Transactions Papers

A View of Gaussian Elimination Applied to Early-Stopped Berlekamp–Massey Algorithm

Chih-Wei Liu, *Member, IEEE*, and Chung-Chin Lu, *Member, IEEE*

***Abstract*—In this paper, we adopt a restricted Gaussian elimination on the Hankel structured augmented syndrome matrix to reinterpret an early-stopped version of the Berlekamp–Massey algorithm in which only $(t + e)$ iterations are needed to be performed for the decoding of BCH codes up to t errors, where e is the number of errors actually occurred with $e \leq t$, instead of the $2t$ iterations required in the conventional Berlekamp–Massey algorithm. The minimality of $(t+e)$ iterations in this early-stopped Berlekamp–Massey (ESBM) algorithm is justified and related to the subject of simultaneous error correction and detection in this paper. We show that the multiplicative complexity of the ESBM algorithm is upper bounded by $(te + e^2 - 1) \forall e \leq t$ and except for a trivial case, the ESBM algorithm is the most efficient algorithm for finding the error-locator polynomial.**

***Index Terms*—BCH codes, Berlekamp–Massey algorithm, decoding, error-correcting codes, Gaussian elimination.**

I. Introduction

Let $S_j, j = 1, 2, \ldots, 2t$ be the syndromes for a transmitted codeword of a BCH code with design error-correcting capability t. Suppose there are exactly e errors, $e \leq t$. The coefficients of the error-locator polynomial $\Lambda(x) = \sum_{i=0}^{e} \Lambda_i x^i$ with $\Lambda_0 = 1$ satisfy [4]–[6]

$$\sum_{k=0}^{e} \Lambda_k S_{j+e-k} = 0, \quad \forall 1 \leq j \leq 2t - e. \tag{1}$$

With its low complexity of order $O(t^2)$, the Berlekamp–Massey algorithm is one of the most popular algorithms for computing the error-locator polynomial. The Berlekamp–Massey algorithm, as discussed in [7] and [8] interprets the linear recurrence in (1) as a shift-register synthesis problem. The determination of the error-locator polynomial then becomes the synthesis of a minimum-length linear-feedback shift register (LFSR) to generate a known, prescribed sequence of syndromes. In generalizing the Berlekamp–Massey algorithm to decode cyclic codes up to the Hartmann–Tzeng bound and the Ross bound in [1], the determination of the error-locator polynomial is reformulated as the problem of finding the smallest initial set of linearly dependent columns in a matrix formed by syndromes. An algorithm, called the fundamental iterative algorithm (FIA), which is a kind of restricted Gaussian elimination, is proposed in [1] to solve the problem of finding the smallest initial set of linearly dependent columns in a generic matrix. It is shown there that the Berlekamp–Massey algorithm is equivalent to a refinement of the FIA on a $2t \times 2t$ syndrome matrix of the form

$$\begin{pmatrix} S_1 & S_2 & \cdots & S_{2t-1} & S_{2t} \\ S_2 & S_3 & \cdots & S_{2t} & Z_{2t+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{2t} & Z_{2t+1} & \cdots & Z_{4t-2} & Z_{4t-1} \end{pmatrix} \tag{2}$$

where the entries Z_j, $2t + 1 \leq j \leq 4t - 1$, designate "don't cares" and can be so specified as to satisfy the linear dependence relation in (1).

However, a more efficient version of the Berlekamp–Massey algorithm is developed in [2] and [3] and requires only $(t + e)$ iterations to be performed for the decoding of BCH codes up to t errors, where e is the number of errors actually occurred with $e \leq t$, instead of the $2t$ iterations in the conventional Berlekamp–Massey algorithm.

To begin with our excursion from the Gaussian elimination to this early-stopped Berlekamp–Massey (ESBM) algorithm, it turns out that the $2t \times 2t$ syndrome matrix in (2) is too big to be taken. To present a syndrome matrix of the right size as hinted in [2], we first state the following basic theorem which is the foundation of the Gorenstein–Zierler decoding method. For a proof of this theorem, please see [9] or any of the textbooks in [4]–[6].

Theorem 1: Suppose that the number of errors actually occurred is e and $e \leq t$. Then the $v \times v$ syndrome matrix $\boldsymbol{S}_{v\times v}$, where

$$\boldsymbol{S}_{v\times v} = \begin{pmatrix} S_1 & S_2 & \cdots & S_v \\ S_2 & S_3 & \cdots & S_{v+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_v & S_{v+1} & \cdots & S_{2v-1} \end{pmatrix}$$

is nonsingular if $v = e$ and singular if $e < v \leq t$. □

This theorem leads naturally to a top-down approach, beginning with the matrix $\boldsymbol{S}_{t\times t}$ and ending with the matrix $\boldsymbol{S}_{e\times e}$, to check their nonsingularity in order to determine the number e of errors actually occurred, provided that $e \leq t$ (see [4]), and after

Paper approved by T.-K. Truong, the Editor for Coding Theory and Techniques of the IEEE Communications Society. Manuscript received August 21, 2004; revised July 28, 2006. This work was supported by the National Science Council, Taiwan, R.O.C., under Contract NSC89-2213-E-007-060.

C.-W. Liu is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan, R.O.C.

C.-C. Lu is with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TCOMM.2007.898827

the nonsingular square matrix $\boldsymbol{S}_{e\times e}$ stands out, the following matrix equation:

$$\boldsymbol{S}_{e\times e}\begin{pmatrix}-\Lambda_e\\ \vdots\\ -\Lambda_1\end{pmatrix}=\begin{pmatrix}S_{e+1}\\ \vdots\\ S_{2e}\end{pmatrix}$$

can be solved by applying Gaussian elimination first and then backward substitution on the $e\times(e+1)$ augmented syndrome matrix

$$\left(\begin{array}{cccc|c}S_1 & S_2 & \cdots & S_e & S_{e+1}\\ S_2 & S_3 & \cdots & S_{e+1} & S_{e+2}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ S_e & S_{e+1} & \cdots & S_{2e-1} & S_{2e}\end{array}\right).$$

Now, for a bottom-up approach, consider the maximal possible $t\times(t+1)$ augmented syndrome matrix

$$\boldsymbol{S}=\left(\begin{array}{cccc|c}S_1 & S_2 & \cdots & S_t & S_{t+1}\\ S_2 & S_3 & \cdots & S_{t+1} & S_{t+2}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ S_t & S_{t+1} & \cdots & S_{2t-1} & S_{2t}\end{array}\right). \tag{3}$$

From (1), the $(e+j)$th column $\boldsymbol{\xi}_{e+j}$ of $\boldsymbol{S}$ is a linear combination of its previous (left) columns, i.e.,

$$\Lambda_e\boldsymbol{\xi}_j+\Lambda_{e-1}\boldsymbol{\xi}_{j+1}+\cdots+\Lambda_1\boldsymbol{\xi}_{e+j-1}+\boldsymbol{\xi}_{e+j}=0 \tag{4}$$

for each j, $1\le j\le t+1-e$. On the other hand, the first e columns of $\boldsymbol{S}$ are linearly independent since the $e\times e$ principal submatrix $\boldsymbol{S}_{e\times e}$ of $\boldsymbol{S}$ is nonsingular by Theorem 1 and then the e columns of $\boldsymbol{S}_{e\times e}$ are linearly independent. Furthermore, the expression of the $(e+1)$th column $\boldsymbol{\xi}_{e+1}$ of $\boldsymbol{S}$ as a linear combination of the first e columns of $\boldsymbol{S}, \boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_e$, as in (4) must be unique, i.e., the coefficients Λ_i's of the error-locator polynomial can be uniquely determined once the first (i.e., the $(e+1)$th) column of $\boldsymbol{S}$ is expressed as a linear combination of its previous columns in $\boldsymbol{S}$. We summarize the above discussion in the following corollary.

Collorary 2: Suppose that the number of errors actually occurred is e and $e\le t$. Then the first column of the augmented syndrome matrix $\boldsymbol{S}$ being a linear combination of its previous columns is $\boldsymbol{\xi}_{e+1}$, and the linear combination coefficients determine the error-locator polynomial. □

A variant of Gaussian elimination on the rows of the augmented syndrome matrix $\boldsymbol{S}$ in (3) is adopted in [10] to determine the coefficients of the error-locator polynomial. Although this algorithm is shown to be more efficient than the conventional Berlekamp–Massey algorithm when the number e of errors actually occurred is very small, it is still subject to the same high complexity of order $O(t^3)$ as the Gaussian elimination operates on a generic matrix of dimension t. This is because that the row operations are not the right operations to be performed on the augmented syndrome matrix $\boldsymbol{S}$ and the structural properties of the augmented syndrome matrix $\boldsymbol{S}$ are not exploited there.

Structural properties of other syndrome matrices than $\boldsymbol{S}$ are exploited implicitly in [1] and more explicitly in [11] in the derivation of the Berlekamp–Massey algorithm. In [11], various syndrome matrices, being identified as Hankel matrices, are used to derive an inherent rule of how the length of the minimal LFSR, synthesized by an arbitrary recursive algorithm (including the Berlekamp–Massey algorithm), grows with the length of the syndrome sequence input to the algorithm. A necessary and sufficient condition of the uniqueness of the LFSR is derived there, and the early-stopped conditions in [2] and [3] for the Berlekamp–Massey algorithm are also revisited in [11].

In this paper, we will adopt the *right* type of column operations, a restricted Gaussian elimination as that in the FIA in [1], to perform on the *right* augmented syndrome matrix $\boldsymbol{S}$ in (3) with the exploitation of the structural properties of $\boldsymbol{S}$. With this new approach, we will develop an ESBM algorithm, as a re-interpretation of the early-stopped version of the Berlekamp–Massey algorithm in [2] and [3], in which only $(t+e)$ iterations are needed to be performed for the decoding of BCH codes up to t errors, where e is the number of errors actually occurred with $e\le t$, instead of the $2t$ iterations required in the conventional Berlekamp–Massey algorithm. Although an attempt to show the minimality of $(t+e)$ iterations in an ESBM algorithm is made in [11], we think it is vague and not successful. We will prove the minimality of $(t+e)$ iterations through our formulation and relate it to the subject of simultaneous error correction and detection. We will analyze the multiplicative complexity of the developed algorithm and show that except one trivial case, our developed ESBM algorithm is the most efficient algorithm for finding the error-locator polynomial.

II. Left-Column Replacement Operations

In this section, we will present a basic type of column operations performed in the fundamental iterative algorithm (FIA) in [1] for determining the linear dependency of a column on its previous (i.e., left) columns in a matrix. Related concepts and useful results will be further developed.

To determine the linear dependency of a column on its previous columns in a matrix $\boldsymbol{A}$, a restricted Gaussian elimination on columns of $\boldsymbol{A}$ can be performed as follows. For each nonzero column $\boldsymbol{a}_j$ of $\boldsymbol{A}$, we add a multiple of a previous column $\boldsymbol{a}_v$, $v<j$, of $\boldsymbol{A}$ from column $\boldsymbol{a}_j$ to eliminate the leading (i.e., topmost) nonzero entry of column $\boldsymbol{a}_j$ to lower down the position of the leading nonzero entry of the newly resulted column, if possible. More precisely, if the leading nonzero entry of the jth column $\boldsymbol{a}_j$ is the ith entry $a_{i,j}$ and there is a previous column $\boldsymbol{a}_v$, $v<j$, which has a leading nonzero entry $a_{i,v}$ also at the ith position, then we replace column $\boldsymbol{a}_j$ by $(\boldsymbol{a}_j-(a_{i,j}/a_{i,v})\boldsymbol{a}_v)$. The replaced column $(\boldsymbol{a}_j-(a_{i,j}/a_{i,v})\boldsymbol{a}_v)$ is either a zero vector or has a leading nonzero entry below the ith position. Such an elimination operation is called a *left-column replacement operation*. A matrix $\boldsymbol{A}'$ is called a left-reduced matrix of $\boldsymbol{A}$ if it is obtained by applying a series of left-column replacement operations on $\boldsymbol{A}$ and the leading nonzero entry of each column of $\boldsymbol{A}'$, if exists, cannot be further eliminated by any left-column replacement operation. The leading nonzero entry of a column in a left-reduced matrix $\boldsymbol{A}'$ is called a pivot. Pivots must be in different rows of $\boldsymbol{A}'$; otherwise, one of them can be further eliminated by a left-column replacement operation.

