always guess, how do we solve this problem? For example, the points

$$\begin{aligned} y(0) &= 0.1736 \\ y(1) &= 0.9063 \\ y(2) &= -0.6428 \end{aligned} \tag{3}$$

are samples of a sinusoidal waveform, and it is real tough to guess the parameters. We would expect the sinusoidal to be represented by an expression:

$$y(t) = A \sin(2\pi f t + \varphi) \tag{4}$$

where we need to find the amplitude A, the frequency f, and the phase φ. We might suppose that we can do this since we have three parameters to determine, and we have three sample values given. This is correct. But can we just set up and solve the equations - plugging t=0, 1, and 2 into equation (4) with the values of equation (3)? Try it. The equations are non-linear, and you can't solve them directly. [On very rare occasions, you can solve them. The samples of equations (1) and (2) are such a case. The sample x(0)=0 is important, and use the trig identity for $\sin(2\theta)$.]

To actually solve the general case, we need to use what is called "Prony's method." This method is over 200 years old (!) and is beyond the scope of this question, so the interested reader is directed to a reference from this newsletter [1]. The answer is:

$$y(t) = \sin(2\pi(105/360)t + 2\pi(10/360)) \tag{5}$$

continued on pg. 34

Basic Elements of Digital Signal Processing

Finite Wordlength Effects – Part 1

-by Bernie Hutchins

1. INTRODUCTION

Taken as a whole, the topics related to the effects of finite wordlength comprise what are perhaps the most boring and unattractive chapters in the DSP literature. (Are these not just the complications of reality - the grist of the practitioner - not in the realm of high level engineering ideas?) The topics taken individually are, however, not only of great practical importance, but extremely varied and of considerable depth. It is easy to become almost entranced by some of them when you get a close enough look.

In a discrete time system, time is quantized. This time-quantization we call "sampling," and virtually everything we normally discuss under the topic headings of Digital Signal Processing (DSP) depends on this sampling. Strictly speaking however, if something is "digital" it really ought to be discrete in amplitude as well as in time. This means that the values of samples and of multiplying coefficients are not of infinite precision, but rather must take on only certain discrete values. In the case of signal values, this is often termed quantization (we often speak of "quantization noise") while for coefficients, we are likely to speak of "coefficient roundoff." While we tend to group all such finite wordlength effects together, there are many different manifestations in practical cases with numerous interesting consequences.

We can begin by noting that there is one immense difference between the quantization of signal values and the rounding of coefficient multipliers. In both cases, we are talking about replacing some presumed infinite precision number with one that must be selected from a set of values that are actually available. For example, we might suppose a number should be 19.7659243... but we must settle for an integer, and might prefer to choose 20, although 19 might be the choice if we had to settle for truncation. (In most real cases, we are talking about a binary representation of course, but the presentation here, using decimals, is sufficiently similar and easier to follow.)

Now, if the numbers being quantized are signals, the quantization results in an equivalent error in the signal values, and this can sometimes, but by no means always, be characterized as a random noise. Note that this representation is non-linear. This is

perhaps clear when we consider that the transfer curve due to quantization is of a staircase shape instead of a straight ramp. (We may even argue that the non-linearity is very mild when the step size is small, but it is still non-linear.) More to the point perhaps, consider the addition of two numbers, 17.6 and 12.7. Adding, we would get 30.3, which would round to 30. If we round first however, these would become 18 and 13, which add to 31. Thus signal quantization fails the most basic linearity test.

But what happens when we quantize coefficient multipliers? For example, an FIR filter might have 4 taps that should be nominally 3.3, 55.4321, -12.111, and 19.001. If we had to choose only integers by using rounding, we would choose 3, 55, -12, and 19. (With truncation, we might have 3, 55, -13, and 19.) Note that we can characterize this result of quantization as one of having the "wrong filter." We see however that the error is not the same for all coefficients. Indeed, it is very small for the case of 19.001 being rounded to 19, but much larger (as an absolute number, and relative to the coefficient value) in the case of 3.3 rounding to 3. But suppose that someone came along and asked for coefficients of 3, 55, -12, and 19. We would state with pride that we could offer them exactly the filter they wanted. It would be Linear Time-Invariant (LTI) and its frequency response would be error free. Thus we can see that a filter with rounded coefficients is usually the "wrong filter" to some degree, but it is still LTI and is a "perfectly defined wrong filter." In production, we get perfect copies always (contrast this with an analog filter which would depend statistically on component tolerances - no two being exactly the same).

Our first fundamental distinction is thus that signal quantization can result in some form and some quantity of "noise" and/or "distortion" while coefficient roundoff results in frequency responses that are not exactly (and in some cases, unacceptably different from) what we had in mind. We can now move on to a second introduction - here termed an overview.

2. AN OVERVIEW OF FINITE WORDLENGTH PROBLEMS

2a. Quantization Noise

In the introduction, we mentioned that the quantization of signal samples can result in an equivalent noise. This we shall eventually characterize and quantify. Clearly, if we have more bits rather than fewer, we intuitively and correctly expect less noise. In an audio signal, this noise sounds like "white noise" - similar to a hissing noise of a leaking tire. (Also, this is often compared to the background hiss of audio tape, or of the interstation noise of an FM radio - but both these examples are less familiar with fewer and fewer tape devices, and with "muting" FM receivers.)

Actually, the noise modeling of signal quantization error may well not be valid for small signals, and/or for simple signals (one, or only a few frequency components). In such cases, a harmonic distortion model may be much more useful. Surprisingly, one solution to such problems is to add noise - analog noise or so-called "dither" which effectively allows encoding of levels smaller than the least significant bit. Surprisingly, properly employed dither actually greatly reduces the non-linearity of a quantizer, often with a relatively small price to pay in the form of an elevated noise floor.

All this is interesting enough, but it is also possible to accept the inevitability of the quantization noise, and to then shape it so that it is effectively inaudible. This "noise shaping" becomes possible through the use of "oversampling." In essence, the noise is pushed into the frequency range above human hearing - this range having been opened up by the oversampling. This permits such things as bragging about a 1-bit D/A converter on the one hand, or the ability to claim 24 bits in a 16 bit format on the other (to the bewilderment of marketing personnel).

2b. Coefficient Roundoff - and Scaling

In the introduction, we also discussed coefficient roundoff - the well-defined wrong filter. It likely occurred to the reader that while we were talking about coefficients that were relatively close to integers (i.e., the numbers were generally much greater than 1 in magnitude), that many filters have, and must have, coefficients that are small. For example, the feedback coefficient branch of an IIR filer might have coefficients of 6.77, -2.12, 1.04, -0.62, 0.28, -0.09, and 0.015. We would not expect good results if we chose rounded integer coefficients of 7, -2, 1, -1, 0, 0, and 0. We would expect terrible results - for one thing, three of the feedbacks are lost, so three poles of the filter would be lost. We would also consider ourselves lucky if the filter were even stable.

Clearly, we have two options. If we had a floating-point arithmetic available, we could fairly easily implement non-integer numbers quite close to nominal, and this we would expect to be quite useful. However, often being stuck with fixed point arithmetic, how would we get at a coefficient like, for example, 0.2209? Well - as a ratio of integers. We could investigate various integers n_1 and n_2 such that n_1/n_2 is sufficiently close to 0.2209. However, division is not a forte of DSP chips, and we might well want to restrict our values of n_2 to powers of 2 so that division is accomplished with a shift. For example, suppose we chose n_2 to be 1024. This would make n_1 equal to 226 (0.2209 x 1024, rounded). The actual coefficient implemented would then be 226/1024 = 0.2207, an error of about 0.1%. Surprisingly, while this particular example is sometimes plenty good (e.g., for FIR filters) it is not at all hard to find cases (high-order IIR filters) where this is far too much error.

Additional numerical problems are related. We often find ourselves adding up (accumulating) many results of multiplies by a range of coefficients, and would like to do the division by a single post-addition shift. This complicates (restricts) our choice of n_2.

In addition, with fixed-point we have to choose coefficients so that signal levels do not get too small (a permanent loss of actual resolution from that point on) or too large (saturation or overflows to consider). This is the so-called "scaling problem" - keeping the signal level just right at all points within a filter structure.

2c. Overflow Questions

When signals get too small, we expect signal quantization effects to become more pronounced. Why not always keep signals very large? Well, with fixed point, and to a much lesser degree with floating point, there are limits to the absolute value of numbers. For example, with 16-bits, we could only represent integers to about ±32,000. What happens if you exceed these limits by calling for a larger number?

Most DSP chips offer the option of "saturating" or "overflowing." In saturation mode, if you ask for a number greater than the maximum, you get the maximum. This is what happens in an analog system when a signal "clips" against the power supply levels. In general, this is a harmonic distortion generating situation. In overflow mode, things seem even worse. Instead of a (desired) positive number greater than the most positive one available, you get numbers on the negative side, which are much more serious errors. This is in fact worse than saturation, if we are talking about the final result. But, what if the overflow is the result of an as yet uncompleted accumulation of partial products? If we have overflowed in the positive direction, the next product accumulated might be negative, and the overflow error is reversed, exactly (back from never-never land). We hope to end up the accumulation in the original range. Clearly this rescue would never be possible following any case of saturation. Is it realistic to expect such a fortunate circumstance? Not always, but in many cases we can expect and control this occurrence. For example, a series of filter coefficients may alternate in sign while the signal is low-frequency.

2d. Floating Point to the Rescue?

Many things are easier to do with floating point arithmetic. For example, with floating point, overall scaling (through a cascade of digital filter sections) is likely just a matter of one correcting multiply at the very end. With fixed point, we generally need to optimize each section as we go. (Yet such floating point processors may not run as fast as fixed point.) Can anything go wrong with floating point? Of course, and one such thing is SCIMD (Scale-Change InterModulation Distortion). Keep in mind that floating point is not infinite precision. In fact, with floating point, what we really have is a fixed-point mantissa and a fixed-point exponent combined. We gain dynamic range. We can represent much smaller and much larger numbers. But still, while we can get much, much closer to the number zero than we could with fixed point, we don't expect to get closer, to the same accuracy, to larger numbers. That is, while we might be able to represent $3x10^{-25}$, we can't expect to represent $4x10^{21}+3x10^{-25}$ (which would come out $4x10^{21}$).

In the same way that we have lost $3x10^{-25}$ in the example above, in the case of a signal consisting of small and large components, a small component may be lost. Yet the situation is even worse than that, because a large component is not large all the time. A large amplitude sinewave is large near its peaks but small near its zero-crossings. Accordingly, we may have the situation where a large component turns the small one off and on (modulates it). This happens when the sum gets so large that the exponent gets larger and one or more places are lost in the mantissa. Because we assume no specific relationship between the frequencies of the components, the modulation products will in general be in positions where there were no components in the original signal. Thus we have an inter-modulation distortion caused by a change of scale.

2e. Limit Cycles - Small and Large

It is our usual expectation that if we have a filter and if at a certain instance it has a measurable input and a measurable output, that if we subsequently arrange for the input to become and remain zero, that the output will show a continuing trend toward zero as well. By no means do we expect the output to go immediately to zero - the filter needs to work the numbers through, around, and out. Put another way, the filter of course has a transient response, and the event of a zero input is just another transient. But, if the filter is stable, the "energy" cycling inside should at least decay, and further, since a digital filter is subject to signal quantization, we expect decay to reach levels which would actually round to zero (even where the theoretical response only decays exponentially, approaching but never actually reaching zero). This is what we expect for a LTI system. But, quantization is non-linear. The individual steps may be small, but nonetheless they are steps. Even worse, in overflow mode, some severe non-linearities can result. Both of these can result in a filter output, with a zeroed input, ending up not at zero, but in a limit-cycle attractor. By tradition, the limit cycles due to the quantization steps are usually called just "limit cycles" or "small-scale limit cycles" while the large-scale limit cycles are termed "overflow oscillations."

A small-scale limit cycle occurs when a signal at the output is decaying but then gets stuck. This is largely the result of a rounding or truncation effectively increasing the value of a filter coefficient (you multiply x by g and get gx which rounds to a value y > gx which is the same as increasing g). In very rare cases, we can think in terms of an equivalent LTI system. But generally, it is a residual low-level "buzz." It need not always occur. With truncation, it may occur with decay from a state of one polarity but not from the other. These are not always even worth worrying about.

Overflow oscillations on the other hand are always a problem, but more interesting. They are large and unpredictable oscillations, often chaotic and can even demonstrate fractal properties. For the exact same filter, one instance of overflow may stick the filter in a length three overflow sequence. Removing this, we might see a length 147 sequence

the next time it overflows. These may be triggered simply by having the signal level overflow somewhere inside the filter, at which point, getting rid of it is not usually a matter of reducing the input level, even all the way to zero. Sudden, large changes of input amplitude may kick you back out, but you may well need to stop the filter and reset its internal points to zero, or set to saturation mode rather than overflow mode.

2f Noise Gain

We have suggested a "noise model" for the quantization of large (large relative to the quantization interval) complex (many frequency components) signals. Below we will calculate the equivalent noise. There are two cases where we expect this quantization noise to be generated: of course when we quantize an analog signal, but also when we round off to the original number of bits following a multiply. Accordingly, we need to know what level of quantization noise we should expect at the output of a system as a result of multiple quantizations (often with different quantization levels). Do noise sources get amplified? They can - and perhaps by a lot. How do we add different quantization noise contributions? We add up noise powers.

3 SPECIFIC TOPICS IN FINITE WORLDLENGTH

3a Noise Model (QUANTIZATION NOISE POWER)

Recognizing that we are dealing with round-off errors that are not exactly known, we must resort to some model that may offer insight and guidance during design. Here we will look at a standard noise model that applies to large, complex signals. In the next subsection, we will use a harmonic distortion model that offers a better description when we are dealing with small, simple signals. We will then see that additive "dither" is an approach that offers some unexpected relief from harmonic distortion, and indeed, from the quantization non-linearity in general. Eventually we will arrive at useful ideas involving oversampling and noise shaping. We begin with the usual noise model - calculating the noise power associated with a quantization interval Q.

