over $\text{GF}(q)$. Fact 3 thus implies that

$$x^{n/l} - a = \prod_{j=0}^{n/l-1} \left( x - \left( \beta_j \right)^{l} \right)$$

is the factorization of $(x^{n/l} - a)$ over $\text{GF}(q)$. Hence, any divisor $g(x)$ of $(x^{n/l} - a)$ is of the form $f(x^l)$ where $f(x)$ is a divisor of $(x^{n/l} - a)$. It follows that the codewords in the pseudocyclic code of length $n$ generated by $g(x)$ are obtained by intersecting $l$ codewords from the pseudocyclic code of length $n/l$ generated by $f(x)$. As in the proof of Theorem 2, $l$ is the order of the element $a^{(q-1)/n}$.

**Corollary 3.1:** If $n$ is a divisor of $q - 1$, and the multiplicative order of $a$ is not a divisor of $(q - 1)/n$, then there exist pseudocyclic codes $\text{mod}(x^n - a)$ which are the result of interchanging a pseudocyclic MDS code $\text{mod}(x^{n/l} - a)$.

V. Conclusion

We have studied the existence of pseudocyclic MDS codes of lengths $q + 1$, $q - 1$, and divisors thereof. For $n$ a divisor of $q + 1$, cyclic MDS codes do not exist if both $n$ and $k$ are even. On the other hand, by choosing $a$ to be a suitable quadratic nonresidue, we can always find a pseudocyclic MDS code with these parameters. Thus, pseudocyclic MDS codes are an interesting alternative to cyclic MDS codes in applications where specific values of $n$ and $k$ (for which a cyclic MDS code does not exist) are required. Fortunately, the decoding of pseudocyclic MDS codes is not very different from the decoding of cyclic MDS codes. The pseudocyclic MDS codes are variations of BCH codes, and thus, with slight modifications, the standard BCH decoding algorithms such as the Peterson–Gorenstein–Zierler algorithm, the Berlekamp–Massey algorithm and the Euclidean algorithm [4] can be applied to their decoding. Details of the necessary modifications for the case of negacyclic codes can be found in [7]. Thus, one of the major advantages of using cyclic MDS codes, namely, the efficient decoding techniques, is preserved when using pseudocyclic MDS codes.

Acknowledgment

The authors are grateful to the referees for pointing out [6]. We also owe a special debt of gratitude to referee A who suggested that our original results on negacyclic codes could be generalized to pseudocyclic codes.

References


Manuscript received August 30, 1988; revised December 1, 1989. This work was supported in part by the US Army Research Office under Contracts DAAG 29-84-K-0088 and DAAH04-87-K-0097.

D. V. Sarwate is with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

R. D. Morrison is with ITT Aerospace/Optical Division, Fort Wayne, IN 46801.

IEEE Log Number 9034945.
unavoidable, whereas decoder malfunctions can be detected and avoided simply by checking if the decoder output is a valid codeword in \( \mathcal{C} \). Unfortunately, typical descriptions of BCH decoder algorithms [2]-[6] do not mention the necessity for such simple checks that should always be included in all BCH decoder implementations.

Let \( \mathcal{C} \) denote a \( t \)-error-correcting \((n,k)\) BCH-code whose generator polynomial has \( a^3, a^2, a^i, \ldots, a^2 \) as roots where \( a \) denotes a primitive \( n \)th root of unity. Let \( \mathcal{C} \) denote a positive integer no greater than \( t/2 \). We show that if \( e > t \) and \( \rho \) is in \( V_{e-t} (\hat{c}) \), where \( \hat{c} \) is a codeword in the \((t-1)\)-error-correcting supercode of \( \mathcal{C} \), whose generator polynomial has \( a^{2^0}, a^{2^1}, \ldots, a^{2^{e-t-1}} \) as roots, then the output of a BCH decoder based on the SKHN algorithm is \( \hat{c} \). In either case, if \( \hat{c} \) happens to be a codeword in \( \mathcal{C} \), also, a decoder error occurs, while if \( \hat{c} \) belongs to the supercode but not to \( \mathcal{C} \), a decoder malfunction occurs. Similar results hold for a \( (t-1) \)-error-correcting wide-sense BCH code for which the generator polynomial has \( a^{2^0}, a^{2^1}, \ldots, a^{2^{t-n-1}} \) as roots, and the proofs of these more general statements require only minor modifications to the arguments presented in this correspondence.

II. MALFUNCTION IN BCH DECODERS

Let \( C(x) \), \( E(x) \), and \( R(x) = (C(x)+E(x)) \) respectively denote the transmitted codeword polynomial, the channel error polynomial, and the received BCH polynomial. A decoder decides \( R(x) \) as follows.

Step 1) Compute the syndrome sequence \( S_1, S_2, S_3, \ldots, S_n \) where \( S_i = R(a^i) + C(a^i) + E(a^i) \), \( 1 \leq i \leq 2t \).

