

Effects of Finite Register Length in the Signal Processing Arithmetic Unit of the AN/UYK-17

JUDITH N. FROSCHER

*Information Processing Systems Branch
Communications Sciences Division*

June 12, 1974



NAVAL RESEARCH LABORATORY
Washington, D.C.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NRL Report 7732	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EFFECTS OF FINITE REGISTER LENGTH IN THE SIGNAL PROCESSING ARITHMETIC UNIT OF THE AN/UYK-17	5. TYPE OF REPORT & PERIOD COVERED Interim report on a continuing NRL problem.	
	6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) Judith N. Froscher	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, D.C. 20375	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NRL Problem B02-16 ZF11-121-003	
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Naval Undersea Center San Diego, California 92132	12. REPORT DATE June 12, 1974	
	13. NUMBER OF PAGES 40	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Digital filtering	Roundoff	
Fast Fourier transform	Truncation	
Fixed-point arithmetic	Two's complement	
Quantization		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
<p>Because of finite register length, several kinds of error are introduced in NRL's Signal Processing Arithmetic Unit (SPAU). The SPAU has a 16-bit word length and uses two's-complement, fixed-point, truncated arithmetic. A statistical model for arithmetic roundoff is formulated and used in discussing sources of error in multiplication and addition. Simulation proves a useful tool for studying quantization effects in the two fundamental digital signal processing algorithms, the fast Fourier transform (FFT) and recursive filtering. Simulation results indicate that conditional block scaling should be used, that scaling other than scaling the input should be used for the</p>		

4-pole filter to utilize more completely the SPAU's 90-dB dynamic range (6 dB per bit), and that rounding arithmetic should be used in the next version of the SPAU. The possibility of a floating-point machine should be investigated.

CONTENTS

INTRODUCTION	1
ERROR FUNDAMENTALS	2
Two's Complement Representation	2
Error Introduced by Roundoff	3
Statistics of Roundoff	6
Error Introduced by Multiplication and Addition	7
Error Accumulation	8
ERROR ACCUMULATION IN THE FFT	9
Description of the FFT	9
The SPAU Macro	9
Overview of the Literature	10
Scaling for Fixed-Point Arithmetic	11
Simulation Scheme for Error Analysis	12
QUANTIZATION ERROR IN DIGITAL FILTERS	15
Overview of the Literature	16
The SPAU Macro	17
Error Analysis	17
Scaling	20
CONCLUSION	20
ACKNOWLEDGMENTS	20
REFERENCES	20
APPENDIX A—Bad-Bit Distributions for the FFT	23
APPENDIX B—Bad-Bit Distributions for the Filtering Macro ..	34

EFFECTS OF FINITE REGISTER LENGTH IN THE SIGNAL PROCESSING ARITHMETIC UNIT OF THE AN/UYK-17

INTRODUCTION

In general, in the use of digital computers to perform arithmetic calculations, a finite register length introduces errors. This accuracy problem has received a great deal of study recently, especially for signal-processing applications, since special-purpose digital processors are now being built. NRL is developing a microprogrammed signal processor, the Signal Processing Element (SPE), to be part of the Navy's All Applications Digital Computer (AADC) [1]. The SPE consists of four major subsystems: a Microprogrammed Control Unit (MCU), a Buffer Store and Storage Control Unit (SCU), a Signal Processing Arithmetic Unit (SPAU), and Input/Output (I/O) units [2]. This report is a study of the quantization effects of the SPAU in its performance of the fast Fourier transform (FFT) and digital recursive filtering with two test signals and two roundoff rules.

The SPAU has a 16-bit word and two's complement, fixed-point, truncated arithmetic. It operates at 150 ns/cycle. Quantization error is introduced in the following ways:

1. 16-bit representation of input data
2. 16-bit representation of system coefficients
3. Truncation or rounding at multipliers
4. Scaling to prevent overflow in adders because of fixed-point arithmetic.

The first source of error can be disregarded, since the input data will be accurate to 10 bits at most.

First a statistical model of arithmetic roundoff (rounding or truncation) is developed for the SPAU's particular arithmetic. The errors due to roundoff and scaling are then discussed for both addition and multiplication.

One of the fundamental algorithms of digital signal processing is the FFT, a timesaving scheme for calculating the discrete Fourier transform of a sequence of numbers. The FFT output from a simulation of the SPAU arithmetic is compared with a floating-point FFT output, computed with 60-bit arithmetic and considered ideal relative to 16-bit arithmetic. The process loss is displayed with noise-to-signal ratio curves in decibels (NSR(dB)) and "bad-bit" distributions. It is concluded that conditional block scaling should be implemented for the FFT on the SPAU.

The other algorithm studied is recursive digital filtering. A 4-pole filter, realized as two 2-pole filters in cascade, is analyzed using the same type of error study as for the FFT. It is recommended that the scaling be changed to make more complete use of the SPAU's 90-dB dynamic range.

Note: Manuscript submitted February 5, 1974.

ERROR FUNDAMENTALS

Two's Complement Representation

Without loss of generality, it can be assumed that $-1 \leq x < 1$ in a fixed-point representation. In a binary fixed-point machine, a number can be represented in two's complement form as

$$x_2 = b_0 + \sum_{i=1}^N b_i 2^{-i},$$

where each register contains $N + 1$ bits and b_i is zero or one for $0 \leq i \leq N$. For the SPAU, $N + 1 = 16$, and 2^{-15} is defined to be the level of quantization.

Now, represent $|x|$ in binary form;

$$|x| = \sum_{i=1}^N a_i 2^{-i}, \quad -1 \leq x < 1.$$

It is necessary to determine the value of the sign weight b_0 . In two's complement notation, if $x \geq 0$, $b_0 = 0$, and if $x < 0$, $x_2 = 2 - |x|$; hence, the name "two's complement." Therefore, if $x \geq 0$,

$$x_2 = b_0 + \sum_{i=1}^N b_i 2^{-i}$$

where $b_0 = 0$. If $x < 0$,

$$x_2 = 2 - \sum_{i=1}^N a_i 2^{-i} \text{ or } b_0 = 1.$$

For example, represent $x = 3/4$ in two's-complement notation for $N + 1 = 16$:

$$x_2 = 0 + 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2^2} + 0 \cdot \frac{1}{2^3} + \dots + 0 \cdot \frac{1}{2^{15}}.$$

For two's complement, let $y = -x = -3/4$, Then

$$\begin{aligned} y_2 &= 2 - |y| = 2 - \frac{3}{4} \\ &= 1 + 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2^2} + 0 \cdot \frac{1}{2^3} + \dots + 0 \cdot \frac{1}{2^{15}}. \end{aligned}$$

The example above illustrates that the high-order bit acts as a sign bit. If x is positive, $b_0 = 0$, and if x is negative, $b_0 = 1$.

Error Introduced by Roundoff

Rounding Error—When two N -bit numbers are multiplied digitally, the full product contains $2N$ bits. Since the SPAU is a 16-bit machine, the full product must be rounded or truncated to 16 bits. Several rounding methods exist. They result in different trade-offs among timing, hardware, and accuracy in the digital multiplication. Although several rounding rules are available, the SPAU architectural constraints limit the choices.

For simplicity, let each word contain $b + 1$ bits. Probably the most common rounding rule is to add 1 in the highest order bit to be dropped and then to discard those bits [3]. Hence, if the dropped bits are less than $2^{-b}/2$, the number is rounded down; if they are greater than or equal to $2^{-b}/2$, the number is rounded up. Always rounding up when the dropped bits equal $2^{-b}/2$ results in a very small mean error ($2^{-(2b+1)} = 2^{-31}$ for the SPAU) and a maximum error of $2^{-b}/2$. Summing the 1 into the $(b+1)$ -bit position can be done automatically and continuously in the multiplier and takes essentially no extra time, although it requires some additional logic.

Another rounding scheme, which requires less hardware, is to make the least significant retained bit a 1, regardless of the other bits, and then to truncate. In other words, if the last retained bit is 1, add 0; if the last retained bit is 0, add 2^{-b} , a 1 is thus added randomly in the least significant retained bit with a probability of $1/2$. This method, however, produces a maximum error twice as large as that of the previous method and a variance four times as great. For example, suppose a machine has 4-bit arithmetic and the produce is 11000000. By this rule, the rounded result is 1101 and the error is 2^{-b} or $2(2^{-b}/2)$, whereas $2^{-b}/2$ is the maximum error produced in the previous procedure.

A more accurate rounding rule is a variation of the first rule. When the dropped portion of the number is exactly $2^{-b}/2$, a 1 or 0 is randomly placed in the least significant retained bit. The main difficulty, besides the need for a random number generator, is that results cannot be reproduced exactly at will. Another variation on the first rule involves examination of the least significant retained bit; when the bits to be dropped equal $2^{-b}/2$ exactly, add $2^{-b}/2$ if the least significant retained bit is 1 and 0 if it is 0.

Both of these variations require an added delay, since the result must be tested and combined with the correction bit in a carry-propagate adder. The additional add can take up to 30 ns in the SPAU multiplier and impose a serious delay in the already time-critical path of the SPAU.

Because of the difficulties of the last three methods, consideration of rounding will be restricted to the first method.

Suppose a number x is represented in two's-complement notation as

$$x_2 = b_0 + \sum_{i=1}^N b_i 2^{-i}.$$

For a particular machine, x must be represented in N_2 bits, $N_2 < N_1$; x_2 must be rounded or truncated to an N_2 -bit number. The most common rounding rule is to add 1 in the $(N_2 + 1)$ -bit position, then chop off $N_1 - N_2$ bits. Let

$$x_2 = b_0 + \sum_{k=1}^{N_1} b_k 2^{-k}$$

and let

$$\bar{x}_2 = b_0 + \sum_{k=1}^{N_2} b'_k 2^{-k}$$

be the rounded x_2 . The difference $(\bar{x}_2 - x_2)$ is defined as the error e_R , so that

$$\begin{aligned} |e_R| &= \left| b_0 + \sum_{k=1}^{N_2} b'_k 2^{-k} - \left[b_0 + \sum_{k=1}^{N_1} b_k 2^{-k} \right] \right| \\ &= \left| \sum_{k=1}^{N_2} b'_k 2^{-k} - \sum_{k=1}^{N_1} b_k 2^{-k} \right| \\ &= \left| \sum_{k=1}^{N_2} b'_k 2^{-k} - \left[\sum_{k=1}^{N_2+1} b'_k 2^{-k} - 2^{-(N_2+1)} + \sum_{k=N_2+2}^{N_1} b_k 2^{-k} \right] \right| \\ &= \left| (1 - b'_{N_2+1}) 2^{-(N_2+1)} - \sum_{k=N_2+2}^{N_1} b_k 2^{-k} \right|. \end{aligned}$$

Therefore

$$|e_R| \leq 2^{-(N_2+1)}$$

or

$$-\frac{1}{2} 2^{-N_2} \leq e_R \leq \frac{1}{2} 2^{-N_2}.$$

Truncation Error—To determine the error for truncation, it is necessary to look at the sign of the number. Let

$$x_2 = b_0 + \sum_{i=1}^{N_1} b_i 2^{-i}$$

and

$$\bar{x}_2 = b_0 + \sum_{k=1}^{N_2} b_k 2^{-k}$$

where $N_2 < N_1$. Define $e_T = \bar{x}_2 - x_2$. For positive x , the effect of truncation is that the N_2 -bit representation of x is less than the N_1 -bit representation; i.e., $\bar{x}_2 < x_2$. Thus, $e_T < 0$, and $0 \geq e_T \geq -(2^{-N_2} - 2^{-N_1})$. For $x < 0$, expressed in two's-complement notation,

$$|x_2| = 2 - x_2$$

where

$$x_2 = 1 + \sum_{k=1}^{N_1} b_k 2^{-k},$$

$$\bar{x}_2 = 1 + \sum_{k=1}^{N_2} b_k 2^{-k}, N_2 < N_1,$$

and

$$|\bar{x}_2| = 2 - \bar{x}_2.$$

Hence,

$$\begin{aligned} |\bar{x}_2| - |x_2| &= -\bar{x}_2 + x_2 \\ &= \sum_{k=N_2+1}^{N_1} b_k 2^{-k} \\ &< 2^{-N_2}. \end{aligned}$$

The effect of truncation is to increase the magnitude, and thus decrease the value, of a negative number. Thus, e_T is again negative and $0 \geq e_T \geq -(2^{-N_2} - 2^{-N_1})$. Therefore, for both positive and negative numbers, the error due to truncation is negative; i.e., the truncated value is always less than the "true" value.

Statistics of Roundoff

When the quantization level is small compared to the number, the probability densities can be considered uniform [4]. The probability densities for randomized rounding and for two's-complement truncation are shown in Figs. 1a and 1b, respectively, where 2^{-b} is the quantization level.