In Fig. 1 we show particular instances of quantization where a signal is sampled and then quantized to levels separated by interval Q. It is convenient to think of the level Q as being a voltage, and we will associate the actual quantization procedure with rounding to the closest available level. We assume that the input signal is large and complex. By large, we can only mean that the amplitude of the signal is >> Q (no other level is available to serve as a reference to small or large). By "complex" we mean that the signal has many frequency components and likely random components (such as a speech or music signal). The point is that the sequence of quantization errors is to be considered random, and uncorrelated with the signal. The errors are expected to be just about

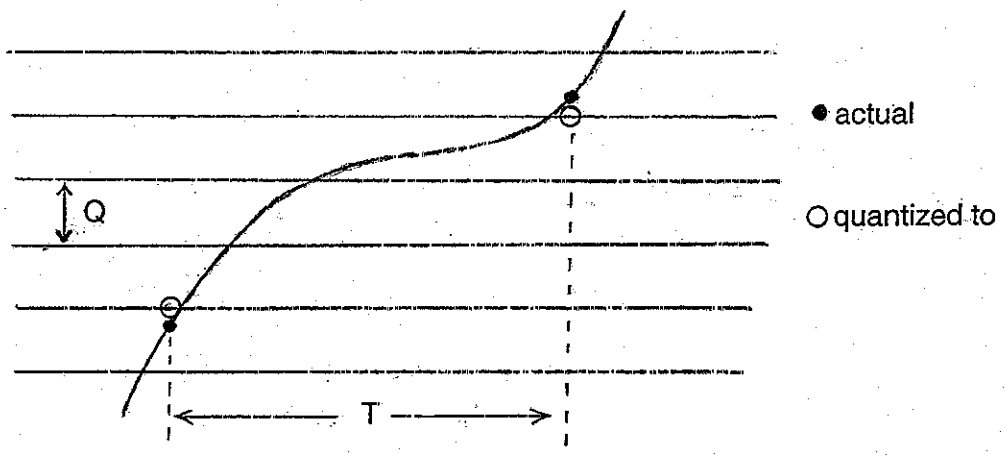


Fig. 1 A large "complex" signal is likely to have errors that may be considered random

anywhere between levels with equal probability. In such an instance, the distribution of errors is flat on the interval -Q/2 to +Q/2 and is normalized to 1 by setting the probability to 1/Q (Fig. 2). The quantization noise power (perhaps most usually called the variance) is obtained by integrating the square of the distribution

$$\sigma_Q^2 = (1/Q) \int_{-Q/2}^{Q/2} x^2 \, dx = Q^2/12 \qquad (1)$$

This is a fundamental and well-known result [1]. The quantization error has an inherent power of $Q^2/12$. Note that the result depends only on Q. Well - almost.

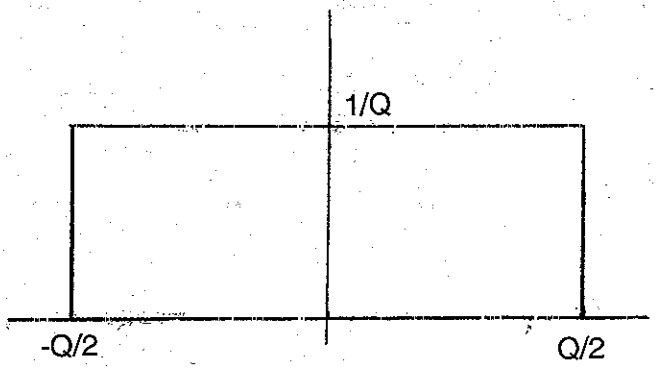


Fig. 2 Uniform probability distribution of the error

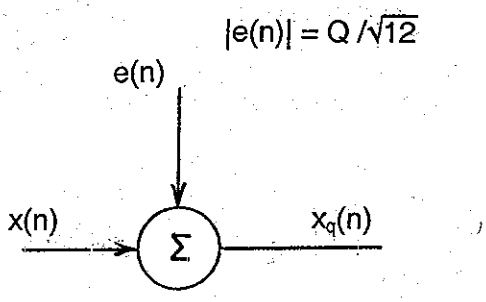


Fig. 3 Noise model of quantization

The result apparently does not depend on the amplitude of the signal, the number of bits (number of levels), or the sampling rate. Actually, we should only say that it does not depend on the sampling rate. While we do not see a factor representing the amplitude, or the number of bits as variables in our equation, we should not forget that we have made some assumptions. The amplitude must be very large relative to Q, and this in turn requires that there be a lot of levels. Thus we need to assume that there are a lot of bits, perhaps 12, 16, or 24 as opposed to only 3, 2, or 1. And the signal is making use of these bits - the more significant bits are not being neglected. Meeting all our assumptions, we can work with a noise power of $Q^2/12$. We thus think of (i.e., model) quantization as being the addition of random noise (Fig. 3).

Having decided that quantization results in a noise amplitude of $Q/\sqrt{12}$, we can calculate a signal-to-noise (S/N) ratio assuming the maximum possible signal level. If we have B bits total available, and one bit is required for sign, we have 2^{B-1} levels separated by interval Q that are available: a maximum amplitude $2^{B-1}Q$. An RMS value of this amplitude would be $2^{B-1}Q/\sqrt{2}$, so the S/N ratio would be:

$$S/N = \frac{2^{B-1}Q/\sqrt{2}}{Q/\sqrt{12}} = \sqrt{1.5}\ 2^B \qquad (2)$$

in decibels, this would be:

$$20 \log_{10}(S/N) = 1.76 + 6.02B \quad db \qquad (3)$$

We thus get the fundamental result (reasonable approximation?) that each bit added improves the S/N ratio by 6 db, and roughly, we expect a S/N ratio of 6•B db. We immediately understand the 6db/bit result as being a consequence of the 2:1 reduction in error for each new bit added.

3b Harmonic Distortion Model

In contrast to large complex signals, we now want to look at small simple signals. By "small" we mean a signal amplitude that is comparable to Q, and for "simple" we can consider a single frequency sinusoidal waveform. Fig. 4a shows a simple sinewave of frequency $0.025f_s$ (f_s is sampling frequency) of amplitude Q/2 (peak-to-peak amplitude Q), centered at Q/2, sampled 40 times/cycle (samples not visible in plot). When we quantize

this by rounding, the result is the square wave of Fig. 4c. The spectrum of the original sinewave is just a single line (Fig. 4b) while the spectrum of the quantized sine (Fig. 4d) shows harmonics typical of a sampled square wave (approximately odd harmonics with the k^{th} harmonic falling off as 1/k).

We can understand the additional frequency components as being a result of the non-linearity of quantization, and such a phenomenon is known as harmonic distortion. Note that there are some additional smaller components as well, and these are essentially an aliased continuation of the harmonic distortion. We will be looking at this aliasing through a second example in Fig. 5. The remaining portion of Fig. 4 (e-j) will be discussed in Section 3c.

Fig. 5 shows a situation that is identical to Fig. 4 except here the frequency is $0.3f_s$ as compared to $0.025f_s$ in Fig. 4. In this case, it is difficult to see the details of the time waveforms (Fig. 5a sinewave; Fig. 5c quantized sinewave). Yet we see a very interesting result in the spectra. Fig. 5b shows a perfectly pure frequency component of $0.3f_s$. Fig. 5d shows its strongest component at $0.3f_s$ and a strong component at $0.1f_s$ (with many lesser components). While in Fig. 4 we saw what was apparently harmonic distortion, in Fig. 5 we see what seems to be a sub-harmonic (1/3 the original). (Of course, the component would sound exactly like a sub-harmonic.) Yet this component is actually a harmonic - the third harmonic. The third harmonic of $0.3f_s$ is of course $0.9f_s$, but this is in turn aliased to $0.1f_s$. Thus we understand that while this non-linear generation of harmonics follows correct bandlimiting and legal sampling, the components generated by the sampling are subject to aliasing just as though they were included prior to sampling.

Yet one more example is shown in Fig. 6 where the frequency is $0.36f_s$. Fig. 6a shows the original spectrum while Fig. 6b shows numerous strong components in addition to the major one at $0.36f_s$. For example, the third harmonic of $0.36f_s$ would of course be $1.08f_s$, which is aliased to $0.08f_s$. A fifth harmonic would be $1.80f_s$ which would be aliased to $0.2f_s$. Fig 6b shows a considerable array of harmonics aliased back to the baseband. Although not shown, we can also choose frequencies such as $0.364326....f_s$ which is subject to the same general processes, but which would not alias its harmonics to equally spaced positions in the baseband. While the view is essentially the same, it is difficult to display a clear result because the FFT's used here for spectral analysis would be subject to "leakage."

3c. Dither - Breaking Up The Harmonic Distortion

There are many portions of Fig. 4, Fig. 5, and Fig. 6 that we have not discussed. A quick glance at these makes it clear that we were talking about problems with harmonic distortion, and that now we are bringing in some random signals, and we usually do not think of bringing in random elements as a means of eliminating problems. Yet it is

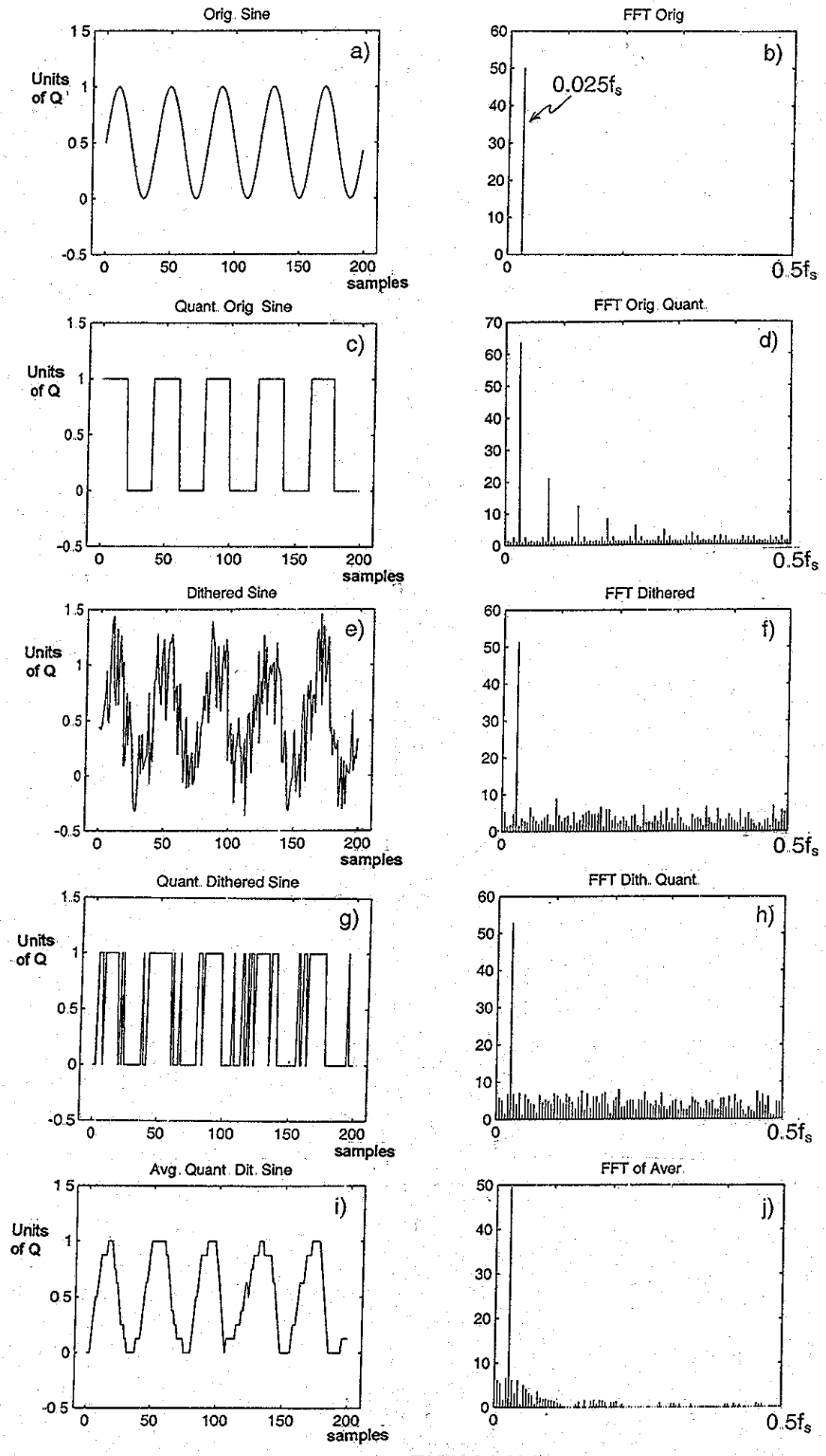


Fig. 4 A small sinewave when quantized shows harmonic distortion. With the addition of noise prior to quantization, the distortion components can be broken up onto the background noise.

Here the spectra are generated using an FFT. The strong DC component has been removed for clarity

precisely this added noise that does end up with an improved signal - at least from the point of view of audio signals. This is the famous "Dither" signal.

Dither has a long history [1], going back to a number of mechanical devices (gear driven computers) that just worked better when they were bounced around a bit (better in an airplane than on the ground). The old practices of electrical engineers tapping the face of an analog "moving-needle" panel meter, or of the chemist reading a balance while it is still swinging are not unrelated. Even more astounding perhaps, some animals are able to take advantage of noise (a process called stochastic resonance) to perceive signals that are otherwise below their threshold of detection (to the animal's advantage one way or another). Yet while it is not "new," it is still a remarkable result.

Fig. 4e shows the addition of a random signal of "amplitude" Q to the original sinewave of Fig. 4a. Exactly as we would predict, the spectrum of this combination (Fig. 4f) shows a strong sinewave over a broadbanded background noise. This is not better than Fig. 4b - but we were not interested in comparing unquantized signals. Fig. 4g shows the quantized version of Fig. 4e. As in the case of Fig. 4c, we still have only two quantization levels total. Yet, as we expect, when the sinewave is near its zero crossing points (crossing 0.5 in this case) we get multiple zero crossings that are due to the noise. In some sense, the hard edges of the blocks have been cracked apart The result is not readily available to the eye from Fig. 4g, but from the spectrum of Fig. 4h, we see that the harmonic distortion seen in Fig. 4d is gone, in favor of a broadbanded background. Put simply, we would have liked a pure sinewave (Fig. 4b) but we can't have that so we prefer Fig. 4h (which is similar to Fig. 4f) to the harmonic distortion of Fig. 4d. (We would expect Fig 4h to sound similar to Fig. 4f, which we know we hear as a sinewave plus random noise.)

