Chapter 15

Roundoff Noise in FIR Digital Filters and in FFT Calculations

FIR (finite impulse response) digital filters are open loop signal processors. These are numerical filters, and roundoff takes place within these filters when certain numerical operations are performed upon the filter input signals. Roundoff also takes place in IIR (infinite impulse response) digital filters. These are feedback filters, and the effects of quantization in IIR filters are more complicated than in FIR filters. We will defer discussion of the IIR case until the next chapter. This chapter focuses on quantization noise in FIR filters. Furthermore, we assume here that the coefficients of the filter are infinitely precisely given. In digital implementation, this will not be true – the consequences of finite precision in the coefficients are discussed in Chapter 21.

15.1 THE FIR DIGITAL FILTER

An FIR filter is a "moving average" filter. This is diagrammed in Fig. 15.1 as a tapped delay line having N taps. The N filter coefficients or weight values are designated by $h(0), h(1), h(2), \ldots, h(N-1)$. They are the samples of the impulse response. The input signal is $x(m)$, and the output signal is $y(m)$. The output signal can be expressed as a linear combination of present and past inputs as follows:

$$\begin{aligned} y(k) &= h(0)x(k) + h(1)x(k-1) + \cdots + h(N-1)x(k-N+1) \\ &= \sum_{m=0}^{N-1} h(m)x(k-m)\,. \end{aligned} \tag{15.1}$$

Eq. (15.1) is a convolution sum. The output is the convolution of the input with the impulse response of the filter, and is a weighted linear combination of present and past input values.

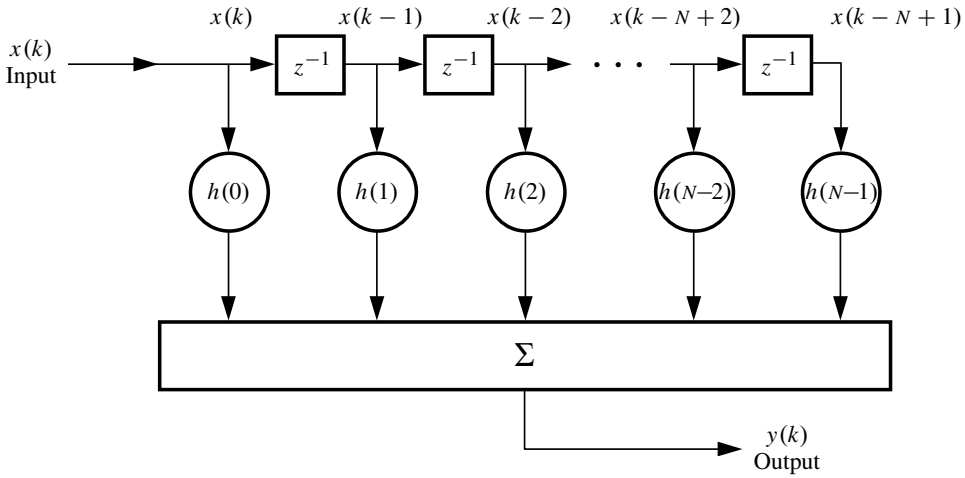

Figure 15.1 An FIR digital filter.

15.2 CALCULATION OF THE OUTPUT SIGNAL OF AN FIR FILTER

Calculation of the output signal $y(k)$ at the kth time instant involves the formation of a sum of products, as in Eq. (15.1). How these products are computed and summed varies from circumstance to circumstance. In some cases, each product would be formed, stored in memory, and then brought back to the accumulator one product at a time and added to the cumulative sum. In other cases, each product would be formed and added directly to the cumulative sum. To do the job this way, it would be necessary for the processor to have two separate accumulators, one used with the multiplication process to form the products, and the other used to form the sum of the products, or to be able to execute the "multiply-and-add" operation, that is, execute

$$\text{acc}(r+1) = \text{acc}(r) + h(r) \cdot x(k-r)\,. \tag{15.2}$$

in one command.

In general, the way the numerical operations would be done would depend on the processor architecture and its operating system and compiler, if everything were to be done with software. How the job would be done when using signal processing chips and associated software might be very different however.

It is quite normal for an accumulator to have more bits than the word length of the memory. When two numbers from memory are multiplied for example, the word length of the product could be the sum of the word lengths of the two numbers. If a product is to be formed and then put in memory, it would be necessary as a matter of routine to round the product so that its word length would be the same as that of the memory. The rounding is quantization and can be represented as such in the usual

way. Fig. 15.2 shows quantizers in the block diagram of the FIR filter performing the requisite rounding.

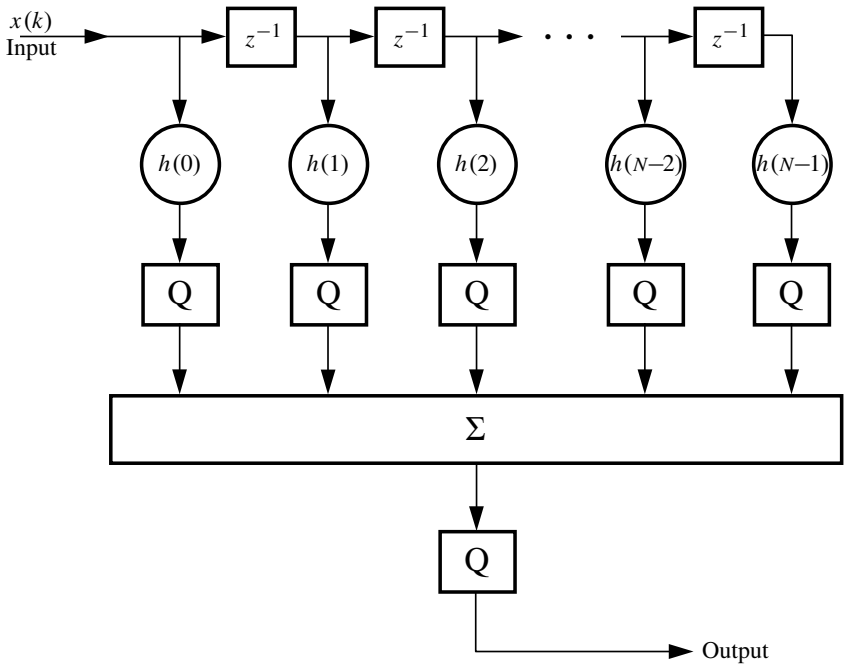

Figure 15.2 An FIR digital filter with rounding in the products and in the final sum.

There is an additional quantizer shown to round the sum. This is needed because, when many numbers are added, each having the word length of the memory, the sum could have a word length that may be considerably greater than that of the constituents of the sum. To be able to place the sum into memory, rounding would be essential.

The more efficient computation and the one that would generate much less quantization noise in the output signal would form the products one product at a time, and without going to memory, the product would be added directly to the accumulated sum. The processor with two accumulators (one to form the products, and the other to sum the products) would require software to make this usage possible. Only the accumulated sum would need to be rounded. The block diagram of the process is shown in Fig. 15.3.

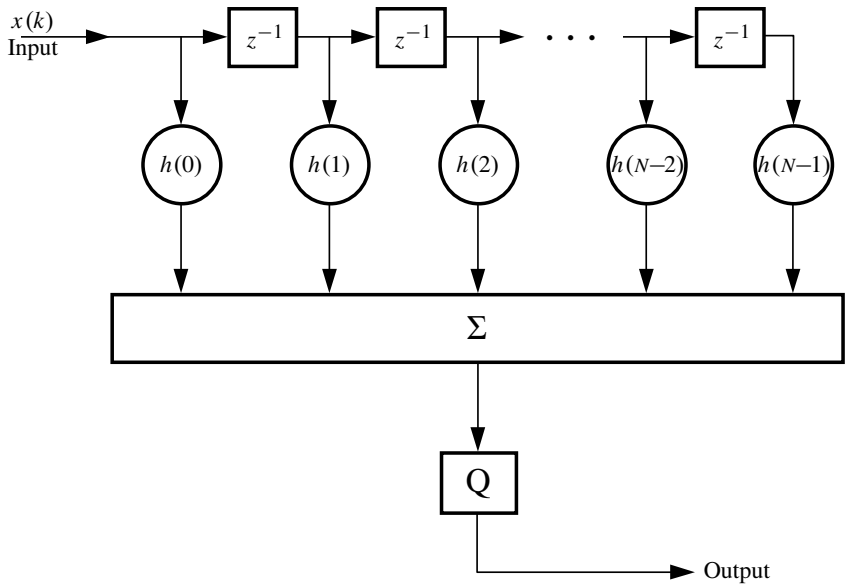

Figure 15.3 An FIR digital filter with rounding in the final sum.

15.3 PQN ANALYSIS OF ROUNDOFF NOISE AT THE OUTPUT OF AN FIR FILTER

Analysis of the quantization noise in the filter output signal can be generally accomplished by replacing the quantizers with sources of additive PQN. This assumes of course that an appropriate quantizing theorem such as QT II is satisfied. The numerical operations could be fixed-point, whereupon the rounding is represented by uniform quantization, or the numerical operations could be floating-point, whereupon the rounding is done by floating-point quantization.

