Determining the Burst-Correcting Limit of Cyclic Codes

HANS J. MATT AND JAMES L. MASSEY, FELLOW, IEEE

Abstract—**Two new computationally efficient algorithms are developed for finding the exact burst-correcting limit of a cyclic code. The first algorithm is based on testing the column rank of certain submatrices of the parity-check matrix of the code. An auxiliary result is a proof that every cyclic (n,k) code, with a minimum distance of at least three, corrects at least all bursts of length $\lfloor (n-2k+1)/2 \rfloor$ or less. The second algorithm, which requires somewhat less computation, is based on finding the length of the shortest linear feedback shift-register that generates the subsequences of length $n-k$ of the sequence formed by the coefficients of the parity-check polynomial $h(x)$, augmented with $\lfloor (n-k)/2 \rfloor - 1$ leading zeros and trailing zeros. Tables of the burst-correcting limit for a large number of binary cyclic codes are included.**

I. Introduction

THIS PAPER presents two new and simple methods for determining the burst-correcting limit of a cyclic code. In the first section we collect some well-known results concerning burst-correction that will be used in the sequel. In Section II we establish the main result of this paper, a simple test for the burst-correcting limit of a cyclic code in terms of its parity-check matrix, which we also use to prove a new lower bound on the burst-correcting limit of a cyclic code. We show in Section III that this test is equivalent to finding the length of the shortest linear feedback shift-register that can generate certain subsequences of a sequence determined by the parity-check polynomial of the cyclic code; this leads to a computationally efficient test for the burst-correcting limit. We conclude in Section IV with a tabulation of the burst-correcting limit for a number of cyclic codes.

Consider the correction of a single burst by a linear block code of length n and dimension k (i.e., an (n,k) code with symbols in the finite field GF(q)). Suppose that the transmitted codeword $\boldsymbol{x}=[x_1,x_2,\cdots,x_n]$ is received as the n-tuple $\boldsymbol{x}+\boldsymbol{e}$ where $\boldsymbol{e}=[e_1,e_2,\cdots,e_n]$, with $e_i \in \mathrm{GF}(q)$, is the *error pattern*. An error pattern $\boldsymbol{e} \neq \boldsymbol{0}$ is said to be an *open-loop burst of length b* if its nonzero components are confined to b consecutive components, the first and last of which are nonzero. An error pattern $\boldsymbol{e} \neq \boldsymbol{0}$ is said to be a *closed-loop burst of length b* if b is the smallest integer such that the nonzero components of $\boldsymbol{e}$ are confined to b consecutive components, where the first component of $\boldsymbol{e}$ is considered to follow the last component in a cyclical fashion. For example, $\boldsymbol{e}=[1,0,1,0,1]$ is a closed-loop burst of length $b=4$; however, there are two choices for the b consecutive components that contain all the nonzero digits. When $b \leqslant n/2$, this type of "ambiguity" for closed-loop bursts does not occur. For both the open-loop and closed-loop case, the error pattern $\boldsymbol{e}=\boldsymbol{0}$ is defined to be the unique burst of length 0.

By the *open-loop burst-correcting limit (closed-loop burst-correcting limit)* of an (n,k) code, denoted $B_o(B_c)$, we mean the largest integer such that the code can correct all open-loop bursts (closed-loop bursts) of length $B_o(B_c)$ or less. Since an open-loop burst of length b is also a closed-loop burst of length at most b, it follows that

$$B_c \leqslant B_o. \tag{1}$$

It might seem that closed-loop bursts have significance only for cyclic codes, but this is not the case. Gallager [1, p. 288] has given a definition of "bursts" on a channel, which is independent of the block structure of the coding system, by first specifying a *guard space g*. The *channel bursts* are then those segments of the semi-infinite channel error sequence that lie between all segments of g or more consecutive zeros, i.e., g or more consecutive error-free transmissions on the channel. A channel burst has length b if the corresponding segment, which must begin and end with a nonzero symbol, contains b symbols. If $g=n-B_c$, then a block code will correct any channel burst of length B_c or less, since any n consecutive symbols in the semi-infinite channel error sequence will be a closed-loop burst of length B_c or less. On the other hand it is necessary to specify $g=n-1$ to ensure that a block code will correct any channel burst of length B_o or less, since $n-2$ consecutive error-free transmissions can lead to the block error pattern $1,0,0,\cdots,0,1$, which is an open-loop burst of length n. It can be shown (cf. [1, p. 290]) that any coding system (block, convolutional, variable-length block, etc.) that has rate R (measured in information symbols per channel symbol) and corrects all "channel bursts" of

Manuscript received February 26, 1979. Parts of this paper were presented at the International Conference on Information and System Theory in Digital Communications, Technische Universität Berlin, Berlin, West Germany, September 1978. Parts of this paper were contained in the dissertation submitted to the Technische Universität Hannover, Hannover, West Germany, by H. Matt in partial fulfillment of the requirements for the Ph.D. degree.

H. Matt is with the Heinrich-Hertz-Institut für Nachrichtentechnik, Einsteinufer 37, 1000 Berlin 10, West Germany.

J. Massey was with the Heinrich-Hertz-Institut für Nachrichtentechnik, Berlin, West Germany, on leave from the Department of System Science, University of California, 4531 Boelter Hall, Los Angeles, CA 90024.