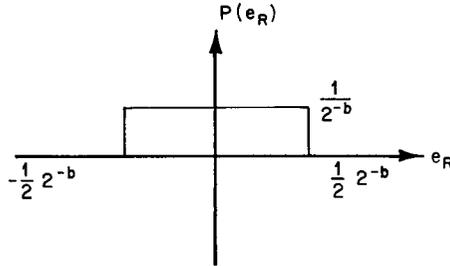


Fig. 1a—Probability density for fixed-point rounding

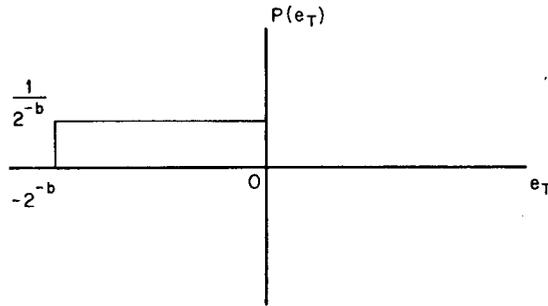


Fig. 1b—Probability density for two's-complement fixed-point truncation

For rounding, the mean is

$$\bar{M}_R = \int_{-2^{-b}/2}^{2^{-b}/2} \frac{x}{2^{-b}} dx = 0,$$

and the variance is

$$\sigma_{e_R}^2 = \int_{-2^{-b}/2}^{2^{-b}/2} \frac{x^2}{2^{-b}} dx = \frac{2^{-2b}}{12} .$$

For truncation, the mean is

$$\begin{aligned}\bar{M}_{e_T} &= \int_{-2^{-b}}^0 \frac{x}{2^{-b}} dx \\ &= -\frac{2^{-b}}{2},\end{aligned}$$

and the variance is

$$\begin{aligned}\sigma_{e_T}^2 &= \int_{-2^{-b}}^0 \frac{1}{2^{-b}} \left(x + \frac{2^{-b}}{2}\right)^2 dx \\ &= \frac{2^{-2b}}{12},\end{aligned}$$

or

$$\sigma_{e_T}^2 = \sigma_{e_R}^2.$$

Because the mean of the truncated error distribution is not zero, the error due to truncation accumulates faster than that for rounding. For the rounding rule used in the SPAU, there is a small mean error, and the variance is slightly smaller, as shown above.

Error Introduced by Multiplication and Addition

The basic operations in the SPAU are multiplication and addition; when these operations are performed, errors due to the limited dynamic range and roundoff are introduced. If the numbers to be multiplied are less than 1 in magnitude, no scaling is necessary. If $|x| \leq 1$ and $|y| \leq 1$, $x \cdot y$ overflows only if $x = -1$ and $y = -1$. Since the product of two b_1 -bit numbers is a $(b_1 + b_1)$ -bit number, the result must be truncated or rounded to b_1 bits. Consider two inputs x and y which have been rounded or truncated to b_1 bits and are represented as \bar{x}_2 and \bar{y}_2 ; $\bar{x}_2 = x_2 + e$ and $\bar{y}_2 = y_2 + e'$. Before p is rounded or truncated,

$$\begin{aligned}p &= \bar{x}_2 \bar{y}_2 \\ &= (x_2 + e) \cdot (y_2 + e').\end{aligned}$$

If we assume statistical independence,

$$E(p) = E(\bar{x}_2) \cdot E(\bar{y}_2).$$

For rounding,

$$\begin{aligned} E[p] &= \{E[x_2] + E[e]\} \cdot \{E[y_2] + E[e']\} \\ &= E[x_2] \cdot E[y_2] \end{aligned}$$

since

$$E[e] = E[e'] = 0.$$

For truncation, since

$$\begin{aligned} E[e] &= E[e'] = \frac{2^{-b}}{2} \\ E[p] &= \left\{E[x_2] - \frac{2^{-b}}{2}\right\} \cdot \left\{E[y_2] - \frac{2^{-b}}{2}\right\} \\ &= E[x_2] \cdot E[y_2] - \frac{2^{-b}}{2} \{E[x_2] + E[y_2]\} + \frac{2^{-2b}}{4} \end{aligned}$$

If x and y have the same sign, the error builds in one direction and can change the value of the least significant bits after truncation. This increased error is caused by the non-zero mean of the truncated error distribution. For multiplication, as the numbers become less random the error becomes more correlated. If two numbers \bar{x}_2 and \bar{y}_2 do not overflow when they are added, the process of fixed-point addition introduces no error. In most cases, however, scaling must be performed to prevent overflow for all fixed-point arithmetic. This scaling reduces the number of bits that can be used effectively for calculations and accordingly decreases the accuracy of the computation.

Error Accumulation

In signal-processing algorithms, large blocks of data are analyzed; therefore, accumulation of error is of interest. The error accumulation in a process requiring many operations is greater for two's complement truncation than for rounding, since the error distribution for truncation has a bias, although the truncation variance equals the rounding variance. The fixed dynamic-range constraint limits the number of bits that can be used for the calculations. Weinstein [5,6] and Knowles [7,8] among others, have statistically modeled the quantization error for fixed-point rounding (FFT and recursive filtering) and fixed-point truncation and rounding (recursive filtering), respectively. However, Jackson [Ref 9, p. 163] warns, "as signals become less random, the uncorrelated error assumption tends to break down more readily for truncation than for rounding."

ERROR ACCUMULATION IN THE FFT

Description of the FFT

One of the most powerful tools of signal processing is the Fourier transform; in digital signal processing, since the data are discrete, this method of analysis is referred to as the discrete Fourier transform (DFT). If one defines an operation to be a multiplication and an addition, n^2 operations are required to perform a DFT on a sequence of n data points. In 1965, Cooley and Tukey introduced the FFT, an algorithm to calculate the DFT of n data points (where $n = 2^m$) in $n \log_2 n$ operations [10-13].

Suppose that the sequence $\{f(nT)\}_{n=0}^{N-1}$ is obtained from a continuous function $f(t)$ by sampling every T seconds. For a discussion on the choice of a suitable T , see Glisson, Black, and Sage [14]. Denote the Fourier transform of $f(t)$ by $F(\omega)$. Then the DFT of $\{f(nT)\}_{n=0}^{N-1}$ is $\{F(k\Omega)\}_{n=0}^{N-1}$, where

$$F(k\Omega) = \sum_{n=0}^{N-1} f(nT)e^{-ink\Omega T}$$

with $\Omega = 2\pi/NT$. [Ref. 4, p. 143].

The SPAU Macro

The SPAU FFT macro is a decimation in time algorithm. The SPAU algorithm does not require bit reversal, which would be required for an "in place" FFT. It uses instead two storage locations per data point. After each stage of the FFT, the newly calculated data points are shuffled between the two locations. Let $T = 1$, $\Omega = 1$, and $W = \exp(-2\pi i/N)$, Then

$$F(k) = \sum_{n=0}^{N-1} f(n) W^{nk}.$$

A flow diagram for $N = 8$ is shown in Fig. 2. The arrows indicate multiplication by W^ℓ , and the nodes indicate addition of the two quantities connected. The basic FFT computation, called a butterfly, is shown in Fig. 3. To determine the exponent of W , $\ell = [(kN/2)/2^m] \bmod (N/2)$, where m is the stage number; this holds except for stage 0, as shown in Fig. 2. The butterfly computation is performed $N/2$ times for each stage. A perfect card shuffle is performed on the data at the first stage; in the second stage, the new data are shuffled as if two cards were pasted together; then as if 4 cards were pasted together, and so on.

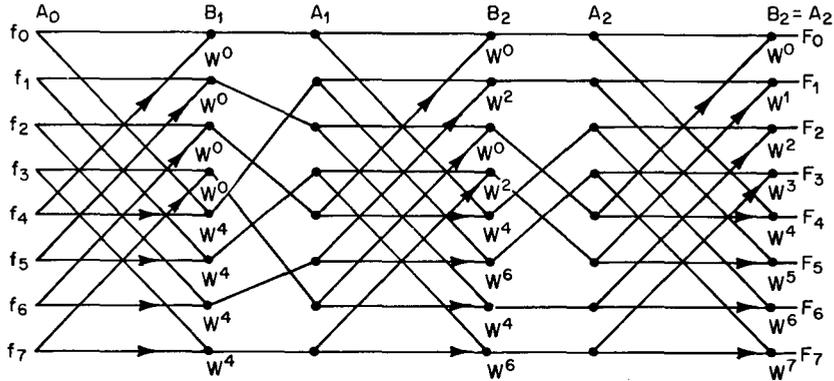


Fig. 2—Flow diagram for SPAU FFT macro, $N = 8$

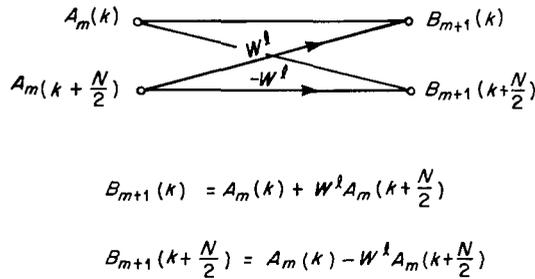


Fig. 3—Butterfly for SPAU FFT

Overview of the Literature

Several studies of error from finite register length have been done for the FFT. Welch [15] did an error analysis of a fixed-point FFT and derived approximate upper and lower bounds on the root-mean-square error for rounding. He experimentally obtained an upper bound estimate (three times that of rounding) for two's-complement truncation. SPAU simulation provided results which were within this bound. Kaneko and Liu [16] statistically modeled the mean-square error for a floating-point rounded FFT and derived an upper and lower bound on the mean-square error for floating-point truncation. Experimental results verified their model. In his thesis, Kaneko [17] pursued this problem further and undertook a study in which he analyzed the isolated effect of each of the three sources of quantization error for floating-point arithmetic. Glisson, Black, and Sage [14] did a dynamic range study for the FFT and showed that experimental results tend to support this model. They also examined the effects of automatic block scaling versus conditional block scaling on computational accuracy as a function of three variables: the input word length, the system coefficient word length, and the internal arithmetic word length. They concluded that the number of bits used for internal arithmetic needed to be greater than the number used for input data or system coefficients and that, in general, conditional block scaling is better than automatic block scaling. The use of nonuniform word lengths could permit considerable savings in hardware, storage, and speed. For a white-noise input, Weinstein and Oppenheim [6] statistically modeled the

FFT quantization error for rounded fixed-point arithmetic as an additive noise source. Simulation results verified their model. Weinstein [5] further explored quantization effects for fixed-point, block floating-point, and floating-point FFTs. Although he did no modeling for truncation, he experimentally obtained NSR curves for truncated arithmetic. No statistical model for a truncated, fixed-point FFT has been derived, since the chopping noise becomes correlated as more operations are performed.

Scaling for Fixed-Point Arithmetic

In a fixed-point implementation of the FFT, scaling is absolutely necessary. Input data are bounded in magnitude by 1, $|f(n)| \leq 1$. The DFT of $\{f(n)\}_{n=0}^{N-1}$ can be expressed as

$$F(k) = \sum_{n=0}^{N-1} f(n) \exp(-2\pi ink/N)$$

Hence,

$$|F(k)| \leq \sum_{n=0}^{N-1} |f(n) \exp(-2\pi ink/N)|,$$

by the triangle inequality

$$\leq \sum_{i=0}^{N-1} |f(n)|,$$

since $|\exp(-2\pi ink/N)| = 1$. Therefore,

$$|F(k)| \leq N.$$

To absolutely prevent overflow, the input data can be scaled by N before they are ever processed. With this scaling, a great deal of accuracy is lost at the beginning and the error is allowed to accumulate at each stage. For example, if a 4096-point FFT is to be done and the data are good to only 10 bits, this scaling would require that the data be right-shifted 12 bits. Since this scaling leads to a large error, another approach would seem necessary.

Let $W = \exp(-2\pi i/N)$. From Fig. 3,

$$\begin{aligned} B_m(k) &= A_{m-1}(k) + W^k A_{m-1}(k + N/2) \\ B_m(k + N/2) &= A_{m-1}(k) - W^k A_{m-1}(k + N/2). \end{aligned}$$

Consider

$$\begin{aligned} & |A_{m+1}(j)|^2 + |A_{m+1}(j + N/2)|^2 \\ &= 2 [|A_m(j)|^2 + |A_m(j + N/2)|^2]. \end{aligned}$$

Therefore, scaling by 1/2 at each stage would prevent overflow. This automatic block scaling would shift out only one bit of significance at each stage and, in addition, would shift out noise accumulated at earlier stages.

There is another type of scaling, called conditional block scaling. The conditional block scaling implemented for the SPAU FFT macro tests each output data point during each stage of computation to determine if it is greater than 1/2. If it is, the next stage is executed with a scale factor of 1/2. However, whether scaling is required or not, there is an overflow test after each addition or subtraction. Although the two data points that enter the butterfly computation are bounded by 1/2, overflow is still possible. Consider

$$B_m(k) = A_{m-1}(k) \pm W^k A_{m-1}(k + N/2).$$

Thus,

$$\begin{aligned} |B_m(k)| &\leq 0.7071 + 0.7071 \\ &= 1.4142 > 1, \end{aligned}$$

since

$$\begin{aligned} |A_{m-1}(k)|^2 &\leq |\operatorname{Re} A_{m-1}(k)|^2 + |\operatorname{Im} A_{m-1}(k)|^2 \\ &= \frac{1}{4} + \frac{1}{4} \\ &= \frac{1}{2} \end{aligned}$$

and

$$|A_{m-1}(k)| \leq 0.7071.$$

If overflow is detected, the data points are scaled and the complete stage is recalculated. It is possible to retain more significance in the data with conditional block scaling than with automatic block scaling by 1/2 at each stage. It also enables the SPAU to use more effectively the full dynamic range of its 16-bit registers.

Simulation Scheme for Error Analysis

To study the error caused by a finite word length for the SPAU, the scheme shown in Fig. 4 was used. Two input signals were used: a sum of decaying sinusoids and a

sequence of random numbers $\{x_n\}$, $0 \leq x_n \leq 1$. Results from the floating-point 60-bit FFT were considered ideal as compared to results from the 16-bit SPAU-simulated FFT. The difference was calculated and the process loss was exhibited in two ways: noise-to-signal ratio (dB) and bad-bit distributions. If it is assumed that the signal has already been sampled and time-truncated, the sources of error in the FFT can be enumerated as follows:

1. Quantization of data to 16 bits
2. Quantization of sines and cosines to 16 bits
3. Truncation or rounding error after each multiplication
4. Scaling to prevent adder overflow.