The FIR filter containing rounding is truly nonlinear. There is no question about this. However, the behavior of the filter can be analyzed statistically, making use of linear theory, when the quantizing theorem is satisfied and the quantizers, for purposes of analysis, can be replaced by additive PQN. The analysis is quite precise. We are not approximating away the nonlinearities. The beauty of the method is that in almost all practical circumstances, the PQN model works with remarkably high accuracy. This is true with fixed-point computation and especially true with floating-point computation.

It is useful to consider some examples of FIR filtering with internal roundoff. Let us work with a 128-weight lowpass linear phase FIR filter. One such filter was designed with the Remez algorithm (Oppenheim, Schafer and Buck, 1998) by using Matlab. The specification called for flat 0 dB response from zero to one-tenth of the

sampling frequency, and zero response from two-tenths to one-half of the sampling frequency. The frequency response is shown in Fig. 15.4(a), and the phase response is shown in Fig. 15.4(b). The weight values are plotted in Fig. 15.5. This is the

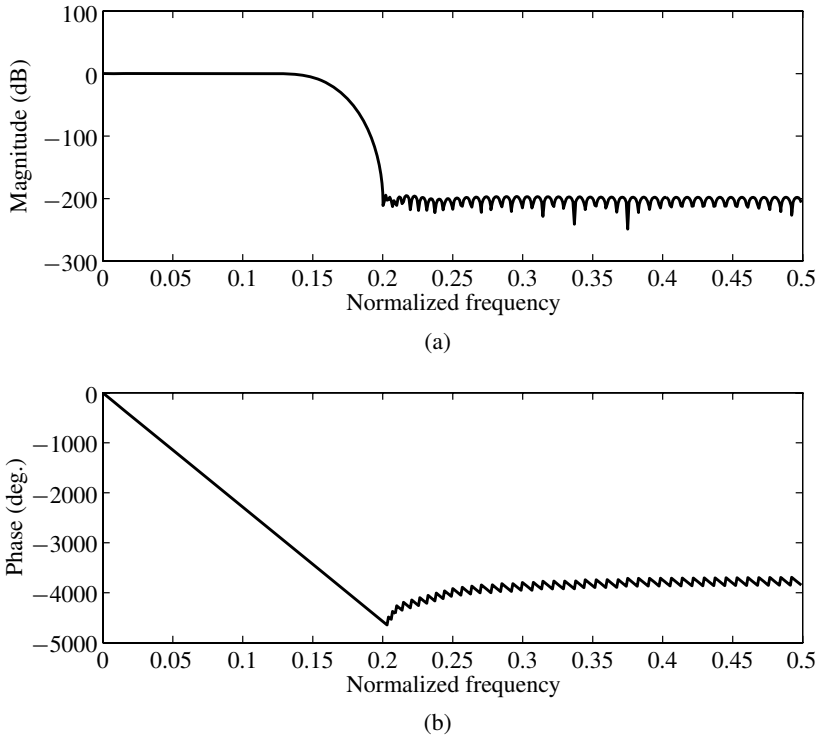

Figure 15.4 Frequency and phase response of a 128-weight FIR lowpass filter: (a) response magnitude; (b) phase.

filter impulse response. Since the filter is almost an ideal lowpass with linear phase corresponding to a delay of half the length of the filter, the impulse response should, to a good approximation, be a sinc function delayed by half the length of the filter. This result may be confirmed by inspection of Fig. 15.5.

15.4 ROUNDOFF NOISE WITH FIXED-POINT QUANTIZATION

The Matlab simulation of the above filter was done with IEEE double-precision floating-point number representation. As such, the filter was essentially perfect and free of quantization effects. Suppose now that the filter is to be actually implemented in real time using 32-bit fixed-point arithmetic. We shall let the signal samples be

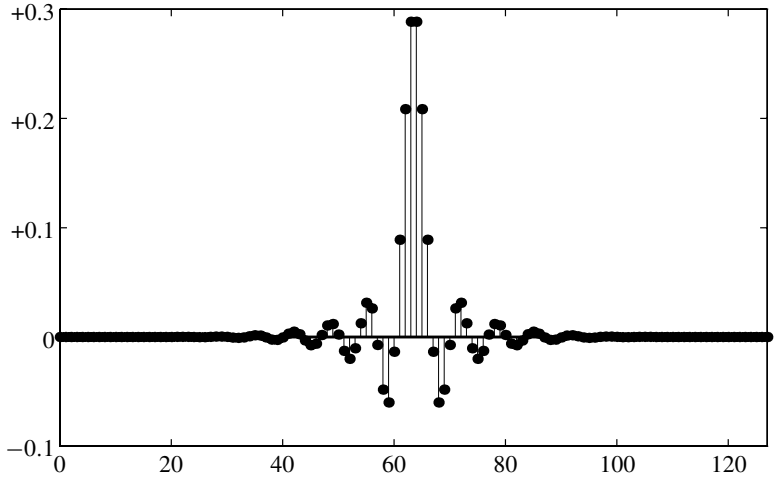

Figure 15.5 Impulse response of 128-weight FIR lowpass filter.

represented with "integer arithmetic." This means that all numbers are whole numbers, and that the binary point is therefore all the way to the right of the represented bits. The most left-hand bit is the sign bit, and the remaining 31 bits represent the magnitude of the number.

Let the filter input be a sinusoid, and let it have an amplitude which is full scale, i.e. 2^{31}. The peak-to-peak range of the sinusoid is therefore 2^{32}. Let the frequency of the sinusoid be within the filter's passband, for example at one-tenth of the sampling frequency.

Since the gain of the filter is 1 at this frequency (its coefficients are defined correspondingly, as fractional numbers), there will be no overload at the output, at least in steady state, when applying the sine wave at the filter input. Implementation of the summation would however need some care to avoid overflow during calculations.[1]

The input sinusoid may be generated as needed or it may be recorded in memory and called as needed. Whichever way this is done, the sinusoid must be quantized. If the filter calculations are to be done in accord with the diagram of Fig. 15.2, the entire process, including the quantization of the input sine wave, will take place as diagrammed in Fig. 15.6. The quantization step size everywhere is $q = 1$.

To analyze the effects of quantization noise at the filter output, the quantizers are replaced with sources of independent additive PQN as shown in Fig. 15.7. The justification for this comes from the fact that the amplitude of the sine wave is very large compared to the quantization step size. This makes the individual quantiza-

[1]The theoretical upper bound of the sum is $\max(|x(k)|) \cdot \sum_r |h(r)|$, thus the conservative scaling design is to keep this under 1.

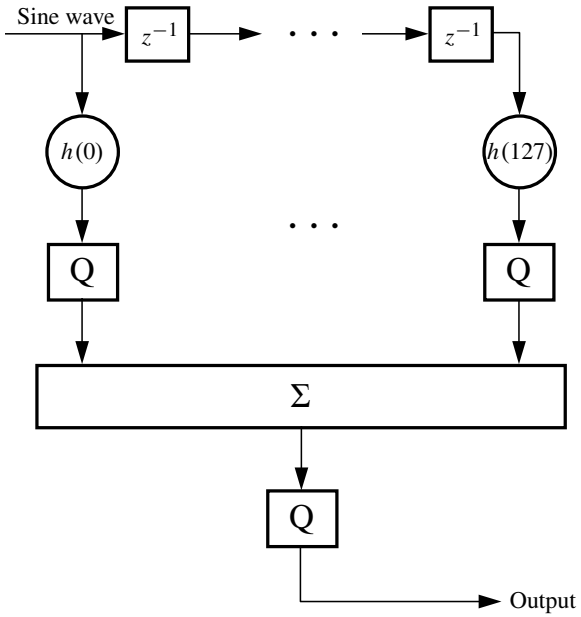

Figure 15.6 A 128-weight digital filter driven by a sine wave.

tion noises uniformly distributed, and uncorrelated with each other, even though the signals at the taps of the tapped delay line are correlated with each other.

The quantization noise at the filter output is the sum of all the quantization noise components. The mean square of the output quantization noise is the sum of the mean squares of the individual quantization noise components.

Referring to Figs. 15.6 and 15.7, the output quantization noise consists of noise from the final output quantizer, plus noise from quantization of 128 weighted signals, plus noise from quantization of the sinusoidal input. Since $q = 1$ everywhere, the output quantization noise power will be equal to 1/12 for the output quantizer, plus $\frac{1}{12}(128)$ for the quantization noise of the 128 weighted signals, plus the output noise power from quantization of the sinusoidal input. This noise originates at the input, and has a power level of 1/12 there.