0018-9448/80/0500-0289$00.75

length b or less requires that the quard space satisfies

$$\frac{g}{b} \geqslant \frac{1+R}{1-R}, \qquad b \geqslant 1. \tag{2}$$

For an (n,k) code, $R=k/n$. Setting $b=B_c$ and $g=n-B_c$ we find from (1) that

$$B_c \leqslant \frac{1}{2}r \tag{3}$$

where $r=n-k$ is the *redundancy* of the code. In fact (3) also holds for B_o, and this stronger version is called the Rieger bound [2, p. 110].

Because $x_1+e_1=x_2+e_2$ is equivalent to $x_2-x_1=e_1-e_2$, a linear code can correct all the error patterns in some set $\mathcal{E}$ if and only if there is no codeword that can be written as the difference of two distinct error patterns in $\mathcal{E}$. Thus one can immediately make the following assertion.

Proposition 1: $B_o(B_c)$ is the largest integer b such that no codeword can be written as the difference of two distinct open-loop (closed-loop) bursts of length b or less.

We shall say that two open-loop (closed-loop) bursts of lengths b_1 and b_2 are *nonoverlapping* if their nonzero spans of b_1 and b_2 consecutive components, respectively, have no common components. But the difference e_1-e_2 of two open-loop (closed-loop) bursts e_1 and e_2 of length b or less can always be written as the difference $e_1'-e_2'$ of two nonoverlapping open-loop (closed-loop) bursts of length b or less; for example, the difference $[1,1,0,0,0,1]-[1,0,1,0,0,0]$, the difference of two closed-loop bursts of length three, can be written as $[0,0,0,0,0,1]-[0,-1,1,0,0,0]$. Thus Proposition 1 is equivalent to the following assertion.

Proposition 2: $B_o(B_c)$ is the largest integer b such that no codeword can be written as the difference of two distinct and nonoverlapping open-loop (closed-loop) bursts of length b or less.

Suppose the codeword x can be written as $x=e_1-e_2$ where e_1 and e_2 are nonoverlapping closed-loop bursts of lengths b_1 and b_2, respectively. If the code is cyclic, then there is a cyclic shift x' of x such that $x'=e_1'-e_2'$ where e_1' and e_2' are nonoverlapping open-loop bursts of lengths b_1 and b_2, respectively, and the last b_2 components of e_2' contain all the nonzero components. Thus Proposition 2 implies the next two propositions.

Proposition 3: For a cyclic code, $B_o=B_c$.

Hereafter we write simply B to denote the burst-correcting limit $B_o=B_c$ of a cyclic code.

Proposition 4: For a cyclic code, B is the largest integer b such that no nonzero codeword has the property that the nonzero components among its first $n-b$ components are confined to b or fewer consecutive components.

If H is the parity-check matrix of an (n,k) code, then $s=eH^T$ is the syndrome of the error pattern e relative to H. Hereafter, we assume H is fixed and call s simply the syndrome of e. For any (n,k) code all the error patterns in a set $\mathcal{E}$ are correctable if and only if they have distinct syndromes. For $b \leqslant n/2$, there are exactly $n(q-1)q^{b-1}$ distinct nonzero closed-loop bursts of length b or less since, for such a burst $e \neq 0$, there are n possible positions for the first nonzero component, which can assume $(q-1)$ different values, whereas the $(b-1)$ following components can assume arbitrary values. Thus we can immediately make the following assertion.

Proposition 5: For a cyclic code, B is the largest integer b such that the $n(q-1)q^{b-1}$ distinct nonzero bursts of length b or less have distinct nonzero syndromes.

Propositions 4 and 5 suggest two methods for determining the burst-correcting limit B of a cyclic code. The first is to examine the first $n-b$ components of the q^k-1 nonzero code words and apply Proposition 4. The second is to compute the syndromes of the $n(q-1)q^{b-1}$ nonzero bursts of length b or less and apply Proposition 5. Previous determinations of B for cyclic codes, e.g. [3] and [4], seem to have used one or both of these methods. However, when k and B are large, both methods become computationally prohibitive.

II. A Matrix Method for Determining B for a Cyclic Code

Hereafter we shall consider only cyclic (n,k) codes. To find B, it follows from Proposition 3 that we can consider only open-loop bursts, which are more convenient than closed-loop bursts. Thus we shall simply say "burst" when we mean "open-loop burst".

Since x is a codeword if and only if $xH^T=0$, and since $(e_1-e_2)H^T=0$ if and only if $s_1=e_1H^T=e_2H^T=s_2$, we see that Proposition 4 implies the following strengthened version of Proposition 5.

Proposition 6: For a cyclic code, B is the largest integer b such that:

i) all bursts whose nonzero components fall entirely in the last b components have distinct syndromes, and
ii) no nonzero burst of length b or less whose nonzero components fall entirely in its first $n-b$ components has the same syndrome as some burst specified in i).

We note next that a burst of length one is also a "single error". Thus $B=0$ if and only if the code cannot correct single errors, i.e., if and only if its minimum distance $d_{\min}$ satisfies $d_{\min}<3$. As the condition $d_{\min}<3$ is easily tested, we now seek a method for finding B when we already know that $B \geqslant 1$.