Step 2) Compute the coefficients \( \lambda_1, \lambda_2, \ldots, \lambda_n, \nu \), \( \nu \neq t \), of the error-locator polynomial

\[
\lambda(x) = (1 - X_1 x)(1 - X_2 x) \cdots (1 - X_n x) = 1 + \lambda_1 x + \cdots + \lambda_{n-1} x^{n-1} + \lambda_n x^n.
\]

Step 3) Determine the error locations \( X_1, X_2, \ldots, X_n \).

Step 4) Determine the error values \( Y_1, Y_2, \ldots, Y_n \).

Step 5) Correct the errors. For \( i = 1, 2, \ldots, \nu \), subtract \( Y_i \) from \( R_i \), where \( j \) is the logarithm of \( X_i \) to the base \( a \), that is, \( a^j = X_i \).

When \( e > t \), this algorithm may fail to decode \( R(x) \). Generally, a decoder failure is declared whenever one of the following events occurs:

1) the decoder is unable to compute the coefficients of \( \lambda(x) \) in Step 2;
2) the error locations determined in Step 3) are not \( \nu \) distinct \( n \)th roots of unity;
3) some of the error values determined in Step 4) are either zero, or do not belong to the symbol field.

Sometimes, when \( e > t \), the decoder may apparently decode successfully, and yet produce an output vector that is not a codeword at all. We examine decoders based on the PGZ algorithm and the SKHN algorithm more closely to determine the cause of such decoder malfunctions.

A. Malfunction in Decoders Based on the PGZ Algorithm

A conventional BCH decoder based on the PGZ algorithm (henceforth a PGZ decoder) computes the error-locator polynomial in Step 2) and the error values in Step 4) as follows [2], [6].

Step 2a) If at least one \( S_i \) is nonzero, determine \( \nu \), the largest integer not exceeding \( t \) such that the matrix \( M_\nu \)

\[
\begin{pmatrix}
S_1 & S_2 & \cdots & S_\nu \\
S_2 & S_3 & \cdots & S_{\nu+1} \\
\vdots & \vdots & \ddots & \vdots \\
S_\nu & S_{\nu+1} & \cdots & S_{2t-1}
\end{pmatrix}
\]

is nonsingular. The decoder hypothesizes that \( \nu \) errors have occurred. If \( e \leq t \), then \( \nu = e \).

Step 2b) Solve the system of linear equations

\[
M_\nu \left[ \lambda_1, \lambda_2, \ldots, \lambda_n \right]^T = -\left[ S_{\nu+1}, S_{\nu+2}, \ldots, S_{2t-1} \right]^T
\]

(1)

to determine the coefficients of \( \lambda(x) \).

Step 4) Solve the system of linear equations shown next for the error values \( Y_1, Y_2, \ldots, Y_n \).

\[
\begin{pmatrix}
X_1^\nu & X_2^\nu & \cdots & X_n^\nu \\
X_1^2 & X_2^2 & \cdots & X_n^2 \\
\vdots & \vdots & \ddots & \vdots \\
X_1^2 & X_2^2 & \cdots & X_n^2
\end{pmatrix}
\begin{pmatrix}
Y_1 \\
Y_2 \\
\vdots \\
Y_2
\end{pmatrix}
= \begin{pmatrix}
S_1 \\
S_2 \\
\vdots \\
S_n
\end{pmatrix}
\]

Whenever \( R(x) \) is in the decoding sphere of some codeword, the PGZ decoder decodes \( R(x) \) into that codeword. When \( R(x) \) is not in any decoding sphere, a decoder failure may occur in Step 2) because all the matrices \( M_\nu, M_{\nu+1}, \ldots, M_{2t-1} \) examined in Step 2a) are singular even though not all the \( S_i \)'s are zero (and hence errors are known to have occurred). Failure may also occur in Steps 3) and 4) as indicated earlier. Example 1 that follows shows that sometimes a malfunction occurs instead.

Example 1: Let \( a \) denote a primitive element of \( GF(2^4) \) satisfying \( a^4 + a + 1 = 0 \). Consider a \((15,7)\) four-error-correcting cyclic Reed–Solomon code over \( GF(2^4) \) whose generator polynomial has \( a, a^2, \ldots, a^8 \) as roots. The code has minimum distance 9. Suppose that \( e = 5 \) and the error pattern is

\[
E(x) = a^3 x + a^6 x^2 + a^{14} x^{10} + a^8 x^{12} + a^x.15.
\]

\( R(x) \) is at distance 5 from \((x)\), and hence at distance at least 4 from all other codewords. The syndrome values are

\[
R(a) = E(a) = a^{10} = S_1, \quad R(a^2) = E(a^2) = a^2 = S_2,
\]

\[
R(a^3) = E(a^3) = a^8 = S_3, \quad R(a^4) = E(a^4) = a^4 = S_4,
\]

\[
R(a^5) = E(a^5) = a^5 = S_5, \quad R(a^6) = E(a^6) = a^6 = S_6,
\]

\[
R(a^7) = E(a^7) = a^7 = S_7, \quad R(a^8) = E(a^8) = a^8 = S_8.
\]