It is clear that each column $\boldsymbol{a}'_j$ of a left-reduced matrix $\boldsymbol{A}'$ is a linear combination of the first j columns $\boldsymbol{a}_v$, $1 \le v \le j$, of the original matrix $\boldsymbol{A}$ as

$$\sum_{v=1}^{j-1} \beta_{j-v} \boldsymbol{a}_v + \boldsymbol{a}_j = \boldsymbol{a}'_j \tag{5}$$

where β_v, $1 \le v \le j-1$, are constants and dependent on the column index j. If $\boldsymbol{a}'_j$ of the left-reduced matrix $\boldsymbol{A}'$ is a zero vector, then column $\boldsymbol{a}_j$ of the original matrix $\boldsymbol{A}$ is a linear combination of its previous columns in $\boldsymbol{A}$. Otherwise, column $\boldsymbol{a}'_j$ has a pivot at the ith coordinate for some i. Since $a'_{u,j} = 0 \; \forall 1 \le u \le i-1$, we have

$$a_{u,j} = -\sum_{v=1}^{j-1} \beta_{j-v} a_{u,v}, \quad 1 \le u \le i-1 \tag{6}$$

from (5) and column $\boldsymbol{a}_j$ of $\boldsymbol{A}$ is said to be a partial linear combination of its previous columns up to the $(i-1)$th entry. A further argument can render us the following lemma.

Lemma 3: A left-reduced matrix $\boldsymbol{A}'$ of a matrix $\boldsymbol{A}$ has a pivot at the (i,j)th position if and only if the jth column of $\boldsymbol{A}$ is a partial linear combination of its previous columns in $\boldsymbol{A}$ up to the $(i-1)$th entry, but not to the ith entry.

Proof: Please see Appendix I. □

Thus, all left-reduced matrices of a matrix $\boldsymbol{A}$ have the same pivot positions and we shall say that the matrix $\boldsymbol{A}$ has a pivot position at the (i,j)th entry if a left-reduced matrix of $\boldsymbol{A}$ has a pivot at that position. A consequence of Lemma 3 is that matrix $\boldsymbol{A}$ has a pivot position in column $\boldsymbol{a}_j$ if and only if column $\boldsymbol{a}_j$ is linearly independent of its previous columns in $\boldsymbol{A}$.

Recall that a square matrix $\boldsymbol{A}$ is symmetric if $a_{i,j} = a_{j,i}$ for all i, j. The following lemma is useful later.

Lemma 4: A square symmetric matrix has a pivot position at the (i,j)th entry if and only if it has a pivot position at the transposed (j,i)th entry.

Proof: Please see Appendix II. □

III. Structural Properties of the Augmented Syndrome Matrix $\boldsymbol{S}$

In this section, we assume that the number e of errors actually occurred is no greater than t. By Corollary 2, the last (i.e., the $(t+1)$th) column $\boldsymbol{\xi}_{t+1}$ of the augmented syndrome matrix $\boldsymbol{S}$ in (3) is a linear combination of its previous columns, and by Lemma 3, all pivot positions of $\boldsymbol{S}$ are located in the principal submatrix $\boldsymbol{S}_{t\times t}$ which is square and symmetric. The next corollary follows from Lemma 4.

Corollary 5: If the number e of errors actually occurred is no greater than t, then the augmented syndrome matrix $\boldsymbol{S}$ has a pivot position at the (i,j)th entry if and only if it has a pivot position at the transposed (j,i)th entry. □

An $m \times n$ matrix $\boldsymbol{A}$ is said to be in a Hankel form if

$$a_{i,j+1} = a_{i+1,j}, \quad \forall 1 \le i \le m-1, 1 \le j \le n-1. \tag{7}$$

The augmented syndrome matrix $\boldsymbol{S}$ in (3) is in a Hankel form since the (i,j)th entry $s_{i,j}$ of $\boldsymbol{S}$ is S_{i+j-1} for all $1 \le i \le t$, $1 \le j \le t+1$. The slanted diagonal in $\boldsymbol{S}$ where all S_r's reside, as shown in Fig. 1 for $r = t, t+1$, is called the S_r-slanted diagonal. While for each r, $t+1 \le r \le 2t$, the S_r-slanted diagonal does not cross the first $r-t$ columns of $\boldsymbol{S}$, we shall still regard every entry in these $r-t$ columns as being above the S_r-slanted diagonal.

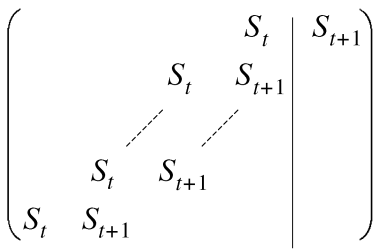

Fig. 1. S_t-slanted diagonal and the S_{t+1}-slanted diagonal in the augmented syndrome matrix $\boldsymbol{S}$.

Lemma 6: Given an r, $1 \le r \le 2t$, if a column of $\boldsymbol{S}$ has a pivot position on or above the S_r-slanted diagonal, then every column of $\boldsymbol{S}$ to the left of that column also has a pivot position on or above the S_r-slanted diagonal.

Proof: Let the column having a pivot position on or above the S_r-slanted diagonal be the jth column $\boldsymbol{\xi}_j$. If $j = 1$, we are done. Suppose that $j > 1$ and that there is a column $\boldsymbol{\xi}_i$, $i < j$, which does not have any pivot position on or above the S_r-slanted diagonal. If $i \le (r-t)$, i.e., all entries of the column $\boldsymbol{\xi}_i$ are above the S_r-slanted diagonal, then the column $\boldsymbol{\xi}_i$ of the augmented syndrome matrix $\boldsymbol{S}$ is a linear combination of its previous columns by Lemma 3. Thus, by Corollary 2, we have $i \ge e+1$. Hence, since $j > i \ge e+1$, the column $\boldsymbol{\xi}_j$ is also a linear combination of its previous columns in $\boldsymbol{S}$ and then does not have any pivot position by Lemma 3, which is a contradiction. Thus, $i > (r-t)$ and the column $\boldsymbol{\xi}_i$ is a partial linear combination of its previous $i-1$ columns in $\boldsymbol{S}$ up to the $(r+1-i)$th entry S_r by Lemma 3. Then there exist coefficients β_v, $1 \le v \le i-1$, such that

$$\sum_{v=1}^{i-1} \beta_{i-v} s_{u,v} + s_{u,i} = 0, \quad \forall 1 \le u \le r+1-i. \tag{8}$$

Since $\boldsymbol{S}$ is in a Hankel form and by (7), we have $s_{u-(j-i),v+(j-i)} = s_{u,v}$ for all $j-i+1 \le u \le r+1-i$, $1 \le v \le i$. With $u' = u-(j-i)$ and from (8), we have

$$\sum_{v=1}^{i-1} \beta_{i-v} s_{u',j-(i-v)} + s_{u',j} = 0, \quad \forall 1 \le u' \le r+1-j$$

which means that the column $\boldsymbol{\xi}_j$ is a partial linear combination of its previous columns in $\boldsymbol{S}$ up to the $(r+1-j)$th entry S_r, which is on the S_r-slanted diagonal, a contradiction, too. This completes the proof. □

Now, for each r, $1 \le r \le 2t$, we define L_r to be the column index of the rightmost column in $\boldsymbol{S}$ which has a pivot position on or above the S_r-slanted diagonal. If there does not exist such a column, we define L_r to be 0. It is clear that the first L_r columns of the augmented syndrome matrix $\boldsymbol{S}$ are linearly independent and the (L_r+1)th column of $\boldsymbol{S}$ is a partial linear combination of its previous columns up to the $(r-L_r)$th entry S_r. Let r_f be the first r such that

$$r - L_r = t. \tag{9}$$

Then the $(L_{r_f}+1)$th column is the first column which is a linear combination of its previous columns in $\boldsymbol{S}$, and by Corollary 2, we have $L_{r_f} = e$ and then

$$r_f = t + L_{r_f} = t + e. \tag{10}$$

If $L_{r-1} = 0$, then $L_{r-1} \leq L_r$ trivially. Now suppose that $L_{r-1} > 0$. Since the column $\boldsymbol{\xi}_{L_{r-1}}$ of $\boldsymbol{S}$ has a pivot position on or above the S_{r-1}-slanted diagonal which is trivially above the S_r-slanted diagonal, we have

$$L_{r-1} \leq L_r, \quad \forall r. \tag{11}$$

Lemma 7: For an r, $1 \leq r \leq 2t$, with $L_r > 0$, the principal submatrix $\boldsymbol{S}_{L_r \times L_r}$ of $\boldsymbol{S}$ is nonsingular. In particular, there are L_r pivot positions of $\boldsymbol{S}$ within the submatrix $\boldsymbol{S}_{L_r \times L_r}$ which are in different columns and in different rows.

Proof: By Lemma 6, each column $\boldsymbol{\xi}_k$, $k \leq L_r$, of $\boldsymbol{S}$ has a pivot position on or above the S_r-slanted diagonal. Thus, the first L_r columns of $\boldsymbol{S}$ are linearly independent. Suppose that there is a column, says column $\boldsymbol{\xi}_j$, $j \leq L_r$, with pivot position located below the L_rth entry, says located at the ith entry. Then we have $L_r < i \leq \min(r+1-j, t)$ which implies $i+j-1 \leq r$. By the symmetry property of the pivot positions in $\boldsymbol{S}$, as stated in Corollary 5, the column $\boldsymbol{\xi}_i$ of $\boldsymbol{S}$ has a pivot position which is located at the jth entry, i.e., the (j,i)th position of $\boldsymbol{S}$. With $i+j-1 \leq r$, this pivot position (i.e., the jth entry) of column $\boldsymbol{\xi}_i$ is on or above the S_r-slanted diagonal, which contradicts to the definition of L_r. Thus, the pivot position in each of the first L_r (linearly independent) columns of $\boldsymbol{S}$ is located on or above the L_rth entry. Hence, there are L_r pivot positions of $\boldsymbol{S}$ located within the principal submatrix $\boldsymbol{S}_{L_r \times L_r}$ of $\boldsymbol{S}$, which are in different columns and in different rows. It is clear that this submatrix $\boldsymbol{S}_{L_r \times L_r}$ is nonsingular. □

Suppose that $L_{r-1} > 0$. Since the submatrix $\boldsymbol{S}_{L_{r-1} \times L_{r-1}}$ of $\boldsymbol{S}$ is nonsingular by Lemma 7, any vector of dimension L_{r-1} must be a linear combination of the columns in $\boldsymbol{S}_{L_{r-1} \times L_{r-1}}$. Thus, each column $\boldsymbol{\xi}_j$, $L_{r-1}+1 \leq j \leq t+1$, of $\boldsymbol{S}$ is a partial linear combination of its previous columns in $\boldsymbol{S}$ up to the L_{r-1}th entry $S_{j+L_{r-1}-1}$. The smallest j, $L_{r-1}+1 \leq j \leq t+1$, such that $r \leq j + L_{r-1} - 1$ (which means that the column $\boldsymbol{\xi}_j$ is a partial linear combination of its previous columns up to an entry on or below the S_r-slanted diagonal and then cannot have a pivot position on or above the S_r-slanted diagonal) is $\max(L_{r-1}+1, r - L_{r-1} + 1)$. This implies that

$$L_r \leq \max(L_{r-1}, r - L_{r-1}) \tag{12}$$

by Lemma 6. Thus, if $r \leq 2L_{r-1}$, i.e., $r - L_{r-1} \leq L_{r-1}$, then we have $L_r = L_{r-1}$ from (11) and (12). Now suppose that $r > 2L_{r-1}$, i.e., $r - L_{r-1} > L_{r-1}$. If $r - L_{r-1} > t$, then all entries of the column $\boldsymbol{\xi}_{L_{r-1}+1}$ are on or above the S_{r-1}-slanted diagonal and by the definition of L_{r-1}, the column $\boldsymbol{\xi}_{L_{r-1}+1}$ is a linear combination of its previous columns in $\boldsymbol{S}$. By Corollary 2, every column of $\boldsymbol{S}$ to the right of the column $\boldsymbol{\xi}_{L_{r-1}+1}$ is also a linear combination of its previous columns in $\boldsymbol{S}$. Thus, we have $L_r \leq L_{r-1}$ and then $L_r = L_{r-1}$ by (11). Now consider $r - L_{r-1} \leq t$. If the column $\boldsymbol{\xi}_{L_{r-1}+1}$ is a partial linear combination of its previous columns up to the $(r - L_{r-1})$th entry S_r, then it is clear that $L_r \leq L_{r-1}$ by Lemma 6 and then $L_r = L_{r-1}$ by (11). If not, the column $\boldsymbol{\xi}_{L_{r-1}+1}$ has a pivot position at the $(r - L_{r-1})$th entry S_r, and by Corollary 5, the column $\boldsymbol{\xi}_{r-L_{r-1}}$ has a pivot position at the $(L_{r-1}+1)$th entry S_r. Now, by Lemma 6, we have $L_r \geq r - L_{r-1}$ and then $L_r = r - L_{r-1}$ by (12). We summarize the results in the following theorem, where we let $L_0 = 0$ for convenience.

