the symbol to be corrected. The AND gate
produces a 1 output only when 1, 0, ... 0 appears in the syndrome register and corrects
the symbol in error.

3. Bose-Chaudhuri-Hocquenghem (BCH) codes
These codes were discovered by Hocquenghem
in 1959 and independently by Bose and
Chaudhuri in 1960. The BCH codes are the most
powerful cyclic block codes known for
correcting random errors. However, their
performance on random channels cannot
match convolutional codes with sequential or Viterbi
decoding (particularly with soft decision decoding).

Def: Least common multiple (lcm)

The least common multiple of \(a(d), b(d), \ldots d(d)\)
is the polynomial of lowest degree that is
divisible by \(a(d), \ldots, d(d)\).

Def: BCH Code

Let \(x\) be any nonzero element of \(GF(q^m)\)
and order of \(x = n\). Let \(m_0\) be any integer and\(d\) any integer in the range \(2 \leq d \leq n\). The
BCH code corresponding to \(x, m_0\), and \(d\) is the
cyclic code with symbols from \(GF(q)\) where
the generator polynomial \(g(d)\) is the minimum
degree non-zero polynomial over \(GF(q)\) having
\(1^{m_0}, x^{m_0}, \ldots, x^{m_0 + d - 2}\) as roots.
remark: since $d < n=ord(a)$ these $d-1$

roots are distinct.

Let $m_i(x)$ be the minimal polynomial
for $x^i$ over $GF(q)$. Then

$$g(x) = \gcd \{ m_0(x), m_1(x), \ldots, m_{ord(x)-1} \}$$

Even if $d \mid n$ it may happen that $m_d(x)$
is already a factor of $g(x)$ since

$m_{ord-1}$ may have the same minimal polynomial
as $x^i$ for $m_0 \leq i \leq m_0+ord-2$. A BCH code
is not defined for such $d$ since it is exactly
the same as the BCH code for a larger $d$.

Ex: $q=2$, $m_0=1$

Assume $d=2$ then $x^1, x^{ord-2}$ are
roots of $g(x)$, but $x^2$ must also be a root
of $g(x)$ so no BCH code is defined with $d=2$
in this example. The code for $d=2$ is the
same as the code for $d=2$.

**Theorem 17** Calculation of $N$

Let $g(x) = \prod m_i(x)$ where $1 \leq m_i(x)$ are
distinct, monic, irreducible polynomials over $GF(q)$.
Let $n_i$ be the multiplicative order of the roots
of $m_i(x)$. Then $N = \gcd (n_1, n_2, \ldots, n_e)$

**Proof**

For each $i$, $n_i$ is the smallest integer such that
$m_i(x)$ divides $x^{n_i} - 1$. Since $g(x)$ must divide
$x^{n_i} - 1$, $x_i$ is also a root of $x^{n_i} - 1$ and therefore
\( n \), divides \( N \). Therefore \( N \geq \text{lcm}(n, n_2, \ldots, n_d) \).

On the other hand \( \text{lcm}(n, \ldots, n_d) = 1 \) for all \( i \) so that \( m_i(\alpha) \) divides \( \text{lcm}(n, \ldots, n_d) - 1 \) and therefore \( g(\alpha) \) divides \( \text{lcm}(n, \ldots, n_d) - 1 \). Thus \( N = \text{lcm}(n, \ldots, n_d) \).

\[ \text{G.E.D.} \]

**Theorem 18.** Length of BCH Codes

If \( d = 2 \), \( g(\alpha) \) is the minimal polynomial of \( \alpha^{m_0} \) and \( N = \text{order} \alpha^{m_0} \).

For \( d \geq 3 \), \( N = \text{order} \alpha \)

**Proof:**

Let \( n = \text{order} \alpha \). Then \( \alpha^n = 1 \) and \( \sum_{i=0}^{\infty} \alpha^{i} = 1 \) so that \( n \) divides \( n \). By Theorem 17, \( N \leq n \). Since \( g(\alpha) \) divides \( \alpha^{N-1} - 1 \), \( \alpha^i \) for \( m_0 \leq i \leq m_0 + d - 2 \) is a root of \( \alpha^{N-1} - 1 \).

