Introduction to Error Correcting Codes

Ling-Pei Kung

Table of Content

List of Tables

List of Figures

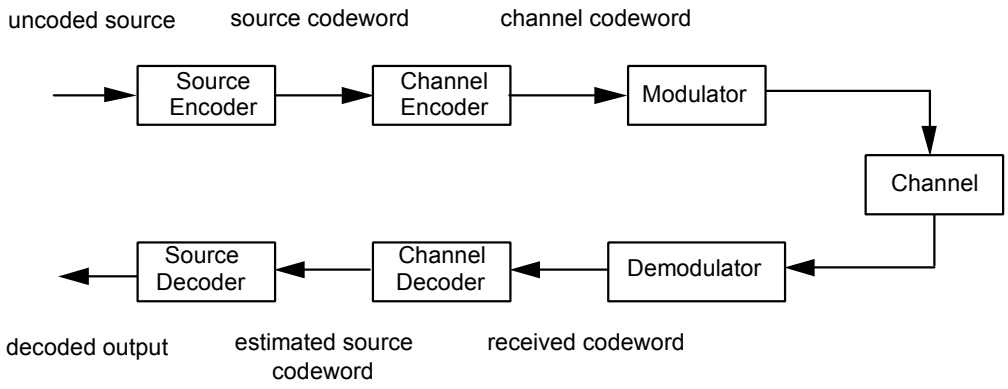

Figure 1: An simplified communication system.

1 Error Correction Codes for Transmission and Storage

1.1 Type of Codes

There are two types of code: block code a nd convolutional code. block codes do not depend on previous messages while convolutional codes do.

1.1.1 Block Codes

A block code of size k over an finite alphabet with q symbols {0, 1, 2, _, q-1} is a set of k q-ary sequences of length n called codewords.

A message $u = (u_0, u_1, u_2, ..., u_{k-1})$, code word $v = (v_0, v_1, v_2, ..., v_{n-1})$

$$v1 \infty n = u1 \infty k \infty Gk \infty n$$

where $u_i, v_i \in GF(q)$ and $v \in GF(q)^n$

v does not depend on past u, memoryless

There are q^k possible code words of length n, they are called q-ery (n, k) block code.

The Code rate is defined as R = k/n.

Example: Binary (7, 4) linear block code

Table 1: A Binary (7, 4) linear block code

Messages	Code Words
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 1)	(0 1 1 0 1 0 1)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)

(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

The generator matrix $G_{k\infty n}$ is given by

$$G_{4\infty 7} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1\ 1\ 0\ 1\ 0\ 0\ 0 \\ 0\ 1\ 1\ 0\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ 1\ 0\ 1\ 0\ 0\ 0\ 1 \end{bmatrix}$$

where g_0, g_1, g_2, g_3 are 4 (=k) linearly independent code words.

1.1.2 Convolutional Codes

message $u_i = (u_{i0}, u_{i1}, u_{i2}, ..., u_{i(k-1)})$, code word $v_i = (v_{i0}, v_{i1}, v_{2i}, ..., v_{i(n-1)})$

$v_i = f(u_i, u_{i-1}, u_{i-2}, ..., u_{i-m})$

v depends on current and last m message units -- memory order of m, sequential logic

(n, k, m) convolutional code

Code rate R = k/n

Hard decision: the output of demodulator is quantized

Soft decision: the output of demodulator is left unquantized, decoder accepts multilevel or analog inputs, soft-decision decoding offers significant performance improvement over hard-decision decoding.

1.2 Modulation and Demodulation

AWGN: Additive White Guassian Noise

DMC: Discrete Memoryless Channel

Information transmission rate: R/T (bits/second)

Symbol transmission rate: Baud rate 1/T (symbols/sec)

1.3 Maximum Likelihood Decoding (MLD)

Minimum Distances Decoder: chooses a code word that minimizes the distance between r and v. This is equivalent to minimize the meas square error (MSE).

1.4 Type of Errors

Random Error Channel: transmission errors occur randomly in the received sequence, memoryless channels

Burst Error Channel: channel with memory

Compound Channel: both random error and burst error

1.5 Error Control Stragies

Forward Error Correction (FEC): error control for a one-way system, can not request for retransmission. receiver must correct errors itself.

Examples:

- Block Codes
  - -Linear Block code
    - -Cyclic Code
- Convolutional Codes
  - Viterbi Decoding
  - Sequential Decoding
  - Majority-Logic Decoding
  - Burst-Error Correction

Automatic Repeat Request (ARQ): error control for a two-way system

- Stop-and-Wait ARQ: ACK and NAK, designed for half-duplex channels
- Continuous ARQ: NAK, designed for full-duplex channels
  - go-back N ARQ
  - selective-repeat ARQ
- Hybrid ARQ: ARQ+ FEC for two-way system

2 Introduction to Abstract Algebra

2.1 Group

A set G on which a binary operation * is defined is called "a group under *" if the following conditions are satisfied

(i) Closure $\forall a, b \in G \quad c = a * b \quad$ then $c \in G$

(ii) Associativity The binary operation * is associative
i.e. $\forall a, b, c \in G \quad a*(b*c) = (a*b)*c$

(iii) Identity G contains an identity element e such that,
$\forall a \in G \quad a*e = e*a = a$

(iv) Inverse G contains an inverse a', such that
$\forall a \in G$ and $a \_ 0 \quad a*a' = a'*a = e$

Order of a group: the number of elements in a group
Finite group: A group of a finite number of elements.

A group is called a commutative group or Abelian group if
$\forall a, b \in G \quad a*b = b*a$

A group that consists of all the powers of one of its element is called a cyclic group.

Example: {0, 1, 2, 3} is a Abelian group under modulo-4 addition, but is not a group under modulo-4 multiplication.

Table 2: Modulo-4 addition

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Table 3: Modulo-4 multiplication

.	0	1	2	3

0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Because 2 does not have an inverse. In fact, this set is an integer ring of modulo-4.

Note the symmetry of the tables is the result of commutativity under + and · .

Example: The elements of GF(4) form a Abelian group under "addtion" and "multiplication". The group of nonzero elements of GF(q) under multiplication is a cyclic group.

Table 4: Addition of GF(4) elements.

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Table 5: Multiplication of GF(4) elements.