Theorem 8: For $r \leq 2L_{r-1}$ or $r > t + L_{r-1}$, we have $L_r = L_{r-1}$, and for $2L_{r-1} < r \leq t + L_{r-1}$, if the column $\boldsymbol{\xi}_{L_{r-1}+1}$ is a partial linear combination of its previous columns up to the $(r - L_{r-1})$th entry S_r, then $L_r = L_{r-1}$ and if not, $L_r = r - L_{r-1}$. □

From (11), the sequence $\{L_r, r = 1, \ldots, 2t\}$ is nondecreasing. The next corollary describes the pattern of pivot positions in the augmented syndrome matrix $\boldsymbol{S}$.

Collorary 9: If $L_r = L_{r-1}$, then there does not exist any pivot position of $\boldsymbol{S}$ on the S_r-slanted diagonal, and if $L_r > L_{r-1}$, then there are exactly $L_r - L_{r-1}$ pivot positions of $\boldsymbol{S}$ lying on the S_r-slanted diagonal, beginning at the $(L_r, L_{r-1}+1)$th entry and ending up at the $(L_{r-1}+1, L_r)$th entry.

Proof: By Lemma 6, each of the first L_r columns of $\boldsymbol{S}$ has a pivot position on or above the S_r-slanted diagonal and none of the rest $t+1-L_r$ columns have a pivot position on or above the S_r-slanted diagonal. Thus, if $L_r = L_{r-1}$, then there does not exist any pivot position of $\boldsymbol{S}$ on the S_r-slanted diagonal. Now for $L_r > L_{r-1}$, we have $L_{r-1} < r - L_{r-1} \leq t$ and $L_r = r - L_{r-1}$ by Theorem 8. Then the S_r-slanted diagonal crosses the column $\boldsymbol{\xi}_{L_{r-1}+1}$ at the $(r - L_{r-1}, L_{r-1}+1)$th entry S_r up to the column $\boldsymbol{\xi}_{L_r}$ at the $(r - L_r + 1, L_r)$th entry S_r. Thus, there are exactly $(L_r - L_{r-1})$ pivot positions of $\boldsymbol{S}$ lying on the S_r-slanted diagonal, beginning at the $(L_r, L_{r-1}+1)$th entry and ending up at the $(L_{r-1}+1, L_r)$th entry. This completes the proof. □

IV. Early-Stopped Berlekamp–Massey (ESBM) Algorithm

Suppose that there are J jumps in the nondecreasing sequence $\{L_r, r = 1, \ldots, 2t\}$ at indices r_j, $1 \leq j \leq J$, i.e., $L_r > L_{r-1}$ if $r = r_j$ and $L_r = L_{r-1}$ otherwise, where we have defined $L_0 = 0$ for convenience. From Theorem 8, we have

$$r_j = L_{r_j} + L_{r_{j-1}}, \quad \forall j \in [1, J]. \tag{13}$$

Let $n_j = L_{r_j} - L_{r_{j-1}}$ be the size of the jump at $r = r_j$. By Corollary 9, n_j is the number of (consecutive) pivot positions of $\boldsymbol{S}$ lying on the S_{r_j}-slanted diagonal and there is no pivot position of $\boldsymbol{S}$ lying on the S_r-slanted diagonal if r is not a jump index. Then we have

$$L_{r_j} = \sum_{k=1}^{j} n_k, \quad \forall 1 \leq j \leq J \tag{14}$$

and, in particular, by assuming that the number e of errors actually occurred is no greater than t, the total number of pivot positions in $\boldsymbol{S}$ is e from Corollary 2, and we have

$$L_{r_J} = n_1 + n_2 + \cdots + n_J = e. \tag{15}$$

Since the principal submatrix $\boldsymbol{S}_{L_{r_j} \times L_{r_j}}$ of $\boldsymbol{S}$ for each j, $1 \leq j \leq J$, is nonsingular by Lemma 7, the $(L_{r_j}+1)$th column of

$\boldsymbol{S}$ is a linear combination of its previous L_{r_j} columns in $\boldsymbol{S}$ up to the L_{r_j}th entry $S_{2L_{r_j}}$, i.e.,

$$\sum_{v=1}^{L_{r_j}} \sigma_{L_{r_j}+1,L_{r_j}+1-v}^{(L_{r_j})} \cdot s_{u,v} + s_{u,L_{r_j}+1} = 0, \quad \forall 1 \le u \le L_{r_j} \tag{16}$$

where the linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})} = (\sigma_{L_{r_j}+1,L_{r_j}}^{(L_{r_j})}, \ldots, \sigma_{L_{r_j}+1,1}^{(L_{r_j})}, 1)$ must be unique (We will use a j-tuple, denoted by $\boldsymbol{\Lambda}_j^{(i)}$, with the rightmost component to be 1 to represent the coefficients of a linear combination with which the jth column is a partial linear combination of its previous columns up to the ith component as in above. In general, $\boldsymbol{\Lambda}_j^{(i)}$ is not necessarily unique). We next describe an iterative procedure to find r_j, L_{r_j}, and $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})}$ from r_{j-1}, $L_{r_{j-1}}$, and $\boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}$.

To find the next jump index r_j, and then L_{r_j}, is equivalent to find the r, beginning at $r = 2L_{r_{j-1}} + 1$ and ending at $r = t + L_{r_{j-1}}$, such that the $(L_{r_{j-1}} + 1)$th column $\boldsymbol{\xi}_{L_{r_{j-1}}+1}$ of $\boldsymbol{S}$ is not a partial linear combination of its previous columns up to the $(r - L_{r_{j-1}})$th entry S_r by Theorem 8, i.e., to find the first r in $[2L_{r_{j-1}} + 1, t + L_{r_{j-1}}]$ such that the discrepancy

$$\begin{aligned} d_r &= \sum_{v=1}^{L_{r_{j-1}}} \sigma_{L_{r_{j-1}}+1,L_{r_{j-1}}+1-v}^{(L_{r_{j-1}})} \cdot s_{r-L_{r_{j-1}},v} \\ &\quad + s_{r-L_{r_{j-1}},L_{r_{j-1}}+1} \\ &= \sum_{v=1}^{L_{r_{j-1}}} \sigma_{L_{r_{j-1}}+1,L_{r_{j-1}}+1-v}^{(L_{r_{j-1}})} \cdot S_{r-L_{r_{j-1}}-1+v} + S_r \end{aligned} \tag{17}$$

is nonzero by Lemma 3. This is called the search phase for r_j and then for L_{r_j}.

If all the discrepancies d_r, $2L_{r_{j-1}} + 1 \le r \le t + L_{r_{j-1}}$, are zeros, then the $(L_{r_{j-1}} + 1)$th column $\boldsymbol{\xi}_{L_{r_{j-1}}+1}$ is the first column of $\boldsymbol{S}$ which is a linear combination of its previous columns. By Corollary 2, the linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}$ is equal to the coefficient vector $\boldsymbol{\Lambda} = (\sigma_e, \ldots, \sigma_1, 1)$ of the error-locator polynomial $\Lambda(x)$ and no more pivot positions of $\boldsymbol{S}$ can be found. In this case, the final index $r = L_{r_{j-1}} + t$ has $d_r = 0$ and $L_r = L_{r_{j-1}}$ such that this final index r is the first index with $d_r = 0$ and $r - L_r = t$. By (9), this final index r is equal to r_f. Thus, no more jump index can be found, and we have $J = j - 1$. Otherwise, we find r_j as the first r in $[2L_{r_{j-1}} + 1, t + L_{r_{j-1}}]$ with $d_r \neq 0$. This completes the phase of search for r_j, and then we have

$$L_{r_j} = L_{r_{j-1}} + (r_j - 2L_{r_{j-1}}) = r_j - L_{r_{j-1}} \tag{18}$$

as stated in Theorem 8.

Now, to determine the linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})}$ with which the $(L_{r_j} + 1)$th column $\boldsymbol{\xi}_{L_{r_j}+1}$ of $\boldsymbol{S}$ is a unique linear combination of its previous L_{r_j} columns in $\boldsymbol{S}$ up to the L_{r_j}th entry $S_{2L_{r_j}}$, we will apply left-column replacement operations on the augmented syndrome matrix $\boldsymbol{S}$. This is called the update phase for $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})}$.

At first, we note that the $(L_{r_{j-1}}+1)$th column of $\boldsymbol{S}$ is a linear combination of its previous columns in $\boldsymbol{S}$ up to the $(r_j - L_{r_{j-1}} - 1)$th entry S_{r_j-1}, i.e.,

$$-\sum_{v=1}^{L_{r_{j-1}}} \sigma_{L_{r_{j-1}}+1,L_{r_{j-1}}+1-v}^{(L_{r_{j-1}})} \cdot s_{u,v} = s_{u,L_{r_{j-1}}+1}$$

for all u, $1 \le u \le r_j - L_{r_{j-1}} - 1$, since the discrepancy d_r in (17) is zero from $r = 2L_{r_{j-1}} + 1$ to $r = r_j - 1$. Now since $\boldsymbol{S}$ is in a Hankel form, we have

$$-\sum_{v=1}^{L_{r_{j-1}}} \sigma_{L_{r_{j-1}}+1,L_{r_{j-1}}+1-v}^{(L_{r_{j-1}})} \cdot s_{u-n_j,n_j+v} = s_{u-n_j,L_{r_j}+1}$$

for all u, $n_j + 1 \le u \le r_j - L_{r_{j-1}} - 1$, i.e.,

$$-\sum_{v=1}^{L_{r_{j-1}}} \sigma_{L_{r_{j-1}}+1,L_{r_{j-1}}+1-v}^{(L_{r_{j-1}})} \cdot s_{u',n_j+v} = s_{u',L_{r_j}+1}$$

for all u', $1 \le u' \le r_j - L_{r_j} - 1$, which says that the $(L_{r_j}+1)$th column of $\boldsymbol{S}$ is a linear combination of its previous columns in $\boldsymbol{S}$ up to the $(r_j - L_{r_j} - 1)$th entry S_{r_j-1} with linear combination coefficient vector

$$\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r_j-L_{r_j}-1)} = \left(\mathbf{0}^{n_j}, \boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}\right). \tag{19}$$

