

Galois Fields (GF): Evariste Galois (galwa): October 25, 1811-May 31, 1832



Introduction to Finite Fields

- Everyone knows how to add, subtract, multiply and divide real numbers (assuming we are not dividing by 0).
- If for a set of numbers we can add, subtract, multiply and divide (except by 0) then that set of numbers is called a **field**.
- If the number of elements in the field is finite, it is called a **FINITE FIELD**.
- Note this implicitly requires that there be a number 0 that is the difference of two identical elements and a number 1 that is the ratio of a number with itself.
- So each number has an additive inverse and each number (except 0) has a multiplicative inverse.
 - The additive inverse of a is $-a$.
 - The multiplicative inverse of a is a number b such that $ab = 1$ or $b^{-1} = a$.

Examples of Finite Fields

- Consider the set $\{0, 1\}$ with addition and multiplication done mod 2.

+	0	1
0	0	1
1	1	0

\times	0	1
0	0	0
1	0	1

- The additive inverse of 1 is 1. That is $-1=1$. The multiplicative inverse of 1 is 1.
- This is a FINITE FIELD

Examples of Finite Fields

Consider the set $\{0, 1, 2\}$ with addition and multiplication done mod 3.

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

×	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

The additive inverse of 1 is 2. That is $-1=2$. The multiplicative inverse of 1 is 1. The multiplicative inverse of 2 is 2. Every nonzero number has a multiplicative inverse. This is a FINITE FIELD.

$$\frac{1}{2} = 1(2^{-1}) = (\cdot 2 = 2$$

Examples of NOT a Finite Field

Consider the set $\{0, 1, 2, 3\}$ with addition and multiplication done mod 4.

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

\times	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

- The additive inverse of 1 is 3. That is $-1=3$. The additive inverse of 2 is 2.
- The multiplicative inverse of 2 does not exist. That is there is no number in $\{0, 1, 2, 3\}$ that when multiplied by 2 gives 1.
- So this is NOT a finite field.

$1/2$ — not defined
 2^{-1} doesn't exist

Examples of Finite Fields

Consider the set $\{0, 1, 2, 3, 4\}$ with addition done mod 5.

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

- The additive inverse of 1 is 4. That is $-1=4$, $-2=3$, $-3=2$, and $-4=1$.
- The multiplicative inverse of 2 is 3. The multiplicative inverse of 3 is 2. The multiplicative inverse of 4 is 4.
- We can divide! For example $\frac{2}{3} = 2 \div 3 = 2 \times (3^{-1}) = 2 \times 2 = 4$
- This is a finite field.
- Fact: The set of elements $\{0, 1, \dots, p-1\}$ where p is prime with addition and multiplication done mod p is a finite field.

2 3 4 7

Examples of Finite Fields


- Electrical engineers know how to add, subtract, multiply, and divide vectors of length 2 with the components being real numbers.

$$(a, b) + (c, d) = (a + c, b + d)$$

$$(a, b) - (c, d) = (a - c, b - d)$$

$$(a, b) \times (c, d) = (ac - bd, ad + bc)$$

$$(a, b)/(c, d) = \left(\frac{ac + bd}{c^2 + d^2}, \frac{bc - ad}{c^2 + d^2} \right)$$



$$(a + jb)(c + jd) = ac - bd + j(bc + ad)$$

Examples of Finite Fields

- But where did the last two equations come from?
- They came from actually representing the numbers as complex numbers (so the first part is the real part and the second part is the imaginary part).
- So

$$\begin{aligned}(a + jb)(c + jd) &= ac + j(ad + bc) + j^2bd \\ &= ac - bd + j(ad + bc)\end{aligned}$$

because $j^2 = -1$. That is, j is the solution of the equation $x^2 = -1$.

Examples of Finite Fields

- Similarly

$$\begin{aligned}
 (a + jb)/(c + jd) &= \frac{(a + jb)(c - jd)}{(c + jd)(c - jd)} = \frac{ac + j(bc - ad) - j^2 bd}{c^2 + d^2} \\
 &= \frac{ac + bd}{c^2 + d^2} + j \frac{bc - ad}{c^2 + d^2} \\
 &= \left(\frac{ac + bd}{c^2 + d^2}, \frac{bc - ad}{c^2 + d^2} \right)
 \end{aligned}$$

- Note that it is much easier to multiply and divide in the “log” domain.
- That is, if we represent numbers in polar coordinates
 $(a + jb) = r_1 e^{j\theta_1}$ and $(c + jd) = r_2 e^{j\theta_2}$ then
 $(a + jb)(c + jd) = r_1 r_2 e^{j(\theta_1 + \theta_2)}$ and
 $(a + jb)/(c + jd) = (r_1/r_2) e^{j(\theta_1 - \theta_2)}$

Introduction to Finite Fields

- Now let us repeat this for the case where instead of the components of the vector being real numbers the components are just 0 and 1.
- The underlying addition is done mod 2. That is, $0 + 0 = 0$, $0 + 1 = 1$, $1 + 1 = 0$.
- The equation $x^2 + x + 1 = 0$ does not have a solution from the set $\{0, 1\}$.
- So, just like the complex number case, let's make up a solution and call the solution α . That is $\alpha^2 + \alpha + 1 = 0$ or $\alpha^2 = \alpha + 1$.
- So numbers have a “real” part and an “imaginary” part. So the number $a + \alpha(b)$ is a “complex” number with real part a and imaginary part b and can be written as a vector (a, b) of length 2.

Addition and Subtraction

- Now, how do we add, subtract, multiply and divide vectors of length two?
- First, note that the only possible vectors are $\{(0, 0), (1, 0), (0, 1), (1, 1)\}$.
- So there are a finite number of elements in this field.
- For addition and subtraction the process is to do component-wise addition and subtraction mod 2. Note that subtraction is the same as addition with binary vectors.

Multiplication

- For multiplication we repeat the same process used for complex numbers.

$$\begin{aligned}(a, b) \times (c, d) &= (a + \alpha b)(c + \alpha d) \\&= ac + \alpha(ad + bc) + \alpha^2 bd \\&= ac + \alpha(ad + bc) + (\alpha + 1)bd \\&= ac + bd + \alpha(ad + bc + bd) \\&= (ac + bd, ad + bc + bd)\end{aligned}$$

Division

- Now consider division. Note that we can not divide by $(0, 0)$.
- If we divide by $(1, 0)$ we get the same vector.
- So what if we divide by $(c, 1)$ where c is 0 or 1?

$$\begin{aligned}
 (a + \alpha b)/(c + \alpha) &= \frac{(a + \alpha b)(1 + c + \alpha)}{(c + \alpha)(1 + c + \alpha)} \\
 &= \frac{(a + ac + \alpha(a + b + bc) + \alpha^2 b)}{c(1 + c) + \alpha(1 + c + c) + \alpha^2} \\
 &= \frac{(a + ac + \alpha(a + b + bc) + (\alpha + 1)b)}{\alpha + \alpha + 1} \\
 &= (a + b + ac) + \alpha(a + bc) \\
 &= (a + b + ac, a + bc)
 \end{aligned}$$

Introduction to Finite Fields

We can build a table for addition and multiplication in this example.

+	$0 = (0, 0)$	$1 = (1, 0)$	$0 + \alpha = (0, 1)$	$1 + \alpha = (1, 1)$
$0 = (0, 0)$	0	1	α	$1 + \alpha$
$1 = (1, 0)$	1	0	$1 + \alpha$	α
$0 + \alpha = (0, 1)$	α	$1 + \alpha$	0	1
$1 + \alpha = (1, 1)$	$1 + \alpha$	α	1	0

\times	$0 = (0, 0)$	$1 = (1, 0)$	$0 + \alpha = (0, 1)$	$1 + \alpha = (1, 1)$
$0 = (0, 0)$	0	0	0	0
$1 = (1, 0)$	0	1	α	$1 + \alpha$
$0 + \alpha = (0, 1)$	0	α	$1 + \alpha$	1
$1 + \alpha = (1, 1)$	0	$1 + \alpha$	1	α

Introduction to Finite Fields

We can represent the field by powers of α and 0.

00	0	$\alpha^{-\infty}$
10	1	α^0
01	α^1	α^1
11	$\alpha + 1$	α^2

This is a FINITE FIELD OF SIZE 4. Consider

$$\begin{aligned}
 \alpha^3 &= \alpha^2(\alpha) \\
 &= (\alpha + 1)(\alpha) \\
 &= \alpha^2 + \alpha \\
 &= (\alpha + 1) + \alpha \\
 &= 1
 \end{aligned}$$

So $\alpha \times \alpha^2 = \alpha^3 = 1$. Similarly $\alpha^2 \times \alpha^2 = \alpha^4 = \alpha^3 \alpha = \alpha$.

Finite Fields

- We can also do arithmetic on vectors of length 3 with component being 0 or 1.
- The equation $x^3 + x + 1 = 0$ does not have a solution in the binary field.
- So let us call α the solution.
- So a binary vector of length 3 corresponds to a polynomial in α of degree 2 ($a\alpha^2 + b\alpha + c$).
- In this case we let the first component of the vector represent the coefficient of α^2 , the second component as the coefficient of α and the third component as the constant.

$$(1, 1, 0) = 1\alpha^2 + 1\alpha^1 + 0$$

$$(1, 0, 1) = 1\alpha^2 + 0\alpha^1 + 1$$

Finite Fields

000	0	$\alpha^{-\infty}$
001	1	α^0
010	α^1	α^1
011	$\alpha + 1$	α^3
100	α^2	α^2
101	$\alpha^2 + 1$	α^6
110	$\alpha^2 + \alpha$	α^4
111	$\alpha^2 + \alpha + 1$	α^5

Arithmetic

$$\begin{aligned}
 \alpha^7 &= \alpha^6 \alpha \\
 &= (\alpha^2 + 1) \alpha \\
 &= \alpha^3 + \alpha \\
 &= \alpha + 1 + \alpha \\
 &= 1
 \end{aligned}$$

- α is said to be a primitive root of the field because all nonzero elements of the field are generated by powers of α .
- Each nonzero element has a multiplicative inverse.
 $(\alpha^n)^{-1} = \alpha^7 \alpha^{-n} = \alpha^{7-n}$. For example,
 $(\alpha + 1)^{-1} = (\alpha^3)^{-1} = \alpha^7 \alpha^{-3} = \alpha^4 = (\alpha^2 + \alpha)$.

