

12. Correcting Errors in BCH Codes

Our polynomial code is now a product of a primitive polynomial, $p(x)$, and a second polynomial $p_3(x)$ and perhaps a third, $p_5(x)$, and even more. These polynomials are defined so that the **remainder of $p_j(x^j)$ is 0** upon dividing by $p(x)$.

If we evaluate our received message polynomial $r(y)$ at $y=x^j$ we will find that the message part of that received polynomial will have 0 remainder, since it will have $p_j(x^j)$ as a factor and that factor will have 0 remainder, so we will **obtain the remainder of the sum of the j -th powers of the errors.**

We will illustrate the situation when there are two errors, but the general case can be treated in exactly the same way.

When there are two errors, suppose they are in powers e_1 and e_2 . Our task is to determine these two unknown values.

The code in the two error case is $p(x)p_3(x)$, and we have

$$r(x) = m(x)p(x)p_3(x) + x^{e_1} + x^{e_2}.$$

By evaluating the remainder of $r(x)$ and of $r(x^3)$ on dividing by $p(x)$ we obtain

$$\text{rem } r(x) = \text{rem } x^{e_1} + x^{e_2}$$

$$\text{rem } r(x^3) = \text{rem } x^{3e_1} + x^{3e_2}.$$

Our task is to find the two error powers, e_1 and e_2 , from these two pieces of information.

We will do so by finding a magical “error locator” polynomial, which will have 0 remainder when evaluated at powers e_1 and e_2 , and non zero remainder otherwise.

Finding the errors then involves evaluating the remainder of this error locator polynomial at each of the powers of x in turn. If we change each bit for which this polynomial has 0 remainder, we will correct the errors.

We will see exactly how to do this once we have this polynomial. First though, we will investigate how we describe the polynomial and how we determine it.

If we let y be our variable, the polynomial which will vanish (that is, have vanishing remainder) for $y=x^{e_1}$ and $y=x^{e_2}$, will have to obey

$$\text{rem } (y - x^{e_1})(y - x^{e_2}) = 0$$

or

$$\text{rem } y^2 - (x^{e_1} + x^{e_2})y + x^{e_1}x^{e_2} = 0.$$

Notice that the coefficient in this equation will necessarily be powers of x , and not just 0 or 1.

Notice also that we can determine this equation by determining the two coefficients $(x^{e_1} + x^{e_2})$ and $x^{e_1}x^{e_2}$, that is, by determining their remainders.

Finally, notice that the remainder of the first of these two coefficients is exactly what we do determine when we find the remainder of $r(x)$ on dividing by $p(x)$.

So all we need do to find this error locator equation is to determine the remainder of $x^{e_1}x^{e_2}$, from the information we have, which information consists of the remainders of the sums of the first and third powers of the error monomials.

We now introduce some definitions. Notice that here and in general, the coefficients of the error locator polynomials will each

be characterized by its degree.

There will be a linear term, (which will be the coefficient of y^{n-1} which here is y) a quadratic term, (a term of degree three and so on)

The linear term is always just the sum of the error monomials. The quadratic term will always be the sum of all product $x^{e_i} x^{e_j}$ with $i < j$; the next term will consist of the sum of all products $x^{e_i} x^{e_j} x^{e_k}$ for $i < j < k$, and so on.

These coefficients are called the “**elementary symmetric functions**” of the error monomials. **We will denote them by s_1, s_2, \dots ,**

Thus, our error locator polynomial can be written, when there are two errors, as

$$\text{rem } y^2 + s_1(x) y + s_2(x) = 0,$$

or, with 5 errors

$$\text{rem } y^5 + s_1 y^4 + s_2 y^3 + s_3 y^2 + s_4 y + s_5 = 0.$$

On the other hand, the information that we do have, that we get from $r(x)$, are the “**power sums**” of the error monomials, or at least the odd **power sums** of the remainders. **We will denote these as t_j .** That is, t_j is the sum over all errors e_k of $x^{j e_k}$.

So our task in general is to go from the sums of powers of our error monomials to the elementary symmetric functions of these same monomials, which are the coefficients in the error locator polynomial.

By the way, because we are using binary addition with $1+1=2$, we have $t_{2j} = t_j^2$, and so we get the even power sums without much work from the odd ones.

We already know some things about the relations between the s 's and the t 's, which is the key to our error correcting.

In particular, we know $s_1 = t_1$ since both are the sum of the error monomials.

To correct two errors we can get a second relation between s 's and t 's as follows. Take the defining equation for s_1 and s_2 :

$$\text{rem } y^2 + s_1 y + s_2 = 0$$

when y is an error monomial;

multiply through by y , evaluate at $y=x^{e_1}$ and $y=x^{e_2}$ (which are supposed to obey this equation), and sum the two.