Since

$$\begin{aligned} &\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r_j-L_{r_j}-1)} \cdot \left(s_{r_j-L_{r_j},1}, \ldots, s_{r_j-L_{r_j},L_{r_j}+1}\right) \\ &\quad = \boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})} \cdot \left(S_{r_j-L_{r_{j-1}}}, \ldots, S_{r_j}\right) \\ &\quad = d_{r_j} \neq 0 \end{aligned}$$

from (17), and there is a pivot position at the $(L_{r_{j-1}}, L_{r_{j-2}} + 1)$th entry (i.e., the $(r_j - L_{r_j}, L_{r_{j-2}} + 1)$th entry since $r_j = L_{r_{j-1}} + L_{r_j}$ by (18)) of $\boldsymbol{S}$, i.e.,

$$\begin{aligned} &\boldsymbol{\Lambda}_{L_{r_{j-2}}+1}^{(L_{r_{j-2}})} \cdot \left(s_{L_{r_{j-1}},1}, \ldots, s_{L_{r_{j-1}},L_{r_{j-2}}+1}\right) \\ &\quad = \boldsymbol{\Lambda}_{L_{r_{j-2}}+1}^{(L_{r_{j-2}})} \cdot \left(S_{L_{r_{j-1}}}, \ldots, S_{r_{j-1}}\right) \\ &\quad = d_{r_{j-1}} \neq 0, \end{aligned}$$

we have

$$\begin{aligned} &\left\{\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r_j-L_{r_j}-1)} - \frac{d_{r_j}}{d_{r_{j-1}}} \left(\boldsymbol{\Lambda}_{L_{r_{j-2}}+1}^{(L_{r_{j-2}})}, \mathbf{0}^{n_{j-1}+n_j}\right)\right\} \\ &\quad \cdot \left(s_{r_j-L_{r_j},1}, \ldots, s_{r_j-L_{r_j},L_{r_j}+1}\right) \\ &\quad = \left\{\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r_j-L_{r_j}-1)} - \frac{d_{r_j}}{d_{r_{j-1}}} \left(\boldsymbol{\Lambda}_{L_{r_{j-2}}+1}^{(L_{r_{j-2}})}, \mathbf{0}^{n_{j-1}+n_j}\right)\right\} \\ &\qquad \cdot \left(S_{r_j-L_{r_j}}, \ldots, S_{r_j}\right) \\ &\quad = d_{r_j} - \frac{d_{r_j}}{d_{r_{j-1}}} d_{r_{j-1}} = 0. \end{aligned}$$

This implies that the $(L_{r_j}+1)$th column $\boldsymbol{\xi}_{L_{r_j}+1}$ of $\boldsymbol{S}$ is a partial linear combination of its previous columns in $\boldsymbol{S}$ up to the $(r_j -$

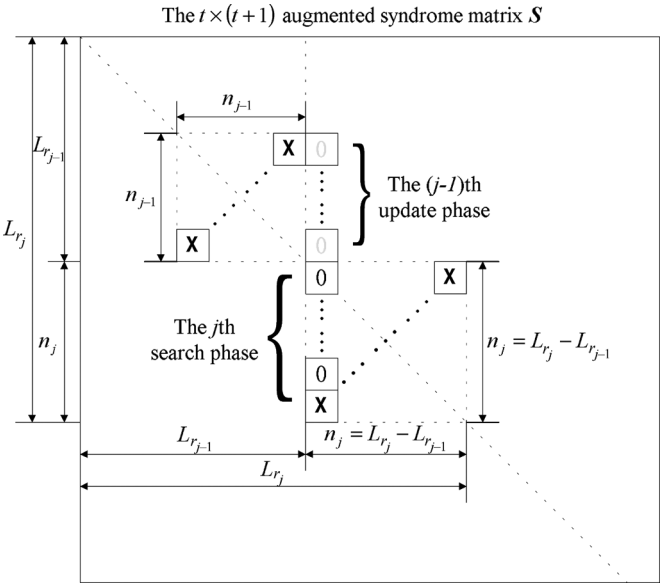

Fig. 2. Alternative search and update phases in the iteration process of determining r_j, L_{r_j} and $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})}$ for $j = 1, 2, \ldots, J$, where the pivot positions are marked by "×."

L_{r_j})th entry S_{r_j} with the linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r_j-L_{r_j})}$ by updating the vector $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r_j-L_{r_j}-1)}$ to

$$\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r_j-L_{r_j})} = \left(\mathbf{0}^{n_j}, \boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}\right) - \frac{d_{r_j}}{d_{r_{j-1}}}\left(\boldsymbol{\Lambda}_{L_{r_{j-2}}+1}^{(L_{r_{j-2}})}, \mathbf{0}^{n_{j-1}+n_j}\right) \quad (20)$$

by (19), which is, indeed, the result via a series of left-column replacement operations on $\boldsymbol{S}$ to eliminate all nonzero entries at or above the $(r_j - L_{r_j})$th position of the $(L_{r_j}+1)$th column $\boldsymbol{\xi}_{L_{r_j}+1}$ of $\boldsymbol{S}$.

Now for each r, $r_j + 1 \le r \le r_j + n_j = 2L_{r_j}$, we continue to eliminate the $(r - L_{r_j})$th entry S_r of the $(L_{r_j}+1)$th column by employing the pivot at the $(r - L_{r_j}, L_{r_j} + 1 - (r - r_j))$th position, lying on the S_{r_j}-slanted diagonal as stated in Corollary 9, and by updating the vector $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r-L_{r_j}-1)}$ to

$$\boldsymbol{\Lambda}_{L_{r_j}+1}^{(r-L_{r_j})} = \boldsymbol{\Lambda}_{L_{r_j}+1}^{(r-L_{r_j}-1)} - \frac{d_r}{d_{r_j}}\left(\mathbf{0}^{n_j-(r-r_j)}, \boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}, \mathbf{0}^{r-r_j}\right) \quad (21)$$

where

$$\begin{aligned} d_r &= \boldsymbol{\Lambda}_{L_{r_j}+1}^{(r-1-L_{r_j})} \cdot \left(s_{r-L_{r_j}}, 1, \ldots, s_{r-L_{r_j}, L_{r_j}+1}\right) \\ &= \sum_{v=1}^{L_{r_j}} \sigma_{L_{r_j}+1, L_{r_j}+1-v}^{(r-1-L_{r_j})} \cdot S_{r-L_{r_j}-1+v} + S_r. \end{aligned} \quad (22)$$

After completing the phase of updating coefficient vector at $r = 2L_{r_j}$, we obtain the desired linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})}$, which is unique.

We now summarize the iterative procedure to find r_j, L_{r_j} and $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})}$ as shown in Fig. 2. In general, there are two phases in each jth iteration. The first phase is a search phase for r_j in the interval $[2L_{r_{j-1}} + 1, t + L_{r_{j-1}}]$ of the index r by detecting a nonzero discrepancy d_r in (17) by using the most recently obtained linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}$. If none of the discrepancies d_r with $r \in [2L_{r_{j-1}} + 1, t + L_{r_{j-1}}]$ is detected to be nonzero, then the number e of errors actually occurred is $L_{r_{j-1}}$ and the coefficient vector $\boldsymbol{\Lambda}$ of the error-locator polynomial $\Lambda(x)$ is $\boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}$. Then the algorithm stops. Otherwise, r_j is found and $L_{r_j} = r_j - L_{r_{j-1}}$. The end of the first phase triggers immediately the begin of the second phase. The second phase is an update phase for obtaining the next linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r_j}+1}^{(L_{r_j})}$ in the interval $[r_j, 2L_{r_j}]$ of the index r. We use (20) to update the coefficient vector when $r = r_j$ and use (21) when $r \in [r_j + 1, 2L_{r_j}]$.

There are totally $J+1$ iterations and the last, i.e., the $(J+1)$th, iteration consists of only a search phase. For each j, $1 \le j \le J$, the jth iteration consists of $2(L_{r_j} - L_{r_{j-1}}) = 2n_j$ steps for the index r from $2L_{r_{j-1}} + 1$ to $2L_{r_j}$. Among the $2n_j$ steps, the first n_j steps for the index r from $2L_{r_{j-1}} + 1$ to $L_{r_{j-1}} + L_{r_j}$ is in the jth search phase and the last $n_j + 1$ steps for the index r from $L_{r_{j-1}} + L_{r_j}$ to $2L_{r_j}$ is in the jth update phase. Note that the jth search phase and the jth update phase overlap at the n_jth step of the jth iteration for the index $r = r_j = L_{r_{j-1}} + L_{r_j}$, where the jth jump of the sequence $\{L_r, r = 1, \ldots, 2t\}$ occurs. In each step of the jth iteration, we should at first calculate the discrepancy d_r by (17) or (22), depending on the index r being in the search phase or in the update phase. Since

$$L_r = \begin{cases} L_{r_{j-1}}, & r \in \left[2L_{r_{j-1}} + 1, L_{r_{j-1}} + L_{r_j} - 1\right] \\ L_{r_j}, & r \in \left[L_{r_{j-1}} + L_{r_j}, 2L_{r_j}\right] \end{cases} \quad (23)$$

we have

$$\boldsymbol{\Lambda}_{L_r+1}^{(r-L_r)} = \begin{cases} \boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}, & r \in \left[2L_{r_{j-1}} + 1, L_{r_{j-1}} + L_{r_j} - 1\right] \\ \boldsymbol{\Lambda}_{L_{r_j}+1}^{(r-L_{r_j})}, & r \in \left[L_{r_{j-1}} + L_{r_j}, 2L_{r_j}\right] \end{cases} \quad (24)$$

by (17), (20), and (21). Thus, we can rewrite the calculation of the rth discrepancy d_r as

$$d_r = \boldsymbol{\Lambda}_{L_{r-1}+1}^{(r-1-L_{r-1})} \cdot \left(S_{r-L_{r-1}}, \ldots, S_r\right) \quad (25)$$

by (17), (22), and (24). Since the discrepancy d_r is zero in the search phase except at the last step, i.e., the r_jth step, there is no need to update the linear combination coefficient vector. A similar case occurs if the discrepancy d_r is zero in the update phase. However, if the discrepancy d_r is nonzero in the update phase except at the first step, i.e., the r_jth step, we will update the linear combination coefficient vector as

$$\boldsymbol{\Lambda}_{L_r+1}^{(r-L_r)} = \boldsymbol{\Lambda}_{L_{r-1}+1}^{(r-1-L_{r-1})} - d_r D_{r-1}^{-1}(\tilde{\boldsymbol{\Lambda}}_{r-1}, 0)|_{L_{r-1}+1} \quad (26)$$

for $r \in [r_j + 1, 2L_{r_j}]$, where we have used two auxiliary variables

$$D_{r-1} = d_{r_j} \quad (27)$$

$$\tilde{\boldsymbol{\Lambda}}_{r-1} = \left(\mathbf{0}^{n_j}, \boldsymbol{\Lambda}_{L_{r_{j-1}}+1}^{(L_{r_{j-1}})}, \mathbf{0}^{r-1-r_j}\right) \quad (28)$$

by (21) and (24) and the notation $\boldsymbol{v}\,|_\ell$ for a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ in (26) with $\ell \le n$, is defined to be the right ℓ-truncate of the vector $\boldsymbol{v}$, i.e., $\boldsymbol{v}\,|_\ell = (v_{n-\ell+1}, v_{n-\ell+2}, \ldots, v_n)$.