For \( d \geq 3 \)

\[(\alpha^{m_0})^N = 1 \quad \text{and} \quad (\alpha^{m_0 + 1})^N = 1 \Rightarrow \alpha^{m_0 N} \alpha^N = 1 \Rightarrow \alpha^N = 1 \quad \text{since} \quad \alpha^{m_0 N} = 1, \text{Thus} \quad N \text{ divides } N \quad \text{and} \quad n \leq N.\]

\[ \text{Q.E.D.} \]

In most cases \( \alpha \) is taken as a primitive element of \( GF(\alpha^m) \) so that for \( d \geq 3 \), \( N = \alpha^{m-1} \). \( m_0 \) is usually chosen as 1.

**Example:** \( m_0 = 1, \ d = 5, \ g = 2, \ m = 4 \)

Let \( \alpha \) be a root of \( \sqrt{f(\alpha)} = \alpha^4 + \alpha + 1 \)
$GF(2^4)$ Field Elements

\begin{align*}
0 & \quad 0 \quad 0 \quad 0 \quad 0 \\
1 & \quad 1 \quad 0 \quad 0 \quad 0 \\
2 & \quad 0 \quad 1 \quad 0 \quad 0 \\
2^2 & \quad 0 \quad 0 \quad 1 \quad 0 \\
4 & \quad 0 \quad 0 \quad 0 \quad 1 \\
2^4 & \quad 1 \quad 1 \quad 0 \quad 0 \\
2^5 & \quad 0 \quad 1 \quad 1 \quad 0 \\
2^6 & \quad 0 \quad 0 \quad 1 \quad 1 \\
2^7 & \quad 1 \quad 1 \quad 0 \quad 1 \\
2^8 & \quad 1 \quad 0 \quad 1 \quad 0 \\
2^9 & \quad 0 \quad 1 \quad 0 \quad 1 \\
2^{10} & \quad 1 \quad 1 \quad 1 \quad 0 \\
2^{11} & \quad 0 \quad 1 \quad 1 \quad 1 \\
2^{12} & \quad 1 \quad 1 \quad 1 \quad 1 \\
2^{13} & \quad 1 \quad 0 \quad 1 \quad 1 \\
2^{14} & \quad 1 \quad 0 \quad 0 \quad 1 \\
\end{align*}

$m_0 + d = 1 + 5 = 4 \quad \text{ so that } \quad q(d) \text{ must have}$

$a, 2^2, 2^3, 2^4 \quad \text{ as roots}$

$a, 2^2, 2^4 \quad \text{ are roots of } \quad m_3(d) = d^4 + d + 1$

For $\ m_3(d) \text{ roots are } 2^3, 2^6, 2^{12}, 2^{24} = 2^9$

$m_3(d) = c_0 + c_1 d + c_2 d^2 + c_3 d^3 + d^4$

\[ m_3(2) = c_0 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + c_1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + c_3 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0 \]
\[ c_6 = 1, \quad c_5 = 1, \quad c_2 = 1, \quad c_1 = 1 + 1 + 1 = 1 \]

\[ m_3 (D) = 1 + D + D^2 + D^3 + D^4 \]

and \[ g(D) = m_1 (D) m_3 (D) = D^8 + D^7 + D^6 + D^4 + 1. \]

Therefore \( n - k = 8, \ n = 15, \ k = 7 \) giving a \((15, 7)\) code.

**Comment:**

For \( q = 2 \), \[ m_1 (D) = m_2 (D). \] Therefore with \( m_0 = 1 \), \( q = 2 \), \( d \) odd

\[ g(D) = \text{lcm} \left\{ m_1 (D), m_3 (D), \ldots, m_{d-2} (D) \right\} \]

Each \( m_i (D) \) has degree \( i \) so that \( n - k = \deg g(D) \leq m(d-1)/2 \). It will be shown that \( d \) is a lower bound on \( \text{dim} \).

Therefore codes of length \( 2^m - 1 \) exist that correct all patterns of \( t \) or fewer errors with at most \( nt \) check symbols. For \( n_c = 1, \ q \neq 2 \) the weaker bound \( n - k \leq 2nt \) applies.

Specification of \( H \) in terms of \( \text{roots of } g(D) \)

For an arbitrary cyclic code, it has already been shown that a check matrix can be determined from \( h(D) = (D^{n-1})/g(D) \). In some cases, particularly for BCH codes, it will be more convenient to specify \( H \) in terms of the roots of \( g(D) \). This representation
Theorem 18.4

Let \( g(x) = \prod_{i=1}^{t} m_i(x) \) where \( \{m_i(x)\} \) are distinct, monic, irreducible polynomials. Let \( \alpha \) be a root of \( m_i(x) \). Then a check matrix for the code is:

\[
H = \begin{bmatrix}
1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1} \\
1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1}
\end{bmatrix}
\]

That is, \( x \) is a codeword \( \iff \) \( x^T H^T = 0 \)

Proof:

Let \( C \) be the space of codewords, and \( H \) the null space of the rows of \( H \). Let \( x \) be a vector such that \( x^T H = 0 \). Then \( \alpha, \alpha^2, \ldots, \alpha^t \) are roots of \( g(x) \).

