Galois Fields and Cyclic Codes

Phil Lucht
Rimrock Digital Technology, Salt Lake City, Utah 84103
last update: Aug 31, 2013
rimrock@xmission.com

Maple code is available upon request. Comments and errata are welcome.

The graphics look ratty in Windows Adobe PDF viewers when not scaled up, but look just fine in this excellent freeware viewer: http://www.tracker-software.com/pdf-xchange-products-comparison-chart .

The table of contents has live links.

"Now, what I want is, Facts. Teach these boys and girls nothing but Facts. Facts alone are wanted in life. Plant nothing else, and root out everything else."

Charles Dickens, *Hard Times*

Preface

This document is written for readers who are non-experts in modern algebra and coding theory. It is mainly a "theory" document, but still contains many down-to-earth examples. There are no discussions of detailed error-correction implementations as are found in books on error correction (e.g., Rhee), but the tight connection between Galois fields and cyclic codes is hopefully made very clear.

In some coding texts, the review of modern algebra is so brief and dense that comprehension is quite difficult, especially for someone totally unfamiliar with the subject. Conversely, in some math books the discussion of modern algebra is so comprehensive that one is forced to invest in many concepts unnecessary for Galois field applications We have attempted to bridge this gap.

The wonderfully efficient and dense mathematical notations like $\in$ $\forall$ | $\exists$ $\Leftrightarrow$ iff are generally replaced by words. Some well-known theorems are not proved, but those that are proved are treated with a (hopefully) reasonable amount of rigor. Lots of "words" are used to reinforce the various concepts, many examples are provided, and there is constant (perhaps excessive) repetition to grind in definitions and "facts" (Mr. Thomas Gradgrind, schoolmaster, is the character quoted above).

After doing manual examples, it is often shown how algorithms can be automated using very simple Maple programs, Maple being a commercial symbolic computer algebra system. Freeware systems exist (see wiki) and our short Maple programs can easily be translated into other languages which support the constructs used.

The structure of a document like this one involves certain design issues.

On the one hand, if a simple Fact applies to a larger class of objects than we are really interested in, but if proving the Fact for that larger class of objects is no harder than for the smaller class, one might as well prove the Fact in its more general application. The reader is then forced to learn somewhat more than necessary, but this seems fairly harmless. Some Facts are so simple to prove, and perhaps so interesting, that they are included in the forest of Facts, even though they are not directly needed on this particular voyage through the forest.

On the other hand, one might argue that the forest then becomes so cluttered with trees that a voyager loses track of where he or she is going, and which trees are important and which are not. The relative importance of various trees only becomes clear later in the trip when certain applications of the Facts are considered. It is useful to pause from time to time and review the trip up to the current point of rest, and that is done in several places in the document.

In an earlier version of this document, there were no equation numbers but the Facts in each chapter had Fact numbers starting with Fact 1. Although now redundant, most of these Fact numbers have been retained. One might then see Fact 4 (7.35) as a cross-reference. Equation numbers of the form (3.14) are applied to equations, Facts and certain definitions. When a numbered item is quoted later in the document, the equation number is put in italics. Some proofs end with the letters **QED** so the reader knows where the text flow continues.

Summary

This is a detailed summary. The reader is directed to the Table of Contents for a more concise overview.

Chapter 1 [**Algebra**] is a partial review of Modern Algebra which includes only concepts that will be needed in later sections. The basic subjects here are groups, fields, rings and ideals. The so-called residue class ring is formed as R/I where R is a ring and I is an ideal. In particular, Z/(n) is such a residue class ring where Z are the integers and (n) is the ideal which consists of integers which are multiplies of n. It is shown that this residue class ring is isomorphic to the ring of integers mod n, called Z_n. It is then shown that if n is a prime number p, the rings Z/(n) and Z_n are fields. Many "facts" are accumulated in this chapter, and most of them have analogs in the polynomial world introduced in Chapter 3.

Chapter 2 [GF(p)] provides more information about Z_n with a few examples. It then shows that the fields Z/(p) and Z_p are isomorphic to the Galois Field GF(p). The field operations of GF(p) = Z_p = Z/(p) are here called • and + and we learn how to construct the addition and multiplication tables for GF(p). Various facts about GF(p) are then developed. Some authors refer to Galois Fields simply as finite fields, since that is what they in fact are. The argument of GF(*) denotes the number of elements in the finite field. Two of these elements are always 0 and 1.

Chapter 3 [**Polynomials**] discusses another ring, the ring of polynomials R whose coefficients lie in Z_p= GF(p). In this chapter, the operations of GF(p) are called $\oplus$ and $\otimes$. Basic facts concerning such polynomials are developed in analogy with similar properties of integers presented back in Chapter 1. Within the polynomial world, the notion of an irreducible polynomial is introduced and is seen to be analogous to the notion of a prime number in the integer world. An ideal within the polynomial ring R, called (f(x)), consists of all polynomials which are multiples of a polynomial f(x). Then, just as in Chapter 1 for integers, here the residue class ring R/(f(x)) is shown to be a field when f(x) is an irreducible polynomial in R.

In **Chapter 4** [**GF(q)**] it is shown that, if irreducible polynomial f(x) is of degree m, then the field R/(f(x)) is in fact isomorphic to Galois Field GF(p^m). Each element of GF(p^m) can be associated with a possible remainder polynomial which is obtainable when a polynomial in R is divided by f(x). Since these remainder polynomials are of degree < m and have coefficients in GF(p) = Z_p, there are p^m of them. Each remainder polynomial, and thus each element of GF(q), can be represented as an m-tuple of Z_p elements which are the remainder polynomial coefficients.

Since the only finite fields that exist are these GF(p^m) where p is a prime number and m a positive integer, we have at this point a "model" (realization, representation) for *all* the Galois Fields. A method for determining the + and • tables for any GF(q = p^m) is then developed based on this model.

There follows a lengthy discussion of the notion of cyclic groups with respect to the GF(q) fields, and after much work it is shown that the non-zero elements of every Galois Field GF(q) form a cyclic group with respect to the • operator. This means that it is possible to find some element in GF(q) (a generator) whose powers enumerate all non-zero elements of the field, an extremely useful fact. This is the "power basis" and serves as a second method of labeling elements of GF(q), the first being the m-tuple basis mentioned above. Each basis leads to a different labeling of the headings of the + and • tables for GF(q).

It is then shown that the polynomial $x^q - x$ can be written as a product of q factors of the form $(x-a_i)$ where the a_i are the q elements of GF(q):

$$(x^q - x) = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_q) \ .$$

In other words, $x^q - x$ can be fully factored in GF(q). Another way to say this is that the q elements of GF(q) are all roots of the polynomial $x^q - x$. This implies that $\alpha^q = \alpha$ for any α in GF(q).

The chapter then closes with a few facts about GF(q) similar to those presented at the end of Chapter 2 for GF(p). It is shown that GF(p) is a subfield of $GF(p^m)$ and both these fields have the same 0 and 1 elements. In some ways, this is similar to the real numbers being a subfield of the complex numbers, those two fields of course being infinite fields and thus not Galois fields.

Chapter 5 [Minimum and Primitive Polynomials] broaches the topic of the minimum polynomial m(x) of an element α of GF(q). Such a polynomial is simply a portion of the above displayed product of factors $(x-a_i)$ which includes $(x-\alpha)$ and includes the smallest set of other $(x-a_i)$ factors which, when multiplied out, results in m(x) having coefficients all lying in Z_p = GF(p). In general, some *arbitrary* product of the $(x-a_i)$ factors will form a polynomial with coefficients in GF(q) which contains GF(p). Since the factor $(x-\alpha)$ is included, $m(\alpha) = 0$. It is shown that any minimum polynomial is irreducible in GF(p). It turns out that a given minimum polynomial m(x) is the minimum polynomial of all the GF(q) elements a_i that appear in those other $(x-a_i)$ factors which make up its portion of the fully factored $x^q - x$. The set of GF(q) elements for which some m(x) is the minimum polynomial is called a conjugate set. If element α of GF(q) is a primitive element of GF(q), meaning its powers can enumerate all the non-zero elements of GF(q) as noted above, then the minimum polynomial of α is called a primitive polynomial of GF(q). We determine exactly how many primitive polynomials GF(q) has. The question of how minimum and primitive polynomials are determined is then discussed with various examples. A primitive polynomial f(x) of $GF(q=p^m)$ is always of degree m, and can therefore serve as the f(x) in the residue class ring R/(f(x)) which represents GF(q). It is shown that the conjugate sets partition the elements of GF(q), and this is then related to the subject of cyclotomic cosets. Section (i) discusses certain products of minimum polynomials which will appear later in the theory of BCH codes. Section (j) shows that the order of an element of GF(q) is equal to the period of the corresponding minimum polynomial. Finally, section (k) provides a short Maple program which generates all the minimum polynomials (in factored form) of any Galois Field.

Chapter 6 [GF(q) Enumeration Table] shows how one determines the "enumeration table" of any Galois Field $GF(q=p^m)$. This table is based on a selected primitive element α and its corresponding primitive polynomial m(x). Since $m(\alpha) = 0$, this equation gives a way to express α^m as a sum of lower powers of α times coefficients. One first enumerates all the non-zero elements of GF(q) as powers of primitive element α up to power α^{q-2}, and then one uses this α^m reduction equation to express the higher powers in terms of lesser powers, and the result is the "table" for GF(q). The table is useful because, once it is known, one can immediately obtain the addition table for the field GF(q). The multiplication table in this "powers basis" is completely trivial.

In section (c) a very simple Maple program is used to directly construct enumeration tables for several GF(q) fields. Given the enumeration table for a field, section (d) shows how one can then expand the factored minimum polynomials of Chapter 5 to verify that they do indeed have coefficients in GF(p). Along the way we show how Maple can be used to accomplish various tasks like factoring a polynomial

in GF(q), finding roots, and multiplying and dividing polynomials in the ring of polynomials R whose coefficients lie in Z_p= GF(p). The final section (e) provides an example of how the + and • tables for GF(2^2) are computed using the GF(2^2) enumeration table.

With **Chapter 7 [Block Codes]**, there is a rather sudden shift in topic. This chapter provides the basic facts of the linear block codes (n,k) which are used in forward-error-correction systems. In each block of such a code, k data symbols are combined with n-k parity check symbols to form a code word of n symbols The immediate Galois connection is that the all these symbols are assumed to be elements of some GF(q). For example, for GF(2) the coding symbols are bits, and for GF(2^8) they are bytes. The codes are *linear* because the n code word symbols in a block, treated as a vector **c**, are generated by the application of a generator *matrix* G to the vector **d** of data symbols, **c** = G**d**.

When a transmitted code word **c** is received at the end of a "transmission", one or more of the symbols might have been damaged by the effect of "noise". The damaged code word **c'** will normally not be in the allowed "code book" of legal code vectors **c**, and then the parity check symbols can possibly be used to correct **c'** back to **c**. A certain parity check matrix H which is related to the generator matrix G is used to check incoming code words for errors. If H**c'** = **0**, then **c'** is a valid code word. If H**c'** = **s** ≠ 0, there has been some kind of error. The vector **s** is called the syndrome, and it can be used in various schemes to correct the error.

Various drawings are used to illustrate how code words appear as points in an embedding vector space V^n which contains many non-code-word points. The closest pair of code words in V^n are separated by a certain distance d known as the Hamming distance of the code. This allows there to exist a private sphere of protection around each legal code point of radius t ≈ d/2 (t = Int[(d-1)/2]). The integer t is the number of bad symbols per code word that the code can correct. Within the sphere of radius t, any non-code word gets corrected to the legal code word at the center of the sphere.

The chapter concludes with a brief history of coding theory including mention of non-block codes.

In **Chapter 8 [Cyclic Codes]** we have a mighty confluence of the flowing river of Chapters 1-6 with the tributary of Chapter 7. The signpost overlooking this confluence reads "Galois Cyclic Codes". It is shown how the data and code words of the block codes of Section 7 appear as coefficients of polynomials of the type described in Chapter 3, but now the coefficients are in general elements of GF(q=p^m) instead of GF(p). The ring of such polynomials is called R_q. The full power of the theory of Galois Fields is brought to bear in the theory of cyclic codes; almost every concept of all earlier chapters makes an appearance. The codes are cyclic because a rotation of any code word's symbols by any number of places generates a new code word, but this fact lays hidden in the background until section (f).

In the (n,k) cyclic code world, the block-code generator matrix G is replaced by a generator polynomial g(x) of degree n-k, and the formal encoding process is c(x) = g(x)d(x). Here the k data symbols are the coefficients of d(x), and the n code word symbols are the coefficients of c(x). In this so-called cyclic basis, the parity check symbols are convoluted into the n c(x) coefficients. An equivalent basis called the systematic basis encodes a little differently and has implementation benefits since the data symbols are exposed in the code words. Section (c) discusses how encoders and decoders are implemented, but only at a very high level. Section (d) shows how CRC works and discusses the types of errors that can be detected. Section (e) ferrets out the fact that the two bases just mentioned are rearrangements of each other. Then section (f) shows that the cyclic codes as defined in this chapter are in fact cyclic.

In section (g) the dormant residue-class-ring machinery introduced in Chapter 1 and used in Chapter 3 is powered up again, this time in double overdrive. First, a nameless ring A_n with q^n elements is defined as $R_q/(x^n-1)$; its elements are associated with remainder polynomials of degree < n and can be regarded as the points of the code-embedding vector space V^n. This ring A_n contains the q^k code word polynomials c(x) along with the q^n-q^k "illegal" code word polynomials, all mixed together. In a second residue-class-ring application, the elements of A_n are used in $A_n/(g(x))$ to form what is called the standard array. In this array, all the legal code words are rounded up into one bin which is the ideal (g(x)), the first row of the standard array. Then in section (h) it is shown how this standard array "does" error correction. It is a rather amazing logical thread.

Section (i) then shows the form of the parity check matrix H for a cyclic code, and gives a top level picture of the code families associated with the names BCH, Hamming and Reed-Solomon. The first and last code families can be designed to correct an arbitrary number t of symbol errors in a code word and are very efficient at doing it. Section (j) explains why one might want the coefficients of the generator polynomial g(x) to lie in G(p) rather than G(q). The reason is that this makes the design of hardware polynomial multipliers and dividers extremely simple. This desire to have g(x) have coefficients in GF(p) is a major driver for all the work of Chapter 5 on minimal polynomials, as is seen in Chapter 9. The final section (k) shows that, for our Galois-induced cyclic codes, one can generate all the code words of the code by rotating any one of them, as long as the generator polynomial is a primitive polynomial.

Chapter 9 [Small Code Survey] states the BCH Bound Theorem and defines the BCH cyclic codes as those which optimize this theorem. The theorem provides a *floor* for distance d and therefore error-correcting ability t. The order of a BCH code with respect to GF(q) is always n = q-1 = p^m-1, whereas the k value of the (n,k) code designation is an integer forced by the code. The so-called narrow-sense BCH codes are a special case which have the coefficients of g(x) in GF(p). The generators $g_N(x)$ of such codes are the "least common multiples" of N of the minimum polynomials described in Chapter 5. An example with GF(2^5) is worked out in some detail, providing several BCH codes with code length n = 31. A simple subset of the narrow-sense BCH codes have g(x) = a single minimum polynomial, and from this subset various Hamming codes are derived, though historically the Hamming codes were known before the BCH codes revealed their Galois Field underpinnings. Finally it is shown how the Reed-Solomon codes have g(x) coefficients in GF(q) rather than GF(p) and, despite increased implementation costs, are able to correct the most possible errors that any (n,k) code can correct ; they are maximum-distance separable. The final section (e) comments on how Galois logic uses the algebra of GF(q), whereas digital filters use that of Z_q.

Chapter 10 [Matrix Representation of GF(q)] is another change of topic. It is shown how one can easily construct a representation of any GF(q=p^m) field as a set of q mxm matrices whose elements lie in GF(p). Two critical ingredients are the Cayley-Hamilton Theorem and the notion of a Companion Matrix. A very simple Maple program is then presented which generates a set of q matrices which represent any GF(q) Galois field.

(continued on next page)

Appendix A gives a proof of a certain claim (4.29) made in Chapter 4 upon which is based the derivation of the important fact that all Galois fields are cyclic.

Appendix B explains the basic nature of the conjugate set of α as encountered in Chapter 5.

Appendix C evaluates the polynomial product a(x)b(x).

Appendix D is a brief matrix review in support of Chapter 10.

Appendix E discusses how cyclic code generators g(x) can be found. Then through a guided thread of Facts, it shows that, for a code using symbols in GF(p) and for code length $n \neq Np$, an irreducible g(x) exists and is in fact a minimum polynomial of $GF(p^{\varphi(n)})$ where φ(m) is Euler's totient function.

Appendix F discusses in detail the nature of the errors that a CRC error detection system system can detect. CRC means Cyclic Redundancy Check and is used for example to protect Ethernet packets.

Appendix G is a brief excursion into the dense terrain of Number Theory resulting in the derivation of two classic results: Euler's Theorem and Fermat's Little Theorem. Euler's totient function φ(n) is defined and it is shown how φ(n) determines the number of primitive elements of a Galois Field and also the number of primitive polynomials.

A few **References** are then provided.

Chapter 1: Modern Algebra

The purpose of this chapter is to point out and give recognizable names to some of the animals which inhabit the landscape of modern algebra. We do so in a non-rigorous manner, because we want to get done as fast as possible. We try to include only those ideas that are necessary for our development to come in later chapters. Not everything is proved, because the proofs all exist in standard texts. We are more interested in showing how things are defined and how they fit together. The reader looking for more on this subject would do well to consult Birkhoff and MacLane (see References).

When a word of phrase is being defined, that word or phrase is put in bold font to make the definition easier to locate later on. ("Now let's see, what exactly *was* a primitive polynomial?")

(a) Groups, Fields and Rings

A **group** G is a set of elements g (a,b,c,...) and an operation * such that the following are true:

Closed under	Associative	Identity exists	Inverse exists	Commutative
*	(a*b)*c=a*(b*c)	"1": 1*g = g*1 = g	"g^{-1}": $g*g^{-1} = g^{-1}*g = 1$	a*b = b*a ?

GROUP (1.1)

Closed under some operation means that if one applies the operation to two elements in some set, the result is also in that set. **Associative** says (a*b)*c=a*(b*c) regardless of the order in which the pairwise operations are carried out. Notice that the order a,b,c must be the same on both sides. The **identity** must exist as a "two sided" identity, so that 1*g = g*1 = g . Similarly, the **inverse** must exist as a two-sided inverse $g*g^{-1} = g^{-1}*g = 1$. The notation g^{-1} for an inverse is written as (-g) if the operation * is "addition", but we can regard g^{-1} as a generic notation valid for all * cases. Similarly. "1" is written 0 for addition, but again we regard "1" as a valid generic notation for all cases.

The **power notation** g^2 means g*g, g^3 = g*g*g and so on. According to the associative rule, $g^2g = gg^2$ and in general $g^ng^m = g^mg^n = g^{n+m}$. A group element g always "commutes with itself". In the case of addition, g^3 = g+g+g = 3g, and so we might use the specific notation 3g instead of the generic notation g^3. To emphasize that g is the group element and 3 is just an integer, we might sometimes write $\mathbf{g}^3 = 3\mathbf{g}$ in the additive case (g bolded).

Sometimes we might informally write g_1g_2, but this really means $g_1 * g_2$.

Fact: For any group, the identity not only exists, but is unique. (1.2)

Proof: Suppose we had two identities 1_a and 1_b, Since 1_a is an identity, we have $1_a * 1_b = 1_b$. Since 1_b is an identity, we have $1_b * 1_a = 1_a$. But since an identity must be two-sided, $1_a * 1_b = 1_b * 1_a$ so $1_a = 1_b$.

Examples: Suppose $g_1 * g_2 = g_2$ for some particular g_1 and g_2. Then it must be that $g_1 = 1$.
Suppose $g * g^m = g^m$. Then it must be that g = 1. So if $g \neq 1$, then g^{m+1} and g^m must be different elements. So for example we know that $g^3 \neq g^2$ and $g^4 \neq g^3$. It might be, however, that $g^2 = g^4$.

Fact : For any group, the inverse of any element g, called g^{-1}, not only exists, but is unique. (1.3)

Proof: Suppose g had two inverses g^{-1}_a and g^{-1}_b. Then $g^{-1}_a\, g = 1$. Right multiply by g^{-1}_b to get $(g^{-1}_a * g)* g^{-1}_b = 1* g^{-1}_b$. From associative and identity rules this says $g^{-1}_a * (g* g^{-1}_b) = g^{-1}_b$. But we know that $g* g^{-1}_b = 1$ so we get $g^{-1}_a * 1 = g^{-1}_b$ and the identity rule then gives $g^{-1}_a = g^{-1}_b$.

Finally, if the operation * is **commutative**, then a*b = b*a for *any* elements a,b of the group. This property is an optional one. If it is valid, then we have a **commutative group**, otherwise the group is a **non-commutative group**. In honor of Norwegian mathematician Niels Henrik Abel (1802-1829), a commutative group is also called an **abelian group**, and the other a **non-abelian group**.

The number of elements n in a group is called its **order**. We might denote the above group as

{G, *} order n

The notation here is **{ set, list of operations }**. Sometimes we will use **{a,b,c...}** to represent a set of the elements listed, a very traditional notation.

In defining fields and rings below, we shall make use of operations called **multiplication** and **addition**, with symbols • and +. The reader is warned that these operations are in general *not* what one is used to for say the integers or real numbers. For addition, the identity element is called "0", with g + 0 = 0 + g = g, and the inverse is called -g, with g + (-g) = (-g) + g = 0. The above little table then becomes these two separate tables:

Additive Group (1.1a)

Closed under	Associative	Identity exists	Inverse exists	Commutative
+	(a+b)+c=a+(b+c)	"0": 0+g = g+0 = g	"-g": g+(-g) = (-g)+g = 1	a+b = b+a

Multiplicative Group (1.1b)

Closed under	Associative	Identity exists	Inverse exists	Commutative
•	(a•b)•c=a•(b•c)	"1": 1•g = g•1 = g	"g^{-1}": g•g^{-1} = g^{-1}•g = 1	a•b = b•a ?

The idea is that the additive group is some generalization of the additive group of real numbers or of integers or of matrices in linear algebra. In these three cases "addition" is commutative, so by definition the general additive group is specified as commutative, no question mark on the right. Thus, any additive group is abelian.

Similarly, the multiplicative group is a generalization of the multiplicative group of real numbers or matrices. Although multiplication of real numbers is commutative, the multiplication of matrices is well known not to be so. For this kind of group the question mark remains. As we shall see later, the set of square matrices forms a group only if the matrices have non-zero determinant, which assures that every matrix has an inverse. Similarly, the integers do not form a group because inverses are not included in the set.

Example for • : Consider $g_\theta = R_z(\theta)$, the 3x3 matrix for rotations about the z axis. The set of all such rotations forms an abelian group with an infinite order: θ can be any real number from 0 to 2π, and $R_z(\theta_1)R_z(\theta_2) = R_z(\theta_2)\,R_z(\theta_1)$ so that $g_{\theta 1} \bullet g_{\theta 2} = g_{\theta 2} \bullet g_{\theta 1}$. Associative is pretty obvious, the inverse is $g^{-1} = R_z(-\theta)$ and identity is $1 = R_z(0)$ = 3x3 identity matrix. In contrast, the set of *arbitrary* 3x3 rotation matrices forms a non-abelian group. For example, $R_x(\theta_1)R_z(\theta_2) \neq R_z(\theta_2)\,R_x(\theta_1)$. In these examples, the abstract group elements like g_θ are "represented" by 3x3 matrices, and the operation • is represented by the usual multiplication of 3x3 matrices. All rotation matrices have determinant +1.

Another Example: In hadron physics, which is the study of strongly interacting particles, the Lagrangian density from which the equations of motion are derived has an internal symmetry group which contains pieces reminiscent of rotations in physical space. Like the group of rotations, this internal symmetry group is not commutative. The symmetry is an extension of a symmetry that arises in electromagnetic theory known as gauge invariance. The resulting class of hadron theories are known as non-abelian gauge theories.

A **field** is a heavier duty algebraic entity since it involves *both* operations • and + at the same time. A field is a set of elements with the *two* operations + and • such that the following conditions are valid for all a,b,c,g... in the field:

Closed under	Associative	Identity exists	Inverse exists such that:	Commutative
+	(a+b)+c=a+(b+c)	"0": 0+g = g+0 = g	"-g": g+(-g) = (-g)+g = 1	a+b = b+a
•	(a•b) •c=a• (b•c)	"1": 1•g = g•1 = g	"g^{-1}": $g \bullet g^{-1} = g^{-1} \bullet g = 1$	a•b = b•a
and:	a• (b+c) = a•b + a•c (distributive property)			

FIELD (1.4)

A field must be commutative under both the + and • operations, so the • question mark is now gone in the right column. Notice the newly appearing **distributive property** which involves both operations • and +. The • inverse of the element "0" does not need to exist. Somehow we would have to have some element ∞ such that 0 • ∞ = 1. For a finite field of order n, we might knock out the 0 element when thinking about •, and write these two groups within the field.

{ F, +} order n {F - 0, •} order n-1

where F - 0 means the that 0 has been deleted from the set F.

A **ring** is a field which has two of the • properties missing, and the • commutative property is optional :

Closed under	Associative	Identity exists	Inverse exists such that:	Commutative
+	(a+b)+c=a+(b+c)	"0": 0+g = g+0 = g	"-g": g+(-g) = (-g)+g = 1	a+b = b+a
•	(a•b) •c=a• (b•c)	["1" may exist]	[g^{-1} may exist]	a•b = b•a ?
and:	a• (b+c) = a•b + a•c (distributive property)			

RING (1.5)

The elements of either a field or a ring form an abelian group under operation +.
The elements of a field (less the 0 element) form an abelian group under •.
The elements of a ring in general don't even form a group under • .
The elements of a ring can form an **abelian ring**, or a **non-abelian ring** (the question mark).
Every field is also a ring. A ring is therefore a less strict entity than a field.

Reminders:

Rings and fields have two operations + and •. A group has only one operation.

In general, a group may or may not be commutative with respect to its operation. Similarly, a ring may or may not be commutative with respect to •, but it is always commutative with respect to +. A field requires that both operations be commutative.

Another word for commutative is *abelian*.

Infinite Fields: The are two *infinite* fields we are extremely familiar with: the real numbers and the complex numbers. The integers are only a ring because, for example, the inverse of 5 is 1/5 which is not an integer. For these fields, + and • are what we are used to.