.	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

A group H is called a subgroup of G if

$\forall a \in H \quad a \in G$

$\forall a, b \in H \quad a*b \in H$

i.e. H is closed under *.

Example: The elements of GF(2), i.e.{0, 1}, form a subgroup of the elements of GF(4).

+	0	1
0	0	1
1	1	0

.	0	1
0	0	0
1	0	1

Given a finite group G and a subgroup H, a construction known as coset decomposition of G will produce a rectangular array called standard array. Each row of the array is called a coset. The first row of a standard array consists of elements from the subgroup H. The first element on the left of each row is called a coset leader.

Every element of G appears once and only once in a coset decomposition of G

If H is a subgroup of G, then the number of elements in H divides the number of elements in G.

The order of a finite group is divisible by the order of any of its elements.

2.2 Ring

A ring is a set of elements R on which two binary operations addition ″+″ and ″·″ are defined and the following are satisfied

(i) The set R is an Abelian group under addition (+)
(ii) R is closed under multiplication (not necessarily Abelian)
$\forall\ a, b \in R \qquad a \cdot b \in R$
(iii) R is associative
$a(bc) = (ab)c$
(iv) Multiplication is distributive over addition
$a(b + c) = ab + ac$
$(b + c)a = ba + ca$

2.2.1 Integer Ring

A commutative ring is one in which multiplication is commutative.

Example: An integer ring of modulo q. Let q be a positive integer. The quotient ring is the set {0, 1, _, q-1}with addition and multiplication defined by

$a + b = R_q[a + b]$

$a \cdot b = R_q[ab]$

and is denoted by MOD(q)

Example: GF(2) = MOD(2), GF(3) = MOD(3), but GF(4) _ MOD(4)

In fact, GF(q) = MOD(q) if and only if q is a prime integer.

2.2.2 Polynomial Ring

A polynomial over a field GF(q) denoted by

$$f(x) = f_0 + f_1x + f_2x^2 + ...+ f_{n-1}x^{n-1}$$

where $f_i \in GF(q)$

The symbol x is called an indeterminate.

If $f_{n-1} = 1$, f(x) is called a *monic polynomial*

Example: Polynomials over integer

$f(x) = x^2+3x+2$ x= -2, -1

Roots are intergers

$f(x) = 4x^2+5x+1$ x= -1, -1/4

Roots are rational numbers.

$f(x) = x^2+3x+1$ x=

Roots are real numbers.

$f(x) = x^2+x+1$ x =

Roots are complex numbers.

These polynomials are over an infinite interger ring. The roots can be in any sets, interger, rational numbers, real, complex.

Example: Polynomials over GF(q)

$f(x) = x^2+x+1$ $x= \alpha, \ \alpha^2$

The coefficients are from GF(2) and the roots are from GF(4) called symbols.

$f(x) = x^3+x+1$ $x = \alpha, \ \alpha^2, \alpha^4$

The coefficients are from GF(2) and the roots are from GF(8).

A polynomial p(x) of degree m is irreducible over GF(q) if p(x) is not divisible by any polynomial over GF(q) of degree less than m but greater than 0.

A *prime polynomial* is both monic and irreducible.

For any monic polynomial p(x) with nonzero degree over the field F, the ring of polynomials modulo p(x) is the set of all polynomials with degree samller than that of p(x), together with polynomial addition and polynomial multiplication modulo p(x), denoted by MOD(p(x)).

Example: A ring of polynomials over GF(2), generated by MOD(x^3+1) consists of the following elements:

$$0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1$$

This set of elements form a ring not a field since not all of them have inverses. There are a total of eight elements which are all the possible combinations of polynomials with degree less than 3. In general, there are q^n possible elements over GF(q) over x^n+1.

2.3 Field

A set of elements F on which two binary operations, addition "+" and multiplication "·" are defined. The set F together with + and · are a field if

(i) F is a commutative group under addition.
(ii) The set of nonzero elements in F is a commutative group under multiplication.
(iii) Multiplication is distributive over addition. that is

$$\forall\ a, b, c \in F \quad a.(b+c) = a.b + a.c$$

A field consists of at least two elements: additive identity and multiplicative identity

Order of a field: the number of elements in a field
Finite field: A field contains finite number of elements
Prime Field: The set of integers {0, 1, 2 ..., q-1}, where q is prime, is a field under modulo-q addition and multiplication and is denoted by GF(q).

2.3.1 Finite Field Based on Integr Ring

The quotient ring MOD(q) is a field if and only if q is a prime integer.

Example: GF(7)

Table 6: modulo-7 addition

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Table 7: modulo-7 multiplication

3	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Extension Field: for any positive integer m, a field of p^m elements is called an extension field of GF(p) and is denoted by GF(p^m).

The order of a finite field is a power of prime. Finite fields are called *Galois fields*.

The number of elements in the smallest subgroup of GF(q) is called the characteristic of GF(q), which is equavalent to the smallest positive integer λ such that $\sum_{i=1}^{\lambda} 1 = 0$.

Each Galios field contains a uniques smallest subfield which has a prime number of elements. Hence the characteristic λ of a finite field is prime.

Any finite field GF(q) of characteristic λ contains a subfield of GF(λ). If q _ λ, then q is a power of λ.

The order of a finite field element a is the smallest positive integer n such that $a^n=1$. The nonzero elements of a finite field form a group under multiplication of GF(q).

$\forall$ a $\in$ GF(q), and a _ 0, $a^{q-1}=1$

$\forall$ a $\in$ GF(q), and a _ 0, if $a^n=1$, (i.e. n is the order of a), then n divides q-1.

A *primitive element* a of GF(q) whose order is q-1.

The powers of a primitive element generate all the nonzero elements of GF(q).

2.3.2 Finite Field Based on Polynomial Ring

The ring of polynomials modulo a monic polynomial p(x) is a field if and only if p(x) is a prime polynomial.

Example: A field of polynomials over GF(2) generated by MOD(x^3+x+1) consists of the same elements of last example. The difference are the modulo addition and multiplication over p(x) which are part of the definition a field.

2.4 Primitive Elements

A field element α of GF(q) is a primitive element if the power of α generate all the nonzero elements of GF(q). In other words, every nonzero field element can expressed as a power of α.

Every Galois field has a primitive element because the nonzero elements of GF(q) form a cyclic group.

A primitive element has an order of q-1.