It is easily verified that the matrices \( M_2, M_3 \) are singular, while \( M_5 \) is nonsingular. Hence, the decoder hypothesizes that two errors have occurred and solves the equations

\[
M_2 \left[ \lambda_1, \lambda_2 \right]^T = -\left[ S_1, S_2 \right]^T
\]
to obtain \( \lambda_2 = a^{11}, \lambda_1 = a^{12} \). Since \( 1 + a^2 x + a^{11} x^2 = (1 - x)(1 - a^2 x + a^{11} x^2) \), the decoder decides that errors have occurred in locations \( X_1 = a^0 \) and \( X_2 = a^1 \). The error values are the solutions to

\[
Y_1 X_1 + Y_2 X_2 = S_1, \quad i.e., Y_1 + a^4 Y_2 = a^{10},
\]

\[
Y_1 X_1^2 + Y_2 X_2^2 = S_2, \quad i.e., Y_1 + a^2 Y_2 = a^2
\]
giving $Y_1 = a^6$, $Y_2 = a^{11}$. Thus, in attempting to correct the errors, the PGZ decoder introduces two additional errors and produces the output vector $C(x) + E'(x)$ where $E'(x)$ is the “residual” error pattern given by

$$E'(x) = a^6 + a^x + a^{x^2} + a^{x^3} + a^{x^4} + a^{11}x^{11} + a^{11}x^{11} + a^{ax^{12}} + a^{ax^{13}}.$$ 

Since $E'(x)$ has weight 7, it cannot be a codeword in $C$, and thus the output decoder $C(x) + E'(x)$ cannot be a codeword in $C$.

Example 1 shows that a PGZ decoder can malfunction when confronted with an error pattern which it is not designed to correct. Indeed, the decoder actually may have a great deal of confidence in its results since it “finds” only two errors while the code is actually capable of correcting four errors. Of course, decoder malfunction can be easily avoided by checking if the decoder output is a valid codeword, or, equivalently, by checking if the hypothesized error pattern $(a^6 + a^{11}x^{11}$ in Example 1) does indeed produce the syndrome values computed from $R(x)$.

It is easy to avoid decoder malfunction, and thus it is quite remarkable that typical descriptions of the PGZ decoding algorithm [2], [6] do not even mention the necessity of such checks for avoiding decoder malfunction.

We now exhibit one of the possible causes for decoder malfunction in PGZ decoders. When $e$ errors have occurred, the decoder actually has available to it the following set of $2t - e$ linear equations that relate the syndromes to the coefficients of the error locator polynomial $\Lambda(x)$:

$$S_{L_1} + S_{L_2} + \cdots + S_{L_{2t-1}} = -S_{L_{1+e}}, \quad 1 \leq j \leq 2t - e.$$ 

When $e < t$, there are at least as many equations as unknowns, and the equations can be solved for the $L_i$. Of course, the decoder does not know the value of $e$. However, suppose that $e < t$, and consider (2) for $j = 1, 2, \cdots, t$. These $t$ equations imply that the $(e + 1)$th row of the $t \times t$ matrix $M_t$ is a linear combination of the first $e$ rows. Similarly, for each of the matrices $M_{t, j_1}, M_{t, j_2}, \cdots, M_{t, j_e}$, the $(e + 1)$th row is a linear combination of the first $e$ rows. Hence, all these matrices are singular. On the other hand, if $e \leq t$, then $M_e$ is nonsingular [2], [4], [6]. Thus, the PGZ decoder finds the largest integer $\nu$ such that $M_\nu$ is nonsingular, and hypothesizes that $e = \nu$. Whenever $e \leq t$, this hypothesis is correct and the decoder successfully corrects the error locations and error values.

Let $i_0$ denote a positive integer no greater than $t/2$. Suppose that $e > t$ and that $R(x)$ is in the decoding sphere $C(x)$ where $C(x)$ is a codeword in a $(t - i_0, \nu)$-error-correcting BCH supercode of $C$. Suppose that the generator polynomial of the supercode $\nu^*$ has $a, a^2, \cdots, a^{2t-1}$ as roots. Consider what would happen if $R(x)$ were decoded using a decoder for $\nu^*$. It is helpful to imagine that $\nu^*$ is being used instead of $C$, that $C(x)$ is transmitted and $R(x)$ is received, and that $f \leq t - 2i_0$ errors have occurred. In Step 2a), the decoder for $\nu^*$ would find that the matrices $M_{e, i_0}, M_{e, i_0+1}, \cdots, M_{e, e}$ are singular but $M_e$ is nonsingular, and would hypothesize that $e$ errors have occurred. Then, in Step 2b), the decoder would solve

$$M_e[\Lambda_{e, 1}, \Lambda_{e, 2}, \cdots, \Lambda_{e, e}]^T = -[S_{e, 1}, S_{e, 2}, \cdots, S_{e, e}]^T$$

for the $\Lambda_i$'s. The decoder would then execute Steps 3)–5) and thus decode $R(x)$ into $C(x)$. Now, under the helpful hypothesis that $C(x)$ is transmitted and $R(x)$ is received and that $f$ errors have occurred, the decoder for $\nu^*$ has available the following set of $2t - (i_0) - f$ equations of which only the first $f$ are used for determining the $\Lambda_i$'s:

$$S_{e, 1} \Lambda_1 + S_{e, 1+1} \Lambda_{1+1} + \cdots + S_{e, e-1} \Lambda_{e-1} = -S_{e, e}, \quad 1 \leq j \leq 2t - (i_0) - f. \quad (3)$$