Fig. 4i and Fig. 4j show a final manipulation where the samples of Fig. 4g are averaged together eight at a time. This gives us a reasonable feel for how dither works. The averaging is a low-pass filtering, as can be seen in Fig. 4j. Note that the output time signal (Fig. 4i) tends to reach the quantization limits where the original sinewave peaked. Near the zero-crossings of the original sinewave, the dither caused a rapid fluctuation between quantization levels, which roughly speaking, balanced to the quantization midpoint. The procedure is essentially the same as pulse-width modulation. As the sinewave moved up above 0.5Q, it increased the probability of levels Q=1, and as it moved below 0.5Q, approaching 0, it increased the probability of Q=0. It is as though the noise searches beyond the limits of the sinewave for the better estimate of the truth. While the result of Fig. 4i is not a great imitation of the original sinewave, it is the spectrum of Fig. 4j that is most useful. In this case, dither allows us to trade a undesirable distortion for an acceptable random noise.

The additional graphs in Fig. 5 are similar to the corresponding ones of Fig. 4. The most important comparison is probably that between Fig. 5d ("subharmonic") and Fig. 5h (background noise). In this case, we see that averaging (Fig. 5i and Fig. 5j) does not

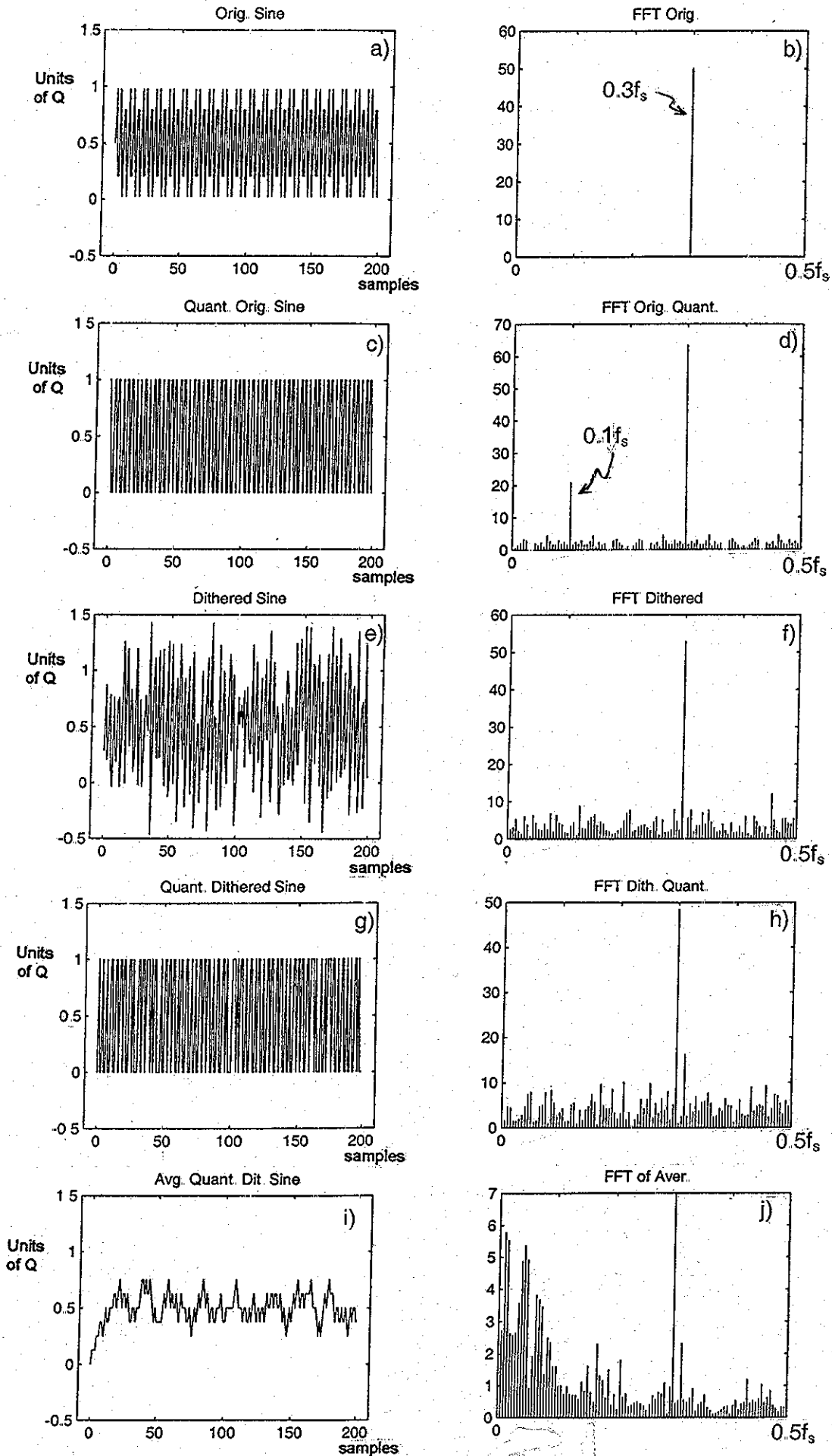


Fig. 5 When the frequency is increased to $0.3f_s$, the "harmonics" now appear in aliased form, and here the third harmonic looks like a sub-harmonic. Dither is effective in this case as well.

Here the spectra are generated using an FFT. The strong DC component has been removed for clarity

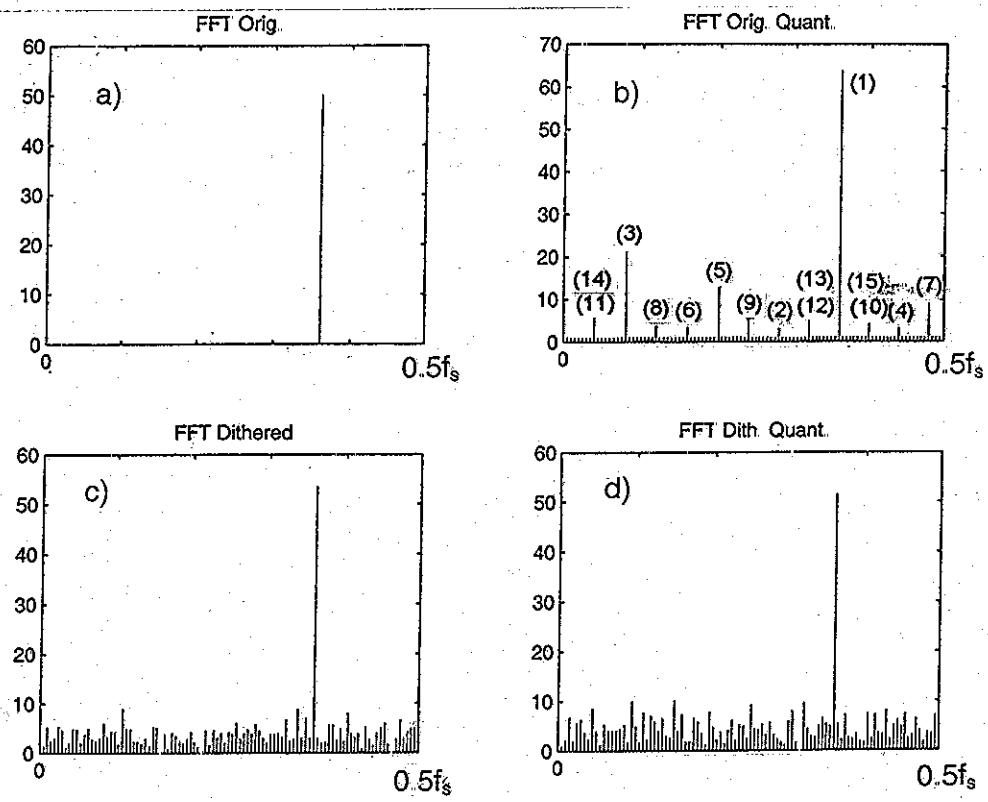


Fig. 6 Here the situation is similar to Fig. 5 except the frequency is $0.36f_s$. The spectrum in (a) is pure, but the spectrum of the quantized signal shows an array of aliased harmonics (as numbered). Once again, dither breaks up the individual components, except for the desired fundamental.

appear so effective, and this we can understand by the oversimplified nature of this low-pass filter: the $0.3f_s$ component sinewave survives this length-8 averaging only in the second sidelobe. In Fig. 6, we see the strong contrast between harmonic distortion components (6b) and noise (6d).

3d Dither - Increasing Resolution

In Section 3c we used dither to break up frequency components that were heard as harmonic distortion and replace them with a more benign background noise. While this is impressive, the use of dither goes even further. In fact, it allows us to effectively encode signal levels that are below the quantization interval (Fig. 7), and to effectively increase resolution to achieve signal levels that are not small, but which are between available quantization levels (Fig. 8).

Fig. 7a shows a sinewave of amplitude 0.4Q. Its spectrum in Fig. 7b is a pure sinewave of frequency $0.3f_s$. Yet when quantized, we get everything rounding to zero

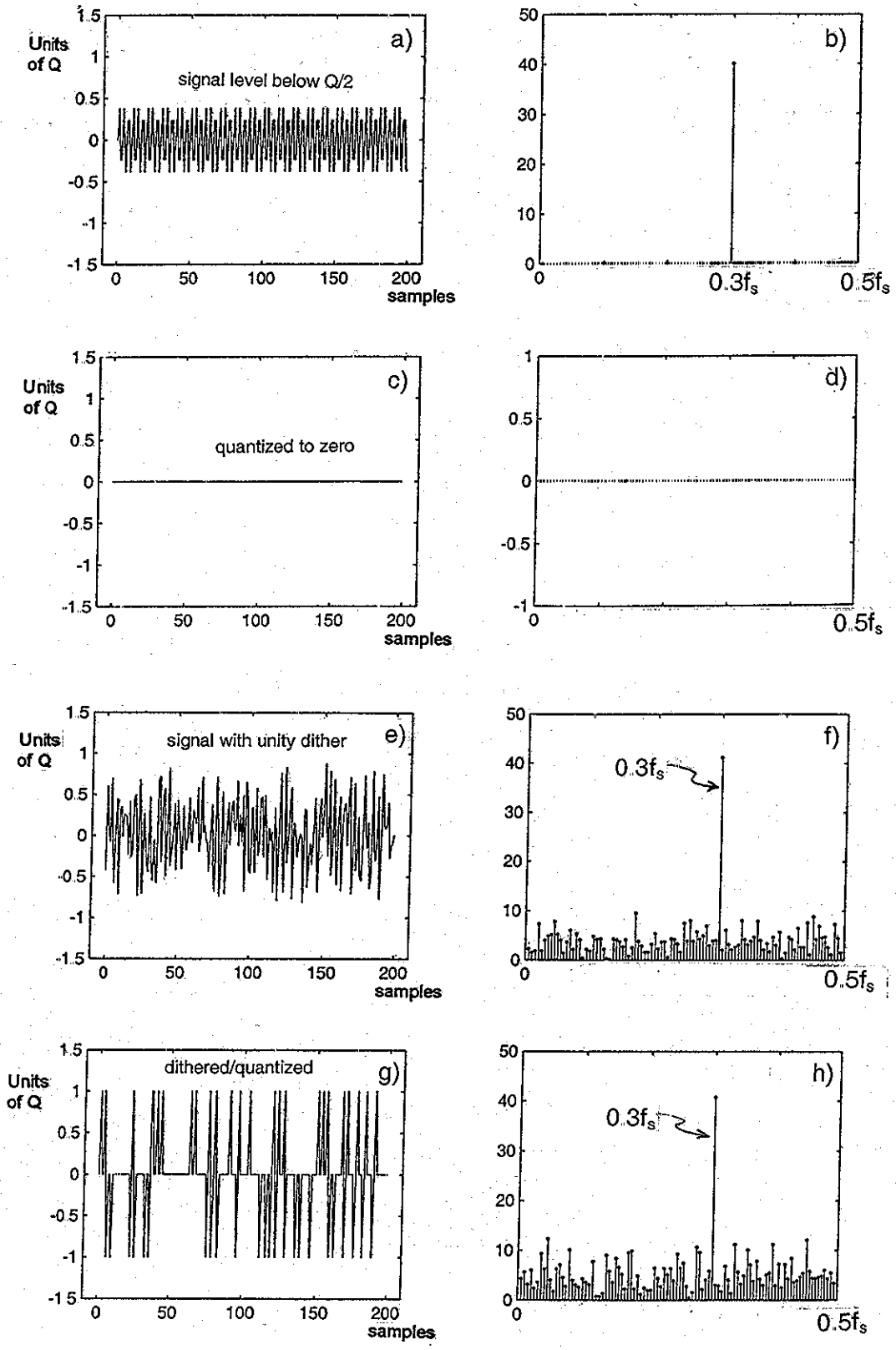


Fig. 7 A small signal (a) quantizes to zero (c,d) and is lost. With the addition of dither (e) the signal now triggers the quantizer and is detected (g,h).

(Fig. 7c, 7d). The signal is lost. When dither is added (Fig. 7e, 7f), we get a noisy sinewave as before. Now with quantization (Fig. 7g, 7h), the lost sinewave reappears. Keep in mind that we are comparing Fig. 7h to the null result of Fig. 7d. This suggests that even very small signals compared to the quantization level may be usefully encoded. [From the point of view of a crayfish in a babbling stream, the babble noise might be added to the vibrations of a hungry bass, making it possible to just trigger a threshold for firing a nerve cell.]

