IBM Journal
of research and development

Vol. 28, No. 2, pp. 123–230, March 1984

Thomas D. Howell

Analysis of Correctable Errors in the IBM 3380 Disk File

A method of analyzing the correctable errors in disk files is presented. It allows one to infer the most probable error in the encoded-data stream given only the unencoded readback and error-correction information. This method is applied to the errors observed in seven months of operation of four IBM 3380 head-disk assemblies. It is shown that nearly all the observed errors can be explained as single-bit errors at the input to the channel decoder. About 90 percent of the errors were related to imperfections in the disk surfaces. The remaining 10 percent were mostly due to heads which were unusually susceptible to random noise-induced errors.

Introduction

Disk files such as the IBM 3380 use error-correcting codes to achieve very high data reliability. The error-correcting code for a disk file must be chosen long before any experience is available which would give insight into the kinds of errors which must be corrected by the code. The code must be chosen on the basis of experience with earlier files, test results with development-level components, and extrapolation. This paper attempts to look back at the errors corrected by the code in the finished product, and to see how well the code matches the actual error statistics and what lessons can be learned and applied to the next generation.

Disk files also use channel codes to control intersymbol interference and to make the data self-clocking. Errors are first introduced into the channel encoded form of the data, but are not visible to the user until after decoding. The channel decoding process masks the nature of the errors.

In this paper, a method is presented for analyzing the correctable error data available to the disk user in order to determine the encoded form of each error. This method is new, and it allows detailed study of the actual correctable errors seen in disks while they are in use. This method is then applied to the correctable error data gathered automatically by several 3380s over an extended period of time. The results of this analysis lead to some interesting conclusions about the nature of the errors, their causes, the channel code, and the error-correcting code.

The first section describes the method for analyzing the correctable errors. Next the experiment is described. The results of the analysis are given, followed by conclusions.

Analysis of errors

The novel feature of this paper is the analysis of the errors in terms of the encoded readback data produced by the data detector. These encoded bits pass through a channel decoder before the error-correcting-code decoder. Error propagation in the channel decoder converts simple errors such as missing bits and shifted bits into slightly more complicated errors. In the case of the IBM 3380, the channel code is a (2,7) run-length limited code [1, 2], and its channel decoder propagates each error in the encoded bit string into at most four errors in the decoded string. A k-bit burst of errors propagates to at most $k + 3$ bits. We shall see that k is almost always 1 or 2. Only the decoded data strings before and after error correction are available to the user. Details of the encoder and decoder can be found in the Appendix. The analysis method is presented here for the specific case of the (2,7) code. The method can be adapted to any of the codes commonly used in magnetic recording.

We shall make the assumption that the most likely errors are the ones which involve the fewest "mistakes" by the detector. The detector detects a 1-bit by a peak in the readback signal and a 0-bit by the absence of a peak. Detector mistakes include shifted bits, missing bits, and extra bits. In this context the word "bit" is an abbreviation for 1-bit. Shifted bits are caused by readback peaks being shifted in time. They are characterized by bits 0 1 0 being read back as 1 0 0 or 0 0 1. Missing bits and extra bits are caused by missing peaks and extra peaks in the readback signal. They are characterized by bits 0 1 0 being read back as 0 0 0 (missing bit), and 0 0 0 being read back as 0 1 0 (extra bit). Since the error rate of the 3380 is very low, error patterns with one mistake would be

very much more likely than patterns with more mistakes if mistakes were independent events. The assumption made for this analysis is that the most likely explanation of a given readback error is the one with the fewest detector mistakes. In particular, errors which can be explained by single mistakes probably were caused by single mistakes.

The analysis is best explained by means of examples. Consider the case in which the data written were (in hexadecimal) 20904824, and the received data before error correction were 7C904824. We can simply encode each segment using the (2,7) code: The written data encode to

2412490492412490,

and the received data encode to

2212490492412490.

This is an example of a shifted bit; the second 1-bit shifted one position to the right:

written: ··· 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 ···

received: ··· 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 ···.

A second example shows that the situation is not always so simple. Suppose the written data were 33A678AF19 and the received data were 33A620AF19. Encoding both, we find a large region of disagreement:

written: ··· 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 ···

received: ··· 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 ···.