Now, when $r = r_j$, the discrepancy d_r is nonzero and the linear combination coefficient vector will be updated as

$$\mathbf{\Lambda}_{L_r+1}^{(r-L_r)} = \left(\mathbf{0}^{n_j}, \mathbf{\Lambda}_{L_{r-1}+1}^{(r-1-L_{r-1})}\right) - d_r D_{r-1}^{-1}(\tilde{\mathbf{\Lambda}}_{r-1}, 0)|_{r-L_{r-1}+1} \tag{29}$$

where

$$D_{r-1} = d_{r_{j-1}} \tag{30}$$

$$\tilde{\mathbf{\Lambda}}_{r-1} = \left(\mathbf{0}^{n_{j-1}}, \mathbf{\Lambda}_{L_{r_{j-2}}+1}^{(L_{r_{j-2}})}, \mathbf{0}^{n_{j-1}+n_j-1}\right) \tag{31}$$

by (20) and (24). From (27) and (30), we have

$$D_r = \begin{cases} D_{r-1}, & r \in \left[2L_{r_{j-1}}+1, 2L_{r_j}\right] \setminus \{r_j\} \\ d_r, & r = r_j \end{cases} \tag{32}$$

and from (28) and (31), we have

$$\tilde{\mathbf{\Lambda}}_r = \begin{cases} (\tilde{\mathbf{\Lambda}}_{r-1}, 0), & r \in \left[2L_{r_{j-1}}+1, 2L_{r_j}\right] \setminus \{r_j\} \\ \left(\mathbf{0}^{n_j}, \mathbf{\Lambda}_{L_{r-1}+1}^{(r-1-L_{r-1})}\right), & r = r_j. \end{cases} \tag{33}$$

With (23), (26), (29), (32), and (33), we now write a pseudocode for the procedure in above, based on the index r of the syndromes S_r, $r = 1, 2, \ldots, 2t$. In the pseudocode, the following variables are used.

L	To store the index L_r of the rightmost column in $\boldsymbol{S}$ which has a pivot position on or above the S_r-slanted diagonal.
$\mathbf{\Lambda}$	To store the coefficient vector $\mathbf{\Lambda}_{L_r+1}^{(r-L_r)}$ with which the (L_r+1)th column of $\boldsymbol{S}$ is a partial linear combination of its previous columns up to the $(r-L_r)$th component S_r.
D	To store the pivot value D_r to be used to update the coefficient vector to eliminate the $(r-L_r, L_r+1)$th entry S_r.
$\tilde{\mathbf{\Lambda}}$	To store the linear combination coefficient vector $\tilde{\mathbf{\Lambda}}_r$ associated with the pivot D_r in above to be used to update the coefficient vector to eliminate the $(r-L_r, L_r+1)$th entry S_r.
$\mathbf{\Lambda}_{\text{temp}}$	To store the linear combination coefficient vector $\mathbf{\Lambda}_{L_r+1}^{(r-L_r)}$ temporarily for later use.

Since the algorithm begins in searching for a nonzero entry (i.e., a pivot position) in the first column of the augmented syndrome matrix $\boldsymbol{S}$, the initial values of the first four variables are set to be

$$L = 0, \mathbf{\Lambda} = 1, D = 1, \text{ and } \tilde{\mathbf{\Lambda}} = 0. \tag{34}$$

For r from 1 to $2t$

Calculate d_r of S_r by $d_r = \mathbf{\Lambda} \cdot (S_{r-L}, \ldots, S_r)$;

If $d_r = 0$ and $r - L = t$

/*at the end of the last search phase*/

$e = L$;

the error − locator polynomial $= \mathbf{\Lambda}$;

the algorithm stops.

Else if $d_r = 0$

/*either in a search phase or in an update phase*/

$\tilde{\mathbf{\Lambda}} \leftarrow (\tilde{\mathbf{\Lambda}}, 0)$ [by (33)].

Else if $d_r \neq 0$ and $r \leq 2L$

/*in an update phase but not in a search phase*/

$\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} - d_r D^{-1}(\tilde{\mathbf{\Lambda}}, 0)|_{L+1}$ [by (26)];

$\tilde{\mathbf{\Lambda}} \leftarrow (\tilde{\mathbf{\Lambda}}, 0)$ [by (33)].

Else if $d_r \neq 0$ and $r > 2L$

/*at the overlap of a search & an update phases*/

$\mathbf{\Lambda}_{\text{temp}} = \mathbf{\Lambda}$;

$\mathbf{\Lambda} \leftarrow (\mathbf{0}^{r-2L}, \mathbf{\Lambda}) - d_r D^{-1}(\tilde{\mathbf{\Lambda}}, 0)|_{r-L+1}$ [by (29)];

$\tilde{\mathbf{\Lambda}} \leftarrow (\mathbf{0}^{r-2L}, \mathbf{\Lambda}_{\text{temp}})$ [by (33)];

$D \leftarrow d_r$ [by (32)];

$L \leftarrow r - L$ [by (23)].

The algorithm in above is equivalent to the modified Berlekamp–Massey algorithm in [2], which is the same as the conventional Berlekamp–Massey algorithm in [7] and [8] except for the detection of an early stop by the conditions $d_r = 0$ and $r - L = t$. However, the distinction between the algorithm developed in above and the modified Berlekamp–Massey algorithm in [2] stems from the distinction between the refined FIA in [1] and the conventional Berlekamp–Massey algorithm in [7] and [8]. Moreover, while Hankel properties of various syndrome matrices are exploited in [11], the manipulation of syndrome matrices in [11] remains the style of the conventional Berlekamp–Massey algorithm. While the conventional Berlekamp–Massey algorithm is shown to be equivalent to a refinement of FIA in [1], the refined FIA has lower complexity than the conventional Berlekamp–Massey algorithm as can be seen in Section VI. Thus, the developed algorithm in above has lower complexity than the modified Berlekamp–Massey algorithm in [2]. This algorithm will be referred as the ESBM algorithm. From (9) and (10), the algorithm stops at $r = r_f = t + e$. Thus, only $t + e$ iterations are needed to be performed in the ESBM algorithm for the decoding of BCH codes up to t errors, where e is the number of errors actually occurred with $e \leq t$, instead of the constant $2t$ iterations required in the conventional Berlekamp–Massey algorithm. Note that in the completion of the ESBM algorithm, $\mathbf{\Lambda}$ stores the linear combination coefficient vector $\mathbf{\Lambda}_{L_{r_f}+1}^{(r_f-L_{r_f})} = \mathbf{\Lambda}_{e+1}^{(t)} = (\sigma_e^{(t)}, \ldots, \sigma_1^{(t)}, 1)$ and the error-locator polynomial is

$$\Lambda(x) = 1 + \sigma_1^{(t)} x + \cdots + \sigma_e^{(t)} x^e.$$

A scenario of the execution of the ESBM algorithm is illustrated in Fig. 3, where there are $e = 10$ errors actually occurred in the decoding of a codeword from a BCH code with error-correcting capability $t = 16$. In Fig. 3, the progression of

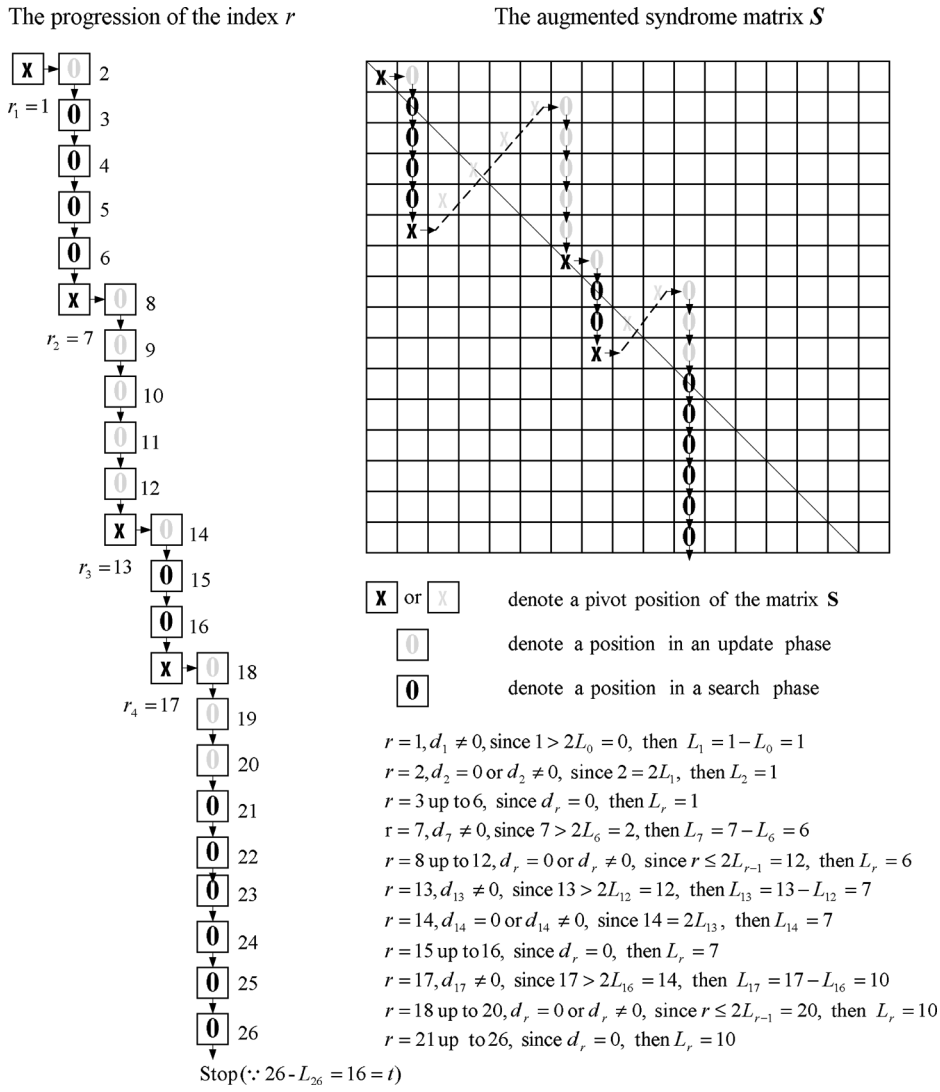

Fig. 3. Illustrative scenario of applying the ESBM algorithm to the decoding of a BCH code with designed error-correcting capability $t = 16$ for the case of $e = 10$ errors occurred.

the index r of the syndrome sequence $\{S_r, r = 1, 2, \ldots, 2t\}$ is described in an alternative search and update phases manner and all the pivot positions of the augmented syndrome matrix $\boldsymbol{S}$ are marked by "×." Suppose further that there are four jumps, i.e., $J = 4$, occurred at $r_1 = 1$, $r_2 = 7$, $r_3 = 13$, and $r_4 = 17$, respectively. Then $L_{r_1} = r_1 - 0 = 1$, $L_{r_2} = r_2 - L_{r_1} = 6$, $L_{r_3} = r_3 - L_{r_2} = 7$, and $L_{r_4} = r_4 - L_{r_3} = 10$, and $n_1 = 1$, $n_2 = 5$, $n_3 = 1$, $n_4 = 3$.

The first search phase begins at the step of $r = 1$ to detect a nonzero discrepancy d_r up to the step of $r = t$ in the first column of $\boldsymbol{S}$. Since $d_1 \neq 0$, i.e., $S_1 \neq 0$, the first search phase ends at the step of $r = 1$ and then $r_1 = 1$, $L_{r_1} = 1$. The first update phase is triggered immediately but without doing anything at the step of $r = 1$, as by setting the initial value of $\tilde{\boldsymbol{\Lambda}}$ to be 0, since the S_1-slanted diagonal is above the second column of $\boldsymbol{S}$, i.e., the second column does not have S_1 as an entry. The first update phase ends at the step of $r = 2L_1 = 2$ with $L_2 = 1$, where the execution of an update of the linear combination coefficient vector depends on whether the discrepancy d_2 is zero or not.

The second search phase begins at the step of $r = 2L_{r_1} + 1 = 3$ to detect a nonzero discrepancy d_r up to the step of $r = t + L_{r_1} = 17$ in the $(L_{r_1} + 1)$th column $\boldsymbol{\xi}_2$ of $\boldsymbol{S}$. Since $d_3 = \cdots = d_6 = 0$ and $d_7 \neq 0$, the second search phase ends at the step of $r = 7$ and then $r_2 = 7$, $L_{r_2} = 6$. The second update phase is triggered immediately for updating the linear combination coefficient vector to eliminate the nonzero leading entries of the $(L_{r_2} + 1)$th column $\boldsymbol{\xi}_7$ of $\boldsymbol{S}$, when the discrepancy d_r is nonzero, up to the L_{r_2}th entry $S_{2L_{r_2}}$ for the step index r from $r_2 (= 7)$ to $2L_{r_2} (= 12)$.

