Lecture Notes for Math 696
Coding Theory

Michael E. O'Sullivan
`mosulliv@math.sdsu.edu`
`wwww-rohan.sdsu.edu/ ˜ mosulliv`

January 7, 2004

Reed-Solomon Codes and the Berlekamp-Massey Algorithm

1 Codes

Let F be a field and let F^n be a vector space of dimension n over F. We will write elements of F^n as row vectors. We define the *Hamming distance* function D on F^n by

$$D(x,y) = \#\{i : x_i \neq y_i\}$$

It is straightforward to prove that D is a metric (do it!); that is

$$\begin{aligned} D(x,y) &\geq 0 \quad \text{with equality iff } x = y \\ D(x,y) &= D(y,x) \\ D(x,y) + D(y,z) &\geq D(x,z) \end{aligned}$$

The *Hamming weight* of a vector is the number of nonzero positions in x; so $\mathrm{wt}(x) = D(x,0)$. The *support* of x is the set of indices i for which x_i is nonzero.

An $[n,k,d]$ *linear code* C is a vector subspace of F^n of dimension k whose minimum distance is d. The *minimum distance* is defined by $d = \min\{\mathrm{wt}(c) : c \in C \text{ and } c \neq 0\}$. We will usually say that C is a code and not mention the linearity, although it will always be assumed. The *dual code* of C is the set of vectors orthogonal to all vectors in C,

$$C^\perp = \{v : v \cdot c = 0 \text{ for all } c \in C\}$$

It is also a vector space.

An $[n,k,d]$ code C can be specified by a $k \times n$ *generator matrix* G. Let the rows of G be a basis for C. Then $C = \{uG : u \in F^k\}$. We can also specify C using a $n \times (n-k)$ *parity check matrix* H. Let the columns of H be a basis for $C^\perp$. Then $C = \{c \in F^n : cH = 0\}$. If G is a generator matrix for C and H is a check matrix for C then H^T is a generator matrix for $C^\perp$ and G^T is a check matrix for $C^\perp$.

Since the parity check matrix has $n-k$ columns, any $(n-k+1)$ rows of H are linearly dependent. Thus for any choice of a subset $I \subseteq \{1,\ldots,n\}$ with $(n-k+1)$ elements there is a codeword c whose support is contained in I. In particular, the minimum distance of C is at most $n-k+1$. We have just derived the *Singleton bound*:

Proposition 1.1. *Let C be an $[n,k,d]$ linear code. $d \leq n-k+1$.*

Definition 1.2. A code meeting the Singleton bound is called an *maximum distance separable* or MDS code.

Using a code for error correction

Let C be an $[n,k,d]$ code over $\mathbb{F}_q$ with generator matrix G and check matrix H. I will sketch here a standard protocol for using C for error correction.

- We assume that information is presented to the *sender* as packets of k elements from $\mathbb{F}_q$. This is treated as a vector in $\mathbb{F}_q^k$.

- *Encoding* adds redundancy: $v = uG$ is in $\mathbb{F}_q^n$.
- Vector v is sent to the *receiver.*
- *Error* in the transmission channel is modeled as a vector $e \in \mathbb{F}_q^n$. The receiver gets $w = v + e$.
- *Decoding* is the attempt to recover v. Given w find the closest vector $c \in C$ to w.
- Once v is found, matrix inversion of some $k \times k$ submatrix of G will produce u.

In practice, the final matrix inversion is avoided by using *systematic* encoding. This simply means that we choose a generator matrix which has a $k \times k$ submatrix which is the identity matrix (or some permutation of it).

As for decoding, the situation we explore in this chapter is the following. An efficient algorithm is employed that correctly decodes all vectors of weight less than $d/2$. For errors of larger weight the algorithm may correctly decode (but this is rare) or it may decode incorrectly, or it may fail to produce a vector. The system designer usually has some idea, based on the electronics and experience, about the likelihood p_s of a symbol being in error. The code is chosen so that $d/2$ is "comfortably" larger than the expected number of symbol errors np_s. The likelihood of more than t error occuring is then

$$\sum_{i=\lceil d/2 \rceil}^{n} \binom{n}{i} p_s^i (1 - p_s)^{n-1}$$

This is an upper bound on the probability of not decoding correctly. Provided p_s is reasonably low, by choosing t large enough this can be made very small.

2 Reed-Solomon Codes

In this section I will give three different definitions of Reed-Solomon codes. We (well, you) will show that the definitions are equivalent. The reason to introduce all three perspectives is that each is useful for a purpose which the others do not satisfy as well. One gives a direct description of the codewords, another is useful for decoding and the third is useful for encoding. Each method also has its own utility in constructing broader families of codes.

Throughout this section we will work over the field $\mathbb{F}_q$ with α a primitive element of $\mathbb{F}_q$. The length of the code will always be $n = q - 1$ and k will designate the dimension of the code. It will be useful to think of an element $v \in \mathbb{F}^n$ as indexed from 0 to $n - 1$, $v = (v_0, \ldots, v_{n-1})$.

Reed-Solomon codes as evaluation codes

Let $RS_G(k)$ be the code with $k \times n$ generator matrix

$$G = \begin{bmatrix} 1 & 1 & 1 & 1\dots & \dots & \dots & \dots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \dots & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \dots & \dots & \dots & \alpha^{n-2} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \\ 1 & \alpha^{k-1} & \alpha^{2k-2} & \alpha^{3k-3} & \dots & \dots & \dots & \alpha^{n-k+1} \end{bmatrix} \tag{1}$$

More precisely, the matrix entries are $G_{ij} = \alpha^{ij}$ for $i = 0, \dots, k-1$ and $j = 0, \dots, n-1$

It is useful to think of this code as being obtained by evaluating polynomials at the field elements. Consider the non-zero field elements enumerated as α^j for j from 0 to $n-1$. The ith row of the G may be obtained by evaluating the polynomial x^i at the field elements α^j. The code itself can be described independently of the generator matrix as

$$RS_G(k) = \{(f(\alpha^0), f(\alpha^1), f(\alpha^2), \dots, f(\alpha^{n-1}))\} : f \in \mathbb{F}_q[x] \text{ and } \deg f < k\}$$

This gives a nice explict description of the codewords.

Reed-Solomon codes as duals of evaluation codes

Let $RS_H(k)$ be the code with check matrix

$$H = \begin{bmatrix} 1 & 1 & 1 & \dots & \dots & 1 \\ \alpha & \alpha^2 & \alpha^3 & \dots & \dots & \alpha^{n-k} \\ \alpha^2 & \alpha^4 & \alpha^6 & \dots & \dots & \alpha^{2n-2k} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \alpha^{n-1} & \alpha^{n-2} & \alpha^{n-3} & \dots & \dots & \alpha^k \end{bmatrix} \tag{2}$$

Here we have constructed the jth column of the check matrix for $j = 1, \dots, k$ by evaluating the monomial x^j at the field elements α^i for $i = 0, 1, \dots, n-1$. Notice the that H is not quite the transpose of the generator matrix for $RS_G(n-k)$.

One can now prove that the minimum distance of $RS_H(k)$ meets the Singleton bound, $n-k+1$. One simply checks that any $(n-k) \times (n-k)$ submatrix of H has nonzero determinant. This means that any $(n-k)$ rows of H are linearly independent. Therefore there can be no nonzero codeword whose support has $n-k$ or fewer positions.