It is extremely unlikely for the detector to be that badly in error. At least four mistakes are required to produce the hypothetical received pattern shown: two left-shifted bits, one extra bit, and one right-shifted bit. A much more likely explanation for the received string, 33A620AF19, is that the detector made a simple error which produced a string of bits not in the range of the (2,7) encoder. The following is the most likely such string as measured by the number of mistakes:

received: ··· 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 ···.

This string decodes to 33A620AF19, which is the received data string. It differs from the written data by a single mistake, a left-shifted bit.

The general problem is illustrated in **Figure 1.** Let $E(\cdot)$ be the (2,7) encoder function on binary strings of length n bits. In our case, $n = 8 \times 47\,467$. Let $D(\cdot)$ be the (2,7) decoder function. The domain of E is $S^n = \{0,1\}^n$ and the range of E is a subset R of $S^{2n} = \{0,1\}^{2n}$. The decoder D is the inverse of E as long as it operates on elements of R; D maps R one-to-one onto S^n. However, the domain of D is all of S^{2n}; D does something with any string of bits it is given. Thus D is a many-to-one mapping of S^{2n} onto S^n. For any y in S^n, there is a subset $P(y)$ of S^{2n} of preimages of y under D, $P(y) = \{z \mid D(z) = y\}$. Our problem may now be stated as follows. Given the written data $x \in S^n$ and the received data $y \in S^n$, choose from $P(y)$ the preimage z which was most likely to have been produced from $E(x)$ by errors in the write/read process.

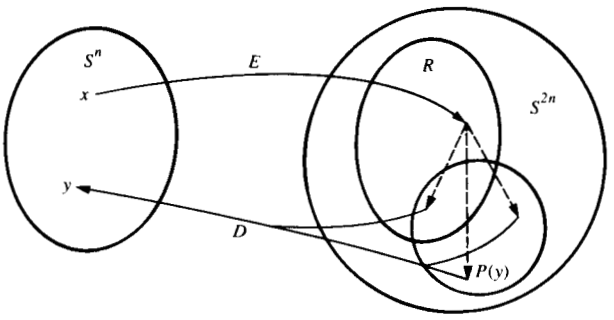

Figure 1 Domains and ranges of the (2,7) encoder and decoder.

The analysis procedure is to search for elements of $P(y)$ which differ from $E(x)$ by a single mistake. This can be done by enumerating elements z of S^{2n} which differ by single mistakes from $E(x)$. For each such z, $D(z)$ must be computed and compared with y. The search can be simplified by enumerating only those z which differ from $E(x)$ in approximately the right place. The decoding procedure given in the Appendix shows that only seven bits of an encoded string z affect a given bit of $D(z)$. If $y = D(z)$ and $y_k \neq x_k$, then at least one of the seven bits of z affecting y_k must differ from the corresponding bit of $E(x)$. It is necessary to enumerate only those patterns z differing from $E(x)$ by a single mistake affecting at least one of these seven bits.

For each string z enumerated by this procedure one must test whether $D(z) = y$. It is possible to avoid decoding the whole 47 467 bytes of z. Each y_j is a function of seven z_k. At most two of these z_k are different from the corresponding bits of $E(x)$. The equation $[D(z)]_j = y_j$ must be checked only for those indices j for which either $y_j \neq x_j$ or the seven bits of z which determine y_j include one or both of the bits which differ from the corresponding bits of $E(x)$. The equation holds for other values of j because $D[E(x)] = x$.

Some errors can result in two or more ways from single detector mistakes. One might look like either an extra bit or a right-shifted bit, for example. These errors are called ambiguous. A good way to resolve these ambiguities is to use a computer model of the readback process such as the one described in [3] to determine the most likely of the alternative explanations for each ambiguous error. The model requires detailed information on the shape of the readback pulse. This is normally obtained by laboratory measurements. Even a very rough approximation of the pulse shape is sufficient to resolve many ambiguities if measured pulses are not available.