Let $h(x)=x^k+h_1x^{k-1}+\cdots+h_{k-1}x+h_k$, $h_k \neq 0$, be the *parity-check polynomial* [2, p. 208] of an (n,k) cyclic code over GF(q). The $r \times n$ matrix

$$H=\begin{bmatrix} h_k & h_{k-1} & \cdots & h_1 & & 1 & 0 & \cdots & 0 & 0 \\ 0 & h_k & & h_2 & & h_1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & & \vdots & \vdots \\ 0 & 0 & \cdots & h_k & \cdots & h_r & & \cdots & h_1 & 1 \end{bmatrix} \tag{4}$$

is a parity-check matrix for the code. It follows from the fact that the last r columns of H form an $r \times r$ lower

triangular matrix that, for $b \leqslant r$, all bursts whose nonzero components fall in the last b positions have distinct syndromes and that the first $r-b$ components of $s=eH^T$ are identically zero for all such bursts e whereas the last b components of s range over all q^b possibilities. But some nonzero burst whose nonzero components fall within some given b consecutive components within the first $n-b$ components will have a syndrome which is identically zero in its first $r-b$ components if and only if the $(r-b)\times b$ matrix, formed by the corresponding b consecutive columns of H with the last b rows deleted, has linearly dependent columns. Invoking Proposition 6 we have our first main result.

Theorem 1: For a cyclic code with $B \geqslant 1$, B is the largest integer b ($b<r$) such that every set of b consecutive columns of the matrix M_b is linearly independent, where M_b is the $(r-b)\times(n-b)$ matrix formed by deleting the last b rows and last b columns from the parity-check matrix H of (4).

For any $b>r-b$, any set of b columns of M_b must be linearly dependent. Thus Theorem 1 implies

$$B \leqslant \frac{1}{2}r=\frac{1}{2}(n-k) \tag{5}$$

in agreement with the Rieger bound.

We digress to remark that in view of (5) the only property of H that is needed to establish Theorem 1 is that H contains an $r\times r$ triangular (or identity) submatrix with zeros either below or above the main diagonal in its right r columns. It follows that the following more general result holds.

Theorem 1': Theorem 1 holds, with the condition $b<r$ weakened to the condition $b \leqslant r/2$, when H is any $r\times n$ parity-check matrix for a cyclic code having an $r\times r$ triangular (or identity) submatrix in its right r columns.

We return to our examination of the consequences of Theorem 1. From (4) and the definition of M_b we see that the first $r-b$ columns of M_b form a square matrix that is nonsingular, since $h_k \neq 0$. Thus columns $i, i+1,\cdots,i+b-1$ of M_b must be linearly independent for $i=1,2,\cdots,r-2b+1$. Similarly, the last $r-b$ columns of M_b (i.e., columns $n-r+1,n-r+2,\cdots,n-b$) also form a nonsingular matrix. Thus columns $i,i+1,\cdots,i+b-1$ of M_b must be linearly independent for $i=n-r+1,n-r+2,\cdots,n-2b+1$ or, equivalently, for $i=k+1,k+2,\cdots,n-2b+1$. We see then that every set of b consecutive columns of M_b is linearly independent if $n-r+1 \leqslant (r-2b+1)+1$ or, equivalently, if $b \leqslant \lfloor (n-2k+1)/2 \rfloor$, where $\lfloor \cdot \rfloor$ denotes the integer part of the enclosed number. Thus we have proved what appears to be a new lower bound on B for low-rate cyclic codes.

Theorem 2: For every cyclic code with $B \geqslant 1$ (or, equivalently, with $d_{\min} \geqslant 3$),

$$B \geqslant \lfloor (n-2k+1)/2 \rfloor. \tag{6}$$

This theorem shows, for instance, that any (127, 15) primitive Bose–Chaudhuri–Hocquenghen (BCH) code (regardless of the choice of primitive element used to define the code) will have $B \geqslant 49$. Notice that the bound (5) gives $B \leqslant 56$ for such a code.

Our observation as to certain sets of b consecutive columns of H which must be linearly independent also allows us to write the following strengthened version of Theorem 1.

Theorem 1S: For a cyclic code with $B \geqslant 1$, B is the largest integer b in the range $\max\{1, \lfloor (r-k+1)/2 \rfloor\} \leqslant b \leqslant \lfloor r/2 \rfloor$ such that columns $i,i+1,\cdots,i+b-1$ of M_b are linearly independent for $i=r-2b+2,r-2b+3,\cdots,k$.

Theorem 1S implies the following algorithm.

Algorithm 1 to determine B for a cyclic (n,k) code with $d_{\min} \geqslant 3$

Step 0: Set $b=\lfloor r/2 \rfloor$ and $i=2$.

Step 1: Check columns i, $i+1,\cdots,i+b-1$ of M_b for linear independence. If they are linearly independent go to Step 2. Otherwise go to Step 3.

Step 2: If $i=k$, stop and announce $B=b$. Otherwise increase i by one and return to Step 1.

Step 3: Decrease b by one. If now $b=\max\{1,\lfloor (r-k+1)/2 \rfloor\}$, stop and announce $B=b$. Otherwise return to Step 1.

Perhaps the only question in the reader's mind about Algorithm 1 is why one can dispense with resetting i to two when going from Step 3 back to Step 1. The justification is that columns $i,i+1,\cdots,i+b'-1$ of $M_{b'}$ contain, in their uppermost $r-b$ components, columns $i,i+1,\cdots,i+b'-1$ of M_b for $b'<b$. Thus, if columns $i,i+1,\cdots,i+b-1$ of M_b have been found to be linearly independent, it is certain that columns $i,i+1,\cdots,i+b'-1$ of $M_{b'}$ will also be independent for all $b'<b$.