In addition to establishing the minimum distance, this formulation of Reed-Solomon codes is useful for decoding.

Exercises 2.1.

1. Prove that every $(n-k) \times (n-k)$ submatrix of H has nonzero determinant. (Compare with a Vandermonde matrix.)

Reed-Solomon codes as cyclic codes

In this description of Reed-Solomon codes we use an association between vectors $(v_0, v_1, \ldots, v_{n-1})$ and polynomials $v_0 + v_1 x + v_2 x^2 + v_{n-1} x^{n-1}$.

$$RS_C(k) = \{(c_0, c_1, \ldots, c_{n-1}) : \sum_{j=0}^{n-1} c_j x^j \text{ vanishes at } \alpha, \alpha^2, \ldots, \alpha^{n-k}\}$$

This formulation of Reed-Solomon codes is useful for encoding.

You can prove the following theorem as an exercise. We computed the minimum distance in the section on RS_H.

Theorem 2.2. $RS_G(k) = RS_H(k) = RS_C(k)$. *The minimum distance of a* $RS(k)$ *meets the Singleton bound* $n - k + 1$.

From now on we will just write $RS(k)$ for the Reed-Solomon code of dimension k

Exercises 2.3.

1. Write Maple code to construct the matrices G and H for $RS(k)$ and check that the product is a matrix of zeros.

2. Prove the theorem. Show first that G and H are full rank, and then that GH is a $k \times n - k$ matrix of zeros. This shows that $RS_G = RS_H$ Show directly from the definition that $c \in RS_C(k)$ if and only if $c.H = 0$.

3 Cyclic Codes and Systematic Encoding

Let F be a field and let $R = F[x]/(x^n - 1)$. In this section we show the connection between cyclic codes of length n over F and ideals of R. We will show that there is a simple algorithm for encoding cyclic codes. We will also discuss the dual code to a cyclic code.

Definition 3.1. Let τ be the cyclic shift operator,

$$\tau(c_0, c_1, c_2, \ldots, c_{n-1}) = (c_{n-1}, c_0, c_1, c_2, \ldots, c_{n-2})$$

A code of length n over F is called *cyclic* if $\tau(c) \in C$ whenever $c \in C$.

Consider the ring $R = F[x]/(x^n - 1)$. We know from the section on polynomial rings that the set of polynomials of degree less than n is a system of representatives for R and that equivalence classes of $1, x, x^2, \ldots, x^{n-1}$ form a basis for R as a vector space over F. With respect to this basis, we can write an arbitrary element $a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ in R as a vector $(a_0, a_1, \ldots, a_{n-1})$. The multiplicative structure of R translates nicely to vector notation because multiplication by x corresponds to cyclic shift,

$$x(a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}) \equiv a_{n-1} + a_0 x + a_1 x^2 + \cdots + a_{n-2} x^{n-1} \mod x^n - 1$$

We also showed in an exercise in §**??** that any ideal in R is generated by some factor $g(x)$ of x^{n-1}.

Proposition 3.2. *Let $g(x)$ divide $x^n - 1$. Then the dimension of the ideal $< g(x) >$ in $R = F[x]/(x^n - 1)$ as a vector space over F is $n - \deg g(x)$.*

Proof: Let $g(x)h(x) = x^n - 1$ and let $d = \deg g(x)$. Since $1, x^2, \ldots, x^{n-1}$ generate R, any element of $< g(x) >$, is a linear combination of $x^i g(x)$ for $i = 0, \ldots, n-1$. I claim that $g(x), xg(x), \ldots, x^{n-d-1}g(x)$ are linearly independent in R. This is equivalent to showing that no nontrivial linear combination of them is divisible by $x^n - 1$. But, $\sum_{i=0}^{n-d-1} a_i x^i g(x)$, has degree $d + \max\{i : a_i \neq 0\}$, which is at most $n - 1$. Thus $\sum_{i=0}^{n-d-1} a_i x^i g(x)$ cannot be divisible by $x^n - 1$.

On the other hand, let $h(x) = x^{n-d} + \sum_{i=0}^{n-d-1} h_i x^i$. Since $g(x)h(x) = x^n - 1$,

$$x^{n-d}g(x) \equiv -\sum_{i=1}^{n-d-1} h_i x^i g(x) \qquad \bmod (x^n - 1)$$

Thus $x^{n-d}g(x)$ is linearly dependent, modulo $x^n - 1$, on $g(x), xg(x), \ldots, x^{n-d-1}g(x)$. By similar methods, for $r > n - d$ one can show that $x^r g(x)$ is equivalent modulo $x^n - 1$ to a sum of $x^i g(x)$ $i = 0, 1, \ldots, n-d-1$. Thus $x^i g(x)$ $i = 0, 1, \ldots, n-d-1$ is a basis for $< g(x) >$. □

Theorem 3.3. *Let C be a cyclic code of length n and dimension k over F. The set of polynomials $I = \{c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1}x^{n-1} : (c_0, c_1, \ldots, c_{n-1}) \in C\}$ is an ideal of $R = F[x]/(x^n - 1)$. There is a unique codeword $(g_0, g_1, \ldots, g_{n-k}, 0, 0, \ldots, 0)$ in C with $g_{n-k} = 1$ such that a basis for C is $g, \tau g, \tau^2 g, \ldots, \tau^{k-1} g$.*

Conversely, any ideal of $F[x]/(x^n - 1)$ gives rise to a cyclic code, and distinct ideals give distinct codes.

Proof: Since C is closed under addition, the set I is also closed under addition. Since C is closed under multiplication by an element of F, so is I. Since C is closed under cyclic shift, τ, I is closed under multiplication by x. Now let $a_0 + a_i + \ldots, + a_{n-1}x^{n-1}$ be an arbitrary element of R and let $c(x)$ in I, we seek to prove that their product is in I. Since I is closed under multiplication by x and by elements of F, $a_i x^i c(x) \in I$. Since I is closed under addition $(\sum_{i=0}^{n-1} a_i x^i)c(x) \in I$. This shows that I is closed under multiplication by an arbitrary element of R, and is therefore an ideal of R.

By an exercise in §**??** any ideal of $F[x]/(x^n - 1)$ is generated by a factor of $x^n - 1$. Let $g(x)$ be the generator for I. The previous proposition says that the dimension of I is $n - \deg g(x)$. Since the map from C to I is one-to-one, and C has dimension k we have $k = n - \deg g(x)$ so $\deg g(x) = n - k$. Furthermore the proof of the proposition shows that $x^i g(x)$ for $i = 0 \ldots n - d - 1$ is a basis for I. Thus $g, \tau g, \tau^2 g, \ldots, \tau^{k-1} g$ is a basis for C.

