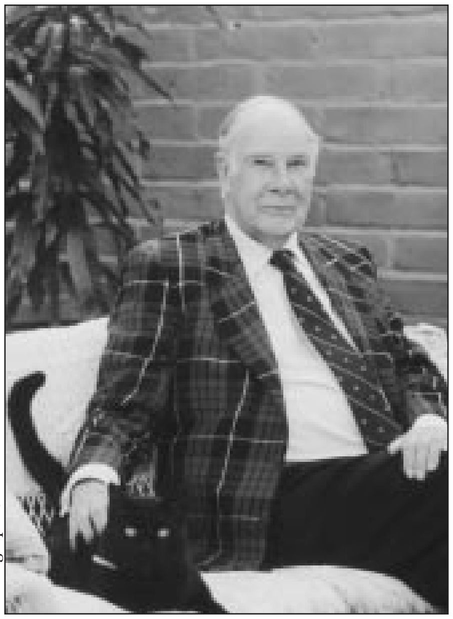
# Richard Wesley Hamming (1915–1998)

*Samuel P. Morgan*



**Richard W. Hamming**

Richard Wesley Hamming, mathematician, pioneer computer scientist, and professor, died of a heart attack on January 7, 1998, in Monterey, California, at the age of eighty-two. His research career began at Bell Laboratories in the 1940s, in the early days of electronic computers, and included the invention of the Hamming error-correcting codes. In the 1970s he shifted to teaching, and at his death he was Distinguished Professor Emeritus of computer science at the Naval Postgraduate School. He is survived by his wife, Wanda, a niece, and a nephew.

Dick Hamming, as he was known to friends, was born in Chicago on February 11, 1915. He received a B.S. in mathematics from the University of Chicago in 1937, an M.A. from the University of Nebraska in 1939, and a Ph.D. in mathematics from the University of Illinois in 1942. His doctoral thesis, written under the supervision of W. J. Trjitzinsky, was entitled "Some Problems in the Boundary Value Theory of Linear Differential Equations".

After brief teaching positions at the University of Illinois and the University of Louisville, Dick was recruited in 1945 to work at Los Alamos to run the IBM machines that were doing calculations for the Manhattan Project. Wanda followed him to Los Alamos, where she was hired to use a desk calculator, working eventually for Enrico Fermi and Edward Teller. Although Dick jokingly described his position at Los Alamos as "computer janitor", the work gave him a vision of the role that numerical computation was destined to play in the scientific and technological world of the future. He saw that experiments were going to be possible with computers that were not possible in the laboratory, and he stayed at Los Alamos for six months after most of the other scientists had left "to figure out," as he told *IEEE Spectrum* in 1993, "what had happened there, and why it had happened that way."

Dick arrived at Bell Laboratories in 1946 and joined a group of applied mathematicians that included the communication theorist Claude Shannon and the statistician John Tukey. The group regarded itself as chartered to "do unconventional things in unconventional ways and still get valuable results." Dick was hired to do elasticity theory, but the presence of computers required him to spend more and more time on them, and his career became centered on bringing large-scale scientific computation into Bell Labs. Much of his research between 1946 and 1960 dealt with error-correcting codes and predictor-corrector methods for numerical integration. At this time he also developed an interest in digital filters that continued throughout his career. He was from time to time promoted to head a department of re-

*Samuel P. Morgan retired in July 1998 as a Distinguished Member of Technical Staff at Bell Laboratories. His e-mail address remains* `spm@research.bell-labs.com`.

*The author wishes to thank Wanda Hamming and E. N. Gilbert, J. F. Kaiser, H. Loomis, M. D. McIlroy, R. Pinkham, N. L. Schryer, R. L. Urbanke, and M. H. Wright for their help with this article.*

searchers, but since he explicitly did not want management responsibilities, these assignments always came to an end. My contacts with Dick were during the years 1947–76 while we were colleagues, first in the Mathematics and Statistics Research Center and then in the Computing Science Research Center at Bell Labs.