We get

$$\text{rem } x^{3e_1} + x^{3e_2} + s_1 (x^{2e_1} + x^{2e_2}) + s_2 (x^{e_1} + x^{e_2}) = 0,$$

or

$$\text{rem } t_3 + s_1 t_2 + s_2 t_1 = 0.$$

This equation, with the facts: $t_2 = t_1^2$, $s_1 = t_1$, gives us

$$\text{rem } s_2 = (t_3 + t_1^3)/t_1.$$

This equation gives us the ability to correct errors when there are at most two of them. The procedure is as follows.

We find $\text{rem } t_1$ (same as $\text{rem } s_1$) and $\text{rem } t_3$ from $r(x)$ as previously discussed. **We then find $\text{rem } s_2$ from this last equation, and then check each power to see if it obeys the error locator equation.**

We still have to describe how we do these two things.\

We will proceed as follows:

First we will discuss how we can find s_2 from the equation just above.

Then we discuss how to set up the power tester to find and correct errors.

11.2 Finding s_2 .

The equation for s_2 just obtained requires that we be able to add and multiply and divide out remainders. We can do these things, but you should not take that for granted.

We add remainders by treating them as vectors and adding with $2=0$, as we have been doing.

We must also **multiply** our remainders. You should realize that the only way we can multiply remainders A and B is to consult our remainder table for the polynomial $p(x)$, and determine which powers have these remainders; if they are a and b , so that $\text{rem } x^a = A$ and $\text{rem } x^b = B$, we determine z by $z = a + b \pmod{2^k - 1}$. The remainder that is the product of A and B will be the remainder of x^z . Our remainder table is like a table of logarithms. To multiply you need only add the powers.

The reason we have to consult the remainder table to find products is that our products are defined among remainders obtained when dividing by a specific primitive polynomial. If we divide by a different primitive polynomial, the rules for multiplying remainders will be different. Thus, the rule cannot be deduced from the remainders themselves, but requires information about which powers get which remainders. This information is all contained in our remainder table.

So what should we actually do to find errors, given remainders t_1 and t_3 ?

STEP 1: find t_1^3 ; to do this,

form the identifier for t_1 ;

then find the power that has remainder with this identifier;

then triple that power mod $(2^k - 1)$ (here k is the degree of $p(x)$.)

then find the identifier for this power.

And then find the remainder from this identifier.

(There is an easier way to do this when you have an ordinary remainder table and a going up by 3 one side by side. Then when you find the identifier for t_1 you can **find the identifier in the up by 3 table in the row having the given identifier in the ordinary table**.

That will be the identifier for t_1^3 .)

STEP 2: find $t_3 + t_1^3$; to do this

Add the remainders of t_3 and t_1^3 mod 2. That will be the remainder of $t_3 + t_1^3$.

STEP 3: find $(t_3 + t_1^3)/t_1$; to do this

Find the identifier of $(t_3 + t_1^3)$ and its power.

If t_1 has identifier 0 then there are no errors, and we can stop

Add $2^k - 1$ – (the power having remainder t_1) to the power of $(t_3 + t_1^3)$, mod $(2^k - 1)$.

Find the indicator and remainder of this power

You can see from this description that performing additions and multiplications here involves repeated application of the following steps

Going from a remainder to its identifier; (if the remainder is (a,b,c,d,e) this consists of forming $a+b*2+c*4+d*8+e*16$.)

Going from an identifier id to a power $p(id)$; this involves a look up in the remainder table. You can do this on a spreadsheet by forming a column anywhere to the right of the remainder table with the entry $=if(id=id(j),j,0)$ and summing this over the column. **The sum will be $p(id)$. This method however will fail if $id=0$. So the sum should really be replaced by $if(id=0,"no power", sum over column)$**

Going from a power p to an identifier $id(p)$; you can do this on a spreadsheet by forming a column to the right of the remainder table with the entry $=if(p=j,id(j),0)$ and summing it. **The sum will be $id(p)$. If power = "no power" then the answer will be 0. If that comes out as an error in your spreadsheet it should be converted to 0.**

Going from an identifier id to a remainder.

This can be done as follows

With id in any location put under it, $=mod(id,2)$ and next to id on the right put $=(id-mod(id,2))/2$

And copy those two instructions to the right a so that there are k columns all together.

The lower row will be the desired remainder.

(Please note here that when we write something like $=mod(id,2)$ on a spreadsheet you should enter the location of id on the spreadsheet and not the symbols id .)

11.3 Having formed s_1 and s_2 what then?

The next task is to set up a power tester. You can do this by setting up **three remainder- like tables, one for each term in the error locator polynomial.**

The first, the y^2 term table, starts with $1000\dots$ (0^{th} power) and **powers go up by 2** from row to row.