The proof that any ideal gives a cyclic code should be clear. □

The polynomial $g(x)$ in the theorem is called the *generator polynomial* for C. We will also call any constant multiple of $g(x)$ a generator polynomial. One generator matrix for C is

$$\begin{bmatrix} g_0 & g_1 & g_2 & \dots & \dots & \dots & \dots & g_{n-k} & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & g_0 & g_1 & g_2 & \dots & \dots & \dots & \dots & g_{n-k} & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \dots & \dots & \dots & \dots & g_{n-k} & 0 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \dots & g_0 & g_1 & g_2 & \dots & \dots & \dots & \dots & g_{n-k} \end{bmatrix} \tag{3}$$

Encoding

Let C be a cyclic code and suppose that we want to encode the message vector $u = (u_0, \dots, u_{k-1})$. We can multiply u by G to get a codeword, $v = uG$ and send v. Let $u(x) = \sum_{i=0}^{k-1} u_i x^i$. You should check that the components of v are, in appropriate order, the coefficients of the polynomial product $u(x)g(x)$. Using electronic circuitry it is much simpler to compute the polynomial product than to compute an arbitrary matrix product. See Blahut Ch. 6 for numerous approaches to hardware implementation.

The product $u(x)g(x)$ has the disadvantage of not being a systematic encoding of $u(x)$. Fortunately there does exist a method for systematic encoding that is just as simple. Let $q(x)$ and $r(x)$ be the quotient and remainder when $x^{n-k}u(x)$ is divided by $g(x)$. Then

$$x^{n-k}u(x) - r(x) = q(x)g(x) \tag{4}$$

The left-hand side of the equation is

$$u_{k-1}x^{n-1} + u_{k-2}x^{n-2} + \cdots + u_1 x^{n-k+1} + u_0 x^{n-k} + r_{n-k-1}x^{n-k-1} + \cdots + r_1 x + r_0$$

Furthermore since the LHS is a multiple of $g(x)$ the associated vector is a codeword of C. So this polynomial division gives a systematic encoder. The circuitry for polynomial division is similar to polynomial multiplication and can be found in Blahut.

Refer back to the section on Reed-Solomon codes for the description of $RS_C(k)$ over the field $\mathbb{F}_q$ with α a primitive element. You should see that $RS_C(k)$ is a cyclic code of length $n = q - 1$ with generator polynomial $g(x) = \prod_{i=1}^{n-k}(x - \alpha^i)$. We can therefore systematically encode a Reed-Solomon code using polynomial division as described above.

The dual code of a cyclic code

Let C be a cyclic code of length n over a field F and let $g(x)$ be the generator polynomial for C. We know $g(x)$ is a factor of $x^n - 1$, so let us suppose that $g(x)h(x) = x^n - 1$. Let

$h(x) = h_0 + h_1 x + h_2 x^2 + \cdots + h_k x^k$, and consider the matrix

$$H = \begin{bmatrix} h_k & 0 & 0 & 0 & \dots & \dots & \dots & 0 & 0 \\ h_{k-1} & h_k & 0 & 0 & \dots & \dots & \dots & 0 & 0 \\ h_{k-2} & h_{k-1} & h_k & 0 & \dots & \dots & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ h_0 & h_1 & h_2 & h_3 & \dots & \dots & \dots & \dots & \dots \\ 0 & h_0 & h_1 & h_2 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & h_0 & h_1 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & h_0 & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \dots & \dots & h_0 & h_1 \\ 0 & 0 & 0 & 0 & \dots & \dots & \dots & 0 & h_0 \end{bmatrix} \tag{5}$$

Computing the product of G from (3) and H one finds that the product of the ith row of G and the jth column of H ($i = 0, \dots, k-1$, $j = 0, \dots, n-k-1$) is the coefficient of x^{k+j-i} in the product of $g(x)$ and $h(x)$ (see exercise below). Since $g(x)$ and $h(x)$ were chosen to have product $x^n - 1$, as we let i and j wander over the ranges given we always get 0. This is because $k + j - i$ is minimized when $i = k - 1$ and $j = 0$ for which we get $k + j - i = 1$ and it is maximized when $i = 0$ and $j = n - k - 1$ for which we get $k + j - i = n - 1$. Clearly all the coefficients of $x^n - 1$ for powers of x between 1 and $n - 1$ are indeed 0.

Notice that the transpose of H is not in the same form as G, but is quite similar. In fact, if we define a polynomial $\overline{h}(x) = h_0 x^k + h_1 x^{k-1} + \cdots + h_{k-1} x + h_k$, then H^T coincides with the matrix for the generating polynomial $\overline{h}(x)$. This would seem to imply that $\overline{h}(x)$ is a factor of $x^n - 1$. Well, it is. Notice that $\overline{h}(x) = x^k h(1/x)$. Let $\overline{g}(x) = x^{n-k} g(1/x)$. Then

$$\begin{aligned} x^n - 1 &= -x^n((1/x)^n - 1) \\ &= -x^n(g(1/x)h(1/x)) \\ &= -(x^{n-k} g(1/x))(x^k h(1/x)) \\ &= -\overline{g}(x)\overline{h}(x) \end{aligned}$$

So $\overline{h}(x)$ is a factor of $x^n - 1$, and it is therefore a generator polynomial of a cyclic code.

Definition 3.4. Let $g(x)$ be a factor of $x^n - 1$ over the field F, and let $d = \deg g(x)$. The *reciprocal polynomial* of $g(x)$ (relative to $x^n - 1$) is $\overline{g}(x) = x^d g(1/x)$.

Proposition 3.5. *Let $g(x)h(x) = x^n - 1$ over some field F. The dual of the cylic code of length n with generator polynomial $g(x)$ is cyclic with generator polynomial $\overline{h}(x)$, the reciprocal polynomial of $h(x)$.*

Exercises 3.6.

1. Identify all cyclic codes of length n over $\mathbb{F}_2$ for several values of n in the range $n = 5, \dots, 20$. For each n make a table showing the possible generator polynomials,

the dimension of the code, and the polynomial generating the dual code. You can write the generator in factored form with variable exponents. How many distinct codes are there altogether?

2. Do the same over $\mathbb{F}_4$
3. Do the same over $\mathbb{F}_3$.
4. Justify the claim in the subsection on dual codes that the product of the ith row of G and the jth column of H is the coefficient of x^{k+j-i} in the product of $g(x)$ and $h(x)$.
5. Explain why $RS_C(k)$ is a cyclic code. In the sections on decoding we will use the code $RS(n-k)^{\perp}$. What is the generator polynomial for $RS(n-k)^{\perp}$?
6. Let $\alpha \in \mathbb{F}_9$ satisfy $\alpha^2 = \alpha + 1$. We will use the code $RS(n-k)^{\perp}$ which has dimension 5. Use Maple and exercise 5) to compute the generator polynomial. Systematically encode $x^2 + 1$ for this code.
7. Implement systematic encoding for Reed-Solomon codes in Maple.

4 Decoding of Reed-Solomon codes

Decoding can be broken down into 6 steps:

1. Compute the syndrome;
2. Compute the error-locator polynomial;
3. Compute the error-locations;
4. Compute the error values;
5. Compute the codeword;
6. Compute the information vector.

Let C be a Reed-Solomon code of dimension k and length $n = q - 1$ over $\mathbb{F}_q$. The generator matrix G, is assumed to be systematic. Let

u be the information vector;

$v = uG$ the codeword;

e the error vector;

$w = v + e$ the received vector.