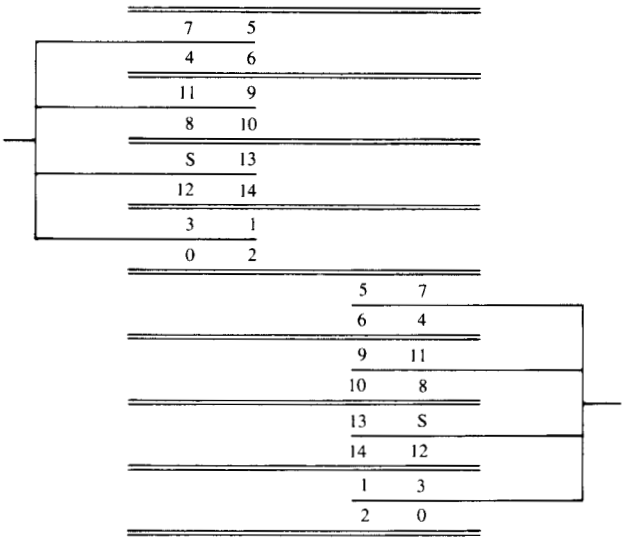

Figure 2 Configuration of 3380 head/disk assembly.

Experiment

The data upon which this report is based were gathered over a seven-month period during which up to four 3380 head-disk assemblies were operated 24 hours per day. The experiment was designed to determine how each of several factors affects the numbers and types of correctable errors observed. The factors studied were data pattern, storage location, frequency of rereading, and frequency of rewriting.

A head-disk assembly has two actuators occupying an even and an odd address. Each actuator has 15 data heads and one servo head, two heads on each of eight recording surfaces. A recording surface has two concentric bands of 885 tracks each. **Figure 2** shows the head configuration for a 3380 head-disk assembly. Heads 0, 3, 4, 7, 8, 11, and 12 are on the outer band; heads 1, 2, 5, 6, 9, 10, 13, and 14 are on the inner band. A cylinder is the set of 15 tracks accessible at a given time without moving the actuator.

Each head-disk assembly was exercised in the same way. Up to four were running simultaneously. The unit (actuator) on the even address was stationary on cylinder 400, and the actuator on the odd address was exercised as follows. The cylinders were divided into four groups of 200 with the remaining 85 cylinders not used. On units 1 and 3 the groups were 0–199, 200–399, 400–599, and 600–799. On units 5 and 7 the groups were 685–884, 485–684, 285–484, and 85–284. Cylinder 0 is at the largest radius, and cylinder 884 is at the smallest radius within each band. One iteration on a group of cylinders consisted of processing each cylinder in the group, followed by processing cylinder 400 on the stationary actuator. The first group was processed for 100 iterations, then the second group for 10 iterations, then the third and fourth groups for one iteration each. The highest activity groups were at the outside on units 1 and 3 and at the inside on units 5 and 7.

Processing a cylinder consisted of reading all 15 tracks, then rewriting six of the 15 tracks, then reading all 15 tracks again. The six heads which rewrote their data were 3, 4, 6, 8, 9, and 14, three on the inner band and three on the outer band.

The data always consisted of a single record of 47 467 bytes. This record had four approximately equal-length segments. The first segment was chosen to maximize peak-shift errors, the second to maximize missing-bit errors, the third maximized transition density, and the fourth was a standard test pattern containing several kinds of high-stress conditions. The patterns used are given in **Table 1**.

Following each correctable error in a data field, the processor issues a SENSE command. The control unit responds by sending back information about the error location and the correction information. These sense data were time stamped and saved in files for analysis.

For the last three months of the experiment, unit 4 was used for a special test. It was not held stationary, but its read/write activity was still minimal. Only three errors were observed on the even-numbered actuators, all on unit 4. In the rest of the discussion, we ignore the even-numbered units, and concentrate on units 1, 3, 5, and 7.

The number of bytes processed in one iteration on a group of 200 cylinders was $47\,467 \times 36 \times 200 = 341\,762\,400$. This took about nine minutes elapsed time. The entire cycle consisting of 112 iterations took about 16 hours when all the units were operating.

The 3380 error-correcting code corrects all error bursts which are contained within two consecutive 16-bit blocks. This includes all bursts of 17 or fewer bits and some as long as 32 bits. It can correct one such burst per record. Records with uncorrectable errors are reread several times with and without head offset. Only if this procedure fails is a permanent error reported. No permanent errors were reported during this experiment. Rereads were not logged during the experiment and are not reported to the program, so no information is available about the number of errors recovered by rereads.