Similar operations are done in the third and the fourth iterations, which are from the step of $r = 13$ to the step of $r = 14$ and from the step of $r = 15$ to the step of $r = 20$ respectively. We have $r_3 = 13$, $L_{r_3} = 7$ and $r_4 = 17$, $L_{r_4} = 10$, and the fifth search phase begins at the step of $r = 2L_{r_4} + 1 = 21$ to detect a nonzero discrepancy d_r up to the step of $r = t + L_{r_4} = 26$ in the $(L_{r_4} + 1)$th column $\boldsymbol{\xi}_{11}$ of $\boldsymbol{S}$. Since $d_{21} = \cdots = d_{26} = 0$, we detect an early stop of the algorithm such that the number e of errors actually occurred is $L_{r_4} = 10$.

It is now clear that the algorithm stops at the step of $r = r_f = t + e = 26$ which is really six steps earlier than the conventional Berlekamp–Massey algorithm which stops at the step of $r = 2t = 32$.

V. Minimality of $(t + e)$ Iterations

In this section, we are ready to show the minimality of $(t + e)$ iterations in the ESBM algorithm, which has been an open problem presented in the Conclusion of [2]. We also draw a wider picture of the minimality of $(t + e)$ iterations within the context of simultaneous error correction and detection.

As suggested in [2], the answer to this question is related to the $t' \times (t'+1)$ augmented syndrome matrices for $t', 1 \le t' \le t$

$$\boldsymbol{S}_{t'\times(t'+1)} = \left(\begin{array}{cccc|c} S_1 & S_2 & \cdots & S_{t'} & S_{t'+1} \\ S_2 & S_3 & \cdots & S_{t'+1} & S_{t'+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{t'} & S_{t'+1} & \cdots & S_{2t'-1} & S_{2t'} \end{array}\right). \tag{35}$$

Note that each $\boldsymbol{S}_{t'\times(t'+1)}$ is an upper-left submatrix of the maximal augmented syndrome matrix $\boldsymbol{S} = \boldsymbol{S}_{t\times(t+1)}$. We first consider the case that the number e of errors actually occurred is no greater than t'. Then all the pivot positions of $\boldsymbol{S}$ must be within the principal submatrix $\boldsymbol{S}_{e\times e}$ of $\boldsymbol{S}$, which is also a principal submatrix of the matrix $\boldsymbol{S}_{t'\times(t'+1)}$, and the $(e+1)$th column of $\boldsymbol{S}$ is the first column of $\boldsymbol{S}$ which is a linear combination of its previous columns in $\boldsymbol{S}$ by Corollary 2. This implies that the $(e+1)$th column of $\boldsymbol{S}_{t'\times(t'+1)}$ is the first column of $\boldsymbol{S}_{t'\times(t'+1)}$ which is a linear combination of its previous columns in $\boldsymbol{S}_{t'\times(t'+1)}$ with the same linear combination coefficients in above, and from Corollary 2, the obtained linear combination coefficients are just the coefficients of the error-locator polynomial. Thus, the matrix $\boldsymbol{S}_{t'\times(t'+1)}$ has the same structural properties as those of the matrix $\boldsymbol{S}$ in Section III if $e \le t'$, and the Berlekamp–Massey algorithm can be early stopped by the detection conditions of $d_r = 0$ and $r - L_r = t'$ as developed in Section IV. The final iteration index r_f is $t' + e$, i.e., only $t' + e$ iterations are needed.

On the other hand, we consider the case that the number e of errors actually occurred is greater than t' but no greater than t with $t' < t$. Since the matrix $\boldsymbol{S}_{t'\times(t'+1)}$ has one more columns than rows, there exists at least one column of $\boldsymbol{S}_{t'\times(t'+1)}$ which has no pivot positions, and since the matrix $\boldsymbol{S}_{t'\times(t'+1)}$ is an upper-left submatrix of the matrix $\boldsymbol{S}$, the execution of the Berlekamp–Massey algorithm on the matrix $\boldsymbol{S}$ is exactly the same on the matrix $\boldsymbol{S}_{t'\times(t'+1)}$ up to the detection of the first column of $\boldsymbol{S}_{t'\times(t'+1)}$ to have no pivot positions by the conditions of $d_r = 0$ and $r - L_r = t'$. Thus, the ESBM algorithm with detection conditions of $d_r = 0$ and $r - L_r = t'$ will make a decoding error by claiming an error-locator polynomial of degree no greater than t'. We summarize the above discussion in the following theorem.

Theorem 10: For the decoding of a BCH code with t-error-correcting capability, the Berlekamp–Massey algorithm with the early-stopped detection conditions of $d_r = 0$ and $r - L_r = t', 1 \le t' \le t$, can correct up to t' errors but has a decoding error if the number of errors actually occurred is greater than t' and no greater than t. □

From the above theorem, it is now clear that the early-stopped detection conditions should be set to $d_r = 0$ and $r - L_r = t$ in order to correct up to t errors. Thus, the conjecture of the minimality of $t + e$ iterations raised in [2] is affirmed by the following corollary.

Corollary 11: To decode a BCH code up to its error-correcting capability t, the minimal number of iterations to be performed in the Berlekamp–Massey algorithm is $t + e$, where e is the number of errors actually occurred. □

Again, consider the case that the number e of errors actually occurred is greater than t', but no greater than t. As stated in Theorem 10, the ESBM algorithm with detection conditions of $d_r = 0$ and $r - L_r = t'$ cannot detect this error pattern. However, if we keep checking $d_r = 0$ after the early-stopped conditions are detected, we must find a violation, i.e., an r with $d_r \neq 0$, within $(t - t')$ steps (actually $(e - t')$ steps). Otherwise, we find the first column of the matrix $\boldsymbol{S}$ which has no pivot positions and the number e of errors is, thus, no greater than t', a contradiction. We state the above discussion in a more general setting in the following theorem.

Theorem 12: Let $1 \le t' < t'' \le t$. For a t-error-correcting BCH code, the Berlekamp–Massey algorithm with the early-stopped detection conditions of $d_r = 0$ and $r - L_r = t'$ and with up to $(t'' - t')$ extra steps to check $d_r = 0$ can correct up to t' errors and detect up to t'' errors simultaneously. □

Theorem 12 is more efficient than the results in [6, Theorem 7.5.3] where the early-stopped detection conditions are not adopted.

VI. Complexity Analysis

The conventional Berlekamp–Massey algorithm takes constant $2t$ steps to use $2t$ syndromes to determine the coefficients of the error-locator polynomial, while the ESBM algorithm takes only $(t + e)$ steps to use $(t + e)$ syndromes without requiring extra computation efforts. Thus, the ESBM algorithm can save both the processing time and the computational power. In this section, we will analyze the multiplicative complexity of the ESBM algorithm and compare it with those of the conventional Berlekamp–Massey algorithm and a variant of the Gaussian elimination in [10].

As shown in Section IV, the discrepancy d_r should be calculated at each rth step in the ESBM algorithm. We first note that in the first search phase, no finite-field multiplications are needed to calculate d_r since $\boldsymbol{\Lambda}_{L_{r-1}+1}^{(r-1-L_{r-1})} = 1 \; \forall r \in [1, n_1]$, and at most $(r - 1 - n_1)$ finite-field multiplications are needed to calculate d_r at each rth step in the first update phase except at the overlap with the first search phase, i.e., $r \in [n_1 + 1, 2n_1]$. This is because that the linear combination coefficient vector $\boldsymbol{\Lambda}_{L_{r-1}+1}^{(r-1-L_{r-1})}$ for r in $[n_1+1, 2n_1]$ has nonzero components only in the right $(r-1-n_1)$ positions, excluding the rightmost position where the component is always 1. Now for the jth iteration with $j \ge 2$, the number of finite-field multiplications needed to calculate d_r is $L_{r_{j-1}}$ when the rth step is in the jth search phase, i.e., $r \in [2L_{r_{j-1}} + 1, r_j]$, by (17), and the number of finite-field multiplications needed to calculate d_r is L_{r_j} when the rth step is in the jth update phase, except $r = r_j$, i.e., $r \in [r_j + 1, 2L_{r_j}]$,

by (22). Thus, the total number C_d of finite-field multiplications to calculate $t+e$ discrepancies is upper bounded by

$$\begin{aligned} C_d &\le \sum_{r=n_1+1}^{2n_1} (r-1-n_1) + n_2 L_{r_1} + \sum_{j=2}^{J} (r_{j+1}-r_j) L_{r_j} \\ &= \sum_{j=1}^{J} \left(L_{r_{j+1}} - L_{r_{j-1}}\right) L_{r_j} - \frac{n_1(n_1+1)}{2} \text{ by (13)} \\ &= L_{r_{J+1}} L_{r_J} - \frac{n_1(n_1+1)}{2} \end{aligned}$$

where we define $r_0 = 0$ and $r_{J+1} = t+e$ for convenience. Since $L_{r_J} = e$ by (15) and $r_{J+1} = L_{r_{J+1}} + L_{r_J}$ by an extension of (13), we have $L_{r_{J+1}} = t$. Thus, we have

$$C_d \le te - \frac{n_1(n_1+1)}{2} \le te - 1. \tag{36}$$

Additional finite-field multiplications should be performed in each update phase for updating the linear combination coefficient vector. In the first update phase, there is no multiplication to be done at $r = r_1$ since $\mathbf{\Lambda}_D = 0$. However, one multiplication is needed for the calculation of $d_r D^{-1}$ for each $r \in [r_1, r_1+n_1]$ if $d_r \neq 0$. Now if the rth step is in the jth update phase, $j \ge 2$, with $d_r \neq 0$, then $(L_{r_{j-2}}+1)$ and $(L_{r_{j-1}}+1)$ finite-field multiplications are needed for $r = r_j$ and for $r \in [r_j+1, r_j+n_j]$ by (20) and (21), respectively. Thus, the total number C_σ of finite-field multiplications for updating the linear combination coefficient vector is upper bounded by

$$\begin{aligned} C_\sigma &\le n_1 + \sum_{j=2}^{J} \left(\left(L_{r_{j-2}}+1\right) + n_j \left(L_{r_{j-1}}+1\right) \right) \\ &= \sum_{j=1}^{J} n_j \cdot L_{r_{j-1}} + \sum_{j=1}^{J-2} L_{r_j} + (J-1) + e \end{aligned} \tag{37}$$

since $n_1 + \cdots + n_J = e$ and $L_{r_0} = 0$, where equality holds when every discrepancy d_r is nonzero in each update phase. Since $0 = L_{r_0} < L_{r_1} < \ldots < L_{r_{J-1}} < L_{r_J} = e$, $n_1 + \cdots + n_J = e$, and $n_j = L_{r_j} - L_{r_{j-1}} > 0 \, \forall j$ by (14) and (15), each of the first three terms of the upper bound in (37) has the maximum value when $J = e$, i.e., $n_j = 1$ and $L_{r_j} = j$, $\forall 1 \le j \le J = e$. The worst case is shown in Fig. 4, where the discrepancy d_r is nonzero for $1 \le r \le 2e$ and is zero for $2e+1 \le r \le t+e$. Note that the multiplicative complexity C_d of calculating the discrepancies in this worst case reaches the upper bound $te-1$ in (36). Now the total number C_σ of finite-field multiplications for updating the linear combination coefficient vector is upper bounded by

$$C_\sigma \le e^2. \tag{38}$$

Combining (36) and (38), the multiplicative complexity C_{ESBM} of the ESBM algorithm is

$$C_{\mathrm{ESBM}} = C_d + C_\sigma \le te + e^2 - 1. \tag{39}$$

Since the conventional Berlekamp–Massey algorithm cannot stop earlier, additional $(t-e)$ steps are needed to check the discrepancies d_r, for r from $t+e+1$ to $2t$, which must be zeros, the multiplicative complexity C_{BM} of the conventional Berlekamp–Massey algorithm is upper bounded by

$$C_{\mathrm{BM}} \le C_{\mathrm{ESBM}} + (t-e)e \le 2te - 1 \tag{40}$$

where the upper bound $(2te-1)$ is reached by the worst case shown in Fig. 4.