Since G is systematic, some k components of v are the components of u. We call these the *information symbols*, the other $n-k$ components are called the *check symbols*. Clearly step 6) in the algorithm is trivial. We simply strip the check symbols from a codeword and we are left with the information symbols. Step 5) is also simple, we subtract the output of step 4), the error vector e, from the received vector w to get $v = w - e$. Step 1) is a straight forward computation. The *syndrome* of w is the vector $s = wH$. It is independent of the codeword that was sent, since $cH = 0$ for any codeword c. If the syndrome is 0 the received word is a codeword and we presume that it equals the codeword that was sent. The algorithm proceeds to step 6).

Computation of s is actually easier than a general matrix multiplication because of the nice structure of H in equation (2). If we interpret w as a polynomial, then wH is the row vector $(w(1), w(\alpha), w(\alpha^2), \ldots, w(\alpha^{k-1}))$. For each column of H, the polynomial evaluation can be done efficiently with Horner's method (see the exercise below). In hardware this is implemented with an adder and a single constant multiplier (α^j for the jth column).

To summarize, steps 1), 5), and 6) of decoding are fairly simple. Steps 2), 3), and 4) are more complex, enough so that each will be treated in its own section.

Exercises 4.1.

1. Let $f(x)$ be a polynomial of degree m and let α be a constant. Horner's method for the computation of $f(\alpha)$ is iterative

$$\begin{aligned} c_0 &= f_m \\ c_i &= f_{m-i} + c_{i-1} * \alpha \end{aligned}$$

   Prove inductively that $c_k = \sum_{i=m-k}^{m} f_i \alpha^{i-m+k}$. Conclude that $c_m = f(\alpha)$.

2. Implementation of Horner's method can be done with the following circuit. Verify that it works.

5 The Error Locator Polynomial

It will be convenient to make the following subtle change to our definition of Reed-Solomon codes. Recall that the generator matrix for $RS(k)$ and the check matrix for $RS(n-k)$ were almost transposes of each other. They differ only in the first and last rows. We will now switch the roles. Henceforth, we use the parity-check matrix

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & \dots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \dots & \alpha^{n-k-1} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \dots & \dots & \alpha^{2n-2k-2} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \alpha^{n-1} & \alpha^{n-2} & \alpha^{n-3} & \dots & \dots & \alpha^{k+1} \end{bmatrix} \tag{6}$$

The generator polynomial for this version of $RS(k)$ is therefore $\prod_{i=0}^{n-k-1}(x-\alpha^i)$.

Definition 5.1. For a vector e we define the *locator polynomial* to be

$$f^e(x) = \prod_{i:e_i \neq 0} (x - \alpha^i) \tag{7}$$

Step 2) of the algorithm takes the syndrome vector $s = eH$ as input and produces the error-locator polynomial. Before we get into the details of the algorithm for step 2) it will be useful to broaden the notion of the syndrome.

Let $f(x) = f_0 + f_1x + f_2x^2 + \cdots + f_{n-k-1}x^{n-k-1}$ be any polynomial and let $f = (f_0, f_1, \dots, f_{n-k-1})$ be the associated vector. Then

$$Hf = \begin{bmatrix} 1 & 1 & 1 & \dots & \dots & 1 \\ 1 & \alpha & \alpha^2 & \dots & \dots & \alpha^{n-k-1} \\ 1 & \alpha^2 & \alpha^4 & \dots & \dots & \alpha^{2(n-k-1)} \\ \dots & \dots & \dots & \dots & \dots & \\ \dots & \dots & \dots & \dots & \dots & \\ \dots & \dots & \dots & \dots & \dots & \\ 1 & \alpha^{n-1} & \alpha^{n-2} & \dots & \dots & \alpha^{k+1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \dots \\ \dots \\ f_{n-k-1} \end{bmatrix} = \begin{bmatrix} f(1) \\ f(\alpha) \\ f(\alpha^2) \\ \dots \\ \dots \\ \dots \\ \dots \\ f(\alpha^{n-1}) \end{bmatrix}$$

For an error vector e and syndrome vector s we have

$$\begin{aligned} sf &= eHf \\ &= e \begin{bmatrix} f(1) \\ f(\alpha) \\ f(\alpha^2) \\ \dots \\ \dots \\ \dots \\ \dots \\ f(\alpha^{n-1}) \end{bmatrix} \\ &= \sum_{i=0}^{n-1} e_i f(\alpha^i) \end{aligned}$$

Notice that letting $f(x) = x^m$ we get $sf = \sum_{i=0}^{n-1} e_i(\alpha^i)^m = s_m$. This discussion motivates the following definition.

Definition 5.2. Given an error vector e we define the *syndrome map*

$$\begin{aligned} S^e : \mathbb{F}_q[x] &\to \mathbb{F} \\ f(x) &\mapsto \sum_{i=0}^{n-1} e_i f(\alpha^i) \end{aligned}$$

Here is an important property of the error locator polynomial:

$$S^e(f^e) = \sum_{i=0}^{n-1} e_i f^e(\alpha^i) = 0$$

This is because $f^e(\alpha^i) = 0$ whenever $e_i \neq 0$. Furthermore, $S^e(x^r f^e(x)) = 0$ since $x^r f^e(x)$ also vanishes at all α^i such that $e_i \neq 0$.

Proposition 5.3. *Let e have weight t and suppose that $f(x)$ satisfies $S^e(x^r f(x)) = 0$ for $r = 0, 1, \dots, t-1$. Then $f(x)$ is a multiple of $f^e(x)$, the locator polynomial for e. In particular, if $\deg f(x) \leq t$ then $f(x)$ is a constant multiple of $f^e(x)$.*

Proof: If $S^e(x^r f(x)) = 0$ for all $r = 0, \dots, t-1$ then $S^e(g(x)f(x)) = 0$ for any polynomial $g(x)$ with $\deg g(x) < t-1$. Suppose that $e_k \neq 0$. Let

$$g(x) = \prod_{\substack{e:e_i \neq 0 \\ i \neq k}} (x - \alpha^i)$$

Then $g(x)$ vanishes at all error positions except α^k. Consequently,

$$\begin{aligned} S^e(g(x)f(x)) &= \sum_{i=0}^{n-1} e_i g(\alpha^i) f(\alpha^i) \\ &= e_k g(\alpha^k) f(\alpha^k) \end{aligned}$$

Now $e_k \neq 0$, $g(\alpha^k) \neq 0$ and, since $\deg g(x) = t-1$, we know $S^e(f(x)g(x)) = 0$. This forces $f(\alpha^k) = 0$. Since k was chosen arbitrarily such that $e_k \neq 0$, we conclude that $f(\alpha_k) = 0$ for all k such that $e_k \neq 0$. Thus $f(x)$ is a multiple of $f^e(x)$. Finally, if $\deg f(x) = t$, then, since $\deg f^e(x) = t$, $f(x)$ must be a constant multiple of $f^e(x)$. □

We now come to the algorithm for computing the error-locator polynomial.

The Berlekamp-Massey algorithm

Definition 5.4. Let $f(x) \in \mathbb{F}_q[x]$. Suppose r is such that $S^e(x^r f(x)) \neq 0$ but $S^e(x^i f(x)) = 0$ for $0 \leq i < r$. We define the span, fail and discrepancy of $f(x)$ to be

- $\operatorname{span} f(x) = r$,
- $\operatorname{fail} f(x) = \deg f(x) + r$,
- $\operatorname{discr} f(x) = S^e(x^r f(x))$.

If there is no such r then $\operatorname{span} f(x)$ and $\operatorname{fail} f(x)$ are defined to be infinite.