But, $2t - (i_0) - f \geq 2t - i_0 - (t - 2i_0) = t$, and the $t$ equations obtained by setting $j = 1, 2, \cdots, t$ in (3) imply that the $(f + 1)$th row of the $t \times t$ matrix $M_t$ is a linear combination of the first $f$ rows of $M_t$. We conclude that whenever $e > t$ and $R(x)$ is in the decoding sphere $C(x)$, the matrices $M_1, M_2, \cdots, M_{t-1}, M_{t-0}, M_{t-1-1}, \cdots, M_{t-1-1}$ are all singular, while $M_t$ is nonsingular. Now, $M_{t-1}, M_{t-1-1}, \cdots, M_{t-1-1}$ are not examined by the decoder for $\nu^*$ at all, but they are examined by the decoder for $C$. Hence, when the decoder for $C$ is used to decode $R(x)$, it mistakenly hypothesizes that $f$ errors have occurred. Consequently, in Step 2b), the decoder for $C$ solves the same system of linear equations

$$M_t[\Lambda_{t, 1}, \Lambda_{t, 2}, \cdots, \Lambda_{t, t}]^T = -[S_{t, 1}, S_{t, 2}, \cdots, S_{t, t}]^T$$

as the decoder for $\nu^*$. In fact, both decoders do the same computations in Steps 2b)–5), and in both cases, the decoder output is $C(x)$, a codeword from $C$. If $C(x)$ happens to belong to $C$, a decoding error occurs, while if $C(x)$ does not belong to $C$, a decoder malfunction occurs.

PGZ decoders go astray because “not all the syndrome values are used in the decoding.” In Step 2a), the value of $v$ is determined from $S_1, S_2, \cdots, S_t$ only and $S_2$ is completely ignored. Furthermore, if $v < t$, then $S_2$ is not used at all in the decoding. In contrast, Berlekamp–Massey decoders [2], [6] do use all the syndromes. When a Berlekamp–Massey decoder has iteratively solved the first $v$ equations of the form (2) for an error-locator polynomial $\Lambda(x)$ of degree $v < t$ (and thus formulated the hypothesis that $v$ errors have occurred), it checks whether the remaining $2t - 2v$ equations of the form (2) are also satisfied by the hypothesized coefficients of $\Lambda(x)$. The decoder does this by continuing to iterate and verifying that the “discrepancy” is zero at each iteration. It is important to carry out such checks in the PGZ decoder as well. When the decoder hypothesis that $v < t$, it should check whether the $\Lambda_i$'s satisfy the remaining $2t - 2v$ equations available to the decoder or if $S_j = \Sigma_{j}y(x)^{v+1}$ for $2v < j \leq 2t$. Equivalently, after correcting the errors in Step 5), the decoder should check whether the result is a valid codeword. Thus, by taking a few pains and performing a simple check, the decoder can easily avoid a malfunction.

B. Malfuction in Decoders Based on the SKHN Algorithm

A conventional BCH decoder based on the SKHN algorithm (henceforth an SKHN decoder) finds not only the error-locator polynomial $\Lambda(x)$ of degree $v$ but also the error-evaluator polynomial $\eta(x)$ which is defined as

$$\eta(x) = \sum_{i=1}^{v} x^i \cdot \prod_{j=1}^{j \neq i} (1 - X_i x). \quad (4)$$

for a $t$-error-correcting wide-sense BCH code whose generator polynomial has roots $a^i, a^{i+1}, \cdots, a^{i+2t-1}$. In Step 2), the SKHN decoder uses the extended Euclidean algorithm for computing the greatest common divisor of two polynomials to find both $\Lambda(x)$ and $\eta(x)$ as follows [2], [4], [5].

Step 2a) Set $a_0(x) = x^{2t}$, $a_1(x) = S_0 + S_{i_0}x + \cdots + S_{i_0+1}x^{2t}$. Define the auxiliary polynomials $b_0(x) = 0, b_1(x) = 1$. For $i = 2, 3, \cdots$, compute the quotient $q_i(x)$ and the remainder $a_i(x)$ when
function and produce an output vector which is not a codeword in \( C \).