Galois Fields: There are many *finite* fields as well. For any prime number p (p = 2,3,5,7,11,...), and for any integer m (m = 1,2,3....) , there exists a finite field which contains p^m elements. These finite fields are given the name GF(p^m), where GF = Galois Field in honor of the frustrated Frenchman Mr. Évariste Galois who, after a fiery life, died in a duel in 1832 at age 20. Legend claims he wrote down much of what he knew about math the night before. Apparently, his math was better than his shooting. Galois also invented the term "group" as we define it above, but only in a specific case. See wiki.

These GF(p^m) are the *only* finite fields there are. Any finite field you come up with is equivalent to one of the Galois Fields (a theorem we are not proving.) In general, the meaning of operations + and • for the elements of Galois fields is *not* what we are used to. Much more on this subject in later chapters.

The Binary World: We shall generally be interested in Galois Fields in which p = 2, so this means GF(2^m). In the special case m=1, we get GF(2). This field has only two elements, so they must be 0 and 1 (see properties list above). Here are the + and • tables for GF(2):

+	0	1
0	0	1
1	1	0

•	0	1
0	0	0
1	0	1

(1.6)

The field GF(2) is the lonely world of a binary digit -- a bit. Notice that addition is XOR, while multiplication is the normal thing. The field GF(2) is the underpinning of the digital era.

The Ring of Integers Z. Looking at the above ring definition, we find that for the usual + and • operations on integers, the set of all integers (plus, minus and 0) does in fact form a ring of infinite order. For example, any integer n has an additive inverse -n. This set Z fails to be a field because the only non-zero integer having a multiplicative inverse is 1. As noted, the multiplicative inverse of 5 being 1/5 is not an integer. In a field, every element (except 0) must have a multiplicative inverse that lies in the field. The full set of real numbers and the subset of rational numbers are genuine fields as well as rings.

(b) Groups and Subgroups

1. Subgroups, cosets, coset leaders, coset decomposition, N/k = m, normal subgroups

Consider a group G with N elements and some operation *. Later on, we may identify * with either + or •, depending on our context. In the + case, the term "product of group elements" a*b of course means the sum of group elements a+b.

A **subgroup** H is a group H within a group G. Obviously, H has to contain the identity 1 if it is really a group. [Again, if * = +, identity is called 0, since a+0 = a.] And if H really is a group, it must be closed under *, so for any h_1 and h_2 in H, the products h_1* h_2 and h_2* h_1 must lie in H. This idea is compactly expressed as h*H = H*h = H where H is the set of elements in H, and h is some particular element.

Let us assume that the order of G is n, and the order of H is k. It turns out that n/k = m, an integer. That is to say, the order of any subgroup divides evenly into the order of the group. One explanation (not a proof) of this claim is the following construct. You can arrange the group elements into a little **chart** where the top row contains the subgroup elements h_i. Then you randomly pick some other group elements g_i and put them into the leftmost column under 1, and you then form a multiplication table like so:

h_1=1	h_2	h_3	h_4	...	h_k	// coset #1	{1}
g_1	g_1*h_2	g_1*h_3	g_1*h_4	...	g_1*h_k	// coset #2	{g_1}
g_2	g_2*h_2	g_2*h_3	g_2*h_4	...	g_2*h_k	// etc	{g_2}
more rows like the above							(1.7)

If you pick the g_i for the left column clumsily, you end up with some repeat rows. It is possible (a claim) to pick the g_i so that all rows are different. In this case, it turns out (but we are not proving it here) that every group element appears exactly once somewhere in the chart. Since the chart has n elements, and a row has k elements, n/k must be an integer m, the number of rows. The rows of the chart are called **cosets**, and the first item in each row is called the **coset leader** for that coset. The H cosets *partition* the group G in what is called the **coset decomposition**. Therefore we have "Lagrange's Theorem" :

Fact: (Lagrange) The order of any subgroup divides evenly into the order of the parent group. (1.8)

In fact what we show above is a "left coset decomposition". If the group is not commutative, one could also form a "right coset decomposition" by putting the g_i elements in the g*h products on the right, and the elements would be arranged differently in the rows. If it happens that g*h = g*h for all g in G and all h in H, then the two coset decompositions are the same. In this case, the subgroup is called a **normal subgroup**, or an **invariant subgroup**. The fact that g*h = g*h can be written g^{-1}*H*g = H. If the group G is abelian (commutative), then every subgroup is an invariant subgroup since then g*h = g*h .

2. The Factor Group G/H formed from a group G and a normal subgroup H

Suppose we have a group G (order n) and an invariant subgroup H (order k) as shown above, and the group elements are put into a chart by the coset decomposition above. We know that n/k = m, an integer.

It is possible now to define a new group which has m elements. These elements are the **rows of the chart**! Although each row contains more than one element of group G, we think of the entire row as one element of the new group we are talking about. A standard notation is to label each row by some representative element, like the coset leader, and put this in **curly brackets**. Thus, the top row of the chart is our first group element which is {1}, and the next row is $\{g_1\}$ and so on. Recall that earlier we used {a,b,c...} to represent a set of elements. One can think of $\{g_1\}$ as a shorthand for $\{g_1, g_1*h_2, g_1*h_3 \ \ldots\ \}$ which is indeed the set of elements on a row of the chart. Normally one does not think of a set {...} as being a group element, but here that is exactly the case.

Now we have to define what it means to "multiply" two rows of the chart. We write

$\{g_1\} * \{g_2\} = \{g_3\}$, where $g_3 = g_1*g_2$.

What exactly does this mean? Each row has k elements. Suppose one creates all the products possible by multiplying some element of row 1 by some element of row 2. There are k^2 of these products. We claim (without proof) that all of these products will lie in the same row of the chart, and that row will be $\{g_3\}$ where $g_3 = g_1*g_2$. Obviously, many of these products must be the same, since row 3 only contains k elements.

The general idea is that one can represent a row by any of its elements. All the elements of a row have something in common. Notice that the operation * in our new group, whose elements are the rows, is defined *in terms of* what * does to elements of the underlying group G.

Do the m rows $\{g_i\}$ really form a group with m elements? The set of rows is closed under * as shown above since G is closed under *. The other group properties of G are similarly induced into our new group. For example, the inverse of $\{g_1\}$ will be $\{g_1^{-1}\}$, since $\{g_1\} * \{g_1^{-1}\} = \{ g_1g_1^{-1}\} = \{1\}$. The inverse of the row {1} is itself, which corresponds to H being closed under * (it is, after all, a subgroup). So yes, it is easy to show that G/H is in fact a group with the * operation as shown above.

This new group, whose elements are the rows or cosets of the group G with respect to the normal subgroup H, has a special name and notation. It is called the **factor group** of G with respect to H, and the notation for this new group is **G/H**. This is just a notation, we are not trying to divide group G by group H.

factor group = G/H . // has m = (n/k) elements, the rows (cosets) of the chart.

It is certainly not obvious why anybody in their right mind would have any interest in such a curiosity as this "factor group". Although we won't be applying this construct, we will apply its sister construct called the residue class ring coming up in section (c) below. One never knows what might be useful.

3. Cyclic Groups and Cyclic Subgroups

If g is in an element of group G then so is $g*g = g^2$, and $(g*g)*g = g^3$, and so on (because a group is closed under *). If the group has a finite number of elements n, we can write this list of n elements as $\{g^0, g^1, g^2 \ldots g^{n-1}\}$ where of course $g^0 = 1$ and $g^1 = g$. If there exists a g in G such that this list of n elements are all distinct, which is to say the list exhausts all elements of G, then G is said to be a **cyclic**

group. If the n elements are *not* all distinct, then perhaps we find that $g^5 = g^2$. We can write this as $g^3 g^2 = g^2$ which then means that $g^3 = 1$ (identity is unique, or, inverse g^{-2} exists). Thus, before we discovered that $g^5 = g^2$, we would have discovered that $g^4 = g^1$ and $g^3 = 1$. In general, if the n elements are *not* all distinct, we will get $g^k = 1$ for some integer k that is less than n. The smallest k for which $g^k = 1$ is called the **order of g** because g generates a cyclic subgroup of G which has order k. That is, the number of distinct elements in the subgroup $\{ g^0, g^1, g^2 ... g^{k-1}\}$ is k. This subgroup is a group because all the group properties are satisfied, notably closure. If it turns out that if k = n, the order of G, then the entire group G is cyclic and that g is a generator. If k = 1, then the cyclic subgroup is just the set {1}. According to Lagrange's Theorem (1.8), we know that k must divide n, so we can write n = Nk. Then since there is some k such that $g^k = 1$ for every g in the group, we know that $(g^k)^N = 1$ and thus $g^n = 1$ for every g in the group. We have now proven the following claims:

Facts: For finite groups:

(a) Any group element g of any group G must be a member of *some* cyclic subgroup of G.
(b) That cyclic subgroup could be the entire group, just the identity, or something in between.
(c) For any g in group G there is some minimum exponent k such that $g^k = 1$. We refer to this exponent as **the order of g**, since it is the order of the cyclic subgroup generated by g.
(d) the order k of the cyclic group generated by g must divide the order of G.
(e) for any g in G, $g^n = 1$ where n is the order of G. (1.9)

To summarize, a cyclic group contains at least one element g such that the powers of g exhaust the group:

$\{ 1, g, g^2, g^3, ...g^{n-1}\}$ = cyclic group, order = n elements, $g^n = 1$ (1.10)

Every element of the group can be written as a power of g, for power = 0,1,2....n-1.

An element g which allows a cyclic group to be fully enumerated as above is called a **generator**. In general, not every group element can serve as a generator, certainly not the identity. There may exist several **alternative generators** of the same cyclic group.

The **Greatest Common Divisor** of two integers i and j, written GCD(i,j), is the largest integer that divides evenly into both numbers. For example GCD(3,7) = 1, GCD(3,6) = 3.

An integer is **prime** (or "a prime", or "a prime number") if its only integer divisors are itself and 1.

Two integers i and j are **relatively prime** (**coprime**) when GCD(i,j) = 1, an example being GCD(3,8) = 1 (three is prime, eight is not prime, 3 and 8 are relatively prime). If p is a prime and q is any other integer not a multiple of p, then GCD(p,q) = 1 since p has no divisors other than p and 1. In this case, p and q are relatively prime.

Fact: If g is a generator of a cyclic group G of order n, then $g_1 \equiv g^i$ $(1 \le i \le n)$ is the generator of a cyclic subgroup of order j = n/GCD(n,i). (1.11)

Note the validity of the two extreme cases: if i = 1, then j = n which is correct since g is a generator. And if i = n, then j = 1 which is again correct since $g^n = 1$.

Proof: The above Fact will be proven as (4.21), where cyclic subgroups of GF(q) are explored in great detail. For now, we state some Corollaries to this Fact, and then prove one of them.

Corollary 1: If n is prime and i < n, then in (1.11) j = n/GCD(n,i) = n. In this case, our Fact above says that if g is a generator of a cyclic group, then the group element $g_1 \equiv g^i$ (1 < i < n) is an alternative generator. Even if n is not prime, such a g^i is an alternative generator if GCD(n,i) = 1, which is to say, if n and i are "relatively prime". (1.12a)

Corollary 2: All elements (except the identity element) of a cyclic group G of prime order n are alternative generators. This is a direct result of Corollary 1. (1.12b)

Corollary 3: If n/i = k for i>1, then $g_1 = g^i$ is *not* an alternative generator to g. In this case, g_1 generates a smaller cyclic subgroup of order j = n/GCD(n,i) = n/i = k. (1.12c)

Proof of Corollary 3: This follows directly from (1.11), but here is a proof anyway. Let n/i = k, an integer. Since i>1, k < n. If one tries to list off the group elements using g^i as a generator, one gets $\{1, g^i, g^{2i}, g^{3i} \ldots g^{(k-1)i}\}$. The next element in the list would be g^{ki}, but $g^{ki} = g^n = 1$ (since g is a generator of cyclic group G). The next element would then be $g^{(k+1)i} = g^i$, so we are just repeating elements. We know that the cyclic group G has n elements, but we have only been able to enumerate k of them, and k < n.

Example: Let n = 12, i = 3, and k = n/i = 4. Then here is our partial enumeration:

$\{ 1, g^3, g^6, g^9 \}$ $g^{12} = 1$ // only get 4 elements out of the 12.

Fact: A cyclic group is commutative (abelian). (1.13)

Proof: Let g_1 and g_2 be any elements of the group. If g is a generator, then $g_1 = g^a$ and $g_2 = g^b$. It then follows that $g_1 g_2 = g^a g^b = g^{a+b} = g^b g^a = g_2 g_1$. Thus any pair of elements commutes.

4. Additive Groups and Additive Cyclic Groups : the Vector Space of group elements

Additive Groups

By "additive group" we mean a group whose operation * is +. By definition (1.1a), such a group is abelian. For the + operation, the term "powers of g" means "multiples of g", since for example $\mathbf{g}^3 = \mathbf{g}*\mathbf{g}*\mathbf{g} = \mathbf{g}+\mathbf{g}+\mathbf{g} = 3\mathbf{g}$. The notation "3**g**" means just the sum shown. The thing "3" is not in the group. The 3 is a scalar multiple of the group element **g**. It happens that the scalar in question, 3, is a element of the ring of integers. For the moment, we display group elements in bold font just to distinguish them from the integer multipliers as in 3**g**.

Let's now pick some **g** and try to enumerate the group with it :

$\{ 0\mathbf{g} = 0, 1\mathbf{g} = \mathbf{g}, 2\mathbf{g}, 3\mathbf{g}, 4\mathbf{g}, \ldots.\}$

If our additive group G has finite order n, then this list cannot continue forever to generate new elements of G. As explained at the start of section 3 above (* notation), there will be some $k \leq n$ such that $k\mathbf{g} = \mathbf{0}$. After that, we have $(k+1)\mathbf{g} = \mathbf{g}$, $(k+2)\mathbf{g} = 2\mathbf{g}$, and so on, and elements repeat as k gets larger and larger. In this case, since $k\mathbf{g} = 0\mathbf{g} = \mathbf{0}$, there is only a need for integers 0,1,2,...k-1. The largest k can be is n, so one might require integers 0,1,2...n-1 at most, since $n\mathbf{g} = 0$ in that case. As we shall see below, these integers are elements of the ring $Z_n = \{\text{mod-n}, +, \bullet\}$.

We would like to claim that we can think of the elements of our additive group as vectors in a vector space, and that gives us even more motivation to write them in bold font, just as we write $\mathbf{r} = (x,y,z)$ for vectors in Euclidean space R^3.

If one looks at the definition of a **vector space over field F**, there are two sets or required properties. The first set says that the elements (**vectors**) of the vector space must form an abelian additive group under +. These properties are associative, commutative, and the existence of identity and inverses. Our additive group obviously meets all these requirements. The second set of requirements deals with both elements of the vector space G and elements (**scalars**) of the underlying field F :

$\alpha\mathbf{a}$ is defined	one must clarify what it means to multiply α in F by **a** in G
$\alpha\mathbf{a} = \mathbf{a}$ for $\alpha = 1$	the multiplication rule above must work this way for $\alpha = 1$
$\alpha(\mathbf{a}+\mathbf{b}) = \alpha\mathbf{a} + \alpha\mathbf{b}$	distributive but α is in F while **a** and **b** are in G
$(\alpha+\beta)\mathbf{a} = \alpha\mathbf{a} + \beta\mathbf{a}$	distributive but α and β are in F while **a** is in G
$\alpha(\beta\mathbf{a}) = (\alpha\beta)\mathbf{a}$	associative but α and β are in F while **a** is in G

These requirements are also met. For example, $\alpha\mathbf{a}$ means $\mathbf{a} + \mathbf{a} + ... + \mathbf{a}$ α times. Then $1\mathbf{g} = \mathbf{g}$ is pretty clear. Then for example $2(\mathbf{a}+\mathbf{b}) = (\mathbf{a}+\mathbf{b}) + (\mathbf{a}+\mathbf{b}) = 2\mathbf{a} + 2\mathbf{b}$, which is the third property. This can be generalized for 2→any α. Similarly the other requirements are satisfied.

But there is one *more* requirement, and that is that F be a field. The elements of our additive group G are defined over Z_n which happens to be a ring, not a field. As we shall see, this is because if α lies in Z_n, α^{-1} (the multiplicative inverse of α) might not exist. Nevertheless, one is allowed to have a vector space over a ring, and such a creature is formally called a **module**, but we shall just think of it as a vector space over a ring. Thus, we can regard our additive group elements as vectors in a vector space over the ring Z_n, or for that matter, over the larger ring Z (all integers) which contains Z_n.

We shall often be interested in additive groups G whose order n is a prime number. When n is prime, then, as we shall show below, the ring Z_n elevates itself to field status, and then we can truly say that the elements of G are vectors in a vector space over the field Z_n. In this case, we cannot say G is also a vector space over field Z since Z is not a field.

Fact : In general, the elements of an additive group of order n form a vector space over the rings Z_n or Z. In the case that n is prime, these elements form a vector space over the field Z_n. (1.14)

Galois Field elements as vectors in a vector space.

Below we shall be studying the Galois Fields GF(p) and GF($q=p^m$) where p is prime. Note that GF(p) is a special case of GF(q). The elements of GF(p) and GF(q) form additive groups of order p and q. This follows from the fact that they are both fields. We shall find in (4.5) that $p\mathbf{g} = 0$ for **g** in either GF(p) or GF(q), so in either case we can "make do" with the integers in the ring Z_p. According to the Fact above, since p is prime, the elements of both GF(p) and GF(q) are vectors in a vector space over the field Z_p. So,

Fact: The elements of GF(p) and GF(q) form a vector space over the field Z_p . (1.14a)

Corollary: The elements of GF(q) form a vector space over the field GF(p). (1.14b)

Proof: This is a combination of the (1.14a) and the Super Big Fact (2.5a) to be shown below that GF(p) = Z_p , meaning the two fields are isomorphic.

Comment: Except in Chapter 10, we don't use a special symbol like $\doteq$ to indicate isomorphism, we just say the two fields are "the same" with an = sign. **Isomorphism** means there is a clean one-to-one correspondence between the elements and all properties of the two isomorphic sets.

Additive Cyclic Groups

If there exists some $\mathbf{g} \neq \mathbf{0}$ such that the smallest integer k for which k**g** = **0** is k = n, then our additive group G is an additive cyclic group, according to the cyclic definition of section 3 above.

Recall that, for any cyclic group, $g^n = 1$, where 1 is the identity of the group, n is the order of the cyclic group, and g is a generator. For an additive cyclic group, this statement becomes n**g** = **0,** since the nth power of g is n**g**, and since **0** is the identity for + .

Thus, we can enumerate the elements of an additive cyclic group as follows (if **g** is a generator)

{ **0**, **g**, 2**g**, 3**g**, ...(n-1)**g**} = cyclic group, order = n elements, n**g** = **0** . (1.15)

Fact: If the order n of an additive cyclic group G is a *prime number*, then any element (other than 0) can serve as a generator. In this case, n**g** = 0 for any element of G. (1.16)

Proof: This is just the statement of Corollary 2 (1.12b).

Example: If n is prime, we could take **h** = 3**g** and enumerate the above as group as

{ **0**, **h**, 2**h**, 3**h**, ...(n-1)**h**} = cyclic group, order = n elements, n**h**= **0**

Fact: If the order n of an additive cyclic group is a *prime number*, then -(i**g**) = (n-i)**g** . (1.17)

Proof: The object -(i**g**) means the additive inverse of element i**g**. Since

(n-i)**g** +ig = n**g** - i**g** + i**g** = n**g** = 0 // since n prime

we conclude that (n-i)**g** must be the inverse of i**g**.

Summary of the properties of an additive cyclic group of finite order n: (1.18)

1) the identity element is **0**
2) every element **g** must have an inverse **(-g)** such that **g** + **(-g)** = **0**.

3) the group can be enumerated as { **0**, **g**, 2**g**, 3**g**, ...(n-1)**g** } for at least one **g** // since cyclic
4) such a **g** is by definition a generator
5) n**g** = **0** for this generator **g** // (1.15)
6) if n is prime, then all non-zero group elements are generators // (1.16)
7) if n is prime, then n**g** = **0** for any **g** in the group. // (1.15)
8) if n is prime, then n is the smallest positive integer for which n**g** = 0 for any **g**
9) if n is prime, then for any **g** we can write -(i**g**) = (n-i)**g** // (1.17)

Suppose n = 7. One could perversely enumerate such an additive cyclic group as shown on the left below, instead of as shown on the right :

{ -3**g**, -2**g**, -**g**, **0**, **g**, 2**g**, 3**g** } instead of { **0**, **g**, 2**g**, 3**g**, 4**g,** 5**g,** 6**g** } .

We know from item 9 above that - 3**g** = 4**g**, -2**g** = 5**g**, -**g** = 6**g**.

In section (c) 5 below we shall consider a ring whose order n is infinite. In this case, the idea of (1.17) doesn't work since n = ∞. Then if we enumerate the group as { **0**, **g**, 2**g**, 3**g**, 4**g,** 5**g,** 6**g** ....} , the inverse of 3**g** is not included! In this case, we must enumerate elements in this manner

{ ... -3**g**, -2**g**, -**g**, **0**, **g**, 2**g**, 3**g**, ... } (1.19)

where **g** is a generator of the infinite additive group.

Are Galois Fields cyclic under • or + ?

We shall find in (4.30) that both {GF(p)-0} and (GF(q)-0} are cyclic under • .

What about under + ? Are GF(p) and GF(q) additive cyclic groups?

We shall see in (2.5) that GF(p) is isomorphic to Z_p, the field of modulo p integers. Since 1 is a viable additive generator for Z_p, that is also true for GF(p). Thus, GF(p) is an additive cyclic group. In fact, since p is prime, any non-zero element g of GF(p) is an additive generator by (1.16), and p**g**=0.

For GF(q=p^m) with m > 1 the story is different. We will find in (4.5) that p**g** = 0 for any element **g** of GF(q). Since q = p^m > p when m>1, we know that GF(q) for m>1 is *not* an additive cyclic group. We can try to additively enumerate all elements of GF(q) starting with any **g**, but the farthest we will get is p elements.

5. Example of an additive cyclic group: {Mod-n,+}

The "mod-n additive group" has as its elements {**0**,**1**,**2**,**3**,..**n-1**}, and the + operation is mod-n addition, which is to say,

a + **b** = **Rem**[(a+b)/n] = Rem[(a+b)/n] **1** . (1.20)

Example: Suppose n = 4. Then **2** + **3** = **Rem**[(2+3)/4] = **Rem[**5/4] = **1** = Rem[5/4] **1** = 1 **1** = **1**. This is certainly a ponderous exercise in using a bold font to represent group elements.

Why is {Mod-n,+} an additive cyclic group? It is certainly closed under addition. Element **1** (not the identity) is certainly a generator which enumerates all the elements of the group, so the group is cyclic.

Here is a list of observations about the additive cyclic group mod-n:

1) there is no element **n**.
2) the identity element is **0**.
3) the group is cyclic with **1** as a generator, and n**1** = **0**
4) all non-zero elements can be written as multiples of the generator, **m** = m**1**.
5) the inverse of element **m** must be **(-m)** ≡ (n-m)**1. (**Proof: **m** + **(-m)** = m**1** + (n-m)**1** = n**1** = **0**.)
6) if n is prime, then any element **m** (other than **0**) is a generator, and n**m** = **0**.
7) mod-n forms a "principle ideal" within the integers (see below) (1.21)

These facts come from (1.18) which applies to any additive cyclic group.

(c) Rings and Ideals

1. Ideals, residue classes, residue class leaders, residue class decomposition, N/n = m

We are now going to repeat the above song and dance almost verbatim, but this time for a *ring* instead of a *group*. A ring has two operations • and +, whereas a group has only one operation, so things will be just a little different. We started the above harangue by imagining that group G had a subgroup H. Here we might suppose that a ring R has a subring, but this turns out to be not the right thing; the correct sub-thing is called an *ideal*, and one uses the letter I.

So what is an **ideal** I of *ring* R? First of all, with respect to the + operation, I must be a subgroup of R. Thus, I must contain the additive identity which we call 0. Secondly, with respect to the • operation, I must have this property: r•I = I•r = I. This means that if you pick any r in R, and any i in I, the product r•i lies somewhere in I, and so does the other product i•r. In particular, we have i•I = I, so I is closed under • as well as +. So an ideal is in fact a subring of R, but it is more specific since r•I = I even for r not in I. (1.22)

So let R have n elements, and assume there is an ideal I with k elements. As before, we are going to build a chart, and we will then claim that n/k = m = an integer. We lay down I itself as the top row. Since we are really thinking about the + operation now, the top left element is the + identity 0 (the thing you put in r + 0 = r). We next start picking random elements of R and plop them down in the left column, then we form rows by doing sums of the left element with the i's along the top. Here is the chart for ring R with respect to ideal I:

i_1=0	i_2	i_3	i_4	...	i_k	// residue class #1
r_1	r_1+ i_2	r_1+ i_3	r_1+ i_4	...	r_1+ i_k	// residue class #2
r_2	r_2+ i_2	r_2+ i_3	r_2+ i_4	...	r_2+ i_k	// etc
more rows like the above						(1.23)

Again, we make the claim that if you pick the left elements right and get things so that you throw out any repeating rows, you end up with a coset decomposition that partitions the ring. Each ring element appears exactly one place in the chart. Thus, as before, we conclude that n/k = m, and integer.

Because mathematicians cannot leave well enough alone, they decided that they had to invent a new name for everything since we are doing rings instead of groups. Thus, what was called a coset before is now called a **residue class**. A residue class is a row of the chart. And the first item in a row is now the **residue class leader**. Big deal. Note that, although a ring has • and + operations, it is only the + operation we are talking about in the above chart. The ring is a group under +, so this chart really is analogous to our early group chart.

2. The Residue Class Ring R/I formed from a ring R and an ideal I

Before, our next step was to claim that you could make a fancy new group by considering each row of the chart to be an element of that new group. This was the factor group of G with respect to H. Here we are going to do the same thing, but as you might expect, the new "thing" which has the rows as elements is going to be a ring, not a group, since R is a ring. Thus, admittedly, we cannot call this thing a factor group. So they make a new name. This new ring with m elements being the rows of the above chart is called the **residue class ring**, or the **quotient ring** To be consistent, they should have called the other thing a coset group, but they called it a factor group instead. The notation is pretty much the same:

residue class ring = R/I has m = (n/k) elements, the rows (residue classes) of the chart. (1.24a)

We now have to say what it means to do the + *and* • operations on "rows" of the chart. We have two operations to worry about instead of one :

$$\{r_1\} \bullet \{r_2\} = \{r_3\}, \text{ where } r_3 = r_1 \bullet r_2$$
$$\{r_1\} + \{r_2\} = \{r_4\}, \text{ where } r_4 = r_1 + r_2 \qquad (1.24b)$$

Again, $\{r_2\}$ is an element of the new "residue class ring", and r_2 is some representative element of the row of the chart by which this residue class ring element is labeled. We need then to make exactly the same interpretation as before for what the above lines mean, only here we say the same thing for each operator + and •.