Arithmetic

$$\begin{aligned}(\alpha^2 + 1)(\alpha + 1) &= \alpha^6 \alpha^3 \\ &= \alpha^9 \\ &= \alpha^7 \alpha^2 \\ &= \alpha^2\end{aligned}$$

$$\begin{aligned}(\alpha^2 + 1)(\alpha^2 + 1) &= \alpha^6 \alpha^6 \\ &= \alpha^{12} \\ &= \alpha^7 \alpha^5 \\ &= \alpha^5 \\ &= \alpha^2 + \alpha + 1\end{aligned}$$

Factorization

In GF(2) the polynomial $x^7 - 1$ factors as

$$x^7 - 1 = (x - 1)(x^3 + x^2 + 1)(x^3 + x + 1).$$

If we define α to be a root of $x^3 + x + 1 = 0$ (that is $\alpha^3 = \alpha + 1$) then we can actually factor $x^7 - 1$ over GF(8) as follows:

$$\begin{aligned} x^3 + x + 1 &= (x - \alpha)(x - \alpha^2)(x - \alpha^4) \\ x^3 + x^2 + 1 &= (x - \alpha^3)(x - \alpha^6)(x - \alpha^5) \\ x^7 - 1 &= (x - 1)(x - \alpha)(x - \alpha^2)(x - \alpha^4) \\ &\quad (x - \alpha^3)(x - \alpha^6)(x - \alpha^5) \end{aligned}$$

Hamming Code: Revisited

The Hamming code has generator polynomial

$$\begin{aligned}g(x) &= x^3 + x^2 + 1 \\ &= (x - \alpha^3)(x - \alpha^5)(x - \alpha^6)\end{aligned}$$

Codewords are multiples of the generator polynomial

$c(x) = m(x)g(x)$. Thus the codewords also have roots $\alpha^3, \alpha^5, \alpha^6$.

Hamming Code: Revisited

Suppose the message polynomial is $m(x) = 1 + x$. The codeword is then

$$\begin{aligned} c(x) &= m(x)g(x) \\ &= 1 + x + x^2 + x^4 \end{aligned}$$

Note that

$$\begin{aligned} c(\alpha^3) &= 1 + \alpha^3 + \alpha^6 + \alpha^{12} = 1 + \alpha^3 + \alpha^6 + \alpha^5 \\ &= 1 + (\alpha + 1) + (\alpha^2 + 1) + (\alpha^2 + \alpha + 1) \\ &= 0 \end{aligned}$$

$$\begin{aligned} c(\alpha^5) &= 1 + \alpha^5 + \alpha^{10} + \alpha^{20} = 1 + \alpha^5 + \alpha^3 + \alpha^6 \\ &= 0 \end{aligned}$$

$$\begin{aligned} c(\alpha^6) &= 1 + \alpha^6 + \alpha^{12} + \alpha^{24} = 1 + \alpha^6 + \alpha^5 + \alpha^3 \\ &= 0 \end{aligned}$$

Hamming Code: Revisited

Suppose the received signal is

$$r(x) = c(x) + e(x)$$

If only one error occurred then $e(x) = e_j x^j$. So

$$S_1 = r(\alpha^5) = e(\alpha^5) = e_j \alpha^{5j}$$

$$S_2 = r(\alpha^6) = e(\alpha^6) = e_j \alpha^{6j}$$

Let $Y_1 = e_j$ and $X_1 = \alpha^j$. Then

$$S_1 = Y_1 X_1^5$$

$$S_2 = Y_1 X_1^6$$

So $X_1 = S_2/S_1$ and $Y_1 = S_1/X_1^5$.

Hamming Code: Revisited

Suppose the received signal is

$$r(x) = c(x) + e(x) = 1 + x + x^2 + x^4 + x^6$$

$$\begin{aligned} S_1 &= r(\alpha^5) = 1 + \alpha^5 + \alpha^{10} + \alpha^{20} + \alpha^{30} \\ &= 1 + \alpha^5 + \alpha^3 + \alpha^6 + \alpha^2 \\ &= \alpha^2 \end{aligned}$$

$$\begin{aligned} S_2 &= r(\alpha^6) = 1 + \alpha^6 + \alpha^{12} + \alpha^{24} + \alpha^{36} \\ &= 1 + \alpha^5 + \alpha^3 + \alpha^6 + \alpha^1 \\ &= \alpha \end{aligned}$$

Hamming Code: Revisited

- So $X_1 = S_2/S_1 = \alpha/(\alpha^2) = \alpha^{-1} = \alpha^6$.
- So the error occurred in location 6.
- Now $Y_1 = S_1/(X_1^5) = \alpha^2/(\alpha^6)^5 = \alpha^2/(\alpha^{30}) = \alpha^2/(\alpha^2) = 1$.
- Since we are transmitting information on a binary symmetric channel, we already know what the error should be (it has to be a 1), but we don't know where the error occurred.

Length 15 codes

The factorization of $x^{15} - 1$ over $\text{GF}(2)$ is

$$x^{15} - 1 = (x - 1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x^4 + x^3 + 1).$$

Consider the field $\text{GF}(16)$ using polynomial $x^4 + x + 1$. Let α be the root of $x^4 + x + 1$. Then we have the following factorization of these polynomials over $\text{GF}(16)$.

$$\begin{aligned} x^4 + x + 1 &= (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8) \\ x^4 + x^3 + x^2 + x + 1 &= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^9) \\ x^2 + x + 1 &= (x - \alpha^5)(x - \alpha^{10}) \\ x^4 + x^3 + 1 &= (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{13})(x - \alpha^{11}) \\ x^{15} - 1 &= \prod_{i=0}^{14} (x - \alpha^i) \end{aligned}$$

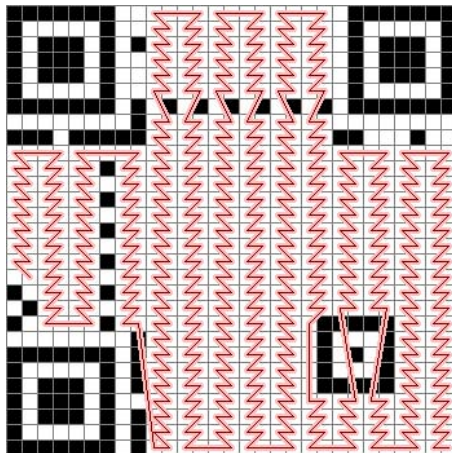
Reed-Solomon Codes



Reed-Solomon Codes



Reed-Solomon Codes



QR v3 order, from bottom right

Error Correction with QR Codes



Error Correction with QR Codes



Error Correction with QR Codes



Error Correction with QR Codes



Error Correction with QR Codes



Error Correction with QR Codes



Error Correction with QR Codes



Error Correction with QR Codes



Error Correction with QR Codes



Reed-Solomon Codes

- Reed-Solomon codes are widely used in digital communication systems.
- An (N, K) Reed-Solomon code over $GF(2^l)$ is a code with K information symbols that determine N coded symbols.
- Each symbol is an element of $GF(2^l)$.
- For Reed-Solomon codes we require that $N \leq 2^l + 1$. Standard RS codes have $N = 2^l - 1$. That is the length is one less than the field size.
- For example, the compact disc player uses two Reed-Solomon codes.
- These codes are nonbinary. That is, the code symbols are from an alphabet of size larger than two.

Reed-Solomon Codes

- For example, the (255,223) NASA standard code contains 223 information symbols that are 8 bit bytes. So one codeword contains 223 times 8 or 1784 information bits. These 223 information bytes or symbols are mapped by the encoder into 255 code symbols which are again eight bit bytes or 2040 bits.
- Reed-Solomon codes are linear codes so the parity check equations that apply to linear codes also apply to Reed-Solomon codes.

$$d_{\min} = N - K + 1 \quad (\because \text{MDS code})$$

- These codes are used in compact digital discs, spread-spectrum systems, computer memories.

(7,4) Reed-Solomon Code

For example the (7,4) Reed-Solomon code has symbols that are three bit bytes. Let us represent these symbols in the following way. This representation is called a finite field. (In this case the equation $x^3 + x + 1 = 0$ does not have a solution in the set $\{0, 1\}$). We let α be that solution.

000	0	$\alpha^{-\infty}$
001	1	α^0
010	α^1	α^1
011	$\alpha + 1$	α^3
100	α^2	α^2
101	$\alpha^2 + 1$	α^6
110	$\alpha^2 + \alpha$	α^4
111	$\alpha^2 + \alpha + 1$	α^5

Notice that $\alpha^7 = 1$ so when multiplying two elements of this field we can add the exponents of α modulo 7.

(7,4) Reed-Solomon Code

So when doing arithmetic with 3 bit bytes, adding two elements corresponds to bit-wise modulo two addition of the components and multiplication is best done by representing the two symbols as a power of α and then adding the exponents modulo 7.

(7,4) Reed-Solomon Code

The encoding is done as follows. There are four information symbols i_1, i_2, i_3, i_4 . There are three parity symbols p_1, p_2, p_3 . Each of these is a three bit byte.

The parity symbols are determined from the following equation.

$$\begin{aligned}p_1 &= \alpha^6 i_1 + \alpha^6 i_2 + \alpha^3 i_3 + \alpha i_4 \\p_2 &= \alpha i_1 + \alpha^2 i_2 + \alpha^4 i_3 + i_4 \\p_3 &= \alpha^6 i_1 + \alpha^5 i_2 + \alpha^5 i_3 + \alpha^2 i_4\end{aligned}$$

(7,4) Reed-Solomon Code

For example, the four information symbols

$i_1 = \alpha^5, i_2 = \alpha^3, i_3 = \alpha^2, i_4 = \alpha^0$ are mapped into the parity bits $p_1 = \alpha^5, p_2 = \alpha^4, p_3 = 1$. Thus the information bits (111 011 100 001) are mapped into the channel bits (111 011 100 001 111 110 001) for transmission.