After 1960 Dick became increasingly interested in teaching and writing. Between 1960 and 1976, while retaining his base at Bell Labs, he held visiting or adjunct professorships at Stanford University, the City College of New York, the University of California at Irvine, and Princeton University. In 1976 he retired from Bell Labs to become an adjunct professor (later senior lecturer) of computer science at the Naval Postgraduate School. He became Distinguished Professor Emeritus in 1997 and taught his last class in December 1997.

Among Dick's professional honors were the Turing Award (1968) of the Association for Computing Machinery (ACM), the Emanuel R. Piore Award (1979) of the Institute of Electrical and Electronics Engineers (IEEE), and the Harold Pender Award of the University of Pennsylvania (1981). In 1988 the IEEE Richard W. Hamming Medal, "For exceptional contributions to information sciences and systems," was named after him, and he was the first recipient. In 1996 in Munich he received the prestigious $130,000 Eduard Rhein Award for Achievement in Technology for his work on error-correcting codes. He was president of the ACM (1958–60), a member of the National Academy of Engineering, and a Fellow of IEEE.

Dick wrote nine books, some of which went through multiple editions (see sidebar, page 1015). He published some seventy-five technical articles and held three patents.

## Error-correcting Codes

Dick is most famous for inventing the Hamming error-correcting codes [1] and for the concept of Hamming distance, which is central to coding theory. Data in digital systems are typically stored, transmitted, and processed in binary form as blocks of bits. If a single bit is in error, the message is garbled or the computation spoiled. In large-scale computers or telephone switching systems, an enormous number of computations must be performed without a single error in the end result. Dick set himself the task of making the computer itself detect and correct isolated errors so that the computation could proceed in a way that would be more efficient than simply doing everything three times and accepting the majority result.

His approach was based on a generalization of parity checking. A simple parity check works as follows. Suppose we have a block of $n$ bits and add an $(n+1)$st bit so that the whole message has an even number of 1's in it. This is called an even parity check. At the receiving end, if there are not an even number of 1's in the message, then there must be an odd number of errors in the message. If bit errors occur independently and if the message is short and the bit error rate is small, then the message most probably contains a single error, but we do not know which bit is incorrect.

The Hamming codes use multiple parity checks to locate and correct single-bit errors. Each check is now a sum only over bits in selected positions. In the simplest case, message words of length $2^r - r - 1$ bits, where $r$ is any integer, are to be sent together with $r$ check bits, so that each code word (message bits plus check bits) contains $2^r - 1$ bits. The positions of the code word are numbered from left to right. The first check bit is in position 1 and is a parity check over the positions that have a 1 as the least significant bit of their binary representations (that is, positions 1, 3, 5, 7, …). The second check bit is in position 2 and is a parity check over the positions that have a 1 as the second least significant bit of their binary representations (that is, positions 2, 3, 6, 7, …). The third check bit is in position 4 and checks the positions that have a 1 as the third least significant bit (positions 4, 5, 6, 7, 12, …), and so on. If no parity checks fail, the code word is assumed to be correct. If one bit of the code word is in error, the error is at the location whose binary representation equals the pattern of the failed parity checks.

| Position | | | | | | | Decimal Value |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| **0** | **0** | 0 | **0** | 0 | 0 | 0 | 0 |
| **1** | **1** | 0 | **1** | 0 | 0 | 1 | 1 |
| **0** | **1** | 0 | **1** | 0 | 1 | 0 | 2 |
| **1** | **0** | 0 | **0** | 0 | 1 | 1 | 3 |
| **1** | **0** | 0 | **1** | 1 | 0 | 0 | 4 |
| **0** | **1** | 0 | **0** | 1 | 0 | 1 | 5 |
| **1** | **1** | 0 | **0** | 1 | 1 | 0 | 6 |
| **0** | **0** | 0 | **1** | 1 | 1 | 1 | 7 |
| **1** | **1** | 1 | **0** | 0 | 0 | 0 | 8 |
| **0** | **0** | 1 | **1** | 0 | 0 | 1 | 9 |
| **1** | **0** | 1 | **1** | 0 | 1 | 0 | 10 |
| **0** | **1** | 1 | **0** | 0 | 1 | 1 | 11 |
| **0** | **1** | 1 | **1** | 1 | 0 | 0 | 12 |
| **1** | **0** | 1 | **0** | 1 | 0 | 1 | 13 |
| **0** | **0** | 1 | **0** | 1 | 1 | 0 | 14 |
| **1** | **1** | 1 | **1** | 1 | 1 | 1 | 15 |