Perhaps the aspect of using dither that is of the most general importance is that of increased resolution, the ability to achieve effective levels that are between available

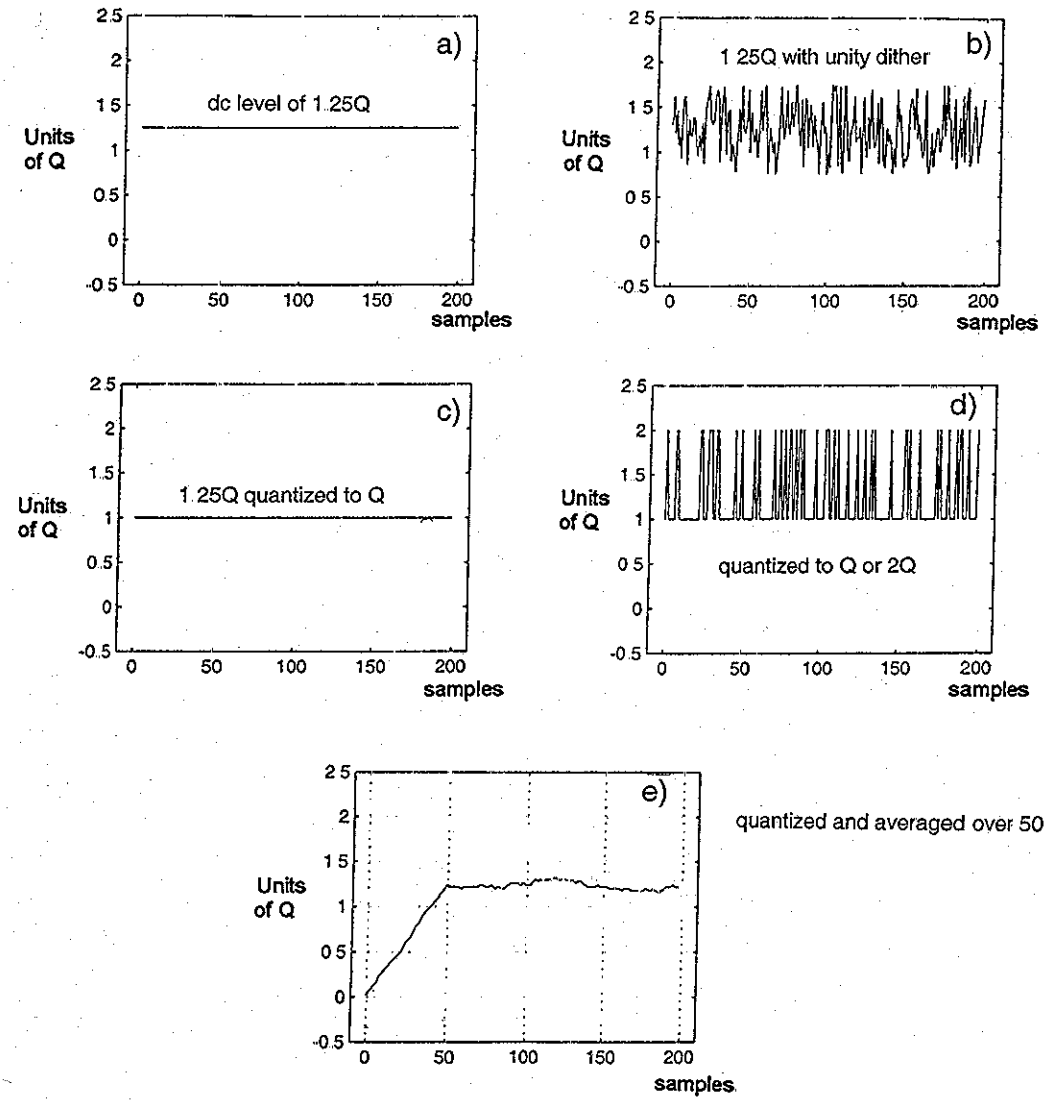


Fig. 8 Although 1.25Q is not an actual quantization level, through the use of dither, we can approach the 1.25Q level as an average.

quantization levels. Fig. 8a shows a dc level of 1.25Q, which is rounded to Q (Fig. 8c). With the addition of dither (Fig. 8b) of amplitude Q, we see that now multiple quantization levels are reached: many (about 3/4) are Q and fewer (about 1/4) are 2Q (Fig. 8d). Now by averaging (Fig. 8e), we can achieve an output that hovers about 1.25, the true value. Here we have averaged over 50 samples, so it is only the output samples from 50 to 200 in Fig. 8e that represent a typical result.

It is probably clear that nothing depended on our choosing 1.25Q. We might have chosen 1.26Q or 1.7234Q, and we would still expect some useful trend toward that level. The exact degree to which a quantizer achieves increased linearity and increased resolution by this procedure depends on correct selection of dither amplitude and probability distribution shape. In most cases, the dither would be supplied by some pseudo-random digital method.

3e. Oversampling and Noise Shaping

3e-1 Introduction

Here we will be looking at two methods that are usually very effective if used together. The first is oversampling, taking samples (or obtaining them by interpolation) at rates that are substantially higher than that required by the sampling theorem. There are many reasons why oversampling is useful. The second topic is noise shaping. This means that we may be able to arrange things so that while the quantization noise power is necessarily present, most of it can be made inaudible. In the context of this section, we are attempting to reduce quantization noise, and at the same time, reduce the number of bits in our data converters, which seems contradictory.

In the derivation of equation (1) for quantization noise power, we noted that only the quantization interval Q was involved in the final result. We argued that in the derivation there were some implied conditions (of importance) on signal level and on the number of bits, but none on sampling frequency. In fact, the assumptions with regard to the characterization of the quantization noise in the frequency domain are generally extended to suggest that the spectrum of the quantization noise is white (the values of the error are uncorrelated), from $-f_s/2$ to $+f_s/2$. (For simplicity, we only need to speak of the noise from 0 to $f_s/2$.) Accordingly, suppose that we have a sampling rate of 44.1 kHz and are digitizing a music signal. We know that the standard of 44.1 kHz was selected in part (as a major "ballpark" factor) with the idea that half this rate just manages to exceed the audible bandwidth. This audible bandwidth is variously taken as 15 kHz, 18kHz, or 20 kHz - it varies with different individuals and drops with advancing age. But suppose we instead chose 88.2 kHz, so that the quantization noise is distributed from 0 to 44.1 kHz. We would only hear the noise inside an audible bandwidth from about 15 Hz to slightly

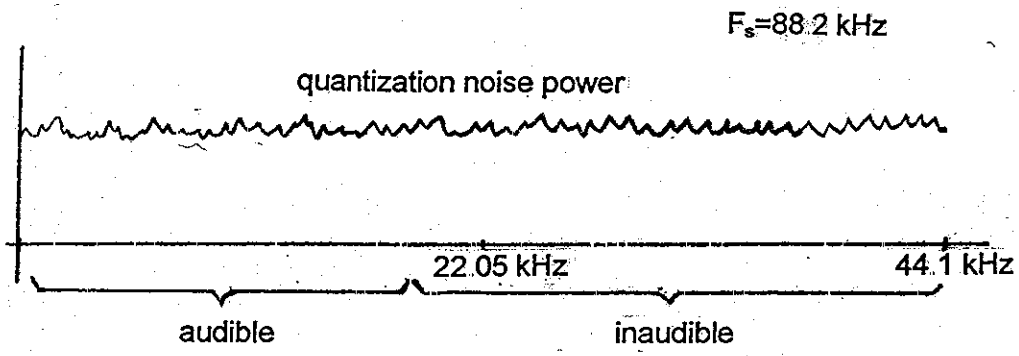


Fig. 9 Oversampling by 2. About half the noise is inaudible. This amounts to 3db improvement (half a bit) for this one octave of oversampling.

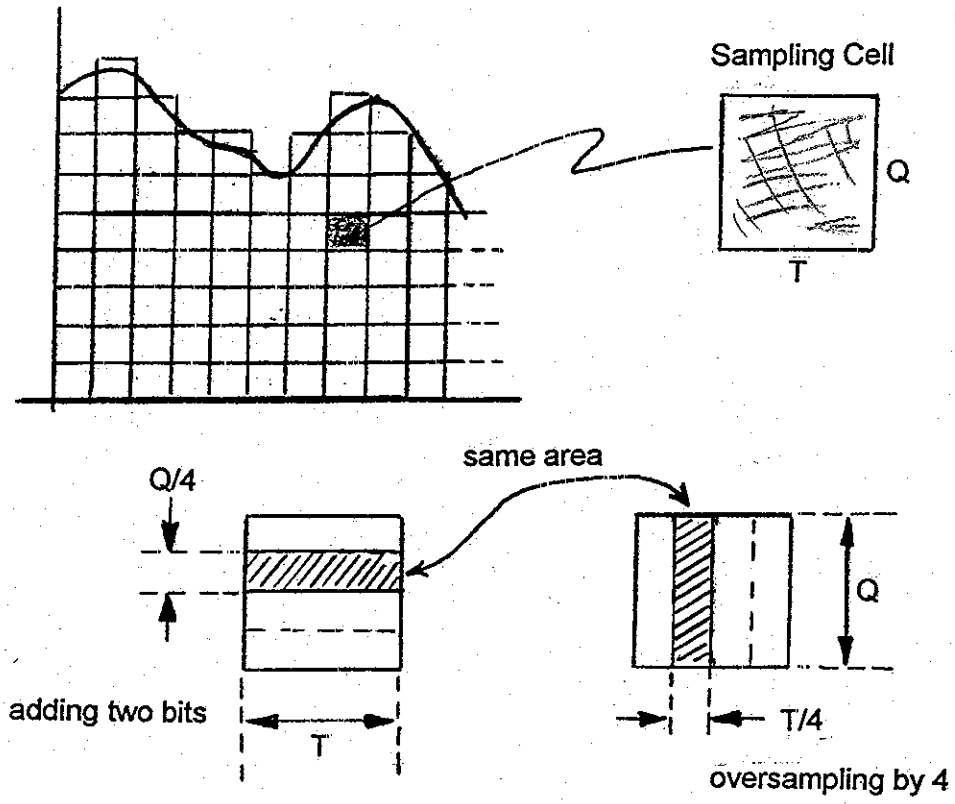


Fig. 10 A sampling cell divided in two ways

under 20 kHz. About half the noise would become, automatically, inaudible (see Fig. 9). (Incidentally, there are other ways in which the high-frequency noise might be blocked. The amplifier might not pass it, or the loudspeakers might not respond, so the noise might never even get to the ear.)

It is important to understand why this oversampling is a reasonable thing to do - why we might hope to win the game we are playing. Of course, it makes perfectly good sense that if we take more samples per unit of time, we have, in a very general sense, a better approximation to the actual analog waveform. But the sampling theorem has told us that (mathematically) we need not try harder in the direction of increased sampling rate. What we are after is less quantization noise. We need to understand two things: that we can use a higher sampling rate to obtain the equivalent of more bits, and that it makes sense (considering the electronics involved) to use the oversampling approach.

Fig. 10 shows a typical "sampling cell" [2]. The cell's sides are T (= $1/f_s$) and Q. In reconstructing from digital samples to an analog waveform, we are first building with blocks this size (the output of a D/A for example). After that, we will smoothen the waveform with some analog filtering. This analog filtering closely resembles integration in a local frame. We are thus concerned mainly with the area of the building block, QT in this case, but the exact shape of the block becomes less important. If we add a bit, our blocks become smaller, and we have an area (Q/2)T, half the original. Two added bits would mean working with an area of (Q/4)T. The same areas could be obtained with oversampling by 2, Q(T/2), or oversampling by 4, Q(T/4), as shown in Fig. 10. So our trade-off seems to be between increased resolution in amplitude, or increased resolution in time. Why do we choose time (oversampling)?

The first reason for choosing time is that, while we are specifically concerned here with reducing quantization noise, we already needed to choose a time (or sampling rate) approach because of another problem: the analog filtering problem. We have the problem of implementing analog input guard (anti-aliasing) filters and reconstruction (smoothing) filters, and with a sampling frequency only slightly greater than twice the bandwidth, these need to be very sharp - of high order and extreme sensitivity. If we opt instead for even just twice the original sampling frequency, the filters are much easier to design. (In fact, we more-or-less change from a difficult analog filtering problem to a doable digital filter problem by doing this.) No alternative approach of instead increasing the number of bits would affect these analog filtering difficulties in any way. So we have a first reason for oversampling.

The second reason for choosing increased subdivision of the sampling cell in time has to do with analog signal-to-noise considerations. Any analog voltage source (perhaps a microphone amplifier) has a maximum size signal, and some background noise-floor. At the same time, some voltage levels we may consider are likely to have offsets and other variations (drift) with time and temperature that are non-negligible. And the components

used are known only within tolerances of a few percent. What this means is that when we start with a certain voltage level, say ±10 volts, if we expend 8 bits, we have values of Q less than 100 mV, and by the time we get to 16 bits, the levels would have to be less than 1 mV apart. Typically, analog noise levels could be larger than this. These would be difficult to work with, and way too expensive. In fact, we are hoping to work with fewer bits, not more.

Given this difficulty of further subdividing the amplitude scale, do we have a similar sort of problem in asking our electronic circuitry to divide time up into smaller and smaller intervals? Well, we are talking about audio rates, and we know that radios exist. We have a lot of room in that direction. A sampling rate of even as much as 12 MHz might be workable. Moreover, we have so much potential in the direction of higher sampling rates that we generally think not just of getting to something workable with 16 bits devices (breaking even), but getting to the same performance with far fewer bits - perhaps a single bit. Of course, as mentioned, we are not going to get to 1-bit on oversampling alone - we will also need noise shaping.

This idea of using a sampling rate that is substantially higher than the minimum required (commonly factors of 2, 4, 8, and 16 higher, and may go as high as 256), is known as "oversampling." In some cases, this simply means recording at a higher rate -- actually a sampling process. In many other cases, the actual oversampling technique refers to the recreation of additional samples that are calculated from stored samples (e.g., a CD) rather than the taking of original samples at a higher rate. For example, if we have sampled at 44.1 kHz we could in theory perfectly reconstruct (return to analog) a signal with a 20 kHz bandwidth (nearly so anyway). This means that we could then resample (oversample) the recovered analog signal at a higher rate, for example, 176.4 kHz. A better, less round-about method would be to construct the additional digital samples numerically from the available samples - the process of digital interpolation, which is well understood [3]. There are many advantages to oversampling. For whatever reasons we have for resorting to oversampling, we are going to get, for free, a reduction of half the noise power each time we double the sampling frequency. This amounts to a reduction of the noise amplitude of $1/\sqrt{2}$, which is a 3db improvement or half a bit. Not a great deal, but it might be simply a free bonus. This 1/2 bit per octave of oversampling is a reference values we need to keep in mind below.

More to the point, what if we could arrange for the quantization noise, now uniformly distributed in frequency, and therefore, partly in an inaudible region, to be preferentially distributed in the unimportant region above the audible range? This is what we call "noise-shaping" (see Fig. 11) [4,5]. Noise shaping and oversampling are separate notions, but we almost always consider how the two work together, and we will discuss these procedures essentially as one here.