(7,5) Reed-Solomon Code over GF(8)

The (7,5) code over GF(8) has 5 information symbols and 2 parity check symbols. Let (i_0, i_1, \dots, i_4) be the information symbols. The parity symbols are determined from the following equation.

$$\begin{aligned} p_1 &= \alpha^4 i_0 + i_1 + \alpha^5 i_2 + \alpha^5 i_3 + \alpha^4 i_4 \\ p_0 &= \alpha^3 i_0 + i_1 + \alpha^3 i_2 + \alpha i_3 + \alpha i_4 \end{aligned}$$

The codeword is

$$c(x) = p_0 + p_1 x + i_0 x^2 + i_1 x^3 + i_2 x^4 + i_3 x^5 + i_4 x^6$$

(7,5) Reed-Solomon Code

Note that $c(\alpha) = c(\alpha^2) = 0$. Thus the codewords are multiples of $g(x) = (x - \alpha)(x - \alpha^2)$.

If $r(x)$ is the received polynomial then we can determine the syndromes

$$S_1 = r(\alpha) = c(\alpha) + e(\alpha) = e(\alpha)$$

$$S_2 = r(\alpha^2) = c(\alpha^2) + e(\alpha^2) = e(\alpha^2)$$

(7,5) Reed-Solomon Code

Suppose that just one error occurred. Then

$$e(x) = e_j x^j$$

so that

$$\begin{aligned} S_1 = e(\alpha) &= e_j \alpha^j = Y_1 X_1 \\ S_2 = e(\alpha^2) &= e_j \alpha^{2j} = Y_1 X_1^2. \end{aligned}$$

So

$$X_1 = S_2(S_1)^{-1}, \quad Y_1 = S_1(X_1)^{-1}.$$

Algorithm for Single Error Correction

Assume $g(x) = (x - \alpha)(x - \alpha^2)$.

- 1 Compute syndromes $S_1 = r(\alpha)$, $S_2 = r(\alpha^2)$.
- 2 Compute $X_1 = S_2/S_1$, $Y_1 = S_1/X_1$.
- 3 Error at location $j = \log_\alpha(X_1)$.
- 4 Error magnitude is Y_1 .
- 5 Subtract error polynomial from received polynomial

(4,2) Reed-Solomon code over GF(5)

- Consider the (4,2) Reed Solomon code over $\text{GF}(5)=\{0, 1, 2, 3, 4\}$.
- Addition and multiplication are done mod 5 so that $2 \times 3 = 6 = 1 \bmod 5$ and thus $2^{(-1)} = 3 \bmod 5$.
- Notice that over GF(5) we have the following factorization

$$x^4 - 1 = (x - 1)(x - 2)(x - 3)(x - 4).$$

- Consider the generator polynomial

$$g(x) = (x - 2)(x - 4) = x^2 - 6x + 8 = x^2 + 4x + 3.$$

- Since this is a factor of $x^4 - 1$ the code is a cyclic code.

Codewords of the (4,2) Reed-Solomon code over GF(5)

$(0 + 0x)g(x) = 0x^3 + 0x^2 + 0x + 0$	$(1 + 0x)g(x) = 0x^3 + 1x^2 + 4x + 3$
$(2 + 0x)g(x) = 0x^3 + 2x^2 + 3x + 1$	$(3 + 0x)g(x) = 0x^3 + 3x^2 + 2x + 4$
$(4 + 0x)g(x) = 0x^3 + 4x^2 + 1x + 2$	$(0 + 1x)g(x) = 1x^3 + 4x^2 + 3x + 0$
$(1 + 1x)g(x) = 1x^3 + 0x^2 + 2x + 3$	$(2 + 1x)g(x) = 1x^3 + 1x^2 + 1x + 1$
$(3 + 1x)g(x) = 1x^3 + 2x^2 + 0x + 4$	$(4 + 1x)g(x) = 1x^3 + 3x^2 + 4x + 2$
$(0 + 2x)g(x) = 2x^3 + 3x^2 + 1x + 0$	$(1 + 2x)g(x) = 2x^3 + 4x^2 + 0x + 3$
$(2 + 2x)g(x) = 2x^3 + 0x^2 + 4x + 1$	$(3 + 2x)g(x) = 2x^3 + 1x^2 + 3x + 4$
$(4 + 2x)g(x) = 2x^3 + 2x^2 + 2x + 2$	$(0 + 3x)g(x) = 3x^3 + 2x^2 + 4x + 0$
$(1 + 3x)g(x) = 3x^3 + 3x^2 + 3x + 3$	$(2 + 3x)g(x) = 3x^3 + 4x^2 + 2x + 1$
$(3 + 3x)g(x) = 3x^3 + 0x^2 + 1x + 4$	$(4 + 3x)g(x) = 3x^3 + 1x^2 + 0x + 2$
$(0 + 4x)g(x) = 4x^3 + 1x^2 + 2x + 0$	$(1 + 4x)g(x) = 4x^3 + 2x^2 + 1x + 3$
$(2 + 4x)g(x) = 4x^3 + 3x^2 + 0x + 1$	$(3 + 4x)g(x) = 4x^3 + 4x^2 + 4x + 4$
$(4 + 4x)g(x) = 4x^3 + 0x^2 + 3x + 2$	

(4,2) Reed-Solomon code over $GF(5)$

The codewords are easily seen to be cyclic shifts of the following vectors

0 0 0 0

0 1 4 3

0 2 3 1

0 3 2 4

0 4 1 2

1 3 4 2

1 1 1 1

2 2 2 2

3 3 3 3

4 4 4 4

(4,2) Reed-Solomon code over GF(5)

Note that because this is a linear code the minimum distance is the minimum weight which is 3. Thus this code can correct one error. Suppose the code is used on a channel and suppose one error is made. Then

$$r(x) = c(x) + e(x)$$

where

$$e(x) = e_j x^j$$

This implies the error has magnitude e_j and occurred in the j -th position.

(4,2) Reed-Solomon code over GF(5)

Because $c(x) = g(x)i(x)$ for some polynomial $i(x)$ it is clear that

$$\begin{aligned} S_1 = r(2) &= g(2)i(2) + e(2) = e(2) = e_j 2^j \\ S_2 = r(4) &= g(4)i(4) + e(4) = e(4) = e_j 2^{2j} \end{aligned}$$

Thus the location of the error can be determined from

$$j = \log_2\left(\frac{r(4)}{r(2)}\right) = \log_2(2^j).$$

The magnitude of the error can be determined by

$$e_j = S_1/X_1 = S_1^2/S_2.$$

Example

Suppose we receive

$$r(x) = 2x^3 + 4x^2 + 2x^1 + 3$$

Then $r(2) = 4$ and $r(4) = 3$. The error locator is $X_1 = 2^j = \frac{r(4)}{r(2)} = 2^1$ thus $j = 1$. The error magnitude is $r(2)/X_1 = 4/2 = 4 * 3 = 2$. Thus the error polynomial is $e(x) = 2x^1$. Thus

$$\begin{aligned} c(x) = r(x) - e(x) &= 2x^3 + 4x^2 + 2x^1 + 3 - 2x^1 \\ &= 2x^3 + 4x^2 + 0x^1 + 3 \end{aligned}$$

which is indeed a codeword.

(10,6) Reed-Solomon code over GF(11)

The (10,6) code over GF(11) has distance 5 which means it can correct two errors. Let's see how this is done.

First note that 2 is a primitive element in this field. That is, the powers of 2 generate all the nonzero elements of the field.

i	2^i	i	2^i
0	1	5	10
1	2	6	9
2	4	7	7
3	8	8	3
4	5	9	6

Second note that

$$x^{10} - 1 = (x - 1)(x - 2)(x - 3)(x - 4)(x - 5)(x - 6)(x - 7)(x - 8)(x - 9)(x - 10)$$

so that any combination of these factors will yield a cyclic code.

(10,6) Reed-Solomon code over GF(11)

Consider the generator polynomial for this code is

$$g(x) = (x - 2)(x - 2^2)(x - 2^3)(x - 2^4).$$

A codeword $c(x)$ is a multiple of the generator polynomial $c(x) = g(x) * m(x)$. A received polynomial is $r(x) = c(x) + e(x)$. We evaluate the received polynomial at the zeros of the generator polynomial. Suppose only two errors occurred at locations i and l . Let Y_1 and Y_2 be the errors that were made. Let $X_1 = 2^i$ and $X_2 = 2^l$ be indicators of the locations of the errors.

$$\begin{aligned} S_1 &= r(2^1) = Y_1 2^i + Y_2 2^l = Y_1 X_1 + Y_2 X_2 \\ S_2 &= r(2^2) = Y_1 2^{2i} + Y_2 2^{2l} = Y_1 X_1^2 + Y_2 X_2^2 \\ S_3 &= r(2^3) = Y_1 2^{3i} + Y_2 2^{3l} = Y_1 X_1^3 + Y_2 X_2^3 \\ S_4 &= r(2^4) = Y_1 2^{4i} + Y_2 2^{4l} = Y_1 X_1^4 + Y_2 X_2^4 \end{aligned}$$

(10,6) Reed-Solomon code over GF(11)

The goal is to solve these four equations in four unknowns to determine the error magnitudes and the error locations. Note that if we determine X_1 and X_2 then we can easily solve for Y_1 and Y_2 because the equations become linear. As such,

$$\begin{aligned} \begin{bmatrix} X_1 & X_2 \\ X_1^2 & X_2^2 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} &= \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \\ \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} &= \begin{bmatrix} X_1 & X_2 \\ X_1^2 & X_2^2 \end{bmatrix}^{-1} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \\ \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} &= (X_1 X_2^2 - X_1^2 X_2)^{-1} \begin{bmatrix} X_2^2 & -X_2 \\ -X_1^2 & X_1 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}. \end{aligned}$$

(10,6) Reed-Solomon code over GF(11)

Consider the polynomial $\Lambda(x)$ given by

$$\Lambda(x) = (1 - xX_1)(1 - xX_2) = 1 + \Lambda_1x + \Lambda_2x^2$$

Clearly $\Lambda(X_1^{-1}) = \Lambda(X_2^{-1}) = 0$. So

$$\begin{aligned} 1 + \Lambda_1 X_1^{-1} + \Lambda_2 X_1^{-2} &= 0 \\ Y_1 X_1^{j+2} (1 + \Lambda_1 X_1^{-1} + \Lambda_2 X_1^{-2}) &= 0 \\ Y_1 X_1^{j+2} + \Lambda_1 Y_1 X_1^{j+1} + \Lambda_2 Y_1 X_1^j &= 0. \end{aligned}$$