Results

All errors analyzed in this study were correctable errors. Each event requiring use of the error-correcting code was counted as one error, whether it involved one bit or several.

In the seven months of operation 18 753 errors were corrected by the error-correcting code. Of these, 99.8 percent

Table 1 The data pattern segments (in hexadecimal code).

Segment name	*One period: User data in hexadecimal, Run lengths of 0s in encoded data*	*Repeated for*
Peak shift	AF19E33A678, 32362736274347363	11 869 bytes
Missing bit	412090482, 25222252222522225222	11 871 bytes
High frequency	492, 22222222	11 871 bytes
Complex	2492AAAAAAAAA925F03B91225DC88BBB, 22222222233333333333333333333222222 23456424224222422224224224242425	11 856 bytes

Table 2 Error types by head and unit.

Unit \ *Head*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
1	1	1	2	2	0	3	0	1	3	4	3	0	1	3	2
3	9	5	20	5	4	3	6	0	9	4	1	0	0	2	6
4	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
5	3	267	48	1	0	17	8	2	4	3	11	0	1	10	5
7	2	6	5	1	1	45	36	1	2	15	5	0	1	1203	19

were explainable as single bits being changed or shifted at most one bit position. This includes 99.1 percent of the 1836 different types of error.

Some errors were sensed in which no correction was required. These happen when the error location falls in the bytes beyond the end of the data record but before the end of the region protected by the error-correcting code. This region always ends at a multiple of 32 bytes from index. These errors are referred to as 0-bit errors.

The errors were highly clustered at certain sites on the disk surface. This indicates the presence of "minor defects." Only 1836 different errors were observed. Errors are considered different if they differ in physical location or received pattern, i.e., anything except time of occurrence. These different errors are called "error types." Of the 1836 error types, 217 were repeated and 1619 occurred just once. One error type was repeated 5744 times, over 30 percent of the total errors. Over 85 percent of the errors were of the most frequent two percent of the error types. In normal operation, error sites at which several errors have occurred would be marked and subsequently not used to store data. Procedures for doing so are outlined in [4]. This was not done for this test. It would have required rewriting the affected tracks, some of which were in the group designated as read-only for this test. The effect of these maintenance procedures can be approximated by counting repeated errors only once in the analysis. For this reason, most of the results given here include counts of both errors and error types.

Error types were distributed very unevenly across heads; see **Table 2.** The heads on the inner bands, numbers 1, 2, 5, 6, 9, 10, 13, and 14, have far more error types than the heads on the outer bands. The inner tracks are shorter, resulting in a higher recording density, so more errors are to be expected.

Two heads, unit 7 head 13 and unit 5 head 1, account for 1203 and 267 error types, respectively. These two heads are probably marginal performers, making them more susceptible to errors caused by factors such as difficult data patterns and electronic noise. None of the 1470 errors on these two heads were missing bits. It is interesting that these weaker heads were much more susceptible to peak-shift errors than other heads, but not more susceptible to missing-bit errors.

The causes of the errors can be summarized as follows. About 90 percent of the errors and 12 percent of the error

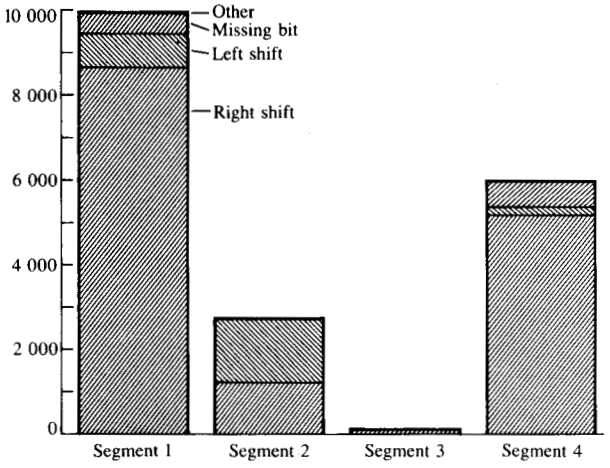

Figure 3 Errors by category for each pattern segment.

types were repeated ones. Most of these were caused by defects. They would have been eliminated by normal maintenance procedures which were not used during this test. Most of the remaining 10 percent of the errors and 88 percent of the error types were caused, at least partially, by two heads.