The noise of the sinusoidal input propagates down the tapped delay line, and is weighted and summed before appearing at the filter output. This noise is uncorrelated from tap to tap of the tapped delay line. Therefore, the output noise power due to quantization of the sinusoidal input is

$$\frac{1}{12}\left(h^2(0) + h^2(1) + \cdots + h^2(127)\right) = \frac{1}{12}\begin{pmatrix}\text{sum of squares of the samples}\\ \text{of the impulse response}\end{pmatrix}. \tag{15.3}$$

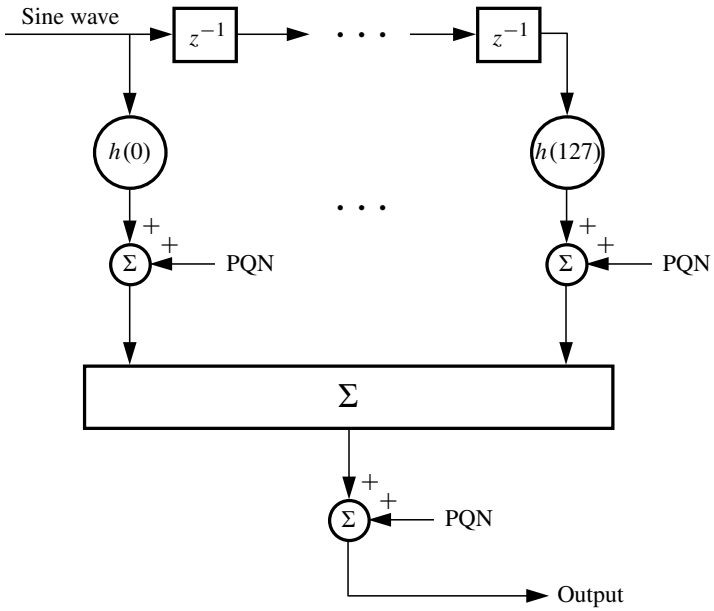

Figure 15.7 The 128-weight digital filter with quantizers replaced with sources of additive PQN.

The total output quantization noise is a sum of the three components. Its mean square is given by

$$\begin{pmatrix} \text{output} \\ \text{quantization} \\ \text{noise power} \end{pmatrix} = \frac{1}{12}\begin{pmatrix} \text{sum} \\ \text{of} \\ \text{squares} \end{pmatrix} + \frac{1}{12}(128) + \frac{1}{12}\cdot 1\,. \tag{15.4}$$

The sum of squares has been calculated for the impulse response of Fig. 15.5, and this is 0.2874. Therefore, the mean square of the output quantization noise is

$$\begin{aligned} \begin{pmatrix} \text{output} \\ \text{quantization} \\ \text{noise power} \end{pmatrix} &= \frac{1}{12}(0.2874) + \frac{1}{12}(129) \\ &= 10.77\,. \end{aligned} \tag{15.5}$$

Since the gain of the filter is unity, the sinusoidal output signal will have an amplitude of 2^{31}. The output signal power will be $\frac{1}{2}\left(2^{31}\right)^2 = 2.3\cdot 10^{18}$. Therefore, the output signal to noise ratio will be

$$\left(\text{output SNR}\right) = 10\log_{10}\frac{2.3\cdot 10^{18}}{10.77} = 173.3\,\text{dB}\,. \tag{15.6}$$

This very nice SNR is to be expected with 32-bit fixed-point computation with the signal amplitude at full scale. This result was checked and verified by Matlab simulation by comparing the outputs of the fixed-point filter with that of the double-precision floating-point filter.

The quantization noise would be even lower if the computation were done in accord with the diagram of Fig. 15.3 (multiply-and-add operation: negligible rounding after the multiplications). For this case,

$$\begin{aligned}\begin{pmatrix}\text{output}\\ \text{quantization}\\ \text{noise}\\ \text{power}\end{pmatrix} &= \frac{1}{12}\begin{pmatrix}\text{sum}\\ \text{of}\\ \text{squares}\end{pmatrix} + \frac{1}{12}\cdot 1\\ &= \frac{1}{12}(0.2874) + \frac{1}{12}\\ &= 0.107\,.\end{aligned} \tag{15.7}$$

The output signal-to-noise ratio for this case would be

$$\left(\text{output SNR}\right) = 10\log_{10}\frac{2.3\cdot 10^{18}}{0.107} = 193\,\text{dB}\,. \tag{15.8}$$

This result was also checked and verified by simulation.

15.5 ROUNDOFF NOISE WITH FLOATING-POINT QUANTIZATION

The arithmetic computations in the FIR filter could be done in floating-point. Suppose that we implement the 128-weight FIR filter with single-precision floating-point arithmetic ($p = 24$). We can now explore the workings of the filter with floating-point quantization.

The diagram of Fig. 15.7 can be used for output noise calculations. The output noise power due to quantization of the input sine wave is obtained by making use of Eq. (12.24). The noise power is proportional to the power of the sine wave. This output noise power is given by

$$\begin{aligned}\left(0.180\cdot 2^{-2p}\right)\frac{1}{2}\left(2^{31}\right)^2\cdot(\text{sum of squares}) &= \left(0.180\cdot 2^{-48}\right)\frac{1}{2}2^{62}(0.2874)\\ &= 423.8\,.\end{aligned} \tag{15.9}$$

The output noise power due to the output quantizer is proportional to the power of the input to this quantizer. The sinusoidal signal dominates, and since the filter gain is unity, its power is the same as at the filter input. Accordingly, the output quantization noise power due to the output quantizer is

$$\left(0.180 \cdot 2^{-48}\right) \frac{1}{2} \left(2^{31}\right)^2 = 1475 \,. \tag{15.10}$$

The output noise power due to the other 128 quantizers is proportional to the sum of their input powers. This sum is equal to the power of the input sine wave multiplied by the sum of squares of the samples of the impulse response. Referring again to Fig. 15.7, one can calculate the output noise power due to the 128 quantizers as

$$\left(0.180 \cdot 2^{-48}\right) \frac{1}{2} \left(2^{31}\right)^2 (\text{sum of squares}) = 423.8 \,. \tag{15.11}$$

The output noise power due to the 128 quantizers turns out to be equal to the output noise power due to the single quantizer that quantized the sinusoidal input.

The total quantization noise power at the filter output is obtained by adding Eq. (15.9), Eq. (15.10), and Eq. (15.11).

$$\begin{aligned} \begin{pmatrix} \text{total output} \\ \text{quantization} \\ \text{noise power} \end{pmatrix} &= 423.8 + 1475 + 423.8 \\ &= 2322.6 \,. \end{aligned} \tag{15.12}$$

The output signal to noise ratio is

$$\begin{aligned} (\text{output SNR}) &= 10 \log_{10} \left(\frac{\frac{1}{2}\left(2^{31}\right)^2}{2322.6} \right) \\ &= 150 \,\text{dB} \,. \end{aligned} \tag{15.13}$$

This is a poorer result than that obtained with 32-bit fixed-point arithmetic, given by Eq. (15.6). Floating-point computation does not always give a better result than fixed-point computation. In this example, the floating-point mantissa has only 24 bits, much less than the 32 bits of the fixed-point arithmetic. The missing 8 bits in the mantissa explain the difference of 23.3 dB:

$$20 \log_{10} 2^8 = 48.2 \,\text{dB} \gg 23.3 \,\text{dB} \,. \tag{15.14}$$

The floating-point SNR does not depend on signal amplitude, but the fixed-point SNR does depend on signal amplitude. Floating-point number representation causes all samples to have the same relative error. For the present example, the signal amplitude has been set at maximum value. For lower levels of signal, the fixed-point SNR is worse, and could be much worse than the floating-point SNR. Floating-point also allows a much greater signal dynamic range than fixed-point.

If the computations were done as illustrated in Fig. 15.3, the 128 quantizers would no longer be used and the total output quantization noise power would be the sum of Eq. (15.9) and Eq. (15.10). Accordingly,

$$\begin{pmatrix} \text{total output} \\ \text{quantization} \\ \text{noise power} \end{pmatrix} = 423.8 + 1475 = 1898.8\,. \tag{15.15}$$

The output signal to noise ratio would be

$$\begin{aligned} (\text{output SNR}) &= 10\log_{10}\frac{2.3\cdot 10^{18}}{1898.8} \\ &= 150.8\,\text{dB}\,. \end{aligned} \tag{15.16}$$

This result is still poorer than that obtained with 32-bit fixed-point computation, given by Eq. (15.8). But the reasoning is the same as for the previous case.

The above examples give us some idea of how quantization theory can be used to calculate roundoff noise at the output of FIR digital filters. We will extend these ideas further for the analysis of roundoff noise in FFT calculations.

15.6 ROUNDOFF NOISE IN DFT AND FFT CALCULATIONS

The discrete Fourier transform (DFT) has found wide application in all branches of science and technology. This became possible with the development of low-cost computing and with the advent of the Cooley–Tukey fast Fourier transform or FFT (Cooley and Tukey, 1965). The discrete Fourier transform of a sampled signal resembles the Fourier transform of the continuous signal, but differs from it for two reasons, (a) the DFT transforms a finite number of samples over a finite time, thus the original signal is "windowed" (the part to be processed is extracted from the signal) before transformation, (b) the DFT is the Fourier transform of samples rather than being the Fourier transform of the original continuous signal. Sampling and windowing have effects on the Fourier transform, and these effects are well understood in the literature. When using the DFT, one must take these into account. Good descriptions of the DFT and the various form of the FFT algorithm for calculating it are contained in textbooks on digital signal processing, like the book by Oppenheim, Schafer and Buck (1998).

Roundoff errors occur when the DFT is calculated. It is the purpose of this section to show how quantization noise originates and propagates into the output of the DFT calculation. There are many ways to calculate the DFT, and the method of computation has a significant effect on the amount of quantization noise that appears in the final output.

Calculation of the DFT by means of the FFT algorithm is equivalent to FIR filtering with complex weights. The methods for predicting output roundoff noise

resemble those for predicting the quantization noise power at the output of an FIR digital filter.