(10,6) Reed-Solomon code over GF(11)

Similarly

$$\begin{aligned}
 1 + \Lambda_1 X_2^{-1} + \Lambda_2 X_2^{-2} &= 0 \\
 Y_2 X_2^{j+2} (1 + \Lambda_1 X_2^{-1} + \Lambda_2 X_2^{-2}) &= 0 \\
 Y_2 X_2^{j+2} + \Lambda_1 Y_1 X_2^{j+1} + \Lambda_2 Y_2 X_2^j &= 0.
 \end{aligned}$$

So

$$\begin{aligned}
 Y_1 X_1^{j+2} + \Lambda_1 Y_1 X_1^{j+1} + \Lambda_2 Y_1 X_1^j &= 0 \\
 Y_2 X_2^{j+2} + \Lambda_1 Y_1 X_2^{j+1} + \Lambda_2 Y_2 X_2^j &= 0.
 \end{aligned}$$

(10,6) Reed-Solomon code over GF(11)

This holds for $j = 1, 2$. So for $j = 1$ we have

$$\begin{aligned} Y_1 X_1^3 + \Lambda_1 Y_1 X_1^2 + \Lambda_2 Y_1 X_1^1 &= 0 \\ Y_2 X_2^3 + \Lambda_1 Y_2 X_2^2 + \Lambda_2 Y_2 X_2^1 &= 0. \\ S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 &= 0 \end{aligned}$$

(10,6) Reed-Solomon code over GF(11)

The last equation is the sum of the first two since $S_3 = Y_1X_1^3 + Y_2X_2^3$.
For $j = 2$ we have

$$Y_1X_1^4 + \Lambda_1 Y_1X_1^3 + \Lambda_2 Y_1X_1^2 = 0$$

$$Y_2X_2^4 + \Lambda_1 Y_1X_2^3 + \Lambda_2 Y_2X_2^2 = 0.$$

$$S_4 + \Lambda_1 S_3 + \Lambda_2 S_2 = 0$$

$$S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 = 0$$

$$S_4 + \Lambda_1 S_3 + \Lambda_2 S_2 = 0.$$

(10,6) Reed-Solomon code over GF(11)

Putting these equations into matrix form yields

$$\begin{aligned}
 \begin{bmatrix} S_2 & S_1 \\ S_3 & S_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} &= \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} \\
 \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} &= \begin{bmatrix} S_2 & S_1 \\ S_3 & S_2 \end{bmatrix}^{-1} \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} \\
 &= (S_2^2 - S_1 S_3)^{-1} \begin{bmatrix} S_2 & -S_1 \\ -S_3 & S_2 \end{bmatrix} \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} \\
 &= (S_2^2 - S_1 S_3)^{-1} \begin{bmatrix} S_1 S_4 - S_2 S_3 \\ S_3^2 - S_2 S_4 \end{bmatrix}
 \end{aligned}$$

RS(10,6) over GF(11)

Consider the field of 11 elements $\{0, 1, 2, \dots, 10\}$ with arithmetic done mod 11. Note that the powers of 2 generate all the elements in the field (except 0). Let

$$g(x) = (x - 2)(x - 2^2)(x - 2^3)(x - 2^4) = (x - 2)(x - 4)(x - 8)(x - 5).$$

This generates a Reed-Solomon code over GF(11) with $d_{\min} = 5$ and thus can correct 2 errors. Suppose

$$r(x) = 1 + 9x + 3x^2 + 2x^3 + 7x^4 + 7x^5 + 3x^6 + 9x^7 + 4x^8 + 1x^9$$

$$S_1 = r(2) = 7$$

$$S_2 = r(2^2) = 7$$

$$S_3 = r(2^3) = 0$$

$$S_4 = r(2^4) = 2$$

RS(10,6) over GF(11)

$$\begin{aligned} \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} &= \begin{bmatrix} 7 & 7 \\ 0 & 7 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -2 \end{bmatrix} \\ &= \begin{bmatrix} 8 & 3 \\ 0 & 8 \end{bmatrix} \begin{bmatrix} 0 \\ 9 \end{bmatrix} \\ &= \begin{bmatrix} 5 \\ 6 \end{bmatrix}. \end{aligned}$$

RS(10,6) over GF(11)

Thus $\Lambda(x) = 1 + 5x + 6x^2$. The solutions of $\Lambda(x) = 0$ is $x = 5$ and $x = 7$. Since $\Lambda(X_1^{-1}) = \Lambda(X_2^{-1}) = 0$ we conclude that $X_1^{-1} = 5$ and $X_2^{-1} = 7$. Thus $\Lambda(x) = (1 - 5x)(1 - 7x)$

$$X_1 = (5)^{-1} = 9 = 2^6$$

$$X_2 = (7)^{-1} = 8 = 2^3.$$

So the errors occurred in the third and sixth position of the codeword (i.e. the coefficient of x^3 and x^6).

RS(10,6) over GF(11)

Now we can find the error magnitudes.

$$S_1 = Y_1 X_1 + Y_2 X_2$$

$$S_2 = Y_1 X_1^2 + Y_2 X_2^2$$

$$7 = 9Y_1 + 8Y_2$$

$$7 = 4Y_1 + 9Y_2$$

$$\begin{aligned} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} &= \begin{bmatrix} 9 & 8 \\ 4 & 9 \end{bmatrix}^{-1} \begin{bmatrix} 7 \\ 7 \end{bmatrix} \\ &= \begin{bmatrix} 4 & 5 \\ 8 & 4 \end{bmatrix} \begin{bmatrix} 7 \\ 7 \end{bmatrix} \\ &= \begin{bmatrix} 8 \\ 7 \end{bmatrix} \end{aligned}$$

RS(10,6) over GF(11)

Thus $e(x) = 7x^3 + 8x^6$ and

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

$$r(x) = 1 + 9x + 3x^2 + 2x^3 + 7x^4 + 7x^5 + 3x^6 + 9x^7 + 4x^8 + 1x^9$$

$$e(x) = 7x^3 + 8x^6$$

$$c(x) = 1 + 9x + 3x^2 + 6x^3 + 7x^4 + 7x^5 + 6x^6 + 9x^7 + 4x^8 + 1x^9$$

$$c(x) = g(x)(1 + x + x^2 + x^3 + x^4 + x^5)$$

Algorithm for Correcting Two Errors

This assumes that $g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)$.

- 1 Evaluate the syndromes S_1, S_2, S_3, S_4 .

$$S_1 = r(\alpha), S_2 = r(\alpha^2), S_3 = r(\alpha^3), S_4 = r(\alpha^4)$$

- 2 From syndromes find Λ_1, Λ_2 .

$$\begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} = (S_2^2 - S_1 S_3)^{-1} \begin{bmatrix} S_1 S_4 - S_2 S_3 \\ S_3^2 - S_2 S_4 \end{bmatrix}$$

- 3 From Λ_1, Λ_2 , find X_1, X_2 , the error locations

$$\Lambda(x) = (1 - xX_1)(1 - xX_2) = 1 + \Lambda_1 x + \Lambda_2 x^2$$

- 4 From S_1, S_2, X_1, X_2 find Y_1, Y_2 .

$$\begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_1^2 & X_2^2 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \Rightarrow \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_1^2 & X_2^2 \end{bmatrix}^{-1} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

GF(16)

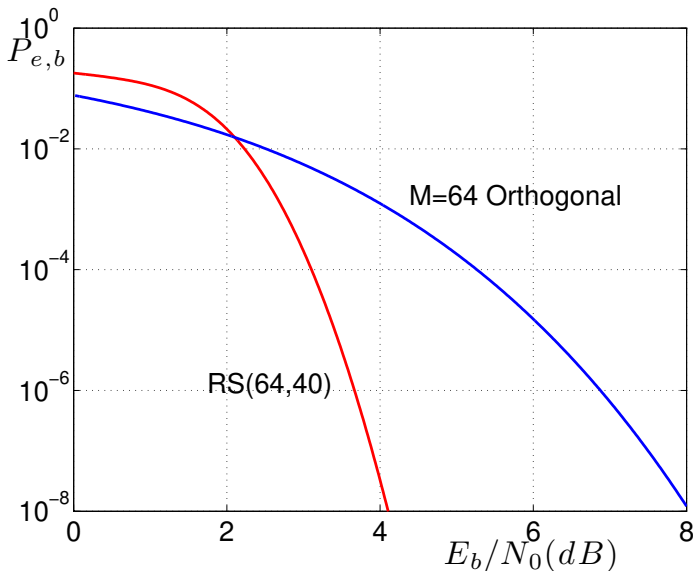
Here is $GF(2^4)$ using primitive polynomial $x^4 + x + 1$.