**Table 1. Hamming code with $r = 3$.**

Table 1 shows the Hamming code for $r = 3$. The check bits, in boldface, are in positions 1, 2, and 4 of each code word. Their values may be computed from the remaining (message) bits, which represent the numbers 1 through 15 in binary.

The photograph (next page) shows the reverse of the IEEE Hamming medal, which carries the parity check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

for the Hamming code of Table 1. $\mathbf{H}$ is used as follows, with a permuted version of Table 1 in which the positions are renumbered in the order 7 6 5 3 4 2 1 to bring the check bits to the end of

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.
RICHARD W. HAMMING
FOR
EXCEPTIONAL CONTRIBUTIONS
TO INFORMATION SCIENCES
AND SYSTEMS
1988

**Reverse of the IEEE Hamming Medal, showing a parity check matrix for a Hamming error-correcting code.**

Property of AT&T Archives. Reprinted with permission of AT&T.

the code word. Let **r** be a binary column vector of length 7 representing any received word, not necessarily a code word. Using Boolean arithmetic, calculate

$$\mathbf{s} = \mathbf{H}\mathbf{r},$$

where **s** is a binary column vector of length 3. If $\mathbf{s} = 0$, then **r** is a code word. Otherwise, **s** will coincide with one of the 7 columns of **H**. If **s** coincides with the $i$th column of **H**, then the $i$th bit of **r** is in error and must be reversed in order to recover the correct code word.

Error-correcting codes can be interpreted geometrically. Define the *Hamming distance* between two code words as the number of positions in which the code words differ. The minimum Hamming distance between code words in Table 1 is 3. Since a single-bit error moves a received word a distance 1 from the correct word, single-bit errors can be unambiguously corrected by changing the received word to the nearest code word. Furthermore, Hamming codes are *perfect*, in the sense that every received word is at a distance at most 1 from a code word. It is easy to verify that the number of code words times the number of words that are at a distance no greater than 1 from a code word is equal to the total number of possible words. This means that every pattern of check failures actually occurs for some single-bit error in the transmitted word. More recent codes, which undertake to correct more than a single error, are rarely perfect; that is, some patterns of bit errors can occur that do not lead to unambiguous decoding.

Dick observed that a code having a minimum Hamming distance $2t+1$, where $t$ is any integer, can correct $t$ errors; and if the minimum distance is $2t+2$, the code can correct $t$ errors and detect, but not correct, $t+1$ errors. The codes described above are single error correcting, and by adding an additional parity bit to each word they become double error detecting. These codes solved a large part of the maintenance problem for telephone company switching equipment, and "Hamming bits" went into computer memories in the late 1950s, for example, in the IBM 7030 Stretch supercomputer.

We can indicate briefly how the Hamming codes are related to some families of multiple-error-correcting codes that are in current use. In general, a linear error-correcting code may be characterized by the number triple $[n, k, d]$, where $n$ is the number of symbols in the code words, $k$ is the number of symbols in the message words, and $d$ is the minimum distance. Thus, the code of Table 1 is a $[7, 4, 3]$ code.

In algebraic coding theory, message words and code words are represented by polynomials with coefficients over a Galois field $GF[q]$ of order $q$, where $q$ is a power of a prime. The polynomial $c(X)$ representing a code word is obtained by multiplying the message polynomial $m(X)$ by a fixed generator polynomial $g(X)$. Two related families of codes, both invented about 1960, permit the correction of an arbitrary number of errors by the use of appropriate redundancy.