The discrete Fourier transform of a sequence of numbers $x(0), x(1), \ldots, x(N-1)$ is

$$X(m) = \sum_{k=0}^{N-1} x(k) e^{-j2\pi \frac{mk}{N}} . \tag{15.17}$$

This sequence could consist of samples of a time function. The members of the sequence are generally real, but they could also be complex.

The inverse discrete Fourier transform is

$$x(k) = \frac{1}{N} \sum_{m=0}^{N-1} X(m) e^{j2\pi \frac{mk}{N}} . \tag{15.18}$$

Equations (15.17) and (15.18) comprise a transform pair. Transforming any sequence $x(1), x(2), \ldots, x(N-1)$ yields a sequence $X(0), X(1), \ldots, X(N-1)$ that, when inverse transformed, yields the original sequence precisely. However, when these transformations are actually computed, roundoff error makes the results not quite perfect.

It is useful to make the following definition:

$$W_N \stackrel{\triangle}{=} e^{-j2\pi/N} . \tag{15.19}$$

Using this definition, the DFT and its inverse can be written as

$$\begin{aligned} X(m) &= \sum_{k=0}^{N-1} x(k) W_N^{mk} \\ x(k) &= \frac{1}{N} \sum_{m=0}^{N-1} X(m) W_N^{-mk} . \end{aligned} \tag{15.20}$$

If the sequence $x(1), x(2), \ldots, x(N-1)$ is a sequence of samples of a time function, then k is the time index or time sample number. The DFT takes us from the "time domain" to the "frequency domain" where m is the frequency number or frequency index.

The DFT can be calculated directly by making use of (15.20). Writing out the sum, we have

$$\begin{aligned} X(m) = \; & x(0) W_N^{m \cdot 0} \\ & + x(1) W_N^{m \cdot 1} \\ & + x(2) W_N^{m \cdot 2} \\ & \vdots \\ & + x(N-1) W_N^{m \cdot (N-1)} . \end{aligned} \tag{15.21}$$

For any frequency index m, the DFT is seen as a sum of products. Multiplication products have a number of digits approximately equal to the sum of the numbers of digits of the constituents. Therefore roundoff is necessary to prevent growth in the number of digits in each product.

15.6.1 Multiplication of Complex Numbers

Extra complication comes from the fact that this multiplication involves complex numbers. The first step is to study roundoff when forming the product of two complex numbers.

The product of two complex numbers is

$$(a + jb)(c + jd) = (ac - bd) + j(bc + ad)\,. \tag{15.22}$$

The real part of the product is $(ac - bd)$, and the imaginary part is $(bc + ad)$. Complex numbers cannot be directly represented, but their real and imaginary parts can be represented as two numbers.

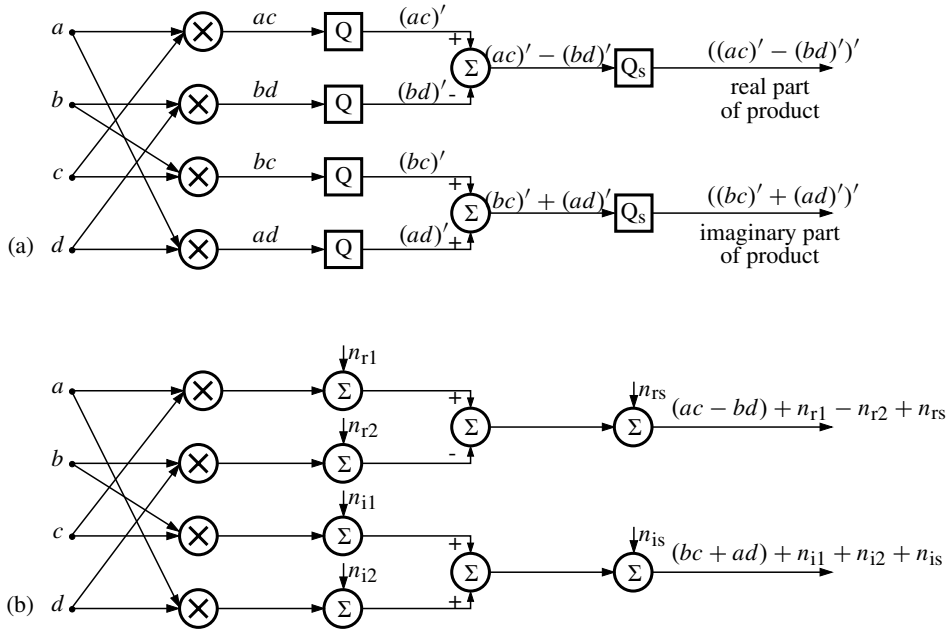

Figure 15.8 Block diagram of complex multiplication and quantization: (a) multiplication and quantization. The quantizers after the summers are denoted by Q_s to remind us that they quantize the sum of two quantized numbers, therefore modeling their effect by PQN is a rough approximation only. (b) representation of multiplication and addition using the PQN noise model; n_{r1}, n_{r2}, and n_{rs} are noises of the real parts, and n_{i1}, n_{i2}, and n_{is} are the noises of the imaginary parts. The noises are all uncorrelated with each other. Also, the real and imaginary parts of the total error of the result are uncorrelated with each other.

Fig. 15.8 is a block diagram showing how the complex product is formed and quantized within a digital signal processor or an arithmetic unit.

Fig. 15.8(a) illustrates the actual quantization of the real and imaginary parts of the product. The resulting quantization noises are usually assumed to be in accord with the (fixed-point or floating-point) PQN model. This is true for the quantizers after the multiplications, but not quite true after the additions, because here quantization of sums of already quantized numbers happens.

Figure 15.8(b) shows the general PQN model with the quantizers replaced by additive noises. The properties of the noises depend on the quantizer characteristic, which is determined by the number representation used in the digital signal processing algorithm. Therefore, this deserves some consideration.

15.6.2 Number Representations in Digital Signal Processing Algorithms, and Roundoff

In digital signal processing there are three different number representations that are important in practice.

- In *fixed-point number representation*, all numbers have the decimal point at the same position. Multiplication is scaled to assure that the products of the maximal numbers is still representable, so after multiplication there is no overflow. The number of bits in the result is almost double that of the individual numbers, so roundoff is necessary. PQN is closely approximated.
  Addition clearly deviates from this. It is an advantage that there is no roundoff error after addition, so the quantizers Q_s in Fig. 15.8(a) are not necessary, thus they *may be removed* (see Fig. 15.9). On the other hand, overflow can easily happen e.g. when adding two large-amplitude positive or two large-amplitude negative numbers. To avoid this, careful pre-scaling of the signal samples would be necessary.

- *Block floating-point number representation* (Oppenheim, Schafer and Buck, 1998) is a commonly used extension of fixed-point representation in DSPs (sometimes, e.g. when discussing the FFT, this is also referred to as fixed-point representation). Since downscaling of the numbers is often necessary to avoid overflow after additions/subtractions, a separate number is used as a common exponent to each block of numbers. Arithmetic operations are executed in fixed-point, and when overflow occurs, the exponent is increased by 1, and all the numbers are divided by 2, so that proper amplitude of the signal is maintained, but overflow is immediately eliminated. This solves the problem of overflow, but downscaling (division by 2) causes special roundoff. During downscaling, the resulting fractional parts of the numbers equal either zero or $0.5q$ (0.5 LSB). Thus, its distribution is binary, and depends on the input. The expected value is usually not zero: the generally applied truncation always decreases the numbers. This can be avoided if convergent rounding is applied: in

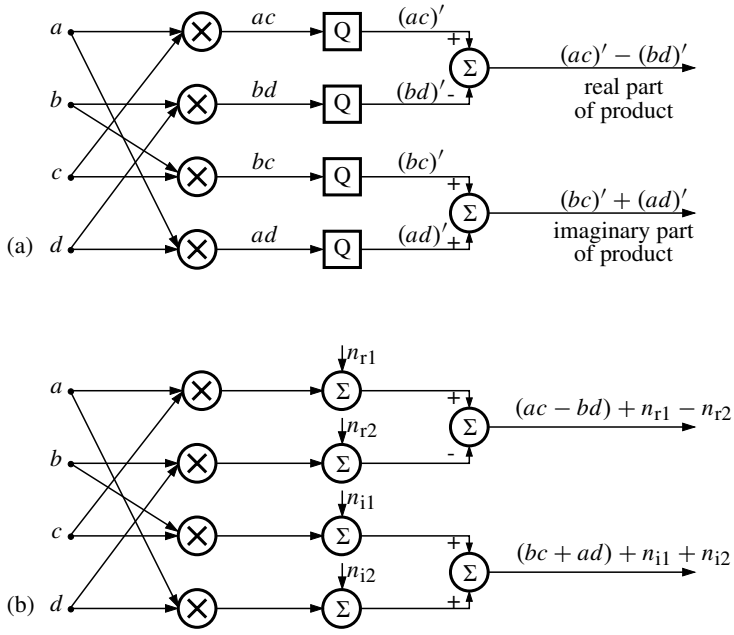

Figure 15.9 Block diagram of complex multiplication and quantization, implemented in fixed-point: (a) multiplication and quantization. After the summers there is no quantizer. (b) representation of multiplication and addition using the PQN noise model of quantization.

this rounding algorithm, 0.5 LSB is rounded always to the closest number *with zero last bit*, so that the roundoff errors of neighboring to-be-rounded numbers with half LSBs average out each other.