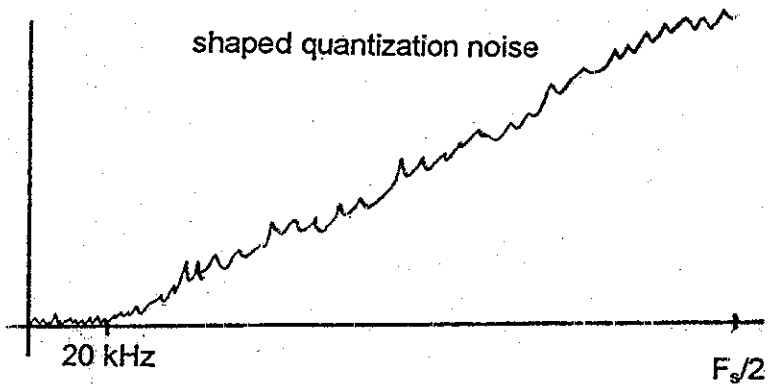


Fig. 11 Oversampled and Noise Shaped

3e-2 Two Applications to Consider

The two major applications of Oversampling/Noise-Shaping (OSNS) are, not surprisingly, recording and playback. In engineering, there are usually trade-offs and no "free lunches." But oversampling comes close to being a free lunch. While it requires some cost (at least in engineering effort), once implemented, there are usually only good results from an implementation/performance point of view. We get less quantization noise, less phase distortion, simpler analog interfaces. All of this can result in better performance at lower prices. In the two applications, the resources are allocated in different ways however. This is because there are usually relatively few recorders and many, many players. The recorders are optimized for high performance and reasonable cost. The players are optimized for reasonable performance and low cost.

The structures and procedures surrounding OSNS are unusual (such as discrete-time filters that work with unquantized samples), sometimes counter-intuitive, and often appear in their most extreme versions (such as quantizing to only one bit). In Sections 3e-3 and 3e-4, we will attempt to present the noise shapers in a complete and fairly realistic context. Before we begin these studies, we can introduce some to the items that will appear in the figures.

Fig. 12a shows a box labeled Q which we will consider to be a quantizer or a re-quantizer. In the quantizer case, the box takes in samples x(n), which we take to be discrete in time, but not yet in amplitude. In a re-quantizer case, we would be thinking of x(n) as being represented by a number of bits, and we want to reduce the number of bits used (perhaps to just a single bit). In either case, we are interested in the errors that result from quantization. Accordingly, below the Q box we show a summer that calculates the error e(n).

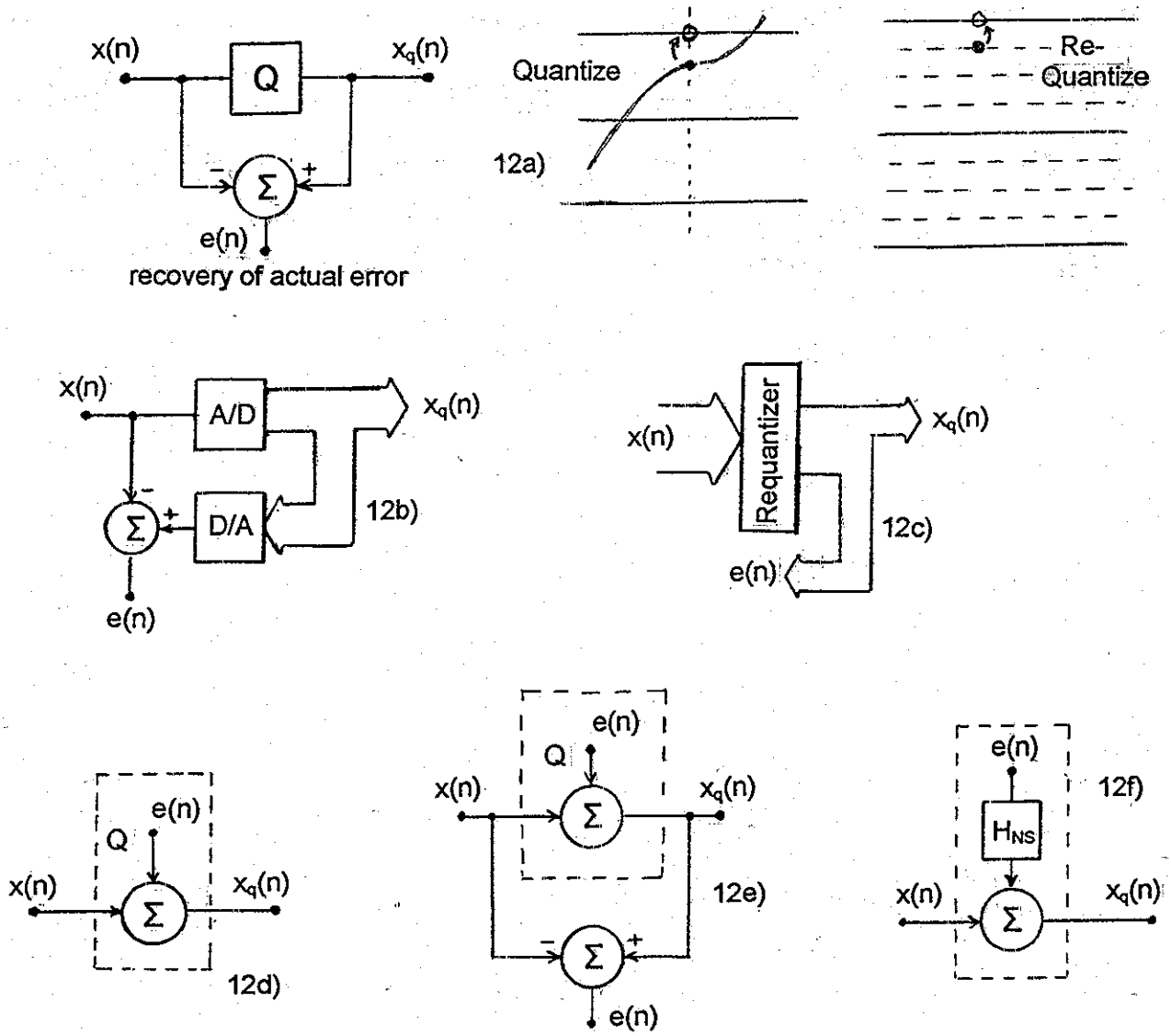


Fig. 12 Devices Seen In OSNS

Fig. 12b shows details of an actual quantizer realization. This perhaps looks silly because we use an Analog-to-Digital (A/D) converter, which we might think is itself a quantizer. It is. Then in calculating the error, we need in turn to use a Digital-to-Analog (D/A) converter. In a sense, a scheme like Fig. 12b is already pretty much a model. The A/D and the D/A are part of some overall converter chip which handles sampled data. At one extreme, the A/D could be just a single bit device (a comparator) and the corresponding D/A would be just this one bit fed back. Even simpler is the re-quantizer of Fig. 12c. This re-quantization is actually just a matter of not moving some of the bits further along. The bits passed along are the most significant bits while the least significant bits are automatically the error. Again, in many cases, only one bit is kept.

In Fig. 12a, we saw that a quantizer produced an error that could be recovered by subtraction. It follows that we can model the quantizer as a summer where the error e(n) is added (Fig. 12d). We need to note that this is a model. There is no actual noise generator or summer. It's just that the quantization process produces an error which we treat as additive. The noise e(n) is internal to the Q box. We can not feed noise into the Q box, nor take it out, except as we can recover e(n) if we have the input and the output of the Q box. At best, we know something about the statistics of e(n). Perhaps the best way to view the quantizer model is seen in Fig. 12e, where the e(n) in the Q box is unknown, but is being measured by the e(n) below the summer.

Eventually we will see that the noise generated internally to the quantizer can have an altered spectral shape. This is a matter of the overall feedback structure around the quantizer. In such a case, an altered model (Fig. 12f) is sometimes seen. Like e(n) itself, the noise-shaping filter $H_{NS}(z)$ is a model. Typically, $H_{NS}(z)$ will be first-order $(1 - z^{-1})$ or second-order $(1 - z^{-1})^2$. While there is no noise generator, nor any hardware noise-shaping filter, the noise-shaping transfer functions are real enough in terms of the overall structure.

3e-3 Oversampling Noise-Shaping Recording

In the recording case, we have in mind an analog signal with an audio bandwidth of perhaps 20 kHz. We want to convert this to a standard format of perhaps 16-bits at a sampling rate of 44.1 kHz. Initially we might think of using "brute-force," an analog input-guard filter of high order (perhaps 12^{th} to 16^{th} order) cutting off sharply between 20kHz and 24kHz, followed by an analog-to-digital conversion system which would need to be 18 bits or more to get 16 "good bits" (the error is specified for most converters as 1/2 the least significant bit). With recording, as we said, price is less of a consideration, but we still need performance. We hope we can get better performance at less cost by using OSNS, rather than brute-force.

Fig. 13 shows the application of the OSNS procedure to this recording process. The central portion of the scheme is what is called a "sigma-delta" converter. [The alternative name "delta-sigma" converter is also used, but refers to the same thing. There is perhaps currently an edge to the sigma-delta choice of name, particularly as the literature refers to sigma-delta modulation (SDM)]. At the very left, we see a microphone fed to a "wimpy" analog low-pass filter (perhaps just a first-order R-C low-pass). Here we assume that the low-pass filter gets down to a negligible level by some frequency $F_s/2$. So the analog signal can be sampled at a rate F_s, producing a discrete-time signal x(n). Here we will assume that $F_s >> f_s$ where f_s is the sampling frequency we eventually want to end up with. For example, we might have F_s=705.6 kHz while f_s=44.1 kHz, which is oversampling times 16.

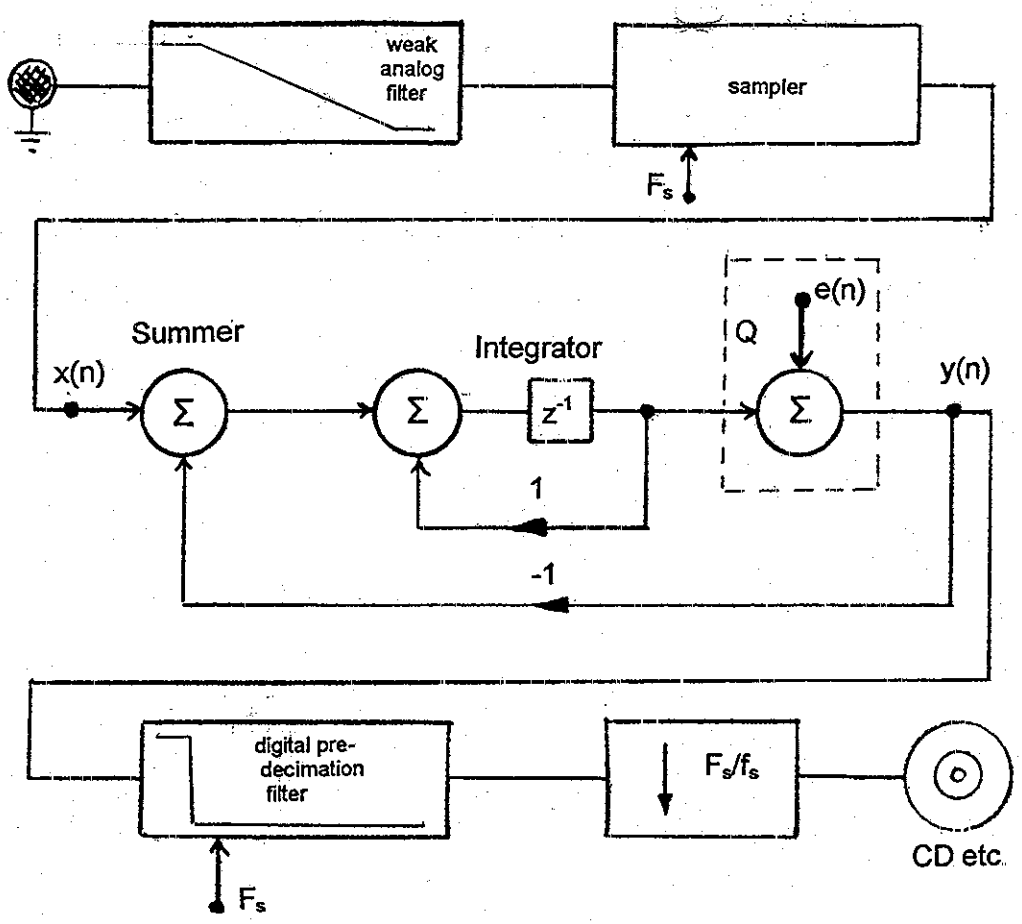


Fig. 13 OSNS For Recording

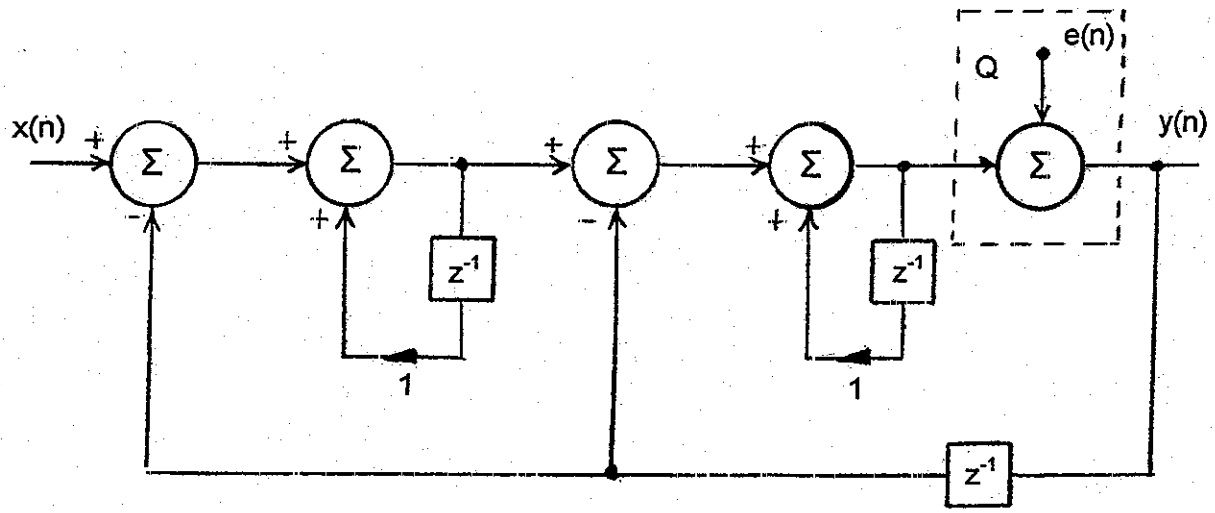


Fig. 14 Second-Order Noise-Shaping (Recording)

We now turn our attention to the central portion between x(n), the unquantized discrete time signal, and y(n), the quantized signal. Note that the system has two inputs, x(n) and the internally generated noise e(n). We see that the system consists of a summer, a discrete-time integrator, and a modeled quantizer. It is fairly easy to solve for Y(z) as:

$$Y(z) = E(z) [1 - z^{-1}] + X(z) z^{-1} \quad (4)$$

The significance of this equation is that the noise is subjected to a high-pass shaping, while the signal is merely delayed (no spectral shaping). The noise shaper, $[1 - z^{-1}]$, has a response of 0 at f=0 and a response (gain) of 2 at $f=F_s/2$. It is not difficult to integrate the frequency response of the noise shaper from 0 to $f_s/2$ (or to the edge of the audible band or other specified cutoff) to get the total noise power [6]. The general result (Hauser's rule of thumb) is that this first-order noise shaping will add the equivalent of 1.5 bits per octave of oversampling with a "penalty" of one bit (the gain of 2). For our example, there were four octaves of oversampling ($1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16$) so one would expect a 4x1.5-1=5 bit improvement.