<i>Decimal</i>	<i>Binary</i>			<i>Decimal</i>	<i>Binary</i>		
0	0000	$\alpha^{-\infty}$	0	11	1011	α^7	$\alpha^3 + \alpha + 1$
1	0001	α^0	1	5	0101	α^8	$\alpha^2 + 1$
2	0010	α^1	α	10	1010	α^9	$\alpha^3 + \alpha$
4	0100	α^2	α^2	7	0111	α^{10}	$\alpha^2 + \alpha + 1$
8	1000	α^3	α^3	14	1110	α^{11}	$\alpha^3 + \alpha^2 + \alpha$
3	0011	α^4	$\alpha + 1$	15	1111	α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$
6	0110	α^5	$\alpha^2 + \alpha$	13	1101	α^{13}	$\alpha^3 + \alpha^2 + 1$
12	1100	α^6	$\alpha^3 + \alpha^2$	9	1001	α^{14}	$\alpha^3 + 1$

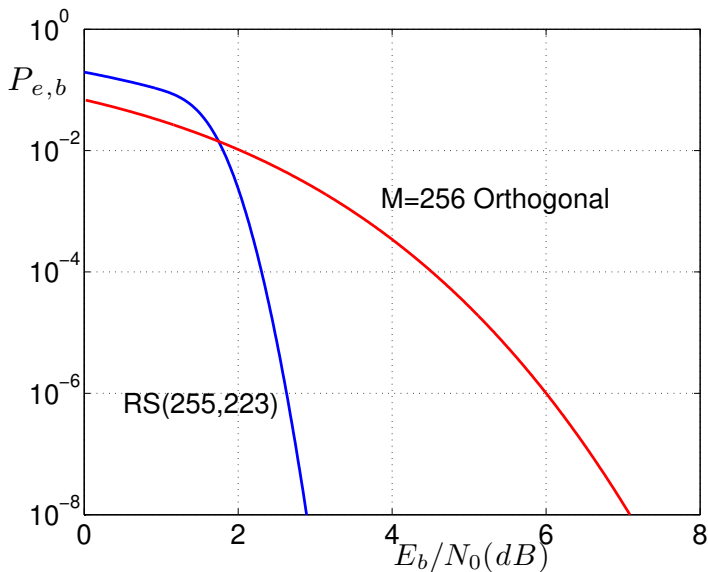
Reed-Solomon Codes and Orthogonal Modulation

- A natural combination of Reed-Solomon codes is with orthogonal modulation.
- Reed-Solomon codes typically operate with symbols of size M where M is a power of 2.
 - So, for example, a Reed-Solomon code of length 64 uses symbols from an alphabet of size 64. Each symbol can be mapped into one of 64 orthogonal waveforms.
 - Similarly the NASA standard code of length 255 uses symbols from an alphabet of size 256 (eight bit bytes). We can map each symbol into one of 256 different orthogonal signals.
- Below we show two curves.
 - The first is for a Reed-Solomon code of length 64 with 40 information symbols using 64-ary orthogonal modulation.
 - The second is a Reed-Solomon code of length 255 with 223 information symbols using 256-ary orthogonal modulation.

(64,40) Reed-Solomon code over GF(64)



(255,223) Reed-Solomon code over GF(256)



Bounded Distance Decoding

- A code with minimum distance d_{\min} can correct any error pattern of weight $\lfloor \frac{d_{\min}-1}{2} \rfloor$ or less.
- If more than $\lfloor \frac{d_{\min}-1}{2} \rfloor$ errors occur a decoder may or may not be able (in theory) to correct it.
- A bounded distance decoder is a decoder that corrects any error pattern of weight ℓ or less ($\ell \leq \lfloor \frac{d_{\min}-1}{2} \rfloor$) and either fails or errors otherwise.
- The decoder fails if no codeword is within distance ℓ or less of the received vector.
- The decoder makes an error if one codeword is distance ℓ or less from the received vector.
- Most cyclic codes can be decoded efficiently only by a bounded distance decoder.

Bounded Distance Decoding

- Note that a bounded distance decoder is not a MAP or maximum likelihood decoder since some received vectors are not decoded into the closest codeword. In fact, there are received vectors that are not mapped (decoded) into any codeword. These cause the decoder to fail.
- The decoding complexity of a bounded distance decoder of a cyclic code is roughly $(n - k)^2$, which is much less than the complexity of syndrome decoding or brute force decoding for reasonable values of n and k .

Complexity

Structure	Decoding	Complexity
Arbitrary	Brute Force	$M = 2^k$
Linear	Syndrome/Brute Force	$\min(2^k, 2^{n-k})$
Cyclic	Bounded Distance	$(n - k)^2$

Bounded Distance Decoding

- For a bounded distance decoder with error correction capability ℓ there are three possible decoder outcomes.
 - Correct Decoding
 - Decoding to a wrong codeword (decoder error)
 - Failure to decode (not within the correcting capability of the decoder).
- The probability of correct decoding is

$$\begin{aligned} P_{CD} &= P\{\ell \text{ or fewer errors}\} \\ &= \sum_{i=0}^{\ell} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned}$$

Bounded Distance Decoding

- The probability of incorrect decoding (decoding error) can be upper bounded by assuming any error pattern of weight $d - \ell$ or larger is within distance ℓ of some codeword

$$P_{ICD} \leq \sum_{i=d-\ell}^n \binom{n}{i} p^i (1-p)^{n-i}$$

- The probability of decoder failure or error detection (no codeword within distance ℓ of received vector) can be lower bounded as follows.

$$\begin{aligned} P_{CD} &= 1 - P_{ED} - P_{ICD} \\ P_{ED} &= 1 - P_{ICD} - P_{CD} \\ &\geq 1 - \sum_{i=d-\ell}^n \binom{n}{i} p^i (1-p)^{n-i} - \sum_{i=0}^{\ell} \binom{n}{i} p^i (1-p)^{n-i} \\ &= \sum_{i=\ell+1}^{d-\ell-1} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned}$$

Bounded Distance Decoding

- The bit error probability can be upper bounded by assuming that all incorrect decoding events caused by i errors cause the decoder to insert at most $i + \ell$ errors.

$$P_b \leq \sum_{i=d-\ell}^n \frac{i+\ell}{n} \binom{n}{i} p^i (1-p)^{n-i}.$$

Note: As the rate of the code decreases the error probability decreases.

Fact: For binary symmetric channel for any $\epsilon > 0$ there exist codes with error probability $< \epsilon$ provided the code rate is less than the capacity,

$$r < C = 1 - H_2(p)$$

$$H_2(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

Bounded Distance Decoding

- Up to now we have only considered binary codes transmitted over a binary symmetric channel.
- However there are codes with symbol alphabet size larger than two.
- The most important such code is the Reed-Solomon code.
- Let N and K be the length and dimension (number of information symbols of a Reed-Solomon code).
- The alphabet size is 2^m where m is an integer.
- The choice of m determines the possible lengths of the RS code.

Bounded Distance Decoding

- For example if $m = 5$ then N can have length $2^5 + 1$ or less.
- Typically it is either 31 or 32.
- The code can correct a number of symbol errors determined by the minimum distance.
- An (N, K) Reed-Solomon code has minimum Hamming distance $d_{\min} = N - K + 1$ and can correct $\lfloor (d_{\min} - 1)/2 \rfloor$ symbol errors.
- The Reed- Solomon code is also a cyclic code and can be (bounded distance) decoded with an efficient algorithm (for a hard decision channel).
- The symbol error probability is closely approximated by the error probability of a bounded distance decoder.

Applications of Reed-Solomon Codes

This code has many applications. One example is the Compact Disc system. In this case actually two slightly different Reed-Solomon codes are used. The compact disc stores binary data. Eight bits of data are group together to form a symbol. The alphabet size is thus 256. If one of the bits in a symbol is in error then the symbol is in error. However, more than one bit error within a symbol still just cause a single symbol to be in error. In this sense, RS codes are burst error correcting codes. The CD system uses a (32,28) code with symbols of size 2^8 and a (28,24) code with the same size symbols. NASA has a standard for a (255,223) Reed-Solomon code that can correct up to 16 symbol errors (symbol size =256).

Geometry of Codes, Revisited

- Consider the (7,4) binary Hamming code
 - The Hamming code has 16 codewords out of a total of 128 possible vectors (12.5 percent of the vectors are codewords).
 - The (7,4) Hamming code can correct one error ($d_{\min} = 3$) and each decoding region contains 8 vectors.
 - The decoding regions surrounding the 16 codewords account for $8 \times 16 = 128$ vectors; 100 percent of the vectors in the space.

Geometry of Codes, Revisited

- Consider the (7,4) Reed-Solomon code with symbols size 8.
 - The Reed-Solomon code has $8^4 = 4096$ codewords and the total number of vectors is $8^7 = 2097152$ so 0.195 percent of vectors are codewords.
 - The (7,4) Reed-Solomon code has a minimum distance 4 and also can correct one symbol error.
 - The decoding region of a bounded distance decoder contains $1 + 7 \times 7 = 50$ vectors.
 - The total number of vectors in all decoding regions is thus $50 \times 4096 = 204800$ of the total of 2097152 vectors or 9.76 percent of the total number of vectors.
 - Thus given that the number of errors is greater than the correcting capability for a Reed-Solomon code, the received vector is likely not to be in the decoding region of any codeword.

Soft and Hard Decision Decoding of Block Codes for an Additive White Gaussian Noise Channel

So far we have only consider decoding algorithms for block codes when the channel is the binary symmetric channel. These algorithms can be used when the channel is the additive Gaussian channel. Basically, the idea is to convert the additive Gaussian channel into a binary symmetric channel. To do we restrict the input to the channel to be from the set $\{+\sqrt{E}, -\sqrt{E}\}$ and use the following mapping from the set $\{0, 1\}$ into the set $\{+\sqrt{E}, -\sqrt{E}\}$

$$0 \rightarrow \sqrt{E}$$

and

$$1 \rightarrow -\sqrt{E}.$$

Soft and Hard Decision Decoding of Block Codes for an Additive White Gaussian Noise Channel

If the allowable inputs to the additive Gaussian channel are $\{+\sqrt{E}, -\sqrt{E}\}$ then we can use a binary block code to transmit information over a additive Gaussian channel. Thus each codeword is transmitted as a sequence of $\pm\sqrt{E}$'s. The output of the channel is the input plus a Gaussian noise variable.

$$r_i = c_i + n_i$$

where c_i is the i th component of the codeword (after the mapping) and n_i is Gaussian noise.

The optimal decision rule is to decide codeword c_j was transmitted if it is closest in Euclidean distance to what was received. This can be very difficult for block codes. We can use decoding algorithms for a binary symmetric channel by making a hard decision of each of the symbols. That is, we decide that the i -th bit was $+\sqrt{E}$ if $r_i > 0$ and the i -th bit was $-\sqrt{E}$ if $r_i < 0$.

Soft and Hard Decision Decoding of Block Codes for an Additive White Gaussian Noise Channel

We can then do the inverse map

$$1 \rightarrow \sqrt{E}$$

and

$$-1 \rightarrow -\sqrt{E}.$$

The probability that a channel output is received as $+1$ given the input was $-\sqrt{E}$ is the same as the probability that a channel output is received as $-\sqrt{E}$ given the input was $+\sqrt{E}$ (due to the symmetry of the noise distribution). We can then do the inverse map

$$\sqrt{E} \rightarrow 0 \quad \text{and} \quad -\sqrt{E} \rightarrow 1.$$

and get a binary symmetric channel for which known algorithms can be used for decoding. This is called hard decision decoding since on every symbol of the received vector we make a decision as to what the transmitted symbol was.