Thus, \( m_1(x), \ldots, m_t(x) \) divide \( x(x) \) so that \( g(x) \) divides \( x(x) \) and \( x \in C \). (Only if)

On the other hand, any codeword has \( \alpha, \ldots, \alpha^t \) as roots so that all code words \( x \in C \). Q.E.D.

Example 1 (cont.)

\[
H = \begin{bmatrix}
1 & \alpha & \alpha^2 & \cdots & \alpha^{14} \\
1 & \alpha & \alpha^2 & \cdots & \alpha^{14} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha & \alpha^2 & \cdots & \alpha^{14}
\end{bmatrix}
\]

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

where the representations for \( \alpha \) from p. 156 were used.
EXAMPLE 2  Golay (23, 12) Code [See Lin for simple decoding algorithm]

To find binary cyclic codes with \( N = 23 \) it is necessary to find an element \( \beta \in GF(2^n) \) with order 23 for some \( k \). On \( GF(2^n) \) there are
\[ 2^n - 1 = 2047 = 23 \times 89 \] non-zero field elements. Let \( \alpha \) be a primitive element \( \in GF(2^n) \) and let \( \beta = \alpha^{89} \).

Then order \( \beta = 2047 / \gcd (2047, 89) = 23 \). The roots of the minimal polynomial for \( \beta \) are
\[ \beta, \beta^2, \beta^3, \beta^4, \beta^8, \beta^{16}, \beta^{32}, \beta^9, \beta^{18}, \beta^{36}, \beta^{13}, \beta^{26}, \beta^6, \beta^12 \]
Thus \( \deg m_{\beta} (1) = 11 \). Consider the element \( f = \beta^5 \) which is not a root of \( m_{\beta} (1) \). The order \( f = 23 / \gcd (23, 5) = 23 \)
so that by similar reasoning \( \deg m_{f} (1) = 11 \). Thus the factorization into irreducible polynomials over \( GF(2) \)
\[ \beta^{23} + 1 = (\beta + 1) m_{\beta} (1) m_{f} (1), \]

Through trial and error it can be shown that
\[ m_{\beta} (1) = 1 + \beta + \beta^7 + \beta^6 + \beta^5 + \beta^4 + \beta^3 + \beta^2 + \beta + 1 \]
and
\[ m_{f} (1) = 1 + \beta + \beta^7 + \beta^6 + \beta^5 + \beta^4 + \beta^3 + \beta^2 + 1. \]
The generator polynomial for the Golay code is \( g (x) = m_{\beta} (1) \).

The elements \( \beta, \beta^2, \beta^3, \beta^4 \) are all roots of \( g (x) \) so the code is a BCH code with \( n = 11 \) and \( d = 5 \).

Thus \( d_{\text{min}} \geq 5 \). By generating all code words or by the analysis in Sec 15.2 of Berlekamp it can be shown that actually \( d_{\text{min}} = 7 \) so that the code can correct all patterns of \( t = 3 \) or fewer errors.

Since
\[ \sum_{i=0}^{3} \binom{12}{i} = 2^{12}, \]
the code is a perfect code.
Ex. 3  Hamming Codes

Let \( f(D) \) be a primitive polynomial of degree \( n-K \) over \( \mathbb{F}_2 \) with \( \alpha \) as a root, \( m_0 = 1, \ d = 3 \). Then \( g(D) \) must have \( \alpha, \alpha^2 \) as roots. So

\[
g(D) = f(D)
\]

Then

\[
H = [1 \ \alpha \ \ldots \ \alpha^{n-1}]
\]

Suppose \( E(D) = D^k \), then

\[
s = E H^\top = \alpha^k
\]

so \( k = \log_\alpha s \) is error position.
Theorem 19: $d_{\min}$ for BCH codes

Let a BCH code over $GF(\phi)$ have parameters $a, m_0, d$. Then $d_{\min} \geq d$.