The errors were classified according to the following criteria: time, frequency of access, whether the track was periodically rewritten, data pattern, track radius, and logical unit. Using the procedure previously described, each error was assigned a category: right-shifted bit, left-shifted bit, missing bit, or other. The category "other" contains 30 0-bit errors, three extra bit errors, and 33 errors of 17 error types which involve more than one bit changing or shifting. These multiple-mistake errors represent about 0.2 percent of the total errors and 0.9 percent of the error types.

Right-shifted bits accounted for 78.9 percent of the errors and 40.6 percent of the error types. Left-shifted bits were 6.7 percent of the errors and 53.9 percent of the error types. Missing bits were 14 percent of the errors and 2.7 percent of the error types.

The pattern segments designed to enhance peak-shift and missing-bit errors did so quite well. This shows that difficult data patterns significantly increase the error probability compared with the average. The bars in **Figure 3** show, from the bottom up, the numbers of right-shift, left-shift, missing-bit, and other errors observed in each pattern segment. Segment 1 is above average in both right and left peak-shifts, while segment 2 is enriched in missing bits. Very few errors happened in segment 3. The complex test pattern, segment 4, is intermediate in all types of errors.

Frequency of rereading and rewriting did not have a large effect on the numbers of errors or error types. The only discernable trend was that on the inner band of tracks the heads which rewrote their data had fewer errors and error types than those which did not rewrite. The outer band had far fewer error types, and rewriting had no significant effect on the numbers of errors or error types.

Conclusions

A method for determining the nature of the correctable errors in files using the (2,7) code has been presented. The method requires only knowledge of the stored data in the unencoded form available to the programmer. Analysis using this method shows that almost all the errors can be explained as the results of very simple one- and two-bit detector errors propagated into bursts of up to five bits by the channel decoder. The lengths of the error bursts are thus determined mostly by the properties of the channel decoder. This emphasizes the importance of limiting the error propagation to very small values as was done for the (2,7) code.

The main contributing factors to these one- and two-bit detector errors are imperfections in the disk coating, weaker than average heads, difficult data patterns, and electronic noise. A typical error occurs when a difficult pattern is written at a disk imperfection or read by a weak head. This enables electronic noise to push the detector across the error threshold. For weaker heads and larger imperfections less noise is needed to cause errors. Therefore, weaker heads have more different error sites than average, and larger imperfections cause errors more frequently than average.

The error-correcting code used in the IBM 3380 is more than adequate to correct the errors observed in the experiment. The data presented here suggest that a code capable of correcting only much shorter bursts would have performed just about as well. A code capable of correcting only five-bit bursts would have corrected all but 33 of the 18 753 errors and all but 17 of the 1836 error types. That means about one more error per week would have had to be recovered by rereads. This difference would not have been noticeable.

When speaking of disk imperfections, weak heads, and difficult data patterns, we should keep in mind that these are relative terms. Half the members of any sample are below average, but in this case the average is remarkably good.

Acknowledgments

The special program to exercise the disks and gather the SENSE information was written by J. Wyllie. D. Woo helped to manage the data and monitor the experiment. R. Adler, B. Kitchens, and P. Siegel provided helpful discussions about the analysis method.

Appendix

The (2,7) code is described by **Table 3.** To encode a string u, find the entry in the *IN* column of the table which matches

the first two, three, or four bits of u which have not yet been encoded. The table is constructed so that no more than one match can be found, and one match will be found if the unencoded part of u is at least four bits long. The encoded version of this substring appears in the *OUT* column. Concatenate the encoded substring to the output string e, which is initially empty. Repeat the process until all bits of u have been encoded. If the final k-bit substring of u (k = 1, 2, or 3) does not match an entry in the *IN* column, extend it with zeros until a match is found. The encoded version of the final substring is the first $2k$ bits of the corresponding *OUT*-column string. The length of e is exactly twice the length of u. An example is included with the table to illustrate the encoding procedure.