These sources of error were not studied separately, since the purpose was to evaluate the total process loss.

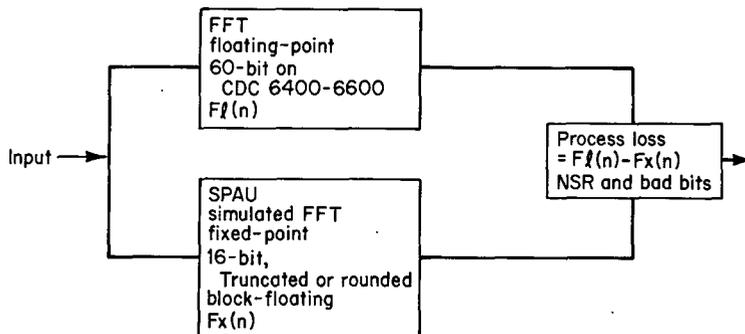


Fig. 4—Scheme for SPAU FFT error analysis

Description of Test Signals — Dynamic range constraints were clearly exhibited by the first test function, a sum of decaying sinusoids.

$$s(t) = \sum_{k=0}^N 10^{-k/8} \sin(2\pi f_k t)$$

where

$$f_k = 10 \text{ kHz}$$

$$N = 12 \text{ for a 256-point FFT}$$

$$N = 20 \text{ for 512-through-4096-point FFTs.}$$

An FFT (as described in the first paragraph of this section) with automatic block scaling of 1/2 at each stage was first implemented. The power spectrum was calculated by squaring the real and imaginary parts and then summing. The squaring, however, doubled the dynamic range needed to represent the new sequence. Since the numbers were scaled at each stage, small numbers were squared, and some were too small to be represented in 16 bits. This phenomenon is called thresholding or underflow. As a consequence, the

FFT was used with conditional block scaling. Fewer underflows were observed, but thresholding was still a problem. It might be advisable to use an algorithm to approximate the square root of the power spectrum. These algorithms have gained considerable interest recently. An even better approach would be to investigate the possibilities of floating-point arithmetic.

The other test signal was a sequence of uniform random numbers such that

$$\{X_n\}_{n=0}^{N-1}, 0 \leq X_n \leq 1.$$

Even with conditional block scaling, the FFT data points were scaled at each stage by 1/2, since the mean of the sequence, 1/2, appears at the 0-frequency point. For this test function, conditional block scaling was equivalent to automatic block scaling. This test signal was a worst-case test for the SPAU.

Noise-to-Signal Ratio Graphs—First the noise-to-signal ratios for rounding and truncating were computed. Denote by $F\ell(k)$ the ideal floating-point output and by $Fx(k)$ the fixed-point output, which was converted to floating-point representations. Then

$$\text{NSR} = \frac{\text{noise}^2}{\text{signal}^2} = \frac{\sum_{k=1}^{N-1} [F\ell(k) - Fx(k)]^2}{\sum_{k=0}^{N-1} [F\ell(k)]^2}$$

and

$$\text{NSR}(\text{dB}) = 10 \log_{10} (\text{NSR}).$$

NSR(dB) was plotted against $\log_2 N$ for the FFT of both signals for rounding and truncation (Fig. 5). As noted earlier, the random-number input was scaled automatically; hence, the NSR curves are relatively linear, with the rounded curve lower than the truncated one. The NSR curves for the sinusoids have discontinuities caused by the discontinuous conditional-block scaling, fixed-point arithmetic. As Weinstein notes [5], the curves for both rounding and truncation have the same general shape, since the scaling is done in the same way.

Bad-Bit Distribution—Of interest to the SPAU designers was the number of bits in error (bad bits) due to processing. A bit-by-bit comparison of floating-point results and SPAU simulated results was made. Appendix A contains the bad-bit distributions for both test signals in both truncated and rounded arithmetic. From these graphs, the average number of bad bits for truncation is about twice the average number for rounding. The SPAU designers are contemplating a change to rounded arithmetic in the next version.

Simulation Results—To gain confidence in these results, the rms error was calculated and compared with the results of Welch [15]. The results fell within his upper and lower bounds. Since Weinstein's block floating-point FFT algorithm is not exactly the same as the SPAU's conditional block-scaling FFT and his modeling was for white noise, an exact

comparison could not be made [6]. For the sum of decaying sinusoids, however, the NSR curves (from SPAU simulation) were 65 dB down on the average and agreed closely with Weinstein's results (within 3 dB). The random noise signal did not compare as well, since scaling occurred at each stage with this signal, and Weinstein did not present NSR curves for this case.

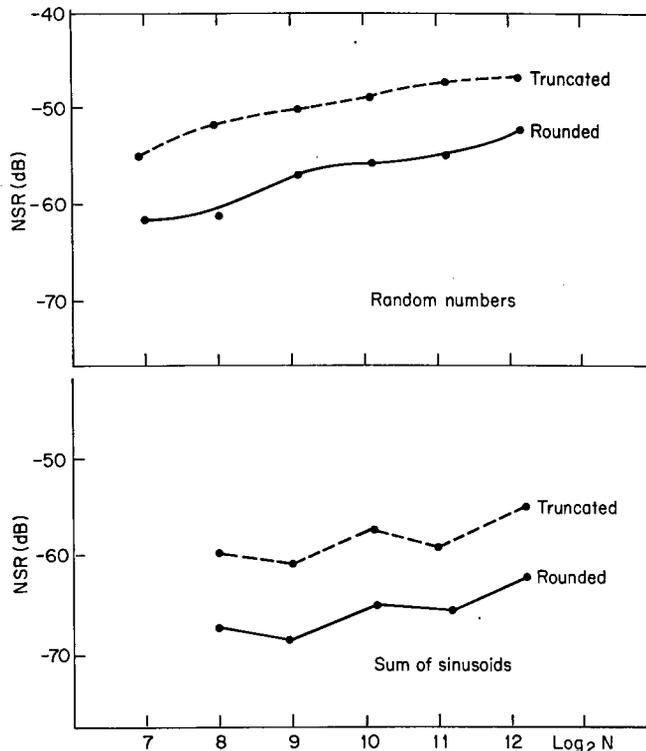


Fig. 5—Results of NSR simulation

With the SPAU's 16-bit registers, a 90-dB dynamic range is possible. With truncation, processing resulted in an average loss of 2 bits and a maximum loss of 6. In practical applications, the input data will be accurate to at most 10 bits. Hence, the accuracy of the output will be even better for these data. In actuality, the SPAU's processing loss for the FFT is tolerable if conditional block scaling is used. Also, if the power spectrum is needed, algorithms other than squaring should be considered. Rounding is strongly recommended. A study of the tradeoffs for floating-point arithmetic is advised.

QUANTIZATION ERROR IN DIGITAL FILTERING

The other fundamental algorithm of signal processing is linear filtering. Before the advent of digital signal processing, filtering was done in the continuous domain. With the increased speed and efficiency of digital computers, the conversion to digital signal processing began. The introduction of the FFT algorithm has certainly added impetus to the

trend toward digital processing. Steiglitz [8] published a very interesting thesis on the equivalence of the analog and digital domains by exhibiting a mapping from the analog domain to the digital domain and rigorously proving that this mapping was an isomorphism. A digital filter can be defined as the computational process into which a sampled signal or sequence of numbers is fed and out of which comes a numerical sequence, the output signal. The digital filter can act as a band-pass filter, a differentiator, an integrator, etc., as can an analog filter. The filtering algorithm is realized as a difference equation.

Overview of the Literature

The basic operations of digital filtering are multiplication by a constant, multiplication by a filter coefficient, and addition. For 16-bit, fixed-point, truncated arithmetic, the sources of quantization error are:

1. Quantization of input data to 16 bits
2. Quantization of filter coefficients to 16 bits
3. Truncation or rounding after multiplication
4. Scaling to prevent overflow after addition.

Quantization effects have been enthusiastically researched for digital filters. Kaiser's study [19] is a brief overview of the field, including design techniques and computational problems associated with the finite word length requirement. He also studies pole positions for different quantization levels and for different filter configurations.

Knowles and Olcayto [8] demonstrate a method for evaluating the rms value of output noise from input quantization and roundoff error accumulation for any quantization level and for different realizations. They also statistically predict the expected values of the mean-square difference in the real frequency responses of the ideal and actual filter, the actual filter having quantized coefficients. Experimental results verify this analysis. They note that this method is suitable for direct and parallel realizations but is generally unsuitable for the cascade realization. In another very useful work, Knowles and Edwards [7] examine quantization effects for both rounded and truncated fixed-point arithmetic. They consider roundoff error as an additive noise source and derive a statistical model for the system rms error. They verify their model with experimental evidence. In fact, the SPAU simulation yields comparable results.

Jackson [9] investigates the interaction between roundoff noise output from a digital filter and associated dynamic range limitations for the case of uncorrelated rounding errors since the "uncorrelated error assumption breaks down more rapidly for truncation as the signal becomes less random." In another paper, [20] Jackson studies different realizations of digital filters and indicates useful "rules of thumb" for a choice of configuration. Computational results verify the analysis.

In his thesis, Kaneko [17] does a floating-point error analysis. He rigorously derives the mean-square error due to roundoff accumulation and input quantization for both rounding and truncation for floating-point digital filters. He supports his model with experimental evidence. Oppenheim and Weinstein [6] statistically derive a noise-to-signal ratio for rounded fixed-point arithmetic for a white-noise input. They also establish bounds on the input magnitude to prevent overflow, but they point out that these bounds

are sometimes very difficult to calculate. In his thesis, Weinstein studies the effect of coefficient quantization on pole positions. He compares fixed-point, block floating-point, and floating-point realizations of a digital filter. Simulation results verify his analysis. In the first five chapters of their book, Gold and Rader [4] deal extensively with all aspects of digital filtering; however, they do not analyze truncated fixed-point arithmetic.

The SPAU Macro

The SPAU has fast multipliers. To use this hardware efficiently, the linear filtering macro consists of a 4-pole filter realized as two 2-pole filters in cascade. The filter transfer function is

$$H(z) = \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}},$$

where a_1 , a_2 , b_1 , and b_2 are real. The 2-pole configuration is shown in Fig. 6. The circles represent adders, the boxes represent delays, and the coefficients at the arrows indicate multiplication. The difference equations for this configuration are

$$W(n) = X(n) + b_1 W(n - 1) + b_2 W(n - 2)$$

and

$$y(n) = W(n) + a_1 W(n - 1) + a_2 W(n - 2)$$

where, for simplicity, $T = 1$. To use the SPAU hardware more efficiently, the filter was implemented as indicated by the dotted line. The difference equations for this realization are

$$W'(n) = X(n) + b_1 W(n - 1) + b_2 W(n - 2)$$

and

$$y(n) = X(n) + (a_1 + b_1)W(n - 1) + (a_2 + b_2)W(n - 2).$$

Since the adder does not have to wait for $W(n)$ to be calculated, this configuration is more regular (Fig. 7).

The 4-pole filter macro consists of two 2-pole filters in cascade, as shown in Fig. 6, i.e., the output from the first filter is fed into the second and processed.

Error Analysis

In the same vein as the FFT error analysis, the scheme of Fig. 8 was used. The same two test functions were used as for the FFT error analysis. Filter coefficients and input data alike were bounded by 2 in magnitude. Scaling for the 4-pole macro was done

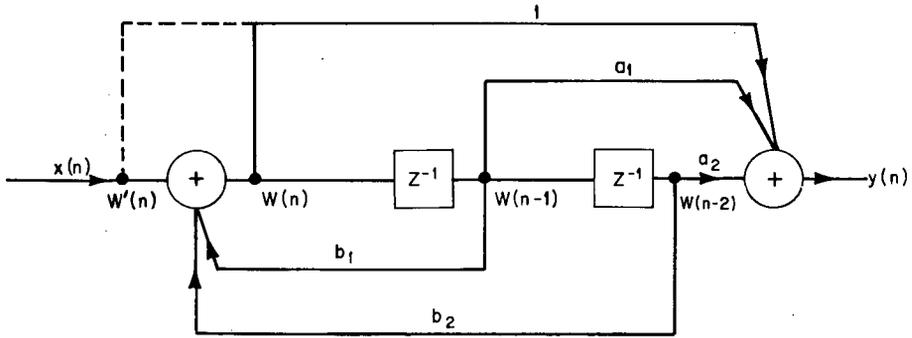


Fig. 6—The 2-pole filter implemented in the SPAU macro

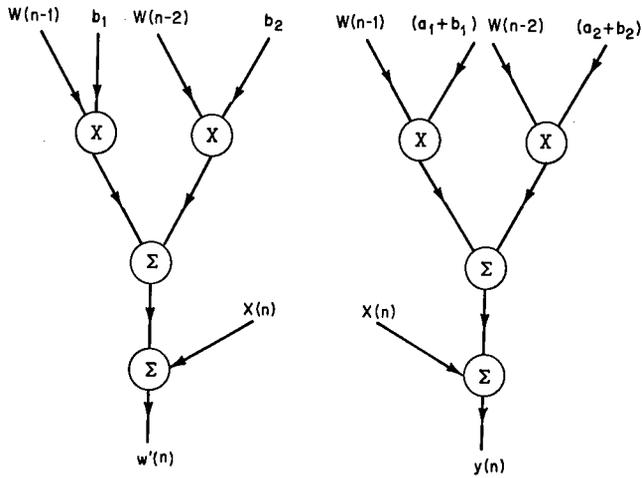


Fig. 7—SPAU hardware configuration for 2-pole filter

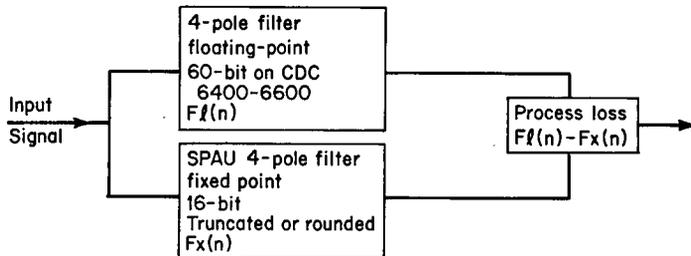


Fig. 8—Scheme for SPAU 4-pole filter analysis

on the input data at the beginning; the worst case was thus taken, in that significance in the input data was lost and the error was allowed to accumulate.