We state some of the properties (see e.g., [1],[2]) of the Euclidean algorithm that are useful in exhibiting one of the possible causes for decoder malfunction in SKHN decoders. The Euclidean algorithm computes a sequence of remainder polynomials \( a_{-j}(x) \) of strictly decreasing degrees and auxiliary polynomials \( b_j(x) \) of strictly increasing degree satisfying the congruence

\[
b_j(x)a_{-j}(x) = a_{j}(x) \mod a_{j-1}(x), \quad i = 1, 2, 3, \cdots.
\]

The congruence can be rewritten as

\[
b_j(x) + d_j(x)a_{-j}(x) = a_{j}(x), \quad i = 1, 2, 3, \cdots
\]

where \( d_j(x) = d_{-j}(x) - q_{j}(x)a_{j-1}(x) \) is another auxiliary polynomial computed via a recurrence identical to the one used to compute \( b_j(x) \) except that the initial conditions are \( d_0(x) = 1 \), \( d_1(x) = 0 \). The degrees of these polynomials satisfy

\[
\deg b_j + \deg a_{-j} = \deg a_j.
\]

Since the degrees of the remainders are strictly decreasing, we have

\[
\deg b_j + \deg a_j < \deg a_{j-1}.
\]

Let \( t_0 \) denote a positive integer no greater than \( t/2 \). Suppose that \( \varepsilon > \tau \) and that \( R(x) \) is in the decoding sphere \( V_{\varepsilon}(\hat{c}(x)) \) where \( \hat{c}(x) \) is a codeword in a \((t - t_0)\)-error-correcting BCH supercode of \( \varepsilon \). Suppose that the generator polynomial of the supercode \( \varepsilon^* \) has \( a_{2i+1}^*, \ldots, a_{2i}^* \) as roots. As before, \( \varepsilon^* \) being used instead of \( \varepsilon \), that \( \hat{c}(x) \) is transmitted and \( R(x) \) is received, and that \( f \leq t - 2t_0 \) errors have occurred. The syndrome sequence is \( S_{2t_0+1}, S_{2t_0+2}, \cdots, S_{2t} \). The decoder sets

\[
\tilde{a}(x) = x^{2t_0+1},
\]

and then uses the Euclidean algorithm to find \( \tilde{\Lambda}(x) \) and \( \tilde{\eta}(x) \). Thus, the decoder finds the sequence of remainder polynomials \( \tilde{a}(x) \) and auxiliary polynomials \( \tilde{b}(x) \) in Step 2a. Since \( R(x) \) lies in the decoding sphere of \( \hat{c}(x) \), decoding failure cannot occur, that is, the Euclidean algorithm can not terminate with a remainder (greatest common divisor) of degree \( t - t_0 \) or more. Thus the decoder can find the index \( j \) such that \( \deg \tilde{a}_j < t_0 \) and \( \deg \tilde{a}_{j-1} \geq t - t_0 \), and set

\[
\tilde{\Lambda}(x) = \tilde{b}_0(x) / \tilde{b}_0(0) \quad \text{and} \quad \tilde{\eta}(x) = \tilde{a}_j(x) / \tilde{b}_0(x).
\]

Since \( R(x) \) lies in the decoding sphere of \( \hat{c}(x) \), \( \tilde{\Lambda}(x) \) and \( \tilde{\eta}(x) \) are valid error-locator and error-evaluator polynomials, and hence the SKHN decoder for \( \varepsilon^* \) can complete Steps 3–5 and decode \( R(x) \) into \( \hat{c}(x) \).

Several properties of the polynomials \( \tilde{\Lambda}(x) \) and \( \tilde{\eta}(x) \) are of interest. We note that \( \tilde{\Lambda}(x) \) and \( \tilde{\eta}(x) \) are relatively prime polynomials with degrees \( \deg \tilde{\Lambda} = f \) and \( \deg \tilde{\eta} = \deg \tilde{a}_j = f - 1 \leq t - 2t_0 - 1 \). The stopping condition of Step 2b gives only the looser bound \( \deg \tilde{\eta} = \deg \tilde{a}_j < t - t_0 \). Next, note from (7) that \( \deg \tilde{a}_{j-1} < \deg \tilde{a}_j = f - 1 \). Hence \( \deg \tilde{a}_{j-1} \leq t - 2t_0 \), we have that \( \deg \tilde{a}_{j-1} \geq t \) whereas the stopping condition of Step 2b gives only the looser bound \( \deg \tilde{a}_{j-1} \geq t - t_0 \). Finally, we note that in Step 4, the decoder computes the error values
using the formula
\[ Y_t = -(X_t)^{-2\nu_t} \tilde{\eta}(X_t^{-1})/\tilde{\lambda}(X_t^{-1}). \]  
(8)

Consider the SKHN decoder for \( \mathcal{E}' \). It computes the syndrome sequence \( S_0, S_1, \ldots, S_{2\nu_t}, S_{2\nu_t+1}, S_{2\nu_t+2}, \ldots, S_{2\nu_t+2\nu_t} \), and executes the Euclidean algorithm with
\[ a_0(x) = x^{2\nu_t} \tilde{\eta}(x) \]  
and
\[ a_i(x) = S_i + S_{i+1} x + \cdots + S_{2\nu_t} x^{2\nu_t-1} + S_{2\nu_t+1} x^{2\nu_t} + \cdots + S_{2\nu_t+2\nu_t-1} x^{2\nu_t} \]
\[ = [S_i + S_{i+1} x + \cdots + S_{2\nu_t} x^{2\nu_t-1}] + x^{2\nu_t} \tilde{\eta}(x). \]  
(9)