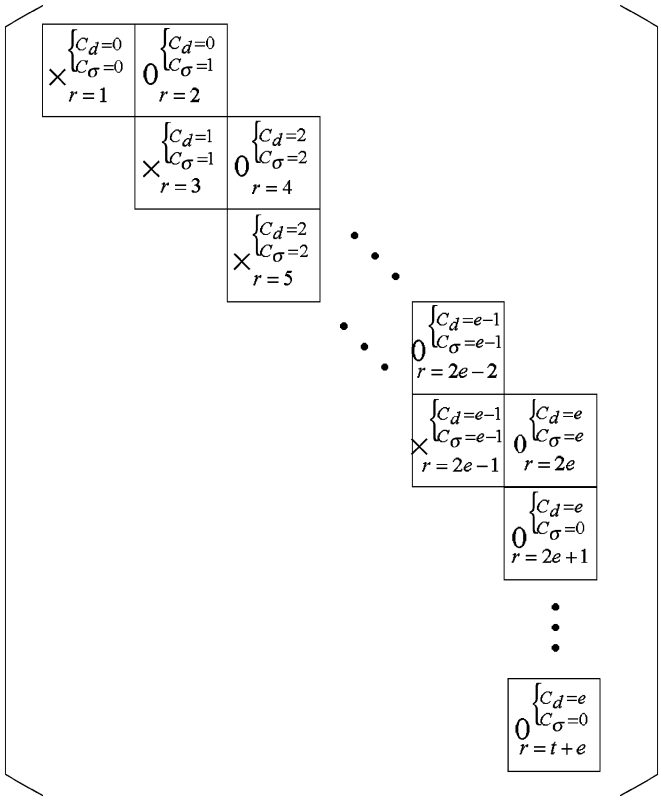

Fig. 4. Worst case with the maximum number of finite-field multiplications in the ESBM algorithm, where the pivot positions are marked by "×."

We must address here that the upper bound of C_{BM} in (40) is owing to the equivalence of the conventional Berlekamp–Massey algorithm to the refined FIA shown in [1]. In the classical presentation of the conventional Berlekamp–Massey algorithm, for example, please see the flow chart in [6, Fig. 7.5], the coefficients of the backup polynomial $B(x)$ in this flow chart are normalized before being stored in a shift register. This is equivalent to multiplying the vector $\mathbf{\Lambda}_D$ by D^{-1} before storing the vector $\mathbf{\Lambda}_D$ at the first step of each jth update phase, i.e., $r = r_j$, without storing D^{-1} for later use as in the pseudocode of the ESBM algorithm. Thus, the normalization operation introduces additional finite-field multiplications. The case shown in Fig. 4 is still the worst case for the classical presentation of the conventional Berlekamp–Massey algorithm. Thus, the multiplicative complexity of the classical presentation is upper bounded by

$$(2te-1) + \sum_{i=1}^{e} (i-1) + 1 = 2te + \frac{1}{2}(e^2 - e) \tag{41}$$

where the additional one finite-field multiplication is from the initialization of $B(x) = 1$ as shown in [6, Fig. 7.5]. Comparing (41) with (40), the refined FIA presentation of the conventional Berlekamp–Massey algorithm is superior in the computational complexity.

TABLE I
UPPER BOUNDS OF FINITE-FIELD MULTIPLICATIVE COMPLEXITY FOR THE ESBM ALGORITHM, THE VARIANT OF GAUSSIAN ELIMINATION IN [10] AND THE CONVENTIONAL BERLEKAMP–MASSEY ALGORITHM

e	C_{ESBM}	C_{HV}	C_{BM}
1	t	t	$2t-1$
2	$2t+3$	$3t$	$4t-1$
3	$3t+8$	$6t-1$	$6t-1$
4	$4t+15$	$10t-4$	$8t-1$
5	$5t+24$	$15t-10$	$10t-1$
6	$6t+35$	$21t-20$	$12t-1$
7	$7t+48$	$28t-35$	$14t-1$
8	$8t+63$	$36t-56$	$16t-1$
9	$9t+80$	$45t-84$	$18t-1$
10	$10t+99$	$55t-120$	$20t-1$
11	$11t+120$	$66t-165$	$22t-1$
12	$12t+143$	$78t-220$	$24t-1$
13	$13t+168$	$91t-286$	$26t-1$
14	$14t+195$	$105t-120$	$28t-1$
15	$15t+224$	$120t-455$	$30t-1$
16	$16t+255$	$136t-560$	$32t-1$
17	$17t+288$	$153t-680$	$34t-1$
18	$18t+323$	$171t-816$	$36t-1$
19	$19t+360$	$190t-210$	$38t-1$
20	$20t+399$	$210t-1140$	$40t-1$

Note that the exact multiplicative complexity of decoding a specific codeword depends on the codeword itself and the error pattern occurred. Thus, an exact analysis of the multiplicative complexity is intractable. However, by our experience in doing computer simulation, the worst case event as described in Fig. 4 occurs with very high probability so that the worst case complexity analysis we conducted in above is able to address the expected performance of the early-stopped and the conventional Berlekamp–Massey algorithms.

In Table I, we list the upper bounds of finite-field multiplicative complexity of the ESBM algorithm, $C_{\rm ESBM}$ in (39), the conventional Berlekamp–Massey algorithm, $C_{\rm BM}$ in (40), and a variant of Gaussian elimination, $C_{\rm HV}$ taken from [10]. All the three algorithms requires exactly e finite-field inversions within a decoding cycle. As demonstrated in Table I, the variant of Gaussian elimination in [10] is more efficient than the ESBM algorithm only in the case of $t = e = 2$. They have the same multiplicative complexity when $e = 1$ or $2 \leq e \leq t = 3$, and the ESBM algorithm is superior to this variant of Gaussian elimination in any other case. The variant of Gaussian elimination is more efficient than the conventional Berlekamp–Massey algorithm when $e = 1$ or 2 with $e < t$. They have the same multiplicative complexity when $t = e = 1, 2$ or when $e = 3$. However, when $e > 3$, the conventional Berlekamp–Massey algorithms are superior to this variant of Gaussian elimination.

VII. CONCLUSION

In this paper, we have presented an exposition from a restricted Gaussian elimination to several aspects of the ESBM algorithm. It can be seen that the prominent features of the Berlekamp–Massey algorithm can be fully reached naturally by operating a restricted Gaussian elimination on an appropriate syndrome matrix which has structural properties. The formulation taken in this paper makes the deep insights of the Berlekamp–Massey algorithm crystal clear. With this formulation, we have justified the minimality of $(t+e)$ iterations in the ESBM algorithm and related it to the subject of simultaneous error correction and detection. We have been able to derive the exact upper bound of the multiplicative complexity of the ESBM algorithm and have shown that except for a trivial case, the ESBM algorithm is the most efficient algorithm for the determination of the error-locator polynomial.

APPENDIX I
PROOF OF LEMMA 3

With (5) and by induction, the kth column $\boldsymbol{a}_k$, $1 \leq k \leq n$, of the original matrix $\boldsymbol{A}$ is a linear combination of the first k columns $\boldsymbol{a}'_v$, $1 \leq v \leq k$, of the left-reduced matrix $\boldsymbol{A}'$

$$\boldsymbol{a}_k = \gamma_{k-1}\boldsymbol{a}'_1 + \cdots + \gamma_1 \boldsymbol{a}'_{k-1} + \boldsymbol{a}'_k \tag{42}$$

where γ_v, $1 \leq v \leq k-1$, are constants and dependent on the column index k.

Now assume that the left-reduced matrix $\boldsymbol{A}'$ of $\boldsymbol{A}$ has a pivot at the (i,j)th position. Then as stated in (6), column $\boldsymbol{a}_j$ of $\boldsymbol{A}$ is a partial linear combination of its previous columns up to the $(i-1)$th entry. Suppose that column $\boldsymbol{a}_j$ of $\boldsymbol{A}$ is also a partial linear combination of its previous columns up to the ith entry, i.e., there exist $\beta_1, \ldots, \beta_{i-1}$ such that

$$a_{u,j} = -\beta_{j-1}a_{u,1} - \cdots - \beta_1 a_{u,j-1}, \quad \forall 1 \leq u \leq i. \tag{43}$$

Then, by recursively using (42) from $k = 1$ to $k = j$, we have

$$\beta_{j-1}\boldsymbol{a}_1 + \cdots + \beta_1 \boldsymbol{a}_{j-1} + \boldsymbol{a}_j = \beta'_{j-1}\boldsymbol{a}'_1 + \cdots + \beta'_1 \boldsymbol{a}'_{j-1} + \boldsymbol{a}'_j \tag{44}$$

for some constants $\beta'_1, \ldots, \beta'_{j-1}$. Thus, column $\boldsymbol{a}'_j$ of the left-reduced matrix $\boldsymbol{A}'$ can be replaced by the column vector

$$\beta'_{j-1}\boldsymbol{a}'_1 + \cdots + \beta'_1 \boldsymbol{a}'_{j-1} + \boldsymbol{a}'_j \tag{45}$$

by a series of left-column replacement operations. Since the first i entries of the resulted column in (45) are all zeros by (44) and (43), the leading nonzero entry of column $\boldsymbol{a}'_j$ of the left-reduced matrix $\boldsymbol{A}'$ can be eliminated by such a series of left-column replacement operations, which is a contradiction to the definition of a left-reduced matrix. Thus, column $\boldsymbol{a}_j$ of $\boldsymbol{A}$ cannot be a partial linear combination of its previous columns in $\boldsymbol{A}$ up to the ith entry.

Conversely, we assume that column $\boldsymbol{a}_j$ of $\boldsymbol{A}$ is a partial linear combination of its previous columns up to the $(i-1)$th entry but not to the ith entry. With similar argument as in the last paragraph, the first $(i-1)$ entries of column $\boldsymbol{a}'_j$ of the left-reduced matrix $\boldsymbol{A}'$ must be all zeros, otherwise the leading nonzero entry of column $\boldsymbol{a}'_j$ can be eliminated by a series of left-column replacement operations on $\boldsymbol{A}'$, a contradiction. Thus, the leading nonzero entry of column $\boldsymbol{a}'_j$ must be below the $(i-1)$th coordinate, if exists. However, if the leading nonzero entry of column $\boldsymbol{a}'_j$ is below the ith coordinate or does not exist, then column $\boldsymbol{a}_j$ of $\boldsymbol{A}$ is a partial linear combination of its previous columns at least up to the ith entry by (5), which is a contradiction too. Thus, column $\boldsymbol{a}'_j$ has a leading nonzero entry at the ith coordinate, i.e., the left-reduced matrix $\boldsymbol{A}'$ has a pivot at the (i,j)th position. This completes the proof of this lemma.

APPENDIX II
PROOF OF LEMMA 4

The lemma is trivial if $\boldsymbol{A}$ is a zero matrix. Now let $\boldsymbol{A}$ be a nonzero $n \times n$ square symmetric matrix. We shall produce a sequence of $n_k \times n_k$ square symmetric matrices $\boldsymbol{A}^{(k)}$, $k = 0, 1, \ldots$, by iteration as follows. Let $\boldsymbol{A}^{(0)} = \boldsymbol{A}$ and $n_0 = n$ for $k = 0$. By assuming that the $n_k \times n_k$ square symmetric matrix $\boldsymbol{A}^{(k)}$ is given, we construct the next $n_{k+1} \times n_{k+1}$ square symmetric matrix $\boldsymbol{A}^{(k+1)}$ by considering the following two cases.