The complexity of determining B by Algorithm 1 is determined primarily by the complexity of determining in Step 1 whether the given b column vectors of length $r-b$ are linearly independent. This step is performed about k times, each time with $b \approx r/2$ in the worst case. Checking the linear dependence by a Gauss reduction requires about $b^2(r-b)/2 \approx r^3/16$ GF(q) operations. Thus finding B by Algorithm 1 requires approximately $kr^3/16$ GF(q) operations.

In the next section we develop an alternative algorithm to compute B which exploits certain sequence properties to reduce the number of symbol operations required.

III. A Sequence Method for Determining B for a Cyclic Code

We begin by restating Theorem 1S in terms of linear *dependence* rather than independence, recognizing that the condition on the range of b was included in Theorem 1S only to minimize the testing needed to determine B.

Lemma 1: For a cyclic code with $B \geqslant 1$, B is the smallest positive integer b such that columns $i, i+1, \cdots, i+b$ of M_b are linearly dependent for some i in the range $r-2b+1 \leqslant i \leqslant k$.

But the facts that $B \leqslant \lfloor r/2 \rfloor$ and that the last $r-b$ columns of M_b are linearly independent now permit us to make the following assertion.

Lemma 2: For a cyclic code with $B \geqslant 1$, B is the smallest positive integer b such that the ith column of M_b can be written as a linear combination (possibly with all coefficients zero) of columns $i+1, i+2, \cdots, i+b$, for some i in the range $r-2b+1 \leqslant i \leqslant k$.

From (4), we see that the columns of M_b referred to in Lemma 2, namely columns $r-2b+1, r-2b+2, \cdots, k, k+1, \cdots, k+b$, form the following persymmetric (constant-minor-diagonals) matrix:

$$M_b' = \begin{bmatrix} h_{k-r+2b} & \cdots & & h_1 \\ h_{k-r+2b+1} & \cdots & & h_2 \\ \vdots & & & \\ h_k & & & \\ 0 & & & \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & h_k \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & \cdots & 0 \\ & & & \vdots \\ h_1 & & & \\ & & \ddots & \\ & & & 0 \\ & & & 1 \\ \vdots & & & \vdots \\ h_{r-b} & & \cdots & h_{r-2b} \end{bmatrix}.$$

Note that there are $b-1$ zeros at the end of the first row of M_b' and, since $h_k \neq 0$, also $b-1$ zeros at the bottom of the first column of M_b'. Lemma 2 can thus be recast as follows.

Lemma 3: For a cyclic code with $B \geqslant 1$, B is the smallest positive integer b such that some column of M_b' is a linear combination of the following b columns.

We next note that every set of $b+1$ consecutive columns of M_b' form a persymmetric matrix

$$A = \begin{bmatrix} a_{b+1} & a_b & \cdots & a_1 \\ a_{b+2} & a_{b+1} & \cdots & a_2 \\ \vdots & \vdots & & \vdots \\ a_r & a_{r-1} & \cdots & a_{r-b} \end{bmatrix} \tag{8}$$

where $a_1, a_2, \cdots, a_r$ is a subsequence of length r of the sequence

$$0, 0, \cdots, 0, 1, h_1, h_2, \cdots, h_k, 0, 0, \cdots, 0$$

where this latter sequence has exactly $b-1$ leading zeros and (since $h_k \neq 0$) exactly $b-1$ trailing zeros. We note also that the first column of A will be a linear combination of the remaining b columns if and only if there exist constants $c_1, c_2, \cdots, c_b$ (possibly all zeros) such that the first column, added to c_1 times the second column, plus c_2 times the third column, etc., gives the all-zero column. But, from (8), we see that this vector condition is equivalent to the following scalar equations:

$$a_j + c_1 a_{j-1} + \cdots + c_b a_{j-b} = 0, \quad \text{for } j = b+1, b+2, \cdots, r. \tag{9}$$

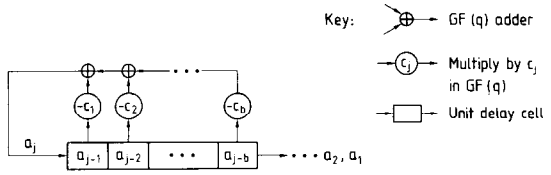

Fig. 1. General linear feedback shift-register of length b for field GF(q).

We recognize that (8) is precisely equivalent to the statement [5, p. 122] that the sequence $a_1, a_2, \cdots, a_r$ can be generated by the length-b linear feedback shift-register (LFSR) with feedback coefficients $-c_1, -c_2, \cdots, -c_b$ as illustrated in Fig. 1. Thus we have proved the following result (which holds when $a_1, a_2, \cdots, a_r$ are elements of an arbitrary field and which appears to be of independent interest).

Lemma 4: The first column of the persymmetric matrix A of (8) is a linear combination of the b remaining columns if and only if the sequence $a_1, a_2, \cdots, a_r$ can be generated by an LFSR of length b.

Combining Lemmas 3 and 4 and recalling that $B \leqslant \lfloor r/2 \rfloor$ we arrive at our second main result.

Theorem 3: For a cyclic code with $B \geqslant 1$,

$$B = \min\{L_1, L_2, \cdots, L_{k-\delta}\}$$

where L_i is the length of the shortest LFSR that can generate the length $r = n-k$ subsequence that starts in position i of the sequence

$$0, 0, \cdots, 0, 1, h_1, h_2, \cdots, h_k, 0, 0, \cdots, 0 \tag{9}$$

where there are $\lfloor r/2 \rfloor - 1$ leading zeros and trailing zeros, and where $\delta = 0$ if r is even and $\delta = 1$ if r is odd.