NSR Curves—The transfer functions for the two 2-pole filters in cascade are

$$H_1(z) = \frac{1 + 1.4767z^{-1} + 0.9999z^{-2}}{1 + 0.1061z^{-1} - 0.8758z^{-2}}$$

and

$$H_2(z) = \frac{1 + 0.2253z^{-1} + 1.0000z^{-2}}{1 + 0.5048z^{-1} - 0.2912z^{-2}}$$

The transfer function for the 4-pole filter is the product of $H_1(z)$ and $H_2(z)$. The poles of $H_1(z)$ are $Z = 0.8843$ and $Z = 0.9904$ and the poles of $H_2(z)$ are $Z = 0.3937$ and $Z = 0.8985$. The noise-to-signal ratios were calculated as before, as

$$NSR(\text{dB}) = 10 \log_{10} \left[\frac{\sum_{k=0}^{N-1} [f\ell(k) - fx(k)]^2}{\sum_{k=0}^{N-1} [f\ell(k)]^2} \right]$$

As opposed to the FFT, the data samples go through the filter algorithm only once. The data pass through the algorithm, and the roundoff noise for the k th input point damps out after several iterations; hence, the error is not averaged over many iterations. Another source of error accumulation is the scaling. The error is usually in the low-order bits. In the FFT algorithm, a bit of error is shifted out as scaling is required. With scaling of the input data, the error tends to be greater. Thus, for different input sequence lengths, the error is fairly constant. The NSR (dB) curves certainly exhibit this behavior. The graphs are horizontal and the rounding curve is 10 dB down from that for truncation (Fig. 9). The sum of sinusoids produced a slightly higher NSR curve (a larger error) since this input was scaled by 8, or three bits, and the random input was scaled by 4, or two bits.

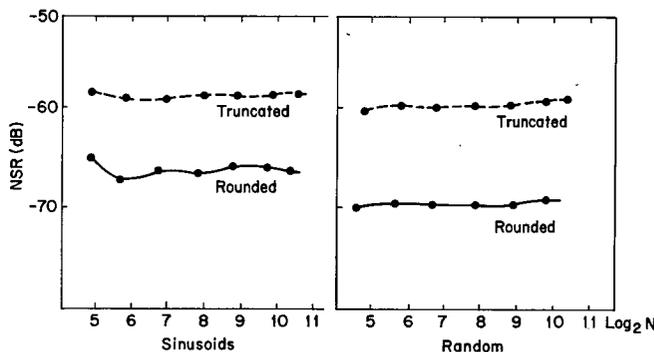


Fig. 9—NSR (dB) curves for random numbers and a sum of sinusoids

Bad-Bit Distributions—Bad-bit distributions were plotted as for the FFT, and are shown in Appendix B. Truncation produces about twice as much error in bits as does rounding. The mean of the bad-bit distribution for random numbers is larger than that for the sinusoids. The effect of the truncation bias is clearly exhibited, especially for the uniform random input, which is a worst case. Knowles and Edwards [7] predicted the rms error for fixed-point truncation; the rms error for the SPAU arithmetic is of the same order of magnitude as that predicted.

Scaling

Scaling presents a problem in recursive digital filtering. It is recommended that scaling be done in at least two stages. First scale the input to prevent overflow in the first 2-pole filter, and then scale the output from the first filter to prevent overflow in the second filter. An alternative is to consider the block-floating-point realization, as suggested by Oppenheim [21]; this implementation however, would make the algorithm use more machine cycles. Floating-point arithmetic would greatly facilitate implementation of the filtering algorithm.

CONCLUSION

For both the FFT and recursive filtering, dynamic-range limitations necessitate scaling, which poses a problem as far as minimization of quantization error is concerned. While fixed-point addition is more accurate than floating-point addition if the full register length can be used, it is not possible to use the full register length since the 16-bit fixed-point arithmetic imposes scaling because of the dynamic-range constraints. Based on simulation results, the decision was made to implement conditional block scaling for the FFT, and results have been satisfactory. The scaling problem for recursive filtering is still under study.

For the designers of the SPAU, a bad-bit distribution proved to be a more satisfactory measure of system performance than NSR (dB) curves. Both representations, however, showed the SPAU's performance to be acceptable. Finally, simulation results for both algorithms indicate that the implementation of rounding arithmetic would significantly decrease the quantization error. Also, floating-point arithmetic would greatly alleviate the dynamic-range constraint, i.e., the scaling problem.

ACKNOWLEDGMENTS

The author thanks L. Russo, J. L. Schilling, B. Shay, H. Smith, W. Smith, J. Speiser, and Y. S. Wu for helpful discussions on signal processing and error analysis. Also, the author wishes to thank T. Rauscher and J. Roberts for programming assistance.

REFERENCES

1. W.R. Smith and J.P. Ihnat, "Signal Processing Element Users' Reference Manual," NRL Report 7488, Sept. 5, 1972.

2. W.R. Smith and H.H. Smith, "Signal Processing Element Functional Description Part 2 (Preliminary)—Signal Processing Arithmetic Unit," NRL Memorandum Report 2522, Oct. 1972.
3. R.K. Richards, *Arithmetic Operations in Digital Computers*, D. Van Nostrand, Princeton, N.J., 1955, pp. 174-176.
4. B. Gold and C.M. Rader, *Digital Processing of Signals*, McGraw-Hill, New York, 1969.
5. C.J. Weinstein, "Quantization Effects in Digital Filters," M.I.T. Lincoln Lab. Tech. Rep. 468, Nov. 21, 1969, AD 706862.
6. A.V. Oppenheim and C.J. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform," *Proc. IEEE* **60**, 957-976 (1972).
7. J.B. Knowles and R. Edwards, "Effect of a Finite-word-length Computer in a Sampled-data Feedback System," *Proc. Inst. Elec. Eng.* **112**, 1197-1207 (1965).
8. J.B. Knowles and E.M. Olcayto, "Coefficient Accuracy and Digital Filter Response," *IEEE Trans. Circuit Theory* **CT-15**, 31-41 (March 1968).
9. L.B. Jackson, "On the Interaction of Roundoff Noise and Dynamic Range in Digital Filters," *Bell Syst. Tech. J.* **49**, 159-184 (Feb. 1970).
10. J.W. Cooley and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math. Comput.* **19**, 297-301 (1965).
11. G-AE Subcommittee on Measurement Concepts, "What is the Fast Fourier Transform?" *IEEE Trans. Audio Electroacoust.* **AU-15**, 45-55 (June 1967).
12. J.W. Cooley, P.A.W. Lewis, and P.D. Welch, "Historical Notes on the Fast Fourier Transform," *IEEE Trans. Audio Electroacoust.* **AU-15**, 76-79 (June 1967).
13. W.M. Gentleman and G. Sande, "Fast Fourier Transforms—For Fun and Profit," in *Proc. Fall Joint Computer Conf., AFIPS Conf. Proc.*, pp 563-578, 1966.
14. T.H. Glisson, C.I. Black, and A.P. Sage, "The Digital Computation of Discrete Spectra Using the Fast Fourier Transform," *IEEE Trans. Audio Electroacoust.* **AU-18**, 271-287 (Sept. 1970).
15. P.D. Welch, "A Fixed-point Fast Fourier Transform Error Analysis," *IEEE Trans. Audio Electroacoust.* **AU-17**, 151-157 (June 1969).
16. T. Kaneko and B. Liu, "Accumulation of Roundoff Error in Fast Fourier Transforms," *J. Assoc. Comput. Mach.* **17**, 637-654 (Oct. 1970).
17. T. Kaneko, "Accuracy Problems of Digital Signal Processing," Ph.D. dissertation, Princeton University, Dept. Elec. Eng., Princeton, N.J., Feb. 1970.
18. K. Steiglitz, "The Equivalence of Digital and Analog Signal Processing," *Inform. Contr.* **8**, 455-467 (1965).
19. J.F. Kaiser, "Digital Filters," in *System Analysis by Digital Computer*, John Wiley & Sons, New York, 1966, pp 218-285.
20. L.B. Jackson, "Roundoff-noise Analysis for Fixed-point Digital Filters Realized in Cascade or Parallel Form," *IEEE Trans. Audio Electroacoust.* **AU-18**, 107-122 (June 1970).

21. A.V. Oppenheim, "Realization of Digital Filters Using Block-floating-point Arithmetic," *IEEE Trans. Audio Electroacoust.* AU-18, 130-136 (June 1970).