You will prove in an exercise that if $\operatorname{span} f(x) = c$ then for any polynomial $g(x)$ of degree less than c, $S^e(f(x)g(x)) = 0$.

We need two basic results to establish the validity of the Berlekamp-Massey algorithm. The first concerns the impact of the existence of a polynomial having span c on polynomials of degree c. The second is the basic idea in the update process in the Berlekamp-Massey algorithm.

Proposition 5.5. *Let $\deg f(x) = b$, $\operatorname{span} f(x) = c$. For any polynomial $g(x)$ of degree c, $\operatorname{span} g(x) \le b$ and $\operatorname{fail} g(x) \le c + b$.*

Furthermore, for any $g(x)$ of degree at most c, $\operatorname{fail} g(x) \le c + b$.

Proof: We know that $S^e(x^i(f(x)) = 0$ for $i < c$ and that $S^e(x^c f(x)) \neq 0$. Let $g(x)$ be any polynomial of degree c. We want to show that $S^e(x^i g(x)) \neq 0$ for some $i \le b$. It is sufficient to show that $S^e(f(x)g(x)) \neq 0$. at most b.

Let $\alpha \in \mathbb{F}_q$ be the leading coefficient of $g(x)$. Then $\deg(g(x) - \alpha x^c) < c$. Therefore

$$\begin{aligned} S^e(f(x)g(x)) &= S^e(f(x)(g(x) - \alpha x^c)) + S^e(\alpha x^c f(x)) \\ &= \alpha S^e(x^c f(x)) \end{aligned}$$

This is nonzero, so $\operatorname{span} g(x) \le b$ and $\operatorname{fail} g(x) \le b + c$ as was to be proved.

For any i between 0 and c, $\deg x^i f(x) = b + i$, $\operatorname{span} x^i f(x) = c - i$. Therefore any polynomial of degree $c - i$ has fail at most $c + b$. This establishes the last statement of the theorem. □

Corollary 5.6. *Let e have weight t. If $\operatorname{span} f(x) \ge t$ then $\operatorname{span} f(x)$ is actually infinite: $f(x)$ is a multiple of $f^{(e)}(x)$.*

Proof: Suppose that $\operatorname{span} f(x) \ge t$ but is finite. By the proposition, for any polynomial $g(x)$ of degree t, $\operatorname{fail} g(x)$ must be finite. This contradicts the existence of $f^{(e)}(x)$ which has degree t and infinite fail. Thus any polynomial of finite span has span less than t. □

Proposition 5.7. *Suppose that $f(x)$ has span r and discrepancy μ and that $g(x)$ has span c and discrepancy ν. If $r \le c$ then $h(x) = f(x) - (\mu/\nu)x^{c-r}g(x)$ has span larger than r.*

Proof: Notice that $\operatorname{span} x^{c-r}g(x) = r$ since

$$\begin{aligned} S^e(x^i x^{c-r} g(x)) &= S^e(x^{c-(r-i)} g(x)) \\ &= \begin{cases} 0 & \text{if } i < r \\ \nu & \text{if } i = r \end{cases} \end{aligned}$$

Since S^e is linear it is clear that $\operatorname{span} h(x) \ge r$, and the coefficient of $g(x)$ is chosen so that we get cancellation of the discrepancies.

$$\begin{aligned} S^e(x^r(f(x) - (\mu/\nu)x^{c-r}g(x))) &= S^e(x^r f(x)) - (\mu/\nu)S^e(x^c g(x)) \\ &= \mu - (\mu/\nu)\nu \end{aligned}$$

This shows that $\operatorname{span} h(x) > r$. □

Here is the Berlekamp-Massey algorithm, for the code $RS(k)$ over $\mathbb{F}_q$ with check matrix (6).

Data: Compute for each m, polynomials $f^{(m)}(x)$ and $g^{(m)}(x)$.

Initialization: For $m = -1$, $f^{(-1)}(x) = 1$ and $g^{(-1)}(x) = 0$.

Algorithm: For $m = 0$ to $n - k - 1$, let

$$\begin{aligned} c &= \deg f^{(m-1)}(x) - 1 \\ r &= m - \deg f^{(m-1)}(x) = m - c + 1 \\ \mu &= S^e(x^r f^{(m-1)}(x)) \end{aligned}$$

If $r \le c$ or $\mu = 0$

$$\begin{bmatrix} f^{(m)}(x) \\ g^{(m)}(x) \end{bmatrix} = \begin{bmatrix} 1 & -\mu x^{c-r} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f^{(m-1)}(x) \\ g^{(m-1)}(x) \end{bmatrix}$$

Otherwise,

$$\begin{bmatrix} f^{(m)}(x) \\ g^{(m)}(x) \end{bmatrix} = \begin{bmatrix} x^{r-c} & -\mu \\ \mu^{-1} & 0 \end{bmatrix} \begin{bmatrix} f^{(m-1)}(x) \\ g^{(m-1)(x)} \end{bmatrix}$$

Output $f^{(n-k-1)}(x)$.

Theorem 5.8. *At the end of the mth iteration,*

1) $f^{(m)}(x)$ *is monic and* $\operatorname{fail} f^{(m)}(x) > m$

2) $g^{(m)}(x)$ *is either 0 or*

$$\begin{aligned} \operatorname{span} g^{(m)}(x) &= \deg f^{(m)}(x) - 1, \\ \operatorname{fail} g^{(m)}(x) &\le m, \text{ and} \\ \operatorname{discr} g^{(m)}(x) &= 1 \end{aligned}$$

Proof: We proceed by induction. The initial case, $m = -1$ is easily verified. Assume the statements of the theorem are true for $m - 1$; we will prove them for m.

Consider the mth iteration of the algorithm. If $\mu = 0$ then $f^{(m-1)}(x)$ satisfies $\operatorname{fail} f^{(m-1)}(x) > m$, so the algorithm sets $f^{(m)}(x) = f^{(m-1)}(x)$ and $g^{(m)}(x) = g^{(m-1)}(x)$. Item 2) of the proposition is true by the induction hypothesis.

Now consider $\mu \ne 0$. If $r \le c$, then the algorithm sets

$$f^{(m)}(x) = f^{(m-1)}(x) - \mu x^{c-r} g^{(m-1)}(x)$$

Both terms have span r, so by Proposition 5.7, $\operatorname{span} f^{(m)}(x) > r$. We will show next that $\deg f^{(m)}(x) = \deg f^{(m-1)}(x)$, so $\operatorname{fail} f^{(m)}(x) > m$ as desired. Furthermore, since $g^{(m)} = g^{(m-1)}$, the induction hypothesis shows that item 2) is satisfied.

By the induction hypothesis on $g^{(m-1)}(x)$,

$$\begin{aligned}\deg g^{(m-1)}(x) &= \operatorname{fail} g^{(m-1)}(x) - \operatorname{span} g^{(m-1)}(x)\\ &\leq m-1-c\end{aligned}$$

Therefore we have

$$\begin{aligned}\deg x^{c-r} g^{(m-1)}(x) &\leq c-r+(m-1-c)\\ &= m-r-1\\ &< \deg f^{(m-1)}(x)\end{aligned}$$

This shows that $\deg f^{(m)}(x) = \deg f^{(m-1)}(x)$ and also that $f^{(m)}(x)$ is monic.