We remark that since the first $\lfloor r/2 \rfloor$ digits of the sequence in (9) form the subsequence $0, 0, \cdots, 0, 1$, we always have $L_1 = \lfloor r/2 \rfloor$ or $L_1 = \lfloor r/2 \rfloor + 1$ [5, Theorem 2]. Thus, since $B \leqslant \lfloor r/2 \rfloor$, there is in fact no need to compute L_1 when finding B.

Our interest in Theorem 3 arises from the fact that there is a computationally efficient algorithm, the "LFSR synthesis algorithm" of [5] (which is a variant of Berlekamp's "iterative algorithm" [6, p. 184] for decoding the BCH codes) for finding the length L of the shortest LFSR that can generate a given finite sequence of digits in any field. We incorporate this algorithm in the following algorithm.

Algorithm 2 to Determine B for a Cyclic (n,k) Code with $d_{\min} \geq 3$

Step 0: Set $b = \lfloor r/2 \rfloor$ and $i = 2$.

Step 1: Apply the LFSR synthesis algorithm to the subsequence of length r starting at position i of the sequence in (9) to find the minimum length L required. Exit from the LFSR algorithm as soon as $L \geq b$ is implied and go to Step 2. If $L < b$ when the LFSR synthesis algorithm is completed, replace b by L and go to Step 2.

Step 2: If $i = k - \delta$ (where δ is zero if r is even and is one otherwise) or if $b = \lfloor (r-k+1)/2 \rfloor$, stop and announce $B = b$. Otherwise increase i by one and return to Step 1.

Example: For the (31,21) binary cyclic code with

$$h(x) = x^{21} + x^{20} + x^{18} + x^{16} + x^{14} + x^{10} + x^8 + x^7 + x^6 + x^4 + x + 1,$$

the sequence (9) becomes

$$0,0,0,0,1,1,0,1,0,1,0,1,0,0,0,1,\ 0,1,1,1,0,1,0,0,1,1,0,0,0,0.$$

The subsequence of length $r = 10$ beginning in position 17,

$$0,1,1,1,0,1,0,0,1,1,$$

has $L_{17} = 3$ (the length three LFSR which generates this subsequence is shown in Fig. 2), which is the smallest of the L_i for $i = 1,2,\cdots,k-\delta = 21$. Thus $B = 3$ for this code.

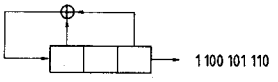

Fig. 2. Binary LFSR of Example 1 for generating sequence 0, 1, 1, 1, 0, 1, 0, 0, 1, 1.

The complexity of determining B by Algorithm 2 can be estimated as follows. For the worst case, i.e., for $b \approx r/2$, the use of the LFSR synthesis algorithm in Step 1 of Algorithm 2 will require about 4 $b^2 \approx r^2$ GF(q) operations. This step is performed $k - \delta$ times so that finding B by Algorithm 2 requires approximately kr^2 GF(q) operations. For sufficiently large b, Algorithm 2 will require fewer than the approximately $kr^3/16$ GF(q) operations required by Algorithm 1.

IV. Remarks and Tables

In Tables I–III, we list the values of B for a large number of binary Fire codes [7], binary nonprimitive BCH codes, and binary primitive BCH codes [6, p. 172], respectively. For these cyclic codes, B was found by use of Algorithm 1 (actually, by use of an earlier and slightly less efficient version of Algorithm 1 [8]). For many of the codes in Tables I–III, the determination of B by previ-

TABLE I
BURST-CORRECTING-LIMIT B FOR THOSE OF WAGNER'S "BEST" FIRE CODES FOR WHICH THE FIRE LOWER BOUND B_F ON B IS CONSERVATIVE ($r = n - k$)

n	k	B	B_F	r/n	B/r	c	$p(x) = g(x)/(x^c - 1)$
15	6	4	3	60.0	44.4	5	23
21	8	6	4	61.9	46.2	7	127
33	12	9	6	63.6	42.9	11	3043
39	14	12	7	64.1	48.0	13	13617
45	18	12	8	60.0	44.4	15	10011
57	20	16	10	64.9	43.2	19	1341035
69	24	22	12	65.2	48.9	23	34603145
75	30	20	13	60.0	44.4	25	4100001
87	30	28	15	65.5	49.1	29	3706175715
99	36	27	17	63.6	42.9	33	11000100011
75	40	15	8	46.7	42.9	15	4100001
65	40	10	7	38.5	40.0	13	10761
145	88	24	15	39.3	42.1	29	3572445367
155	104	20	16	32.9	39.2	31	5521623
215	144	28	22	33.0	39.4	43	3657555473
105	72	12	11	31.4	36.4	21	13321
165	112	20	17	32.1	37.7	33	6130725
175	120	20	18	31.4	36.4	35	4102041
63	48	6	5	23.8	40.0	9	103
91	66	11	7	27.5	44.0	13	15173
105	78	12	8	25.7	44.4	15	13321
189	144	18	14	23.8	40.0	27	1011011
231	168	30	17	27.3	47.6	33	11274767701
245	189	21	18	22.9	37.5	35	10040001
63	50	5	4	20.6	38.5	7	103
117	92	9	7	21.4	36.0	13	10377
171	134	15	10	21.6	40.5	19	1055321
385	300	30	28	22.1	35.3	55	16471647235
85	72	5	3	15.3	38.5	5	567
143	120	8	7	16.1	34.8	13	3777
165	140	10	8	15.2	40.0	15	3043
187	160	10	9	14.4	37.0	17	3777
275	230	20	13	16.4	44.4	25	4345543
297	246	18	17	17.2	35.3	33	1001001
495	410	30	28	17.2	35.3	55	11000100011
153	136	6	5	11.1	35.3	9	763
273	240	12	11	12.1	36.4	21	13077
459	408	18	14	11.1	35.3	27	100011011
525	470	20	18	10.5	36.4	35	4100001
391	363	11	9	7.2	39.3	17	5343
437	407	11	10	6.9	36.7	19	5343
675	628	18	14	7.0	38.3	27	4100001
725	676	18	15	6.8	36.7	29	4102041
897	836	22	20	6.8	36.1	39	34603145
1127	1045	33	25	7.3	40.2	49	107753475213