The variance of this error is about $(0.25q)^2 = q^2/16$ on average.

- In *floating-point number representation*, the quantization error of the product corresponds very well to floating-point PQN. After addition/subtraction, the properties of the quantization noise depend on the difference of the exponents, but on average, the floating-point PQN model is well applicable.

Roundoff errors in the DFT or in the FFT are very sensitive to implementation. The cause is explained below.

15.6.3 Growing of the Maximum Value in a Sequence Resulting from the DFT

A quick study of the transform pair (15.17) and (15.18) reveals an important asymmetry. The latter expression contains a multiplier $1/N$. This scale factor is necessary

to assure that after DFT, the original sequence can be properly retrieved by the inverse DFT. The output of the DFT must therefore contain larger numbers than the input sequence.

Indeed, if $\{x(k)\}$ is a random white sequence, it can be seen that the RMS value of the transformed sequence $X(m)$ is significantly larger than that of the original sequence:

$$\mathrm{RMS}\{X(m)\} = \sqrt{N} \cdot \mathrm{RMS}\{x(k)\}\,. \tag{15.23}$$

The increased RMS value usually means an increased maximum value of the sequence.

For a sine wave, the increase in the amplitude of the frequency domain sequence is even more important: if e.g. an integer number of periods of the sine wave of amplitude A is measured, at $m = Nf_1/f_s$ the Fourier transform of its samples is equal to

$$|X(m)| = \frac{N}{2} \cdot A\,. \tag{15.24}$$

This means that when evaluating the DFT, the maximum amplitude of the transformed series is significantly larger than that of the input. Therefore, in fixed-point implementation, overflow can only be avoided by preceding downscaling of the input signal, and in this case, the quantization errors are usually intolerable relative to the downscaled signal. This is why in practical implementations *block floating-point number representation* (see page 386) is used in fixed-point machines. If, during FFT, a sample of the calculated sequence would overflow, the whole sequence is scaled down (usually by 2), and the common exponent is increased by 1. Thus, the available precision is continuously utilized for the maximum-amplitude sample. This is very advantageous from the point of view of quantization errors, but analysis of the noise is more involved than with a fixed-point implementation.

15.7 A FIXED-POINT FFT ERROR ANALYSIS

In the following, we will analyze the FFT for fixed-point number representation. This case best corresponds to basic quantization theory. A few results concerning the more commonly used block floating-point and floating-point cases are given in Section 15.8.

15.7.1 Quantization Noise with Direct Calculation of the DFT

The *direct computation* of the DFT, in accord with Eq. (15.21), results in a sum of products having quantization noise in its real part equal to the sum of the quantization noise of all the real parts of the products, and likewise the quantization noise in its imaginary part is equal to the sum of the quantization noise of all the imaginary parts of the products. If all input samples are real, multiplications will have only the

half of the quantizations in Fig. 15.9 ($b = 0$, $n_{r2} = 0$, $n_{i1} = 0$). Since all noise components of the sums are uncorrelated with each other, it is clear for fixed-point that the real part of $X(m)$ will have roundoff noise with zero mean, a mean square of $N \cdot q^2/12$, and that the imaginary part of $X(m)$ will have roundoff noise with the same properties, and the real and imaginary noises will be mutually uncorrelated. This will be true for all frequency indices m.

15.7.2 Sources of Quantization Noise in the FFT

The quantization noise picture would be quite different from above if the DFT were calculated by means of the Cooley–Tukey FFT algorithm. There are a number of forms of this algorithm. The form that we will examine is called "decimation-in-time". This corresponds to the form originally given by Cooley and Tukey (1965). A flow graph illustrating the signal processing is shown in Fig. 15.10(a) for $N = 8$. Fig. 15.10(b) shows the basic "butterfly" structure.

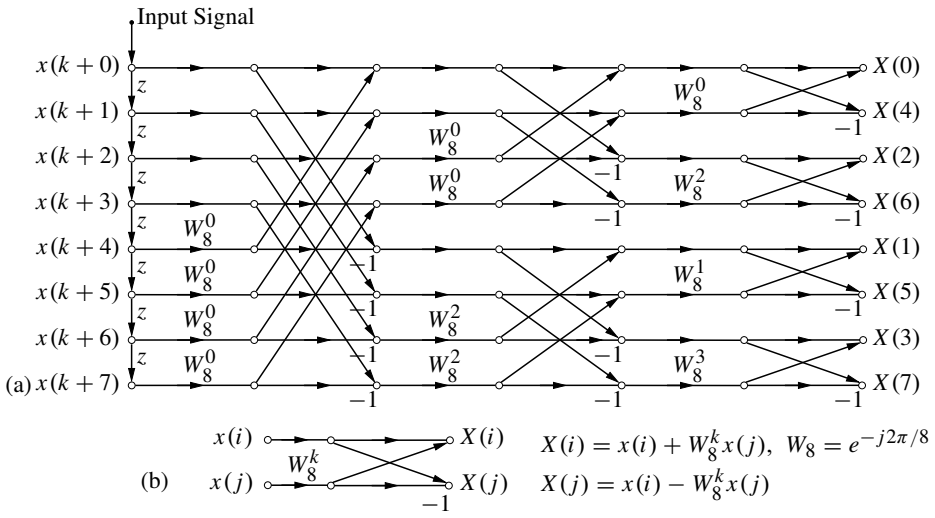

Figure 15.10 A flow graph showing the computation of the discrete Fourier transform, for $N = 8$, by means of the radix-2, decimation-in-time FFT algorithm, see (Oppenheim, Schafer and Buck, 1998, Fig. 9.14, page 645). Each branch has a unity transfer function unless otherwise labeled: (a) full flow graph, (b) general form of the DIT butterfly.

The input is assumed to consist of samples of a time function. These samples are $x(m)$, $x(m + 1)$,... $x(m + 7)$. The input samples are formed from $x(m)$ (mathematically) by time advances, represented by z in the diagram. The outputs are the transforms of the inputs. Each is a transform at a specific frequency. The frequencies

turn out to be not in usual order from 0, 1, 2, . . . , 7, but are instead in "bit-reversed" order[2] as 0, 4, 2, 6, 1, 5, 3, 7.

After execution, each of the outputs is a complex number corresponding to the DFT at the designated frequency of the sequence of numbers within the "time window", i.e. the sequence of input numbers $x(k)$, $x(k+1)$,... $x(k+7)$. The frequency resolution of this transform, the spacing in frequency between $X(5)$ and $X(6)$ for example, is equal to the sampling frequency divided by N.

The calculation of the DFT with the FFT algorithm of Fig. 15.10 will involve quantization. Every branch of the flow graph of Fig. 15.10 has a gain of unity, except for the branches labeled W_8^1, W_8^2, W_8^3, and -1. The branches carry complex signals. The summation points within the flow graph add complex signals, keeping separate their real and imaginary parts.

When a complex signal goes through a branch of W_8 raised to a power, the result is multiplication by W_8 to the power. This multiplication leaves the magnitude unchanged since the magnitude of W_8 raised to any power is unity, but the phase angle of the complex signal is rotated, so products are evaluated. Multiplication by W_8 raised to a power necessitates quantization in order to keep the number of digits in the product from growing. Therefore, during each multiplication by W_8 to a power, quantization must occur, as illustrated in Fig. 15.9(a). A flow graph illustrating this is given in Fig. 15.11. A special "quantized complex multiplication"

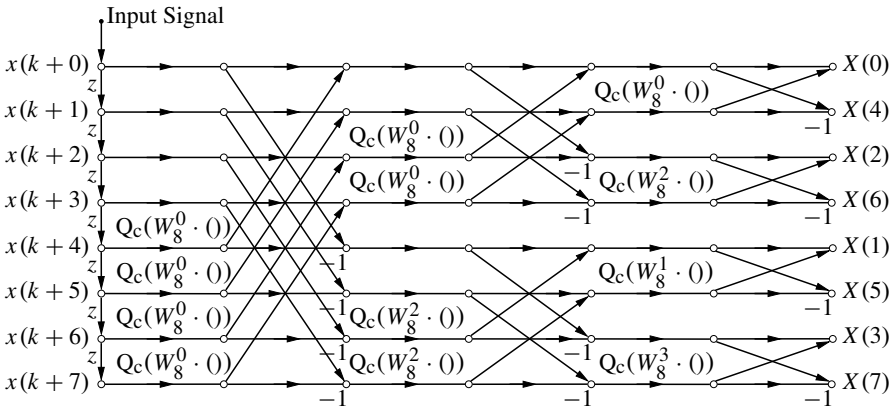

Figure 15.11 A flow graph of the FFT algorithm showing quantization after multiplication by W_N raised to a power. For this flow graph, $N = 8$. The symbol $\mathrm{Q_c}\left(W_8^m \cdot (\,)\right)$ denotes "quantized complex multiplication": complex multiplication executed in fixed-point, corresponding to Fig. 15.9(a).

symbol, $\mathrm{Q_c}\left(W_8^m \cdot (\,)\right)$, denotes each complex multiplication executed in fixed-point,

[2]By counting in binary form 0–7 and reversing the order of the bits, the resulting numbers would be in bit-reversed order.

corresponding to Fig. 15.9(a). The outputs of the butterflies will be sums of complex numbers, and these will not be quantized in fixed-point.