Finally, suppose that $\mu \neq 0$ and $r > c$. Similar computations to those in the preceeding case show that $f^{(m)}(x) = x^{r-c} f^{(m-1)}(x) - \mu g^{(m-1)}(x)$ is monic, of degree $r+1$ and with $\operatorname{fail} f^{(m)}(x) > m$. Furthermore, $g^{(m)}(x) = \mu^{-1} f^{(m-1)}(x)$ has span r, fail m, and discrepancy 1. □

Remark 5.9. Since $\operatorname{span} g^{(m)}(x) = \deg f^{(m)}(x) - 1$ and $\operatorname{fail} g^{(m)}(x) \leq m$, Proposition 5.5 tells us that no polynomial of degree less than $\deg f^{(m)}(x)$ has fail larger than m. We know that $\operatorname{fail} f^{e}(x) > m$ for all m, so $\deg f^{(m)}(x) \leq \deg f^{e}(x) = \operatorname{wt} e$.

In the algorithm, the case where $r \geq \deg f^{(m)}$ and $\mu \neq 0$ leads to a change of degree in the f polynomial. The reason for the change is evident from the remark, if $\operatorname{span} f^{(m)}(x) \geq \deg f^{(m)}(x)$, then no polynomial of degree $\deg f^{(m)}(x)$ can have fail larger than m.

Corollary 5.10. *Let* $\operatorname{wt} e = t$. *For* $m \geq 2t-1$, $f^{(m)}(x) = f^{e}(x)$. *Therefore, using the Berlekamp-Massey algorithm for decoding the code* $RS(k)$ *whose minimum distance is* $d = n-k+1$, *the error locator can be found provided* $\operatorname{wt} e \leq (d-1)/2$.

Proof: Let $m \geq 2t-1$. By the remark, $\deg f^{(m)}(x) \leq t$. By the theorem,

$$\begin{aligned}&\operatorname{span} f^{(m)}(x) > m \geq m - \deg f^{(m)}(x)\\ &\operatorname{span} f^{(m)}(x) 2t-1-t\end{aligned}$$

Thus $\operatorname{span} f(^{(m)}(x) \geq t$. By Corollary 5.6 $f^{(m)}(x)$ must be a multiple of $f^{e}(x)$. But $f^{(m)}(x)$ is monic of degree t so it must equal $f^{e}(x)$.

For $RS(k)$ with check matrix H in (6), the algorithm iterates from 0 to $n-k-1$. If the number of errors is t and $t \leq (d-1)/2$ then

$$\begin{aligned}2t-1 &\leq d-2\\ &= n-k-1\end{aligned}$$

so $f^{(n-k-1)}(x)$ is the error locator. □

Exercises 5.11.

1. For a vector e, suppose that $\operatorname{span} f(x) = r$. Using the linearity of S^e, show that $S^e(fg) = 0$ for all g with degree less than r, but that $S^e(fg) \neq 0$ for all g with degree r.

2. Fill in the details in the proof of Theorem 5.8 for $\mu \neq 0$ and $r \geq \deg f^{(m)}(x)$.

3. Let $\alpha \in \mathbb{F}_9$ satisfy $\alpha^2 = \alpha + 1$. Consider the RS code over $\mathbb{F}_9$ with $k = 4$ and check matrix from (6). Compute the syndrome and apply the Berlekamp-Massey algorithm for the received vector $w = (\alpha^7, 0, 1, \alpha, \alpha^4, \alpha^4, \alpha^3, \alpha^6)$.

4. Let $\alpha \in \mathbb{F}_{16}$ satisfy $\alpha^4 = \alpha+1$. Consider the RS code over $\mathbb{F}_{16}$ with $k = 9$ and check matrix from (6). Compute the syndrome and apply the Berlekamp-Massey algorithm for the received vector $w = (1, 1, \alpha^2, \alpha^{11}, 1, \alpha^7, 1, \alpha^6, \alpha^8, \alpha, \alpha^9, \alpha^5, 1, \alpha^6, \alpha^7)$

6 Finding the Error Locations

Step 3) of the algorithm is fairly simple. The input is the error-locator polynomial $f^e(x)$. The output is a list of the error locations. Since the error locations are the roots of the polynomial $f^e(x)$, we need only evaluate $f^e(x)$ at all the field elements α^i. We define a vector $\tilde{e}$ by

$$\tilde{e}_i = \begin{cases} 1 & \text{if } f^e(\alpha^i) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Efficient evaluation of $f^e(\alpha^i)$ can be done using Horner's method and implemented in circuitry with an adder and constant multiplier.

Exercises 6.1.

1. Find the error locations for problem 3) of Section 5.

2. Find the error locations for problem 4) of Section 5.

7 The Error Evaluator Polynomial

Step 4) of the algorithm takes as input the error-locator polynomial $f^e(x)$, the list of locations $\tilde{e}$, and the syndrome vector s. Recall that we defined the syndrome vector for an error vector e and later introduced the syndrome map $S^e : \mathbb{F}_q[x] \to \mathbb{F}_q$. We expand the notion of the syndrome a bit further by defining a syndrome polynomial. Recall that $n = q - 1$, and that for each nonzero $\beta \in \mathbb{F}_q$, $(x - \beta)$ is a factor of $x^n - 1$. Note that

$$\frac{x^n - 1}{x - \beta} = x^{n-1} + \beta x^{n-2} + \beta^2 x^{n-3} + \cdots + \beta^{n-2} x + \beta^{n-1}$$

For a vector e, consider the weighted sum of polynomials like those above:

$$\begin{aligned}\sum_{i=0}^{n-1} e_i \frac{x^n-1}{x-\alpha^i} &= \sum_{i=0}^{n-1} e_i \sum_{m=0}^{n-1} x^{n-1-m}(\alpha^i)^m \\ &= \sum_{m=0}^{n-1} x^{n-1-m} \sum_{i=0}^{n-1} e_i(\alpha^i)^m \\ &= \sum_{m=0}^{n-1} x^{n-1-m} s_m\end{aligned}$$

where $s_m = \sum_{i=0}^{n-1} e_i(\alpha^i)^m$ is a coordinate of the syndrome vector.

Definition 7.1. For $m = 0, 1, \ldots, n-1$ let $s_m = S^e(x^m)$. The *syndrome polynomial* associated to e is

$$\begin{aligned}s^e(x) &= s_0x^{n-1} + s_1x^{n-2} + \cdots + s_{n-2}x + s_{n-1} && (8)\\ &= \sum_{i=0}^{n-1} e_i \frac{x^n-1}{x-\alpha^i} && (9)\end{aligned}$$

Here is another characterization of the error locator polynomial, obtained from the syndrome polynomial.

Proposition 7.2. *Let $s^e(x)$ be the syndrome polynomial associated to e and let $f(x)$ be an arbitrary polynomial. Then $f(x)s^e(x)$ is in the ideal generated by $x^n - 1$ if and only if $f(x)$ is a multiple of $f^e(x)$.*

Proof: We have

$$s^e(x) = (x^n - 1)\sum_{i=0}^{n-1} \frac{e_i}{x-\alpha^i}$$

Therefore

$$\begin{aligned}f^e(x)s^e(x) &= \left(\prod_{j:e_j\neq 0}(x-\alpha^j)\right)(x^n-1)\sum_{i=0}^{n-1}\frac{e_i}{x-\alpha^i} \\ &= (x^n-1)\left[\sum_{i=0}^{n-1} e_i \prod_{\substack{j:e_j\neq 0\\ j\neq i}}(x-\alpha^j)\right]\end{aligned}$$

This computation shows that $f^e(x)s^e(x)$ is in the ideal generated by $x^n - 1$.