Bose-Chaudhuri-Hocquenghem (BCH) codes can be (although they need not be) constructed over the field $GF(2)$. A binary BCH code with code word length $2^r - 1$ and minimum distance at least $2t+1$, so that it will correct $t$ errors, can always be constructed with at most $rt$ check digits. That is, the code will have performance at least as good as

$$[2^r - 1,\ 2^r - 1 - rt,\ 2t+1].$$

When $t = 1$, this BCH code is equivalent to a Hamming single-error-correcting code, up to a permutation.

Reed-Solomon (RS) codes are constructed over a field $GF(q)$, where $q$ is a prime power and $q > 2$. Reed-Solomon codes are maximum distance separable codes; that is, the code constructed over $GF(q)$ with minimum distance $d$ is described by the triple

$$[q-1,\ q-d,\ d].$$

For example, NASA uses a $[255, 223, 33]$ RS code over $GF(2^8)$ for deep-space communication. In another widely used application, a $[32, 28, 5]$ code and a $[28, 24, 5]$ code are used in an interleaved scheme to correct burst errors of length up to 4000 bits on compact discs [2].

In a sense, Dick's 1950 paper set off the current avalanche of coding theory and applications, although he left it to others to ride the avalanche.

## Numerical Analysis

Much of Dick's early work was in numerical analysis. One of his contributions to this field was the Hamming predictor-corrector (PC) set for ordinary differential equations.

Numerical integration of an ordinary differential equation, say, $y' = f(x, y)$, consists of finding approximate values of $y$ at a set of equispaced values of $x$. Milne's method was a once-popular approach. Briefly, Milne predicts the next value of $y$ using the predictor

$$\bar{y}_{n+1} = y_{n-3} + \frac{4h}{3}[2y'_n - y'_{n-1} + 2y'_{n-2}],$$

where $h$ is the step size or spacing between values of $x$. Milne then corrects the value of $y$ using the corrector

$$y_{n+1} = y_{n-1} + \frac{h}{3}[\bar{y}'_{n+1} + 4y'_n + y'_{n-1}].$$

For any PC method the starting values of $y$ have to be found by some other method, such as Runge-Kutta.

Unfortunately, Milne's method is unstable; that is, errors due to roundoff noise are amplified as the solution progresses. The problem may not be serious if one is trying to follow a growing solution, such as the solution $\exp(\lambda x)$ of $y' = \lambda y$ when $\lambda > 0$. However, if one attempts to track a decreasing solution, such as $\exp(\lambda x)$ when $\lambda < 0$, the roundoff noise eventually swamps the desired solution.

Dick made a general study [3] of PC methods using Milne's predictor together with correctors of the form

$$\begin{aligned} y_{n+1} &= ay_n + by_{n-1} + cy_{n-2} \\ &\quad + h[d\bar{y}'_{n+1} + ey'_n + fy'_{n-1}], \end{aligned}$$

where $a, b, \dots, f$ are arbitrary constants. On the basis of various criteria, he proposed the corrector

$$\begin{aligned} y_{n+1} &= \frac{1}{8}[9y_n - y_{n-2}] \\ &\quad + \frac{3h}{8}[\bar{y}'_{n+1} + 2y'_n + y'_{n-1}]. \end{aligned}$$

The Hamming corrector is stable—that is, roundoff errors are damped out—for equations like $y' = \lambda y$ when $\lambda$ is negative so long as $h$ satisfies $-2.6 < h\lambda < 0$.

The Hamming PC set held the field for a number of years. Recently the Adams-Bashforth set, which uses an extra value of the derivative and has a slightly larger stability region, has found favor [4, 5]. However, the nature of the problem to be solved often plays a role in choosing a numerical integration method.

### Hamming Maxims

The purpose of computing is insight, not numbers.

It is better to do the right problem the wrong way than to do the wrong problem the right way.

Let's not raise the falutin' index [an injunction against the use of pretentious, "high-falutin'" terminology].

If you don't work on important problems, it's not likely you'll do important work.