A primitive polynomial p(x) over GF(q) is a prime polynomial over GF(q) with the property that in the extension field constructed from modulo p(x), the field element represented by x is primitive.

Polynomials over GF(2) satisfiy

commutative and associative over + and ·

· is distributive over +

Polynomials with even number of terms over GF(2) are divisible by x+1

Any irreducible polynomial over GF(2) of degree m divides $x^{2^m-1}+1$.

divides X^n+1 is $n=2^m-1$

$f^2(x) = F(x^2)$, f(x) is over GF(2)

2.5 Construction of Galois Field GF(q^m)

From last section, we know if we have a primitive polynomial p(x) (will denote it by g(x) from now on) of degree m over GF(q), we can generate a finite field with q^m elements. In general, the i^{th} element in this field can be written as

$$a_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + ...+ a_{i,m-1}x^{m-1}$$

where $a_i \in$ GF(q) and x is a primitive element in the extention field GF(q^m).

This is a polynomial representation (convenient for addition) for the field elements.

We can also represent field elements by the power of this primitive element (denoted by α)

$$\alpha^i = a_i(\alpha) = a_{i0} + a_{i1}\alpha + a_{i2}\alpha^2 + ...+ a_{i,m-1}\alpha^{m-1}$$

This is called power representation (convenient for multiplication).

Since α generate all the nonzero elements (i.e. its order is q-1), the roots of

$$x^{q^m-1} - 1 = 0$$

form the finite field GF(q^m) = {0, 1, α, α^2, ..., α^{2^m-2}}

We can also take the coefficients of the polynomial form and represent each element as a q-ary M-tuple (a_0, a_1, a_2, ..., a_{m-1}).

The following table shows the primitive polynomial over GF(2).

Example: GF(8)

Table 8: Representation for the elements of GF(2^3) generated by $p(x) = 1+x+x^3$

Power Form	Polynomial Form			Binary 3-tuple	Decimal	Index
				LSB-MSB		
0	0			(0 0 0)	0	-1
1	1			(1 0 0)	1	0
α^1		α		(0 1 0)	2	1
α^2			α^2	(0 0 1)	4	2
α^3	1	α		(1 1 0)	3	3
α^4		α	α^2	(0 1 1)	6	4
α^5	1	α	α^2	(1 1 1)	7	5
α^6	1		α^2	(1 0 1)	5	6

Table 9: Representation for the elements of GF(2^4) generated by $p(x) = 1+x+x^4$

Power	Polynomial				Binary 4-tuple	decimal	index
0	0				(0 0 0 0)	0	-1
1	1				(1 0 0 0)	1	0
α^1		α			(0 1 0 0)	2	1
α^2			α^2		(0 0 1 0)	4	2
α^3				α^3	(0 0 0 1)	8	3
α^4	1	α			(1 1 0 0)	3	4
α^5		α	α^2		(0 1 1 0)	6	5
α^6			α^2	α^3	(0 0 1 1)	12	6
α^7	1	α		α^3	(1 1 0 1)	11	7
α^8	1		α^2		(1 0 1 0)	5	8
α^9		α		α^3	(0 1 0 1)	10	9
α^{10}	1	α	α^2		(1 1 1 0)	7	10
α^{11}		α	α^2	3	(0 1 1 1)	14	11
α^{12}	1	α	α^2	α^3	(1 1 1 1)	15	12
α^{13}	1		α^2	α^3	(1 0 1 1)	13	13
α^{14}	1			α^3	(1 0 0 1)	9	14

2.6 Basic Properites of Galois Field GF(p^m)

β^{p^l} is a conjugate of β.

If β is a primitive element of GF(p^m), all its conjugates are also primitive elements of GF(p^m).

If β is an element of order n in GF(p^m), all its conjugates have the same order n.

If β, an element in GF(p^m), is a root of a polynomial f(x) over GF(p), then all the distinct conjugates of β, also elements in GF(p^m), are roots of f(x).

The p^m-1 nonzero elements of GF(p^m) form all the roots of $x^{p^m-1} - 1 = 0$. The elements of GF(p^m) form all the roots of $x^{p^m} - x = 0$.

Let β be in GF(p^m), $\phi(x)$ is the minimal polynomial of β over GF(p), if $\phi(x)$ is the polynomial of smallest degree over GF(p) such that $\phi(\beta) = 0$.

Every element β of GF(p^m) has a unique minimal polynomial over GF(p).

The minimal polynpmial $\phi(x)$ of a field element β is irreducible.

An irreducible polynomial f(x) over GF(p) is the minimal polynomial of an element β in GF(p^m) if $f(\beta) = 0$.

Let $\phi(x)$ be the minimal polynomial of β, If β is a root of f(x), then f(x) is divisilbe by $\phi(x)$.

The minimal polynomial $\phi(x)$ of an element β in GF(p^m) divides $x^{p^m}-x$.

The minimal polynomial $\phi(x)$, which is also irreducible, of an element β in GF(p^m) is

$$\phi(x) = \prod_{i=0}^{e-1}(x+\beta^{p^l})$$

Let e be the degree of a minimal polynomial $\phi(x)$ of an element β in GF(p^m), then e is the smallest integer such that $\beta^{p^e} = \beta$, and $e \le m$, e divides m.

2.7 2.7. Vector Space

A set of vectors is said to span a vector space V if every vector in V is a linear combination of the vectors in the set. The set is called a basis of the vector space. The number of vectors in a basis of a vector space is called the dimension of the vector space.

Row space of a matrix: A space spanned by the linear combination of row vectors of a matrix G

For any kxn matrix G over GF(2) with k linearly independent rows, there exists an (n-k)xn matrix H over GF(2) with n-k linearly independent rows such that for any row g_i in G and any h_i in H, $g_i.h_i = 0$. The row space of G is the null (dual) space of H and vice versa.

A submatrix of a matrix G is a matrix that is obtained by striking out given rows or columns of G.

3 Linear Block Codes

Linear block codes is a subclass of block codes. Any combination of its code words is another code word in the code word space. An important subclass of linear block codes is linear systematic block codes. In addition to the linearity, a systematic block code is easy to encode and decode using linear sequential circuits such as shift registers.

3.1 Introduction

A block of length n and q^k code words is called a linear (n, k) code if and only if its q^k code words form a k-dimensional subspace of the vector space of all the n-tuples over the field GF(q).