In Step 2a), the SKHN decoder for \( \mathcal{E}' \) finds the sequence of remainder polynomials \( a_i(x) \) and auxiliary polynomials \( b_i(x) \) where
\[ b_i(x) a_i(x) + d_i(x) a_{i+1}(x) = a_i(x), \quad i = 1, 2, 3, \ldots. \]
Equations (9) and (10) show that the coefficients of the high degree terms in \( a_i(x) \) are the same as the coefficients of the high degree terms in \( \tilde{\eta}(x) \), and that the coefficients of the high degree terms in \( a(x) \) are the same as the coefficients of the high degree terms in \( \tilde{\eta}(x) \). Since the quotient of two polynomials depends only on the high degree terms, the quotients (and hence the auxiliary polynomials) computed by the two decoders should be identical for several iterations. In fact, for \( 0 \leq i \leq j \), the auxiliary polynomials \( b_i(x) \) computed by the SKHN decoder for \( \mathcal{E}' \) are the same as the auxiliary polynomials \( b_i(x) \) computed by the SKHN decoder for \( \mathcal{E} \), and the remaining polynomials \( a_i(x) \) and \( a_i(x) \) are very closely related. The details are as follows.

Let \( m \) denote the unique index such that \( \deg \tilde{\eta}_m > t - i_0 \) and \( \deg \tilde{\eta}_{m+i} \leq t - i_0 \). But we know that \( \deg \tilde{\eta}_{m+i} \geq t \) and \( \deg \tilde{\eta}_m < t - i_0 \), so that it must be that \( m = j - 1 \). From (9) and (10) and Lemmas 8.6 and 8.7 of [1], we have that \( b_i(x) = \tilde{b}_i(x) \) and \( d_i(x) = \tilde{d}_i(x) \) for \( i = 0, 1, 2, \ldots, m + 1 = j \). In other words, both SKHN decoders find the same auxiliary polynomials for \( 0 \leq i \leq j \).

The SKHN decoder for \( \mathcal{E}' \) finds that
\[ \tilde{b}_i(x) a_i(x) + \tilde{d}_i(x) x^{2\nu_t} = a_i(x), \quad 0 \leq i \leq j. \]  
(11)

while the SKHN decoder for \( \mathcal{E} \) finds
\[ \tilde{b}_i(x) a_i(x) + \tilde{d}_i(x) x^{2\nu_t} = a_i(x), \quad 0 \leq i \leq j. \]  
(12)

From (9)-(12) we get that the remainder polynomials found by the SKHN decoder for \( \mathcal{E}' \) are given by
\[ a_i(x) = x^{2\nu_t} \tilde{\eta}(x) + \tilde{b}_i(x) [S_i + S_{i+1} x + \cdots + S_{2\nu_t} x^{2\nu_t-1}], \quad 0 \leq i \leq j, \]
and thus are related to the remainder polynomials found by the SKHN decoder for \( \mathcal{E} \). Next, we note that
\[ \deg x^{2\nu_t} \tilde{\eta}_m(x) = 2i_0 + \deg \tilde{\eta}_m(x) \geq t, \]
\[ \deg \tilde{b}_i(x) \left[ S_i + S_{i+1} x + \cdots + S_{2\nu_t} x^{2\nu_t-1} \right] < f + 2i_0 - 1 \leq t - 1, \]
\[ \deg x^{2m} \tilde{\eta}_m(x) = 2i_0 + \deg \tilde{\eta}_m(x) \leq 2i_0 + f - 1 \leq t - 1, \]  
(13)
\[ \deg \tilde{b}_i(x) \left[ S_i + S_{i+1} x + \cdots + S_{2\nu_t} x^{2\nu_t-1} \right] \leq f + 2i_0 - 1 \leq t - 1. \]  
(14)

Thus, when the SKHN decoder for \( \mathcal{E}' \) invokes the stopping condition of Step 2b), it finds \( \deg \tilde{\eta}_m(x) \geq t \) and \( \deg \tilde{\eta}_m(x) < t \).