For example, if you form all k^2 products between elements of two rows, the results all lie in the same row of k elements. And the same applies if you make all k^2 possible sums. Of course the results row will likely not be the same row for • and +, so we have used r_3 and r_4 above. We have not proven that the residue class ring is a ring, we are just claiming that it is.

You may remember that a ring does not in general have a 1 element for •, but it always has a 0 element. Thus, every row $\{r_1\}$ must have some corresponding "negative" row $\{r_2\}$ such that

$$\{r_1\} + \{r_2\} = \{0\} .$$

And as before, since our ideal I is a subgroup of R relative to +, we know that the negative of the top row is itself (the subgroup I is closed so $\{-i_r\}$ and $\{i_r\}$ are in the first row since $-i_r$ and i_r are in I). This is expressed by the incredibly boring statement: $\{0\} + \{0\} = \{0\}$. Since we might not have a $\{1\}$ row, we have nothing to say about $\{1\}$. (yet) A quotient ring example is coming very soon, please hold on.

3. Principle Ideals and Principle Ideal Rings

Now we are ready to pursue the ring analog of the cyclic subgroup discussion above.

If **i** is in an element of ring R, then so is any multiple of **i**, such as 3**i**. If the ring has a finite number n of elements, you must, as you keep adding more terms, come to a point where k**i** = **0** for some k ≤ n. After this point, if you keep adding **i**, things just repeat, for example, (k+1)**i**= **i**.

This set of multiples of some ring element **i** together with **0** forms an additive cyclic group of order k, as discussed in much detail above. We have,

{ 0, **i**, 2**i**, 3**i**, ...(k-1)**i** } = additive cyclic group of order k (1.25)

If you find that you have exhausted all the elements of the ring in this way, then it must be that k = n, the order of the ring, and **i** is a generator. The other possibility is that you reach the point k**i**= 0 *before* all ring elements have been hit. In this case, you have found an additive cyclic subgroup (order k) of the ring.

In the case of rings, we are more interested in ideals than we are in subgroups. The above cyclic subgroup of order k may or may not be an ideal of R. Recall (1.22) that for an ideal, it must be true that r•I = I•r = I for elements r that are outside the ideal as well as inside. The above enumeration and the fact that a set forms an additive cyclic subgroup within R does not guarantee this extra required property to make that subgroup be an ideal.

If the subgroup is not an ideal, then we have nothing more to say. But if the subgroup *is* an ideal, then it is called a **principle ideal**. Thus, a principle ideal is an ideal whose elements can be fully enumerated as sums of some generator **i**. Alternatively, a principle ideal is an ideal which is an additive cyclic subgroup of R. (1.26)

If every ideal in a ring is a principle ideal, then the ring as a whole is called a **principle ideal ring.**

We can now go back and consider the case we quietly skipped. If you have exhausted *all* ring elements by taking multiples of some r, then the set of all these multiples is an ideal of the ring. Every ring is an ideal of itself, just look at the definition (1.19) of an ideal. Thus, in this case of exhaustion, you again end up with a principle ideal ring.

Fact: The order of any principle ideal of a ring must divide evenly into the order of the ring. (1.27)

Proof: We already know from the residue class decomposition that the order of *any* ideal I of a ring R must divide evenly into the order of R, so saying this for a principle ideal is nothing new.

4. Example of a ring: $Z_n \equiv$ {Mod-n,+,•} ; Modulo Arithmetic

Earlier we discussed the additive cyclic group {Mod-n,+} which consisted of n elements {**0**,**1**,**2**....**n-1**} with addition defined by **a** + **b** = Rem[(a+b)/n] **1**. The additive generator is **1**. Any element can be written in the form **m** = m**1**, a multiple of the additive generator.

In order to have a ring, we need a second operation •. It is defined in a manner similar to +, so here are both operations:

$$\mathbf{a} \bullet \mathbf{b} = \text{Rem}[ab/n]\ \mathbf{1} \qquad \mathbf{a} + \mathbf{b} = \text{Rem}[(a+b)/n]\ \mathbf{1} \tag{1.28}$$

Claim: {Mod-n,+,•} forms a "ring with identity". A standard notation for this ring is $Z_{\mathbf{n}}$, though some authors use the notation Z/nZ. (1.29)

Proof: Since this ring is so important, we will make a reasonable attempt to do a real proof:

1) The elements of Mod-n are clearly *closed* under this •, since ab/n always produces a remainder in the range (0,n-1).

2) Also, ab/n = ba/n so we have **a • b** = **b • a** and we have *commutative* (it's an abelian ring).

3) The *associativity* proof requires Little Lemma 2 below, then you get:

$$(a\bullet b)\bullet c = \text{Rem}[ab/n]\ \mathbf{1} \bullet \mathbf{c} = \text{Rem}\ \{\ \frac{\text{Rem}(ab/n)\ c}{n}\ \}\ \mathbf{1} = \text{Rem}\ \{\ (abc)/n\}\ \mathbf{1}\ .$$

Since this result is symmetric in a, b and c, it must be equal to any grouping (x•y)•z you want, in particular it is equal then to a•(b•c) so we have associative.

4) The *distributive* property looks like this:

$$\mathbf{a}\bullet(\mathbf{b}+\mathbf{c}) = \text{Rem}[a(b+c)/n]\ 1 = \text{Rem}[\frac{ab + ac}{n}\ ]\ \mathbf{1}$$

$$\mathbf{a}\bullet\mathbf{b} + \mathbf{a}\bullet\mathbf{c} = \text{Rem}(ab/n)\mathbf{1} + \text{Rem}(ac/n)\ \mathbf{1} = \text{Rem}[\ \frac{\text{Rem}(ab/n) + \text{Rem}(ac/n)}{n}\ ]\ \mathbf{1}$$

and these are equal from Little Lemma 3 below.

5) The element **1** is an identity for •, since **1•m** = (1m)**1** = m **1** = **m .**

Thus, we have shown that {mod-n, + • } has all the properties of a ring, and it has a multiplicative identity as well, so it is a **ring with identity**. As noted, a common notation for {mod-n, + • } is $Z_{\mathbf{n}}$.

Consider the following "little lemmas":

Little Lemma 1: $\text{Rem}\{\frac{\text{Rem}(x/n)+y}{n}\} = \text{Rem}[\frac{x+y}{n}]$

Little Lemma 2: $\text{Rem}\{\frac{\text{Rem}(x/n)\,y}{n}\} = \text{Rem}[\frac{xy}{n}]$

Little Lemma 3: $\text{Rem}[\frac{\text{Rem}(x/n)+\text{Rem}(y/n)}{n}] = \text{Rem}[\frac{x+y}{n}]$

Little Lemma 4: $\text{Rem}[\frac{\text{Rem}(x/n)\text{Rem}(y/n)}{n}] = \text{Rem}[\frac{xy}{n}]$ (1.30)

As needed, the remainders can be rewritten using q/n = Q + Rem(q/n) so that

Rem(x/n) = x - Xn where X = integer
Rem(y/n) = y - Yn where Y = integer.

Then each little lemma can be proved in the same manner using Rem[(a + nb)/n] = Rem(a/n +b) = Rem(a/n). For example, in Little Lemma 4,

$$\text{Rem}[\frac{\text{Rem}(x/n)\text{Rem}(y/n)}{n}] = \text{Rem}[\frac{(x-Xn)(y-Yn)}{n}] = \text{Rem}[\frac{xy+n(-yX-xY+XYn)}{n}] = \text{Rem}[\frac{xy}{n}]$$

Modulo Arithmetic

Since x mod n = Rem(x/n), the previous Little Lemmas can be restated in order as,

$$(x+y) \bmod n = ([x \bmod n] + y) \bmod n$$
$$(x*y) \bmod n = ([x \bmod n] * y) \bmod n \quad (1.31a)$$

$$(x+y) \bmod n = ([x \bmod n] + [y \bmod n]) \bmod n$$
$$(x*y) \bmod n = ([x \bmod n] * [y \bmod n]) \bmod n \quad . \quad (1.31b)$$

These can easily be generalized to obtain,

$$(x+y+z+\ldots) \bmod n = ([x \bmod n] + [y \bmod n] + [z \bmod n] + \ldots) \bmod n$$
$$(x*y*z+\ldots) \bmod n = ([x \bmod n] * [y \bmod n] * [z \bmod n] + \ldots) \bmod n \quad (1.31c)$$

in which any [q mod n] on the right could be replaced by q. Examples:

$$(7 + 10 + 9) \bmod 6 = ([7 \bmod 6] + [10 \bmod 6] + [9 \bmod 6]) \bmod 6 = (1+4+3) \bmod 6 = 2$$
$$(7 * 10 * 9) \bmod 6 = ([7 \bmod 6] * [10 \bmod 6] * [9 \bmod 6]) \bmod 6 = (1*4*3) \bmod 6 = 0 .$$

As a shorthand, we sometimes omit the mod descriptors, keeping in mind the mod index. The last line then becomes

$$630 = 7*10*9 = 1*4*3 = 12 = 0$$

The equals signs in this last line are really "congruence" signs mod 6.

5. Example of a Residue Class Ring: Z/(n)

Suddenly all of the above obscure residue class business is going to start making sense. The classic example is to let R be Z, the ring of integers -- plain old integers, plus and minus and 0. The + and • operations are the regular operations we are used to.

Consider the subset of integers which are multiples of 5. We could pick any integer n, but 5 seems nice:

$$\{\ 0 \quad \pm 5 \quad \pm 10 \quad \pm 15 \quad \pm 20 \quad \ldots\ldots\ \} \tag{1.32}$$

This type of list appeared in (1.19) and in that notation we rewrite the members of this ring as

$$\{\ \ldots\ -3\boldsymbol{\alpha},\ -2\boldsymbol{\alpha},\ -\boldsymbol{\alpha},\ \mathbf{0},\ \boldsymbol{\alpha},\ 2\boldsymbol{\alpha},\ 3\boldsymbol{\alpha},\ \ldots..\ \} \qquad \boldsymbol{\alpha} = 5\mathbf{1} \tag{1.33}$$

which is to say

$$\{\ \ldots..\ -3(5\mathbf{1}),\ -\ 2(5\mathbf{1})\ ,\ -\ (5\mathbf{1})\ \mathbf{1},\ 0,\ (5\mathbf{1}),\ 2(5\mathbf{1}),\ \ 3(5\mathbf{1})\ \ldots..\}\ \ . \tag{1.34}$$

The generator is (5**1),** and this ring is formally an additive cyclic group since the generator enumerates all ring elements. The set is closed under either the + or • operations. You cannot, for example, generate a 14 by adding or multiplying two items in the above list. In our former notation, any element of this set can be written as $\mathbf{g} = n(5\mathbf{1}) = (5n)\mathbf{1}$, where n can be any integer, so the elements of the above set form a vector space over the ring (a module) of integers which are multiples of 5. In slightly less formal notation, we write the set as

$$\{\ \ldots..\ -3(5),\ -\ 2(5)\ ,\ -(5),\ \ 1,\ 0,\ (5),\ 2(5),\ \ 3(5)\ \ldots..\} \tag{1.35}$$

or

$$\{\ \ldots..\ -15,\ \ \ -\ 10\ ,\ \ \ -5,\ \ \ 1,\ 0,\ 5,\ \ \ 10,\ \ \ \ 15\ \ldots..\}\ \ . \tag{1.36}$$

The above list of elements forms an *ideal* within the full set Z of integers. Why? Because it is first of all an additive subgroup of Z, and second of all, multiplication of any element of this set by *any* integer gives an element in the set, whether or not that integer is in the set. Thus our additive subgroup fulfills the requirements (1.22) of being an ideal. Since this ideal is a subgroup of Z, it is a principle ideal, (1.26).

Fact: The only ideals one can make inside the set of integers are principle ideals like this, so R is a principle ideal ring, see text below (1.26). (no proof) (1.37)

The usual notation for this ideal is (5), where the parentheses are supposed to suggest all multiples of the thing inside, as in (1.35). Since the parent ring is Z (all integers), we are now going to form the residue class ring which has the notation Z/(5), or more generally, R/I = Z/(n). [Later in a different world we will have an ideal (f(x)) which consists of all polynomials which are a multiple of some f(x).]

In building the residue class chart, we are going to make a slight change in notation. Instead of putting the residue class leaders on the left edge of our chart as done above, we will put them as the middle column of the chart, and we will mark it with a ** so you can find this column. So here is the chart which is the

residue class (row) decomposition of ring R=Z with respect to ideal I = (5). The top row as usual is the ideal.

			**						
...	-10	-5	0	5	10	15	...	{0}	
...	-9	-4	1	6	11	16	...	{1}	
...	-8	-3	2	7	12	17	...	{2}	
...	-7	-2	3	8	13	18	...	{3}	
...	-6	-1	4	9	14	19	...	{4}	(1.38)

That's it! There are only 5 rows. Any other rows would be repeats. Notice that we have *partitioned* the integers into the rows -- each integer appears in exactly one place in this chart. The column on the far right shows the conventional manner in which each row is labeled, as discussed earlier. Each row is represented by the element in the column **.

Notice now an interesting way to think of the residue class leader labels 0,1,2,3,4 under the **. All the elements in a row have something in common. All integers in the second row have remainder 1 when you divide them by 5, and this 1 is the label of the leader. So the row label is the *remainder* of what you get by dividing any row element by 5.

The notion of **remainder** is the key point here. Each row corresponds to a possible remainder. The top row is the ideal which goes with remainder 0.

So the residue class ring thing has 5 elements -- the five rows of the above chart.

Let's find the "additive inverse row" of the second row {1}. It is the last row {4}. Add any pair of items one from each of these rows, and you get something in the first row {0}. Thus,

$$\{1\} + \{4\} = \{0\}$$

The first row is its own inverse, which brings us back to the enduringly exciting fact that {0} + {0} = {0}.

We wish now to explicitly write down the + and • tables for our residue class ring. Addition {i} + {j} means we do normal addition i + j, then decide what row the answer goes into. Clearly, it goes into the row determined by Rem[(i+j)/5]. And Multiplication {i}•{j} means we do normal ij multiplication, then divide by 5 to see what row the result is in. Thus:

$$\{i\} + \{j\} = \{k\} \quad \text{where } k = \text{Rem}[(i+j)/5] = (i+j) \bmod 5$$
$$\{i\} \bullet \{j\} = \{m\} \quad \text{where } m = \text{Rem}[(ij)/5] = (ij) \bmod 5 \qquad (1.39)$$

Fact: The residue class ring we have just constructed is equivalent to the mod-n ring discussed at length above. In other words, $Z/(5) = Z_5$. More generally, $Z/(n) = Z_n = \{ \text{mod-n}, +, \bullet \}$. (1.40)

Proof: Both rings have the same number of elements and the same + and • tables and there is a clear 1-to-1 correspondence between the elements, so Z/(n) and Z_n are isomorphic to each other which we write here as $Z/(n) = Z_n$.

6. Some basic facts about the integer ring Z

Most of these facts are so obvious that one hesitates even to write them down. However, each item carries over into the polynomial world we are about to enter, so we need a solid starting point.

• **Division Algorithm**. Relative to a divisor integer d, an integer n has a unique quotient integer q, and a unique remainder integer r:

n/d = q + r/d or n = q•d + r "Rem(n/d) = r" (1.41)

The second form is nicer since we can avoid talking about the / operation. This operation does not even exist in the ring of integers R, so we will steer clear of it.

Example: 32 = 6•5 + 2

Comment: This is sometimes called the Euclidean Division Algorithm. It is really a theorem, not an algorithm, but various algorithms are derived from it, including what is called the "Euclidian Algorithm" stated in the second item following.

• **Factorization Theorem**. Any integer can be uniquely decomposed into a product of factors where each factor is a prime number raised to an integer power:

$n = (p_1)^{m_1} (p_2)^{m_2} (p_3)^{m_3} \ldots$ (1.42)

Example: $451,008 = 2^6\ 3^5\ 29^1$

• **Euclidean Algorithm and Bezout's Identity**. The Euclidean Algorithm is a set of instructions one can use to find $d = GCD(n_1,n_2)$ for any integers n_1 and n_2. Bezout's Identity says that this common divisor d can be expressed in the following manner, where a and b are integers,

$d = a\ n_1 - b\ n_2$ { replace b with -b to get the usual form of this identity} (1.43)

You are given n_1 and n_2, and regardless of whether you have computed $d = GCD(n_1, n_2)$, you know that there exist two integers a and b (either could be negative) such that $d = a\ n_1 - b\ n_2$.

Example: 7 = 13•(973) - 42•(301) // here 7 = GCD(973,301)
a n_1 b n_2

Fact (1.43) is certainly less obvious than the previous ones. We verify the above example in Maple:

```
13*973 - 42*301;          gcd(973,301);
                7                         7
```

Recall that if the integers n_1 and n_2 have no common factors other than 1 [$GCD(n_1,n_2) = 1$], then n_1 and n_2 are relatively prime (coprime), see above (1.11). In this case, from (1.43) above one can be guaranteed

of the existence of integers a,b such that 1 = a n_1 - b n_2. This occurs, for example, if n_1 is a prime number and n_2 is any integer that is not a multiple of n_1.

7. The Residue Class Ring Z/(n) is a *field* if n is prime

We now close this chapter by coming back to the subject of fields. We have seen how one can take a ring R and an ideal I and how one can make a chart whose rows become elements of something called the residue class ring. We did an example Z/(5) which was seen to be equivalent to $Z_5 \equiv$ {Mod-5,+,•}. Here is a fundamental new result:

Big Fact: If n is a prime number, then the residue class *ring* Z/(n) above is also a *field*. Since we have shown already that Z/(n) = Z_n , this means that $Z_n \equiv$ {Mod-n,+,•} is a field when n is prime. (1.44)

Proof: The proof is really the same whether you do it for Z/(n), or for Z_n. In the former case, we add a few brackets to talk about residue class rows. In the latter case, we just talk about remainders, not rows. What we need to do is show that the two properties omitted in our ring definition above (compared to the field definition) are present in this case. Those properties are existence of an inverse, and existence of an identity, both relative to •. But we already know that the • identity exists since we showed that {Mod-n,+,•} was a "ring with identity". So all that remains is to show there is a • inverse for any element of Z_n when n is prime.

Consider an arbitrary row {r} described by remainder r. We need to show that there exists an inverse row {s} under the operation • such that {r}•{s} = 1.

Since n is prime, and since r < n, we know that n and r are relatively prime, and their greatest common divisor is 1. Using Bezout's Identity (1.43) above, we claim that integers N and M must exist such that 1 = Nr -Mn. Now take the remainder of both sides of this equation divided by n. This says that there exists an N such that 1 = Rem(Nr/n). Next, use (1.41) to expand N = Kn + s, where s < n. This gives 1 = Rem($\frac{Knr + sr}{n}$) = Rem(sr/n). Thus, according to the definition of the operation •, we have found s such that **r•s = 1.** [The Z_n proof just ended here.] These remainders of course are representative of rows, so this means {r}•{s} = {1}. **QED**

Thus, every row has a multiplicative inverse, and this was the only missing factor which prevents our residue class ring Z/(n) from being a field. The proof required that n be prime.

The implication is that some row in the chart must be the identity {1}, and each row must now have an "inverse" row in the multiplicative sense:

identity: $\{r_1\} \bullet \{1\} = \{r_1\}$ inverse: $\{r_1\} \bullet \{r_2\} = \{1\}$

In our example two sections above, 5 was a prime number, so these things must be true. Identity {1} for • is the second row of our chart (1.38) (the row containing the integer 1), as is easy to verify. The • inverse of row {3} is row {2}. Every row has an inverse row. Recall that the first row is the additive identity {0}, which is different from multiplicative identity {1}.

If you try doing the case n = 4 and make the chart, you find a most annoying thing. Yes, the row containing 1 is still the identity {1}, but certain rows don't have any inverses. You multiply some row by all the other rows, and you never get {1} as the answer! The chart is shown below for n=4. Think about the row {2}. It now contains only even numbers. Row {1} contains only odd numbers. What row could there be that such that {2}•{r} = {1} ? We need Rem((2r)/4) = 1, but the remainder of an even number divided by 4 cannot be 1, it can only be 0 or 2. Thus, the row {2} has no inverse.

...	-8	-4	0	4	8	12	...	{0}
...	-7	-3	1	5	9	13	...	{1}
...	-6	-2	2	6	10	14	...	{2}
...	-5	-1	3	7	11	15	...	{3}

(1.45)

So n = 4 does not make a field. Our n must be a prime number, then all the rows will have inverses, and then Z/(n) will be a field.

Chapter 2: The Galois Fields GF(p)

Throughout this Chapter, we shall assume that p is some prime number, p = 2,3,5,7,11,13,...

(a) More Discussion of Z_n = { mod-n,+,•}

In Chapter 1 (b) 5 we showed that that the set of numbers {0,1,2,3,...n-1} formed a cyclic additive group which we called {mod-n,+}. A short list of properties was then presented for {mod-n,+} in (1.21).

The next step was to upgrade this additive group to ring status by endowing it with the modulo • multiplicative operation. This ring was called Z_n = { mod-n,+,•}, and the proof that it really is a ring was presented in full detail below (1.29). In fact, it is a "ring with identity".

Finally, at the end of Chapter 1 we showed in (1.44) that Z_n is a *field* when n is prime. This means that all non-zero elements have inverses. This leads to the next claim:

Fact: The ring Z_n has n elements which are {**0**,**1**,**2**,**3**...**n-1**}, and we have modulo-n operations + and •. If we consider the subset of the n-1 non-zero elements {**1**,**2**,**3**...**n-1**}, this set forms a *group* under the mod-n • operation, provided that n is a *prime number*. Although the modulo operation is mod-n, the order of this group is n-1, since order is the number of elements in a group. (2.1)

Proof: We already know from (1.29) that Z_n is a ring with • identity, so it is just the • inverses that are missing. But since Z_n is a field for prime n from (1.44), we know these inverses all exist. **QED**

Fact : The group just mentioned, {**1**,**2**,**3**...**n-1**} under operation •, is a *cyclic* group for prime n. It follows from this Fact that there is at least one generator g, so we can write the group as in (1.10) with n→n-1,

$$\{1, g, g^2, g^3, \ldots g^{n-2}\} = \text{cyclic group, order} = n-1, \text{ where } g^{n-1} = 1 \ (\ g^n = g) \quad . \tag{2.2}$$

Proof: It will later be shown that *all* Galois Fields are cyclic groups under •, and that Z_n for n prime is a Galois Field, so Z_n is a cyclic group under •. This is the content of Big Theorem 1 (4.30) in Chapter 4.

Fact : If n is prime, then for any element **a** of Z_n we have: $\mathbf{a}^n = \mathbf{a}$. (2.3)

Proof: According to (2.2), we may write our arbitrary element **a** as be $\mathbf{a} = g^k$. Then $a^n = (g^k)^n = g^{kn} = (g^n)^k = (g)^k = a$, since $g^n = 1$ from (2.2).

Example: Let n = 5 = prime, then the 4 elements of our multiplicative group are {1,2,3,4}. These are the non-zero elements of the ring (and field) Z_5. An acceptable generator is 2, since (remember mod 5)

$2^1 = 2 \qquad 2^2 = 4 \qquad 2^3 = 8 = 3 \qquad 2^4 = 16 = 1.$

So we can write out our non-zero Z_5 elements using this generator as:

{ 1,2,4,3} = {1, 2, 2^2, 2^3 } where $2^4 = 1$.

Moreover, we can confirm that $a^4 = a$ for all elements of the group:

$1^4 = 1$ $2^4 = 16 = 1$ $3^4 = 81 = 1$ $4^4 = 256 = 1$.

It is important to realize that not every element is a generator. In (1.13) we showed, once one element is known to be a generator, all elements (other than 1) are generators, provided the *order* of the cyclic group is prime. But here, that order is n-1 which is likely not to be prime even though n is prime.

In fact, we showed in (1.14) that g^i is definitely not a generator if order/i = integer. To verify this, our order is n-1 = 4, so we suspect that $g^i = 2^2 = 4$ will not be a generator. Let us check this (implicit mod 5),

$4^1 = 4$ $4^2 = 16 = 1$ $4^3 = 64 = 4$ $4^4 = 256 = 1$.

Sure enough, we are missing half the elements of our group. Element 4 generates a cyclic subgroup of order 2, which we know has to divide evenly into our group order of 4. On the other hand, since 3 does not divide evenly into 4, we are not surprised that $2^3 = 8 = 3$ *is* a viable generator,

$3^1 = 3$ $3^2 = 9 = 4$ $3^3 = 27 = 2$ $3^4 = 81 = 1$.

(b) The relation between GF(p) and Z_p

We have made the claim near the end of Chapter 1 (a) that finite fields (that is, Galois Fields) with q elements exist only if q is of the form p^m where p is a prime number, and m is a positive integer. For example,

these fields exist: GF(2), GF(3), GF(4 = 2^2), GF(5), ...

these fields do not exist: GF(6 = 2*3), GF(15 = 3*5), ... // q ≠ prime to a power

Furthermore, for a given order q, we claim (without proof) that there is only one such field. So if you find a field of order 5, it has to be GF(5), meaning it has to be isomorphic to GF(5).

We know that for *any* positive integer q, we can write down a pair of "modulo-q" tables for the two operations + and •. Here, for example, are the Z_q tables for q = 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

(2.4)

For any integer q, we get similar tables. So we have a set of q elements and two operations + and •, and the set is "closed" under each operation, and distributive holds, and there is an identity, so for any q the set of elements forms a "ring with identity". This is Z_q. If every element has an inverse, *then* the set advances to the front of the class and is proclaimed to be a *field*.

The + table is always boring, since it just contains the q elements cyclically permuted on each new line.

The • table is more interesting. Notice in the above • table that a 1 appears in every row except the first row. This means that every element except 0 has an inverse, so Z_q is a field when q = 5. Each row except the first is just a reordering of all the elements.

Since we have formed a finite field with 5 elements, it must be equivalent to GF(5).