Message units of fixed length k are transformed into a q-ary n-tuple code word or code vector v in the code space C.

$$\mathbf{v}_{1\infty n} = \mathbf{u}_{1\infty k}\,\mathbf{G}$$

Refer to the first example in the introduction

Example: A binary (7, 4) linear code.

$$\mathbf{G}_{4\infty 7} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1\ 1\ 0\ 1\ 0\ 0\ 0 \\ 0\ 1\ 1\ 0\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ 1\ 0\ 1\ 0\ 0\ 0\ 1 \end{bmatrix}$$

Table 10: A Binary (7, 4) linear block code

Messages	Code Words
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 1)	(0 1 1 0 1 0 1)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

Every code word v in C is a linear combination of k linearly independent vectors (they are code words as well) g_0, g_1, _, g_{k-1} in C.

The k∞n generator matrix G has k linear independent row vectors. G is said to generate or span the (n, k) linear code C.

A linear systematic block code consists of k unaltered information digits and n-k parity-check digits, which are linear sums of the information digits.

A linear systematic block code can be generated by a generator matrix $G = [P_{kx(n-k)} \ I_{kxk}]$, where $P_{kx(n-k)}$ generates the parity-check equations and I_{kxk} is the identity matrix.

The parity-check matrix H of a linear systematic block code generated by G above is $[I_{(n-k)x(n-k)} \ P^T_{(n-k)xk}]$.

Example: The parity check matrix of the systematic (7, 4) code.

$$\mathbf{H}_{3\infty 7} = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} 1\ 0\ 0\ 1\ 0\ 1\ 1 \\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\ 0\ 0\ 1\ 0\ 1\ 1\ 1 \end{bmatrix}$$

The row space of G span C, the row space of H span C_d, which is the dual code of C. Every vector in C_d is in the null space of G. C and C_d are said to be orthogonal subspaces of the 2^n space.

3.2 Syndrome and Error Detection

Syndrome is an n-k tuple calculated by

$$\mathbf{s} = (s_0, s_1, \_, s_{n-k-1}) = \mathbf{r} \cdot \mathbf{H}^T$$

$$= (\mathbf{v} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{v} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{0} + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T$$

$$= (r_0,\ r_1, \_, r_{n-k-1}, r_{n-k}, r_{n-k+1}, \_, r_{n-1}) \cdot \begin{bmatrix} I_{(n-k)x(n-k)} \\ P_{kx(n-k)} \end{bmatrix}$$

where r is the received vector

$$\mathbf{r} = \mathbf{v} + \mathbf{e}$$

e is the error vector.

Syndrome depends only on the error vector e.

Syndrome is the vector sum of the received parity digits (r_0, r_1, _, r_{n-k-1}) and the parity-check digits recomputed from the received information digits r_{n-k}, r_{n-k+1}, _, r_{n-1} .

Undetectable errors occur if and only if e is an code vector in C. Decoder will make a decoding error when an undetectable error patterns occurs.

3.3 Minimum Distance of A Block Code

The Hamming weight w(v) of a binary n-tuple code word v is the number of nonzero components in the code word.

The Hamming distance d(v, w) between two q-ary (n, k) code vectors v and w is the number of places in which they differ.

The Hamming distance d(v, w) is the Hamming weight w(v+w).

The minimum weight of a linear block code C is the minimum weight of its nonzero code words.

The minimum distance of a linear block code is equal to the minimum weight of its nonzero code words.

Let C be an (n, k) linear code with parity-check matrix H. For each code vector of Hamming weight l, there exists l columns of H such that the vector sum of these l columns is equal to the zero vector. Conversely, if there exist l columns of H whose vector sum is the zero vector, there exists a code vector of Hamming weight l in C.

The minimum weight w_{min} (or minimum distance d_{min}) of C is equal to the smallest number of columns of H that sum to 0.

3.4 Error-Detecting Capabilities of A Block Code

The random-error-detecting capability of a block code is d_{min} -1, that is it can detect any error pattern with d_{min} -1 or fewer error digits guaranteed.

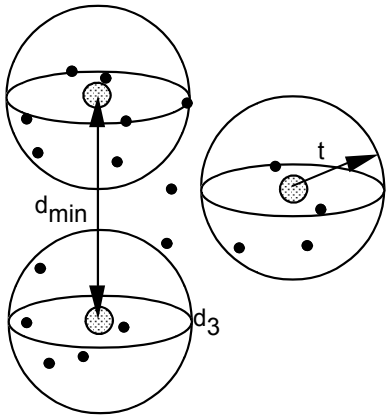

Figure 2: The error detection and correction capabilities based on decoding spheres

There 2^k-1 undetectable error patterns.

There are 2^n-2^k detectable error patterns.

Let Ai be the number of code vectors of weight i in C. The number s A_0, A_1, _, A_n are called the weight distribution of C.

The probablity of an undetected error is

$$P_u(E) = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i}$$

where p is the transition probablity of a BSC.

3.5 Error-Correcting Capabilities of A Block Code

The random-error-correcting capability of block code is t, that is it guarantees correcting all the error patterns of t or fewer digits.

$$t = \left[\frac{d_{min} - 1}{2} \right]$$

where [] denotes the largest integer.

A t-error-correcting (n, k) block code is capable of correcting a total of 2^{n-k} error patterns.

3.6 Standard Array Decoding

A standard array is a 2^{n-k} x 2^k vector array with each element being a n-tuple vector in the 2^n space. The 0th row consists of 2^k code vectors. The 0th column consists of 2^{n-k} nonzero (except the 0 vector) most likely (least weight in the case of a BSC) error vectors from the 2^n-2^k space.

The 2^{n-k} row of vector arrays are called coset.

The error vectors in the 0th column of a standard array (the leftmost vector in a coset) are *coset leaders*. Coset leaders are correctable error patterns.

Every (n, k) linear block code is capable of correcting 2^{n-k} error patterns.

Example: A (6, 3) linear code and its standard array.

000000	011100	101010	110001	110110	101101	011011	000111
100000	111100	001010	010001	010110	001101	111011	100111
010000	001100	111010	100001	100110	111101	001011	010111
001000	010100	100010	111001	111110	100101	010011	001111
000100	011000	101110	110101	110010	101001	011111	000011
000010	011110	101000	110011	110100	101111	011001	000101
000001	011101	101011	110000	110111	101100	011010	000110
100100	111000	001110	010101	010010	001001	111111	100011

The decoding based on the standard array is the minimum distance decoding (i.e. the maximum likelihood decoding).