Correctly encoded strings can be decoded by a similar procedure, transposing the roles of the *IN* and *OUT* columns of the table. Unfortunately, the table gives no information on how the IBM 3380 decodes other strings such as those resulting from errors.

The complete encoding and decoding algorithms used in the IBM 3380 are given in **Table 4.** The encoding algorithm is equivalent to the table-based procedure. The decoding algorithm decodes all strings of length $2n$. It agrees with the table-based procedure where that procedure is defined.

Table 3 Code table for the (2,7) code.

IN	*OUT*
10	0100
11	1000
000	000100
010	100100
011	001000
0010	00100100
0011	00001000

Encoding example

Input:
011 0011 11 000 10 10

Output:
001000 00001000 1000 000100 0100 0100

Table 4 Encoding and decoding algorithms for the (2,7) code.

Encoder

Input: $\{u_k\}$, $0 \le k < n$

Output: $\{e_k\}$, $0 \le k < 2n$

Initialize: $p_{-2} = 0$, $p_{-1} = u_0$, $u_n = u_{n+1} = 0$

For $0 \le k < n$

$$e_{2k} = u_k u_{k+1} \bar{p}_{k-2} + \bar{u}_k u_{k+1} \bar{u}_{k+2} \bar{p}_{k-2}$$

$$e_{2k+1} = \bar{u}_{k+1} p_{k-1}$$

$$p_k = u_k u_{k+1} \bar{p}_{k-1} + \bar{u}_k u_{k+1} + \bar{u}_k \bar{u}_{k+1} \bar{u}_{k+2} \bar{p}_{k-2} \bar{p}_{k-1}$$

The p_k indicate boundaries of codewords in the table: $p_k = 1$ if and only if u_{k+2} is the last bit of a codeword.

Decoder

Input: $\{e_k\}$, $0 \le k < 2n$

Output: $\{u_k\}$, $0 \le k < n$

Initialize: $e_{-4} = e_{-3} = e_{-2} = e_{-1} = 0$, $e_{2n} = e_{2n+1} = 0$

For $0 \le k < n$

$$u_k = e_{2k-2} + \bar{e}_{2k+3} e_{2k} + e_{2k+1} \bar{e}_{2k-1} e_{2k-3} + e_{2k+1} e_{2k-4}$$

Each variable with a subscript is a logical variable. True corresponds to 1, false to 0. Juxtaposition means AND, + means OR, and NOT is indicated by a bar above the variable name.

References

1. P. Franaszek, "Run-Length-Limited Variable Length Coding with Error Propagation Limitation," U.S. Patent 3,689,899, 1972.
2. J. Eggenberger and P. Hodges, "Sequential Encoding and Decoding of Variable Word Length, Fixed Rate Data Codes," U.S. Patent 4,115,768, 1978.
3. P. Siegel, "Applications of a Peak Detection Channel Model," *IEEE Trans. Magnetics* **MAG-18,** 1250–1252 (1982).
4. *IBM Disk Storage Management Guide, Error Handing*, Order No. GA26-1672, available through IBM branch offices.

Received July 8, 1983; revised October 20, 1983

Thomas D. Howell *IBM Research Division, Säumerstrasse 4, 8803 Rüschlikon, Switzerland.* Dr. Howell is a Research staff member at the Zurich Research laboratory. He is currently working on signal processing and coding for magnetic recording channels. He joined IBM at the San Jose, California, Research laboratory in 1977, and worked on a performance evaluation tool for computer systems called the VM Emulator until 1979. He joined the recording channel project at San Jose Research in 1979 and was its manager from 1979 until 1983. Dr. Howell took a temporary assignment at the Zurich Research laboratory in June 1983. He received an IBM Outstanding Technical Achievement Award in 1983 for his work on recording channel modeling. Dr. Howell received his B.S. in mathematics from the California Institute of Technology, Pasadena, in 1973, his M.S. in computer science from Cornell University, Ithaca, New York, and his Ph.D. in computer science in 1976, also from Cornell University. From 1976 to 1977, he was an IBM Postdoctoral Fellow at the University of California at Berkeley. Dr. Howell is a member of the Association for Computing Machinery, the Institute of Electrical and Electronics Engineers, and the Mathematical Association of America.