More generally, since we have already identified a set of fields Z_p for any prime p, these must be the GF(p). Recall from (1.40) that the Z_p rings are equivalent also to the residue class rings Z/(p). Thus, we have arrived at the following

Super Big Fact: For p = prime, GF(p) = Z_p = Z/(p). (2.5)

Remember: GF(p) = one of the Galois Fields having p abstract elements
Z_p = {Mod-p,+,•} having elements 0,1,2..(p-1)
Z/(p) = the residue class ring having elements marked by remainders like {2}

When we say GF(p) = Z_p = Z/(p), we mean these objects are isomorphic to each other. There is a 1-to-1 correspondence between the elements of any two of the three objects, and they all have the same + and • tables.

If p is not a prime, this does not work. For example, here is the • table for Z_4;

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

Notice that the third row does not contain 1, so the element 2 has no multiplicative inverse. This "modulo-4 ring with identity" has four elements, but it is *not* a field.

We know that the field GF(4) = GF(2^2) exists. So we now know that the + and • tables for GF(4) are *not* the Z_4 modulo-4 arithmetic tables we have been talking about above. The correspondence between GF(p) and Z_p only holds when p = a prime number. Just to emphasize this fact, here are the full set of tables for Z_4 and GF(4). The former are by inspection, the latter from (6.24) and (6.27) :

+	0	1	2	3	•	0	1	2	3	Z_4
0	0	1	2	3	0	0	0	0	0	
1	1	2	3	0	1	0	1	2	3	
2	2	3	0	1	2	0	2	0	2	
3	3	0	1	2	3	0	3	2	1	

+	0	1	2	3	•	0	1	2	3	$GF(2^2)$
0	0	1	2	3	0	0	0	0	0	
1	1	0	3	2	1	0	1	2	3	
2	2	3	0	1	2	0	2	3	1	
3	3	2	1	0	3	0	3	1	2	(2.6)

(c) Selected facts about GF(p) = Z_p (p = prime)

Fact: The non-zero elements of GF(p) form a cyclic group under • of order p-1. There exists at least one element to serve as the generator. (2.7)

For example, for p=5 we have 2 as a generator,

$\{2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8 = 3\} = \{ 1,2,4,3 \}$. *(2.2)*

Proof: In Chapter 4 we will prove in Big Theorem 1 that $GF(p^m)$ is cyclic. Our Fact here is then just the special case m =1. See Corollary 1 (4.36).

Fact: For any element of GF(p) = Z_p, we claim that $\mathbf{a}^p = \mathbf{a}$, where **a** = **0**,**1**,**2**....**(p-1).** (2.8)

This fact is very useful later on when we consider the elements of GF(p) = Z_p as coefficients of a polynomial.

Proof: This was proved below (2.3).

Fact: p**a** = **0** for any element **a** of GF(p). (2.9)

Proof: This is because GF(p) = Z_p is an additive cyclic group, see (1.18) item (7) or (1.15).

Example : We are used to the above result in the case of GF(p=2), the binary world, where we say **x** + **x** = 2**x** = **0**. Our usual proof here is to just exhaust the possibilities: 2**1** = **1** + **1** = **0** and 2**0** = **0** + **0** = **0**.

Fact: $(\mathbf{a} + \mathbf{b})^p = \mathbf{a}^p + \mathbf{b}^p$ when **a**,**b** are elements of GF(p), p = prime. (2.10)

Proof: Expanding the left side with the binomial theorem gives p+1 terms and the coefficients are the binomial coefficients for m = 0,1,2....p. Except for the first and last term in the binomial expansion, all

these coefficients are proportional to p, and since p**n** = **0** for any field element **n**, we see that all the cross terms having these coefficients must vanish. In more detail,

$$(\mathbf{a}+\mathbf{b})^p = \mathbf{a}^p + \mathbf{b}^p + \Sigma_{m=1}^{p-1} \frac{p\,(p-1)!}{m!\,(p-m)!}\,\mathbf{a}^p\mathbf{b}^{p-m}, \qquad \text{but } p\mathbf{a}^p\mathbf{b}^{p-m} = (p\mathbf{a})\mathbf{a}^{p-1}\mathbf{b}^{p-m} = \mathbf{0}\,.$$

The key thing here is that, since p is prime, it cannot be eliminated by something in the denominator factorial pair, so p survives in every cross term.

Admittedly the result above conflicts with our usual intuition (field = reals) about what $(a+b)^p$ ought to be. We continue our haphazard notation of sometimes emphasizing group elements with bold font.

Example: $(\mathbf{a}+\mathbf{b})^5 = \mathbf{a}^5 + 5\mathbf{a}^4\mathbf{b} + 10\mathbf{a}^3\mathbf{b}^2 + 10\mathbf{a}^2\mathbf{b}^3 + 5\mathbf{a}\mathbf{b}^4 + \mathbf{b}^5 = \mathbf{a}^5 + \mathbf{b}^5$

Example: $(\mathbf{1}+\mathbf{x})^p = \mathbf{1} + \mathbf{x}^p$, where x is some element in GF(p), p = prime.

Counterexample: $(\mathbf{a}+\mathbf{b})^4 = \mathbf{a}^4 + 4\mathbf{a}^3\mathbf{b} + 6\mathbf{a}^2\mathbf{b}^2 + 4\mathbf{a}\mathbf{b}^3 + \mathbf{b}^4 = \mathbf{a}^4 + \mathbf{b}^4 + 6\mathbf{a}^2\mathbf{b}^2$.

The above fact can be generalized as follows:

Fact: $(\mathbf{a}+\mathbf{b}+\mathbf{c}+\mathbf{d}+\ldots)^p = \mathbf{a}^p + \mathbf{b}^p + \mathbf{c}^p + \mathbf{d}^p + \ldots$ for a,b,c,d... in GF(p), p = prime (2.11)

Proof: Just apply the binomial proof a bunch of times, the first time with **b** + **c** + **d** + ... = **B**, etc.

Conclusion: In this chapter we have identified the Galois Fields GF(p) to be the Z_p. We have not yet nailed down the fields $GF(p^m)$. This is where the notion of polynomials comes into play. As a hint, we note that if you divide a polynomial by some polynomial of degree m, and this polynomial has coefficients in Z_p = GF(p), then there are exactly p^m possible remainders you can get. If we can somehow identify each of these remainders with a row of a residue class ring chart, then we may be on our way to constructing a representation of $GF(p^m)$ which is analogous to Z/(p) for GF(p). Whereas integers sufficed for GF(p) = Z_p and gave us p remainders, we have to go to polynomials to get p^m remainders.

Chapter 3: Polynomials

(a) Specification of the ring R of polynomials with coefficients in Z_p

The highest power of x actually appearing in a polynomial f(x) is called the **degree** of the polynomial. If we have $f(x) = \Sigma_{i=0}^{n} f_i x^i$, then a(x) has degree n only if $f_n \neq 0$. For a polynomial written as a sum in this way, the degree m is the value of the largest index i on f_i for which $f_i \neq 0$.

Let R be the set of polynomials f(x) whose coefficients lie in Z_p. We say *nothing* about the nature of the variable x. It could lie in Z_p or it could lie in some set different from Z_p. In a sense, the variable x is just an inert carrier of the coefficients of the polynomial.

Let a(x) and b(x) be any two polynomials in R. Symbol Σ used below implies the + operation of ring R. For example, if $a(x) = x^3 + 2x$, the + here is the + operation of the ring R (to be defined just below). The Σ_i can be thought of as $\Sigma_{i=0}^{\infty}$ and then the degree of a polynomial is determined by its highest-indexed non-zero coefficient. For the moment, the addition and multiplication operators for the field Z_p will be written ⊕ and ⊗ . Here then are our two arbitrary such polynomials:

$$a(x) = \Sigma_i a_i x^i$$
$$b(x) = \Sigma_i b_i x^i \quad . \tag{3.1}$$

The + and • operations for the ring R are *defined* by the expressions to the right of ≡ on these two lines:

$$a(x) + b(x) = (\Sigma_i a_i x^i) + (\Sigma_i b_i x^i) \equiv \Sigma_i (a_i \oplus b_i) x^i$$

$$a(x) \bullet b(x) = (\Sigma_i a_i x^i) \bullet (\Sigma_j b_j x^j) \equiv \Sigma_{i,j} (a_i \otimes b_j) x^{i+j} \quad . \tag{3.2}$$

Notice that both + and • are commutative because both ⊕ and ⊗ are commutative for Z_p. It is clear that if we write a(x) + b(x) = d(x), then $d_i = a_i \oplus b_i$. If we write a(x) • b(x) = c(x), then Appendix C shows that the c_i are given by

$$c_i = \sum_{j = \max(0,i-A)}^{\min(i, B)} a_{i-j} \otimes b_j \tag{3.3}$$

where A and B are the upper endpoints of the sums in (3.1).

The polynomial a(x) = x has $a_i = \delta_{i,1}$. If a(x) = b(x) = x, the above definitions result in these intuitively desirable results:

$$x + x = \Sigma_i (\delta_{i,1} \oplus \delta_{i,1}) x^i = 1 \oplus 1\ x^1 = 2x \qquad \text{// assuming } p > 2$$
$$x \bullet x = \Sigma_{i,j} (\delta_{i,1} \otimes \delta_{j,1})\ x^{i+j} = 1 \otimes 1\ x^2 = x^2 \quad .$$

Thus, although we have not specified the nature of the polynomial argument x, we have a precise meaning for things like 3x = x+x+x and $x^3 = x \bullet x \bullet x$. In particular, we know that px = 0 :

$$px = x + x + x + ... + x = (1 \oplus 1 \oplus 1 \oplus ..... \oplus 1)x = (0)x = 0. \qquad (3.4)$$

We know that x is one of the polynomials in our set R.

Subtraction of two polynomials in R is just a special case of addition:

$$a(x) - b(x) = (\Sigma_i a_i x^i) - (\Sigma_i b_i x^i) \equiv \Sigma_i (a_i \oplus (p - b_i)) x^i \qquad (3.5)$$

where we know that $- b_i = p - b_i$ in Z_p, since then $b_i + (-b_i) = p = 0$.

We shall now show in full detail that the set R with + and • as defined in (3.2) forms a ring with identity:

+ associative:

$$[\, a(x) + b(x) \,] + c(x) = \Sigma_i (a_i \oplus b_i) x^i + (\Sigma_i c_i x^i) = \Sigma_i (a_i \oplus b_i) \oplus c_i \, x^i = \Sigma_i \, a_i \oplus (b_i \oplus c_i) x^i$$
$$a(x) + [b(x) + c(x)] = (\Sigma_i a_i x^i) + \Sigma_i (b_i \oplus c_i) x^i = \Sigma_i \, a_i \oplus (b_i \oplus c_i) x^i \qquad \textbf{QED}$$

+ identity: $i(x) = 0$ $\quad i(x) + a(x) = (\Sigma_i 0 x^i) + (\Sigma_i a_i x^i) = \Sigma_i (0 \oplus a_i) x^i = \Sigma_i a_i x^i = a(x)$

+ inverse: $-a(x) = \Sigma_i([-a_i]) x^i$ $\quad$ // $[-a_i]$ exists since Z_p is a field
$$a(x) - a(x) = (\Sigma_i a_i x^i) + \Sigma_i([-a_i]) x^i = \Sigma_i \, a_i \oplus [-a_i] \, x^i = \Sigma_i \, 0 \, x^i = 0 \qquad \textbf{QED}$$

+ commutative: $a(x) + b(x) = \Sigma_i (a_i \oplus b_i) x^i = \Sigma_i (b_i \oplus a_i) x^i$
$$b(x) + a(x) = \Sigma_i (b_i \oplus a_i) x^i \qquad \textbf{QED}$$

• associative

$$[a(x) \bullet b(x)] \bullet c(x) = \Sigma_{i,j} (a_i \otimes b_j) \, x^{i+j} \bullet (\Sigma_k c_k x^k) = \Sigma_{i,j,k} (a_i \otimes b_j) \otimes c_k \, x^{i+j+k}$$
$$= \Sigma_{i,j,k} \, a_i \otimes (b_j \otimes c_k) \, x^{i+j+k}$$

$$a(x) \bullet [b(x) \bullet c(x)] = (\Sigma_i a_i x^i) \bullet \Sigma_{j,k} (b_j \otimes c_k) \, x^{j+k} = \Sigma_{i,j,k} \, a_i \otimes (b_j \otimes c_k) \, x^{i+j+k} \qquad \textbf{QED}$$

+/• distributive

$$a(x) \bullet [b(x) + c(x)] = (\Sigma_i a_i x^i) \bullet \Sigma_j (b_j \oplus c_j) x^j = \Sigma_{i,j} \, a_i \otimes (b_j \oplus c_j) \, x^{i+j}$$
$$= \Sigma_{i,j} [\, a_i \otimes b_j \, \oplus \, a_i \otimes c_j \,] \, x^{i+j}$$

$$a(x) \bullet b(x) + a(x) \bullet c(x) = \Sigma_{i,j} (a_i \otimes b_j) \, x^{i+j} + \Sigma_{i,j} (a_i \otimes c_j) \, x^{i+j}$$
$$= \Sigma_{i,j} [(a_i \otimes b_j) \oplus (a_i \otimes c_j)] \, x^{i+j} \qquad \textbf{QED}$$

• identity: $I(x) = 1 \quad \Rightarrow I_i = \delta_{i,0}$

$$I(x) \bullet a(x) = \Sigma_{i,j} (I_i \otimes a_j) \, x^{i+j} = \Sigma_j (1 \otimes a_j) \, x^j = \Sigma_j (a_j) \, x^j = a(x) \qquad \textbf{QED}$$

To summarize, we considered the set of polynomials f(x) having coefficients in Z_p and having the variable x in some unspecified set. We defined certain + and • operations for the sum and product of two arbitrary polynomials in our set R. We then showed that with these operations, this set forms a commutative ring with identity. So,

Definition: R is the commutative ring of polynomials whose coefficients lie in Z_p and whose + and • operations are defined by

$$a(x) + b(x) = (\Sigma_i a_i x^i) + (\Sigma_i b_i x^i) \equiv \Sigma_i (a_i \oplus b_i) x^i$$
$$a(x) \bullet b(x) = (\Sigma_i a_i x^i) \bullet (\Sigma_j b_j x^j) \equiv \Sigma_{i,j} (a_i \otimes b_j) x^{i+j} \qquad (3.2)$$

where $\oplus$ and $\otimes$ are the field operations for Z_p. The argument x is unspecified.

Observation: In all of the above, we could replace Z_p with $\mathcal{R}$, where $\mathcal{R}$ is some arbitrary ring with identity. Then R becomes the ring of polynomials whose coefficients lie in the ring $\mathcal{R}$, and then $\oplus$ and $\otimes$ are the operations which define the ring $\mathcal{R}$. Candidates for $\mathcal{R}$ would be the real numbers, the complex numbers, or the integers Z. One could of course select $\mathcal{R}$ to be a field F as we did above with $\mathcal{R}$ = F = Z_p. The point is that there was never any need for a multiplicative inverse in the above discussion.

Usually, when one speaks of "polynomials over a ring $\mathcal{R}$" or "polynomials over a field F", one implies that both the variable x and the coefficients lie in $\mathcal{R}$ or F, but we do not make that implication for the variable x.

(b) Basic facts about polynomials with coefficients in a ring $\mathcal{R}$

As just noted, $\mathcal{R}$ is really a ring with identity, and it could be a field F.

Here we mimic Chapter 1 (c) 6:

• **Division Algorithm**. Relative to a divisor polynomial d(x), a polynomial n(x) has a unique quotient polynomial q(x), and a unique remainder polynomial r(x):

$$n(x)/d(x) = q(x) + r(x)/d(x) \qquad \text{"Rem}(n(x)/d(x)) = r(x)\text{"}$$

$$n(x) = q(x) \bullet d(x) + r(x) \qquad (3.6)$$

The remainder polynomial r(x) has degree less than the divisor polynomial d(x). This algorithm should be compared to (1.41) for integers, where n = q•d + r .

Example: Let's work with ring $\mathcal{R}$ = F = GF(2), and let $n(x) = 1 + x^2 + x^5$, and let d(x) = 1 + x. It is perhaps useful to remind calculator-era readers how long division works. Recall that in GF(2), the symbols + and – have the same meaning! Also, $2x^5 = (2x)x^4 = 0x^4 = 0$, and so on.

$$
\begin{array}{rl}
 & \underline{x^4 + x^3 + x^2} \\
x + 1 \;\big|\; & x^5 + x^2 + 1 \\
 & \underline{x^5 + x^4} \\
 & \quad x^4 + x^2 + 1 \\
 & \quad \underline{x^4 + x^3} \\
 & \quad\quad x^3 + x^2 + 1 \\
 & \quad\quad \underline{x^3 + x^2} \\
 & \quad\quad\quad 1
\end{array}
\qquad (3.7)
$$

So the idea is to put largest powers first in both divisor and dividend, then turn the crank. So here is our result (where now we put the lowest power first),

$$\frac{1 + x^2 + x^5}{1+x} = (x^2 + x^3 + x^4) + \frac{1}{1+x},$$

which can also be expressed in the other forms,

$$(1 + x^2 + x^5) = (x^2 + x^3 + x^4)\bullet(1+x) + 1 \qquad \text{Rem}\left[\frac{1 + x^2 + x^5}{1+x}\right] = 1$$

$$n(x) \quad = \quad q(x) \quad \bullet d(x) + r(x) \qquad \text{Rem}\ \ (n(x)/d(x)) = r(x)\ .$$

Observation about working with F = GF(p). Suppose the divisor d(x) is a polynomial of degree m. We know that all remainders must have degree less than this. How many possible remainders are there? There are m coefficients in the remainder, and each coefficient must take one of the p values in the field GF(p) = Z_p. Thus, there are **p^m possible remainders** of degree m-1 or less, where we have included the possibility that all coefficients are 0.

• **Factorization Theorem**. Any polynomial can be uniquely decomposed into a product of factors where each factor is an "irreducible" (in R) polynomial raised to an integer power:

$$f(x) = [p_1(x)]^{m1} \bullet [p_2(x)]^{m2} \bullet [p_3(x)]^{m3} \quad ... \qquad (3.8)$$

The meaning of the term irreducible will be clarified in section (c) below. Again, notice the powerful analogy with (1.42) which said $n = (p_1)^{m_1} (p_2)^{m_2} (p_3)^{m_3}$ .... .

Example: for $\mathcal{R}$ = Z (integers)

$$f(x) = x^4 - 4x^3 + 5x^2 - 4x + 4 = [x^2+1]^1 \bullet [x-2]^2$$

• **Euclidean Algorithm and Bezout's Identity**. The Euclidean Algorithm is a set of instructions one can use to find d(x) = GCD($n_1(x)$, $n_2(x)$). Bezout's Identity then says one can write this divisor as a unique linear combination of the two polynomials, where the coefficients are again polynomials:

$$d(x) = a(x)\bullet n_1(x) - b(x)\bullet n_2(x) \qquad \text{// compare to (1.43), } d = a\, n_1 - b\, n_2 \qquad (3.9)$$

There is a mechanical procedure for finding d, a, b once you are given n_1, and n_2. See for example Rhee pages 25-26. Basically, it is the same algorithm used for the integer analog.

(c) The meaning of a polynomial in R being irreducible in R

The term "irreducible" is just a bit slippery so we expend some effort here to nail it down as precisely as possible along with a few examples. The definition of ring R is stated near the end of section (a) above. First, we need the following:

Fact: Let n(x) and d(x) be polynomials in R.
Let n(x) have degree m, and let d(x) have degree k < m.
Then if n(x) is divided by d(x) such that n(x)/d(x) = q(x) + r(x)/d(x), both the quotient polynomial q(x) and the remainder polynomial r(x) will be elements of R (3.10)

Proof by example. Consider this sample division of n(x)/d(x)

$$
\begin{array}{rl}
 & 4x^3 + x^2 + x + 1 \\
2x+3 \;|\; & 3x^4 + 4x^3 + 2 \\
 & \underline{3x^4 + 2x^3} \qquad\qquad // \; 8 = 3 \text{ and } 12 = 2 \\
 & \quad 2x^3 + 2 \\
 & \quad \underline{2x^3 + 3x^2} \\
 & \qquad 2x^2 + 2 \qquad\qquad // \; -3x^2 = 2x^2 \\
 & \qquad \underline{2x^2 + 3x} \\
 & \qquad\quad 2x + 2 \qquad\quad // \; -3x = 2x \\
 & \qquad\quad \underline{2x + 3} \\
 & \qquad\qquad 4 \qquad\quad // \; -1 = 4
\end{array}
$$

We find here that $q(x) = 4x^3 + x^2 + x + 1$ and $r(x) = 4$. In the long division process, we have alternate multiplications and subtractions (additions) of pairs of polynomials. Since these are carried out with the • and + operations of the ring R, the coefficients in the quotient and remainder polynomials always end up lying in Z_p. Here is a Maple verification of the above division:

```
Quo(3*x^4+4*x^3+2,2*x+3,x) mod 5;
```

$$4x^3 + x^2 + x + 1$$

```
Rem(3*x^4+4*x^3+2,2*x+3,x) mod 5;
```

$$4$$

Corollary. In the decomposition n(x) = q(x)d(x) + r(x), if n(x) and d(x) lie in R, then q(x) and r(x) also lie in R. (3.11)

We now list off several equivalent definitions of the meaning of n(x) in R being **reducible** in R. It is understood that all f(x) here are polynomials: (3.12)

1. Given n(x) in R of degree m, *if* there exists some d(x) in R of degree $0 < k < m$ such that in the known unique expansion $n(x) = q(x)d(x) + r(x)$ one gets $r(x) = 0$, *then* n(x) is said to be **reducible** in R. As noted above, q(x) will also be in R.

2. Given n(x) in R of degree m, if there exists some d(x) in R of degree $0 < k < m$ such that $n(x) = q(x)d(x)$ where both d(x) and q(x) are in R, then n(x) is **reducible** in R.

3. If n(x) in R can be factored into a product of factors q(x)d(x) both of which lie in R and both of which have degree > 0, then n(x) is **reducible** in R.

4. Given n(x) in R of degree m, if $GCD(n(x),d(x)) \neq 1$ for some d(x) in R of degree $0 < k < m$, then n(x) is **reducible** in R.

Examples:

$n(x) = (x-a)\bullet(x-b)$ is reducible since $d(x) = (x-a)$ is a dividing factor with $0 < k < 2$ (ie, k= 1). Notice the assumption implicitly made here that both a and b are elements of Z_p. If either a or b does not lie in Z_p, then at least one of the factors does not lie in R, and in that case this n(x) would not be reducible according to item 2 above.

$n(x) = x^2 + 2x + 1$ is reducible since it equals $(x+1)\bullet(x+1)$.

$n(x) = x^2 + 1$ is not reducible for Z_p with $p > 2$ because it cannot be factored in such a way that the factors are both in R with the degree of both factors being > 0. For example, $(x+i)\bullet(x-i)$ is not allowed because (x+i) does not lie in R since i does not lie in Z_p. However, in the special case of Z_2 we can write $x^2+1 = x^2- 1 = (x+1)\bullet(x-1)$ and then x^2 is reducible in this particular R.

$n(x) = (x-a)$ is not reducible because no d(x) exists with $0 < k < 1$ that could be a divisor.

$n(x) = (2x-a)$ is not reducible for the same reason

$n(x) = 2\bullet(x-a)$ is not reducible for the same reason.

If n(x) in R is not reducible in R, then it is **irreducible** in R. The last three examples above show n(x) that are irreducible, and $x^2 + 1$ is irreducible for Z_2 only.

Comment: For the more general R consisting of polynomials with coefficients in a ring-with-identity $\mathcal{R}$, one can define the irreducibility of f(x) exactly as above: just replace Z_p with $\mathcal{R}$. The famous example is that x^2+1 is irreducible in $R^{(r)}$ (R with $\mathcal{R}$ = reals) but is reducible in $R^{(c)}$ (R with $\mathcal{R}$ =complex).

(d) The Residue Class Decomposition of R

In this section we continue to use • and + to indicate the two operations of the ring R defined at the end of section (a) above. Eventually we will stop displaying • so that a(x) • b(x) will be written as a(x)b(x), but for the present • is retained.

Now we are going to repeat exactly the discussion of Chapter 1 (c) 1 about decomposing a ring based on some ideal of the ring. First, we have to find a candidate ideal. We know from the integer ring case that one could make an ideal by taking all multiples of some integer N. In the polynomial case, one can form an ideal within the ring of polynomials R consisting of all polynomials which are multiples of some specified polynomial f(x). We denote by (f(x)) the set of all polynomials which are multiples of f. We shall assume that f(x) is some arbitrary polynomial in R **of degree m**.

Proof that (f(x)) is an ideal within R:

There is nothing mysterious about this. If polynomial F(x) is a multiple of f(x), then there is some q(x) such that F(x) = q(x)f(x). One might have to do a bit of factoring of F(x) to expose the fact that it can be written in this manner. This is trivial, but the point should not be missed. For example, if $\mathcal{R}$ = reals, we have

$(x^2 - 1) = (x + 1)\bullet(x - 1)$ so $(x^2 - 1)$ is a "multiple" of either (x + 1) or (x -1) .

So we have come up with a claimed ideal I of the ring R of polynomials. The ideal consists of only those polynomials in R which are multiples of f(x). Why is this set an ideal? First of all, it is a ring (and an abelian one at that) because the properties of (1.5) are trivially satisfied, including closure under both + and •. To see this, consider two polynomials in the ideal,

$F_1(x) = f_1(x) \bullet f(x)$
$F_2(x) = f_2(x) \bullet f(x)$.

Closure under +: $F_1 + F_2 = f_1\bullet f + f_2\bullet f = (f_1 + f_2)\bullet f = f_3\bullet f$ = in the ideal since multiple of f

Closure under •: $F_1\bullet F_2 = (f_1\bullet f)\bullet (f_2\bullet f) = (f_1\bullet f\bullet f_2)\bullet f = f_4\bullet f$ = in the ideal since multiple of f

So (f(x)) is a subring of the ring of all polynomials. From definition (1.22) we then have to show that for any r(x) in R and i(x) in I, r(x) • i(x) is in I (then i(x)•r(x) will also be in I since • is commutative):

$i(x)\bullet r(x) = (j(x) \bullet f(x)) \bullet r(x) = (j(x) \bullet r(x)) \bullet f(x)$ = in the ideal **QED**

Thus, this restricted set of polynomials in R forms an ideal which we call I = (f(x)).