If the prediction that an airplane can stay up depends on the difference between Riemann and Lebesgue integration, I don't want to fly in it.

### Hamming's Books

*Numerical Methods for Scientists and Engineers*, McGraw-Hill, 1962; 2nd ed. 1973; Dover reprint 1985; translated into Russian.

*Calculus and the Computer Revolution*, Houghton-Mifflin, 1968.

*Introduction to Applied Numerical Analysis*, McGraw-Hill, 1971.

*Computers and Society*, McGraw-Hill, 1972.

*Digital Filters*, Prentice-Hall, 1977; 2nd ed. 1983; 3rd ed. 1989; translated into several European languages.

*Coding and Information Theory*, Prentice-Hall, 1980; 2nd ed. 1986.

*Methods of Mathematics Applied to Calculus, Probability and Statistics*, Prentice-Hall, 1985.

*The Art of Probability for Scientists and Engineers*, Addison-Wesley, 1991.

*The Art of Doing Science and Engineering: Learning to Learn*, Gordon and Breach, 1997.

Dick's greatest influence on numerical analysis was probably through his two books on the subject [4, 6]. The books were organized for the nonspecialist user, with general methods subsuming special cases, and always with an eye toward what works in practice. They were unique in their focus on the whole context in which the numerical analyst should function. Dick insisted that the user of numerical methods should consider both the source of the problem and the use to which the results were going to be put. He enjoyed working with physicists, chemists, and engineers and was remarkably adept at dealing with "walk-in" problems. His clientele extended over a large part of Bell Labs.

Finally, Dick's books were full of unobtrusive good ideas. For example, he suggested a technique for estimating the level of systematic inaccuracy in a computed "black box" function. He observed that taking higher-order differences of a set of function values tends eventually to produce quantities with a telltale pattern of alternating signs, at which point the size of the remaining numbers re-