What this means is that if the quantizer converts say, 12 bits, that the additional 5 due to oversampling would produce 17 bits total. This is where the structures below the noise shaping quantizer come in. Note that we have to reduce the sampling rate from F_s, the output, y(n), to f_s at the CD location. This of course requires a reduction of the bandwidth to $f_s/2$ - the job of the digital pre-decimation filter which would be clocked at F_s, but which would have a low-pass cutoff of $f_s/2$. Note that the predecimation filter is also what we need to remove the shaped noise. Finally we note that the filter itself is likely FIR and produces an output as the result of digital multiply/ accumulate operations. In our example, we assumed that y(n) has 12 bits. If the filter coefficients are 16 bits, the accumulated product could have 28 bits or more, and the noise shaping has made 12+5 = 17 of them significant, which would be enough for a CD.

Fig 14 shows a second-order noise shaper. The analysis of this network is not difficult and yields:

$$Y(z) = E(z) (1 - z^{-1})^2 + X(z) \quad (5)$$

Here the noise is subject to a second-order high-pass, while the signal is still subject to only a flat response. (In this case there is no delay - there are a number of structures that amount to virtually the same thing.) The rule of thumb for the second-order noise shaper results in 2.5 bits for each octave of oversampling, with a 2 bit penalty. Four octaves of oversampling would result in an 8-bit gain.

Note that with second-order noise shaping and an oversampling factor of 256 (8 octaves - which would be F_s=11.29 MHz - possible but not easy) there would be an 18-bit gain. Accordingly even a single bit quantizer could result in 16 or more good bits. This is more or less the idea of getting to a 1-bit converter. It's not that easy in practice - additional tricks are required.

3e-4 Oversampling Noise-Shaping Playback

Fig. 15 shows the playback side of the OSNS procedure. In the "brute force" approach (Fig. 15a), we would expect to take samples from the CD at 16-bits and 44.1 kHz. This would be converted to analog with a good 16-bit D/A conversion (i.e., perhaps an 18-bit converter) followed by a sharp analog filter with a cutoff between 18kHz and 24kHz. As with the recording case, oversampling alone gives us some relief. In using oversampling (Fig. 15b), we would first increase the sampling rate from f_s to F_s through an interpolation procedure, which involves a digital low-pass with a cutoff at $f_s/2$. The interpolated signal is then passed through a good 16-bit D/A conversion, and then, only a simple analog low-pass is needed to remove the residual steps (the images are spaced at F_s, not at f_s). In effect, the strong digital low-pass and the weak analog low-pass act in series to achieve the needed image rejection. Accordingly, oversampling alone is working to greatly ease the analog filtering problem, and it is also the case that we also get some noise relief, the 1/2 bit per octave of oversampling. For example, if we had oversampling by 16 (four octaves) we would gain 4x0.5 = 2 bits, as a bonus. These two bits might well mean that a 16-bit converter would give 16 good bits.

Here in this case we are talking about a playback device. Thus we want to get component cost down, and a 16-bit converter is still very costly. By using noise shaping, we can obtain extra bits. Here, we can use the extra bits to reduce the number of bits in the converter. That intention makes this a re-quantization problem. In fact, one ideal goal would be to get to the extreme of only having to convert a single bit: essentially just a comparator (the sign bit) - not an actual D/A converter.

The noise shaping requantizer is seen between x(n) and y(n) in Fig. 15c. Here it is the error that is fed back, rather than the quantized signal in the cases of the recorder. The signals x(n) and w(n) would be of the original wordsize, while y(n) would be fewer bits, perhaps just one sign bit. In fact, the error need not be computed - it is just the discarded least significant bits (see Fig. 12c). The re-quantizer (Q) is modeled exactly as the quantizer way, and we expect that as long as we are reducing by many bits, the same arguments regarding the error statistics apply. It is easy to compute Y(z) as:

$$Y(z) = X(z) + E(z)\,(1 - z^{-1}) \qquad (6)$$

which is our familiar noise shaper, while the signal itself is unshaped. Fig. 16 shows an alternative, second-order noise shaping requantizer to be used between x(n) and y(n) in Fig. 15. The re-quantized output is now:

$$Y(z) = X(z) + E(z)\,(1 - z^{-1})^2 \qquad (7)$$

which is familiar second-order noise shaping.

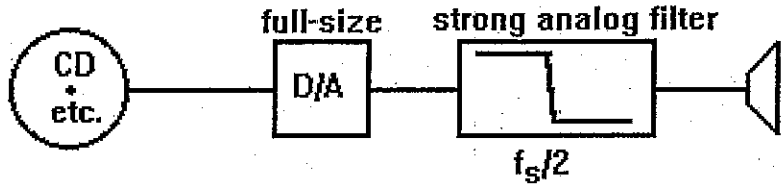


Fig. 15a A brute force playback system would involve a full-size (or slightly larger) D/A converter, and a very sharp analog low-pass filter. Both of these are expensive and/or difficult to implement.

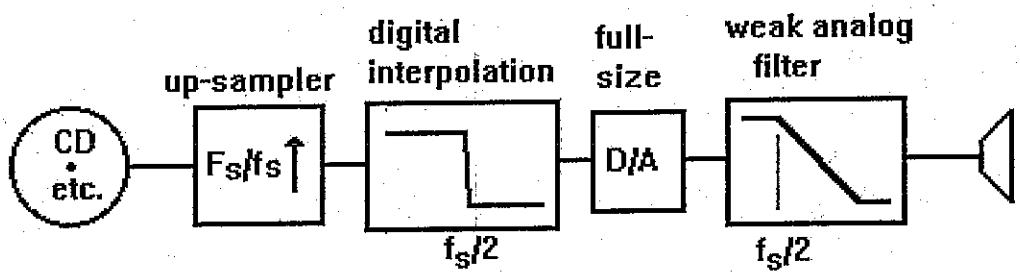


Fig. 15b By using oversampling, interpolation with a digital filter to a much higher sampling rate, we can get around the requirement of having to have a sharp analog filter. That is, the digital filter does most of the work. Because of the oversampling, some improvement in S/N is automatic, but only at the rate of 1/2 bit per octave of oversampling. This might allow us to use a slightly smaller D/A converter.

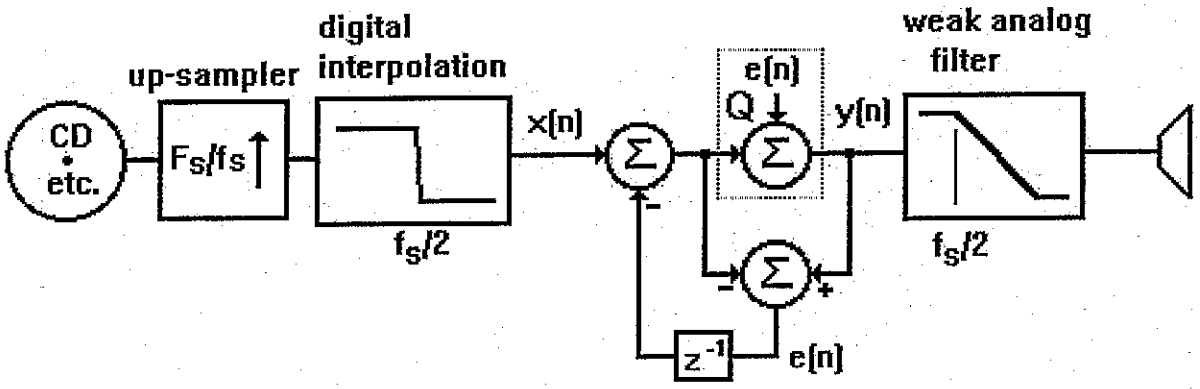


Fig. 15c Here we retain the oversampling interpolation, and the weak analog filter, but we have added the first-order noise-shaping requantizer (requantizer since we are reducing the number of bits). Here, the D/A converter is part of the "Q" box. Note that here, because of the noise shaping, we hope to obtain enough additional bits that the D/A is of a much smaller size – only one bit if possible. The first-order noise shaper gives us 1.5 bits per octave or oversampling, with a 1 bit penalty.

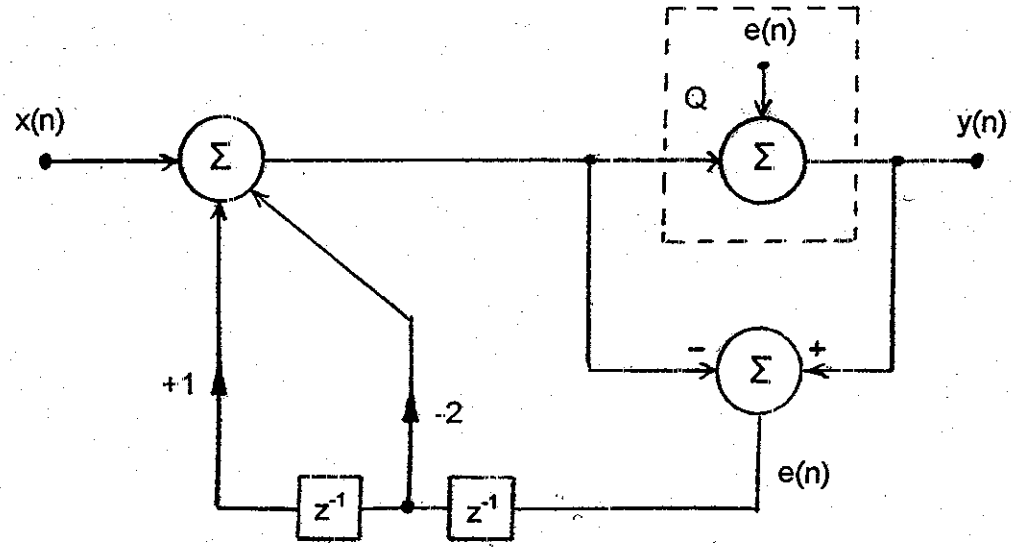
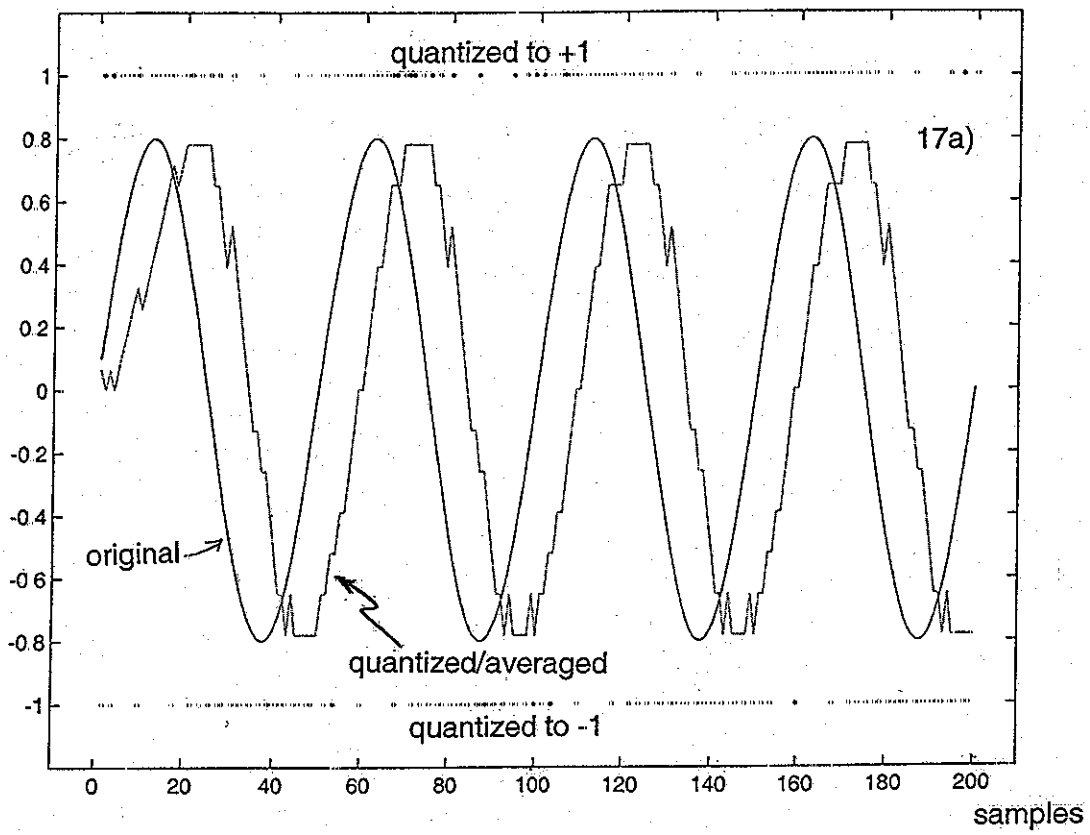


Fig. 16 Second-Order Noise Shaper (Playback)

More details of the noise shaping process are seen in Fig. 17 and Fig. 18. In Fig. 17a we see a sine wave of frequency 0.02 and amplitude 0.8 as the input of the re-quantizer of Fig. 15. Here the quantization is from a very large number of bits (no quantization actually) to just a single bit, as represented by the dots at +1 and -1. Also shown is a reconstructed waveform based on an average over 20 samples. This reminds us a good deal of the experiments with dither looked at earlier. The reconstructed waveform is unimpressive compared to the original sinewave, but extremely impressive relative to the one-bit samples (the dots) from which it is derived. Note that without noise shaping, the one-bit samples would be just a square wave, and a good deal of harmonic distortion would be the result.