Proof:

Let

$$H = \begin{bmatrix}
(x_{m_0})^0 & (x_{m_0})^1 & (x_{m_0})^2 & \cdots & (x_{m_0})^{N-1} \\
(x_{m_0+1})^0 & x_{m_0+1}^1 & x_{m_0+1}^2 & \cdots & x_{m_0+1}^{N-1} \\
& \vdots & \ddots & \ddots & \vdots \\
& & & (\phi_{m_0+d-2})^0 & \phi_{m_0+d-2}^1 & \cdots & (\phi_{m_0+d-2})^{N-1}
\end{bmatrix}$$

It will be shown that any $d-1$ columns of $H$ are linearly independent so that there are no codewords of weight $\leq d-1$. Let

$0 \leq i_1 < i_2 < \cdots < i_{d-1} \leq N-1$ be $d-1$ columns corresponding to the roots raised to the $i_j$ power.

Let $\Delta$ be the determinant over $GF(\phi^m)$ of these columns, i.e.

$$\Delta = \begin{bmatrix}
(x_{m_0})^{i_1} & (x_{m_0})^{i_2} & \cdots & (x_{m_0})^{i_{d-1}} \\
& \vdots & \ddots & \vdots \\
& & \ddots & \ddots \\
& & & (\phi_{m_0+d-2})^{i_1} & (\phi_{m_0+d-2})^{i_2} & \cdots & (\phi_{m_0+d-2})^{i_{d-1}}
\end{bmatrix}$$

$$\Delta = \alpha^{m_0(i_1+i_2+\cdots+i_{d-1})} \begin{bmatrix}
1 & & & \\
\alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_{d-1}} \\
& \vdots & \ddots & \vdots \\
& & & \alpha^{i_{d-2}} & \alpha^{i_{d-2}} & \cdots & \alpha^{i_{d-1}}
\end{bmatrix}$$
A determinant of this form is known as a Vandermonde determinant.

**Lemma:**

Let \( \Delta = \begin{vmatrix} \ 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{vmatrix} \)

Then \( \Delta = -\frac{n!}{\prod_{j=1}^{n} (x_j - x_i)} \)

**Proof:**

If \( \Delta \) is expanded in a sum of products, each product has one term from each row. Therefore, the degree of the factors is \( 1 + 2 + \cdots + n - 1 = (n-1)n/2 \). If \( x_i = x_j \), \( \Delta = 0 \) because two columns are equal, so that \( x_j - x_i \) is a factor of \( \Delta \) for \( j \neq i \). The number of factors of the form \( x_j - x_i \) for \( j > i \) and \( j \neq i \) \( \cdots \) \( n - i \) is \( i + 2 + \cdots + (n-1) \). Thus, the product of these factors is a constant \( x \Delta \). Writing out the factors of this product,

\[ \Delta = -\frac{1}{\prod_{j=1}^{n} (x_j - x_i)} \left[ (x_2 - x_1)(x_3 - x_1) \right] \left[ (x_4 - x_3)(x_5 - x_3) \right] \cdots \\
\left[ (x_n - x_{n-1}) \cdots (x_n - x_i) \right] \\
= (x_2 - x_1)5x_2^2 + 75x_4^3 + \cdots \sum x_n^{n-1} \]

Therefore, one term in the expansion of the right hand side is \( x_2x_3 \cdots (\cdots) \cdots x_n^{n-1} \). This is just the
product of the diagonal terms of the Vandermonde matrix so that \( c = 1 \) \( \text{G.E.D.} \)

Applying the lemma

\[
\Delta = \prod_{i=0}^{d-1} \prod_{j=1}^{m} (\alpha_i^j - \alpha_j^i)
\]

Since \( 0 \leq i < j \leq m+1 \) over \( \alpha \), \( \alpha_i^j \neq \alpha_j^i \)

and \( \Delta \neq 0 \), if \( \Delta \neq 0 \) no nonzero linear combination
of \( d-1 \) columns of \( H \) with coefficients \( \in \text{GF}(q^n) \)
can be \( 0 \) and these columns are linearly independent.

Therefore \( d_{\min} \leq d \) \( \text{G.E.D.} \)

**Comment:**

It has only \( d-1 \) rows when considered as a matrix over \( \text{GF}(q^n) \) so that rank of \( H \leq d-1 \).

Thus any set of \( d \) columns must be linearly dependent over \( \text{GF}(q^n) \). However this does not imply that they are linearly dependent over \( \text{GF}(q) \)
since, in general, all that can be said is that \( d \leq d_{\min} \).