1) The first (i.e., the leftmost) column of $\boldsymbol{A}^{(k)}$ is a zero vector, i.e., $a_{i,1}^{(k)} = 0$ for all $1 \le i \le n_k$. Then there does not exist any pivot position in the first column of $\boldsymbol{A}^{(k)}$, and by the symmetry of $\boldsymbol{A}^{(k)}$, the first (i.e., the topmost) entry $a_{1,j}^{(k)}$ of each column $j, 1 \le j \le n_k$, is zero and then there does not exist any pivot position in the first row of $\boldsymbol{A}^{(k)}$. Let $\boldsymbol{A}^{(k+1)}$ be the submatrix of $\boldsymbol{A}^{(k)}$ by deleting the first column and the first row from $\boldsymbol{A}^{(k)}$. It can be seen that the execution of a left-column replacement operation on the matrix $\boldsymbol{A}^{(k)}$ is equivalent to that on the matrix $\boldsymbol{A}^{(k+1)}$. Note that $\boldsymbol{A}^{(k+1)}$ is an $(n_k - 1) \times (n_k - 1)$ square symmetric matrix. We let $n_{k+1} = n_k - 1$.
2) The first (i.e., the leftmost) column of $\boldsymbol{A}^{(k)}$ is a nonzero vector. Let $a_{\ell,1}^{(k)}$ be the first nonzero entry of column 1. Then by Lemma 3, $\boldsymbol{A}^{(k)}$ has a pivot position at the $(\ell, 1)$th entry. Again by the symmetry of $\boldsymbol{A}^{(k)}$, we have $a_{1,j}^{(k)} = 0$ for all $1 \le j \le \ell - 1$ and $a_{1,\ell}^{(k)} \ne 0$ and then $\boldsymbol{A}^{(k)}$ has a pivot position at the $(1, \ell)$th entry also by Lemma 3. Now, we can eliminate first the $(1, j)$th entries with $\ell < j \le n_k$ to zeros via left-column replacement operations by using the ℓth column, $(a_{1,\ell}^{(k)}, a_{2,\ell}^{(k)}, \ldots, a_{n_k,\ell}^{(k)})^T$, and then, second, the (ℓ, j)th entries with $1 < j \le n_k$ by using the first column, $(0, \ldots, 0, a_{\ell,1}^{(k)}, a_{\ell+1,1}^{(k)}, \ldots, a_{n_k,1}^{(k)})^T$. Let $\boldsymbol{A}'$ be the resulted matrix after all of the above left-column replacement operations have been done on $\boldsymbol{A}^{(k)}$. Then, the (i, j)th entry of $\boldsymbol{A}'$ with $i \ne 1, \ell$ or $j \ne 1, \ell$ is shown in (46)

$$a'_{i,j} = \begin{cases} a_{i,j}^{(k)}, & \text{if } i < \ell \text{ and } j < \ell \\ a_{i,j}^{(k)} - \left(\frac{a_{i,1}^{(k)}}{a_{\ell,1}^{(k)}}\right) a_{\ell,j}^{(k)}, & \text{if } i > \ell \text{ and } j < \ell \\ a_{i,j}^{(k)} - \left(\frac{a_{i,\ell}^{(k)}}{a_{1,\ell}^{(k)}}\right) a_{1,j}^{(k)}, & \text{if } i < \ell \text{ and } j > \ell \\ a_{i,j}^{(k)} - \left(\frac{a_{i,\ell}^{(k)}}{a_{1,\ell}^{(k)}}\right) a_{1,j}^{(k)} - \left(\frac{a_{i,1}^{(k)}}{a_{\ell,1}^{(k)}}\right) a_{\ell,j}^{(k)} & \\ \quad + \left(\frac{a_{i,1}^{(k)} a_{1,j}^{(k)}}{a_{\ell,1}^{(k)} a_{1,\ell}^{(k)}}\right) a_{\ell,\ell}^{(k)}, & \text{if } i > \ell \text{ and } j > \ell. \end{cases} \tag{46}$$

Since the matrix $\boldsymbol{A}^{(k)}$ is symmetric, i.e., $a_{i,j}^{(k)} = a_{j,i}^{(k)}$, and by (46), we have $a'_{i,j} = a'_{j,i}$, if $i \ne 1, \ell$ or $j \ne 1, \ell$. This fact implies that the submatrix of $\boldsymbol{A}'$ by ignoring the first column, first row and the ℓth column, ℓth row is symmetric. For example, consider the following 5×5 symmetric matrix

$$\boldsymbol{A}^{(k)} = \begin{pmatrix} 0 & 0 & c & d & e \\ 0 & f & g & h & i \\ c & g & j & k & l \\ d & h & k & m & n \\ e & i & l & n & o \end{pmatrix} \tag{47}$$

with $c \ne 0$. Since c is the first nonzero entry of the first column and of the first row, the matrix $\boldsymbol{A}$ has two pivot positions at the (3, 1)th and the (1, 3)th entries respectively. After applying left-column replacement operations on $\boldsymbol{A}^{(k)}$ with these two pivots, we have the resulted matrix $\boldsymbol{A}'$ as follows:

$$\begin{pmatrix} 0 & 0 & c & 0 & 0 \\ 0 & f & g & \frac{hc-gd}{c} & \frac{ic-ge}{c} \\ c & 0 & j & 0 & 0 \\ d & \frac{hc-gd}{c} & k & \frac{mc^2-2kdc+jd^2}{c^2} & \frac{nc^2-kec+jde-lcd}{c^2} \\ e & \frac{ic-ge}{c} & l & \frac{nc^2-kec+jde-lcd}{c^2} & \frac{mc^2-2kdc+jd^2}{c^2} \end{pmatrix}.$$

It can be seen that the further execution of left-column replacement operations on the matrix $\boldsymbol{A}^{(k)}$ to find other pivot positions is equivalent to that on the matrix $\boldsymbol{A}'$ and furthermore, equivalent to that on the submatrix of $\boldsymbol{A}'$ by ignoring the first and third columns and the first and third rows. We now let $\boldsymbol{A}^{(k+1)}$ to be this submatrix. For the given example, the matrix $\boldsymbol{A}^{(k+1)}$ is

$$\begin{pmatrix} f & \frac{hc-gd}{c} & \frac{ic-ge}{c} \\ \frac{hc-gd}{c} & \frac{mc^2-2kdc+jd^2}{c^2} & \frac{nc^2-kec+jde-lcd}{c^2} \\ \frac{ic-ge}{c} & \frac{nc^2-kec+jde-lcd}{c^2} & \frac{mc^2-2kdc+jd^2}{c^2} \end{pmatrix}.$$

As claimed before, $\boldsymbol{A}^{(k+1)}$ is readily an $n_{k+1} \times n_{k+1}$ square symmetric matrix, where $n_{k+1} = n_k - 1$ if $\ell = 1$ and $n_{k+1} = n_k - 2$ otherwise.

By iteration, we generate a sequence of $n_k \times n_k$ square symmetric matrices $\boldsymbol{A}^{(k)}$, $k = 0, 1, \ldots$, from the square symmetric matrix $\boldsymbol{A}$. Since the dimension n_k of these matrices decreases strictly as k increases, this iteration process must be terminated, says at $k = M$. By the construction of matrices $\boldsymbol{A}^{(k)}$, $0 \le k \le M$, in the above, we have the following two properties.

1) A pivot position in the matrix $\boldsymbol{A}^{(k)}$, $1 \le k \le M$, corresponds to a pivot position in the matrix $\boldsymbol{A}^{(k-1)}$.
2) A diagonal position in the matrix $\boldsymbol{A}^{(k)}$, $1 \le k \le M$, corresponds to a diagonal position in the matrix $\boldsymbol{A}^{(k-1)}$ and a pair of transposed positions in the matrix $\boldsymbol{A}^{(k)}$ correspond to a pair of transposed positions in the matrix $\boldsymbol{A}^{(k-1)}$, and each corresponding pivot position in the matrix $\boldsymbol{A}^{(k-1)}$ to a pivot position in the matrix $\boldsymbol{A}^{(k)}$ is neither in the first column nor in the first row of the matrix $\boldsymbol{A}^{(k-1)}$.

Let the matrix $\boldsymbol{A}^{(L)}$, $L \in [0, M]$, be the last nonzero matrix in the sequence. Then by the construction in above, $\boldsymbol{A}^{(L)}$ has either a pivot position at a diagonal entry, the (1, 1)th position, or two pivot positions at a pair of transposed entries, the $(\ell_L, 1)$th and the $(1, \ell_L)$th positions for some $\ell_L > 1$. This shows that the matrix $\boldsymbol{A}^{(L)}$ satisfies this lemma, i.e., $\boldsymbol{A}^{(L)}$ has a pivot position at the (i, j)th entry if and only if it has a pivot position at the transposed (j, i)th entry. By the two properties in above, the matrix $\boldsymbol{A}^{(L-1)}$ has either a pivot position at a corresponding diagonal entry or two pivot positions at a corresponding pair of transposed entries, each of which is neither in the first column nor in the first row of the matrix $\boldsymbol{A}^{(L-1)}$. If Case 2 above is applicable, the matrix $\boldsymbol{A}^{(L-1)}$ has either an additional pivot position at the (1, 1)th entry or two additional pivot positions at the $(\ell_{L-1}, 1)$th and the $(1, \ell_{L-1})$th entries for some $\ell_{L-1} > 1$.

This shows that the matrix $\boldsymbol{A}^{(L-1)}$ satisfies this lemma. By induction, all of the matrices $\boldsymbol{A}^{(k)}$, $0 \leq k \leq M$, in the sequence satisfy this lemma. In particular, the first matrix $\boldsymbol{A}^{(0)} = \boldsymbol{A}$ satisfies this lemma. This completes the proof.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments and suggestions that helped improve the presentation of this paper.

REFERENCES

[1] G.-L. Feng and K.-K. Tzeng, "A generalization of the Berlekamp–Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes," *IEEE Trans. Inf. Theory*, vol. 37, no. 5, pp. 1274–1287, Sep. 1991.

[2] C.-L. Chen, "High-speed decoding of BCH codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 2, pp. 254–256, Mar. 1981.

[3] K.-K. Tzeng, C. R. P. Hartmann, and R.-T. Chien, "Some notes on iterative decoding," in *Proc. 9th Annu. Allerton Conf. Circuit and System Theory*, 1971, pp. 689–695.

[4] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed. Cambridge, MA: MIT Press, 1971.

[5] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[6] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

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

[8] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. IT-15, no. 1, pp. 122–127, Jan. 1969.

[9] D. Gorenstein and N. Zierler, "A class of error-correcting codes in p^m symbols," *J. Soc. Ind. Appl. Math.*, vol. 9, pp. 207–214, Jun. 1961.

[10] J. Hong and M. Vetterli, "Simple algorithms for BCH decoding," *IEEE Trans. Commun.*, vol. 43, no. 8, pp. 2324–2333, Aug. 1995.

[11] K. Imamura and W. Yoshida, "A simple derivation of the Berlekamp–Massey algorithm and some applications," *IEEE Trans. Inf. Theory*, vol. 33, no. 1, pp. 146–150, Jan. 1987.

Chih-Wei Liu (M'03) was born in Taiwan, R.O.C. He received the B.S. and Ph.D. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1991 and 1999, respectively.

From 1999 to 2000, he was an Integrated Circuits Design Engineer at the Electronics Research and Service Organization (ERSO) of Industrial Technology Research Institute (ITRI), Hsinchu. Then, near the end of 2000, he joined SoC Technology Center (STC) of ITRI as a Project Leader. He eventually left ITRI at the end of October 2003. He is currently with the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu, as an Assistant Professor. His current research interests are SoC and VLSI system design, processors for embedded computing systems, digital signal processing, digital communications, and coding theory.

Dr. Liu received the Best Paper Award at APCCAS in 2004 and the Outstanding Design Award at ASP-DAC in 2006.

Chung-Chin Lu (S'86–M'88) was born in Taiwan, R.O.C. He received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1981, and the Ph.D. degree from University of Southern California, Los Angeles, in 1987, both in electrical engineering.

Since 1987, he has been with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan. He is currently a Professor. His current research interests include coding theory, digital communications, bioinformatics, and quantum communications.