The view in the frequency domain (Fig. 17b) is even more revealing. Here the original sine wave should be a single dot (seen at amplitude 80) while the output of the quantizer is the ugly spectrum shown by the dashed line. It is appropriate that the erratic dots of Fig. 17a should have such a noisy appearance. We are at first inclined to reject this until we recognize that it was our purpose to accept much more noise (we are increasing the quantization interval) but to force it into the high frequency region. Indeed, we see that the bottom 10% of the spectrum is tiny except for the one desired component. We may have the case that the spectral energy above this is inaudible, or perhaps we have a low-pass filter rejecting it, such as the length 20 averager (dotted line) shown in Fig. 17b.

Fig 18 shows a similar case, except here we have the second-order noise-shaping re-quantizer of Fig. 16.



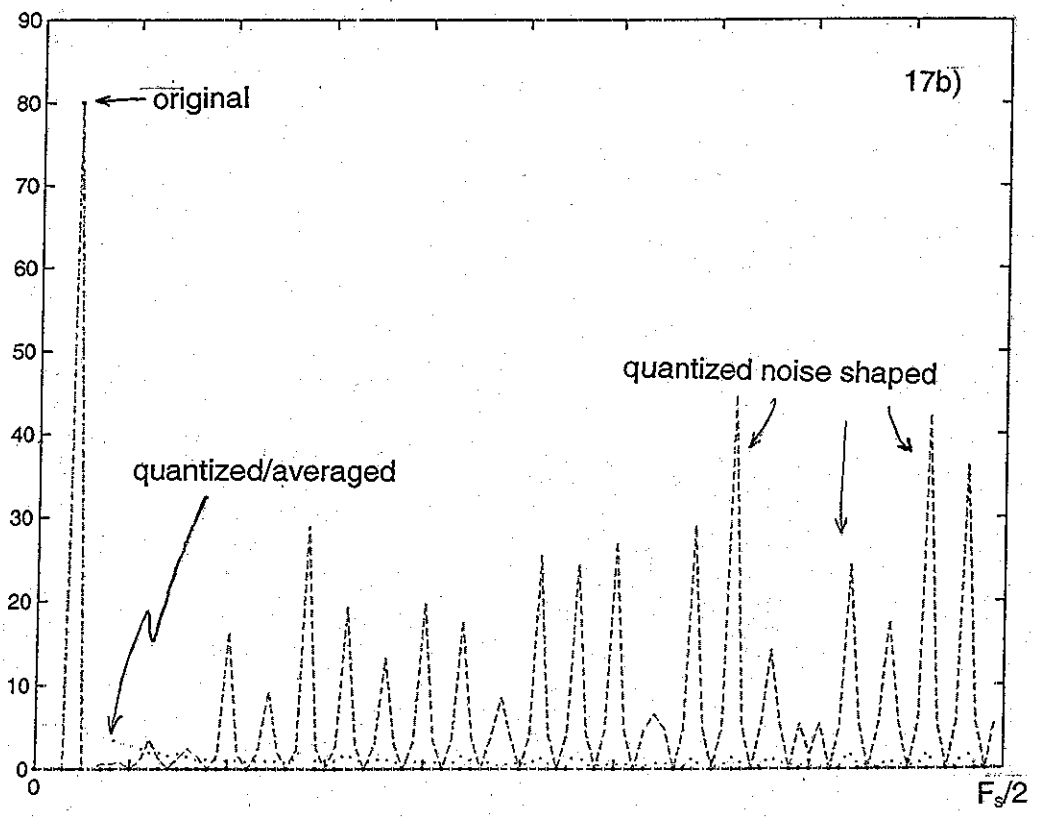
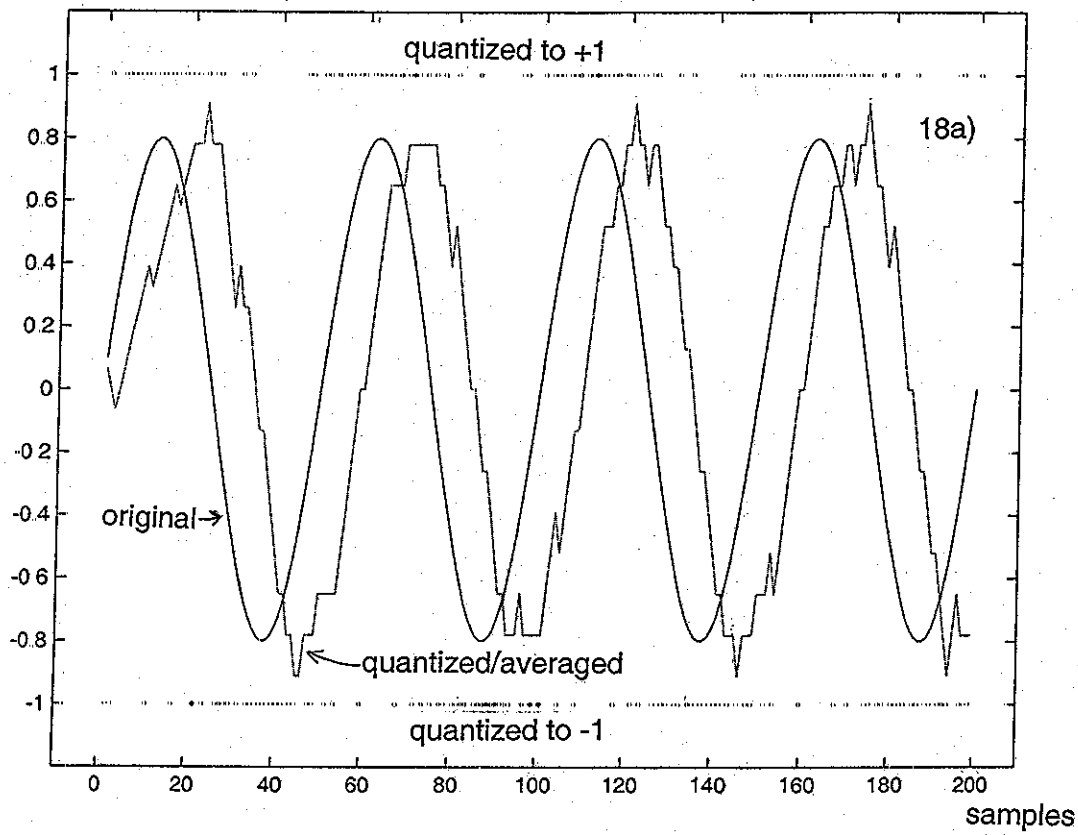


Fig. 17 One-Bit D/A, First-Order Noise Shaping



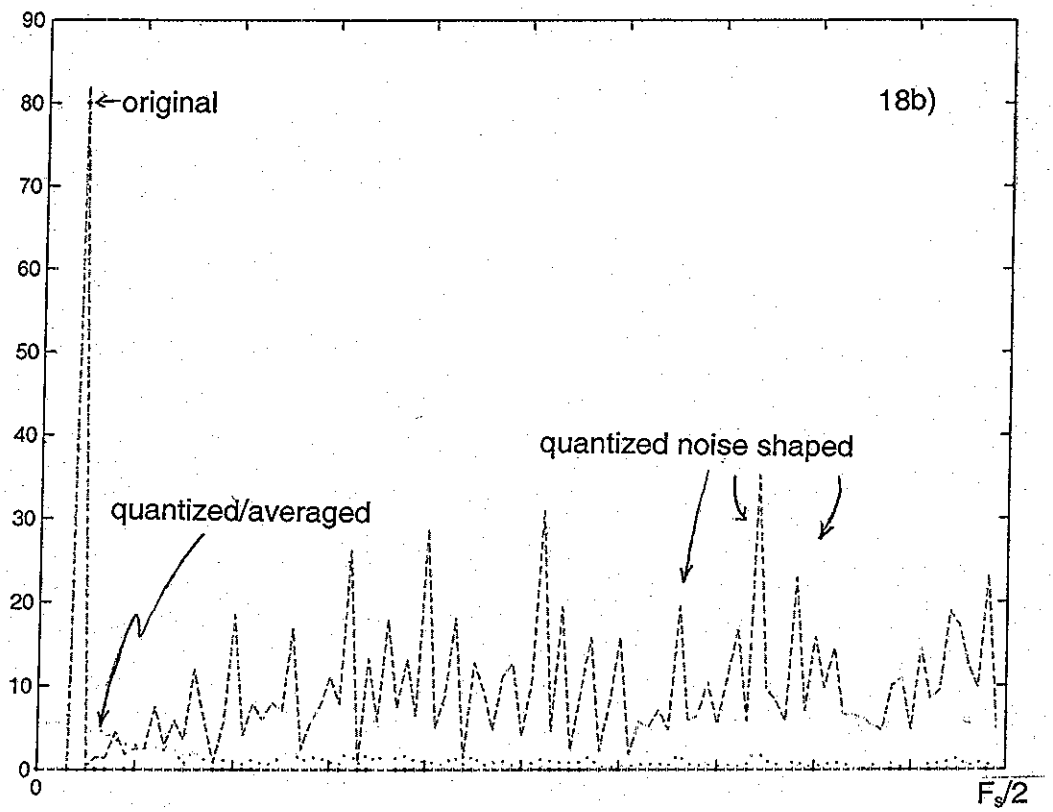


Fig. 18 One-Bit D/A, Second-Order Noise Shaping

REFERENCES:

[1] B. Blesser, "Digitization of Audio: A Comprehensive Examination of Theory, Implementation, and Current Practice," J. Audio Eng. Soc., Vol. 26, No. 10, October 1978, pp 739-771. This is a classic "tutorial" and includes many fine explanations.

[2] K. Pohlmann, Principles of Digital Audio, 4th Edition, McGraw-Hill (2000). A classic. Of particular interest here are Chapter 2, "Fundamentals of Digital Audio;" Chapter 4, "Digital Audio Reproduction;" and Chapter 18 "Sigma-Delta Conversion and Noise Shaping."

[3] B. Hutchins, "Interpolation, Decimation, and Prediction of Digital Signals," Electronotes, Special Issue F, Vol. 15, No. 164-167, July 1986, pp 3-46

[4] M. Hauser, Principles of Oversampling A/D Conversion," J. Audio Eng. Soc., Vol. 39, No. 1/2, Jan/Feb 1991, pp 3-26. Another classic tutorial.

[5] S. Orfanidis, Introduction to Signal Processing, Prentice-Hall (1996)

[6] B. Hutchins, "Improved Signal/Noise Ratio with First-Order Noise Shaping: An Example," Electronotes Application Note No. 345, October 1997

* * * * * * * * * * * * * * * * * * * *

QUESTION (Continued from Page 4)

so the samples of equation (4) came from an amplitude A=1, and the samples are 105° apart, starting at 10°.

So much for being given samples and being told they belong to a sinusoidal waveform. What if we aren't told that? Rather, we just have some samples to ponder.

Well we might suppose that we can learn about the time samples by taking an FFT to see what it looks like in the frequency domain. But, when we use the FFT, two things must be kept in mind. First, the frequency resolution for a length-3 signal is going to be absolutely horrible in general. We will be forced to consider the three samples points to be composed on a single sinusoidal waveform plus a constant. Secondly, the FFT will assume periodic repetition of these three points. Not too much hope. But we learn from what happens. Of course nothing prevents us from taking the length-3 FFT's of the three points.

$$[0 \quad .5 \quad \sqrt{3}/2] \longleftrightarrow [1.3660 \quad -0.6830+0.3170j \quad -0.6830-0.3170j] \qquad (6a)$$

$$[0 \quad \sqrt{3}/2 \quad -\sqrt{3}/2] \longleftrightarrow [0 \quad 0-1.5j \quad 0+1.5j] \qquad (6b)$$

We note right away that the second sequence, equation (6b) has the simpler FFT. Indeed there is no constant (DC) term, and we have a pure sine. This was a

consequence of choosing a frequency of exactly 1/3 for this case. We have an exact integer number of samples (3) per cycle. The resolution is perfect.

For the FFT pair of equation (6a) recall that the frequency was supposed to be 1/12. But of course, the only FFT frequency available for a length 3 FFT is 1/3, so the FFT represents the result accordingly. Because we only have three samples of the length 12 cycle, there is a DC offset. The FFT is lying to us!

To see what bandlimited functions, in the FFT sense, corresponds to these two cases, we can interpolate the three given points by zero-padding the middle of the FFT [2]. Fig. 4b shows the simpler case. It works - we get our sinewave with frequency 1/3 back. Fig. 4a is thus going to be the case from which we can learn something additional. Note that it does fit the three input points, and that it does have a DC offset and a component at frequency 1/3. What it does not show is the "true" frequency of 1/12, which is shown as a dotted line in Fig. 4a.

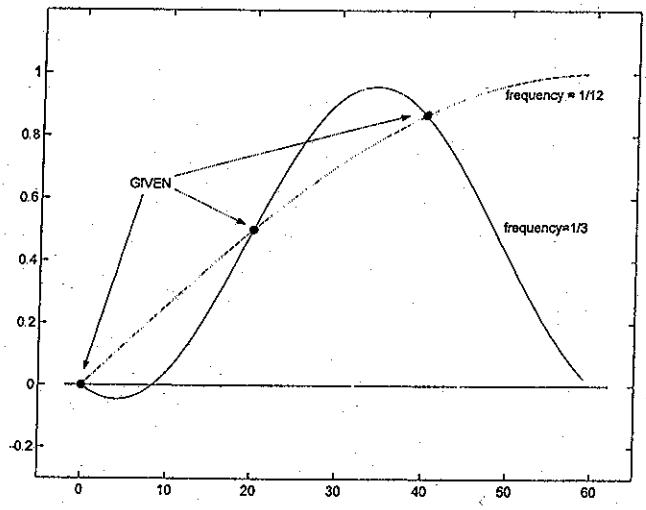


Fig. 4a By taking the FFT (DFT) or the three given points (not much data!) we find a fit to frequency = 1/3. But the points came from frequency = 1/12.