Now we make the chart, just as before. Here is our Chapter 1 chart for general ring R and ideal I:

$i_1=0$	i_2	i_3	i_4	...	i_k	// residue class #1
r_1	$r_1+ i_2$	$r_1+ i_3$	$r_1+ i_4$	...	$r_1+ i_k$	// residue class #2
r_2	$r_2+ i_2$	$r_2+ i_3$	$r_2+ i_4$	...	$r_2+ i_k$	// etc
more rows like the above						*(1.23)*

We now apply this to our polynomial situation (comments below),

$i_1(x)=0$	$i_2(x) = q(x)\bullet f(x)$ for all possible q(x)	
$r_1(x)$	$r_1(x) + i_2(x) = q(x)\bullet f(x) + r_1(x)$, for all possible q(x)	
$r_2(x)$	$r_2(x) + i_2(x) = q(x)\bullet f(x) + r_2(x)$, for all possible q(x)	
more rows like the above		(3.13)

First look at the top row, which is the ideal. We separate out the polynomial 0 and put it first. Then, instead of trying to list off specific elements of the ideal going left to right, we write this list by saying that the polynomial q(x) can be any polynomial in R. As q(x) ranges over all possible polynomials, we get our list for that row. It is of course an infinite list since we can have q(x) of any degree.

Now look at the next row. We pick some $r_1(x)$ from the ring R of polynomials of degree < m and plop it down in the left column, then we form the row as instructed by the previous prototype chart (1.23). Now suddenly one sees an interesting coincidence of the notation. The polynomials $r_i(x)$ which are the residue class (row) leaders, look like "remainder" polynomials, and the q(x) notation suggests "quotient" polynomials. In fact, all polynomials in the second row have $r_1(x)$ as their remainder when they are divided by f(x), and q(x) is in fact the quotient polynomial. There are as many polynomials in the second row as there are quotient polynomials q(x). As noted above, this number of quotient polynomials is infinite. [We can in fact label a chart row by any of the elements in that row, but it is best to use the remainder polynomial as the "residue class leader" . Each row has only one polynomial of degree < m.]

So here is the general idea of the above chart. Across the top you have the ideal which is the set of all polynomials which are multiples of f(x) with *no* remainder. Then each row is represented by some possible remainder (of degree less than that of f(x)) you could get by dividing f(x) into elements of the polynomial ring.

How many rows are there? Since f(x) is of degree m, a remainder polynomial is of degree m-1 or less. A remainder polynomial then has m coefficients. Since each coefficient lies in Z_p, there are p^m such remainder polynomials, and so there are p^m rows in the chart including the first row of the ideal.

{} Notation: Recall that we have used the notation {joe} to label the row of a coset chart in the case of groups and subgroups, and to label the row of a residue class ring chart in the case of rings and ideals. The "joe" object in these two cases was the coset leader or the residue class leader. See for example Chapter 1 (b) 2, Chapter 1 (c) 2 or (1.38). This notation continues in the polynomial world, and $\{r_k(x)\}$ denotes the residue class chart row whose remainder function is $r_k(x)$. For GF(3^m) some sample chart rows would be indicated by {1}, {2}, {x}, {2x}, {x^2}, {$2x^2$} and so on. For GF(2^m) we have {1}, {x}, {x^2}, {$x+x^2$}, etc.

We pause for a moment to note a few facts concerning this notation. We are running out of symbols for multiplication, so we use + and • for both polynomials and chart rows.

Facts using {} notation: (3.14)

(a) $\{r_1(x)+r_2(x)\} = \{r_1(x)\} + \{r_2(x)\}$

(b) $\{r_1(x)\bullet r_2(x)\} = \{r_1(x)\} \bullet \{r_2(x)\}$

(c) $\{c\ r_2(x)\} = c\{r_2(x)\}$ for c in Z_p

(d) $\{cx^k\} = c\{x^k\}$ for c in Z_p

(e) $\{x^k\} = \{x\}^k$

(f) $\{p(x)\} = p(\{x\})$ where p(x) is a polynomial with coefficients in GF(p)

(g) $\{r_1(x)\} + \{r_2(x)\} = \{r_3(x)\}$ where $r_3(x) = r_1(x) + r_2(x)$

(h) $\{r_1(x)\} \bullet \{r_2(x)\} = \{r_4(x)\}$ where $r_4(x) = \text{Rem}[\ (r_1(x)\bullet r_2(x))/f(x)\ ]$

Proof: In some proofs we just give examples and the general proof should be obvious.

(a) and (b) The equations here are just statements of how elements of a residue class ring are added and multiplied. If the degree of $r_1(x)\bullet r_2(x) > m$, it is still OK to label a row as $\{r_1(x)\bullet r_2(x)\}$.

(c) for example, $\{2\ r_2(x)\} = \{\ r_2(x) + r_2(x)\} = \{r_2(x)\} + \{r_2(x)\} = 2\ \{r_2(x)\}$ // used (a)

(d) special case of (c) where $r_2(x) = x^k$

(e) for example, $\{x^3\} = \{x^2\}\bullet\{x\} = \{x\} \bullet \{x\} \bullet \{x\} = \{x\}^3$.

(f) $\{p(x)\} = \{\Sigma_i a_i x^i\} = \Sigma_i \{a_i x^i\}$ (a) $= \Sigma_i a_i \{x^i\}$ (c) $= \Sigma_i a_i \{x\}^i$ (e) $= p(\{x\})$

(g) and (h). Normally we like to label chart rows with remainder polynomials of degree < m, and in these two lines we assume all the $r_i(x)$ have degree < m. For addition, the sum of two polynomials of degree < m itself has degree < m, so no remaindering is necessary. For multiplication, it often happens that the degree of the product of two polynomials of degree < m will have degree ≥ m, so in this case we must do the remaindering as shown. The elements of R/(f(x)) can be regarded either as these chart rows, or as the set of remainder polynomials of degree < n which label the chart rows. In the integer world, the analog statements of (g) and (h) are these, from (1.39) where we replace 5 by n,

$\{i\} + \{j\} = \{k\}$ where $k = \text{Rem}[(i+j)/n] = (i+j) \bmod n$

$\{i\} \bullet \{j\} = \{m\}$ where $m = \text{Rem}[(ij)/n] = (ij) \bmod n$ *(1.39)*

Since the sum of two integers each of which is < n an exceed n, remaindering here is required for both addition and multiplication, whereas in the polynomial case remaindering is needed only for multiplication.

We are now rapidly closing on the target of all our efforts. Admittedly, it has taken a while.

We considered our special ring R of polynomials with coefficients in Z_p, and we considered an ideal consisting of all polynomials which were multiples of some selected polynomial f(x) of degree m. We then used this ideal to construct the "chart" which partitions the ring into rows of polynomials, the residue classes. There is one row for each possible remainder polynomial, and the polynomials across a row are "indexed" by the quotient polynomial. The number of rows equals the number of possible remainder polynomials p^m , and the first row corresponds to remainder 0.

We know from Chapter 1 (c) 2 that these rows can be considered elements of a new ring known as the residue class ring. We indicated this new ring by the notation R/I,

residue class ring = R/(f(x))

If the degree of f(x) is m, the residue class ring has p^m elements.

Here then comes the grand finale, the analogy to the Big Fact of (1.44):

Theorem: If a polynomial f(x) is *irreducible* in R, then R/(f(x)) is a *field*. (3.15)

Proof: The proof exactly parallels the proof of (1.44). We have to show that for some {r(x)} we can find an inverse {s(x)} such that {r(x)}• {s(x)} = {1}. We will mimic every statement of the previous proof:

We note that, since f(x) is irreducible in R and r(x) has smaller degree than f(x), they cannot be multiples, so their greatest common divisor is 1. Using Bezout's Identity (3.5) above, we claim that polynomials N(x) and M(x) must exist such that 1 = N(x)r(x)-M(x)f(x). This says that there exists an N(x) such that 1 = Rem[N(x)r(x)/f(x)]. Now expand N(x) = K(x)f(x) + s(x), where the degree of s(x) is less than that of f(x). This gives $1 = \text{Rem}(\frac{K(x)f(x)r(x) + s(x)r(x)}{f(x)}) = \text{Rem}[s(x)r(x)/f(x)]$. This says that {s(x)r(x)} = 1 and therefore {r(x)}• {s(x)} = {1}.

Thus, if f(x) is *irreducible* in R then the residue class *ring* R / (f(x)) is a *field*. **QED**

Implication: Find some irreducible polynomial f(x) of degree m in the ring R. Then the residue class ring R / (f(x)) is a field with p^m elements, because there are p^m rows in the chart (3.6). *But*, from Chapter 1 we know that all finite fields must be equivalent to the Galois Fields, so we have explicitly constructed a representation of GF(p^m) !! (3.16)

In ring R, the coefficients of the polynomials are elements of Z_p = GF(p), which is the **base field** or **ground field**. The elements of the residue class ring R / (f(x)) represent GF(p^m) which is the **extension field**. So we now have a method whereby we start with a ground field that we know about, namely GF(p), and we construct a pile of new extension fields GF(p^m), where m = any integer. From our construction, we should be able to learn everything about GF(p^m) and therefore everything about any finite field. We have arrived at a major waypoint along our little voyage.

(e) Comparison Between Chapter 3 and Chapter 1

In Chapter 1 we considered the residue class ring Z/(p) for p = prime and showed it was a field having p elements. This got us used to working with a residue class ring. We knew all along that this field was equivalent to Z_p , the modulo-p field, so we didn't really need Z/(p), it was just practice.

In Z/(p) the ring Z was integers. We partitioned up the integers into chart rows, where the integers in each row had something in common: some remainder r < p when divided by p. There were p rows.

Here in Chapter 3 we really need the residue class ring. We need something with p^m elements, and we realize that a R/(f(x)) has p^m remainders if f(x) is of degree m. If we take f(x) to be irreducible in R, then this residue class ring R/(f(x)) becomes a field with p^m elements, and then

$$GF(p^m) = R / (f(x)) \qquad \begin{array}{l} R = \text{polys with coefficients in } Z_p = GF(p) \\ f(x) = \text{degree m, irreducible in R .} \end{array} \qquad (3.17)$$

Obviously, there are p remainder polynomials which are constants independent of x. One can consider these remainders to be generating GF(p) as a subfield of $GF(p^m)$, a subject we examine in the next Chapter.

Chapter 4: The Galois Fields GF(q=p^m).

Note: In this chapter, we continue to assume p = a prime number. Also, to avoid writing p^m very many times, we use the shorthand symbol q to stand for p^m . Thus:

$q = p^m$ p = prime number m = positive integer (4.1)

(a) GF(p) is a subfield of GF(q)

Adding or multiplying two elements in GF(q) means adding or multiplying two rows in the chart (3.13). This in turn means adding or multiplying two representative remainder polynomials, one from each row. These remainder polynomials lie in ring R, meaning they have coefficients in Z_p and variable x in some unspecified set. When we add or multiply polynomials, the + and • operations are those stated in (3.2). When a product of two remainder polynomials is improper relative to f(x), it is understood that this product is replaced by its remainder of division by f(x).

The chart has p (of q) rows which correspond to remainder polynomials which are just numbers of Z_p. These first p rows can be taken as a representation of GF(p). These are the remainder polys of degree 0. If we add or multiply two of *these* rows, we get a remainder which is again just a number. Thus, at the level of the residue class ring where we imagine that the q rows are the elements of GF(q), we can consider the p rows having degree 0 remainders to be the elements of GF(p). Since these rows mix only with themselves under + and •, we see that GF(p) is a subgroup of GF(q) with respect to either operation, with the understanding that 0 is excluded if we use the • operation. Since GF(p) and GF(q) are both fields, we can say that GF(p) is a *subfield* of GF(q). This was noted at the end of the last Chapter, here we firm up the idea.

Since {GF(p) - 0} (order p-1) is a subgroup of {GF(q) - 0} (order q-1) under the • operation of GF(q), we can do the usual coset decomposition as discussed in Chapter 1. So we can make a new chart like (1.7) -- having nothing to do with polynomials -- where we list the p-1 non-zero elements of subgroup H = {GF(p) - 0} across the top, and then we fill out the rows ,

h_1=1	h_2	h_3	h_4	...	h_{p-1}
g_1	$g_1 \bullet h_2$	$g_1 \bullet h_3$	$g_1 \bullet h_4$	...	$g_1 \bullet h_{p-1}$
g_2	$g_2 \bullet h_2$	$g_2 \bullet h_3$	$g_2 \bullet h_4$	...	$g_2 \bullet h_{p-1}$
more rows like the above .					

(4.2)

Since operation • is commutative for GF(p) = Z_p, the subgroup {GF(p) - 0} is an *invariant* subgroup and we can always write g•h= h•g , so the left and right coset decompositions are the same (see text below (1.7).

This decomposition shows that (q-1)/(p-1) = integer. We can express this as:

$$(p^m-1)/(p-1) = 1 + p^2 + p^3 + ... + p^{m-1} \quad (4.3)$$

So, from the above coset decomposition, we arrive at this fact:

Fact: Any non-zero element of GF(q) can be written in the form **g** = **g'•h**= **h•g'** where **h** is an element of GF(p). Of course we can then extend this fact to apply to the zero element as well. (4.4)

We are now in a position to make a claim about GF(q) that is very similar to an earlier claim made about GF(p) :

Fact: p**g** = **0** for any element **g** of GF(q). // compare to (2.9) (4.5)

Proof: Write **g** in the form **g** = **h•g'** as noted above. Then

p**g** = p(**h•g'**) = (**h•g'**) + (**h•g'**) + ... = (**h** + **h** + **h** + ...)**•g'** = (p**h**)**•g'** = 0 .

The last equality comes from (2.9) where we showed that p**h** = 0 for any element h in GF(p).

(b) Representing GF(q) Field Elements as Polynomials and as m-tuples

We have developed the representation GF(q) = R/(f(x)) as in (3.17). In this representation, each element of the field GF(q) is associated with a "row of the chart" (3.13). Each row of the chart in turn is associated with a remainder polynomial which has degree less than m, the degree of f(x).

An obvious notation for describing each remainder polynomial is an **m-tuple** which just lists off the polynomial coefficients. There are of course two ways to do this: lowest power on the left, or highest power on the left. We shall use the notation abc or [abc] for highest power on the left, and <cba> for lowest power on the left. The second notation matches the normal way one writes down a polynomial (constant first), while the first notation matches the conventional manner in which primitive polynomials (Chapter 5) are presented.

Here then is an example for m = 4:

$f(x) = a + bx + kx^3 = \langle ab0k \rangle = [k0ba] = k0ba$ // = a 4-tuple (4.6)

where a,b,0,k are all elements of Z_p.

If p = 2, then Z_2 = {0,1} so each m-tuple is then a sequence of 1's and 0's, for example,

$f(x) = 1 + x + x^3 = \langle 1101 \rangle$.

Comment: For GF(2) a - sign is the same as a + sign. The reason is that -y = y for y = 0 or 1. In the case of y =1, we have -y = -(1) = (-1) = 1, since 1 + (-1) = 1 + 1 = 0, modulo 2 addition. This equivalence of + and - signs will be quietly used in various examples below.

Two constant polynomials are of particular interest for arbitrary m:

f(x) = 0 = <0000...> **0** element of GF(q)
f(x) = 1 = <1000...> **1** element of GF(q) (4.7)

Each element of GF(q) is associated with a particular remainder polynomial of degree less than m, and therefore with a particular m-tuple having m Z_p integers. We can therefore use these m-tuples to label the GF(q) field elements in a clear, unambiguous fashion.

Now consider two remainder polynomials a(x) and b(x), for some m. The m-tuples are also shown.

$$a(x) = a_0 + a_1 \bullet x + a_2 \bullet x^2 + a_3 \bullet x^3 + \ldots \quad = \langle a_0\ a_1\ a_2\ a_3 \ldots \rangle$$
$$b(x) = b_0 + b_1 \bullet x + b_2 \bullet x^2 + b_3 \bullet x^3 + \ldots \quad = \langle b_0\ b_1\ b_2\ b_3 \ldots \rangle$$

As noted above, *adding* two GF(q) field elements means adding rows of the chart, and this means adding remainder polynomials. This in turn means adding the coefficients. But the coefficients live in Z_p, and addition here is just modulo-p addition. Thus, we can add the above remainder polynomials to get the following result:

$$a(x) + b(x) = c(x) = c_0 + c_1 \bullet x + c_2 \bullet x^2 + c_3 \bullet x^3 + \ldots = \langle c_0\ c_1\ c_2\ c_3 \ldots \rangle$$

where

$$c_i = a_i \oplus b_i \quad \text{where } \oplus \text{ means addition modulo p} \qquad (4.8)$$

We have just proven :

Fact: If the elements of GF(q) are represented as polynomial coefficient m-tuples, the addition table can be written down immediately with no further ado. One just does a modulo-p addition on each m-tuple position. Knowledge of the defining polynomial f(x) is not even needed. (4.9)

Example: Here is the addition table for GF(4) = GF(2^2) where recall that the unbracketed m-tuple notation means the lowest power on the right. The same table is shown on the right where each element is given a decimal value from the usual binary evaluation of a sequence of 1's and 0's

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

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

(4.10)

Now what about multiplication?

Following the same line of argument, and writing things out in the obvious manner, one can easily show that the product of two field elements, each represented by some remainder $r_a(x)$ and $r_b(x)$ appearing in the left column of chart (3.13), is a new field element represented by the product of the remainders. However, although each remainder is of degree less than m, it may happen that the product remainder has degree more than m. In this case, as is easily shown, the right thing to do is take this "improper" product remainder, divide it by f(x), and the resulting proper remainder is the remainder you want, call it $r_c(x)$. This of course requires a knowledge of f(x).

So we see that some *work* is needed to produce the GF(q) multiplication table. Consider these two remainders and their product $R_c(x)$,

$$r_a(x) = \Sigma_{i=0}^{m-1} a_i x^i$$
$$r_b(x) = \Sigma_{j=0}^{m-1} b_j x^j$$

$$R_c(x) \equiv r_a(x) \bullet r_b(x) = \Sigma_{i=0}^{m-1} \Sigma_{j=0}^{m-1} (a_i \otimes b_j)\, x^{i+j} \quad .$$

According to Appendix C, the product can be written,

$$R_c(x) = \Sigma_{s=0}^{2m-2} C_s\, x^s \qquad \text{where } C_s = \sum_{j=\max(0,s-m+1)}^{\min(s,\,m-1)} (a_{s-j} \otimes b_j) \tag{4.11}$$

Once we have computed the polynomial $R_c(x)$, we have to see if it is improper -- if it has degree larger than m-1. If so, we have to divide it by the defining polynomial of our construction f(x) and get the remainder. This is then $r_c(x)$, and the coefficients of $r_c(x)$ can then be encoded as an m-tuple, and this is then put into the multiplication table to represent the product of our two particular rows $r_a(x)$ and $r_b(x)$.

So: if $R_c(x)$ has degree $< m$, then $r_c(x) = R_c(x)$
if $R_c(x)$ has degree $\geq m$, then $R_c(x) = q(x)f(x) + r_c(x)$. (4.12)

Although "work" is required to get the multiplication table, it is a purely mechanical crank-turning process. No magic is involved. However, for the improper entries, knowledge of f(x) is required so that the division can be done.

From the basic theorem (3.15) we know that R/(f(x)) is a *field* as long as f(x) is an *irreducible* polynomial of degree m. The term irreducible was defined in section (c) above. Perhaps the simplest definition is that n(x) is irreducible if it cannot be factored into a product q(x)d(x) where both factors are in R and both have degree > 0.

Let us now search for viable candidate polynomials f(x) of degree 2 that are irreducible and which can therefore serve as the defining function for GF($4 = 2^2$). There are not that many to try since the only allowed coefficients are the 0 and 1 of Z_2 :

x^2 = (x)(x)
$1 + x^2$ = (1+x)(1+x) // since 2x=0
$x + x^2$ = (x)(1+x)
$1 + x + x^2$ (4.13)

There are no more. We have shown that all these choices factor in R except the last one. Therefore, there is exactly one defining polynomial for GF(4), and it is $f(x) = 1 + x + x^2 = 111$. In the general case $q = p^m$, there are usually more than one possible irreducible f(x).

Now that we know f(x), we can build the multiplication table for GF(4). This is in fact the complete table, but we shall derive just the bolded red element as illustration. The full table is derived in Chapter 6 (e).

•	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	**01**	10

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

(4.14)

The table above is a shorthand 2-tuple version of the table below which actually shows the polynomials implied by the m-tuple notation. To its right, we show another version where 00=0, 10=1, 01=α, 11=β)

•	0	1	x	1+x
0	0	0	0	0
1	0	1	x	1+x
x	0	x	1+x	1
1+x	0	1+x	1	x

•	0	1	α	β
0	0	0	0	0
1	0	1	α	β
α	0	α	β	1
β	0	β	1	α

(4.15)

As an example of the procedure described above, we will compute the bolded entry in the table (4.14),

$11 \bullet 10 = (1+x)(x) = x + x^2$ // we are not done yet....

Comment: Since $q = p^m = 2^2$ is such a simple case, we don't have to use the complicated formula (4.11) to multiply two polynomials, we can just do it all quickly by hand.

The result $x + x^2$ is improper, having degree ≥ 2, so we divide it by $f(x) = 1 + x + x^2$:

$$\begin{array}{r|l} & 1 \quad . \\ \hline x^2 + x + 1 & x^2 + x \\ & x^2 + x + 1 \\ \hline & 1 \end{array}$$

So in this example, $C(x) = x + x^2$, and the remainder is $c(x) = 1 = [01]$. Thus, we get **01** in the • table. In just this way one can derive the entire table, but an easier way appears in Chapter 6 (e).

By the way, notice that the identity **1** = 01 appears in each row (except the first) of the • table, which means that every non-zero element has an inverse. Recall that the modulo-4 table in (2.6) failed us on this point. GF(4) really is a *field* in which every element has an inverse; Z_4 = {mod-4,+,•} is only a lowly ring.

(c) Extending polynomials from ground field GF(p) to extension field GF(q)

Earlier we considered the polynomial $f(x) = 1 + x^2$. Think of the reals as a ground field, and the complex numbers as an extension field. You can start out thinking of the polynomial as having coefficients in the reals, but you can also think of these coefficients as happening to lie on the real axis of the complex

number plane. In some sense, then, you are extending the interpretation of your polynomial to a larger world. As noted earlier, a crucial difference is that in the reals, this f(x) cannot be factored and is therefore irreducible, but when extended to the complex numbers, suddenly f(x) is no longer irreducible and can be factored into (x + i)(x - i).

The reason that extending the polynomial from the reals to the complex numbers makes sense is that the reals form a subfield of the field of complex numbers. In similar fashion, GF(p) is a subfield of GF(q). Thus, we can take any polynomial having coefficients in GF(p) and then think of this polynomial as being extended so these coefficients are now elements of the larger field GF(q). The polynomial "looks" the same before and after this extension. However, it may turn out that the polynomial is more factorable in GF(q) than it is in GF(p).

Note that in general you cannot go the other direction. If you have a polynomial with general coefficients in GF(q), how can you write it in terms of only GF(p) elements? For example, the polynomial over the complex number field f(x) = (x + i) cannot be written over the real number field.

(d) Cyclic Subgroups and GF(q)

Much of the structure of the Galois Field GF(q) can be brought to light by analyzing its cyclic subgroups under the multiplicative operation •. That is the main subject of this section. A *cyclic subgroup* was defined in Chapter 1 (b) 3, and a few facts about it derived there.

Many of the "facts" below apply to cyclic subgroups in general, or in some cases, to commutative groups in general. However, we will be specific and claim our facts only with respect to the field GF(q).

There are 16 Facts presented in this section, all with proofs. The reader is encouraged to read the proofs, since they serve as checks on the previous material. The main conclusions of this section are presented as Big Theorems 1 and 2. This is a long and somewhat tedious development.

Notation: In the following set of fact developments, the term "cyclic subgroup" refers exclusively to a cyclic subgroup of the group {GF(q) - 0,•}. This means the multiplicative group under • of GF(q) where we have excluded the 0 element. The group {GF(q) - 0,•} thus has order q-1.

A divides B (sometimes written **A|B**) means that there is no remainder when A is divided into B. It means that B is a multiple of A.

If A and B are numbers, it means that B= Ak for some integer k = 1,2,3...

If A and B are polynomials, it means B(x) = A(x)k(x) for some polynomial k(x).

Fact 1: The order n of any cyclic subgroup of {GF(q) - 0,•} divides q-1. (4.16)

Proof: The order of *any* subgroup H divides the order of a group G containing it. This follows from the coset decomposition discussed in Chapter 1 (b) 1.

Fact 2: Start with any α in {GF(q) - 0 ,•}. Start forming a list by taking powers of α:

$\{ 1 = \alpha^0 , \alpha = \alpha^1, \alpha^2, \alpha^3, ... \}$

We claim quite a few things in this one "fact" (proofs to follow):

(a) there exists an integer n such that α^{n-1} is the *last distinct element* of this list. Thus, the complete list has n distinct elements as follows:

$\{ 1, \alpha, \alpha^2, ... \alpha^{n-1} \}$

(b) $\alpha^n = 1$, which repeats the first element of the list (this is the *first* repeat)

(c) all subsequent powers repeat elements of the list.

(d) The list of n distinct elements forms a *cyclic subgroup* of {GF(q) - 0 ,•} of order n.
We can denote this as:

$(\alpha, n, \bullet) =$ cyclic subgroup of {GF(q) - 0 ,•} = (generator, order , • operation)

(e) This integer n is called the **order of α**; α is said to be **of order n**.
[If α is of *order* n, it generates a cyclic subgroup within {GF(q) - 0 ,•} of *order* n.]

(f) The fact that α has order n does *not* imply that other elements of the subgroup have order n.

(g) α is called a **generator** of the cyclic subgroup.

(h) Any other list element $\beta = \alpha^k$ which, upon taking powers 0 through n-1, yields the same list as α (with a possible reordering of the elements) is also regarded as a *generator*. Such an alternate generator has order n if α has order n.

(i) If $\beta^n = 1$, it does *not* follow that n is the order of β.

(j) We have shown in (1.9a) that *any* element of {GF(q) - 0 ,•} lies in *some* cyclic subgroup of *some* order. For example, α lies in (α, n,•). Another example is that 1 lies in (1,1,•).

(k) The **order of α** is the smallest integer n such that $\alpha^n = 1$.

(ℓ) The **order of α** is a divisor of q-1, which is to say, (q-1)/n = integer.

(4.17)

Proof:

(a),(b) All elements in the list are contained in {GF(q) - 0 ,•} which is a finite set. Therefore, there must be a first power where an element of the sequence repeats some earlier element. Suppose this power is n, so that α^{n-1} is the last distinct item on the list, and suppose $\alpha^n = \alpha^k$, some earlier item on the list. If k > 0, then we would have $\alpha^{n-1} = \alpha^{k-1}$, which says that the previous power also repeated an item on the list, which is a contradiction. Thus, we must have k=0, so $\alpha^n = 1$.