Appendix A
 BAD-BIT DISTRIBUTIONS FOR THE FFT

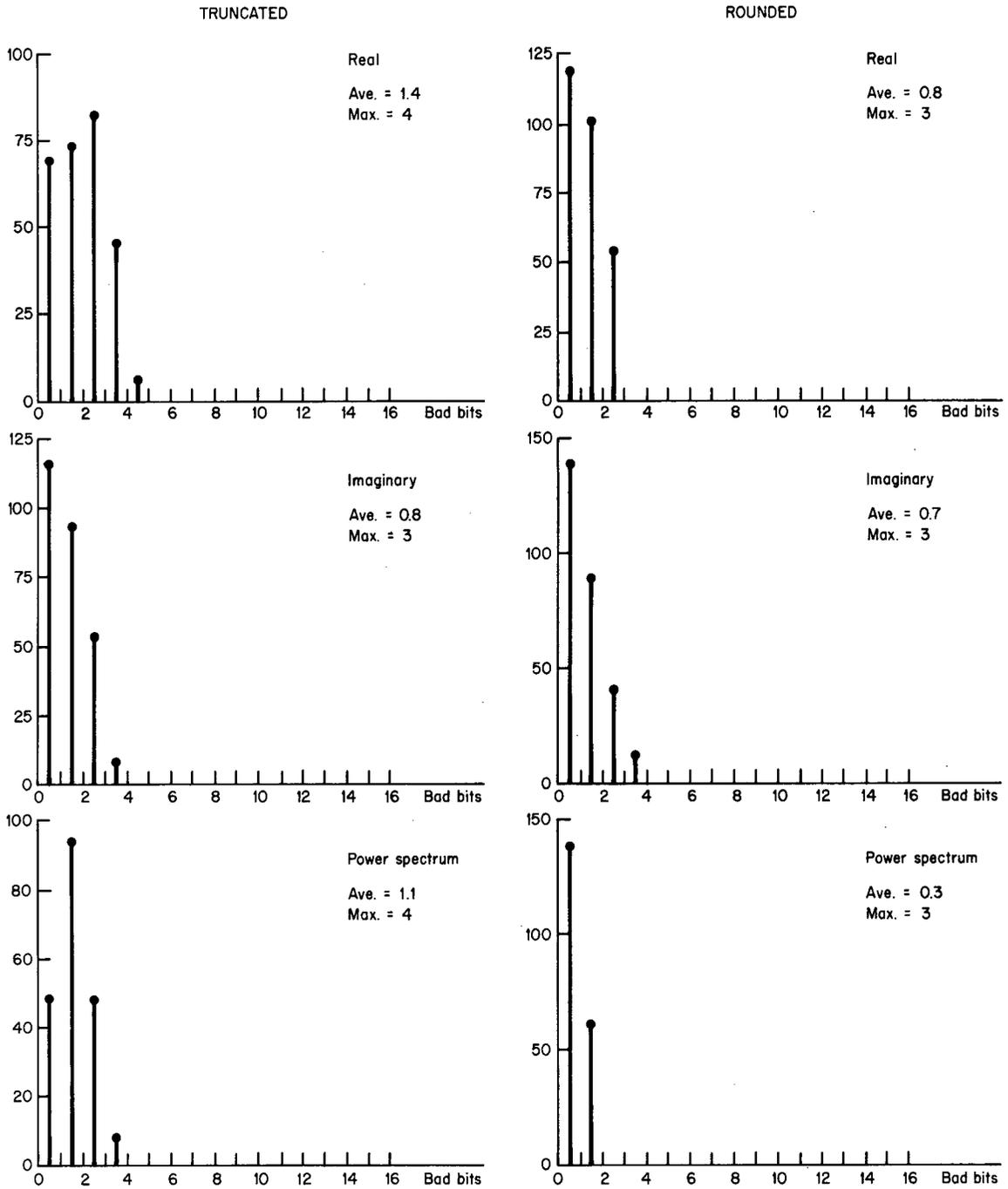


Fig. A1—Bad-bit distribution for $N = 256$, sinusoidal input

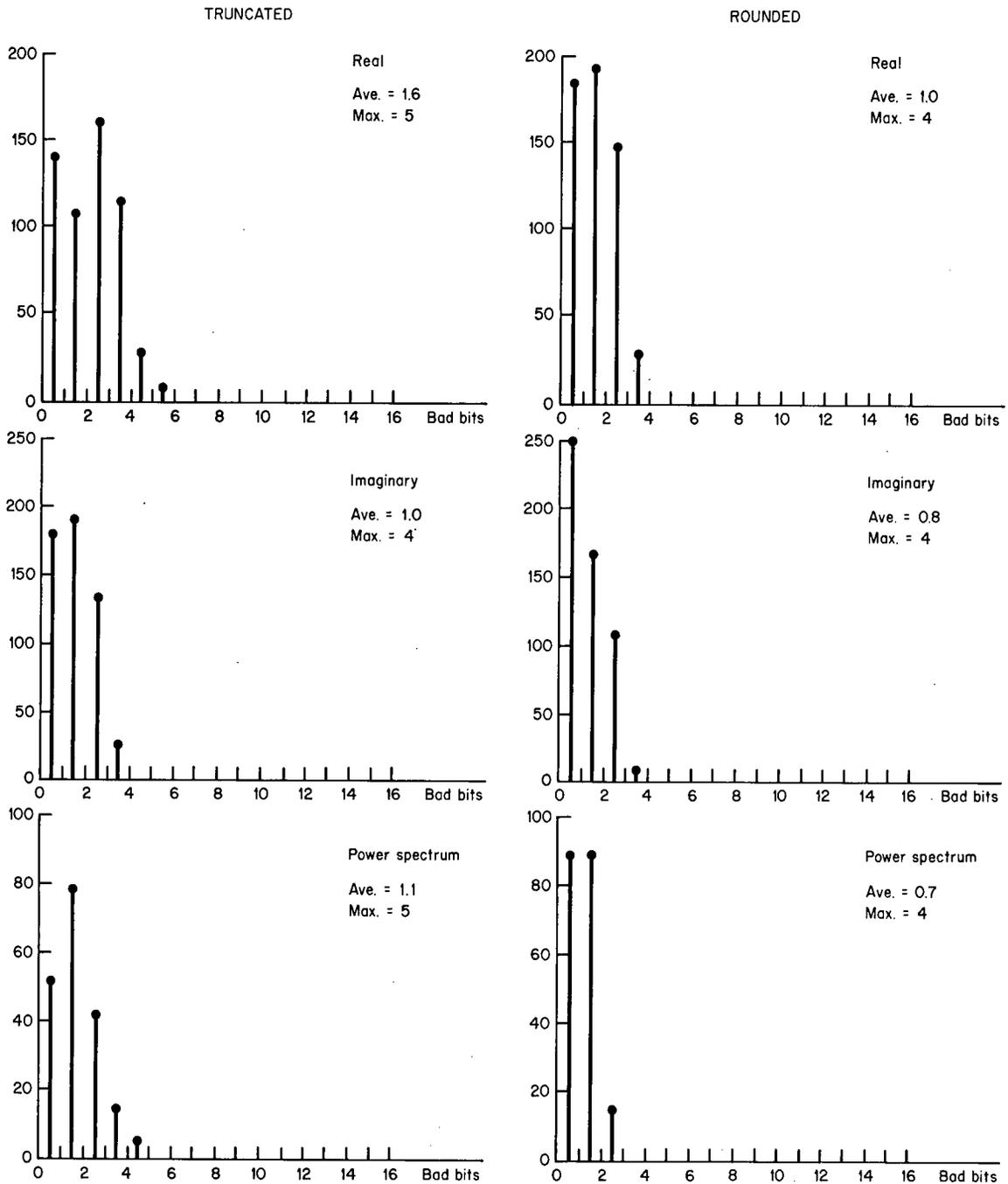


Fig. A2—Bad-bit distribution for $N = 512$, sinusoidal input

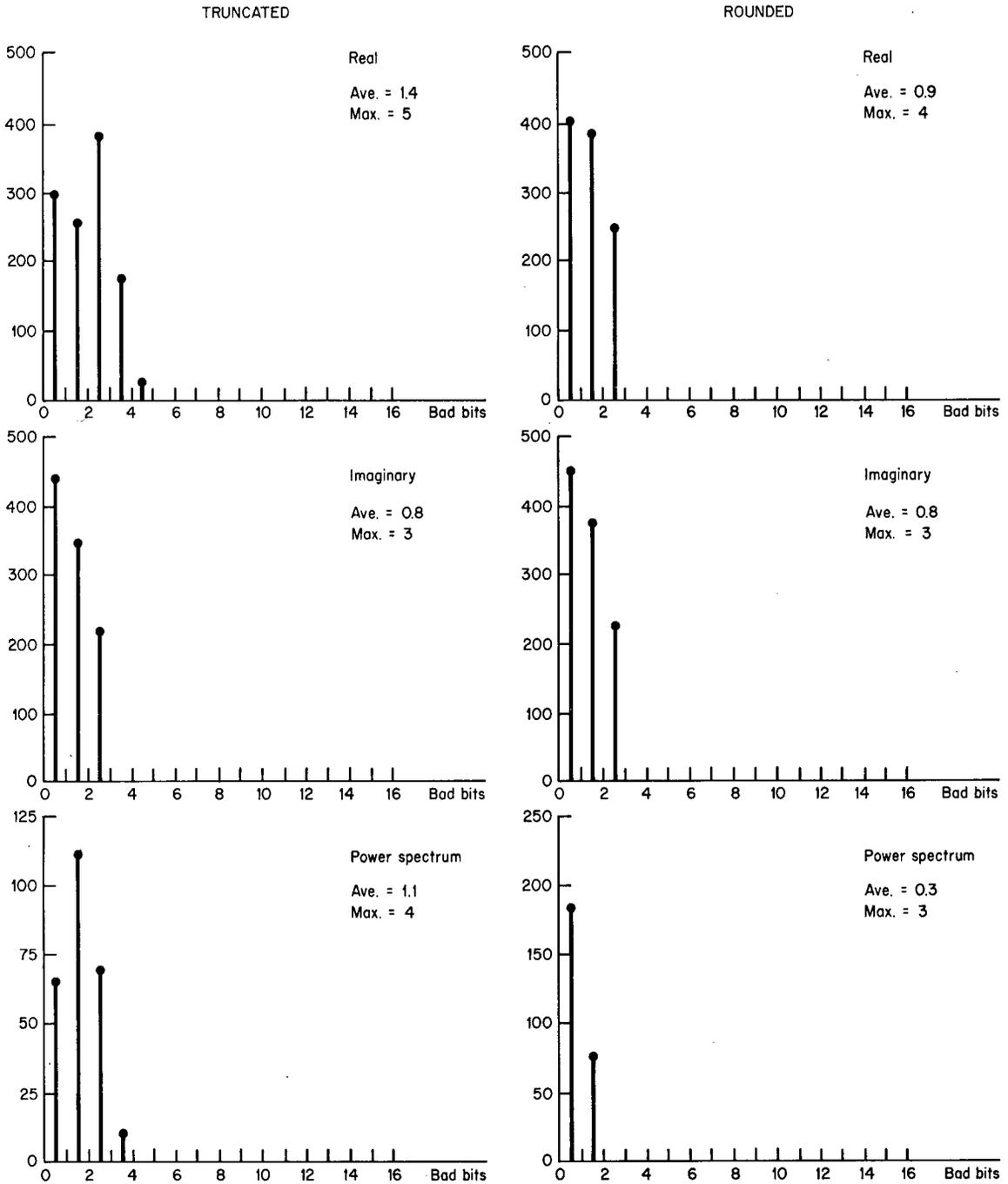


Fig. A3—Bad-bit distribution for $N = 1024$, sinusoidal input

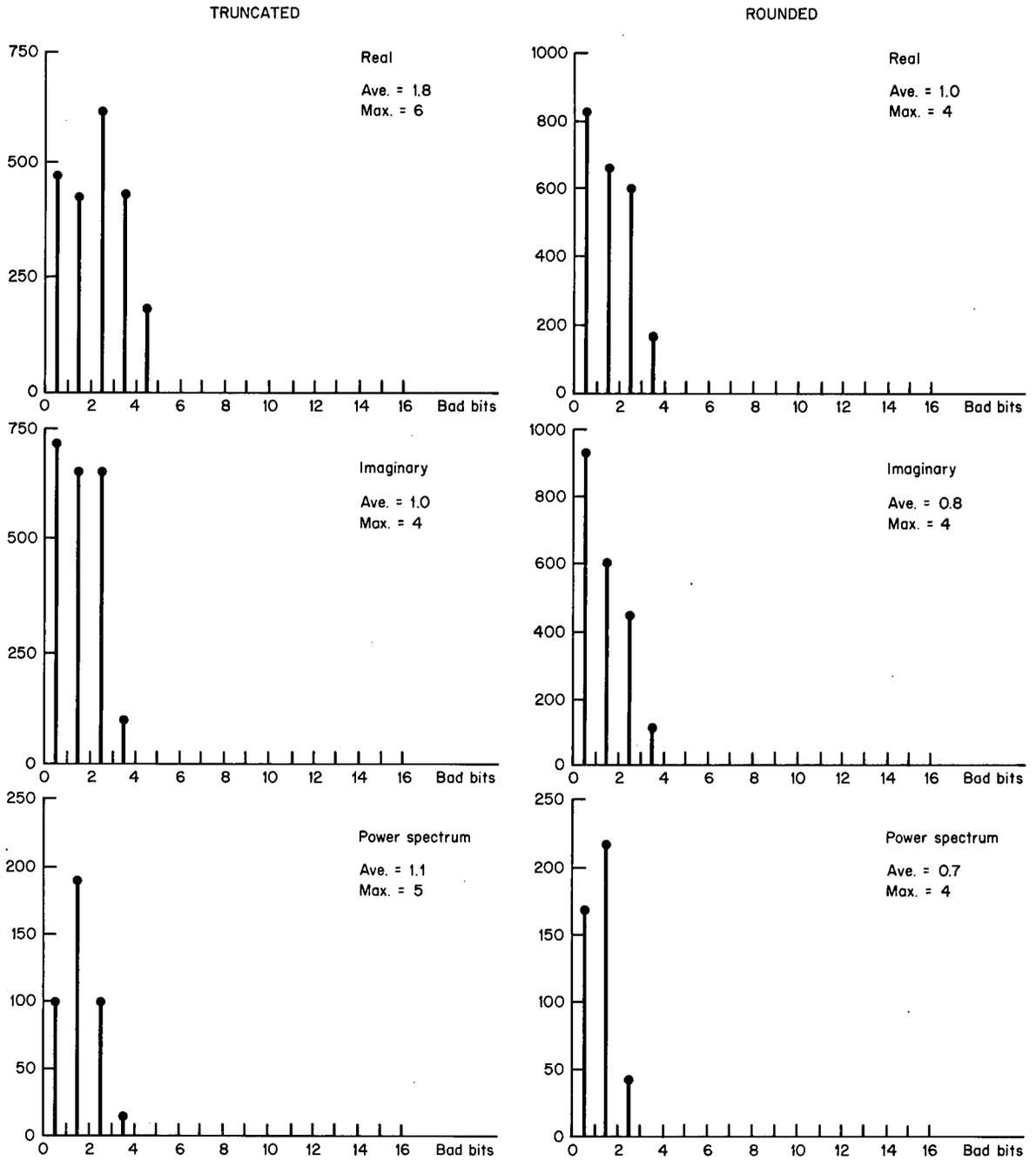