Soft Decision Decoding of Block Codes for an Additive White Gaussian Noise Channel

The optimal decoding algorithm for the additive Gaussian channel (also called soft decision decoding) compares the received vector with every possible transmitted codeword and chooses the codeword that is closest (in Euclidean distance) to what was received. For example, if the repetition code of length 3 is used and the vector

$$(1.5, -.3, -.4)$$

is received the optimal (soft decision) algorithm would choose the codeword $(+\sqrt{E}, +\sqrt{E}, +\sqrt{E})$ as the transmitted codeword. (Note that it is easy to show that the closest codeword to the received vector is also the codeword with largest correlation with the received vector).

Hard Decision Decoding for an Additive White Gaussian Noise Channel

A hard decision algorithm would first make a decision on each symbol and produce the vector $(+\sqrt{E}, -\sqrt{E}, -\sqrt{E})$ then find the codeword closest in Hamming distance $(-\sqrt{E}, -\sqrt{E}, -\sqrt{E})$. For this received vector the two decoding algorithms differ in their output.

Soft and Hard Decision Decoding of Block Codes for an Additive White Gaussian Noise Channel

The binary symmetric channel created by the hard decision device has crossover probability that depends on the energy E and the noise variance $N_0/2$ as

$$p = Q\left(\sqrt{\frac{2E}{N_0}}\right).$$

If E_b is the energy transmitted per information bit then the k information bits of a codeword use nE joules to transmit them. Thus the energy per information bit is related to the energy per channel bit by

$$E_b = nE/k = E/r$$

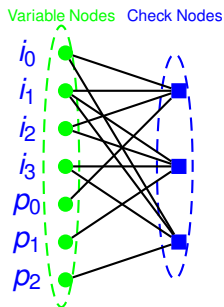
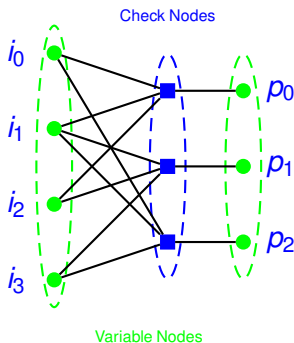
where $r = k/n$ is the rate of the code.

Low Density Parity Check (LDPC) Codes

- Low density parity check codes are a particular type of linear code in which the parity check matrix contains a relatively small number of ones (e.g less than 10%).
- Encoding of these codes can be done efficiently using several different approaches (see Ping, Richardson).
- Decoding of these codes on a binary symmetric channel or an additive white Gaussian noise channel is generally not practical but suboptimum decoding techniques achieve very good performance.
- The decoding is best understood by use of a Tanner graph. A Tanner graph depicts the relation between information bits, parity bits and parity checks. Information bits and parity bits are depicted as variable nodes and then parity checks are depicted as check nodes.

Tanner Graphs for (7,4) Hamming Code

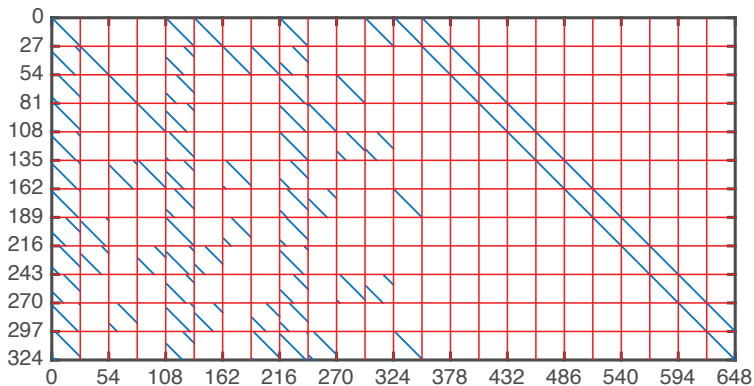
$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$



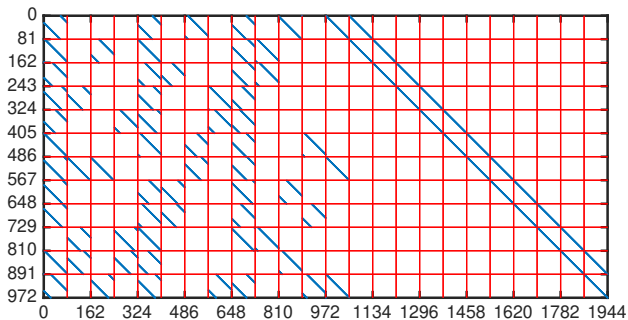
- In IEEE 802.11 (WiFi) codes are used that have a parity check matrix that is low density.
- The figure below shows the parity check matrix is shown for a (648,324) low density parity check (LDPC) code.
- The diagonal lines are locations in the parity check matrix with a 1 entry while all the other locations are 0.
- This parity check matrix is organized into submatrices of size 27x27 (as shown by the red lines).

- Each submatrix is either all zeros, an identity (e.g. the top left submatrix) or an identity matrix cyclicly shifted by a certain number of places.
- As can be seen the parity check matrix contains a small fraction of ones. So the matrix is called a low density parity check matrix and the codes are called low density parity check codes.
- Similarly shown below is the parity check matrix is shown for a (1944,972) LDPC code used in WiFi. In this case the submatrices have size 81 by 81.

Parity check matrix for (648,324) rate 1/2 code

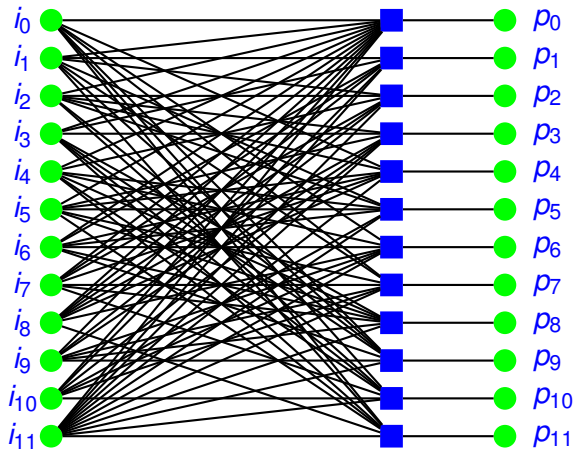


Parity check matrix for (1944,972) rate 1/2 code

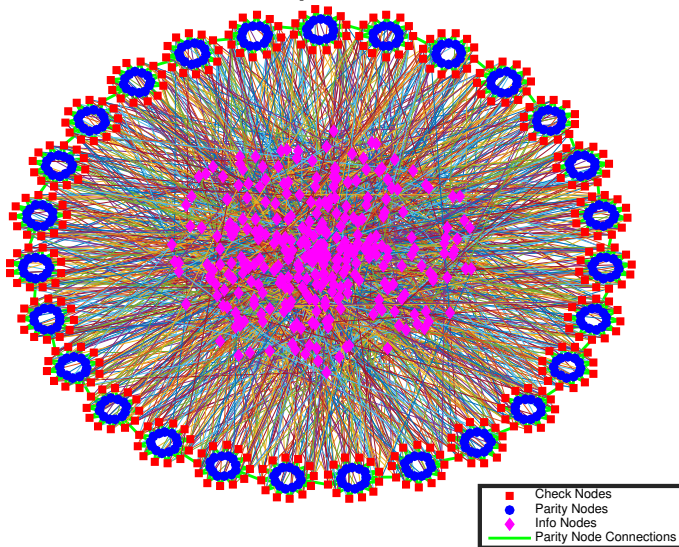


- LDPC Codes can be represented with Tanner graphs.
- A Tanner graphs for an LDPC code consists of *variable nodes* corresponding to each bit in a codeword and *check nodes* corresponding to each parity check equation in the code.
- A variable node is connected to a check node if the corresponding coded bit participates in the parity check of the check node.
- A Tanner graph can be constructed for any linear code.
- For example the Golay code has the Tanner graph shown below.

Tanner graph for extended Golay code



Tanner Graph for Code 1a



Decoding Algorithms

- The (648,324) code used in WiFi standard has $k = 324$ and $n = 648$.
- The complexity of (i) the brute force decoding of finding the codeword closest to the received vector is not practical since there are 2^{324} codewords and (ii) the decoding approach of building a table of syndromes and coset leaders for which there are $2^{648-324} = 2^{324}$ entries is not practical.
- In addition, neither of these techniques will work on an additive white Gaussian noise channel.
- For these reasons the decoding approach is a suboptimal approach that, nevertheless, has excellent performance.
- The decoding algorithm is known as a message passing algorithm.

Information u_0, u_1, \dots, u_{k-1} is encoded into a codeword c_0, c_1, \dots, c_{n-1} . The codeword (consisting of 0's and 1's) is mapped into modulation symbols (+1, -1) using

$$b_i = (-1)^{c_i}$$

so that 0 is mapped to +1 and 1 is mapped to -1. This makes mod two addition in the 0,1 domain isomorphic to multiplication in the +1,-1 domain. The modulation symbols are multiplied by \sqrt{E} and then noise is added with zero mean and variance $\sigma^2 = N_0/2$.

$$r_i = \sqrt{E}b_i + \eta_i.$$

Based on observing r_i alone we can determine the log-likelihood ratio for b_i . Now consider the log-likelihood of b_i given we observe r_i .