(c) As we build subsequent powers, the list repeats: $\alpha^n = 1$, $\alpha^{n+1} = \alpha$, $\alpha^{n+2} = \alpha^2$, and so on.

(f) Suppose the order of α is 4 so list is $\{1, \alpha, \alpha^2, \alpha^3\}$ with $\alpha^4 = 1$. Consider $\beta = \alpha^2$. The order of β is 2, not 4, since the β list is $\{1, \beta = \alpha^2, \beta^2 = \alpha^4 = 1\}$ Notice that 2 divides 4.

(h) Such alternate generators may or may not exist;

(i) Suppose the order of β is 5, so that $\beta^5 = 1$. It is then certainly true that $\beta^{10} = 1$, but this clearly does not imply that β has order 10.

(k) By the definition of n as the order of α above, α^n is the *first* repeat element. There is no earlier repeat element, so there is no smaller subgroup within the one discussed that is generated by α. So for this n which has $\alpha^n = 1$, we know that there is no smaller integer N < n such that $\alpha^N = 1$.

(**ℓ**) This thing (α, n, •) of order n is a cyclic subgroup of {GF(q) - 0 ,•} which has order q-1. According to (1.8), the order of any subgroup divides evenly into the order of the parent group.

Fact 3: Let g ∈ {GF(q) - 0 ,•}. If "g has order n" , then $g^n = 1$ and n is the least integer for which $g^n = 1$. (4.18)

Proof: This follows from the definition of the order of g. We just want to stress the fact.

Fact 4: Let g ∈ {GF(q) - 0 ,•}. If $g^k = 1$, the "order of g" divides k. (4.19)

Proof: Let n be the order of g. We know from the definition that n is the smallest power of g such that $g^n = 1$. Consider some higher power that is not a multiple of n: k = qn + r where 0 < r < n. Then $g^k = g^{qn}\, g^r = g^r$. But we know that $g^r \neq 1$ since 0 < r < n. Thus, if k is not a multiple of n, $g^k \neq 1$. Therefore, if $g^k = 1$, k must be a multiple of n, and n divides k. Look back at the $\beta^{10} = 1$ comment.

Fact 5: If β is any element of the cyclic subgroup (α,n,•), then $\beta^n = 1$. (4.20)

Proof: Since β ∈ (α,n,•), we know that $\beta = \alpha^k$ for some k, and $\alpha^n = 1$. Thus $\beta^n = \alpha^{nk} = 1^k = 1$.

Fact 6: If β is any element of (α,n,•), then the order of β divides n, the order of α. More specifically, we claim that [order of β] = n/GCD(k,n) where $\beta = \alpha^k$. (4.21)

Proof: Since β ∈ (α,n,•), we know that $\beta = \alpha^k$ for some k, and $\alpha^n = 1$. Then as we enumerate the subgroup generated by β, we get { 1, β, β^2, β^3 ... } = { 1, α^k , α^{2k} , α^{3k} ...}. If the order of β is m, then $\beta^m = 1$, and our little cyclic subgroup can be called (β,m,•). This implies that m is the smallest integer such that $\beta^m = (\alpha^k)^m = 1$ or $\alpha^{(km)} = 1$. We now apply Fact 4 with g → α and k → km and "order of g" is "order of α" which is n. Thus, n divides km, so km = nN for some integer N. From the Lemma below, we conclude that m = n/GCD(k,n) so that m = [order of β] = n/GCD(k,n). **QED**

Lemma: The smallest integer m such that km = nN is m = n/GCD(k,n). (4.22)

Proof: In (G.21) it is shown that, for ax = by, the smallest-x solution is x = b/d where d = GCD(a,b). We apply that here with a = k, x = m, b = n and y = N to find that the smallest-m solution is m = n/d where d = GCD(k,n).

We now develop some facts which involve polynomials. These polynomials are not restricted to the special ring R of polynomials discussed in Chapter 3 (a). That is, the coefficients are not necessarily elements of Z_p.

Element a is a **root** of polynomial f(x) means that f(a) = 0. (4.23)

Fact 7: (The Factor Theorem) Polynomial f(x) can be written as f(x) = (x-a)q(x) for some polynomial q(x) if and only if a is a root of f(x). (4.24)

Proof:

($\Leftarrow$) Assume a is a root of f(x). By the Division Algorithm, we can expand f(x) = (x-a)q(x) + r(x) where the remainder polynomial r(x) has degree less than (x-a), which means it has degree 0, which means r(x) = constant K. Since f(a) = 0, we see that r(x) = K = 0.

($\Rightarrow$) Assume f(x) = (x-a)q(x). Since q(x) is a polynomial, q(a) is finite, so f(a) = 0.

Fact 8: The n elements a_i of the cyclic subgroup (α,n,•) are the n roots of the polynomial $1 - x^n$. This polynomial can be fully factored into a product of n linear factors $(x-a_i)$ where the a_i are the roots,

$$x^n - 1 = (x - a_1)\bullet (x - a_2)\bullet (x - a_3)\bullet \ldots (x - a_n) . \quad (4.25)$$

Proof: We know from Fact 5 that $(a_i)^n = 1$ for all n elements in the cyclic subgroup. Thus, these n elements are all roots of $f(x) = 1 - x^n$. According to Fact 7, each such root results in a factor $(x-a_i)$ in f(x). Since this polynomial has at most n roots and we have found them all, it fully factors as claimed.

Fact 9: Consider two cyclic subgroups A = (α,n,•) and B = (β,m,•) of some group.
If n=km for some integer k , then A contains B. (4.26)

Corollary: If the orders of two cyclic subgroups are the same, then the subgroups are the same. (4.27)

Corollary : There can be at most one cyclic subgroup of any given order. (4.28)

Proof: Let b $\in$ (β,m,•). From Fact 5 , $b^m = 1$. Raise both sides of this last equation to power k to get $b^{mk} = 1$. If mk=n, we get $b^n = 1$ Thus, b is a root of $f(x) = 1 - x^n$. Since A is a cyclic subgroup of order n, we know by Fact 8 that the n roots of $f(x) = 1 - x^n$ are all elements of A. Since b is a root of f(x), b must equal one of the elements of a. Thus, b is contained in A. Since this is true for any b in B, this means all of B is contained in A. Corollaries follow from k=1.

The **exponent** e is the order of the *largest* cyclic subgroup E in {GF(q) - 0}, •}.

Heads Up: This is one of several competing meanings for the word "exponent" in Galois Theory. There are after all quite a few exponents! This particular one is going to end up being the same as q-1 according to Big Theorem 1 below, so its interest does not persist.

Fact 10: The order of any cyclic subgroup of {GF(q)-0,•} divides the exponent e. (4.29)

Proof: This Fact 10 looks innocent enough, but it is the real meat and potatoes of this entire section. We are assuming there is some maximal cyclic subgroup called E of order e. As far as we know at this point in the development, E can be less than all of {GF(q)-0,•}. Thus, if there is some cyclic subgroup called G, we do not know that G is contained in E. If we knew that all such G were contained in E, we would immediately know Fact 10, since the order of any subgroup G divides the order of the containing subgroup E.

So at this point, as far as we know, G may or may not be contained within E. Thus, we need to find a more general proof of Fact 10. Although this proof is somewhat complicated, it uses only Facts 3, 4 and 5, and the basic properties of integers. The proof is given in **Appendix A.**

Based on Fact 10, we will quickly arrive at Big Theorem 1 below which says that {GF(q) - 0, •} is cyclic. This implies that E must indeed be all of {GF(q) - 0, •}. In retrospect, Fact 10 seems less impressive. But one must remember that the information that E = {GF(q) - 0, •} is not available for the proof of Fact 10.

Big Theorem 1: The entire group {GF(q) - 0, •} is cyclic (so exponent e = q-1). (4.30)

Proof: Consider *any* element α of {GF(q) - 0, •}. From Fact 2(j), α must lie in some cyclic subgroup of some order n. From Fact 5 we know α is a solution of $1 = x^n$. From Fact 10 we know that e, the order of the largest cyclic subgroup, must be a multiple of n, we can write e = kn. Raise both sides of $1 = x^n$ to the k^{th} power then to get $1 = x^e$. Thus, we have shown that *any* element α of {GF(q) - 0, •} is a root of $1 - x^e$.

But $1 - x^e$ can have at most e roots. Since all q-1 elements of {GF(q) - 0, •} are roots (like α), we know that $q-1 \leq e$. On the other hand, it is clear that $e \leq q-1$ since e is the order of a cyclic subgroup contained within {GF(q) - 0, •} . The only possible solution to this dilemma is e = q-1. Thus, the size of the largest cyclic subgroup *equals* the size of the whole group. Thus, the whole group is cyclic.

The structure of this proof and that of Fact 10 in Appendix A were both taken from Bobrow and Arbib.

Fact 11: There exists at least one generator α for {GF(q) - 0, •}. In terms of it, the entire group GF(q) can be enumerated as follows:

$$\{ 0, 1, \alpha, \alpha^2, \alpha^3, \ldots\ldots \alpha^{q-2} \} \qquad \alpha^{q-1} = 1 \qquad \alpha^q = \alpha \tag{4.31}$$

Proof: Since {GF(q) - 0, •} is cyclic and is of order q-1, it must have a generator α. Then we add back the 0 element to get GF(q).

Any generator of {GF(q) - 0, •} is called a **primitive element** of GF(q). Thus, the α shown above is a primitive element of GF(q). [Note that a primitive element is not a generator of { GF(q), • }. No power of some $\alpha \neq 0$ is going to produce the zero element 0.]

Fact 12: If α is a primitive element of GF(q) then :

(a) order of (α^k) = (q-1)/GCD(k,q-1)

(b) if GCD(k,q-1) = 1, then α^k is also a primitive element of GF(q)

(c) if GCD(k,q-1) ≠ 1, then α^k is not primitive element of GF(q) (4.32)

Proof: Part (a) is just Fact 6 (4.21) applied to the cyclic group {GF(q)-1, • }. For Parts (b) and (c):

If GCD(k,q-1) = 1, then (a) says order of (α^k) = (q-1) and by the definition of a primitive element of GF(q) we conclude that α^k is primitive. Conversely:

If GCD(k,q-1) ≠ 1, then GCD(k,q-1) = N > 1, and order of α^k = (q-1)/N, so in this case α^k is not a primitive element.

Fact 13: If q-1 is prime, all elements of {GF(q) - 0 - 1} are primitive elements of GF(q). (4.33)

Proof: If q-1 is prime, then *any* k [in the range 0<k<q-1] and q-1 are relatively prime, so GCD(k,q-1) = 1. Thus, from Fact 12, *any* power α^k of a primitive element α with 0<k<q-1 is a primitive element.

Example: GF(4 = 2^2) has q-1 = 3 = prime. The elements are {0,1,α,β}. Either α or β can serve as a generator, they are both primitive elements. Thus, $\beta = \alpha^2$ and $\alpha = \beta^2$. In terms of m-tuple notation and the GF(4) multiplication table given in (4.14), we know that:

If α = 10, then $\beta = \alpha^2 = 10 \bullet 10 = 11 = \beta$

If β = 11, then $\alpha = \beta^2 = 11 \bullet 11 = 10 = \alpha$

Example: GF(8 = 2^3) has q-1 = 7 = prime:

$\{ 0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6 \}$ $\alpha^7 = 1$

This field has 6 distinct primitive elements. Here they are: $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$.

Example: GF(16 = 2^4) has q-1 = 15 ≠ prime. If α is a primitive element, then $\beta = \alpha^3$ is not a primitive element. In fact, the order of β's subgroup is 5, not 15: [From Fact 6, 5 = 15/GCD(3,15) = 15/3]

GF(16) = $\{ 0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}$ $\alpha^{15} = 1$

$\beta = \alpha^3$ $\beta^2 = \alpha^6$ $\beta^3 = \alpha^9$ $\beta^3 = \alpha^{12}$ $\beta^5 = \alpha^{15} = 1$

$\{ 1, \beta, \beta^2, \beta^3, \beta^3 \}$ = cyclic subgroup of {GF(16) - 0} of order 5 $\Rightarrow$ β not a primitive element

Big Theorem 2: We shall state this theorem in five equivalent ways: (4.34)

1) The q-1 elements of {GF(q) - 0, •} are the q-1 roots of the polynomial $A(x) \equiv x^{q-1} - 1$
2) The q elements of GF(q) are the q roots of $B(x) \equiv x^q - x = xA(x)$
3) If α is any element of GF(q), then B(α) = 0.
4) If α is any non-zero element of GF(q) , then $\alpha^q = \alpha$ and $\alpha^{q-1} = 1$.
5) The polynomial $x^q - x$ can be fully factored into q linear factors of the form $(x - a_i)$. Each factor contains one of the elements of GF(q). There is one factor for each root. Thus:

$$(x^q - x) = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_q) .$$

We might as well pick out the field elements that we know must be present, namely **1** and **0**. Letting these be a_1 and a_q , we can rewrite the above factorization as:

$$(x^q - x) = (x - \mathbf{1})\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_{q-1})\bullet (x - \mathbf{0}) .$$

Dividing by x gives

$$(x^{q-1} - 1) = (x - \mathbf{1})\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_{q-1}) .$$

Proof: These results are all trivial once we know that {GF(q) - 0, •} is cyclic. Item #1 is true for any cyclic subgroup, as shown in Fact 8. Item #2 is trivial since we have just added a factor of x to account for the 0 element of GF(q). Item #3 and #4 are a restatements of item #2. Item #5 follows because we know that $x^q - x$ has at most q roots, and we have identified all q of them in item #3, so we are just writing them out, as argued in Fact 8. Each form of the above theorem stresses a certain point.

Comment: The polynomial $x^q - x$ has coefficients in Z_p = GF(p) and in general cannot be factored into a product of factors $(x-a_i)$ where the roots a_i are all in GF(p). But item 5 above says that $x^q - x$ *can* be fully factored if we allow the roots a_i to lie in GF(q). Sometimes one then says that GF(q) is an **extension** of GF(p). Because this allows the above factorization in GF(q), GF(q) is referred to as a **splitting field** of the polynomial $x^q - x$ over GF(p).

Fact 14: All facts above apply to GF(p) since this is a special case of GF(q). (4.35)

Corollary 1: The group {GF(p) - 0, •} is cyclic, as was claimed without proof in (2.2). (4.36)

Corollary 2: Big Theorems 1 and 2 apply to GF(p), replace q with p everywhere. (4.37)

Fact 15: {GF(p) - 0, •} is a cyclic subgroup, order p-1, of {GF(q) - 0, •} which has order q-1. (4.38)

Proof: If group H is a subset of group G, then H is by definition a subgroup of G. We know that {GF(p) - 0, •} is a group, and we know it is a subset of {GF(q) - 0, •}, so it is a subgroup. We also know that {GF(p) - 0, •} is cyclic, as just stated above. Thus, it is a cyclic subgroup.

Fact 16: If $\alpha^p = \alpha$ for some element α in GF(q), then α is an element of GF(p). (4.39)

Proof: If $\alpha = 0$, the result is obvious, since 0 is in GF(p). Otherwise, we have $\alpha^{p-1} = 1$, so that α is a root of $x^{p-1}-1$. Since {GF(p) - 0, •} is a a cyclic subgroup of {GF(q) - 0, •} of order p-1, we know from Fact 8 that all p-1 roots of $x^{p-1}-1$ are elements of {GF(p) - 0, •}. Since α is a root of $x^{p-1}-1$, it must be one of the elements of {GF(p) - 0, •}. Thus, whether or not $\alpha=0$, we conclude that $\alpha^p = \alpha \Rightarrow \alpha \in$ GF(p).

(e) An example of root factorization in GF(2^2)

In (4.10) and (4.14) we displayed the + and • tables for GF(2^2). Here they are again,

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

•	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

(4.40)

We know that 00 = **0**, and 01 = **1**. According to Big Theorem 2 (4.34), we should have:

$(x^4 - x) = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)$

$= (x - 00)\bullet(x - 01)\bullet(x - 10)\bullet(x - 11).$?

Let's check to see if it works. We can divide out the x to get:

$(x^3 - 1) = (x - 01)\bullet(x - 10)\bullet(x - 11).$?

But we can factor $(x^3 - 1) = (x - 1)\bullet(x^2 + x + 1)$, so it remains only to show that

$(x^2 + x + 1) = (x - 10)\bullet(x - 11)$. ?

For Z_2 we know that + and – are the same, so we can rewrite the above as

$(x^2 + x + 1) = (x + 10)\bullet(x + 11) = x^2 + (10 + 11)\,x + 10\bullet 11$. ?

Looking up in the tables, we get 10 + 11 = 01 which is the 1 element, and 10•11 = 01 is the 1 element as well. Thus we can erase all the question marks. We have now seen a specific example, GF(2^2), where the polynomial $(x^4 - x)$ fully factors into a product of four linear factors of the form (x-a), one for each element of GF(2^2).

Notice that $(x^2 + x + 1)$ factors in GF(2^2) into (x - 10)•(x - 11). However, in GF(2) -- the ground field -- we cannot factor $(x^2 + x + 1)$. In fact, this is f(x), the irreducible polynomial which defines GF(2^2), as was shown in (4.13).

(f) Two ways to label elements of GF(q) in the + and • tables

One labeling method is to use the m-tuples which consist of m Z_p integers, where m is the degree of f(x). Each m-tuple stands for a remainder polynomial in the residue class ring which we constructed to represent GF(q). In this method, the + and • tables all contain m-tuple entries. We saw back in section (b) how the + table is trivial to construct in terms of m-tuples -- you just do modulo-p addition on each corresponding integer of a pair of m-tuples. However, the • table entry took much more work.

A second labeling method is to find a generator α of GF(q)-0 = a primitive element α of GF(q), and list the elements as powers of α. In this method, each + and • table entry is a power of α. In this case, the • table is trivial to construct. You just multiply powers and use the fact that $\alpha^{q-1} = 1$. However, now the + table is non-trivial to construct. Here, you have to convert the powers of α to m-tuples, add, then convert back.

The two labeling methods are like having two different bases, or coordinate systems. In the m-tuple basis, addition is easy and multiplication is hard. In the α basis, just the opposite is true.

Here are the GF(2^2) tables in both bases:

1. Here is the m-tuple basis, where $f(x) = 1 + x + x^2$. Recall that this f(x) is a viable irreducible polynomial with which we can construct our polynomial remainder representation of GF(4). This f(x) had to be irreducible so that the polynomial remainder representation was a field, not just a ring, so that it could in fact represent GF(4) which is a field. Also, we need to know what f(x) is in order to compute the multiplication table elements in the case that products of remainders are improper and have to be divided by f(x) to obtain proper remainders.

The + table can be verified by inspection, but the • table cannot:

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

•	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

(4.40)

2. Here is the α basis for α = 10 = x, and $\alpha^3 = 1$ (since α is a generator of GF(4)-0). The • table can be verified by inspection, but the + table cannot. Note that $f(x) = 1 + x + x^2 = 1 + \alpha + \alpha^2$.

+	0	1	α	α^2
0	0	1	α	α^2
1	1	0	α^2	α
α	α	α^2	0	1
α^2	α^2	α	1	0

•	0	1	α	α^2
0	0	0	0	0
1	0	1	α	α^2
α	0	α	α^2	1
α^2	0	α^2	1	α

(4.41)

(g) Selected Facts About GF(q)

We arrange this list in the same order as the list for GF(p) in Chapter 2 (c).

Fact: The q-1 non-zero elements of GF(q) form a cyclic group under • of order q-1. The generators of this cyclic group are called primitive elements of GF(q). Thus, GF(q) can be enumerated as :

$\{ \mathbf{0}, \mathbf{1}, \alpha, \alpha^2, \alpha^3, \ldots\ldots \alpha^{q-2} \}$ $\alpha^{q-1} = 1$ $\alpha^q = \alpha$ (4.42)

Saying that the group is cyclic implies that at least one primitive element α exists. If q-1 is prime, then all elements of GF(q) other than 0 and 1 are primitive elements (Fact 13).

Proof: See Big Theorem 1, (4.30) and also (4.31).

Fact: For any element β of GF(q) , $\beta^q = \beta$ (β is a root of $\beta^q - \beta$) (4.43)

Proof: See Big Theorem 2 item (4), (4.34).

Fact: For any element **β** of GF($q = p^m$), $p\boldsymbol{\beta} = \mathbf{0}$. (4.44)

Proof: This is just Fact (4.5).

Fact: $(\mathbf{a} + \mathbf{b})^p = \mathbf{a}^p + \mathbf{b}^p$ when **a**,**b** are elements of GF(q). (4.46)

Proof: Same as the proof of (2.7), except here we say that the binomial coefficients of the cross terms vanish because $p\boldsymbol{\beta} = \mathbf{0}$ for any element **β** of GF(q).

Fact: $(\mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d} + \ldots)^p = \mathbf{a}^p + \mathbf{b}^p + \mathbf{c}^p + \mathbf{d}^p + \ldots$ for a,b,c,d... in GF(q). (4.47)

Proof: Just apply (4.46) multiple times, as in the proof of (2.8).

Chapter 5: The Minimum Polynomial of an element of GF(q)

In this chapter we continue to accumulate information about finite fields. As noted earlier, finite fields exist only for certain values of the order q. These values are of the form $q = p^m$ where p is a prime and m = 0,1,2,3..... The finite fields are denoted GF(q).

Some books refer to the prime number p as the **characteristic** of GF(p^m) and m as the **degree**.

(a) The Minimum Polynomial m(x) of an element α in GF(q).

In Chapter 4's Big Theorem 2 (4.34) it was demonstrated that one could fully factor the polynomial $x^{q-1} - 1$ into a product of q-1 first-degree factors, each of which vanishes at a non-zero element of GF(q):

$$x^{q-1} - 1 = (x - \mathbf{1}) \bullet (x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_{q-1}) . \qquad (5.1)$$

We now keep sharply attuned to the distinction between coefficients in GF(p) and coefficients in GF(q). The polynomial $(x-a_4)$ has coefficients in GF(q), since a_4 is an element of GF(q). However, the polynomials (x-1) and $(x^{q-1}-1)$ have coefficients entirely in GF(p) = Z_p, which we know is a subfield of GF(q). For example, -1 = p-1 in Z_p. Recall that GF(p) is the ground field, and G(q) is the extension field. We shall refer to coefficients being in GF(p), but we have in mind such coefficients being the integers mod-p which comprise the elements of Z_p which is isomorphic to GF(p).

We pose this question: Is it possible to take some *subset* of factors in (5.1) such that, when the factors are multiplied out, one ends up with a polynomial whose coefficients lie entirely in GF(p)? We defer an explanation of *why* this might be desirable to Chapter 8 (j). Clearly such a polynomial would have degree less than q-1 if it had less than all the factors shown in (5.1).

There is one obvious such polynomial of degree q-2. Divide both sides of (5.1) by (x-1) to get:

$$1 + x + x^2 + ... + x^{q-2} = (x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_{q-1}) . \qquad (5.2)$$

The polynomial on the left, which is $(x^{q-1}-1)/(x-1)$, does indeed have coefficients all in GF(p). But is there a still *smaller* grouping that works -- some subset of the above factors which produces a polynomial with coefficients in GF(p)? And assuming there is, we might ask: what is the smallest such polynomial -- the one with the fewest factors? Could there be several such "smallest" polynomials?

One obvious way to find a smallest such polynomial is to make a huge list by multiplying out factors $(x-a_k)$ in all possible ways, then examine the resulting polynomials to see which ones, if any, have coefficients only in GF(p). This the brute force method, and it requires the expansion tools of Chapter 6 (d). We could then pick a particular element $\alpha = a_i$ of GF(q) and ask: which of these polynomials with coefficients in GF(p) contain $(x - \alpha)$ as a factor? We could finally examine this smaller group and select the polynomial (or polynomials) having the lowest degree, i.e., having the least number of other factors. What one arrives at by this method is called a minimal polynomial of α.

To recap: a **minimum polynomial** for some element α of GF(q) is a polynomial which is the product of the smallest set of factors $(x - a_k)$ which contains the factor $(x - \alpha)$, and which (when multiplied out) has coefficients only in GF(p). Denote this minimum polynomial for α by the name **m(x)**. Thus, α is a root of m(x) so $m(\alpha) = 0$. Remember that in general α and the a_i are themselves *not* in GF(p). (5.3)

Note: Any minimum polynomial is an element of the ring R defined in Chapter 3 (a) which, recall, was the set of polynomials with coefficients in Z_p.

Example: The minimum polynomial of the 0 element of GF(q) is $m(x) = (x - 0) = x$.

Comment: One could and probably should write $m_\alpha(x)$ to show that m(x) is specific to some α. We don't do this because: (1) more than one α can have the same m(x); (2) we shall be putting subscripts on m(x) in Chapter 8 which have a different though related meaning.

If the coefficient of the highest power of a polynomial is 1, the polynomial is said to be **monic**. As defined above, any minimum polynomial m(x) is always monic, being a product of (x-a) factors.

For GF(p) with p>2, one can have coefficients other than 1 and 0. In this case, one can construct several superficially different polynomials which have the same roots, but they differ just by a scale factor. The monic requirement of m(x) then removes these superficial duplications. For example, 2x-2 is not really distinct from x-1 in terms of its roots.

It turns out that the minimal polynomial of α is unique, but to show that we first need the following.

Fact 1: If p(x) lies in R and $p(\alpha) = 0$, then $p(x) = q(x)m(x)$ of α . In other words, p(x) having a root α must be a multiple of any minimum polynomial m(x) of α. (5.4)

Proof: Expand $p(x) = q(x)m(x) + r(x)$, where m(x) is a minimum polynomial of α, and where the degree of r(x) is less than that of m(x)-- this is the Division Algorithm (3.6). If $p(\alpha) = 0$, then $r(\alpha) = 0$, since $m(\alpha) = 0$. But if $r(\alpha) = 0$, then m(x) must not be a minimal polynomial because r(x) is a polynomial of lesser degree than m(x), and has α as a root. In other words, then r(x) must be the real minimum polynomial, not m(x). This contradicts the starting assumption, so we must have $r(x) = 0$ and then we end up with $p(x) = q(x)m(x)$, so p(x) is a "multiple" of m(x).

Fact 2: The minimum polynomial m(x) for some element α of GF(q) has these properties:

(a) A unique m(x) exists for any given α
(b) $m(\alpha) = 0$
(c) m(x) is irreducible in GF(p)
(d) m(x) divides evenly into the polynomial $x^{q-1} - 1$ (5.5)

Proof: (a) Certainly we know that a minimum polynomial m(x) for any α must exist. If nothing smaller works out, we know that the polynomial shown above, $x^{q-1} - 1$, is the final candidate.