Conversely, suppose that $f(x)s^e(x) \in < x^n - 1 >$. Then for each i such that $e_i \neq 0$, $f(x)s^e(x) \equiv 0 \mod x - \alpha^i$. But only one term in the sum (9) is nonzero modulo $x - \alpha^i$,

$$s^e(x) \equiv e_i \prod_{\substack{j=1\\ j\neq i}}^{n} (x-\alpha^j) \quad \mod x - \alpha^i$$

This is nonzero. Thus we must have $f(x)$ congruent to 0 mod $x - \alpha^i$. Since i was arbitrary, $f(x)$ is divisible by $x - \alpha^i$ for each error location i. Therefore $f(x)$ is divisible by $f^e(x)$. □

Definition 7.3. Let $s^e(x)$ be the syndrome polynomial for e. Let $\phi^e(x)$ be the polynomial such that $f^e(x)s^e(x) = \phi^e(x)(x^n - 1)$. We call $\phi^e(x)$ the *evaluator polynomial* for e.

The reason we call $\phi^e(x)$ the error evaluator is that it can be used to evaluate errors. Before seeing how that works we need to define the derivative of a polynomial.

The derivative of a polynomial

Over a finite field we cannot define the derivative using limits because we have no way to let the distance between two field elements go to zero. Nevertheless, over any field we can define the derivative using the usual formula.

Definition 7.4. Let $f(x) = f_0 + f_1x + f_2x + \cdots + f_dx^d$ be a polynomial defined over a field F. The derivative of $f(x)$ is the polynomial

$$f'(x) = f_1 + 2f_2x + 3f_3x^2 + \cdots + df_dx^{d-1}$$

The integer coefficient k of f_kx^{k-1} is the k-fold sum of 1_F, so it is computed modulo the characteristic of F.

We must verify that the usual rules for differentiation hold.

Theorem 7.5. *The derivative satisfies the following properties. For $f(x), g(x) \in F[x]$ and $\alpha \in F$,*

- $(f(x) + g(x))' = f'(x) + g'(x)$ *(Linearity)*
- $(\alpha f(x))' = \alpha f'(x)$ *(Homogeneity)*
- $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$ *(Leibnitz' rule)*

Proof: The proofs for linearity and homogeneity are easy. The monomial case of Leibnitz' rule follows:

$$\begin{aligned}(x^ix^j)' &= (x^{i+j})' \\ &= (i+j)x^{i+j-1} \\ &= ix^{i-1}x^j + x^i(jx^{j-1}) \\ &= (x^i)'x^j + x^i(x^j)'\end{aligned}$$

We use the monomial case to derive Leibnitz' rule for a monomial times an arbitrary polynomial. One can easily extend to the product of two arbitrary polynomials. By linearity and homogeneity of the derivative,

$$(x^ig(x))' = \sum_j g_j(x^ix^j)'$$

Applying Leibnitz' rule for monomials

$$\begin{aligned}
&= \sum_j g_j((x^i)'x^j + x^i(x^j)') \\
&= (x^i)'\sum_j g_j x^j + x^i \sum_j g_j (x^j)' \\
&= (x^i)'g(x) + x^i g'(x)
\end{aligned}$$

□

Computing the error values

We now return to step 4) of decoding, computing the error values. The following proposition shows that we need only evaluate the two polynomials ${f^e}'(x)$ and $\phi^e(x)$ at the error locations. The proof is left as an exercise.

Proposition 7.6. *Let $f^e(x)$ and $\phi^e(x)$ be the locator and evaluator polynomial for the error vector e. For each error position k,*

$$e_k = \frac{\phi^e(\alpha^k)}{{f^e}'(\alpha^k)}$$

Computing ${f^e}'(x)$ is a simple matter from the definition of the derivative. Computing $\phi^e(x)$ is basically a polynomial multiplication. Since we know that $f^e(x)s(x) = \phi^e(x)(x^n - 1)$, and $\deg f^e(x) = t$, we have $\deg \phi^e(x) = \deg f^e(x) + \deg s(x) - n \leq t - 1$. Furthermore, the coefficient of x^j in $\phi^e(x)$ is the coefficient of x^{n+j} in $f^e(x)s(x)$. This coefficient is

$$f_{j+1}s_0 + f_{j+2}s_1 + f_{j+3}s_2 + \cdots + f_{t-1}s_{t-2-j} + f_t s_{t-1-j}$$

where $f^e(x) = f_0 + f_1 x + \cdots + f_t x^t$.

It is worth pointing out that the decoder can only compute the first few terms of $s(x)$—namely $s_m x^m$ for $m \leq n - k - 1$. At first glance it seems that we cannot hope to compute $\phi^e(x)$, since it is the product of $f^e(x)$ and $s(x)$. But, the previous paragraph shows that we only need s_m for $m \leq t - 1$ to compute $\phi^e(x)$. Since $t < d/2$ and $d = n - k + 1$, the decoder does have access to the data necessary to compute $\phi^e(x)$.

Exercises 7.7.

1. Prove that for $\alpha_i \in F$ and $f(x) = \prod_{i=1}^r (x - \alpha_i)$ that $f'(x) = \sum_{i=1}^r \prod_{j \neq i}(x - \alpha_i)$.
2. With $f(x)$ as above, show that $f'(\alpha_i) = \prod_{j \neq i}(\alpha_i - \alpha_j)$.
3. Prove Proposition 7.6. Use the formula for $\phi^e(x)$ derived in Proposition 7.2.
4. Find the error values for problem 3) of Section 5
5. Find the error values for problem 4) of Section 5

8 More on Error Evaluation

One drawback of the method for error evaluation described in the previous section is that the computation of the evaluator polynomial introduces a delay between the computation of the locator polynomial and the computation of the error values. We show in this section that the Berlekamp-Massey algorithm can be extended to compute the error evaluator polynomial as well as the error locator polynomial. Using this extension of the Berlekamp-Massey algorithm eliminates the aforementioned delay, but it introduces a different practical problem. It requires extra storage to handle two new polynomials $\phi^{(m)}(x)$ and $\psi^{(m)}(x)$. Fortunately, an even better method will be derived from the extended BM algorithm that eliminates the need to compute the evaluator polynomial.

Here is the Berlekamp-Massey algorithm, for the code $RS(k)$ over $\mathbb{F}_q$ with check matrix (6).