In the second, or $s_1 y$ term table, the 0^{th} power term is the remainder of s_1 , and **powers go up by 1 as in an ordinary remainder table.**

In the third or s_2 table the 0^{th} power term is the remainder form of s_2 , and it is the same throughout the table.

You next form the sum mod 2 of these three tables, and an identifier column for it.

The rows for which the identifier is 0 correspond to the error powers. Switching 0 and 1 in thee rows of the message will correct the message.

There are two problems with what we have been doing. The first is that there may have been more than two errors, and the received word is not Hamming distance 2 from any code word.

If there are more than two errors, our second equation which relates s 's and t 's will not be correct, and any corrections we find using it will be wrong.

Sometimes we will correct to the wrong message, and we lose. But some of the time we will end up with a received word that is not a distance 2 from any message word. In that case we will try to correct, but our corrections will not work. So our "corrected message" will not be divisible by pp_3 .

We can check for this by taking the remainder of our corrected message. If it has non-zero remainder, our corrections will have failed. We can recognize that and detect our failure.

A second problem is that we left the no-error case dangling a bit.

We can handle this in either of two ways.

One is to check early if $\text{rem } r(x) = \text{rem } r(x^3) = 0$, and see that we do not correct the when this happens, by making all corrections contingent on this not happening.

(if $r(x)$ has 0 remainder and $r(x^3)$ does not, there must be at least 3 errors and this should also be noted)

Another way is **to multiply the top ($y(x^0)$) entries of the three tables corresponding to the three terms in the error locator polynomial each by t_1** . Then they will be, in order, $t_1, t_1^2,$ and $t_1^3+t_3$ (instead of 1, $t_1,$ and s_2).

In this case the step of dividing by t_1 to get s_2 is avoided and t_2 which is t_1^2 can be determined from an up-by-two table. If it is, **and we used the trick for finding t_1^3 described in STEP 1**, no multiplications are necessary to find s_1 and s_2 , and so we are not disturbed in our procedure.

(We **would be** troubled if we multiply or divide anything by t_1 when t_1 has 0 remainder. The problem is that we multiply or divide by t_1 by converting it to a power, and if it is 0 it isn't any power. If we are not careful, the method we use for converting to a power will convert it to the power that will be the sum of all 0's, and that will come out to be the 0th power, when it should really be "no power at all".)

When t_1 and t_3 both have 0 remainder every term in each table will be 0, and having them sum to 0 will occur in every row. We therefore **only want to correct a bit if its row sum of these three rows is 0 and the sum of these row sums over all rows is not 0**.

11.4 Can This really be Done?

The following illustrates the error correcting procedure with tricks that can be used, here starting with the primitive polynomial $1+x+x^4$. First we compute p_3 which involves row reducing in the following matrix.

computation of p_3

power	0	1	2	3
	0	1		
	3			1
	6		1	1
	9	1		1
	12	1	1	1

Then we compute the row to row transitions with powers increasing by 1 2 and 3.

transition in remainder table				transition in up by 3 table			
a	b	c	d	a	b	c	d
d	a+d	b	c	b	b+c	c+d	a+d
transition in up by 2 table							
a	b	c	d				
c	c+d	a+d	b				

Then we compute the remainder table and up by two and three tables:

rem table					
power	0	1	2	3	id
0	1	0	0	0	1
1	0	1	0	0	2
2	0	0	1	0	4
3	0	0	0	1	8
4	1	1	0	0	3
5	0	1	1	0	6
6	0	0	1	1	12
7	1	1	0	1	11
8	1	0	1	0	5
9	0	1	0	1	10