The probablity of a decoding error is

$$P(E) = 1 - \sum_{i=0}^{n} e_i p^i (1-p)^{n-i}$$

where e_i is the weight distribution of the coset leaders

All the 2^k n-tuples of a coset have the same syndrome. The syndromes for different cosets are different.

3.7 Syndrome Decoding

The syndrome of a (n, k) code is an (n-k) tuple. There are 2^{n-k} distinct (n-k)-tuple corresponding to 2^{n-k} cosets (or coset leaders, or error patterns).

Syndrome decoding (also table-lookup decoding) is a table which stores 2^{n-k} entries of coset leaders (correctable error patterns) corresponding to every syndrome.

3.8 Hamming Codes

A Hamming code has the following properties:

code length:	$n = 2^m - 1$
number of information symbols:	$k = 2^m - m - 1$
number o parity-check symbols:	$m = n - k$
error-correcting capability:	$t = 1$
minimum distance:	$D_{min} = 3$

Derived properties:

The columns of H consist of all 2^m -1 nonzero m-tuples.
The coset leaders of a Hamming code consist of a zero vector and all the $(2^m - 1)$-tuple of weight 1.

A perfect code's standard array has all the error patterns of t or fewer errors and no others as its coset leaders.

Hamming codes form a class of single-error-correcting perfect codes.

Shortened Hamming codes are derived by deleting columns of parity-check matrix H.

A shortened Hamming has the same number of parity-check symbols but fewer information symbols to increase the minimum distance.

4 Cyclic Codes

Cyclic codes are a subclass of linear codes.

Encoding and decoding of cyclic codes are algorithmic and computationally efficient in contrast to the tabular decoding techinques for arbitrary linear codes.

Encoding and syndrome computation of cyclic codecs can be implemented by linear feedback shift registers (LFSR).

A (n, k) linear code C is called a cyclic code if every cyclic shift of a code vector in C is also a code vector in C. This does not mean every code vector have the same weight. There are groups of code vectors with the same weight.

4.1 Generator Matrices of Cyclic Codes

$$g(x) = 1 + g_1 x + g_2 x^2 + \ + g_{n-k-1} x^{n-k-1} + x^{n-k}$$

$$\mathbf{G}_{k \infty n} = \begin{bmatrix} g(x) \\ g^1(x) \\ \\ g^{k-1}(x) \end{bmatrix} = \begin{bmatrix} g_0 & g_1 & & g_{n-k} & 0 & 0 & 0 \\ 0 & g_0 & g_1 & & g_{n-k} & 0 & 0 \\ \\ 0 & 0 & 0 & g_0 & g_1 & & g_{n-k} \end{bmatrix}$$

where gi(x) is g(x) shifted to the right i times

Example: $g(x) = 1 + x + x^3$

$$\mathbf{G}_{4 \infty 7} = \begin{bmatrix} g^0 \\ g^1 \\ g^2 \\ g^3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Through row operations on generator matrix, we can get

$$\mathbf{G'}_{4 \infty 7} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

G' can be generated by dividing x^{n-k+i} with g(x)

$$x^{n-k+i} = a_i(x)\, g(x) + b_i(x)$$

$$b_i(x) = b_{i0} + b_{i1} x + b_{i2} x^2 + \ldots + b_{2i,n-k-1} x^{n-k-1}$$

$$\mathbf{G'}_{k\infty n} = \begin{bmatrix} b_0(x) \\ b_1(x) \\ \\ b_{k-1}(x) \end{bmatrix} =$$

$$\begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{0(n-k-1)} & 1 & 0 & 0 \\ b_{10} & b_{11} & b_{12} & b_{1(n-k-1)} & 0 & 1 & 0 \\ & & & & & & \\ b_{(k-1)0} & b_{(k-1)1} & b_{(k-1)2} & b_{(k-1)(n-k-1)} & 0 & 0 & 1 \end{bmatrix}$$

4.2 Parity-Check Matrices of Cyclic Codes

Theorem: g(x) divides x^n+1

$$x^n+1 = g(x)\, h(x)$$

$$h(x) = h_0 + h_1 x + \ldots + h_k x^k$$

The parity-check matrix corresponding to the generator matrix G is

$$\mathbf{H}_{(n-k)\infty k} = \begin{bmatrix} h_k & h_{k-1} & & h_0 & 0 & & 0 \\ 0 & h_k & h_{k-1} & & h_0 & 0 & \\ & & & & & & \\ 0 & 0 & & h_k & h_{k-1} & & h_0 \end{bmatrix}$$

The systematic parity-check matrix H' corresponding to G' can be obtained from G'

$$H' = \begin{bmatrix} 1 & 0 & 0 & b_{00} & b_{10} & b_{(k-1)0} \\ 0 & 1 & 0 & b_{01} & b_{11} & b_{(k-1)1} \\ & & & & & \\ 0 & 0 & 1 & b_{0(n-k-1)} & b_{1(n-k-1)} & b_{(k-1)(n-k-1)} \end{bmatrix}$$

4.3 Polynomial Representation of Cyclic Codes

There is one and only one nonzero code polynomial (or code vector) of minimum degree (n-k) in a (n, k) cyclic code. This polynomial is called generator polynomial and is of the form $g(x) = 1 + g_1 x + g_2 x^2 + \ + g_{n-k-1} x^{n-k-1} + x^{n-k}$

A binary polynomial of degree n-1 or less is a code ploynomial if and only if it is a multiple of generator polynomial g(x).

The generator polynomial g(x) of a (n, k) cyclic code is a factor of $x^n + 1$.

If g(x) is a polynomial of degree n-k and is a factor of $x^n + 1$, then g(x) generates a (n, k) cyclic code. For large n, $x^n + 1$ may have many factors of degree n-k. Some of these generate good cyclic codes and some generate bad codes.

A linear systematic cyclic code can be generated by

1. Multiply the message u(x) by x^{n-k}.
2. Obtain the remainder b(x) (the parity-check digits) from dividing $x^{n-k}u(x)$ by the generator matrix g(x).
3. Combine b(x) and $x^{n-k}u(x)$ to obtain the code polynomial $b(x) + x^{n-k}u(x)$.