Fig. A4—Bad-bit distribution for $N = 2048$, sinusoidal input

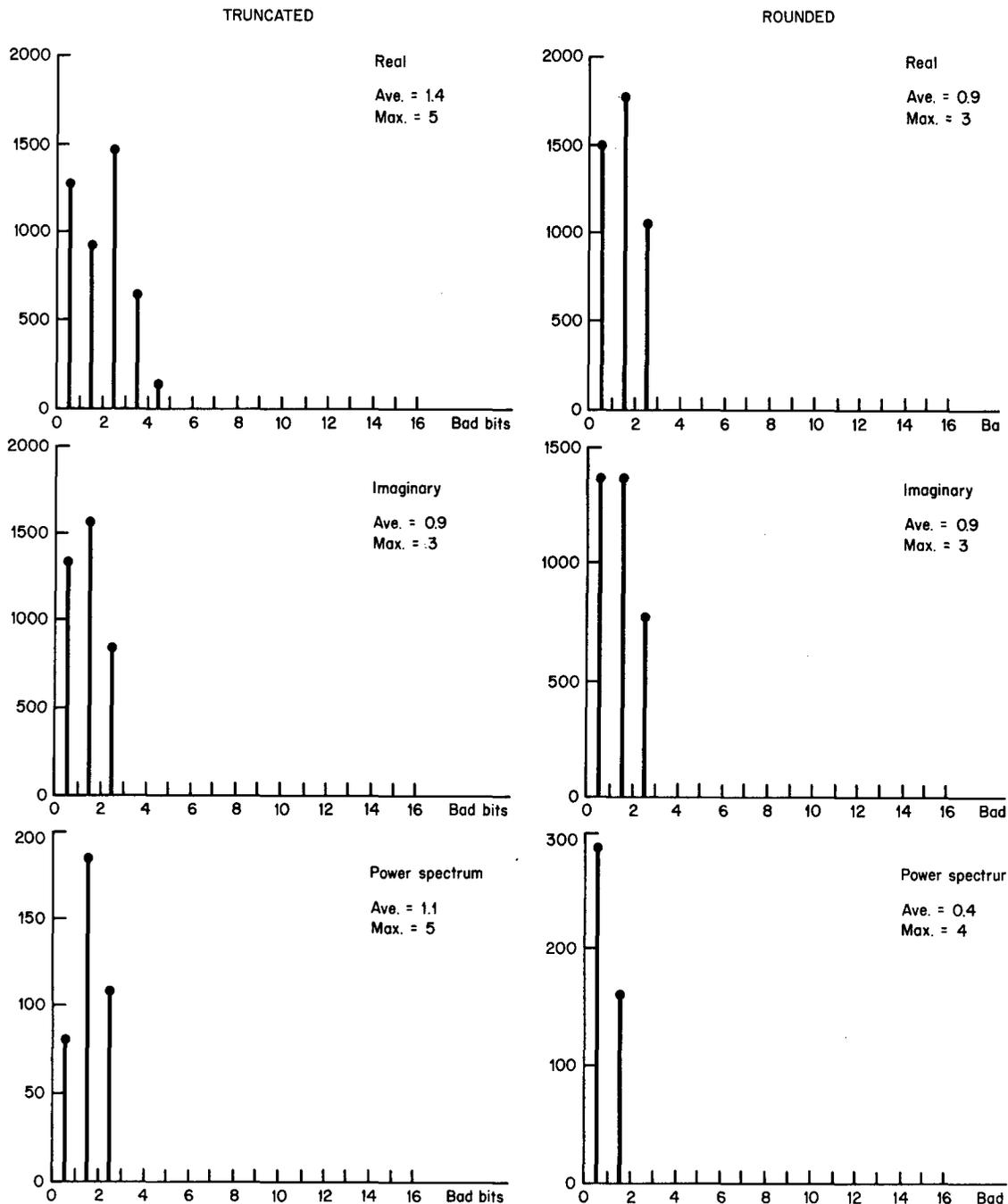


Fig. A5—Bad-bit distribution for $N = 4096$, sinusoidal input

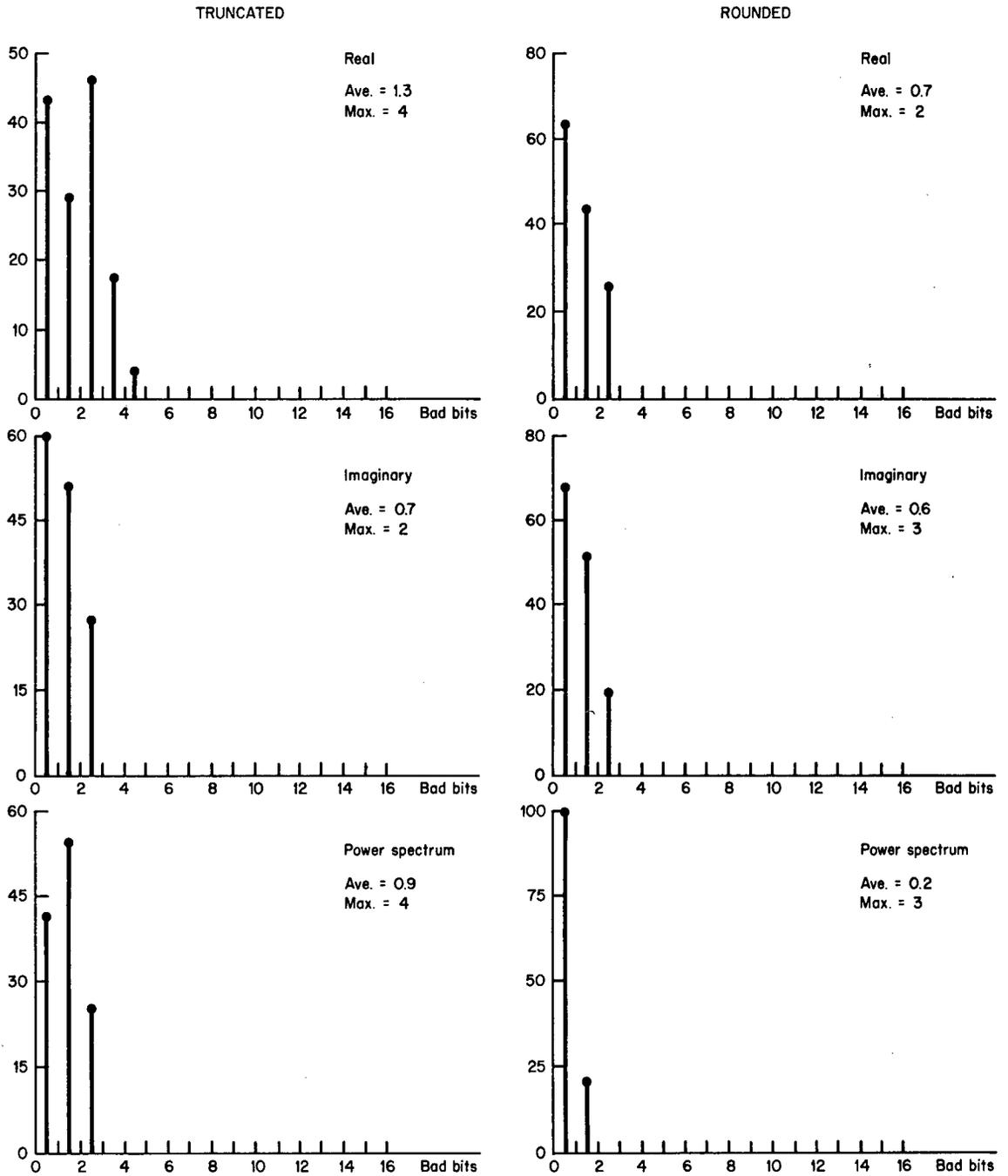


Fig. A6—Bad-bit distribution for $N = 128$, random input

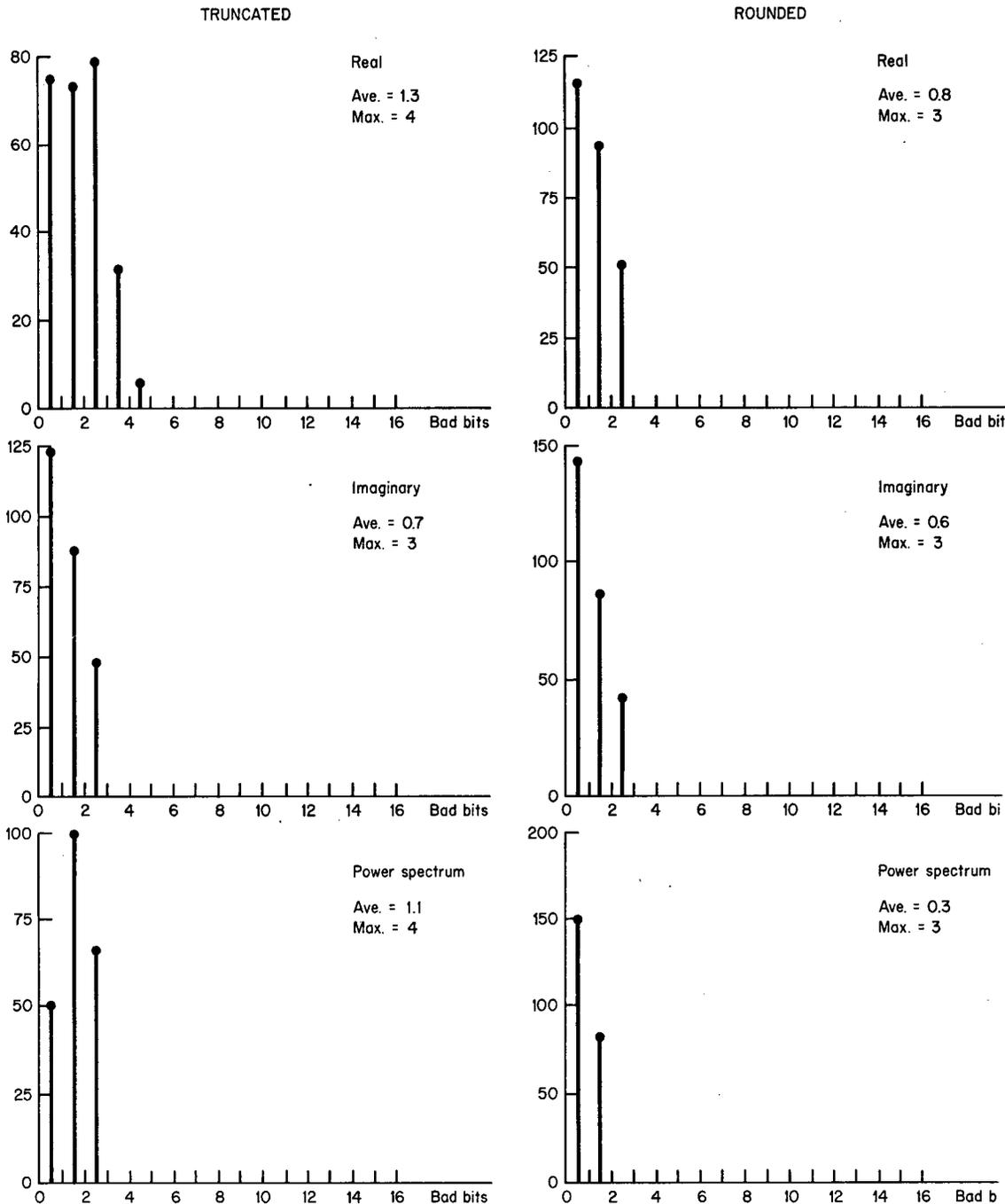


Fig. A7—Bad-bit distribution for $N = 256$, random input

JUDITH N. FROSCHER