1	0	1	1	0	
1	0	0	1	0	
4	0	3	2	0	12 0
0	0	1	0	4	0 0

And here are the formulae that led to it
r dot rem table

0	1	2	3		get t1^3
=\$E10*H10	=\$E10*I10	=\$E10*J10	=\$E10*K10		=IF(L10=AH\$26,T10,0)
=\$E11*H11	=\$E11*I11	=\$E11*J11	=\$E11*K11		=IF(L11=AH\$26,T11,0)
=\$E12*H12	=\$E12*I12	=\$E12*J12	=\$E12*K12		=IF(L12=AH\$26,T12,0)
=\$E13*H13	=\$E13*I13	=\$E13*J13	=\$E13*K13		=IF(L13=AH\$26,T13,0)
=\$E14*H14	=\$E14*I14	=\$E14*J14	=\$E14*K14		=IF(L14=AH\$26,T14,0)
=\$E15*H15	=\$E15*I15	=\$E15*J15	=\$E15*K15		=IF(L15=AH\$26,T15,0)
=\$E16*H16	=\$E16*I16	=\$E16*J16	=\$E16*K16		=IF(L16=AH\$26,T16,0)
=\$E17*H17	=\$E17*I17	=\$E17*J17	=\$E17*K17		=IF(L17=AH\$26,T17,0)
=\$E18*H18	=\$E18*I18	=\$E18*J18	=\$E18*K18		=IF(L18=AH\$26,T18,0)
=\$E19*H19	=\$E19*I19	=\$E19*J19	=\$E19*K19		=IF(L19=AH\$26,T19,0)
=\$E20*H20	=\$E20*I20	=\$E20*J20	=\$E20*K20		=IF(L20=AH\$26,T20,0)
=\$E21*H21	=\$E21*I21	=\$E21*J21	=\$E21*K21		=IF(L21=AH\$26,T21,0)
=\$E22*H22	=\$E22*I22	=\$E22*J22	=\$E22*K22		=IF(L22=AH\$26,T22,0)
=\$E23*H23	=\$E23*I23	=\$E23*J23	=\$E23*K23		=IF(L23=AH\$26,T23,0)
=\$E24*H24	=\$E24*I24	=\$E24*J24	=\$E24*K24		=IF(L24=AH\$26,T24,0)
	=SUM		=SUM		
=SUM(AD10:AD24)	(AE10:AE24)	=SUM(AF10:AF24)	(AG10:AG24)	=SUM(AH9:AH24)	=SUM(AI9:AI24)
				=AD26	
				+AE26*2	
=MOD	=MOD		=MOD	+AF26*4	
(AD25,2)	(AE25,2)	=MOD(AF25,2)	(AG25,2)	+AG26*8	=MOD(AI25,2)

We omit the similar tables for t₂ and t₃

We then compute t₁³ + t₃: after extracting rem t₁³ from its identifier''

0t3	0	1	0	0
t1^3	0	0	1	1
t1s2	0	1	1	1
	12	6	3	1

The formulae for this are, the last line being used to get rem t₁#.

t3	=MOD	=MOD	=MOD	=MOD
	(AL25,2)	(AM25,2)	(AN25,2)	(AO25,2)
t1^3	=MOD	=MOD	=MOD	=MOD
	(AL29,2)	(AM29,2)	(AN29,2)	(AO29,2)
t1s2	=MOD	=MOD	=MOD	=MOD
	(AL26	=MOD(AM26	(AN26	(AO26
	+AL27,2)	+AM27,2)	+AN27,2)	+AO27,2)
	=(AL29-	=(AM29-	=(AN29-	
	=AI25	AL27)/2	AM27)/2	AN27)/2

Finally we construct the tables of the error locator polynomial.

We illustrate the last two, namely the constant term and the sum

t3+t1^3 table				sum table			id	
0	1	1	1	1	0	0	1	9
0	1	1	1	1	1	0	1	11
0	1	1	1	0	1	1	1	14
0	1	1	1	0	0	0	0	0
0	1	1	1	0	0	1	1	12
0	1	1	1	1	1	0	1	11
0	1	1	1	0	0	0	0	0
0	1	1	1	0	1	0	0	2
0	1	1	1	1	0	0	1	9
0	1	1	1	1	0	1	0	5
0	1	1	1	0	0	1	1	12
0	1	1	1	1	0	1	0	5
0	1	1	1	0	1	0	0	2
0	1	1	1	1	1	1	0	7
0	1	1	1	1	1	1	0	7

106

The first two terms had the following tables

error locator polynomial term 1

t1*up by 2 table				t2*up by 1 table			
0	0	1	0t2	1	1	0	0
1	1	0	0	0	1	1	0
0	0	1	1	0	0	1	1
1	0	1	0	1	1	0	1
1	1	1	0	1	0	1	0
1	1	1	1	0	1	0	1
1	0	0	1	1	1	1	0
0	1	0	0	0	1	1	1
0	0	0	1	1	1	1	1
0	1	1	0	1	0	1	1
1	1	0	1	1	0	0	1
0	1	0	1	1	0	0	0
0	1	1	1	0	1	0	0
1	0	1	1	0	0	1	0
1	0	0	0	0	0	0	1

The remaining steps are checking, and dividing. .

We will next discuss generalization of this procedure to correct more errors. (It can be accomplished in exactly the same way), why it works, properties of these codes and further generalizations.

Exercise: Construct a coder and error corrector that corrects two errors with your favorite degree 5 or degree 6 primitive polynomial, (degree 6 is no harder than degree 5 or degree 4).

1. choose a primitive polynomial and construct its remainder table
2. construct p_3 from it
3. construct an encoder that multiplies first by p and then by p_3 (or however you choose to multiply)
4. construct up by two and up by three power-remainder tables
5. take dot products of the received word with each of your tables.
6. find $t_1^3+t_3$.
7. set up the tables for the error locator polynomial terms and their sum.
8. correct the errors.

9. divide twice to get the message back.

You don't have to do it exactly this way but it seems easiest to me.