**Special Case:** \( n=1 \), Reed-Solomon Codes

The Reed-Solomon codes have roots and code symbols taken from \( \text{GF}(q) \).

The argument in the previous paragraph shows that there are codewords of weight \( d \) over \( \text{GF}(q) \) and Theorem 19 implies \( d_{\min} \geq d \).

Therefore \( d_{\min} = d \) for \( n=1 \), since \( \alpha_0, \alpha_{m+1}, \ldots, \alpha_{m+d-2} \)

are in \( GF(q) \), 
\[ g(x) = (x - x_0)(x - x_1) \cdots (x - x_d) \]

The degree of \( g(x) \) is \( d = N - k \), the number of check symbols. \( (d = 2t = N - k) \)

Lemma:

For any linear block code with \( N - k \) check symbols, \( d_{\min} \leq N - k + 1 \)

Proof:

It has rank \( N - k \) so that any set of \( N - k + 1 \) columns must be linearly dependent. This implies \( d_{\min} \leq N - k + 1 \). Q.E.D.

Therefore, the Reed-Solomon codes have the greatest possible minimum distance for a linear block code with \( N - k \) check symbols.

Example:

\( m = 1, m_0 = 1, q = 2^4, d = 5 \)

Let \( \alpha \) be a root of \( D^4 + D + 1 \), \( \Rightarrow N = 2^4 - 1 = 15 \)

Then, \( g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) \)

\[ = \alpha^4 + \alpha^3D^2 + \alpha^2D^2 + \alpha^3D + D^10 \]

Since \( \deg g = 4 \), \( k = 15 - 4 = 11 \) and this is a \((15,11)\) code over \( GF(2^4) \) that can correct 2 or less errors. A single error occurs when the \( GF(q) \) received symbol is not equal to the \( GF(q) \) transmitted symbol. The code symbols are taken from the alphabet of \( 16 \) \( GF(2^4) \) elements.

In practice, the \( GF(2^4) \) elements can be represented as binary 4-tuples. The codes are then
error codes with \( N = 4 \times 15 \) and \( k = 4 \times 11 \).

Notice that 1, 2, 3 or 4 binary bit errors in the 4-tuple representing the GF(2^4) code symbol is not as only one GF(2^4) symbol error.

Def: Burst of length \( b \)

A burst of length \( b \) is a sequence of \( b \) consecutive symbols starting and ending with a nonzero field element.

Consider the Reed-Solomon codes over GF(2^m).

If the code symbols are transmitted as binary m-tuples, a burst error of \( b = (t+1)m + 1 \) channel symbols can affect at most \( t \) GF(2^m) symbols. If \( d \geq 2t+1 \) the code can correct channel burst of length \( b = (t+1)m + 1 \).

Error Correction for BCH Codes

Ref: Salagean - Ch 6, Berlekamp, Peterson

For BCH codes \( d \geq m \geq d \) so that all patterns of \( \left[ \frac{d-1}{2} \right] \) or fewer errors can be corrected.

Except for perfect codes, usually some error patterns of greater weight can be corrected. "No practical algorithm is known to accomplish this."

Note: With advances in VLSI, syndrome decoding tables are often a good approach.

Let \( X(d) \) be the transmitted codeword, \( y(d) \) the received word, and \( E(d) = y(d) - X(d) \) be the channel error sequence. Let the syndrome \( \bar{s} \) be defined as \( \bar{s} = [\bar{y} H] \), where
Thus, $\tilde{s} = [s_0, \ldots, s_{d-2}]$.

where $s_i = y(x^{m_0+i})$ for $0 \leq i < d-2$

$= E(x^{m_0+i})$

Suppose that $t \leq \left\lceil \frac{d-1}{2} \right\rceil$ errors have occurred.

So that $E(D) = e_0^{\overline{n_1}} + e_1^{\overline{n_2}} + \ldots + e_t^{\overline{n_t}}$. Then

$$s_i = \sum_{k=1}^{t} e_k^{(x^{m_0+i})^{n_k}} = \sum_{k=1}^{t} e_k^{(x^n)^{n_{m_0+i}}}$$

Let $v_k = e_k^{\frac{n_k}{n}}$ be the error value and $u = x^{n_k}$ be the error locator. The $u_k$ are all distinct since $0 \leq n_1 < n_2 < \ldots < n_t < N-1 = (\text{ord}_{\alpha}-1)$ for $d \geq 3$.