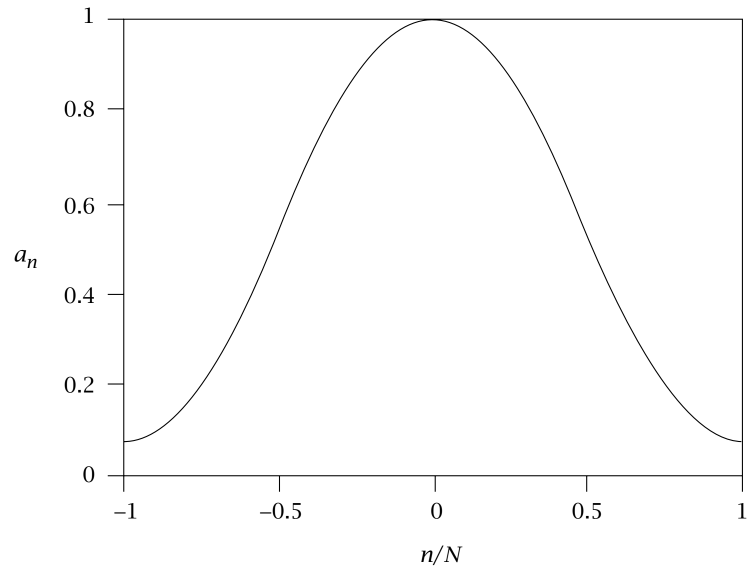


**Figure 1. Hamming window weight coefficients.**

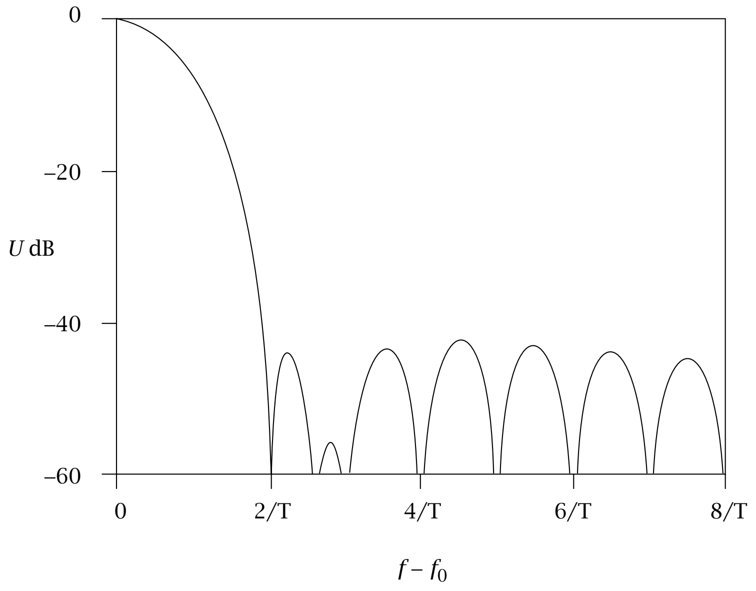


**Figure 2. Hamming window frequency transform.**

flects the absolute error in the computation. This observation is routinely used in scientific computing to estimate accuracy when only function values are available.

## Digital Filters

Dick's contributions to digital filters began during his first years at Bell Labs and were initially stimulated by his association with Ralph Blackman and John Tukey [7]. In this field he is credited with the invention of the *Hamming window*, which may be explained as follows.

In mathematical terms, a digital filter is realized by a linear combination of equispaced samples of a function of time, taken over an interval $T$. For example, if the input function is $u(t)$ and there are $2N+1$ samples, the output of the filter at a particular time is

$$\sum_{n=-N}^{N} c_n u(t + nT/2N),$$

where the $c_n$ are the filter weights.

The properties of a digital filter are defined in terms of its effect on the harmonic time function $u(t) = \exp(2\pi i f t)$. Usually the filter is designed to pass a particular frequency, say $f_0$, and to discriminate against nearby frequencies. In order to make the center frequency $f_0$ explicit, it is convenient to introduce phases into the weight coefficients by writing

$$c_n = a_n \exp(-n\pi i f_0 T/N).$$

Then we are interested in the function

$$U(f) = \sum_{n=-N}^{N} a_n \exp[n\pi i(f - f_0)T/N],$$

which is the response to $\exp(2\pi i f t)$ and is called the *frequency transform*. The coefficients $a_n$ are called the *filter window*.

The frequency transform $U(f)$ typically has a principal maximum, or main lobe, at $f_0$ and subsidiary maxima, or side lobes, at other frequencies. In some applications it is desirable to keep the magnitudes of the side lobes as low as possible with respect to the main lobe.

Dick found a set of filter coefficients [7, 8] that are particularly effective in suppressing side lobes. The coefficients of the Hamming window are given by

$$a_n = 0.54 + 0.46 \cos n\pi/N$$

for $-N \le n \le N$. The Hamming window, which has also been called a raised cosine on a pedestal, is shown in Figure 1.

The series for $U(f)$ can be summed explicitly for a Hamming window with any value of $N$. However, the limiting value for large $N$ is particularly simple. Up to normalization it is

$$U(f) = \frac{0.54 \sin[\pi(f - f_0)T]}{[\pi(f - f_0)T]} + \frac{0.46[\pi(f - f_0)T] \sin[\pi(f - f_0)T]}{\pi^2 - [\pi(f - f_0)T]^2}.$$

The quantity $20 \log_{10} |U(f)/U(f_0)|$, that is, the normalized value of the frequency transform in decibels, is plotted in Figure 2. The transform is symmetric about $f = f_0$, and the lobe widths are inversely proportional to the width $T$ of the time window. The highest side lobe is 42.68 dB below the main lobe.

For comparison, a uniform window, in which all the coefficients $a_n$ are equal, corresponds to the transform

$$U(f) = \frac{\sin[\pi(f - f_0)T]}{[\pi(f - f_0)T]},$$

whose highest side lobe is only 13.27 dB below the main lobe.

Window design involves various tradeoffs: for example, between the width of the main lobe, the height of the largest side lobe, and the rate of rolloff of the distant side lobes. Different window functions have been designed to meet different requirements, but the Hamming window is perhaps the most widely used because of its simplicity and effectiveness.