The procedures above are equivalent to generating G' matrix

$$c(x) = u(x)\, g(x)$$

$$u(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_{k-2} x^{k-2} + u_{k-1} x^{k-1}$$

There are q^k possible code words.

Example: The parity-check matrix for a (7, 4) linear systematic code is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{bmatrix}$$

A code word c(x) can be represented by a code vector $c = (c_0, c_1, \ldots, c_{n-1})$

$$c\, H^t = 0$$

Which is equivalent to the following equation

$$\sum_{i=o}^{6} c_i \alpha^i = 0$$

That is α is a root of c(x), because c(x) is a multiple of g(x) and α is a root of g(x). This is the principle behind BCH code. Since BCH codes are constructed from the first 2t elements ($\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$) as roots, any code word should have these 2t roots. Words don't have these 2t roots are not code words, thus contain errors.

4.4 Syndrome Computation and Error Detection

An (n, k) cyclic code is capable of detecting any error burst of length n-k or less, including the end-around bursts.

4.5 Decoding of Cyclic Codes

Syndrome Computation: calculate the remainder of r(x) divided by g(x), can be achieved by a division circuit.

Association of syndrome to error pattern:

Error correction: by adding the error pattern to the received code vector.

4.6 Shorten Cyclic Codes

Given an (n, k) cyclic code C, the there is a subset of code vectors in which the l leading highest-order information digits are identical to zero. There are 2^{k-l} such code vectors and they form a linear subcode of C. If the l leading zero information digits are deleted from each of these code vectors, we obtain a set of 2^{k-l} vectors of length $n - l$.. These 2^{k-l} shortened vectors form an ($n-l, k-l$) linear code. This code is called a shortened cyclic code (or polynomial code) and is not cyclic.

Any systematic cyclic code can be shortened. Because the unused bits in a shortened code are always set to zero, they need not be transmitted, but the receiver can reinsert them and decode just as if the code were not shortened. If the original cyclic code has minimum distance d, then the shortened code has minimum distance d or larger. Similarly, if the original code can correct burst errors of length t, then the shortened code can also correct burst errors of length t or longer. Using the techniques of shortening and interleaving, a very large collection of good burst-error-correcting codes can be created from the unshortened cyclic codes.

5 BCH Codes

BCH codes are t-error-correcting codes with symbol field from GF(q) and error location field from GF(q^m). In general, m _ 1 for BCH codes, if m = 1, we give it another name -- Reed-Solomon codes.

Let p and m be given and β be any element of GF(q^m) of order n (i.e. $\beta^n = 1$). Then for any positive t and any integer j, the corresponding BCH code is the cyclic code of blocklength n with the generator polynomial

$$g(x) = \text{LCM}\,[f_{j_0}(x), f_{j_0+1}(x), \quad , f_{j_0+2t-1}(x)]$$

where $f_j(x)$ is the minimal polynomial of β^j.

The above equation is a so-called generalized BCH generator polynomial. Usually $j_0 = 1$ and β is the primitive element of GF(q^m) α. Therefore, g(x) will have $\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ as its roots. If the symbol field and the error-locator field are different, i.e., the error-locator field is an extension field of the symbol field, then g(x) will have more than 2t degree. The message length is k < n-2t. In the case of Reed-Solomon codes, the message word length is exactly n-2t. We say this is more efficient because the code rate k/n is larger for the same n.

Example: Construct an n = 15 , 2-error-correcting BCH code.

$$g(x) = \text{LCM}\,[f_1(x), f_2(x), f_3(x), f_4(x)]$$

Since $\alpha^1, \alpha^2, \alpha^4$ are conjugates

$$f_1(x) = f_2(x) = f_4(x)$$

$$g(x) = \text{LCM}\,[f_1(x), f_3(x)] = (x^4 + x + 1)\,(x^4 + x^3 + x^2 + x + 1)$$
$$= x^8 + x^7 + x^6 + x^4 + 1$$

Since g(x) has degree of 8, the message has length of 15 - 8 = 7. This is a (15, 7) double-error-correcting code.

$$c(x) = u(x)\, g(x)$$

$$u(x) = (u_0, u_1, u_2, \ldots, u_6)$$

5.1 The BCH Bound

Let g(x) be the generator polynomial of a cyclic code of length n over GF(q) and let α^{e_1}, α^{e_2}, ..., $\alpha^{e_{2t}}$ be the roots of g(x), possibly in an extension field, where α is an element of order n. The minimum distance of the code is greater than the largest number of consecutive integers modulo n in the set $e = (e_1, e_2, ..., e_{2t})$

5.2 Syndrome Computation and Error Detection

Let $v(x) = v_0 + v_1x + v_2x^2 + ... + v_{n-1}x^{n-1}$ be a code word polynomial of a t-error-correcting BCH code of length $n = 2^m-1$. Since αi is a root of g(x) , hence for v(x)

$$v(i) = v_0 + v_1\alpha^i + v_2\alpha^{2i} + ... + v_{n-1}\alpha^{(n-1)i} = 0 \qquad \text{for } 1 \le i \le 2t$$

We have 2t equations. In general,

$$(v_0, v_1, ..., v_{n-1}) \begin{bmatrix} 1 \\ \alpha^i \\ \alpha^{2i} \\ \alpha^{3i} \\ \\ \alpha^{(n-1)i} \end{bmatrix} = 0 \qquad \text{for } 1 \le i \le 2t)$$

$$H' = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & (\alpha^2)^3 & (\alpha^2)^{n-1} \\ & & & & \\ 1 & \alpha^{2\tau} & (\alpha^{2t})^2 & (\alpha^{2t})^3 & (\alpha^{2t})^{n-1} \end{bmatrix}$$

Let r(x) be the received vector,

$$r(x) = r_0 + r_1x + r_2x^2 + ... + r_{n-2}x^{n-2} + r_{n-1}x^{n-1}$$
$$r(x) = v(x) + e(x)$$
$$\mathbf{s} = \mathbf{r}\,\mathbf{H}^t$$
$$= \mathbf{v}\,\mathbf{H}^t + \mathbf{e}\,\mathbf{H}^t$$
$$= \mathbf{e}\,\mathbf{H}^t$$
$$= (s_1, s_2, ..., s_{2t})$$

where

$$s_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + ... + r_{n-2}\alpha^{(n-2)i} + r_{n-1}\alpha^{(n-1)i}$$
$$r(x) = a_i(x)\,\varphi_i(x) + b_i(x)$$

$\varphi_i(x)$ is the minuimal polynomial of α^i

$$s_i = r(\alpha^l) = b_i(\alpha^l)$$

Thus the syndorm component s_i can be obtained by evaluating b(x) with $x = \alpha^l$.