Suppose there are two different minimum polynomials $m_1(x)$ and m(x) for α. Then $m_1(\alpha) = 0$ and $m(\alpha) = 0$. Both $m_1(x)$ and m(x) have the same degree r (since they are both minimum polynomials) and are both monic. By Fact 1 of (5.4) we can write $m_1(x) = q(x)m(x)$. Thus we have,

(x^r + lower terms) = q(x) (x^r + other lower terms) .

In this equation, if q(x) were of degree s > 0, the right side would generate terms of degree s+r which are not present on the left. Thus q(x) must be of degree 0 -- it is a constant. Only q = 1 can match the coefficients of x^r on both sides, so q(x) = 1 and therefore $m_1(x) = m(x)$ so there cannot be two different minimum polynomials for an element α of GF(q).

(b) That m(α) = 0 follows by definition since m(x) contains (x-α) as a factor.

(c) Recall (3.12) item 3 on reducible polynomials. If m(x) were reducible in R, then we could write m(x) = a(x)b(x) where both a(x) and b(x) lie in R and neither a(x) nor b(x) is a constant. Then, since m(α) = 0, we know that either a(α) = 0 or b(α) = 0 (or both). In this case, we would take whichever one of these vanishes at α, and then that one would be a minimum polynomial for α of lower degree than m(x). But this is a contradiction since m(x) is supposed to be the minimum polynomial. Thus, m(x) must be irreducible.

(d) Since m(x) contains a subset of the factors than make up x^{q-1} - 1, it must divide evenly into x^{q-1} - 1, see (5.1). The quotient is then all the factors of x^{q-1} - 1 which are not contained in m(x).

Fact 3: If monic f(x) in R is irreducible in R and f(α) = 0 for some α in GF(q), then f(x) is the minimum polynomial m(x) of α. (5.6)

Proof: If f(x) were reducible in R, one could write f(x) = q(x) d(x) where both q(x) and d(x) lie in R and neither is a constant. In this case, since f(α) = 0, we have q(α) = 0, d(α) = 0, or both. In any case, we know that f(x) is not the minimum polynomial for α since at least one of q(x), d(x) is of lower degree and has root α and so *it* would then be the minimum polynomial. On the other hand, if f(x) were irreducible in R, then one could *not* write f(x) = q(x)d(x) for non-constant q(x) and d(x) in R, and then we cannot find a polynomial of degree lower than f(x) which has α as a root. Then f(x) must be the lowest degree polynomial which has α as a root, and then f(x) is the minimum polynomial for α.

Corollary: There may exist monic irreducible polynomials which are not minimum polynomials for any α in GF(q). An example is given later in Chapter 6 (d) 3.

(b) Primitive Polynomials and the Period of m(x)

If α is a primitive element of GF(q), meaning α is a generator of cyclic {GF(q) - 0, •}, then the minimum polynomial m(x) of α is called a **primitive polynomial** of GF(q). Conversely, if m(x) is a primitive polynomial for GF(q) for α, then α is a primitive element of GF(q). (5.7)

Corollary: If monic f(x) in R is irreducible over R and has some primitive element α of GF(q) as a root, then f(x) is a primitive polynomial of GF(q). (5.8)

Proof: This is a restatement of Fact 3 (5.6) for α = a primitive element of GF(q).

From Fact 2(d) we know that a minimal polynomial m(x) for any α in GF(q) divides $x^{q-1} - 1$. It may be possible that m(x) also divides x^n-1 for some n < q-1. The *smallest* such power n is called **the period of m(x)**. (5.9)

Fact 4: A primitive polynomial p(x) of GF(q) must have the maximal period q-1. (5.10)

Proof: We know that p(x) is a minimal polynomial for some primitive element α of GF(q). We know by Fact 2(d) that p(x) divides $x^n - 1$ with n = q-1. Is there some smaller n?

Assume there exists some n < q-1 such that p(x) divides $x^n - 1$. Then $x^n - 1 = p(x)q(x)$ for some q(x). Since p(α) = 0, we find $\alpha^n = 1$ for n < q-1. But by the definition of a primitive element, the order of α is q-1, and this means that n = q-1 is the smallest integer such that $\alpha^n = 1$. Thus we have a contradiction, so there is no n < q-1 that works. The smallest n that works is then q-1, so this is the period.

Fact 5: If the period of some minimum polynomial m(x) of α is q-1, then α must be a primitive element of GF(q), and then by definition m(x) is a primitive polynomial of GF(q). (5.11)

Proof: If α is not primitive, then α has some period n < q-1 by Fact 4. Thus, we can consider (α,n,•), the cyclic subgroup generated by α. According to Fact 8 (4.25), the n elements of this subgroup are the n roots of $x^n -1$, and we can factor as follows:

$$x^n -1 = (x - a_1)\bullet (x - a_2)\bullet (x - a_3)\bullet \ldots (x - a_n), \qquad (4.25)$$

where one of the a_i is our element α. Since $x^n -1$ has coefficients in GF(p), it is a candidate for the minimum polynomial m(x) of α. The other possibility is that some subset of the factors shown here forms m(x). In either case, m(x) divides $x^n - 1$, so we have period n < q-1. But by hypothesis, the period of m(x) is supposed to be q-1. Thus, α must be a primitive element of GF(q), and by definition, m(x) is then a primitive polynomial of GF(q).

(c) Formula for the Minimum Polynomial m(x) of α : Conjugate Sets

We would now like to develop an explicit formula for the minimum polynomial m(x) of α. As a very strong hint toward this end, we state and prove the following interesting fact:

Lemma 6: If f(x) is in R, then $f(x^p) = [f(x)]^p$. (5.12)

Proof: Just write out both sides and make use of the expansion formula (2.11). The RHS becomes

$$[f(x)]^p = [\, a + bx + cx^2 + \ldots \,]^p = (a)^p + (bx)^p + (cx^2)^p + \ldots = a^p + b^p(x^p) + c^p(x^p)^2 + \ldots$$

But we also know from (2.8) that $a^p = a$ for any coefficient in GF(p), such as the coefficients a,b,c above. Thus we get ,

$$[f(x)]^p = a + b\,(x^p) + c\,(x^p)^2 + \ldots.$$

But this series is exactly $f(x^p)$, so our Lemma is proved. Notice how the nature of variable x plays no role in this Lemma.

Fact 6: Let min pol mean "the minimum polynomial ". This Fact then claims that

$$[m(x) = \text{min pol of } \alpha] \Leftrightarrow [m(x) = \text{min pol of } \alpha^p] \quad \text{where } \alpha \text{ is in GF}(p^m) \tag{5.13}$$

Proof: From Lemma 6 we know that $m(x^p) = [m(x)]^p$. This means that $m(\alpha^p) = [m(\alpha)]^p = 0$, since m(x) is the minimum polynomial for α. Since m(x) is already known to be irreducible, and since $m(\alpha^p) = 0$, (5.6) says that m(x) is also the minimum polynomial for α^p. Conversely, $[m(\alpha)]^p = m(\alpha^p) = 0$ if m(x) is the minimum polynomial for α^p, which says $m(\alpha) = 0$. Since m(x) is already known to be irreducible, and since $m(\alpha) = 0$, (5.6) says that m(x) is also the minimum polynomial for α.

Fact 7: If m(x) is the minimal polynomial of α, then it is the minimal polynomial for *all* the elements in the following set of m elements (warning: the elements may not be distinct)

$$\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3}, \ldots, \alpha^{p^{m-1}} \} \qquad \alpha^{p^m} = \alpha^q = \alpha \qquad \text{m elements} \tag{5.14}$$

This implies that $m(\alpha^{p^i}) = 0$ for any α^{p^i} in the set. Big Theorem 2 (4.34) item 4 showed that $\alpha^q = \alpha$ for any α in GF(q). For this reason, there can be no elements of the above set beyond those shown -- such extra elements would just be repeats of earlier elements. The set shown nominally has m elements, but they may not all be distinct, a subject discussed soon.

Proof: Each element of this set is the previous element raised to the power p, so we just repeatedly apply Fact 6. For example, consider the element α^{p^2} ,

$$m(x) = \text{min pol of } \alpha^{p^2} \Leftrightarrow m(x) = \text{min pol of } (\alpha^p)^p \Leftrightarrow m(x) = \text{min pol of } (\alpha^p) \Leftrightarrow m(x) = \text{min pol of}(\alpha) .$$

We know from (4.34) item 4 that $\alpha^{p^m} = \alpha^q = \alpha$. This is why the list in (5.14) ends as shown. The next element would be a repeat of α, and the one after that would repeat α^p and so on. As already noted, there is no guarantee that the elements on the list are all distinct.

The *distinct* members of the set of elements in (5.14) form **the conjugate set of α**, and the distinct elements other than α are called the **conjugates** of α. As already noted, each member of the above list is the previous member raised to the power p. Do not confuse this sequence with that of a cyclic group which looks like $\{ \alpha, \alpha^2, \alpha^3, \alpha^4, \ldots\}$ or maybe like $\{ \alpha^p, \alpha^{2p}, \alpha^{3p}, \alpha^{4p}, \ldots\}$. This conjugate list is a different beast.

Fact 8: If α is a primitive element of $GF(p^m)$, then (5.15)

(a) the conjugate set of α contains the full complement of m distinct elements
(b) all members of this conjugate set are primitive elements.
(c) If any element of a conjugate set is primitive, so are all the other elements in the set
(d) The elements of a conjugate set are either all primitive, or they are all non-primitive

The set of *powers* of the conjugate set, { 1, p, p^2, p^3.... p^{m-1} }, is sometimes called a **cyclotomic coset**, see Chapter 5 (h) for more on this subject.

Proof: (a) If α is a primitive element, then from (4.31) GF(q) contains all powers of α out to a maximum exponent of q-2. In the conjugate list above, the largest conjugate has power $p^{m-1} = p^m / p = q/p$. Thus, all the conjugate elements hit distinct elements of GF(q) as long as $q/p \le q-2$ which can be rewritten as $p^{m-1}(p-1) \ge 2$. For any p>2 this is true for all m. For p=2, it is true for all m>1. We don't care about the case p=1 m=1 since this is GF(2) which we know all about. This field has no powers of α, is just has 0 and 1 as elements. Thus in fact $q/p \le q-2$ and so all m elements of the conjugate set are distinct.

(b) If α is a primitive element, we know from (4.32) that α^p is also a primitive element since GCD(p,p^m-1) = 1. Thus α^p is a primitive element, $(\alpha^p)^p = \alpha^{p^2}$ is a primitive element, and so on.

(c) and (d) are just restatements of (b).

Fact 9: If α is *not* a primitive element, then the only possibility is that the first k of the m conjugates of some element α are distinct, where k is very dependent on the field and on the field element selected. In this case, the conjugate set has this form,

$$\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3}, ... \alpha^{p^{k-1}} \} \qquad \alpha^{p^k} = \alpha \qquad \text{k conjugates} \qquad k \le m \qquad (5.16)$$

If α = 1 then only 1 conjugate is distinct. We exclude this trivial case and assume that α ≠ 1.

Proof: This is a slightly tricky proof which we defer to **Appendix B** since it is a little long and takes us off the main line of presentation. It is included in Appendix B because we could not find it in any text and had to do it from scratch. What is happening here is that the conjugates are landing on the elements of a cyclic subgroup { α, α^2, α^3 .... α^{n-1}} where $\alpha^n = 1$. Since there might not be many elements in such a subgroup, it seems reasonable that the m conjugates in the original list might hit the same elements more than once. What is not obvious is that the first repeated element is α, as implied above. This turns out to be true, and moreover, as higher and higher powers are applied, the same elements in the subgroup are hit in the same order that they were first hit, and this hitting sequence cycles over and over. Furthermore, certain members of the cyclic subgroup may never be hit.

Big Theorem 3: The minimum polynomial of an element α of GF(q) may be written as follows:

$$m(x) = (x - \alpha) \bullet (x - \alpha^p) \bullet (x - \alpha^{p^2}) \bullet (x - \alpha^{p^3}) \ ... \ .... \ (x - \alpha^{p^{k-1}}) \qquad \alpha^{p^k} = \alpha \qquad k \le m \qquad (5.17)$$

In other words, the claim is that the minimum polynomial m(x) of α is the product of (x - α) with linear factors of the form $(x-a_i)$, where the a_i are all the conjugates of α. The degree of this polynomial is then equal to the number of distinct elements in the conjugate set of α, which we call k, and we know that $k \le m$.

Proof: The proof is simple and fascinating. First of all, we already know that m(x) must have at least all the factors shown above -- it can have no smaller number of factors. This is because it needs all these factors in order to vanish at all the conjugate elements, as required by Fact 7 (5.14). If we can show that the coefficients of the m(x) given above are all in GF(p), then we are done.

Here is the trick. Write the p^{th} power of m(x) in two different ways. First, compute $[m(x)]^p$ by raising each linear factor to power p. Then apply the theorem (4.46) that $(a+b)^p = a^p + b^p$. Here is what happens to one factor:

$$(x - \alpha^{p^2})^p = (x^p - \alpha^{p^3}) .$$

It becomes the next factor to the right but with x replaced by x^p. So what happens to the final factor on the right?

$$(x - \alpha^{p^{k-1}})^p = (x^p - \alpha^{p^k}) = (x^p - \alpha).$$

It becomes the factor on the far left, with x replaced by x^p. We have thus shown that:

$$[m(x)]^p = m(x^p) = \Sigma_i c_i (x^p)^i . \qquad (5.18)$$

In the last step we just expanded polynomial $m(x^p)$ in terms of its coefficients c_i. On the other hand, we could have *started* with this coefficient expansion of m(x) and raised *it* to the power p, and then used the generalized theorem (4.47) that $(\mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d} + ...)^p = \mathbf{a}^p + \mathbf{b}^p + \mathbf{c}^p + \mathbf{d}^p + ...$ to get:

$$[m(x)]^p = (\Sigma_i c_i x^i)^p = \Sigma_i (c_i x^i)^p = \Sigma_i (c_i)^p (x^p)^i . \qquad (5.19)$$

For (5.18) and (5.19) to be equal, the coefficients must all match, so we get that $(c_i)^p = c_i$. According to Fact 16 (4.39) this means all the c_i are elements of GF(p). Thus, m(x) as shown has coefficients in GF(p) and has the fewest number of factors possible, so it is in fact the minimum polynomial of α. We have successfully produced a "formula" (5.17) for the minimum polynomial of α. We just keep adding factors until we get a repeat, then we know k. **QED**

Fact 10: The degree of any primitive polynomial of $GF(p^m)$ is m. (5.20)

Proof: A primitive polynomial is a minimum polynomial m(x) for some primitive element α. According to Fact 8 (5.15) above, the conjugate set of α has m distinct elements if α is primitive. Then according to Big Theorem 3 (5.17), we see that there is one factor (x - a) for each distinct element in the conjugate set of α, so there are m factors. Thus, the degree of m(x) is m in this case.

Fact 11: The number of primitive polynomials for $GF(p^m)$ is equal to the number of distinct conjugate sets which contain primitive elements. (5.21)

Proof: According to Big Theorem 3 (5.17), the number of distinct minimal polynomials must equal the number of distinct conjugate sets. Suppose there are N such sets, and suppose M of these contain primitive elements. Then there are M unique primitive polynomials.

For $q = p^m$, we know that each conjugate set has at most m distinct elements. If a conjugate set contains a primitive element, Fact 8 says that all m members of the set must be distinct.

We now try to bring the above general discussion down to earth with **three concrete examples**, GF(2^m) for m = 3,4,5. In each example we shall construct all minimum polynomials as products of (x-a) factors, and shall show which of these are primitive polynomials. The process is very straightforward, and it would not be hard (see section 5 below) to write a Maple program to find and classify all the minimum polynomials for any field GF(2^m) and in fact for any GF(p^m). Of course the polynomials so obtained would be in factored form, as in these examples. One then needs the tools of Chapter 6 (d) to expand ("multiply out") these factored forms.

1. Minimum and primitive polynomials of GF(2^3)

We know that from Big Theorem 1 (4.30) that any GF(p^m) is cyclic. This means that there must exist a primitive element α whose powers enumerate the non-zero elements of GF(p^m). Thus, we may "enumerate" GF(2^3) as follows:

$$\{ 0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6 \} \qquad \alpha^7 = 1 \qquad \alpha = \text{some primitive element of GF}(2^3) \qquad (5.22)$$

Which elements are primitive elements besides α? Fact 12 (4.32) tells us that a power α^s is a primitive element if GCD(s,7) = 1. So s = 1,2,3,4,5,6 all define primitive elements. It happens that the number q-1 = 2^3-1= 7 is prime, so *all* α^s will be primitive elements, a Fact noted earlier in (4.33).

Next, let's construct the conjugate sets using the template (5.17). Since all our α^s for s in the list are primitive, we know from (5.15) that all the conjugate sets have m=3 elements. So:

$\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3}$, ... $\alpha^{p^{m-1}} \}$	$\alpha^{p^m} = \alpha^q = \alpha$	m elements	// template
$\{ \alpha, \alpha^2,$ α^4 $\}$	$\alpha^8 = \alpha^8 = \alpha$	3 elements	// GF(2^3)

Our task now is to replace α by α^s in the last line and see if we get a distinct conjugate set. So

$\{ \alpha, \alpha^2, \alpha^4 \}$ // the first conjugate set
$\{ \alpha^s, \alpha^{2s}, \alpha^{4s} \}$ // is this conjugate set distinct from those previously obtained?

Note for example on the last line that $(\alpha^s)^4 = \alpha^{4s}$. We can then start with each candidate α^1, α^2.... α^6 and try to build a conjugate set from it. But if we find that some element has already appeared in a set, we skip that one and move to the next candidate. This is because no element of GF(q) can appear in two distinct conjugate sets, a fact we will formally prove below in (5.38).

The first set is of course $\{ \alpha, \alpha^2, \alpha^4 \}$. Since this includes α^2, we skip α^2 and move on to α^3. This gives $\{ \alpha^3, \alpha^6, \alpha^5 \}$ where the last element is $(\alpha^3)^4 = \alpha^5$ $\alpha^7 = \alpha^5$ 1 = α^5 . We then skip α^4, α^5 and α^6 and we have run the gamut. So we find these three conjugate sets, {1} being a trivial one :

$$\underset{\text{prim}}{\{ \alpha, \alpha^2, \alpha^4 \}}, \underset{\text{prim}}{\{ \alpha^3, \alpha^6, \alpha^5 \}}, \{1\} \qquad (5.23)$$

Notice that the elements of these three sets *partition* the field elements {GF(2^3) - 0}, a fact proven in (5.38) below for any GF(q).

In a denser notation, the above partition can be written in terms of just the powers

{1,2,4}, {3,6,5}, {0}

and this grouping is referred to as the set of cyclotomic cosets for $GF(2^3)$, see Chapter 5 (h).

Since all elements of {$GF(2^3)$-0-1} are primitive, the first two conjugate sets contains only primitive elements. More generally, we know that from (5.15) that either all elements of a conjugate set are primitive, or they are all not-primitive. Of course 1 is not a primitive element.

Here then is a complete list of the minimum polynomials for $GF(2^3)$. Here we label each polynomial by the lowest appearing power of α.

$$\begin{aligned} p_1(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^4) \\ p_3(x) &= (x - \alpha^3)(x - \alpha^6)(x - \alpha^5) \\ m_0(x) &= (x-\alpha^0) = (x-1) \\ m_{zero}(x) &= (x-0) = x \quad . \end{aligned} \tag{5.24}$$

In the next two examples we won't mention the last two trivial minimum polynomials shown here. They are always present. Fact 11 (5.21) then says there are two distinct primitive polynomials for $GF(2^3)$, and we see them as the $p_i(x)$ in (5.24).

So, although there are 6 elements in {$GF(2^3)$-0-1}, there are only two minimal polynomials and they are shared by the 6 elements as shown. Both these minimal polynomials are primitive polynomials since α and α^3 are primitive elements. Each element of {$GF(2^3)$} appears in exactly one (x-a) factor, in line with the partition comment above.

We will show in (5.30) below that the first two polynomials in (5.24) must be $x^3 + x + 1$ and $x^3 + x^2 + 1$. But how can we verify this? And which one is which? At this point all we know about $p_1(x)$ is this:

$$p_1(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4) = x^3 + [\alpha + \alpha^2 + \alpha^4]x^2 + [\alpha^3 + \alpha^5 + \alpha^6]x + \alpha^7 \quad . \tag{5.25}$$

The answer to the second question is that it depends on which α you select at the start as your primitive element. As for the first question, in order to multiply things out, we have to have at hand the addition table for $GF(2^3)$ where the rows and columns are enumerated by powers of α. We showed such a table for $GF(2^2)$ in (4.41), and we showed how such a table can be created, but we have not yet stated the table for $GF(2^3)$, so at this point we really don't have the tools needed to multiply out the polynomials in (5.24). There is a circular route at play here. In order to create an addition table, we have to select an α whose powers will enumerate the table's rows and columns. That implicitly means that we have to know and select a specific primitive polynomial. That in turn will determine which $p_i(x)$ of (5.24) is $x^3 + x + 1$ and which is $x^3 + x^2 + 1$. This subject will be treated in detail in Chapter 6 (d) 2.

2. Minimum and primitive polynomials of $GF(2^4)$

The reader is assumed to have carefully read through the previous example, so not all the supporting "words" will be repeated here. We first enumerate the field $GF(2^4)$, having selected α as some primitive element,

$\{ 0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}$ $\quad\quad$ $\alpha^{15} = 1 \quad \alpha = \text{prim}$

Which elements are primitive elements besides α? Fact 12 (4.32) tells us that a power α^s is a primitive element if GCD(s,15) = 1. So s = 1, 2, 4, 7, 8, 11, 13, 14 all define primitive elements. Things are different now because $q-1 = 2^4-1 = 15$ is *not* a prime number.

Next, let's construct the conjugate sets using the template (5.17). Since *not* all our α^s for s being any integer in 1 to 14 are primitive, we know from (5.15) that *not* all the conjugate sets have m=4 elements. Some will likely have k elements where k < 4. This time from the template we get

$\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3} \quad , \ldots. \ \alpha^{p^{k-1}} \}$ $\quad$ $\alpha^{p^m} = \alpha^q = \alpha$ $\quad$ k elements $\quad$ // general

$\{ \alpha, \alpha^2, \alpha^4, \quad\quad \alpha^8 \quad \}$ $\quad$ $\alpha^{16} = \alpha^{16} = \alpha$ $\quad$ 4 elements $\quad$ // GF(2^4)

Since α is a primitive element, *its* conjugate set has the full m = 4 elements. Here then is the situation:

$\{ \alpha, \alpha^2, \alpha^4, \alpha^8 \}$ $\quad$ // the first conjugate set

$\{ \alpha^s, \alpha^{2s}, \alpha^{4s}, \alpha^{8s} \}$ $\quad$ // is this conjugate set distinct from those previously obtained?

We can then start with each candidate α^1, α^2.... and try to build a conjugate set from it, exactly as outlined in the previous example. This time, however, we show how to "automate" this process with a simple Maple program,

```
for s from 1 to 7 by 2 do
  e1 := (1*s) mod 15:
  e2 := (2*s) mod 15:
  e3 := (4*s) mod 15:
  e4 := (8*s) mod 15:
  print(a^e1,a^e2,a^e3,a^e4);
od:
```

In theory, we should really be using all s values in the range s = 1,2,3...14, but it turns out we can find all the conjugate sets by using just odd values of s from 1 to 7. That is to say, using this limited range of s, we find a group of conjugate sets that partitions {GF(2^4)-1} and so we know we are done. The mod 15 operations take care of things like $(\alpha^7)^8 = \alpha^{56} = \alpha^{45}\alpha^{11} = \alpha^{11}$, since $\alpha^{15} = 1$. Here is the Maple output which we have marked up to remove duplicate elements,

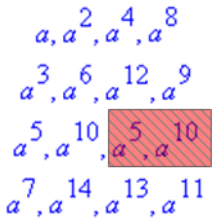


Fig 5.1

Here then are the resulting conjugate sets of GF(2^4) :

$\{ \alpha, \alpha^2, \alpha^4, \alpha^8 \}, \{ \alpha^3, \alpha^6, \alpha^{12}, \alpha^9 \}, \{ \alpha^5, \alpha^{10} \}, \{ \alpha^7, \alpha^{14}, \alpha^{13}, \alpha^{11} \}, \{1\}$ $\quad$ (5.26)

prim $\quad\quad$ prim

Note again that the elements of {GF(2^4)-0} are partitioned into the conjugate sets: each element appears exactly once in some set, see Fact 17 (5.38). Recall from above that the list s = 1, 2, 4, 7, 8, 11, 13, 14 gives the primitive elements, and we thus have two conjugate sets containing primitive elements and 3 conjugate sets containing non-primitive elements, in accordance with Fact 8 (d) of (5.15).

Using the conjugate sets, we construct four minimum polynomials for GF(2^4), but only the first two are primitive polynomials.

$$\begin{aligned} p_1(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8) \\ p_7(x) &= (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{13})(x - \alpha^{11}) \\ m_3(x) &= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^9) \\ m_5(x) &= (x - \alpha^5)(x - \alpha^{10}) . \end{aligned} \qquad (5.27)$$

As in the previous example, we don't yet have the tools to multiply out these polynomials to see them in their simple form with coefficients all being 0 or 1 in the ground field GF(2). This will be done in Chapter 6 (d) 2.

3. Minimum and primitive polynomials of GF(2^5)

One more example, since the results will be used later on. We try now to be more systematic:

1. Write out the elements:

$\{ 0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, ...\alpha^{31}\}$ $\qquad \alpha^{31} = 1 \qquad \alpha$ = primitive element

2. Using Fact 12 (4.32), identify the primitive elements as α^s where GCD(s,31) = 1. Since 2^5-1 = 31 = prime, this example is like the first one GF(2^3) and again all elements of {GF(2^5)-0-1} are primitive elements. Thus, all conjugate sets are going to have m = 5 elements according to (5.15).