Therefore, the SKHN decoder for \( \mathcal{E}' \) sets \( \Lambda(x) = \tilde{\lambda}(x) \) and \( \eta(x) = \tilde{\eta}(x)/\tilde{\lambda}(x) \) as the error-locator and error-evaluator polynomial respectively where
\[ \Lambda(x) = \tilde{\lambda}(x) \]
and
\[ \eta(x) = x^{2\nu_t} \tilde{\eta}(x) + \tilde{\lambda}(x) [S_i + S_{i+1} x + \cdots + S_{2\nu_t} x^{2\nu_t-1}] \]  
(15)

These polynomials satisfy the congruence
\[ \Lambda(x) a_i(x) = \eta(x) \mod x^{2\nu_t}. \]

Furthermore, since \( \tilde{\lambda}(x) \) and \( \tilde{\eta}(x) \) are relatively prime, then so are \( \Lambda(x) \) and \( \eta(x) \). Since \( \Lambda(x) = \tilde{\lambda}(x) \), the SKHN decoder for \( \mathcal{E} \) finds the same error locations as the SKHN decoder for \( \mathcal{E}' \). In Step 4), the SKHN decoder for \( \mathcal{E}' \) uses the formula
\[ Y_t = -(X_t^{-1})^{2\nu_t} \tilde{\eta}(X_t^{-1})/\tilde{\lambda}(X_t^{-1}) \]
which is just the formula (8) used by the SKHN decoder for \( \mathcal{E} \).

In short, both decoders find the same error locations and error values and both decoders decode \( R(x) \) into \( \hat{C}(x) \). If \( \hat{C}(x) \) happens to be a codeword in \( \mathcal{E}' \) also, the SKHN decoder for \( \mathcal{E}' \) commits a decoding error, while if \( \hat{C}(x) \) is not a codeword in \( \mathcal{E}' \), the SKHN decoder for \( \mathcal{E}' \) malfunctions.

SKHN decoders go astray for exactly the same reason that PGZ decoders do, namely, not all the syndrome values are used in determining \( \Lambda(x) \). Since only the coefficients of the high degree terms determine the sequence of quotient polynomials \( q_i(x) \), only the coefficients of the high degree terms of \( a_i(x) \) are used by the SKHN decoder in computing \( \Lambda(x) \), while the coefficients of the lower degree terms are not used at all. A simple method for detecting an impending malfunction in an SKHN decoder is to check whether \( \deg \eta \geq \deg \Lambda \). This test can be performed upon the completion of Step 2b). If the condition is satisfied, the received vector is not within the decoding sphere of any codeword, and therefore, either a decoder malfunction or a decoder failure is certain to occur. Hence, an SKHN decoder should declare a decoding failure if it finds that \( \deg \eta \geq \deg \Lambda \).

On the other hand, if \( \deg \eta < \deg \Lambda \), a decoder malfunction cannot possibly occur, that is, testing whether \( \deg \eta < \deg \Lambda \) suffices to detect all instances of impending decoder malfunction. We are indebted to a very perspicacious anonymous reviewer for pointing this out to us.) When \( \deg \eta < \deg \Lambda \), the decoder either decodes \( R(x) \) into a codeword in \( \mathcal{E}' \), or if fails to decode because the roots of \( \Lambda(x) \) do not consist of \( \nu \) distinct \( n \)th roots or because the error values are not nonzero elements of the symbol field. This fact is proved in the Appendix.

III. DISCUSSION

We have exhibited the phenomenon of decoder malfunction in PGZ decoders and SKHN decoders. Also, we have characterized some classes of error patterns that cause these decoders to malfunction, and have noted that with both types of decoders, a decoder malfunction is quite easy. The cause of decoder malfunction seems to be that PGZ and SKHN decoders do not take into account all of the syndrome sequence in computing \( \Lambda(x) \). We also noted that Berlekamp–Massey decoders do not appear to exhibit this form of misbehavior because they do use all of the syndrome sequence when computing \( \Lambda(x) \) and \( \eta(x) \). Whether Berlekamp–Massey decoders can malfunction in other ways is an open question. Nonetheless, it is worth noting that certain high-speed decoders based on a modified version of the Berlekamp–Massey algorithm can malfunction.
It has been suggested [3] that, to speed up the decoding process, when a Berlekamp–Massey decoder has hypothesized an errorlocator polynomial \( \Lambda(x) \) of degree \( \nu < t \), then, of the \( 2t - \nu \) equations

\[
S_{j,\nu} + S_{j,\nu-1} \Lambda_{\nu-1} + \cdots + S_{j,\nu-t} \Lambda_{\nu-t} = -S_{j,\nu-t},
\]

\( 1 \leq j \leq 2t - \nu \) \hspace{1cm} (16)

available to the decoder, only the \( t \) equations for \( j = 1, 2, 3, \cdots, t \) need be checked to see if they are satisfied. In other words, the decoder need not verify that all the other “discrepancies” are zero. When \( \nu < t \), this strategy gives correct results, and since the number of iterations has been reduced, the decoder can decode at higher speed than a more staid decoder that painstakingly checks all \( 2t - \nu \) equations. Unfortunately, such a high-speed Berlekamp–Massey decoder is liable to malfunction. When confronted with the error pattern \( E(x) \), however, it computes the same error-locator polynomial \( \Lambda(x) = 1 + a_1 x + a_2 x^2 \) as the PGZ decoder. Thus, the high-speed Berlekamp–Massey decoder malfunctions in the same way the PGZ decoder does and for the same reason, namely that not all of the syndrome sequence is used in determining \( \Lambda(x) \). Whenever \( R(x) \) is in the decoding sphere \( V_{\nu-2} (\mathbb{C}(x)) \), where \( \mathbb{C}(x) \) is a code word in the \( (t - \nu) \)-error-correcting BCH supercode \( C \) whose generator polynomial has \( \alpha_1, \alpha_2, \ldots, \alpha_{2t-2\nu} \) as roots, a high-speed Berlekamp–Massey decoder finds the appropriate \( \Lambda(x) \) for decoding \( R(x) \) into \( C(x) \), and then verifies only that the \( t \) equations (16) are satisfied. Since these \( t \) equations form a subset of the \( 2(t - \nu) - \gamma \geq t \) equations (3), the high-speed Berlekamp–Massey decoder malfunctions when it attempts to decode \( R(x) \). In short, the high-speed Berlekamp–Massey decoder [3] does produce results more quickly than a more staid Berlekamp–Massey decoder, but it is also liable to malfunction and to produce output vectors that are not codewords at all.