Example: A double-error-correcting (15, 7) BCH code.

Suppose r = (1 0 0 0 0 0 0 0 1 0 0 0 0 0 0) = $1 + x^8$

$$g(x) = \text{LCM}\,[f_1(x), f_2(x), f_3(x), f_4(x)]$$

Since $\alpha^1, \alpha^2, \alpha^4$ are conjugates

$$f_1(x) = f_2(x) = f_4(x)$$

$$g(x) = \text{LCM}\,[f_1(x), f_3(x)] = (x^4 + x + 1)\,(x^4 + x^3 + x^2 + x + 1)$$
$$= x^8 + x^7 + x^6 + x^4 + 1$$

$$b_1(x) = x^2$$

$$b_3(x) = 1 + x^3$$

$$s_1 = \alpha^2$$

$$s_2 = \alpha^4$$

$$s_4 = \alpha^8$$

$$s_3 = 1 + \alpha^9 = 1 + \alpha + \alpha^3 = \alpha^7$$

$$\mathbf{S} = (\alpha^2, \alpha^4, \alpha^7, \alpha^8)$$

$$s_i = r(\alpha^i) = v(\alpha^i) + e(\alpha^i)$$
$$= e(\alpha^i)$$

where e(x) is the error pattern polynomial.

$$e(x) = x^{j1} + x^{j2} + x^{j3} + \_ + x^{jv} \qquad 0 \le j_1 < j_2 < \_ < j_v \le n$$

$$S_1 = r(\alpha) = \alpha^{j1} + \alpha^{j2} + \alpha^{j3} + \ldots + \alpha^{jv}$$

$$S_2 = r(\alpha^2) = (\alpha^{j1})^2 + (\alpha^{j2})^2 + (\alpha^{j3})^2 + \ldots + (\alpha^{jv})^2$$

$$S_{2t} = r(\alpha^{2\tau}) = (\alpha^{j1})^{2\tau} + (\alpha^{j2})^{2\tau} + (\alpha^{j3})^{2\tau} + ... + (\alpha^{jv})^{2\tau}$$

where α^{ji} are called error location number. In general, there can be many possible solutions to the above equations. Each solution yields a different error pattern. If the number of errors in the actual error pattern is t or less, the solution that yields an error pattern with the samllest number of errors is the right solution. Any mothod to solve these equations is called a BCH decoding algorithm.

Let $\beta_i = \alpha^{ji}$

$$S_1 = r(\alpha) = \beta_1 + \beta_2 + \beta_3 + ... + \beta_v$$

$$S_2 = r(\alpha^2) = \beta_1^2 + \beta_2^2 + \beta_3^2 + ... + \beta_v^2$$

$$S_{2t} = r(\alpha^{2\tau}) = \beta_1^{2t} + \beta_2^{2t} + \beta_3^{2t} + ... + \beta_v^{2t}$$

$$\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x)....(1 + \beta_v x) = 1 + \sigma_1 x + \sigma_2 x^2 + \quad + \sigma_v x^v$$

$$\sigma_0 = 1$$

$$\sigma_1 = \beta_1 + \beta_2 + \beta_3 + ... + \beta_v$$

$$\sigma_2 = \beta_1\beta_2 + \beta_2\beta_3 + ... + \beta_{v-1}\beta_v$$

$$\sigma_v = \beta_1\ \beta_2\ \beta_3\ ..\ \beta_v$$

The roots of $\sigma(x)$ are $\beta_1^{-1}, \beta_2^{-1}, .., \beta_v^{-1}$ which are the inverses of the error location number. $\sigma(x)$ is called the *error-location polynpmial*. σ_i's are related to syndrome components si by the *Newton's Identities*.

$$s_1 + \sigma_1 = 0$$
$$s_2 + \sigma_1 s_1 + 2\sigma_2 = 0$$
$$s_3 + \sigma_1 s_2 + \sigma_2 s_1 + 3\sigma_3 = 0$$

$$s_v + \sigma_1 s_{v-1} + \ + \sigma_{v-1} s_1 + v\sigma_v = 0$$
$$s_{v+1} + \sigma_1 s_v + \ + \sigma_{v-1} s_2 + \sigma_v s_1 = 0$$

In a binary field

$$i\sigma_i = \textbf{Error!})$$

We can find σ_i from these equations and thus the eroro location numbers β_i by solvoing $\sigma(x)$. There may be many solutions, but we want to find a $\sigma(x)$ of minimal degree which will produce an error pattern with minimum number of errors. If $v \leq t$, this $\sigma(x)$ will give the actual error pattern.

In summary, the error-correcting procedure for BCH codes consists of three major steps.

1. Compute the syndrome $S = (s_0, s_1, \_, s_{2t})$ from the received r(x).
2. Determine the error-location polynomial $\sigma(x)$ from syndrome components using Newton's Identities.
3. Determine the error-location number $\beta_1, \beta_2, \beta_3, \ldots, \beta_v$ by finding the roots of $\sigma(x)$ and correct the errors in r(x)

Step 2 is the most complicated part of decoding a BCH code.

5.3 Reed-Solomon Codes

Reed-solomon codes are random single- or multiple-symbol error correcting codes. operating on symbols which are elements of a finite field. The coefficients of the data polynomial and the check symbols are elements of the field, and all encoding, decoding , and correction computations are performed in the field. Reed-Solomon are symbol oriented and the circuits implementing them are typically colocked once per data symbol, although bit-serial techniques are also employed.

Reed-solomon codes are used as part of the forward-error-correction strategy of digital transmission of signals in the four proposed all digital HDTV systems. The following table summaries the length of a code block in bytes[1], the length of parity-check bytes, and the error-correcting capability of these digital HDTV systems.

[1]Here one symbol is equvalent to one byte, since the symbols are elements of GF(256).

Table 11: Reed-Solomon codes used in HDTV systems.