Then

$$s_i = \sum_{k=1}^{t} v_k u_k^{n_{m_0+i}} \quad \text{for } i = 0, \ldots, d-2 \quad (1)$$

calculation of $\tilde{s}$

Let the minimal polynomial for $x^{m_0+i}$ be $m(x)$. Then by the division algorithm

$$y(x) = q(x)m(x) + r(x) \quad \text{for } i = 0, \ldots, d-2$$

and $y(x^{m_0+i}) = r(x^{m_0+i}) = s_i$. 


$y_0, y_1, \ldots, y_{m-1}$

$y_0, y_1, \ldots, y_{m-1}$

$y_0, y_1, \ldots, y_{m-1}$

If $a$ is a root of $f(D) = f_0 + \cdots + f_m D^m$

The circuit below calculates $f_2(a^{m+2})$.

The coefficient $f_{2i}$ is shifted into the register

and then the register is shifted $m_{0+2}$ times

at this point $f_{2i+2}$ is in the register

On the next shift $f_{2i+1}$ is added into the register

and the contents become $f_{2i} + f_{2i+1} a^{m_{0+2}}$

On every $m_{0+2}$ shifts a new coefficient is shifted

into the register so that its final contents are

$r_2(a^{m+2})$. Since $[r(D)]^2 = r(D^2)$, the circuits

shown above can be used to calculate several

of the $r_i$'s by shifting the lower register

the appropriate number of times between

entering coefficients, alternately $y(D^{m_{0+2}})$ can

be shifted into the lower register for

$i = 0, \ldots, d-2$. This usually requires more computation.
\( r(2^{m+1}) \) can also be calculated instantaneously from the coefficients \( \{ r_0, \ldots, r_{M-3} \} \). Any element \( C \in GF(2^m) \) can be represented as a linear combination of \( 1, x, \ldots, x^{m-1} \). Therefore \( x = \begin{bmatrix} 1 & x & \cdots & x^{m-1} \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} \). \\

\[
\begin{bmatrix}
1 & x & \cdots & x^{m-1} \\
0 & c_1 & \cdots & c_m \\
\vdots & \vdots & & \vdots \\
0 & c_{m-1} & \cdots & c_{m-1}
\end{bmatrix}
\]

Therefore, in terms of the \( m \)-tuple representation of \( GF(2^m) \),

\[
s_j = r(2^{m+1}) = CR^{n \times 2}
\]

In the binary case, the elements of \( C \) are just 1's and 0's, so that the coefficients are either connected or not connected to the appropriate adders.

The decoding problem, given \( s_0, \ldots, s_{d-2} \), is to solve Eq. 1 for the error locators \( \{ u_0, \ldots, u_{d-2} \} \) and error values \( \{ e_0, \ldots, e_{d-2} \} \). Eq. 1 is a set of \( d-1 \) nonlinear equations in the \( 2^t \) unknowns \( \{ v_0, \ldots, v_{d-2} \} \). A brute force solution is to substitute all combinations of \( v_0, \ldots, v_{d-2} \) and choose the most likely solution. This normally involves excessive calculation. The approach followed here is a modification of
A method proposed by Berlekamp. Let

\[ s(D) = \sum_{i=0}^{\infty} s_i D^i \]

where

\[ s_i = \sum_{j=1}^{t} V_j U_j^{m_0+i} \]

Then

\[ s(D) = \sum_{i=0}^{\infty} s_i D^i = \sum_{i=1}^{\infty} s_i D^i \]

\[ s(D) = \sum_{j=1}^{t} V_j U_j^{m_0} \frac{1}{1 - U_j D} \]

(2)

Define the error locator polynomial to be

\[ \sigma(D) = \frac{1}{\Pi_{D} (1 - U_j D)} = \sigma_0 + \sigma_1 D + \ldots + \sigma_t D^t \]

(3)

Note: \( \sigma_0 = 1 \)

The roots of \( \sigma(D) \) are the inverses of the error locators, i.e., \( \{ U_j^{-1} \} \). Now

\[ s(D) s(D) = \sum_{j=1}^{t} V_j U_j^{m_0} \Pi_{l \neq j} (1 - U_l D) = A(D) \]

(4)

Let

\[ [B(D)]^j = \begin{cases} \sum_{k=0}^{l - j} b_k D^k & j \leq l \\ 0 & j > l \end{cases} \]

If \( \deg B(D) = l \), then \( b_{l-1}, \ldots, b_0 \) will be taken as 0.