Assume that apriori $p(b_i = +1) = p(b_i = -1) = 1/2$. Then

$$\begin{aligned}
 L_e(b_i) &= \log \left[\frac{p(b_i = +1|r_i)}{p(b_i = -1|r_i)} \right] \\
 &= \log \left[\frac{p(r_i|b_i = +1)p(b_i = +1)/p(r_i)}{p(r_i|b_i = -1)p(b_i = -1)/p(r_i)} \right] \\
 &= \log \left[\frac{p(r_i|b_i = +1)}{p(r_i|b_i = -1)} \right] \\
 &= \log \left[\frac{(2\pi\sigma^2)^{-1/2} \exp(-(r_i - \sqrt{E})^2/(2\sigma^2))}{(2\pi\sigma^2)^{-1/2} \exp(-(r_i + \sqrt{E})^2/(2\sigma^2))} \right] \\
 &= \log \left(\frac{\exp(-(r_i^2 - 2r_i\sqrt{E} + E)/(2\sigma^2))}{\exp(-(r_i^2 + 2r_i\sqrt{E} + E)/(2\sigma^2))} \right) \\
 &= \log(\exp(4r_i\sqrt{E}/(2\sigma^2))) \\
 &= 4r_i\sqrt{E}/N_0.
 \end{aligned}$$

Note that $L_e(b_i)$ represents the likelihood for b_i based solely on the received signal r_i . First consider any likelihood of a binary (+1,-1) variable u . We know that $P(u = +1) + P(u = -1) = 1$. Then

$$\begin{aligned}
 L(u) &= \log\left(\frac{P(u = +1)}{P(u = -1)}\right) \\
 &= \log\left(\frac{P(u = +1)}{1 - P(u = +1)}\right) \\
 e^{L(u)} &= \frac{P(u = +1)}{1 - P(u = +1)} \\
 e^{L(u)}(1 - P(u = +1)) &= P(u = +1) \\
 e^{L(u)} &= P(u = +1)(1 + e^{L(u)}) \\
 P(u = +1) &= \frac{e^{L(u)}}{(1 + e^{L(u)})}.
 \end{aligned}$$

Also

$$\begin{aligned}
 P(u = -1) &= 1 - P(u = +1) \\
 &= 1 - \frac{e^{L(u)}}{(1 + e^{L(u)})} \\
 &= \frac{1}{(1 + e^{L(u)})}.
 \end{aligned}$$

Note that

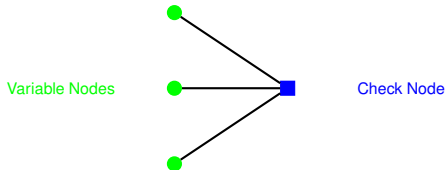
$$\begin{aligned}
 P(u = +1) - P(u = -1) &= \frac{e^{L(u)} - 1}{e^{L(u)} + 1} \\
 &= \frac{e^{L(u)/2} - e^{-L(u)/2}}{e^{L(u)/2} + e^{-L(u)/2}} \\
 &= \tanh(L(u)/2).
 \end{aligned}$$

The probabilities of u can be determined in one of three ways: From i) $P(u = +1)$, ii) $P(u = +1) - P(u = -1)$ or iii) $L(u)$.

Now consider a check node. The variable nodes connected to a check node must sum to 0 (in the 0,1 domain). Suppose three variable nodes (c_i, c_j, c_m or b_i, b_j, b_m) were connected to a check node. Then

$$c_i + c_j + c_m = 0$$

$$b_i \cdot b_j \cdot b_m = 1.$$



Suppose we have some log-likelihoods for b_j and b_m and knowing that the check equations above must be satisfied what is the log-likelihood ratio for b_m . Note that $b_m = b_i \cdot b_j$. That is, if $b_i = b_j = +1$ or $b_i = b_j = -1$ then $b_m = +1$. Otherwise $b_m = -1$. This means that

$$\begin{aligned} P(b_m = +1) &= P(b_i = +1, b_j = +1) + P(b_i = -1, b_j = -1) \\ &= P(b_i = +1)P(b_j = +1) + P(b_i = -1)P(b_j = -1) \end{aligned}$$

where we have assumed that the modulation symbols are independent. While this is not actually true, the algorithm we are describing is not actually an optimal algorithm. Similarly

$$\begin{aligned} P(b_m = -1) &= P(b_i = +1, b_j = -1) + P(b_i = -1, b_j = +1) \\ &= P(b_i = +1)P(b_j = -1) + P(b_i = -1)P(b_j = +1). \end{aligned}$$

For simplicity of notation let $P_m = P(b_m = +1)$ and $Q_m = P(b_m = -1)$ and similarly for P_i, P_j, Q_i, Q_j . We know that

$$\begin{aligned}P_m - Q_m &= \tanh(L(b_m)/2), \\P_i - Q_i &= \tanh(L(b_i)/2), \\P_k - Q_k &= \tanh(L(b_k)/2).\end{aligned}$$

Then

$$\begin{aligned}P_m &= P_i P_j + Q_i Q_j \\Q_m &= P_i Q_j + Q_i P_j \\P_m - Q_m &= P_i P_j + Q_i Q_j - P_i Q_j - Q_i P_j \\P_m - Q_m &= (P_i - Q_i)(P_j - Q_j) \\\tanh(L(b_m)/2) &= \tanh(L(b_i)/2) \tanh(L(b_j)/2) \\L(b_m) &= 2 \tanh^{-1}(\tanh(L(b_i)/2) \tanh(L(b_j)/2)).\end{aligned}$$

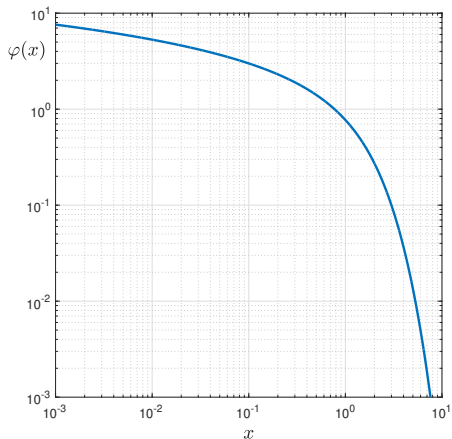
- Now consider again the likelihood for b_m .
- We are going to do the calculation in two steps.
 - Step 1 is determining the sign
 - Step 2 is determining the magnitude.
- Consider first the sign for $L(b_m)$. We have that $\tanh(L(b_m)/2) = \tanh(L(b_i)/2) \tanh(L(b_j)/2)$. Note that the \tanh function is odd symmetric. That is, $\tanh(-x) = -\tanh(x)$. Let $\text{sgn}(x)$ be +1 if $x \geq 0$ and -1 if $x < 0$. Then

$$\text{sgn}(L(b_m)) = \text{sgn}(L(b_i)) \cdot \text{sgn}(L(b_j)).$$

Now consider the magnitude. Consider the function

$$\varphi(x) = -\log(\tanh(x/2)) = -\log\left(\frac{e^{x/2} - e^{-x/2}}{e^{x/2} + e^{-x/2}}\right), \quad x > 0.$$

This function is shown in figure on the next slide. Note that since $0 < \tanh(x) < 1$ for $x > 0$ then $\varphi(x) > 0$ for $x > 0$.

φ function.

As can be seen from the plot if the axes are swapped the function is the same. That is, for $y > 0$, $\varphi^{-1}(y) = \varphi(y) = -\log(\tanh(y/2))$. This is shown as follows.

$$\begin{aligned} y &= -\log(\tanh(x/2)) \\ e^{-y} &= \frac{e^{x/2} - e^{-x/2}}{e^{x/2} + e^{-x/2}} \end{aligned}$$

Now solve for x in terms of y . Let $u = e^{x/2}$. Then

$$e^{-y} = \frac{u - 1/u}{u + 1/u} = \frac{u^2 - 1}{u^2 + 1}$$

$$e^{-y}(u^2 + 1) = u^2 - 1$$

$$e^{-y}u^2 + e^{-y} = u^2 - 1$$

$$e^{-y} + 1 = u^2(1 - e^{-y})$$

$$u^2 = \frac{1 + e^{-y}}{1 - e^{-y}}$$

$$e^x = \frac{1 + e^{-y}}{1 - e^{-y}}$$

$$e^{-x} = \frac{1 - e^{-y}}{1 + e^{-y}} = \frac{e^{y/2} - e^{-y/2}}{e^{y/2} + e^{-y/2}} = \tanh(y/2)$$

$$-x = \log(\tanh(y/2))$$

$$x = -\log(\tanh(y/2)) = \varphi(y).$$

So

$$\begin{aligned} |L(b_m)| &= \varphi^{-1}(\varphi(|L(b_i)|) + \varphi(|L(b_j)|)) \\ &= \varphi(\varphi(|L(b_i)|) + \varphi(|L(b_j)|)) \\ \text{sgn}(L(b_m)) &= \text{sgn}(L(b_i)) \cdot \text{sgn}(L(b_j)) \\ L(b_m) &= \text{sgn}(L(b_m))|L(b_m)|. \end{aligned}$$

- In an LDPC decoder a check node receives likelihoods from variable nodes and then sends back to the variable nodes updated likelihoods of the variable.
- In the example where $b_i \cdot b_j \cdot b_m = 1$ the check node will receive three likelihoods.
- It will use the likelihoods for b_i and b_j to determine a new likelihood for b_m .
- Similarly it will use the likelihoods for b_j and b_m to determine a new likelihood for b_i .
- Also, it will determine a likelihood for b_j based on b_i and b_m .
- This can be extended to more than three variable nodes as input to a check nodes.
- Suppose that the messages received by check node c_l from variable node v_m is indicating the likelihood of the variable node v_m being +1 or -1 is $m(v_m \rightarrow c_l)$.

Then the message sent from check node l to variable node m is

$$m(c_l \rightarrow v_m) = 2 \tanh^{-1}(\prod_{j \in N(l,m)} \tanh(m(v_j \rightarrow c_l)/2))$$

$$|m(c_l \rightarrow v_m)| = \varphi\left(\sum_{j \in N(l,m)} \varphi(|m(v_j \rightarrow c_l)|)\right)$$

$$\text{sgn}(m(c_l \rightarrow v_m)) = \prod_{j \in N(l,m)} \text{sgn}(m(v_j \rightarrow c_l))$$

$$m(c_l \rightarrow v_m) = \text{sgn}(m(c_l \rightarrow v_m)) |m(c_l \rightarrow v_m)|$$

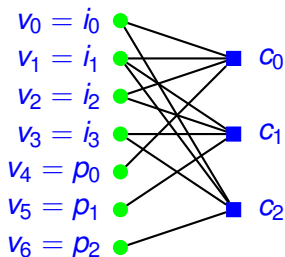
where $N(l, m)$ is the set variable nodes connected to check node c_l except for variable node v_m .