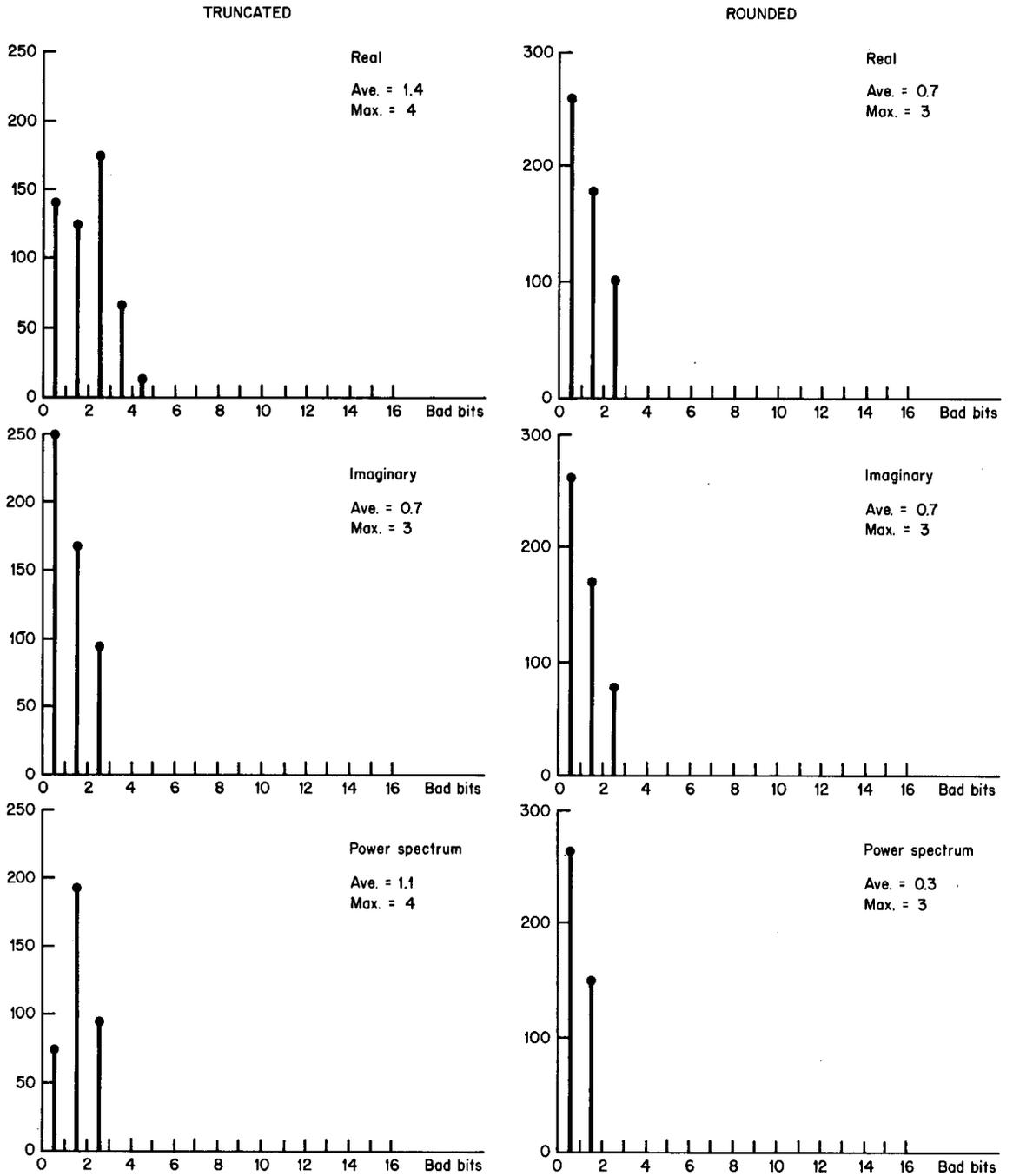


Fig. A8—Bad-bit distribution for $N = 512$, random input

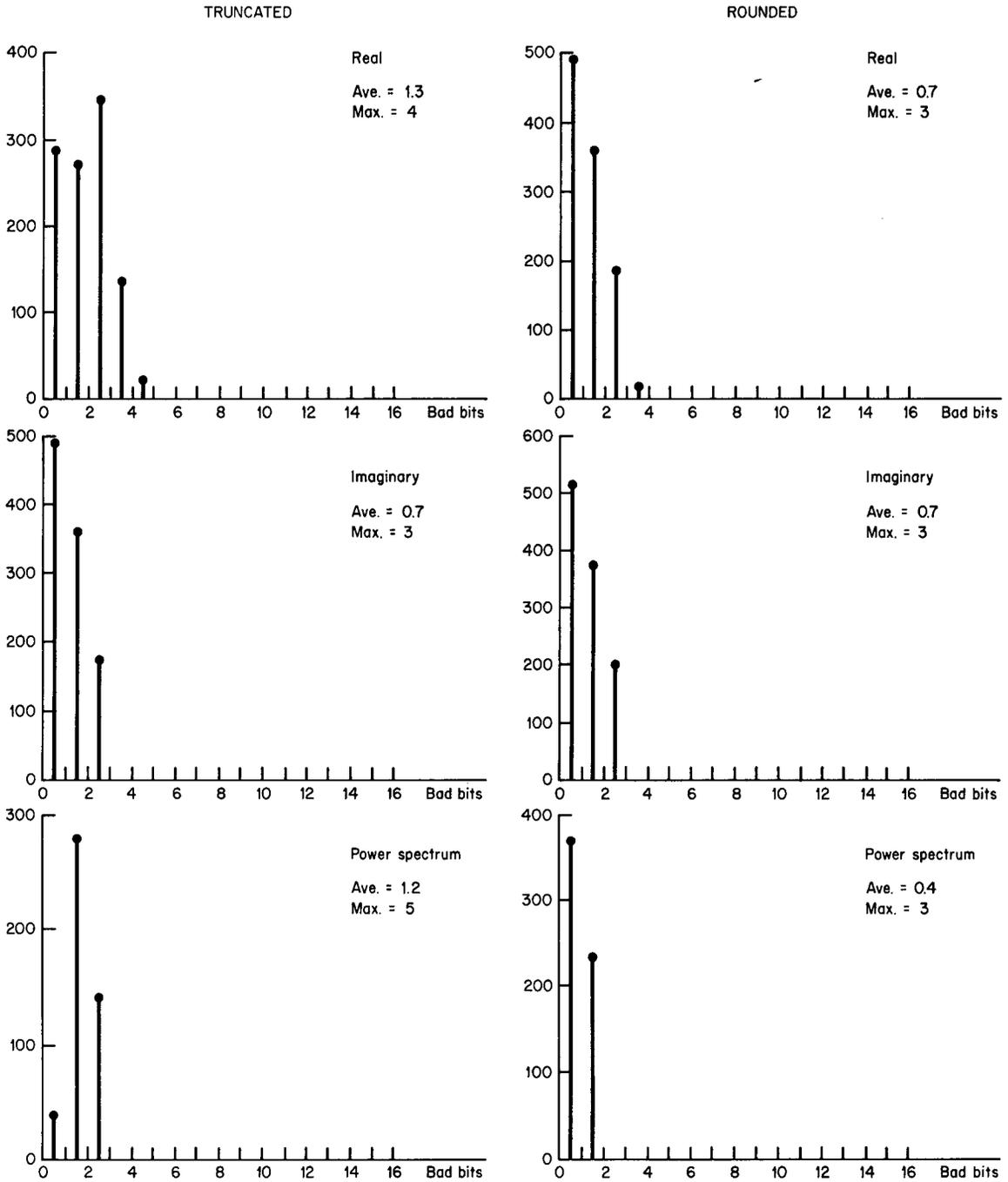


Fig. A9—Bad-bit distribution for $N = 1024$, random input

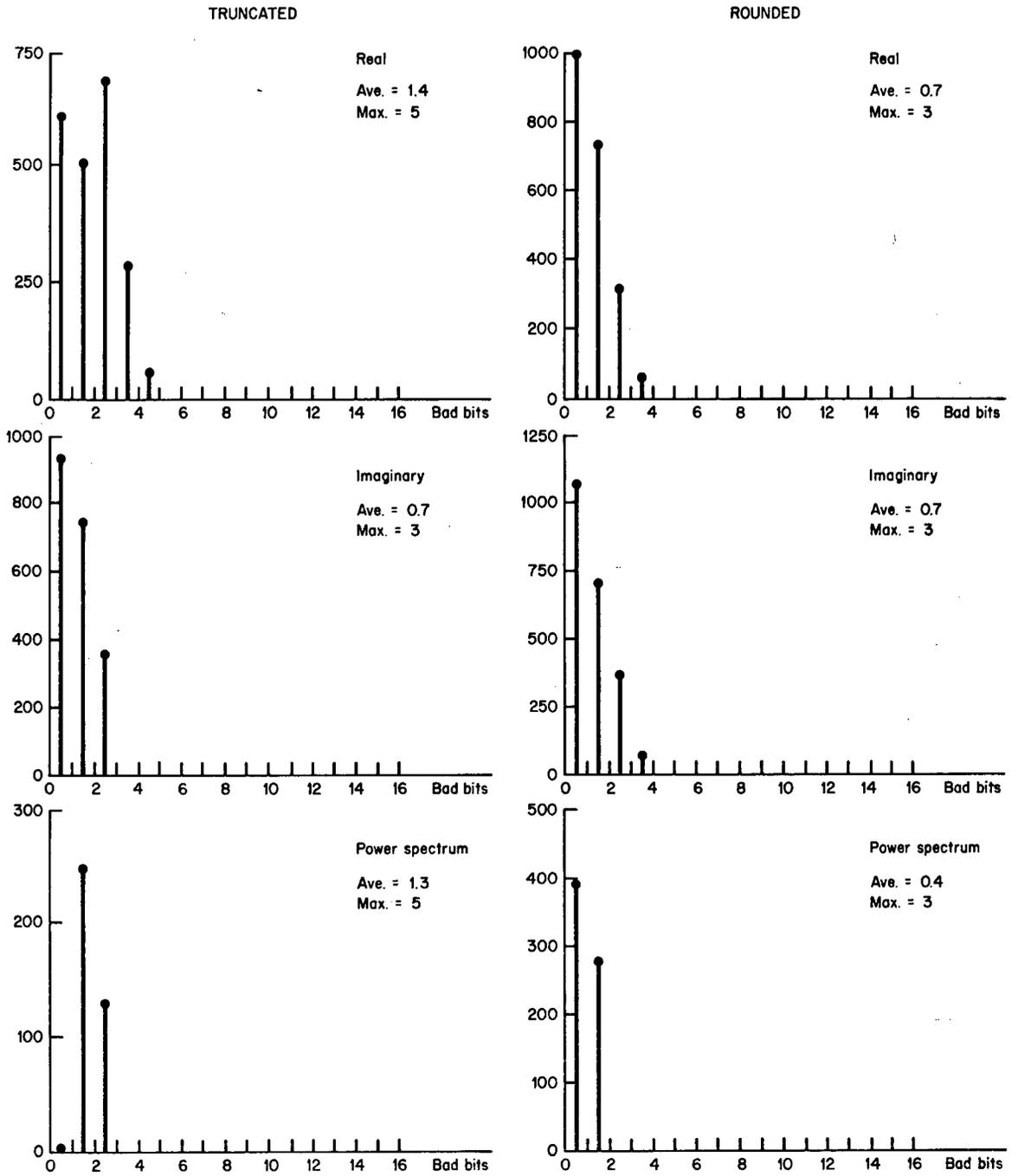


Fig. A10—Bad-bit distribution for $N = 2048$, random input

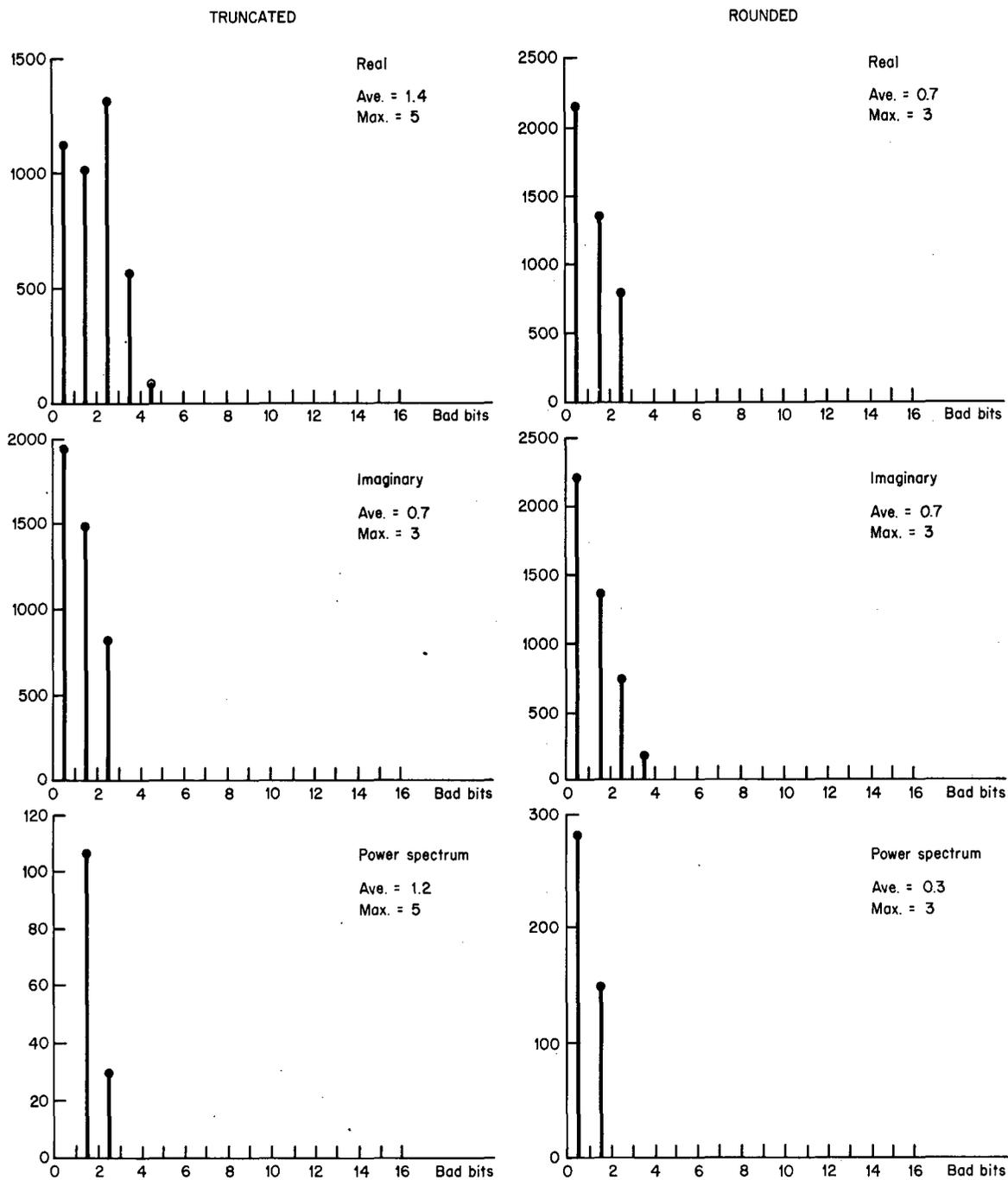
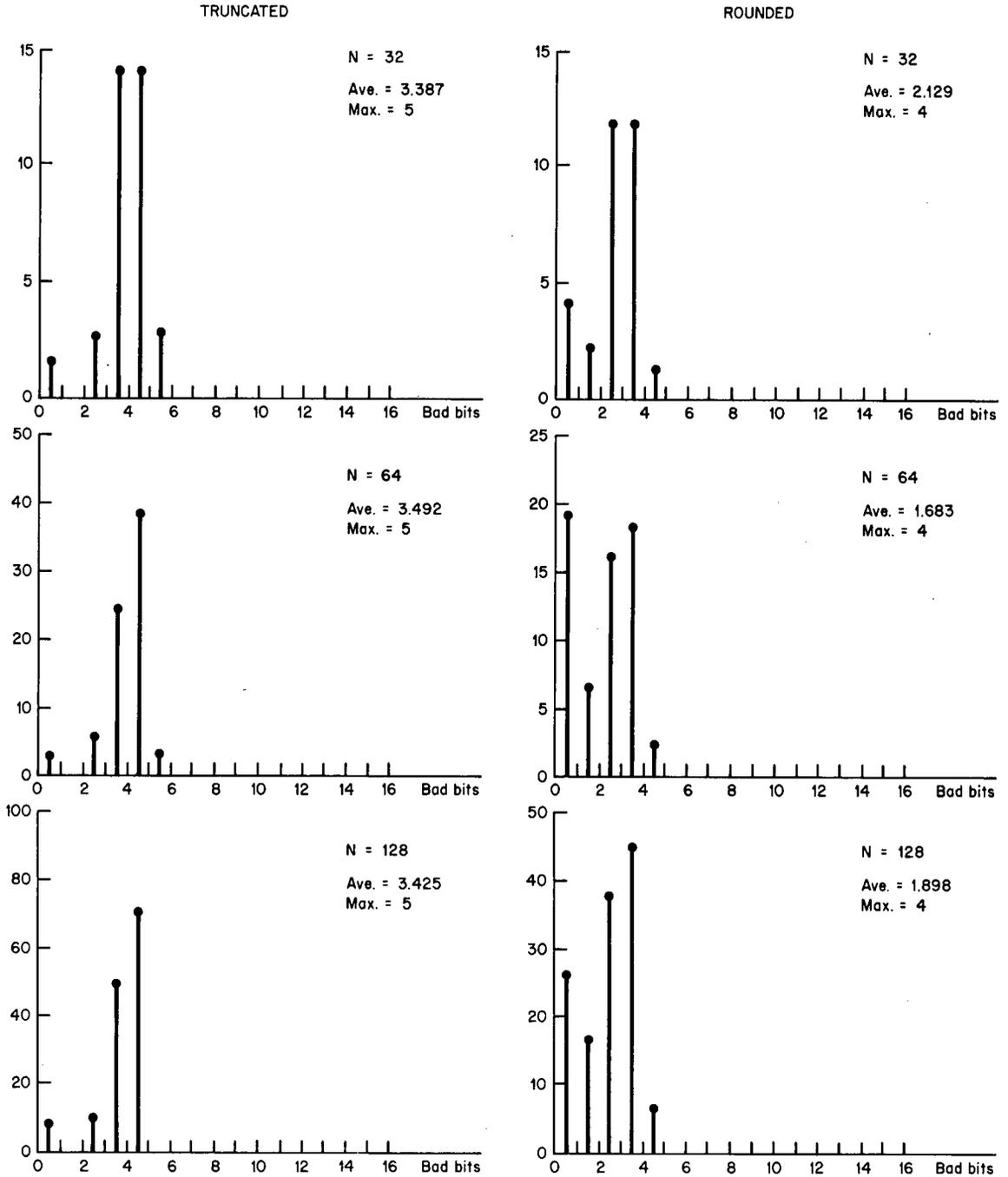