TABLE II
BURST-CORRECTING-LIMIT B FOR SOME NONPRIMITIVE BCH CODES

n	k	t	B	r/n	B/r	$g(x)$
17	9	2	3	47.1	37.5	727
21	12	2	4	42.9	44.4	1663
23	12	3	5	47.8	45.5	5343
33	22	2	3	33.3	27.3	5145
41	21	4	9	48.8	45.0	6647133
47	24	5	11	48.9	47.8	43073357
65	53	2	3	18.5	25.0	10761
65	40	4	10	38.5	40.0	354300067
73	46	4	12	37.0	44.4	1717773537

t is the random-error-correcting-limit, $r=n-k$, r/n, and B/r are expressed in percent and $g(x)$ is the generator polynomial in octal form.

TABLE III
BURST-CORRECTING-LIMIT B OF SOME PRIMITIVE BCH CODES

n	k	t	B	r/n(%)	B/r(%)	$g(x)$
7	4	1	1*	42.9	33.3	13
15	11	1	1*	26.7	25.0	23
15	7	2	4	53.3	50.0	721
15	5	3	5	66.7	50.0	2467
31	26	1	1*	16.1	20.0	45
31	21	2	4*	32.3	40.0	3551
31	16	3	7	48.4	46.7	107657
31	11	5	10	64.5	50.0	5423325
31	6	7	12*	80.6	48.0	313365047
31	26	1	1*	16.1	20.0	75
31	21	2	4*	32.3	40.0	2303
31	16	3	6	48.4	40.0	135273
31	11	5	10	64.5	50.0	6163305
31	6	7	12*	80.6	48.0	331722561
31	26	1	1*	16.1	20.0	67
31	21	2	3*	32.3	30.0	3557
31	16	3	7	48.4	46.7	141225
31	11	5	9*	64.5	45.0	6715141
31	6	7	11*	80.6	44.0	230745335
63	57	1	1*	9.5	16.7	103
63	51	2	4*	19.0	33.3	12471
63	45	3	5	28.6	27.8	1701317
63	39	4	11	38.1	45.8	166623567
63	36	5	12*	42.9	44.4	1033500423
63	30	6	15	52.4	45.5	157464165547
63	24	7	17*	61.9	43.6	173232604044l
63	18	10	21*	71.4	46.7	1363026512351725
63	16	11	22	74.6	46.8	6331141367235453
63	10	13	25*	84.1	47.2	472622305527250155
63	7	15	28	88.9	50.0	5231045543503271737
63	57	1	1*	9.5	16.7	147
63	51	2	3	19.0	25.0	11253
63	45	3	7	28.6	38.9	1431377
63	39	4	11	38.1	45.8	156615307
63	36	5	11	42.9	40.7	1705374561
63	30	6	12*	52.4	36.4	105065105421
63	24	7	18*	61.9	46.2	10611427654563
63	18	10	21*	71.4	46.7	1207106757642651
63	16	11	20	74.6	42.6	6625720617154137
63	10	13	26*	84.1	49.1	743065712726034051
63	7	15	28	88.9	50.0	4567515266076214705
63	57	1	1*	9.5	16.7	155
63	51	2	4	19.0	33.3	16223
63	45	3	8	28.6	44.4	1125063

TABLE III (*Continued*)