Let

\[ s(D) = s_0 + s_1 D + \ldots + s_t D^t \]

Then

\[ s(D) s(D) = \left[ s(D) \right]^{d-2} \] using the notation

\[ \left[ s(D) s(D) \right]^{d-2} = \left[ s(D) s(D) + \sum_{i=1}^{d-1} \sigma(D) s(D) \right]^{d-2} \]

\[ = \left[ \sigma(D) s(D) \right]^{d-2} = \left[ A(D) \right]^{d-2} \]

From Eq. 4 it can be seen that \( \deg A(D) = t - 1 < d - 2 \)

and

\[ \left[ A(D) \right]^{d-2} = \left[ \sigma(D) s(D) \right]^{d-2} = 0 \]
This is equivalent to the following set of \( d-t-1 \) equations:

\[
\begin{align*}
\sigma_0 s_0 + \sigma_1 s_t + \cdots + \sigma_t s_0 &= 0 \\
\sigma_0 s_{t+1} + \sigma_1 s_t + \cdots + \sigma_t s_1 &= 0 \\
&\vdots \\
\sigma_0 s_{d-1} + \sigma_1 s_{d-2} + \cdots + \sigma_t s_{d-2t} &= 0
\end{align*}
\]

From Eq. 3 it can be seen that \( \sigma_0 = 1 \) so that there is a set of \( d-t-1 \) linear equations in \( t \) unknowns. If a solution exists then the error locator can be found from the roots of \( \hat{\sigma}(D) \).

The decoder does not know the number of channel errors \( t \). The decoder can attempt to solve Eq. 5 for different \( t \). For \( t \leq \lfloor \frac{d-1}{2} \rfloor \), the following theorem shows that the correct solution is found.

**Theorem 20**

Let \( t \leq \lfloor \frac{d-1}{2} \rfloor \) and \( \sigma(D) = \frac{\hat{\sigma}}{\hat{d}} \left( 1 - \hat{d}D \right) \).

Let \( \hat{d} \) be the smallest integer for which a polynomial \( \hat{\sigma}(D) = 1 + \hat{\sigma}_1 D + \cdots + \hat{\sigma}_{\hat{d}} D^{\hat{d}} \) exists satisfying

\[
\left[ \hat{\sigma}(D) \hat{\sigma}(D) \right]_{\hat{d}+2} = 0
\]

Then \( t = \hat{d} \) and \( \sigma(D) = \hat{\sigma}(D) \).

**Proof:**

\[
\left[ \hat{\sigma}(D) \hat{\sigma}(D) \right]_{\hat{d}+2} = 0 \quad \text{is equivalent to}
\]

\[
\sum_{t=0}^{\hat{d}} \hat{\sigma}_t s_{t-1} = 0 \quad \text{for } t \leq \hat{d} \leq d-2
\]
\[ \sum_{l=0}^{t} \sigma \sum_{j=1}^{t} V_j U_{j+l}^0 = 0 \]

\[ = \sum_{j=1}^{t} V_j \sigma (U_j^{-1}) U_j^{m_0+t-l} = 0 \quad \text{if} \quad 2t \leq d-1 \]

This is a set of \( d-1-\hat{t} \) equations in the \( \hat{t} \) unknowns \( V_j \sigma (U_j^{-1}) \) for \( j = 1, \ldots, \hat{t} \). By assumption \( \hat{t} \) is the smallest integer such that

\[ [1_{d-\hat{t}} (d-\hat{t})]_{d-\hat{t}} = 0 \]

so that \( \hat{t} \leq t \). If \( \hat{t} \leq \lfloor \frac{d-1}{2} \rfloor \), then \( 2t \leq d-1 \) or \( t \leq d-1 - \hat{t} \), consider only the first \( t \) equations in the set above:

\[
\begin{bmatrix}
U_1^{m_0+t} & U_1^{m_0+t} & \cdots & U_1^{m_0+t} \\
U_1^{m_0+t+1} & U_2 & \cdots & U_2^{m_0+t+1} \\
\vdots & \vdots & \ddots & \vdots \\
U_1^{m_0+t+\hat{t}-1} & \cdots & U_{\hat{t}}^{m_0+t+\hat{t}-1} & U_{\hat{t}}^{m_0+t+\hat{t}} \\
\end{bmatrix}
= 
\begin{bmatrix}
V_1 \sigma (U_1^{-1}) \\
V_2 \sigma (U_2^{-1}) \\
\vdots \\
V_{\hat{t}} \sigma (U_{\hat{t}}^{-1}) \end{bmatrix}
\]