Fig. A11—Bad-bit distribution for $N = 4096$, random input

Appendix B
BAD-BIT DISTRIBUTIONS FOR THE FILTERING MACRO



B1a—Bad-bit distribution for a sinusoidal input

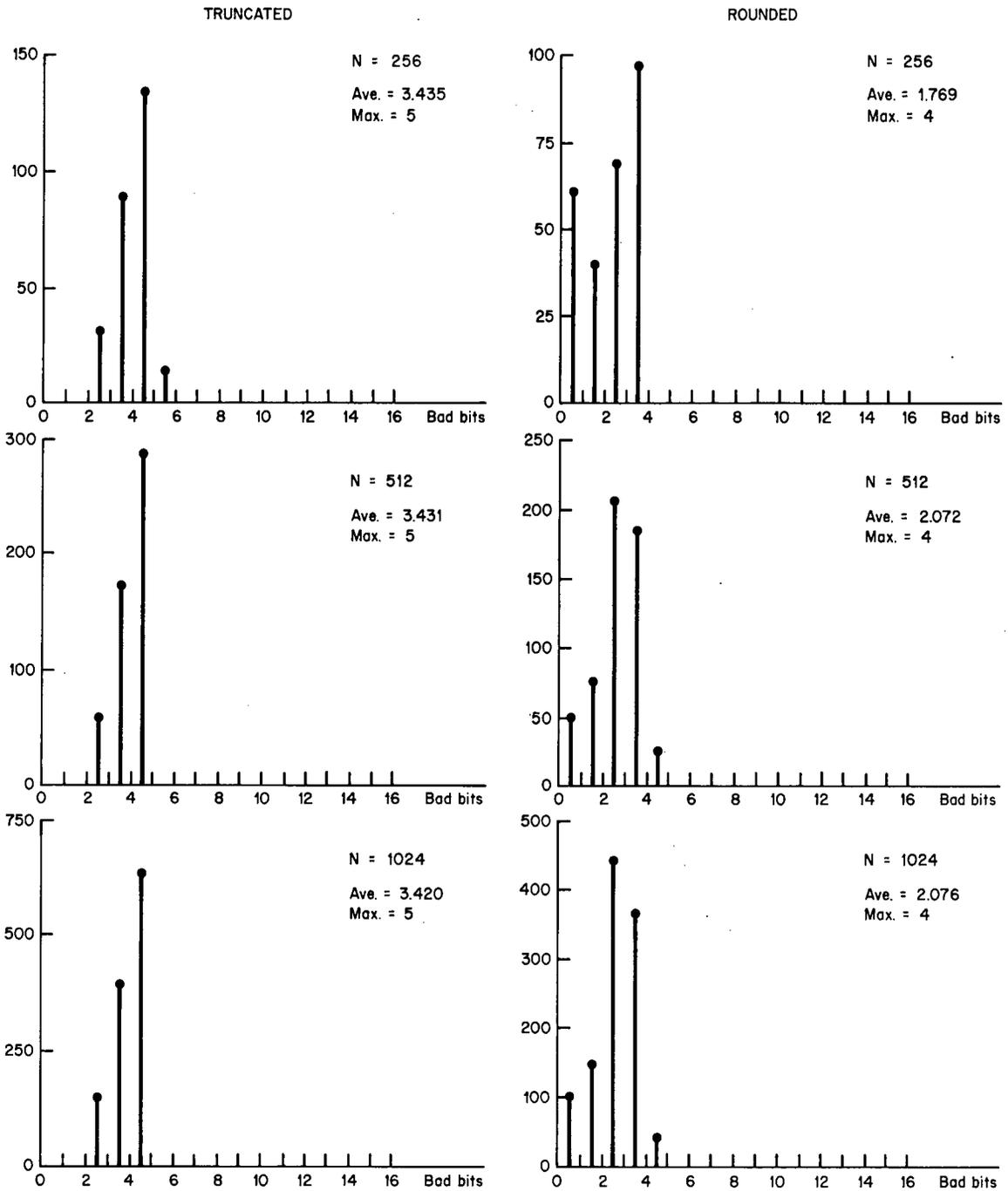


Fig. B1b—Bad-bit distribution for a sinusoidal input (Continued)

JUDITH N. FROSCHER

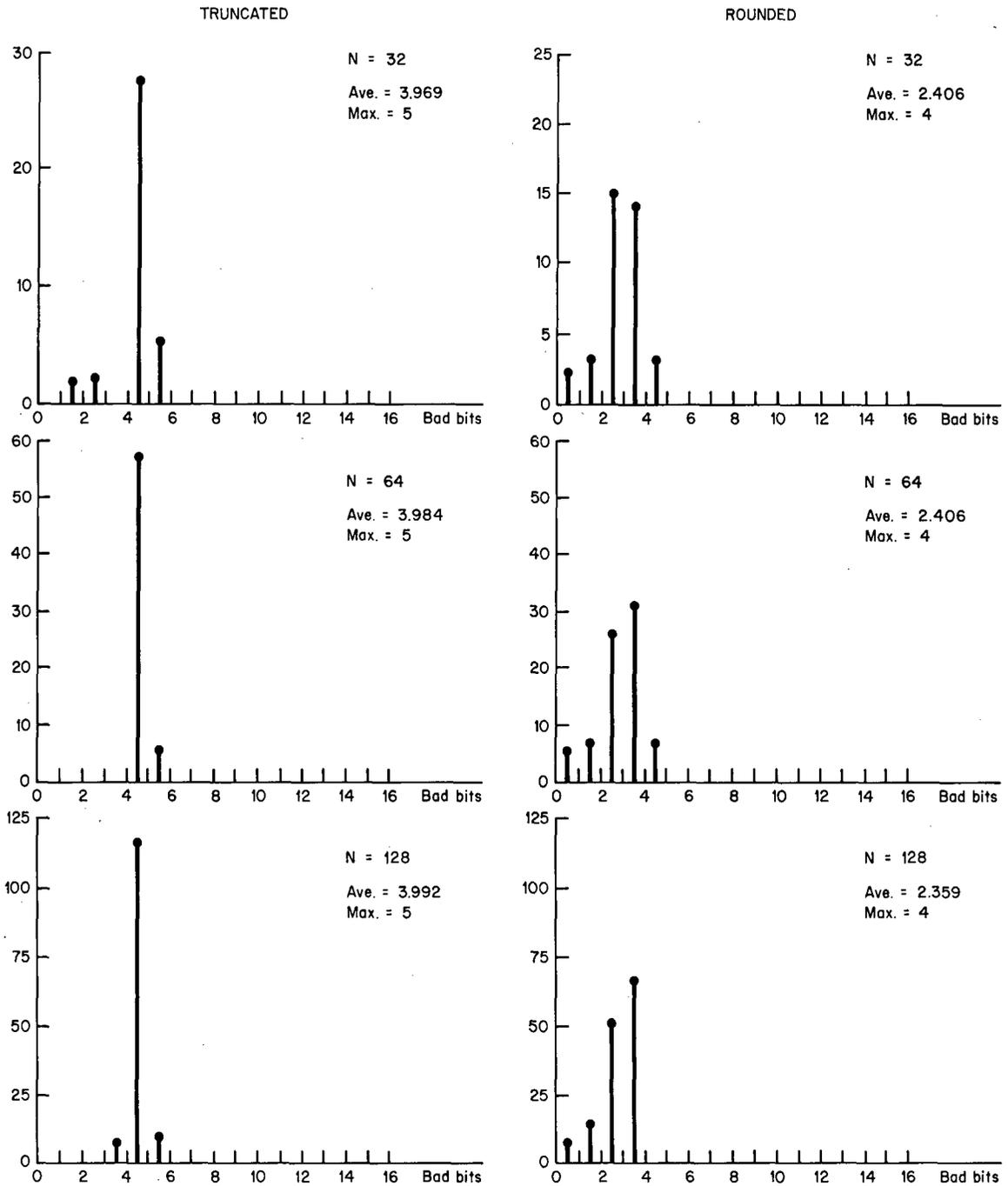


Fig. B2a—Bad-bit distribution for random input

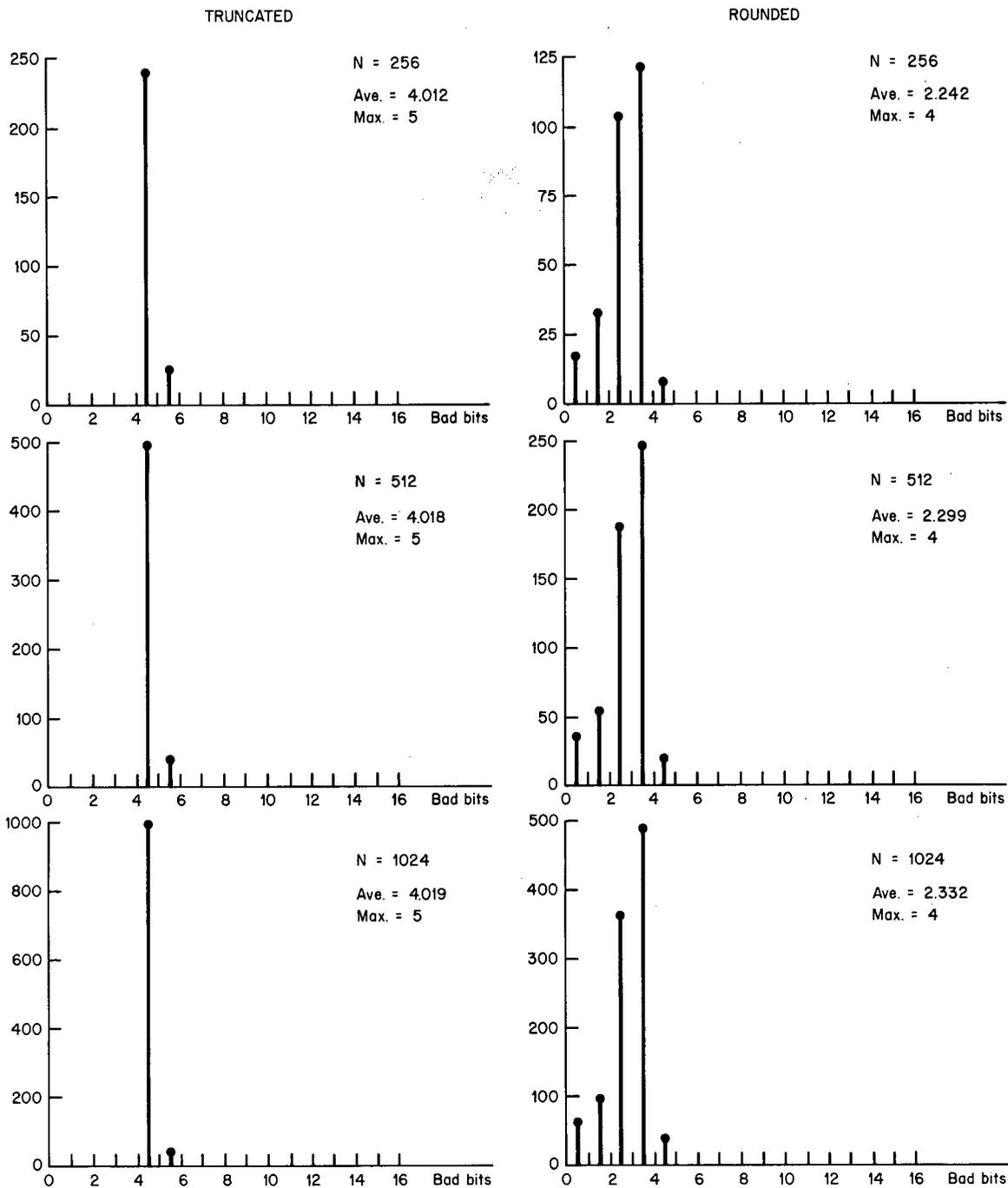


Fig. B2b—Bad-bit distribution for random input (Continued)