Data: Compute for each m, a matrix of polynomials

$$B^{(m)} = \begin{bmatrix} f^{(m)}(x) & \phi^{(m)}(x) \\ g^{(m)}(x) & \psi^{(m)}(x) \end{bmatrix}$$

Initialization: For $m = -1$,

$$B^{(-1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Algorithm: For $m = 0$ to $n - k - 1$, let

$$\begin{aligned} c &= \deg f^{(m-1)}(x) - 1 \\ r &= m - \deg f^{(m-1)}(x) = m - c + 1 \\ \mu &= S^e(x^r f^{(m-1)}(x)) \end{aligned}$$

If $r \leq c$ or $\mu = 0$, set

$$U^{(m)} = \begin{bmatrix} 1 & -\mu x^{c-r} \\ 0 & 1 \end{bmatrix}$$

Otherwise, set

$$U^{(m)} = \begin{bmatrix} x^{r-c} & -\mu \\ \mu^{-1} & 0 \end{bmatrix}$$

Then,

$$B^{(m)} = UB^{(m-1)}$$

Output $f^{(n-k-1)}(x)$ and $\phi^{(n-k-1)}(x)$.

The polynomials $f^{(m)}(x)$ and $g^{(m)}(x)$ in the new algorithm are clearly the same as those produced by the original one. Before proving the algorithm produces $\phi^e(x)$ we need the following proposition.

Proposition 8.1. *For $0 \leq r \leq n-1$, $S^e(x^r f(x))$ is the coefficient of x^{n-1-r} in $f(x)s(x)$.*

Proof: Let $f(x) = f^0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$. Then

$$S^e(f(x)) = f_0 s_0 + f_1 s_1 + \ldots f_d s_d$$

This is also the coefficient of x^{n-1} in $f(x)s(x)$. Likewise $S^e(x^r f(x))$ is the coefficient of x^{n-1} in $x^r f(x)s(x)$ and therefore is also the coefficient of x^{n-1-r} in $f(x)s(x)$. □

Theorem 8.2. *At the end of the mth iteration of the algorithm,*

1) $f^{(m)}(x)s(x) - \phi^{(m)}(x)(x^n-1)$ has degree at most $n - m + \deg f^{(m)}(x) - 2$

2) $g^{(m)}(x)s(x) - \psi^{(m)}(x)(x^n-1)$ is monic and has degree exactly $n - \deg f^{(m)}(x)$.

Proof: We proceed by induction. When $m = -1$, we have $f^{(-1)}(x) = 1$, which has degree 0, and $\phi^{(-1)}(x) = 0$. Therefore item 1) says $\deg s(x) < n$, which is true. We also have $g^{(-1)}(x) = 0$ and $\psi^{(-1)}(x) = 1$ so item 2) says that $x^n - 1$ is monic of degree n, which is also true.

Assume the statements of the theorem are true for $m-1$; we will prove them for m. Let $c = \deg f^{(m-1)}(x) - 1$. By the induction hypothesis,

$$\begin{aligned} \deg f^{(m-1)}(x)s(x) - \phi^{(m-1)}(x)(x^n-1) &\leq n - (m-1) + \deg f^{(m-1)}(x) - 2 \\ &= n - m + c \\ \deg g^{(m-1)}(x)s(x) - \psi^{(m-1)}(x)(x^n-1) &= n - c + 1. \end{aligned}$$

In the algorithm we set $\mu = S^e(x^r f^{(m-1)}(x))$ where $r = m - c + 1$. By the proposition, μ is the coefficient of x^{n-m+c} in $f^{(m-1)}(x)s(x)$. If $\mu = 0$, both items of the proposition are satisfied by setting $B^{(m)} = B^{(m-1)}$.

If $\mu \neq 0$ then the degree of $f^{(m-1)}(x)s(x) - \phi^{(m-1)}(x)(x^n-1)$ is exactly $n - m + c$. If $r \leq c$ then the algorithm sets

$$\begin{aligned} f^{(m)}(x) &= f^{(m-1)}(x) - \mu x^{c-r} g^{(m-1)}(x) \\ \phi^{(m)}(x) &= \phi^{(m-1)}(x) - \mu x^{c-r} \psi^{(m-1)}(x) \end{aligned}$$

Therefore we get,

$$\begin{aligned} f^{(m)}(x)s(x) &- \phi^{(m)}(x)(x^n-1) \\ &= \left[f^{(m-1)}(x)s(x) - \phi^{(m-1)}(x)(x^n-1)\right] \\ &\quad - \mu\left[x^{c-r}\left(g^{(m-1)}(x)s(x) - \psi^{(m-1)}(x)(x^n-1)\right)\right] \end{aligned} \tag{10}$$

By the induction hypothesis, both bracketed terms have degree $n - m + c$. The factor μ ensures the leading terms cancel. Thus $f^{(m)}(x)s(x) - \phi^{(m)}(x)(x^n-1)$ has degree at most $n - m + \deg f^{(m)}(x) - 2$ as required.

The case when $r \geq \deg f^{(m)}(x)$ is similar. □

Corollary 8.3. *Let* $\operatorname{wt} e = t$. *When* $m \geq 2t-1$, $\phi^{(m)}(x) = \phi^e(x)$.

Proof: For $m \geq 2t-1$, Corollary 5.10 says that $f^{(m)}(x) = f^e(x)$. We also know that $f^e(x)s(x) = \phi^e(x)(x^n-1)$. Therefore,

$$\begin{aligned} f^{(m)}(x)s(x) - \phi^{(m)}(x)(x^n-1) &= \phi^e(x)(x^n-1) - \phi^{(m)}(x)(x^n-1) \\ &= (\phi^e(x) - \phi^{(m)}(x))(x^n-1) \end{aligned}$$

The theorem says that $f^{(m)}(x)s(x) - \phi^{(m)}(x)(x^n-1)$ has degree at most $n-m+t-1 \geq n-t$. This is less than n so $\phi^e(x) = \phi^{(m)}(x)$.

□

Error evaluation without the evaluator polynomial

The real benefit of the previous algorithm is not that it is the most practical, but that it gives a new formula for computing error values that does not require the error evaluator $\phi^e(x)$.

From the algorithm it is clear that

$$B^{(m)} = \left(\prod_{i=0}^{m} U^{(i)}\right) B^{(-1)}$$

Since $B^{(-1)}$ is the identity matrix and each $U^{(i)}$ has determinant 1 we get

$$\begin{aligned} \det B^{(m)} &= f^{(m)}(x)\psi^{(m)}(x) - g^{(m)}(x)\phi^{(m)}(x) \\ &= 1 \end{aligned}$$

In particular, when $m \geq 2t+1$ we see that $g^{(m)}(x)$ and $\psi^{(m)}(x)$ are two polynomials that give a combination of $f^e(x)$ and $\phi^e(x)$ which is 1. Let α^i be an error position. Then evaluating at α^i we get $g^{(m)}(\alpha^i)\phi^e(\alpha^i) = -1$. From Proposition 7.6 we have $\phi^e(\alpha^i) = e_i f^{e\prime}(\alpha^i)$. Substituting in the previous equation we have established the following proposition.

Proposition 8.4. *Let* $\operatorname{wt} e = t$. *For* $m > 2t-1$ *and* $g^{(m)}(x)$ *be as in the Berlekamp-Massey algorithm. If* $e_k \neq 0$ *then*

$$e_k = (f^{e\prime}(\alpha^k)\, g^{(m)}(\alpha^k))^{-1} \tag{11}$$

Exercises 8.5.

1. Fill in the details in the proof of Theorem 8.2 for $\mu \neq 0$ and $r \geq \deg f^{(m)}(x)$.
2. Find the error values for problem 3) of Section 5.
3. Find the error values for problem 4) of Section 5.