n	k	t	B	r/n(%)	B/r(%)	$g(x)$
63	39	4	10	38.1	41.7	102673553
63	36	5	12*	42.9	44.4	1537210637
63	30	6	14*	52.4	42.4	106054077561
63	24	7	16*	61.9	41.0	14225100247067
63	18	10	22	71.4	48.9	1142177532557273
63	16	11	20*	74.6	42.6	7456576205014441
63	10	13	25*	84.1	47.2	755340223164614 43
63	7	15	27*	88.9	48.2	6534604245447336175
127	120	1	1*	5.5	14.3	211
127	113	2	4*	11.0	28.6	41567
127	106	3	8*	16.5	38.1	11554743
127	99	4	12	22.0	42.9	3447023271
127	92	5	14*	27.6	40.0	624730022327
127	85	6	19	33.1	45.2	130704476322273
127	78	7	21	38.6	42.9	26230002166130115
127	71	9	27	44.1	48.2	6255010713253127753
127	64	10	29*	49.6	46.0	1206534025570773100045
127	57	11	34	55.1	48.6	335265252505705053517721
127	50	13	37*	60.6	48.1	544465125233140124215 01421
127	43	14	40	66.1	47.6	177217722136512275212205743 43
127	36	15	45	71.7	49.5	31460746665220750447645747217 35
127	29	21	46*	77.2	46.9	403114461367670603667530141176155
127	22	23	52	82.7	49.5	123376070404722522435445626637647043
127	15	27	55*	88.2	49.1	22057042445604554770523013762217604353
127	8	31	59*	93.7	49.6	704726405275103065147622427156773313021 7
127	120	1	1*	5.5	14.3	217
127	113	2	4*	11.0	28.6	54505
127	106	3	8*	16.5	38.1	14517623
127	99	4	12	22.0	42.9	2320637377
127	92	5	15*	27.6	42.9	616051466261
127	85	6	18*	33.1	42.9	152055627024155
127	78	7	24	38.6	49.0	35647104545000377
127	71	9	24*	44.1	42.9	6402400420033061235
127	64	10	30*	49.6	47.6	13463425464255213055 35
127	57	11	34	55.1	48.6	25767162011323336611 0015
127	50	13	37	60.6	48.1	72364124311247042327752451
127	43	14	41	66.1	48.8	16563411316762141523202565773
127	36	15	45	71.7	49.5	30331451133650366274656667045 63
127	29	21	48	77.2	49.0	40345676560627416132406164153546 7
127	22	23	52	82.7	49.5	10432444427223350151717052717357441 7
127	15	27	54*	88.2	48.2	37071231012177064120650613540236515175
127	8	31	59*	93.7	49.6	42205646407374623430507547652266541562 57
127	120	1	1*	5.5	14.3	235
127	113	2	4*	11.0	28.6	76533
127	106	3	6*	16.5	28.6	10513165
127	99	4	12	22.0	42.9	2113100037
127	92	5	16	27.6	45.7	530405706075
127	85	6	18	33.1	42.9	145007126304221
127	78	7	23	38.6	46.9	30222671041133777
127	71	9	26	44.1	46.4	5056513565374533677
127	64	10	29*	49.6	46.0	1337626055235540411717
127	57	11	32	55.1	45.7	22260270302304536723 2111
127	50	13	38	60.6	49.4	73066070324015476437747471
127	43	14	40	66.1	47.6	16156210716167256615031425161
127	36	15	45*	71.7	49.5	31451670344423121514743542 52557
127	29	21	47*	77.2	48.0	41103422054005600403633236553 6535
127	22	23	51	82.7	48.6	15157123765535736752045466770546 2071
127	15	27	56	88.2	50.0	24220353103706645134226343657675776433
127	8	31	58*	93.7	48.7	67723705730713321106232450103635656 40527
127	8	31	59*	93.7	49.6	62253554607042413232615635075713701176 51

TABLE III (*Continued*)

n	k	t	B	$r/n(\%)$	$B/r(\%)$	$g(x)$
255	247	1	1*	3.1	12.5	435
255	239	2	5*	6.3	31.2	267543
255	231	3	9*	9.4	37.5	156720665
255	223	4	11*	12.5	34.4	75626641375
255	215	5	17	15.7	42.5	23157564726421
255	207	6	21*	18.8	43.8	16176560567636227
255	199	7	26	22.0	46.4	7633031270420722341
255	191	8	27*	25.1	42.2	2663470176115333714567
255	187	9	27*	26.7	39.7	527553135400013222363351
255	179	10	35*	29.8	46.1	226247107173404324163004 55
255	171	11	39*	32.9	46.4	154162142123423560770616306 37
255	163	12	43*	36.1	46.7	75004155100756025515747245146 01
255	155	13	47	39.2	47.0	37575130054076650157225064646776 33
255	147	14	50*	42.4	46.3	16421301735371655253041653054410117 11
255	139	15	55*	45.5	47.4	46140173206017556157072273024745356744 5
255	131	18	60*	48.6	48.4	21571333147151015126125027744214202416547 1
255	123	19	64	51.8	48.5	120614052242066003717210326516141226272506 267
255	115	21	68	54.9	48.6	60526665572100247263636404600276352556313 472737
255	107	22	70*	58.0	47.3	22205772322066256312417300235347420176574 750154441
255	99	23	75*	61.2	48.1	10656667253473174222741416201574332252411 076432303431
255	91	25	80*	64.3	48.8	67502650303274441727236317247325110755507 62720724344561
255	87	26	83	65.9	49.4	11013676341474323643523163430717204620672 2545273311721317
255	79	27	86*	69.0	48.9	66700035637657500020270344207366174621015 326711766541342355
255	71	29	90	72.2	48.9	24024710520644321515554172112331163205444 250362557643221706035
255	63	30	93*	75.3	48.4	10754475055163544325315217357707003666111 726455267613656702543301
255	55	31	99	78.4	49.5	73154252035011001330152753060320543254143 267550105570444260354736 17
255	47	42	103	81.6	49.5	25335420170626465630330413774062331751233 341454460450050660245525431 73
255	45	43	104	82.4	49.5	15202056055234161131101346376423701563670 024470762373033202157025051541
255	37	45	107*	85.5	49.1	51363302550670074141774472454375304207357 061743234323476443547374030440 03
255	29	47	111	88.6	49.1	30257155366730714655270640123613771153422 423242011741140602546574104036 5037
255	21	55	116	91.8	49.6	12562152570603326560017731536076121032273 414056530745425211531216144665134737 25
255	13	59	120*	94.9	49.6	46417320050525645444265737142500660043306 774454765614031746772135702613446050 0547
511	502	1	1*	1.8	11.1	1021
511	493	2	6	3.5	33.3	1112711
511	484	3	11*	5.3	40.7	1530225571
511	475	4	14	7.0	38.9	1630256304641
511	466	5	17	8.8	37.8	11127246621617 63
511	457	6	24	10.6	44.4	1142677410335765707
511	448	7	28	12.3	44.4	10341223371643722240 05
511	439	8	33	14.1	45.8	15613500646705437774233 45
511	430	9	37*	15.9	45.7	17274003061276201734614316 27
511	421	10	41	17.6	45.6	13177116252672646103606447075 13
511	412	11	46*	19.4	46.5	13375301644103057123161737671471 01
511	403	12	51*	21.1	47.2	15734363036573117627266577246511 203651
511	394	13	56	22.9	47.9	11025103441303333542704074743053412340 33
511	385	14	60*	24.7	47.6	17755460257777123724554521073005303314440 31
511	376	15	65	26.4	48.1	11167447065217255322271626071462162101067 33203
511	367	17	68*	28.2	47.2	11266572025056663230170016522455626144355 11600655