The input signal $x(m)$ will be assumed to be already quantized. In theory, the outputs of each of the branches labeled W_8^m will be quantized. Since the branches W_8^0 merely multiply the signals by 1, and the inputs to the branches are already quantized, these branches introduce no quantization noise, therefore quantizers can be removed from here.

For the next layer of butterflies, quantizers in the branches labeled W_8^0 can be removed. Those labeled by W_8^2 can also be removed, because multiplication by W_8^2 is equivalent to rotation of the complex signal by 90°, which simply interchanges the real and the imaginary parts. Since these parts are already quantized, no further quantization is necessary.

For the next layer of quantizers, the third and final layer, the quantizers in branches labeled W_8^0 and W_8^2can be removed. However, none of the quantizers in the branches labeled W_8^1 and W_8^3 can be removed because the intermediate products are not already quantized. The branch labeled W_N^1 rotates its complex signal by 45°, so that this signal that was already quantized is now no longer so. The other branch W_8^3 rotates the complex signals by odd multiples of 45° and the result needs to be quantized.

Figure 15.12 shows the flow graph with the redundant quantizers removed for fixed-point, and the remaining quantizers replaced by PQN (the complex-valued noises n_{33} and n_{34}). The two remaining quantizers introduce quantization noise in

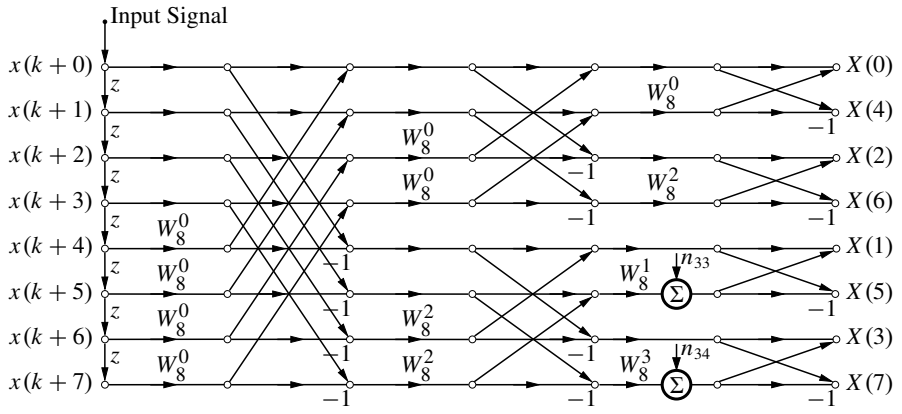

Figure 15.12 A flow graph of the FFT algorithm for $N = 8$ with its quantizers redundant for fixed-point removed, and the remaining quantizers modeled by PQN.

accord with the PQN model.

It will be our next task to evaluate the quantization noise of the FFT in general, for values of N larger than 8. A flow graph like that of Fig. 15.12 could be constructed for $N = 16$. This has been done, and the result is shown in Fig. 15.13. The

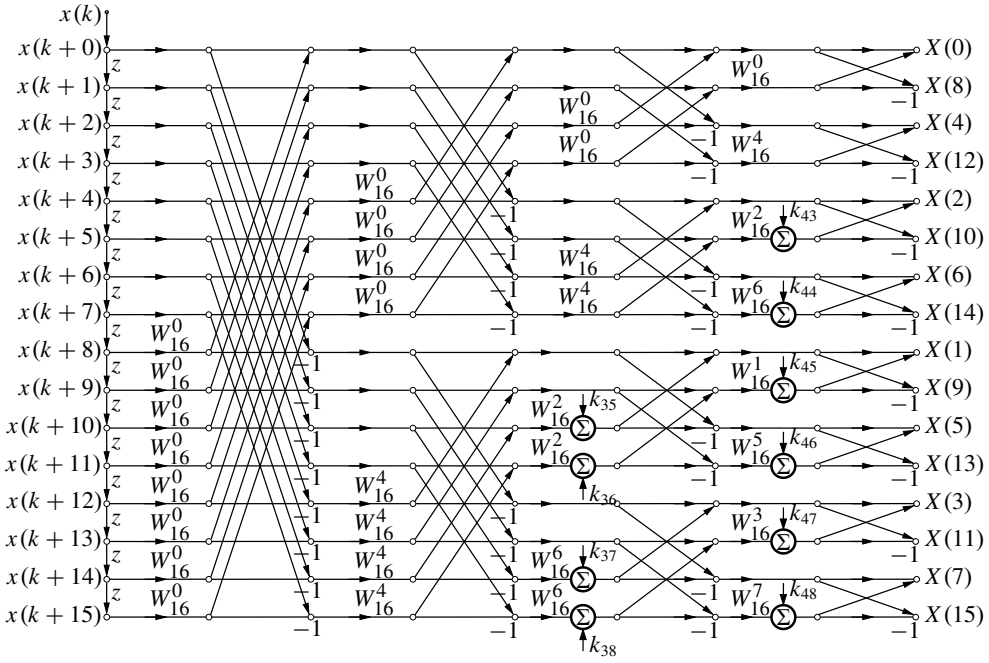

Figure 15.13 A flow graph of the FFT algorithm for $N = 16$. Quantizers redundant for fixed-point have been removed, and the remaining quantizers modeled by PQN.

redundant quantizers have been removed. From this flow graph, it will be possible to induce the general pattern of quantization noise in the FFT output for any value of N of interest.

15.7.3 FFT with Fixed-Point Number Representation

Quantization noise having power $2q^2/12$ for both real and imaginary parts is generated by all PQN models. This is due to multiplication by the W_N^k terms, e.g. for $N = 16, k = 1, 2, 3, 5, 6, 7$.

From the flow graphs of Figs. 15.13 and 15.12, one can see that not all of the FFT outputs have the same amount of quantization noise. For example, for $N = 16$, at zero frequency ($X(0)$), and at half of the sampling frequency ($X(8)$), there is no quantization noise produced by the last butterfly. At the highest frequency (or, in other words, the smallest absolute-value negative frequency), $X(15)$ has quantization noise $2 \cdot 2q^2/12$ caused by two PQN's along the way. The output $X(14)$ goes through one layer of quantization.

The output $X(N-1)$ will go through three layers of quantization for $N = 32$, and so forth. The number of such layers is $(\log_2 N) - 2$. All of these layers contribute $2q^2/12$ each in quantization noise power.

In general, the power of the quantization noise present in both the real and imaginary parts of the highest frequency output will be

$$\left(\begin{array}{c}\text{quantization noise power}\\ \text{in real and in imaginary}\\ \text{parts of output } X(N/2)\end{array}\right) = \frac{2q^2}{12}\left(\log_2 N - 2\right) . \tag{15.25}$$

For other values of m, the quantization noise will be somewhat less, depending on the actual value of m. For any value of N, the quantization noise power in the FFT outputs at all frequencies can be computed. It can be uniformly bounded as

$$0 \le \left(\begin{array}{c}\text{quantization noise power}\\ \text{in real and in imaginary}\\ \text{parts of all FFT outputs}\end{array}\right) \le \frac{2q^2}{12}\left(\log_2 N - 2\right) . \tag{15.26}$$

One may recall that when the DFT is computed directly in fixed-point, the quantization noise power in the real and in the imaginary parts of the DFT outputs at all frequencies is

$$\left(\begin{array}{c}\text{quantization noise power}\\ \text{in real and in imaginary}\\ \text{parts of all DFT outputs}\end{array}\right) = N\frac{q^2}{12} . \tag{15.27}$$

A great advantage of the FFT algorithm over direct computation is that the FFT algorithm requires much less computation when N is large. Another advantage is that the computed results have much less quantization noise with the FFT algorithm. It is apparent that direct calculation of the DFT introduces more quantization noise than does the FFT algorithm. For example, with $N = 1024$, direct DFT computation has real and imaginary outputs having quantization noise power of $1024q^2/12$, in accord with Eq. (15.27). Comparing this with the worst-case quantization noise power for the FFT, which is $16q^2/12$ in accord with the bound (15.26), the advantage of the FFT algorithm is apparent.

In general, the roundoff error also depends on the actual circumstances: the type and properties of the signal, the number representation, the structure and length of the FFT, the rounding algorithm, rounding of fractions of 0.5 LSB. Without knowing all these, one cannot make general statements about the output quantization noise. Therefore, in the literature, formulas for specific signals and specific cases are given. We briefly deal with the two most important number representations in the next section.

15.8 SOME NOISE ANALYSIS RESULTS FOR BLOCK FLOATING-POINT AND FLOATING-POINT FFT

In this section, we briefly discuss some general results concerning FFTs executed with block floating-point and with floating-point number representations. More details about these representations can be found in the literature.