HDTV Systems	Code Structure[2]	Check Bytes	T[3]
ADTV	(147, 127)	20	10
CCDC	(158, 148) (32 QAM) (116, 106) (16 QAM)	10 10	5 5
Digi-Cipher	(155, 145) (32 QAM) (116, 106) (16 QAM)	10 10	5 5
DSC	(167, 147)	20	10

The length of the parity-check symbols are twice the error-correcting capability. This is an unique property of Reed-Solomon codes and is a result of having a generator polynomial of degree 2t.

Reed-Solomon codes is a subset of BCH codes which in tern, is a subset of cyclic codes. The symbol field GF(q) and the error locator field GF(q^m) are the same, i.e.

$$m = 1$$
$$n = q^m - 1 = q - 1$$

Example: The symbols in the original messages are 8-bit numbers, i.e. $q = 2^8 = 256$ ∅ error correction on bytes. The length of the block is 255.

Because the symbol field and the error-locator field are the same field, all minimal polynomials are linear. This is an important property in that it results in a generator polynomial of degree 2t (number of parity-check symbols). Unlike other BCH codes in which the parity-check symbols are usually more than 2t, Reed-Solomon codes are more efficient. That is, Reed-Solomon codes contain more message symbols for the same block length and error-correcting capability.

The minimal polynomial over GF(q) of an element in the same field is

$$f_i(x) = x - \alpha^i$$

and the generator polynomial of a BDH code is derived from the following

$$g(x) = \text{LCM}\ [f_{j0}(x), f_{j0+1}(x), \quad , f_{j0+2t-1}(x)]$$

---

[2]A code block of n total symbols and k message symbols is written as a (n, k) code.

[3]T is the number of symbol errors the code can correct, regardless how many bits in a symbol.

Therefore, for a t-error-correcting Reed-Solomon code, the generator polynomial is

$$g(x) = \prod_{i=0}^{2t-1}(x - \alpha^{j0+i})$$

$$= g_0 + g_1x + g_2x^2 + ... + g_{2t-1}x^{2t-1} + g_{2t}x^{2t}$$

where $j_0 = 1$ is convensional

The code generated is a (n, k) cyclic code, where k = n - 2t.

Let the message to be encoded be u(x)

$$u(x) = u_0 + u_1x + u_2x^2 + ... + u_{k-2}x^{k-2} + u_{k-1}x^{k-1}$$

$$u_i \in GF(q)$$

In systematic form, the 2t parity-check symbols are the coefficients of the remainder

$$b(x) = b_0 + b_1x + b_2x^2 + ... + b_{2t-1}x^{2t-1}$$

resulting from dividing $x^{2t}u(x)$ by the generator polynomial g(x).

Let v(x) be a code word, r(x) be a received word, and e(x) be the error pattern

$$e(x) = r(x) - v(x)$$

$$= e_0 + e_1x + e_2x^2 + ... + e_{n-2}x^{n-2} + e_{n-1}x^{n-1}$$

$$e_i \in GF(q)$$

Suppose e(x) has v errors at locations $x^{j1}, x^{j2}, ..., x^{jv}$, then

$$e(x) = e_{j1}x^{j1} + e_{j2}x^{j2} + e_{j3}x^{j3} + ...+ e_{jv}x^{jv}$$

Let $\beta_i = \alpha^{ji}$

and following the BCH decoding procedures

$$S_1 = r(\alpha) = e_{j1}\alpha^{j1} + e_{j2}\alpha^{j2} + e_{j3}\alpha^{j3} + ...+ e_{jv}\alpha^{jv}$$

$$S_2 = r(\alpha^2) = e_{j1}(\alpha^{j1})^2 + e_{j2}(\alpha^{j2})^2+ e_{j3}(\alpha^{j3})^2+ ...+ e_{jv}(\alpha^{jv})^2$$

$$S_{2t} = r(\alpha^{2\tau}) = e_{j1}(\alpha^{j1})^{2t} + e_{j2}(\alpha^{j2})^{2t} + e_{j3}(\alpha^{j3})^{2t} + ... + e_{jv}(\alpha^{jv})^{2t}$$

can be reformulated as

$$S_1 = r(\alpha) = e_{j1}\beta_1 + e_{j2}\beta_2 + e_{j3}\beta_3 + ... + e_{jv}\beta_v$$

$$S_2 = r(\alpha^2) = e_{j1}\beta_1^2 + e_{j2}\beta_2^2 + e_{j3}\beta_3^2 + ... + e_{jv}\beta_v^2$$

$$S_{2t} = r(\alpha^{2\tau}) = e_{j1}\beta_1^{2t} + e_{j2}\beta_2^{2t} + e_{j3}\beta_3^{2t} + ... + e_{jv}\beta_v^{2t}$$

Since $f_i(x) = x - \alpha^i$

$$r(x) = c_i(x)\,(x - \alpha^i) + b_i(x)$$

$$b_i(x) = b_i$$

since $f_i(x)$ is of degree 1.

$$S_i = b_i$$

To find error-location polynomial

$$\sigma(x) = (1 + \beta_1 x)\,(1 + \beta_2 x)....(1 + \beta_v x)$$

$$= 1 + \sigma_1 x + \sigma_2 x^2 + \quad + \sigma_v x^v$$

Now we can use Berlekamp's iterative algorithm to find the error-location polynomial. The error values are determined by

$$e_{ji} = \frac{Z(\beta_\iota^{-1})}{\prod_{j=1\ j\_i}^{v}(1 + \beta_j\beta_\iota^{-1})}$$

$$Z(x) = 1 + (\,s_1 + \sigma_1\,)x + (\,s_2 + \sigma_1 s_1 + \sigma_2\,)x^2 + ... + (\,s_v + \sigma_1 s_{v-1} + \ + \sigma_{v-1} s_1 + \sigma_v\,)x^v$$

6 REFERENCE

Shu Lin and Daniel Costello, Jr. “Error Control Coding”, Prentice Hall, 1983.

Rechard E. Blahut, “Theory and Practice of Error Control Codes”, Addison Wesley, 1984.

Neal Glover and Trent Dudley, “Practical Error Correction Design for Engineers” 2nd ed., DST, 1988.

W. Weslet Peterson and E. J. Weldon, Jr. “Error-Correcting Codes”, 2nd ed., MIT Press, 1972.