TABLE III (*Continued*)

n	k	t	B	r/n(%)	B/r(%)	$g(x)$
511	358	18	73	29.9	47.7	157445615454545041473341617653515607003776 0411373255
511	349	19	79	31.7	48.8	145501223467575324207450155565737700616552 1557376050313
511	340	20	82*	33.5	48.0	10360754270624746232206620471226116773635 1364110105517777
511	331	21	87*	35.2	48.3	154064472110605057034277240511717745321513 6663677461641457321
511	322	22	91*	37.0	48.1	115361467550611121137436667562467075523652 3645062677061770735073
511	313	23	96*	38.7	48.5	10370430053464111027745164470470707356023 2722463742153673625151743 7
511	304	25	100	40.5	48.3	151073621227213335350235253632070341014722 5273064337077160035254047351
511	295	26	104*	42.3	48.1	111263047753003317004452474772776753275204 6612603077472052247671744467035
511	286	27	108	44.0	48.0	131746340326564504206477532604477573741671 4071756016714523650022734505401471
511	277	28	114*	45.8	48.7	111117075212254710034142277366030225623031 7751245413717303607737426401526326045
511	268	29	118*	47.6	48.6	124116024715136716561537202317022126442722 6437653163043503436310631425301735205601
511	259	30	121*	49.3	48.0	112131411116210153237072224371101446333347 7256025051656614354713760662350433214646117

t is the BCH lower bound on the random-error-correcting-limit, $r=n-k$, $g(x)$ is the code-generating polynomial, and the asterisk denotes that addition of an overall parity-check digit will increase B.

ously available methods would have been computationally prohibitive. The codes are described by their generator polynomials $g(x)$, which are given in the octal notation of Peterson [2, pp. 472–534]; $h(x)$ can of course be found from $h(x)=(x^n-1)/g(x)$.

The Fire codes listed in Table I are those of the "best" Fire codes as found by Wagner [9] for which the lower bound on B as given in [9] is not exact. For comparison this lower bound is also listed in Table I. It will be seen that this lower bound, due to Fire [7], is often very pessimistic.

It may at first seem strange that it is necessary to list the specific $g(x)$ for the BCH codes in Tables II and III. However, unlike the random-error-correcting limit, the burst-correcting limit of a BCH code depends on the particular primitive element used to define the code. This dependence on $g(x)$, or, equivalently, on $h(x)$, could be anticipated from the content of Theorem 3; it is illustrated by numerous instances in Table III where primitive BCH codes with the same n and k, but different $g(x)$, have different B. All of the binary primitive BCH codes of length 63 or less are given in Table III, except for those for which $g(x)$ is the reciprocal of that of a code already given so that its codewords are just the reverse of the other's and hence has the same B. Only a partial list is given in Table III of the primitive codes with lengths 127, 255, and 511. In all cases an asterisk indicates a code for which the addition of an overall parity check increases B.

We remark that it seemed surprising to us, at this late date in the development of coding theory, that there appeared to be so much yet to be said about single-burst-correction for a cyclic code. We remark also that the connection between persymmetric matrices and LFSR's, given in Section III, seems so basic that there well may be applications of this principle to other problems in coding theory.

Acknowledgment

The authors are grateful to H. Schäffner for his programming of the algorithm used to compute the data in Tables I–III. Much of the work by the first author reported here was performed while he was on the technical staff of the Forschungsinstitut AEG–Telefunken in Ulm, West Germany, whose support of this research is gratefully acknowledged.

References

[1] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.

[2] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed. Cambridge, MA: M.I.T., 1972.

[3] B. Elspas and R. A. Short, "A Note on optimum burst-error-correcting codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 39–42, Jan. 1962.

[4] T. Kasami, "Optimum shortened cyclic codes for burst-error-correction," *IEEE Trans. Inform. Theory*, vol. IT-9, pp. 105–109, Jan. 1963.

[5] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122–127, Jan. 1969.

[6] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw–Hill, 1968.

[7] P. Fire, "A class of multiple-error-correcting binary codes for non-independent errors," Sylvania Electric Products, Mountain View, CA, Tech. Rep. RSL-E-2, Mar. 1959.

[8] H. J. Matt, "On burst error correcting ability of cyclic block codes," in *Proc. Int. Conf. on Inform. and System Theory in Digital Comm.* (Techn. Univ. Berlin, W. Germany, Sept. 18–20, 1978), *NTG Band 65*. Berlin: VDE-Verlag, pp. 179–184.

[9] W. Wagner, "Best Fire codes with length up to 1200 bits," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 649–650, Sept. 1970.