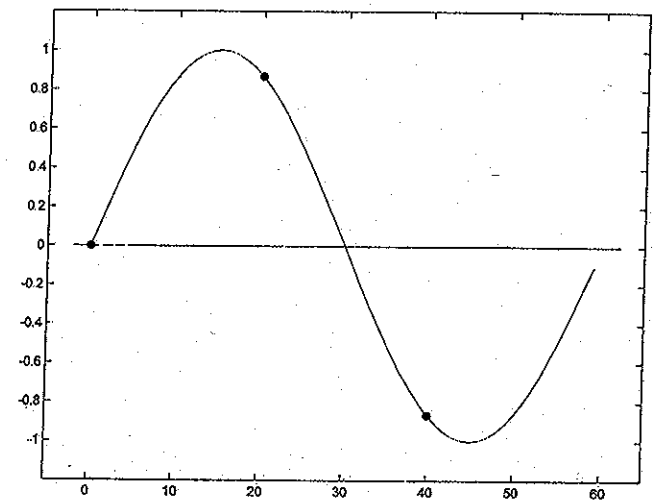


Fig. 4b Here because the frequency of the sinewave actually was 1/3, which is the only DFT sinusoid for length 3, we have perfect resolution and recovery.

This result with the FFT shows us that the FFT is of only very limited use when it comes to reconstruction. Actually we should say that we had no right to expect much from just three samples. The FFT forced the result to be represented in terms of its own frequencies.

What now?

Well, perhaps we should now ask what bandlimited waveform corresponds to the given samples. This seems right, as we eventually expect to recover a signal from samples by using a bandlimiting low-pass filter. Of course we might have tried this before Prony, and before trying the FFT, but we learn a lot by attempting things in different ways. So the question is: If we take three samples of a sinusoidal waveform, and pass them through a bandlimiting low-pass, do we get the sinusoidal waveform out? Fat chance! What happens if you put a localized set of non-zero samples into a low-pass? You get out a smoother, but still localized "blob."

When we get down to looking at how a low-pass filter works, it is convenient to use time-domain sinc interpolation corresponding to an ideal low-pass [3]. This simply means that for each sample, we multiply a sinc, centered on that sample, by the sample value, and sum up all the sincs. Fig. 5 shows a sinc interpolation for the case of just two samples. As suggested above, we get a localized wiggle which dies off when we get away from the non-zero samples.

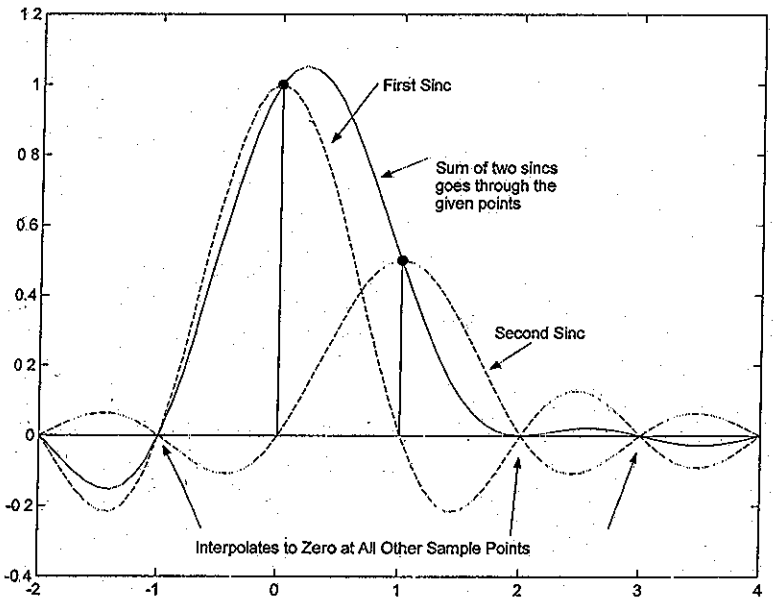


Fig. 5 Classical bandlimited interpolation with sinc waveforms corresponds to some notion of an ideal low-pass filter. Note that the sum is significant only locally about the non-zero samples, and tapers off away from these samples

Fig. 6a shows the sinc interpolation for the three samples 0, 0.5, $\sqrt{3}/2$, along with the sinewave of frequency 1/12 which fits these two samples. The interpolation is perfectly good and is correct – giving us the bandlimited answer. Of course it does not resemble the sinewave, first because it dies off away from the non-zero samples and the sinewave is of constant amplitude for all time. But neither does it resemble the sinewave locally in the vicinity of the non-zero samples. Fig. 6b shows the same interpolation procedures applied to the samples 0, $\sqrt{3}/2$, $-\sqrt{3}/2$; the frequency of 1/3. Again the localized blob does not resemble the constant amplitude sinewave. But there is some resemblance locally in the vicinity of the non-zero samples, better detailed in Fig. 6c. This can be understood in

that the three samples in the case of frequency 1/3 actually do correspond to a full cycle, while three samples are only a fragment of a cycle in the case of the frequency of 1/12. Again - interesting, but not what we were hoping for.

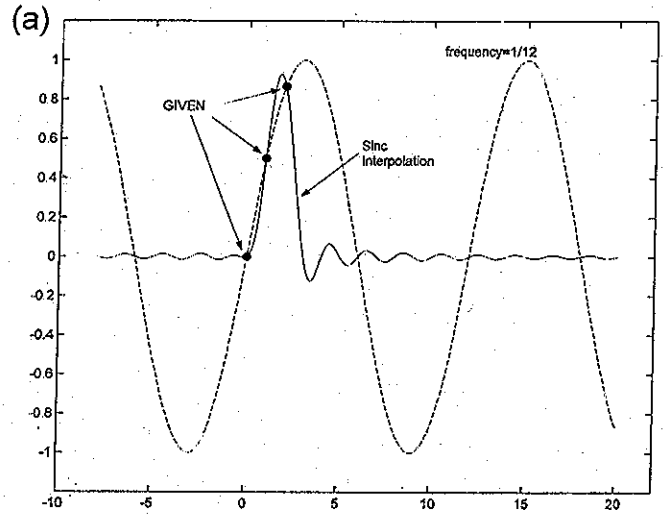
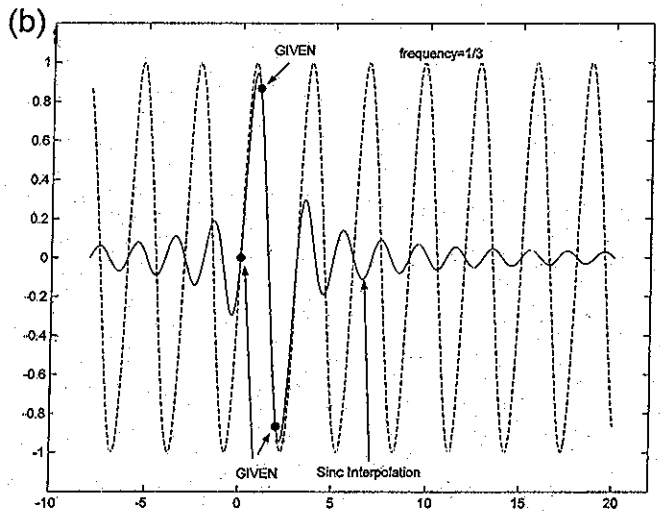
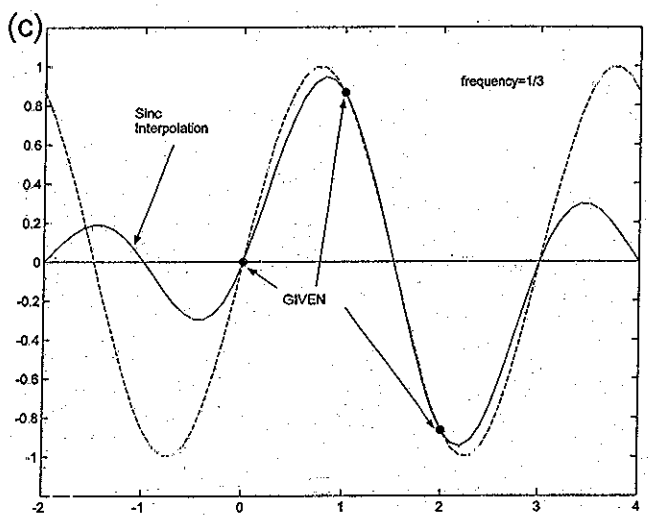


Fig. 6 Sinc interpolation is a "blob" localized around the non-zero samples, in contrast to the sinewave which goes on forever in each direction. Here we have samples of only part of a cycle (Fig. 6a) or only one cycles (Fig. 6b) and interpolation is quite poor in general.





Finally we can try it more or less the right way. We want more samples, and samples of many cycles. After all, small bursts of sinusoidal cycles are heard as "clicks" and are not what we expect musically. The waveform of an individual musical "note" would have, typically, hundreds of cycles. Here we will try something like six full cycles, 18 samples total, and we choose a frequency of 1/2.71 (somewhat arbitrarily) so that the samples are not repeats from cycle to cycle. Fig. 7a shows the result of sinc interpolating the 18 samples, along with the actual sinewave. Here we again see the interpolation sum dieing off away from the actual samples, but the sum does resemble the sinewave, not too well at the ends (Fig. 7b) but quite well nearer the center (Fig. 7c). Since we are actually doing filtering, we can think of the end errors as transients, while the good center is a

steady-state. In terms of the interpolation sum, the errors are due to the truncation (windowing) of the sine wave. The discarded samples just beyond the ends are not supporting the interpolation of the sinewave. This is the result that is important to understand in the context of digital audio. Finally we are more or less recovering the signal from sparse samples - fewer than three per cycle (Fig. 7c).

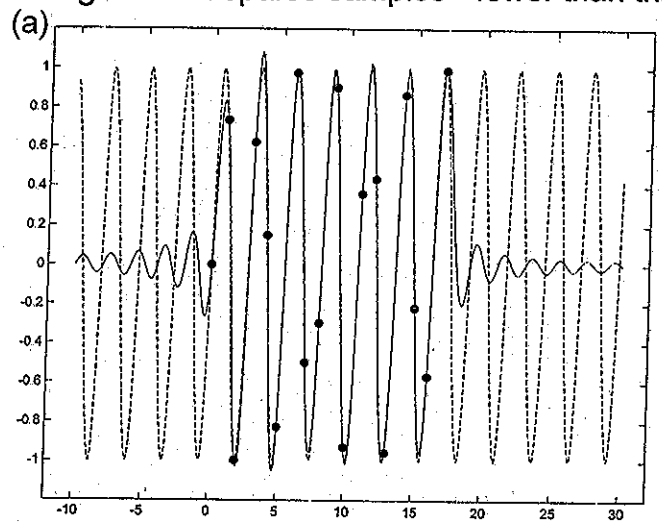
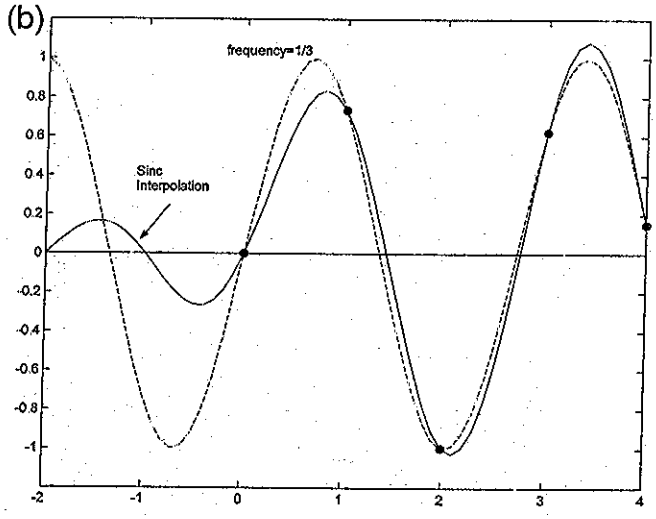
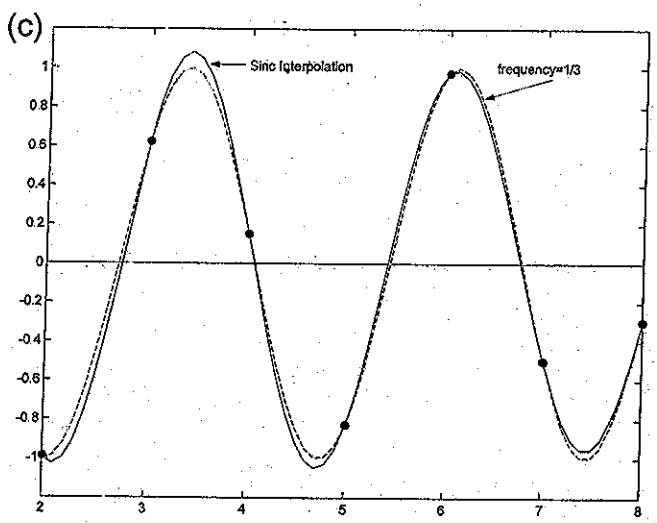


Fig. 7a Sinc interpolation of 18 samples of about a six-cycle "burst" of a sinewave is quite good near the middle of the "burst" (Fig. 7a, Fig. 7c) and not so good at the ends (Fig. 7b). The interpolated results are shown by solid lines – the sinewave by a dashed line.





REFERENCES:

[1] B. Hutchins, "Prediction, Deconvolution and System Identification," Electronotes, Vol. 17, No. 179, June 1992, pp3-29

[2] B. Hutchins, "Interpolating Samples with the DFT/FFT," Electronotes, Vol. 16, No. 172, July 1988, pp 3-34

[3] B. Hutchins, "Recovery with Sinc Interpolation," Section 2c of "Sampling Element," Electronotes, Vol. 21, No. 200, Dec. 2001, pp 12-19

* * * * * * * * * * * * * * * * *

ELECTRONOTES, Vol. 21, No. 204, August 2004
Published by B. Hutchins, 1016 Hanshaw Rd., Ithaca, NY 14850