Dick's long interest in digital filters led him to write a succinct monograph [8] that is now in its third edition. Much of the literature on digital filter design appears in electrical engineering journals and makes heavy use of the terminology of that field. Dick's book aims to make the mathematical ideas that underlie the analysis and design of filters available to a broad audience. A hallmark of Dick's writing is his attempt to avoid jargon. His book on digital filters is a classic example of this writing. The book is filled with his unique insights, which make it a valuable reference for practitioners of the digital filter design art.

Dick came into the world of computing just as it was emerging from the era of desk calculators and entering the era of electronic computers. He saw, much sooner and more clearly than most of his colleagues, how the daily work of almost everybody at Bell Labs would come to depend on computers. In the early 1960s he predicted that half the budget of Bell Labs would eventually go to computing; his estimate, which proved to be low, was much higher than anyone else's. He made it his business to educate the organization for the change.

As a numerical mathematician he undertook to teach scientists and engineers how to use computers to solve their problems in a hands-on way. His best-known maxim, "The purpose of computing is insight, not numbers," anticipated that the user would watch the computation as it proceeded and might use the ongoing results to gain additional insight into the original problem.

Dick's approach to numerical mathematics was highly effective in its day. Today's world of software libraries is superficially quite different. Nowadays most users of numerical mathematics depend on software packages that are written by specialists and may be highly sophisticated. However, in the modern world of canned software, faster computers, and bigger and more complex problems Dick's maxims for and warnings to users are more important than ever before.

Dick was concerned that the user would not understand the algorithms and/or would use them incorrectly. The main concern today is communicating the physical problem to the software package correctly. Dick noted that "a good theoretician can account for almost any result that is produced, right or wrong," which makes it important to be able to tell if we have a sensible answer. Customers of computing software need to be skeptical of the results produced. The worry here is not that the software tools will incorrectly solve the problem as posed. The real worry is that the problem as posed may not be the problem the user wants solved. Because of the difficulty of monitoring what goes on inside a large software package and of interpreting diagnostics that may come from different parts of the package, it is important to lay out in advance some checkable conditions that the solution must satisfy (for example, conservation laws for systems of partial differential equations). There is still no substitute for Dick's emphasis on common-sense thinking.

Dick Hamming made seminal contributions to computer science in its early days, and as a person he was never dull. He had strong opinions, and he liked to express them. His voice comes through in his books in a way that few technical authors achieve. He liked to give people advice, especially young people, whom he would educate and entertain with his often-repeated lecture "You and Your Research". He enjoyed the speaker's platform, and on occasion he enjoyed, as he jokingly said, "hamming" with a small *h*. One might agree or disagree with a Hamming pronouncement (I agreed with him much of the time), but no one who ever met him or heard him is likely to forget him.

## References

[1] R. W. Hamming, *Error-detecting and error-correcting codes*, Bell Sys. Tech. J. **29** (1950), 147–160.

[2] S. B. Wicker and V. K. Bhargava, eds., *Reed-Solomon codes and their applications*, IEEE Press, New York, 1994.

[3] R. W. Hamming, *Stable predictor-corrector methods for ordinary differential equations*, J. Assoc. Comput. Mach. **6** (1959), 37–47.

[4] R. W. Hamming, *Numerical methods for scientists and engineers*, 2nd ed., McGraw-Hill, New York, 1973.

[5] L. Lapidus and J. H. Seinfeld, *Numerical solution of ordinary differential equations*, Academic Press, New York, 1971.

[6] R. W. Hamming, *Introduction to applied numerical analysis*, McGraw-Hill, New York, 1971.

[7] R. B. Blackman and J. W. Tukey, *The measurement of power spectra from the point of view of communications engineering*, Bell Sys. Tech. J. **37** (1958), 185–282, 485–569.

[8] R. W. Hamming, *Digital filters*, 3rd ed., Prentice-Hall, Englewood Cliffs, NJ, 1989.