We have shown that BCH decoders may malfunction and produce output vectors that are not codewords at all. Such malfunctions increase the probability of decoder error (unecessarily), but can be easily avoided by simply declaring a decoder failure whenever the putative output vector is not a codeword. Such checks should be made an integral part of all BCH decoders.

**APPENDIX**

Let \( S(x) = S_1 + S_2 x + S_3 x^2 + \cdots + S_{t-\nu+1} x^{t-\nu-1} \) denote the syndrome polynomial, and suppose that the decoder has found polynomials \( \Lambda(x) \) and \( \eta(x) \) such that

\[
\Lambda(x) S(x) = \eta(x) \mod x^{2^\nu},
\]

\( \deg \eta < \deg \Lambda = \nu \), \hspace{1cm} (A.1)

and

\[
\Lambda(0) = 1.
\]

(A.3)

We may assume \( \Lambda(x) = (1 - X_1 x)(1 - X_2 x) \cdots (1 - X_\nu x) \) where \( X_1, X_2, \ldots, X_\nu \) are distinct \( n \)th roots of unity and that \( Y_i = -\eta(X_i^{-1})/\Lambda(X_i^{-1}), \ i = 1, 2, \ldots, \nu \) are nonzero elements of the symbol field, since the decoder will declare a failure if these conditions do not hold. Let \( T(x) = T_1 + T_2 x + T_3 x^2 + \cdots + T_{2\nu} x^{2\nu-1} \) denote the syndrome polynomial corresponding to the error pattern hypothesized by the decoder, that is, to the error locations and values \( (X_1, y_1), (X_2, y_2), \ldots, (X_\nu, y_\nu) \).

Then

\[
\Lambda(x) T(x) = \prod_{i=1}^{t} \left( (1 - X_i x) \sum_{j=1}^{\nu} Y_i(X_i^{-j}) \right) = \sum_{i=1}^{t} \left( -\eta(X_i^{-1})/\Lambda(X_i^{-1}) \right) \prod_{j=i}^{\nu} (1 - X_i x) \sum_{k=1}^{\nu} X_i^{-k} T_k(X_i^{-1}).
\]

(A.4)

Now, the sum on \( i \) is clearly \( \sum_{i=1}^{\nu} X_i^{-1} T_i(x) = \sum_{i=1}^{\nu} X_i^{-1} (1 - X_i x^{2\nu}) = 0 \) while

\[
\Lambda(X_i^{-1}) = -X_i \prod_{j=i+1, \neq i}^{\nu} (1 - X_i X_j^{-1}).
\]

Consequently, we have

\[
\Lambda(x) T(x) = \sum_{j=1}^{\nu} \eta(X_j^{-1}) \prod_{i=1, \neq j}^{\nu} (1 - X_i x)/(1 - X_i X_j^{-1})
\]

\[
- \sum_{j=1}^{\nu} \eta(X_j^{-1}) (x - X_j)^{2\nu} \prod_{i=1, \neq j}^{\nu} (1 - X_i x)/(1 - X_i X_j^{-1}).
\]

(A.5)

The first sum on the right-hand side of (A.5) is the Lagrange interpolation formula for the unique polynomial that has degree less than \( \nu \) and that takes on value \( \eta(X_j^{-1}) \) at \( X_j^{-1}, j = 1, 2, \ldots, \nu \). Since \( \deg \eta < \nu = \deg \Lambda \), this unique polynomial must be \( \eta(x) \) itself, and we get

\[
\Lambda(x) T(x) = \eta(x) \mod x^{2\nu}.
\]

(A.6)

From (A.1) and (A.3) it follows that \( S(x) = T(x) \). Since \( S(x) \) is the syndrome of the received vector while \( T(x) \) is the syndrome of the hypothesized error pattern, we see that when the decoder subtracts the hypothesized error values from the hypothesized error locations, the resulting word has zero syndrome, that is, the decoder output is a valid codeword. Thus, decoder malfunction cannot occur when \( \deg \eta < \deg \Lambda \). It is interesting to note that the assumption that \( \deg \Lambda \leq t \) is not required to arrive at this conclusion.

**REFERENCES**