By the same argument as in Theorem 19, let \( V \neq 0 \) and the only solution is \( V_j \sigma (U_j^{-1}) = 0 \) for \( j = 1, \ldots, t \). The error values \( \overline{V}_j \) are non-zero so that

\[ \sigma (U_j^{-1}) = 0 \quad j = 1, \ldots, t \]

The degree \( \hat{t} \leq t \) and \( \sigma \) has the same roots as \( \sigma \).
with \( \hat{s}_0 = s_0 = 1 \) and thus \( \sigma(0) = \hat{\sigma}(0) \)

The error location polynomial \( \sigma(0) \) can be found by solving Eq's 5 for the smallest possible value of \( t \). This can be done using standard methods for solving sets of linear equations. Berlekamp has discovered an efficient iterative algorithm for finding the solution which will be discussed shortly.

Finding Error Values

From Eq 4

\[
A(0) = \left[ \sigma(0) \sigma(0) \right]_{\tau=1}^{d-2} = \sum_{j=1}^{m_\tau} \prod_{k=1}^{\tau} (1 - U_j D)
\]

The derivative of \( \sigma(0) = \sigma_1 + \sigma_2 D + \ldots + \sigma_\tau D^\tau \) is defined as \( \sigma'(0) = \sigma_1 + 2 \sigma_2 D + \ldots + \tau \sigma_\tau D^{\tau-1} \). Also

\[
\sigma(0) = \frac{1}{\prod_{j=1}^{\tau} (1 - U_j D)} \quad \text{and it can be shown that}
\]

\[
\sigma'(0) = -\sum_{j=1}^{\tau} U_j \prod_{k=1}^{\tau} (1 - U_j D)
\]

Now

\[
A(U_k^{-1}) = \sum_{j=1}^{m_\tau} \prod_{k=1}^{\tau} (1 - U_k U_k^{-1})
\]

and

\[
\sigma'(U_k^{-1}) = -U_k \prod_{k=1}^{\tau} (1 - U_k U_k^{-1})
\]

So

\[
U_k = -U_k^{-1} m_\tau A(U_k^{-1}) \quad \text{for} \ k = 1, \ldots, t
\]

\[
\sigma'(U_k^{-1})
\]
Having found \( \sigma(0) \) by solving Eq's 5 for \( \sigma_0, \ldots, \sigma_t \) it is necessary to factor \( \sigma(0) \) to find the error locations. The roots of \( \sigma(0) \) can be found by substituting sequentially \( \alpha, \alpha^2, \ldots, \alpha^{N-1} \) for D and checking to see if \( \sigma(\alpha^k) = 0 \). If \( \sigma(\alpha^k) = 0 \) then the error locator is \( \nu_k = \alpha^{N-k} \) so that the code symbol corresponding to \( \alpha^{N-k} \) is in error.

For the binary case \( \nu_{N-K} = 1 \). For higher order alphabets \( \nu_{N-K} \) is given by Eq 6. Thus \( \sigma(\alpha^k) \) checks for an error in \( y_{N-1} \), \( \sigma(\alpha^2) \) for an error in \( y_{N-2} \), \ldots , \( \sigma(\alpha^k) \) for an error in \( y_0 \).

A circuit for performing the tests is shown below.

The registers are initially loaded with \( \sigma_1, \ldots, \sigma_{N-1} \) and shifted once. The contents then sum to
1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_t x^t = \sigma(x^k) \text{ and on the}

\text{ith shift the contents sum to } \sigma(x^k).

\textbf{General Decoder Block Diagram (Berlekamp)}

\[ y(d) \]

\[ y(d) \bmod f_i(d) \]

\[ y(d) \bmod f_i(d) \]

\[ y(d) \bmod f_i(d) \]

\[ y(d) \bmod f_i(d) \]

[Diagram: Buffer, 2N, D, Calculate, Chien Searcher, \sigma(0), \rho(0), Central Processor, etc.]

\[ g(d) = \prod_{i=1}^{k} f_i(d) \]

at a typical time the buffer contains part

of 3 blocks.

\[ \begin{array}{c}
\text{Incoming} \\
\text{Buffered} \\
\text{Outgoing}
\end{array} \]

The Chien search is calculating \( \sigma(x^k) \) to
determine if the next outgoing symbol should be
corrected. The central processor is calculating
\( \sigma(0) \) for the buffered block, and the input
registers are calculating \( y(d) \bmod f_i(d) \) for the
incoming word.