15.8.1 FFT with Block Floating-Point Number Representation

Perhaps the best-known error analysis of the widely used block floating-point FFT (see page 386) was done by Welch (1969), followed by Kaneko and Liu (1970). The main result of Welch is an upper bound for the noise-to-signal ratio of the FFT result (rewritten in our notation):

$$\frac{\mathrm{RMS}\{N_X\}}{\mathrm{RMS}\{X\}} < 2^{\frac{M+3}{2}} 2^{-(B-1)} \frac{C}{\mathrm{RMS}\{x\}} \approx \sqrt{N} \cdot 2^{-B} C_\mathrm{s}\,, \tag{15.28}$$

where $\{X\}$ is the FFT result, $\{N_X\}$ is the quantization noise of $\{X\}$, $M = \log_2 N$, B is the number of bits (including the sign bit),[3] C and C_s are signal-dependent constants, $C \approx 0.4$, $C_\mathrm{s} \approx 7.5$, and $\{x\}$ denotes the input time series, scaled to full number representation range (therefore, in all his examples, $\mathrm{RMS}\{x\} \approx 0.3$). For signals which require less than $\log_2(N) - 1$ downscalings, the constants can be decreased to one half or one third of the above values, but we do not have space here to go into these details. The upper bound is generally applicable as it is.

This formula can be used for the estimation of the total power of the roundoff noise. Assuming that the roundoff error is white, its power can be given at each frequency point.

In order to have an overall impression of the resulting noise, consider that with number representation with $B = 16$ bits (15 bits + sign) the dynamic range (the ratio of the sine waves that can be analyzed jointly) is limited to approximately 75 dB, see Exercise 15.5.

One might also consider the use of dither in the digital processor. A few details will be discussed in Subsection 19.6.7 of Chapter 19.

15.8.2 FFT with Floating-Point Number Representation

If the FFT calculation were done with floating-point arithmetic, then all of the quantizers pictured in Fig. 15.11 would be floating-point quantizers. Moreover, additions would also need to be followed by quantizers. Not all of these quantizers would produce equal quantization noise powers. These noise powers would depend on the

[3] Welch denoted by B the bit length *without* the sign, we have changed this to stay in conformity with our notation.

signal power levels at the inputs to the quantizers. These power levels could be computed if one knew the inputs $x(n)$, $x(n+1)$,..., $x(n+N-1)$, and then the output quantization noises could be computed.

An interesting result was described by Weinstein (1969), which can be explained on the basis of the following ideas. We saw in Chapter 12 that floating-point quantization generates quantization noise that is *proportional* to the quantizer input. Therefore, the generated noise-to-input ratio is constant at the quantizers associated with each non-trivial multiplication. We can assume that after multiplications at each level, the noise variance, *relative* to the square of a sample, is increased by an additive constant. Therefore, at the FFT output the noise-to-signal ratio is proportional to the number of such floating-point quantizations.

However, addition requires more thought. Here are two examples to illustrate what happens.

Example 15.1 Noise Propagation when Adding Two Deterministic Floating-Point Numbers

Let us assume that we add two numbers with the same sign. The two numbers are 2.5 and 1, each corrupted by previous roundoff noise, that is, they are multiplied by $(1+\epsilon_1)$, and $(1+\epsilon_2)$, respectively, with ϵ_1 and ϵ_2 having zero mean and variance $\sigma_1^2 = \sigma_2^2 = \sigma^2 = 2^{-2p}/3$. The sum is 3.5. The variance of the noise after addition and roundoff is

$$\sigma_r^2 = 2.5^2 \cdot \sigma_1^2 + 1^2 \cdot \sigma_2^2 + 3.5^2 \cdot 2^{-2p}/3 = 19.5 \cdot 2^{-2p}/3\,, \tag{15.29}$$

that is, the result is $3.5(1+1.26\epsilon_r)$, where ϵ_r has variance $2^{-2p}/3$. This means that in this case, even with roundoff after summation, the relative variance increases from input relative variance σ^2 only to $1.26^2\sigma^2$, that is, only by a factor of $1.26^2 = 1.6$. In general, the increase in the relative variance will depend on the numbers to be added.

Example 15.2 Noise Propagation when Adding Two Random Numbers, Represented in Floating-Point

Let us assume that x_1 is a sample of a random variable with zero mean and variance σ_1^2, modified by additive roundoff noise with zero mean and variance σ_{1n}^2. Similarly, x_2 is a sample of a random variable with zero mean and variance σ_2^2, modified by additive roundoff noise with zero mean and variance σ_{2n}^2. Furthermore, let us assume that

$$\frac{\sigma_{1n}^2}{\sigma_1^2} < \frac{\sigma_{2n}^2}{\sigma_2^2}\,.$$

When x_1 and x_2 are added or subtracted, the variances add up, and the noise-to-signal ratio is between the previous ones:

$$\frac{\sigma_{1n}^2}{\sigma_1^2} < \frac{\sigma_{1n}^2 + \sigma_{2n}^2}{\sigma_1^2 + \sigma_2^2} < \frac{\sigma_{2n}^2}{\sigma_2^2}\,. \tag{15.30}$$

After the execution of the operation, the result is rounded, thus slightly increasing the noise variance:

$$\mathrm{NSR}_{\mathrm{result}} = \frac{\sigma_{1n}^2 + \sigma_{2n}^2 + \sigma_{\mathrm{roundoff}}^2}{\sigma_1^2 + \sigma_2^2} \,. \tag{15.31}$$

The NSR slightly increases, depending on the magnitude of the result.

As the above examples show, the variance of the result of a combination of multiplications and additions is very difficult to predict. In certain cases, some kind of *average* variance can be given. This was done for the FFT.

Based on the above ideas, Weinstein (1969) gave the following formula for the noise-to-signal ratio at the FFT output:

$$\frac{\begin{pmatrix}\text{output noise}\\ \text{variance}\\ \text{at index } n\end{pmatrix}}{|X(n)|^2} \approx 0.42 \cdot 2^{-2p} \log_2(N) \,. \tag{15.32}$$

This expression may only be used when the signal value $X(n)$ is not very small: for example, $|X(n)| > \log_2(N) \cdot \mathit{realmin}$, where *realmin* is the minimum normalized number which can be represented.

Concerning this result, Weinstein made an important remark. Verification through simulations was possible if for input numbers at equal distance from two representable numbers, rounding was done randomly upwards or downwards. Truncation in these cases caused the right side of (15.32) to grow with N quicker than the indicated $\log_2 N$ factor. This warns us about the non-negligible probability of results at the midpoint between the two closest representable numbers before roundoff. A solution to this problem is *convergent rounding*. This is deterministic, but still "random-like": rounding these numbers to their closest neighbor *containing 0 at the LSB position* effectively breaks up the pattern of these roundoffs.

It is of interest to compare this noise level to the noise of the directly calculated DFT, as we did also for fixed-point. One may argue that each DFT point is obtained after N multiplications and $N - 1$ additions, therefore the variance of the error of the DFT is proportional to N, that is, it is much larger than that of the FFT. This is true, but surprisingly, for floating-point the total error can be effectively diminished by applying a "tree-like" grouping method for summation: that is, as Weinstein says, "the N products were summed in pairs, the $N/2$ results summed in pairs, and so on. Then the bound one would derive for the output mean-squared noise-to-signal ratio would have the same dependence on N as the corresponding bound for the FFT. A tree-like summation technique thus makes the accuracy for the direct Fourier transform essentially the same as for the FFT."

15.9 SUMMARY

FIR filters are moving average filters having finite impulse responses. Feedback is not used in such filters.

Roundoff takes place internally in FIR filters after addition and multiplication operations in order to keep the number of bits representing signals in the filter from growing. Whenever quantization occurs, additive noise is inserted into the signal. Assuming that the PQN model is justified, the quantization noise power at the filter output may be calculated.

Each quantizer may be replaced by a source of additive independent white noise. The total output noise power is the sum of the individual output noise powers. Tracing the signal path from each quantizer to the filter output, the individual output noise powers may be calculated as the power of the quantizer's noise multiplied by the sum of squares of the impulse response of the path. These calculations can be made for both fixed-point and floating-point computation.

Noise analyses have been made for FIR filters with several different computational configurations. Also, noise analyses for fixed-point computation of the discrete Fourier transform, performed directly or by means of the fast Fourier transform, are presented in this chapter. Results are obtained for specific FFT configurations. The reader would need to customize these results for other FFT configurations.

Sometimes FIR filters or FFT algorithms are to be implemented with commercial software, and details of the arithmetic algorithms are not available. Calculation of output quantization noise by the methods of this chapter would therefore not be possible. To calculate this noise, the FIR filter or FFT algorithm could be simulated with Matlab in double precision, as required to get a "perfect" solution compared to that of the commercial software. The difference in their outputs is the quantization noise. Keep in mind that if the commercial software is a floating-point implementation, then the quantization noise power will be proportional to the input signal power. If the commercial software is a fixed-point implementation, then the quantization noise power will be fixed and independent of the input signal power.

15.10 EXERCISES

15.1 A sine wave of unknown frequency is sampled at the rate $f = 1\,\text{kHz}$. Inspecting the DFT, two peaks can be seen, one of them at 113 Hz. Where is the other peak, and what may be the frequency of the sine wave?

15.2 We have a 100-tap FIR filter. Given the coefficients $h(k)$, $k = 1, 2, \ldots, 100$, and assume direct evaluation (summing the terms $h(k)x(k_0 - k)$).

(**a**) Determine the theoretical upper bound of the roundoff error for 16-bit fixed-point arithmetic.

(**b**) Calculate also the standard deviation of the error, using the PQN model.

(**c**) Let all coefficients of the filter be equal so that $h(k) = 1/100$ for all k (moving averaging filter). Calculate the bound and the standard deviation of the roundoff error.