- The variable nodes receive likelihoods from each check node that it is connected to and it receives the likelihood based on the channel.
- The variable node updates its likelihood by adding all the likelihoods it receives.
- It sends back to check nodes the new likelihoods but first subtracts out the likelihood it receives from the check node it is updating.
- That is

$$m(v_m \rightarrow c_l) = L(b_m) + \sum_{p \neq l} m(c_p \rightarrow v_m)$$

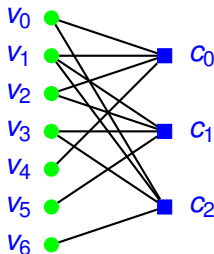
- This method of calculating the likelihood by summing is based on the assumption that all the likelihoods received correspond to independent variables.
- This is not actually true so the algorithm is not optimal.
- In the first iteration of this algorithm the variable sends only the extrinsic information to the check nodes since no information has been received from check nodes yet.
- For the Hamming code the following iterations can be used to decode the code.

STEP 1



$$\begin{array}{l}
 m(v_0 \rightarrow c_0) = L(b_0) \\
 m(v_1 \rightarrow c_0) = L(b_1) \\
 m(v_2 \rightarrow c_0) = L(b_2) \\
 m(v_4 \rightarrow c_0) = L(b_4)
 \end{array}
 \left| \begin{array}{l}
 m(v_1 \rightarrow c_1) = L(b_1) \\
 m(v_2 \rightarrow c_1) = L(b_2) \\
 m(v_3 \rightarrow c_1) = L(b_3) \\
 m(v_5 \rightarrow c_1) = L(b_5)
 \end{array} \right|
 \begin{array}{l}
 m(v_0 \rightarrow c_2) = L(b_1) \\
 m(v_1 \rightarrow c_2) = L(b_1) \\
 m(v_3 \rightarrow c_2) = L(b_3) \\
 m(v_6 \rightarrow c_2) = L(b_6)
 \end{array}$$

STEP 2



$$m(c_0 \rightarrow v_0) = 2 \tanh^{-1} [\tanh(m(v_1 \rightarrow c_0)) \tanh(m(v_2 \rightarrow c_0)) \tanh(m(v_4 \rightarrow c_0))]$$

$$m(c_0 \rightarrow v_1) = 2 \tanh^{-1} [\tanh(m(v_0 \rightarrow c_0)) \tanh(m(v_2 \rightarrow c_0)) \tanh(m(v_4 \rightarrow c_0))]$$

$$m(c_0 \rightarrow v_2) = 2 \tanh^{-1} [\tanh(m(v_0 \rightarrow c_0)) \tanh(m(v_1 \rightarrow c_0)) \tanh(m(v_4 \rightarrow c_0))]$$

$$m(c_0 \rightarrow v_4) = 2 \tanh^{-1} [\tanh(m(v_0 \rightarrow c_0)) \tanh(m(v_1 \rightarrow c_0)) \tanh(m(v_2 \rightarrow c_0))]$$

Similarly for messages from other check nodes to variable nodes.

STEP 3

$$m(v_0 \rightarrow c_0) = L(b_0) + m(c_2 \rightarrow v_0)$$

$$m(v_0 \rightarrow c_2) = L(b_0) + m(c_0 \rightarrow v_0)$$

$$m(v_1 \rightarrow c_0) = L(b_1) + m(c_1 \rightarrow v_1) + m(c_2 \rightarrow v_1)$$

$$m(v_1 \rightarrow c_1) = L(b_1) + m(c_0 \rightarrow v_1) + m(c_2 \rightarrow v_1)$$

$$m(v_1 \rightarrow c_2) = L(b_1) + m(c_0 \rightarrow v_1) + m(c_1 \rightarrow v_1)$$

$$m(v_2 \rightarrow c_0) = L(b_2) + m(c_1 \rightarrow v_2)$$

$$m(v_2 \rightarrow c_1) = L(b_2) + m(c_0 \rightarrow v_2)$$

$$m(v_3 \rightarrow c_1) = L(b_3) + m(c_2 \rightarrow v_2)$$

$$m(v_3 \rightarrow c_2) = L(b_3) + m(c_1 \rightarrow v_2)$$

$$m(v_4 \rightarrow c_0) = L(b_4)$$

$$m(v_5 \rightarrow c_1) = L(b_5)$$

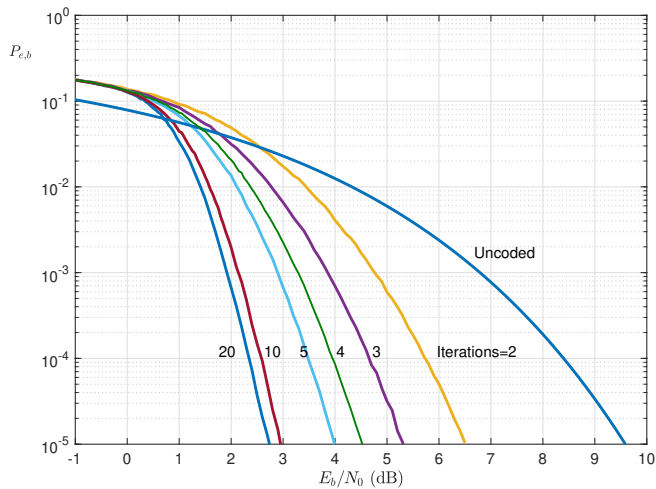
$$m(v_6 \rightarrow c_2) = L(b_6)$$

After STEP 3 go back to STEP 2 and repeat.

- The performance of LDPC codes is not amenable to analysis.
- Even the union bound is not possible to evaluate since the number of codewords such as the 2324 codewords for the short WiFi code impossible to enumerate.
- As a result, simulation is used to evaluate the performance.

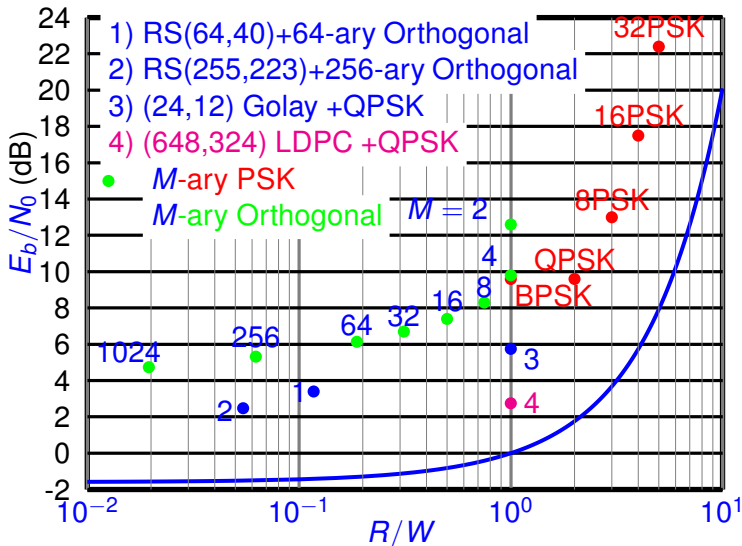
Performance

- Below the bit error probability for the (624,348) LDPC code used in WiFi is shown for an additive white Gaussian noise channel (BI-AWGN).
- This is shown for 5 iterations of the decoder and 20 iterations of the decoder.
- The gain relative to an uncoded BPSK system is roughly 8dB or in other words an 8dB reduction in required E_b/N_0 for an error probability of 10^{-5} with 20 iterations.
- The rate of the code is 1/2 so the data rate is reduced by a factor of 2 while the required energy is reduced by a factor of about 6.3 (8dB).



Comparison of Coded and Uncoded Systems

- If we make comparison between an uncoded system and a coded system with the same energy per information bit then each symbol of the coded system must have smaller energy so that p will be larger for a coded system than an uncoded system.
- Coding will be useful only if the capability to correct errors overcomes this increase in error probability caused by reducing the energy of each transmitted signal.
- Indeed, coding can provide significant gains over uncoded systems, even with the penalty of reduced energy per channel bit.
- The penalty of coding discussed so far is
 - Coding requires either higher bandwidth compared to an uncoded system of the same data rate or lower data rate compared to an uncoded system of the same bandwidth and
 - Increased complexity.

Capacity, Modulation@BER= 10^{-5} 

Block Coding Summary

First, we defined a code as a set of M vectors of length n to be used over a channel. Two important examples of channels were the additive white Gaussian noise channel and the binary symmetric channel.

Code comparison

Second, we derived the receiver that minimizes the probability that a wrong codeword is chosen. This is called the maximum a posteriori probability rule. If all the codewords are equally likely then it is equivalent to the maximum likelihood rule. For a binary symmetric channel this means finding the codeword closest to the received vector in Hamming distance. For an additive Gaussian channel the optimum receiver is to find the codeword closest in Euclidean distance to the received vector.

Code comparison

These algorithms, in general require, M comparisons of the received vector with codewords. For practical purposes this is too complex. To overcome this complexity we added some structure to the code. We first imposed linearity on the code. A linear code has $M = 2^k$ codewords of length n such that the sum of two codewords is also a codeword. For the BSC we derived the syndrome decoding algorithm for linear codes. The complexity of this algorithm was $2^{(n-k)}$. This can be a significant reduction if $n - k$ is small but k and n are large. However, for k large and $n - k$ large the complexity is still large. We imposed further structure on the code by requiring the code to be cyclic, that is require a cyclic shift of a codeword to be a codeword. While we did not derive a simpler algorithm for decoding cyclic codes, algorithms that require on the order of $(n - k)^2$ complexity (for the BSC) exist.

Code comparison

For the additive Gaussian channel, cyclic linear codes do not simplify the decoding algorithms. We can turn the additive Gaussian channel into a BSC by making hard decisions on each code symbol. This degrades the performance of the system relative to a system that does not make hard decisions but allows the use of simple algorithms. For an additive Gaussian channel with an energy constraint of E_b per information bit, we must use part of this energy for the redundant bits transmitted. That is the energy per channel symbol must be reduced by the rate of the code, i.e. $E_b = E/r$. For the hard decision channel this increases the crossover probability of the resulting binary symmetric channel. Coding will only be useful if the error correction capability is sufficient to correct the additional errors caused by the increased crossover probability of the channel relative to an uncoded system.

Code comparison

We examined the error correcting capability in terms of the minimum distance of a binary code. We also derived expressions for the codeword error probability and the bit error probability.