15.3 In the butterfly usually implemented in a decimation-in-time FFT algorithm (see page 389), we execute the following operations:

$$\begin{aligned} X_{l+1}(i) &= X_l(i) + X_l(j)W_N^k \\ X_{l+1}(j) &= X_l(i) + X_l(j)W_N^{k+N/2} = X_l(i) - X_l(j)W_N^k \,, \end{aligned} \tag{E15.3.1}$$

with N the number of samples, and $W_N = e^{-j2\pi/N}$. $X_l(i)$ and $X_l(j)$ are complex numbers.

(**a**) How many times larger can the maximum magnitude of the real and imaginary parts of $X_{l+1}(i)$ be than the maximum magnitude calculated for the real and imaginary parts of the two input variables?

(**b**) As a consequence, how many downscalings by 2 may be necessary in one stage of the FFT to avoid overflow with fixed-point number representation?

(**c**) What is the total number of necessary downscalings?

15.4 In the butterfly usually implemented in a decimation-in-frequency[4] FFT algorithm, we execute the following operations:

$$\begin{aligned} X_{l+1}(i) &= (X_l(i) + X_l(j))W_N^k \\ X_{l+1}(j) &= (X_l(i) - X_l(j))W_N^k \,, \end{aligned} \tag{E15.4.1}$$

with N the number of samples, and $W_N = e^{-j2\pi/N}$. Answer the same questions as in Exercise 15.3.

15.5 The dynamic range of a spectrum analyzer is defined as the power ratio of the maximal input sine wave and the minimal detectable input sine wave:

$$\mathrm{DR} = 10 \cdot \log_{10} \frac{A_{\max}^2/2}{A_{\min}^2/2} \,. \tag{E15.5.1}$$

Let detectability mean that the spectral peak belonging to the sine wave to be detected in the periodogram:

$$P(m) = \frac{1}{N} \left| \sum_{k=0}^{N-1} x(k) e^{-j2\pi \frac{km}{N}} \right|^2 \tag{E15.5.2}$$

is higher than twice the maximal peak belonging to the quantization noise spectrum. Since the lines corresponding to quantization noise are considered as random variables, for the calculations we take the 99.5% value of their distribution as their maximum value. Assume further that sampling is coherent, that is, the DFT of the samples of the sine wave contain just two digital Dirac delta functions (that is, there is no leakage involved).

[4] See e.g. Oppenheim, Schafer and Buck (1998).

(**a**) Determine the dynamic range as determined by the roundoff error in the FFT, using the expression (15.28) for $B = 16$ bits (including the sign bit), and $N = 512$. The spectrum of the roundoff error may be assumed to be white. (Note that this assumption is not quite justified for large-amplitude sine waves with negligible noise, but it may be used to obtain good approximate results.)

(**b**) What is the maximum dynamic range determined by the quantization noise of the input ADC, assuming 12-bit resolution?

15.6 Verify the theoretical result (15.26) for the fixed-point FFT (with no block-floating scaling here), by Matlab-based computer simulation, using the tools available from the web page of this book.[5] Let $N = 256$, the number representation have $B = 32$ bits to represent numbers in $(-1, 1)$, and the input be a zero-mean white sequence uniformly distributed in $(-1, 1)$. In order to have a reference transform result, assume that the error of Matlab's `fft` command may be neglected. When determining the roundoff error,

(**a**) do not include either the effect of input quantization, or that of quantization of the trigonometric coefficients,

(**b**) include also the effect of input quantization, and of quantization of the trigonometric coefficients. Is the roundoff error much larger than without these roundoffs?

15.7 Verify the theoretical result (15.27) for fixed-point DFT with no block-floating scaling, by Matlab-based computer simulation, using the tools available from the web page of this book. Let $N = 256$, the number representation use $B = 32$ bits to represent numbers in $(-1, 1)$, and the input be a zero-mean white sequence uniformly distributed in $(-1, 1)$. In order to have a reference transform result, assume that the error of Matlab's `fft` command may be neglected. When determining the roundoff error,

(**a**) do not include either the effect of input quantization, or that of quantization of the trigonometric coefficients,

(**b**) include also the effect of input quantization, and of quantization of the trigonometric coefficients. Is the roundoff error much larger than without these roundoffs?

15.8 Check by computer simulation if grouped execution (see the end of Section 15.8.2) of the fixed-point DFT decreases the roundoff error obtained in Excercise 15.7. Explain why the technique useful for floating-point is not helpful for fixed-point.

15.9 Verify the result (15.28) given for block-float FFT, by Matlab-based computer simulation, using the tools available from the web page of this book. Let $N = 256$, the number representation use $B = 16$ bits to represent numbers in $(-1, 1)$, and the input be a zero-mean white sequence uniformly distributed in $(-1, 1)$. In order to have a reference transform result, assume that the error of Matlab's `fft` command may be neglected. When determining the roundoff error,

[5] `http://www.mit.bme.hu/books/quantization/`

(**a**) do not include either the effect of input quantization, or that of quantization of the trigonometric coefficients,

(**b**) include also the effect of input quantization, and of quantization of the trigonometric coefficients. Is the roundoff error much larger than without these roundoffs?

15.10 Verify the result (15.32) given for floating-point FFT, by Matlab-based computer simulation with $p = 32$, using the tools available from the web page of this book. Let $N = 256$, and the input be a zero-mean white sequence uniformly distributed in $(-1, 1)$. In order to have a reference transform result, assume that the error of Matlab's `fft` command may be neglected. When determining the roundoff error,

(**a**) do not include either the effect of input quantization, or that of quantization of the trigonometric coefficients,

(**b**) include also the effect of input quantization, and of quantization of the trigonometric coefficients. Is the roundoff error much larger than without these roundoffs?

15.11 Check the error of Matlab's built-in `fft` command, by Matlab-based computer simulation, using the tools available from the web page of this book. Check the result against (15.32). Let $N = 256$, and the input be a zero-mean white sequence uniformly distributed in $(-1, 1)$. As reference, use $p > 56$. When determining the roundoff error,

(**a**) do not include either the effect of input quantization, or that of quantization of the trigonometric coefficients,

(**b**) include also the effect of input quantization, and of quantization of the trigonometric coefficients. Is the roundoff error much larger than without these roundoffs?

15.12 Evaluate experimentally the RMS value of the error of the FFT in Matlab for $N = 256$, with a white Gaussian input, with the help of the inverse FFT. The idea is to transform and transform back again. It is assumed that the MS error of the transform is half the total MS error. Compare the RMS error with the RMS error of the FFT with a sinusoidal input.

Do the results significantly change when N, which is a power of 2 now, is changed to $N + 2$? Why?

15.13 Calculate the dynamic range (that is, the power ratio of the maximal measurable sine wave and the minimum amplitude detectable sine wave) for a Fourier analyzer based on a uniform window with an $N = 512$-point FFT, if no averaging is applied. The amplitude of the minimal detectable sine wave is determined by the input quantization noise, which is produced by $B = 10$-bit A/D converter. Apply the PQN model of quantization.

Detectability means that the peak belonging to the sine wave is with 95% probability higher than twice the maximum peak belonging to the quantization noise spectrum.

(**a**) Assume that for N samples, the sine wave frequency is equal to $\frac{100}{N} f_s$. Thus the frequency of the sine wave falls to the center of an FFT frequency bin. Then, the "picket fence"[6] effect plays no role.

(**b**) Assume that the sine wave frequency is equal to $\frac{100.5}{N} f_s$. Thus the frequency of the sine wave falls to the edge of an FFT frequency bin, and the "picket fence" effect decreases the height of the observed peak.

15.14 Examine the roundoff error for Example 20.10 (weightchecker, page 558), implemented as an FIR filter (use approximations of the dynamic parameters, based on Fig. 20.16(b), with the suggested $N = 1300$ samples),

(**a**) for fixed-point ($B = 16$ bits), and

(**b**) for floating-point ($p = 16$ bits)

number representation.

Compare the roundoff errors to that of an 8-bit ADC.

By comparison to an 1/2 LSB nonlinearity of the above ADC, suggest optimum bit numbers for the data processing in the weightchecker.
Hint: The number of bits should be chosen not too high (keep implementation cost low), and not too low (keep roundoff error low).

15.15 A symmetric square wave of frequency 100 Hz was sampled with sampling frequency $f_s = 1$ kHz, and $N = 1000$ sampling points. The DFT was calculated.

The DFT line corresponding to the base harmonic at 100 Hz is distorted, because the 21st, 41st etc. harmonics alias back to 100 Hz. We want to estimate the error caused by aliasing. We know that the coefficients of the harmonics of the symmetric square wave are proportional to the reciprocal of the harmonic number, so in aliasing, we need to add all the aliased amplitudes. However, it can be seen that the sum

$$\frac{1}{21} + \frac{1}{41} + \frac{1}{61} + \frac{1}{81} + \cdots \tag{E15.15.1}$$

is infinite, not speaking of the similarly aliasing harmonics of index -19, -39, etc. which also alias. Meanwhile, the values of the DFT are all finite ... There seems to be a contradiction here. What is the explanation?

[6] Check the phenomenon of picket fence (or scalloping loss) in any standard book on signal processing, e.g. (Oppenheim, Schafer and Buck, 1998).