3. Construct the first conjugate set using the template (5.17),

$\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3}, ... \alpha^{p^{m-1}} \}$	$\alpha^{p^m} = \alpha^q = \alpha$	m elements	// template
$\{ \alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16} \}$	$\alpha^{32} = \alpha$	5 elements	// GF(2^5)

4. For general s, the conjugate set has the following form, where the exponent must be done mod 31 (we are just replacing α by (α^s) in the second line above)

$$\{ \alpha^s, \alpha^{2s}, \alpha^{4s}, \alpha^{8s}, \alpha^{16s} \} .$$

Here is the Maple program modified for this case,

```
for s from 1 to 15 by 2 do
  e1 := (1*s)   mod 31:
  e2 := (2*s) mod 31:
  e3 := (4*s) mod 31:
  e4 := (8*s) mod 31:
  e5 := (16*s) mod 31:
  print(a^e1,a^e2,a^e3,a^e4,a^e5);
od:
```

and here it the marked-up output,

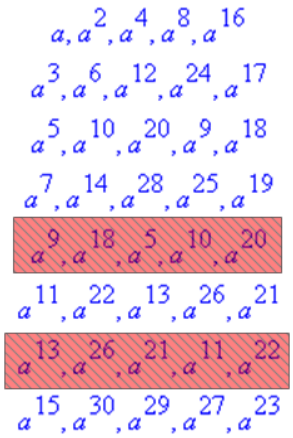


Fig 5.2

Once again, every power appears exactly once in some conjugate set, as required by (5.38). Checking this, one starts to become a believer in Galois Theory.

There are therefore 6 minimum polynomials for GF(2^5) and they are all primitive polynomials:

$$
\begin{aligned}
p_1(x) &= (x-\alpha)\,(x-\alpha^2)\,(x-\alpha^4)\,(x-\alpha^8)\,(x-\alpha^{16}) \\
p_3(x) &= (x-\alpha^3)\,(x-\alpha^6)\,(x-\alpha^{12})\,(x-\alpha^{24})\,(x-\alpha^{17}) \\
p_5(x) &= (x-\alpha^5)\,(x-\alpha^{10})\,(x-\alpha^{20})\,(x-\alpha^9)\,(x-\alpha^{18}) \\
p_7(x) &= (x-\alpha^7)\,(x-\alpha^{14})\,(x-\alpha^{28})\,(x-\alpha^{25})\,(x-\alpha^{19}) \\
p_{11}(x) &= (x-\alpha^{11})\,(x-\alpha^{22})\,(x-\alpha^{13})\,(x-\alpha^{26})\,(x-\alpha^{21}) \\
p_{15}(x) &= (x-\alpha^{15})\,(x-\alpha^{30})\,(x-\alpha^{29})\,(x-\alpha^{27})\,(x-\alpha^{23})
\end{aligned}
\qquad (5.28)
$$

Once again, we do not yet have the tools to multiply out these polynomials to see them in their simple form with coefficients all being 0 or 1 in the ground field GF(2). This will be done in Chapter 6 (d) 2.

4. Minimum and primitive polynomials of GF(p) for p = 2,3,5,7.

The following Facts will be used in this short section. It happens that q = p, but things are stated for general q. Hopefully this list of Facts provides a good review of important ideas and their application below provides the reader some exercise in applying these Facts.

[0] GF(p) = Mod(p) also called Z_p [(2.5)]

[1] Every GF(q) has at least one primitive element. [since {GF(q) - 0} is cyclic under • , see (4.31)]

[2] $\alpha^{q-1} = 1$ for any non-zero element α of GF(q). [(4.34) item 4]

[3] The order of α in GF(q) is the smallest integer n such that $\alpha^n = 1$. [(4.17) item k]

[4] If α is primitive in GF(q), its order is q-1 and vice versa. [(4.31)]

[5] The order n of α in GF(q) divides q-1. [(4.17) item (ℓ)]

[6] The number of primitive elements of GF(q) is φ(q-1) [Section 5 (d) below, with table of Euler φ.]

In each example, we pay attention to the symmetry of the coefficients of the primitive polynomials obtained, since this will be of interest later.

GF(2) { 0, 1 } $\alpha = 1$ α = prim

In this trivial case, since we can enumerate all non-zero elements of GF(2) by powers of 1, element 1 is a primitive element. That will not be true for GF(p) with p > 2 so 1 will never be a primitive element again in this set of examples. For GF(2), the only minimum polynomial is (x-1) and it is primitive. Since + and - are the same for GF(2), we can write this as (x+1) which we note has symmetric coefficients " 11" .

GF(3) { 0, 1, α } $\alpha^2 = 1$ α = prim

From [1] there must be a primitive element, so it must be α = 2, and indeed $2^2 = 4 = 1$ according to the rule [0] of GP(3) = Mod(3). The only minimum polynomial is (x-2) and it is a primitive polynomial. Since –2 = +1 for GF(3), we can write this as (x+1) and again we have symmetric coefficients "11" .

GF(5) { 0, 1, α, α^2, α^3 } $\alpha^4 = 1$ α = prim

From [6] there are φ(p-1) = φ(4) = 2 primitive elements, so in the set {2,3,4} one must *not* be primitive. They all satisfy $\alpha^4 = 1$ according [2], but $4^2 = 16 = 1$ so [3] says the order of 4 is 2 and not 4 = q-1, so 4 is not primitive by [4]. There are three minimum polynomials (x-2),(x-3),(x-4), and only (x-2),(x-3) are primitive. Just as a check, note that $2^2 = 4 \neq 1$ and $3^2 = 9 = 4 \neq 1$, so both 2 and 3 have the full order 4. The two primitive polynomials do not have symmetric coefficients even if written as (x+3), (x+2) = "13" and "12".

GF(7) { 0, 1, α, α^2, α^3, α^4, α^5 } $\alpha^6 = 1$ α = prim

From [6] there are φ(p-1)=φ(6) = 2 primitive elements, so in the set {2,3,4,5,6} only two are primitive. Since q-1 = p-1 = 6, non-primitive elements could have order 2 or 3 according to [5] .

For 2 we find $2^2 = 4$, but $2^3 = 8 = 1$ so 2 has order 3 and is non-primitive by [4].

For 3 we find $3^2 = 9 = 2$, $3^3 = 27 = 6$, so only $3^6 = 1$ so 3 is **primitive** by [4].

For 4, we find $4^2 = 16 = 2$, $4^3 = 64 = 1$, so 4 has order 3 and is non-primitive by [4].
For 5, we find $5^2 = 25 = 4$, $5^3 = 25*4 = 4*4 = 16 = 2$, so 5 is **primitive** by [4].
For 6, we find $6^2 = 36 = 1$, $6^3 = 36*6 = 1*6 = 6$, so 6 has order 2 and is non-primitive by [4].

The minimum polynomials are (x-2), (x-3), (x-4), (x-5), (x-6) but only (x-3), (x-5) are primitive. Even when written as (x+4), (x+2), these primitive polynomials "14" and "12" are not symmetric.

(d) How many primitive elements and primitive polynomials are there for GF(p^m)?

According to (4.31) there always exists a primitive element α for GF(q=p^m). The 0 and 1 elements are never primitive, and the remaining q-2 elements are enumerated as powers of α : $\{\alpha, \alpha^2, \alpha^3 .....\alpha^{q-2}\}$. According to (4.32), a power α^k of a known primitive element α is also primitive iff GCD(k,q-1) = 1. Thus, the number of primitive elements must be the number of integers k in the range 0 to q-2 for which GCD(k,q-1) = 1, which recall means that k and q-1 are coprime. The number of coprime integers < n is called φ(n), Euler's "totient function", so the number of primitive elements is φ(q-1). Each primitive polynomial covers m primitive elements of GF(q) as shown in (5.15a), so the number of primitive polynomials is then φ(q-1)/m. These matters are described in more detail in Appendix G and we just quote the two results which, in fact, we have just proven:

Fact 13: The number of primitive elements in GF(q=p^m) is φ(p^m-1). *(G.19)*
Fact 14: The number of primitive polynomials in GF(p^m) is φ(p^m -1)/m. *(G.20)*

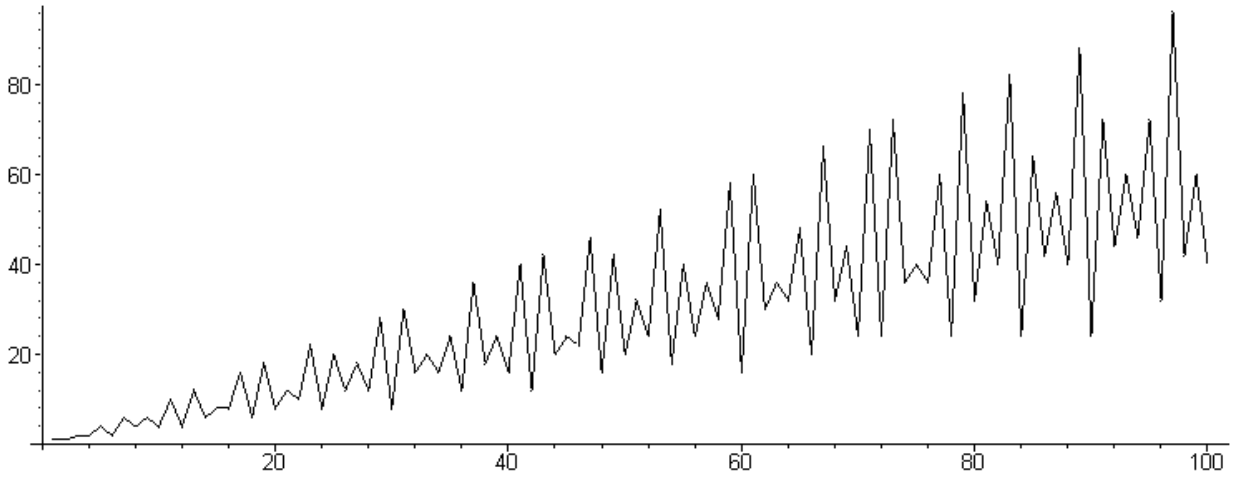
Here is a Maple-generated table of φ(n) for n = 1 to 100 where each bracket is of the form [n, φ(n)].

```
with(numtheory);
for n from 1 to 100 by 10 do
  print(seq([n+i,phi(n+i)],i = 0..9));
od;
```

```
[1, 1], [2, 1], [3, 2], [4, 2], [5, 4], [6, 2], [7, 6], [8, 4], [9, 6], [10, 4]
[11, 10], [12, 4], [13, 12], [14, 6], [15, 8], [16, 8], [17, 16], [18, 6], [19, 18], [20, 8]
[21, 12], [22, 10], [23, 22], [24, 8], [25, 20], [26, 12], [27, 18], [28, 12], [29, 28], [30, 8]
[31, 30], [32, 16], [33, 20], [34, 16], [35, 24], [36, 12], [37, 36], [38, 18], [39, 24], [40, 16]
[41, 40], [42, 12], [43, 42], [44, 20], [45, 24], [46, 22], [47, 46], [48, 16], [49, 42], [50, 20]
[51, 32], [52, 24], [53, 52], [54, 18], [55, 40], [56, 24], [57, 36], [58, 28], [59, 58], [60, 16]
[61, 60], [62, 30], [63, 36], [64, 32], [65, 48], [66, 20], [67, 66], [68, 32], [69, 44], [70, 24]
[71, 70], [72, 24], [73, 72], [74, 36], [75, 40], [76, 36], [77, 60], [78, 24], [79, 78], [80, 32]
[81, 54], [82, 40], [83, 82], [84, 24], [85, 64], [86, 42], [87, 56], [88, 40], [89, 88], [90, 24]
[91, 72], [92, 44], [93, 60], [94, 46], [95, 72], [96, 32], [97, 96], [98, 42], [99, 60], [100, 40]
```

and here is a plot of the digital function φ(n) with linear interpolation of the points n = 1 to 100:

```
p := seq([n,phi(n)],n=1..100):
pointplot([p],style=line);
```



A few examples:

GF(p^m)	# prim elements	# prim polys	
2^2	φ(3) = 2	2/2 = 1	// shown in (5.29)
2^3	φ(7) = 6	6/3 = 2	// listed in (5.24) and (6.20)
2^4	φ(15) = 8	8/4 = 2	// listed in (5.27) and (6.21)
2^5	φ(31) = 30	30/5 = 6	// listed in (5.28) and (6.22)
3^2	φ(8) = 4	4/2 = 2	// quoted below equation (6.15)

(e) On finding the minimum and primitive polynomials of GF(p^m) expressed over GF(p)

In Chapter 6 (d) 2 we will show how to expand all the factored minimum polynomials stated above to obtain their simple forms over GF(2), like $x^5 + x^3 + 1$. This requires *a priori* knowledge of one primitive polynomial. We know in principle how to find such a primitive polynomial by brute force. We need the addition table and that can be obtained from our R/I representation of GF(p^m) of (3.17),

$$GF(p^m) = R / (f(x)) \ . \qquad (3.17)$$

Note that f(x) does not have to be a primitive polynomial, nor does it have even to be a minimum polynomial. It need only be irreducible in R and of degree m. Such a polynomial can easily be found, as in (4.13), and the + and • tables constructed as in (4.10) and (4.14) where the elements are written as m-tuples. Then taking α to be each of these m-tuples one at a time, one can iteratively use the • table to compute the powers of this α. At some point one must find an α of order q-1 since we know such an α must exist for a cyclic group like {GF(q)-0}. In this way one can identify a primitive element α.

This does seem like a lot of work.

Probably a better way is trial and error, at least for smaller GF(q).

GF(2^2): We know from just above that there is exactly one primitive polynomial. All candidates must be of degree m = 2, and must have +1 as the final term. If the +1 were missing, x could be factored out and the candidate would then not be a minimum polynomial. The grand list of candidates is then the following two polynomials :

$x^2 + x + 1$	•	111	
$x^2 + 1$		$= x^2 - 1 = (x-1)(x+1)$ = reducible	(5.29)

Since the second one is reducible, the first one must be the sole primitive polynomial of GF(2^2). Notice that if you reverse the order of the m-tuple symbols 111 you get the same 111.

GF(2^3): Now there are four polynomial candidates,

$x^3 + x^2 + x + 1$		$= x^2 (x + 1) + (x+1) = (x^2+1)(x+1)$ = reducible	
$x^3 + x^2 + 1$	°	1101	
$x^3 + x + 1$	•	1011	
$x^3 + 1$		$= x^3 - 1$ = reducible	(5.30)

Since we know both from the previous section and from (5.23) that there are two primitive polynomials for GF(2^3), and since we have only two irreducible candidates in the list, they must be the ones! We didn't even have to check that the period of each is 7, we know it must be true.

GF(2^4): Now there are more possibilities to worry about:

1	$x^4 + x^3 + x^2 + x + 1$	divides into $x^5 - 1$ so flunks period test, not a primitive polynomial		
2	$x^4 + x^3 + x^2 + 1$			
3	$x^4 + x^3 + x + 1$	$= (x^3+1)(x+1)$ = reducible		
4	$x^4 + x^2 + x + 1$			
5	$x^4 + x^3 + 1$	°	11001	
6	$x^4 + x + 1$	•	10011	
7	$x^4 + x^2 + 1$			
8	$x^4 + 1$	$= x^4 - 1 = (x-1)(x^3+x^2+x+1)$ = reducible		(5.31)

We know from (5.27) that there are only three minimum polynomials for GF(2^4) of degree 4, two of which are primitive polynomials. We can eliminate items 3 and 8 since they are reducible, but this still leaves six candidates (1,2,4,5,6,7), all of which look irreducible. One could apply the period test (5.10) to each to try to eliminate some of the candidates from being the *primitive* polynomials: which ones (if any) divide into $x^n - 1$ for n less than 15? The first one we can see divides into $x^5 - 1$, so we know it is not a primitive polynomial. The others are less obvious.

An excellent test is to see which items on the list divide evenly into $x^{15}-1$ with no remainder. Any minimum polynomial must divide evenly since it is a product of $(x-a_i)$ factors and $x^{15}-1$ is the product of all such factors according to (4.34). One could manually compute all the remainders for (5.31), but Maple is happy to do it for us:

```
Rem(x^15-1,x^4+x^3+x^2+x+1,x) mod 2; #1
                                0
Rem(x^15-1,x^4+x^3+x^2+1,x) mod 2;   #2
                              x + 1
Rem(x^15-1,x^4+x^3+x+1,x) mod 2;     #3
                              x^3 + 1
Rem(x^15-1,x^4+x^2+x+1,x) mod 2;     #4
                              x + 1
Rem(x^15-1,x^4+x^3+1,x) mod 2;       #5
                                0
Rem(x^15-1,x^4+x+1,x) mod 2;         #6
                                0
Rem(x^15-1,x^4+x^2+1,x) mod 2;       #7
                              x^3 + 1
Rem(x^15-1,x^4+1,x) mod 2;           #8
                              x^3 + 1
```

This clinches it! Polynomial #1 is a minimum polynomial, but we know already it is not a primitive polynomial since it divides $x^5 - 1$. Thus, it must be $m_3(x)$ of (5.27). Since there are only two other candidates with 0 remainder, they must be the two primitive polynomials $p_1(x)$ and $p_7(x)$ of (5.27). All the other candidates are not minimum polynomials of any kind. This is a reminder that not all GF(p) polynomials can be factored over GF(q), although all polynomials with real coefficients *can* be factored in the complex numbers.

The following **m=tuple reversal rule** is of some assistance:

Fact 12: (5.32)
(a) If f(x) is a primitive polynomial, then so is F(x) defined by reversing the m-tuple coefficients.
(b) For m > 2, no primitive polynomial can be symmetric under coefficient reversal.

Proof: These two items are derived in Appendix H as (H.15) for (a) and (H.17) + (H.18) for (b). See also Peterson and Weldon, p 169, Problem 6.7. They refer to an order-reversed polynomial as a reciprocal polynomial.

Example: Looking at (5.30) and (5.31) one sees that • and ° have reversed m-tuples in each case.

Example: If $f(x) = x^4 + 2x^3 + x + 1 = 12011$, $F(x) = 1 + 2x + x^3 + x^4 = <12011> = 11021$.

Example: For GF(2^2) we saw above (5.30) that $p(x) = x^2 + x + 1 = 111$. This has reversal symmetry, but that is allowed for $m \leq 2$.

Example: For GF(3^2) we know from section (d) above that there are 2 primitive polynomials. Below (6.15) we claim these are $x^2 + x + 2$ and $x^2 + 2x + 2$, or 112 and 122. These don't appear to be an order reversal pair, and this would contradict Fact 12 (a) above. However, let's look more closely at the order-reversed version of our first polynomial $f(x) = x^2 + x + 2$. Its order-reversed version is $F(x) = 1 + x + 2x^2$.

We can factor out a 2 and write this as $F(x) = 2(x^2 + 2x + 2) = 2x^2 + x + 1$ using Mod(3). Thus, F(x) is twice our second primitive polynomial. If we ignore constant factors like this which arise for p > 2, then we see that $x^2 + x + 2$ and $x^2 + 2x + 2$ are equivalent to $x^2 + x + 2 = 112$ and $2x^2 + x + 1 = 211$ and then we have an order-reversed pair consistent with the claim of Fact 12 (a).

In light of this last example, we might restate Fact 12 (b) this way: For m > 2 one will find that, when primitive polynomials are written in the right way, and when overall constants are ignored, then every primitive polynomial f(x) will have an order-reversed partner primitive polynomial F(x).

Olofsson's website provides much information on primitive polynomials for $GF(2^m)$. One listing provides one primitive polynomial for m from 2 to 1256. Here is part of that list for m = 2 to 32 :

```
x^2 + x + 1
x^3 + x + 1
x^4 + x + 1
x^5 + x^2 + 1
x^6 + x^4 + x^3 + x + 1
x^7 + x + 1
x^8 + x^4 + x^3 + x^2 + 1
x^9 + x^4 + 1
x^10 + x^6 + x^5 + x^3 + x^2 + x + 1
x^11 + x^2 + 1
x^12 + x^7 + x^6 + x^5 + x^3 + x + 1
x^13 + x^4 + x^3 + x + 1
x^14 + x^7 + x^5 + x^3 + 1
x^15 + x^5 + x^4 + x^2 + 1
x^16 + x^5 + x^3 + x^2 + 1
x^17 + x^3 + 1
x^18 + x^12 + x^10 + x + 1
x^19 + x^5 + x^2 + x + 1
x^20 + x^10 + x^9 + x^7 + x^6 + x^5 + x^4 + x + 1
x^21 + x^6 + x^5 + x^2 + 1
x^22 + x^12 + x^11 + x^10 + x^9 + x^8 + x^6 + x^5 + 1
x^23 + x^5 + 1
x^24 + x^16 + x^15 + x^14 + x^13 + x^10 + x^9 + x^7 + x^5 + x^3 + 1
x^25 + x^8 + x^6 + x^2 + 1
x^26 + x^14 + x^10 + x^8 + x^7 + x^6 + x^4 + x + 1
x^27 + x^12 + x^10 + x^9 + x^7 + x^5 + x^3 + x^2 + 1
x^28 + x^13 + x^7 + x^6 + x^5 + x^2 + 1
x^29 + x^2 + 1
x^30 + x^17 + x^16 + x^13 + x^11 + x^7 + x^5 + x^3 + x^2 + x + 1
x^31 + x^3 + 1
x^32 + x^15 + x^9 + x^7 + x^4 + x^3 + 1
```

Fig 5.3

Another listing provides *all* primitive polynomials for degrees m = 2 through 20. For example, we find for m = 5 that

```
x^5 + x^2 + 1
x^5 + x^4 + x^2 + x + 1
x^5 + x^3 + x^2 + x + 1
x^5 + x^3 + 1
x^5 + x^4 + x^3 + x + 1
x^5 + x^4 + x^3 + x^2 + 1
```

Fig 5.4

We shall obtain these in Chapter 6 (d) 2. The website does not give minimum polynomials that are not primitive polynomials.

A classic 1962 note by E.J. Watson gives a half page summary of his algorithm for doing this, though he did not use Maple. Tables of irreducible polynomials are given in Peterson and Weldon, Appendix C, see Chap 6 (d) 7 below. More extensive tables appear in Lidi and Niederreiter.

In section (**ℓ**) below we present a short Maple program that computes all minimum polynomials for any GF(p^m) where the polynomials are in factored form.

(f) Selecting f(x) for GF(q) = R/(f(x)) ; Classifying Irreducible Polynomials

In Chapter 3 we were able to construct a representation of GF(p^m) by starting with any irreducible polynomial of degree m, q = p^m. We made the identification:

GF(p^m) = R/(f(x)) . // irreducible f(x) is of degree m *(3.17)*

As our GF(2^4) example above shows, even when p=2, there are typically many candidates for f(x), all of which give an equivalent representation of GF(p^m). Moreover, for p>2, every candidate has p-2 shadow candidates which are just integer multiples of f(x).

By our definition, a primitive polynomial is monic and so the shadow candidates are eliminated. Since a primitive polynomial of degree m is irreducible, it can serve as an f(x) in the construction of GF(p^m) as the extension field over GF(p). However, it is not necessary to use a primitive polynomial to accomplish the construction. As we shall see in the next chapter, there is a great *advantage* to using a primitive polynomial for this purpose.

We now look into the classification of the irreducible polynomials of GF(q). We start with:

Fact 13: If monic irreducible f(x) is used to construct a representation of GF(q), then f(x) is a minimum polynomial of GF(q) in that representation. Since all representations of GF(q) are equivalent, f(x) is a minimum polynomial of GF(q). (5.33)

Proof: In Fact 1 (6.5) we will show that α = {x} is a root of any degree-m polynomial f(x) used to construct a residue class ring representation of GF(q). Since this α = {x} is an element of GF(q) and is a root of f(x), we can apply Fact 3 (5.6) to conclude that f(x) is a minimum polynomial.

Fact 14:

(a) If monic irreducible f(x) is of degree m, then it is a minimum polynomial for GF(p^m)

(b) If monic irreducible f(x) is not a minimum polynomial for GF(p^m), it must have degree < m.

(5.34)

These are corollaries to Fact 15. If monic irreducible f(x) has degree m, it can be used in GF(q) = R/(f(x)) and it is therefore a minimum polynomial of GF(q). Then (b) is the contrapositive of (a).

Using Fact 14, we can now draw a picture to classify all monic irreducible polynomials of GF(q).

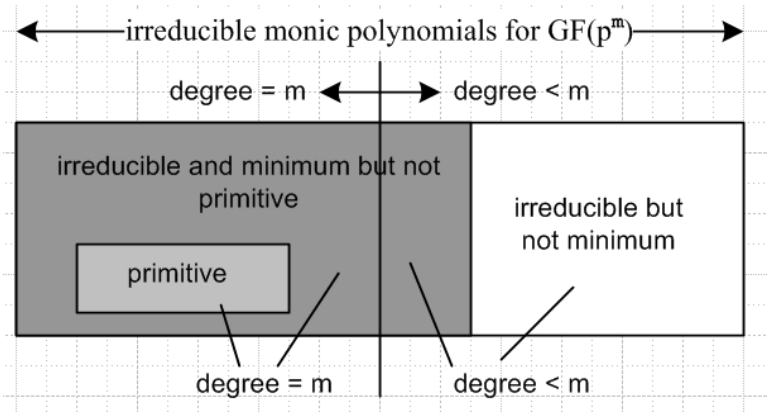


Fig 5.5

Note that minimum polynomials could be in either side of the drawing. A sample member of the white region on the right is $1 + x + x^3$ for GF(2^4), as demonstrated below in Chap 6 (d) 6.

Non-monic polynomials can always be written as multiples of monic polynomials and can then be classified by the resulting monic polynomial. For example, in GF(5^m) we would have

$(2 + x + 3x^3) = 2(1 + 3x + 4x^3)$ since $2 \bullet 3 = 6 = 1$ and $2 \bullet 4 = 8 = 3$

(g) More facts about conjugate sets and minimum polynomials

Fact 15: (a) All elements of a conjugate set have the same order.
(b) If α is primitive then all elements of the conjugate set of α are primitive (order q-1). (5.35)

Proof: (b) If α is primitive, it has order q-1 by definition. From part (a), all other elements of the conjugate set have order q-1. They are thus all primitive elements of GF(q).

(a) According to (5.17), the elements of a conjugate set of α all have the form α^{p^i} where i ranges from 0 to some $k \le m$. From (4.17) (**ℓ**) we know the order of any α must divide $q-1 = p^m-1$ so write

[order of α] $\equiv n = (p^m-1)/K$ where K is an integer.

According to (4.21) applied with $\beta = \alpha^{p^i}$ and $k = p^i$,

[order of α^{p^i}] $= n / GCD(p^i, n) = [\text{order of } \alpha] / GCD(p^i, (p^m-1)/K)$.

According to the Lemma just below, $GCD(p^i, (p^m-1)/K) = 1$, so our Fact is proven since [order of α^{p^i}] = [order of α]. **QED**

Lemma: $GCD(p^i, (p^m-1)/K) = 1$ for i = 0,1....m (note that $m \ge i$), K = integer . (5.35a)

Proof: Assume that $GCD(p^i, (p^m-1)/K) = N > 1$. Then we can write

$p^i/N = I \quad \Rightarrow \quad p^i = NI$
$[(p^m-1)/K]/N = J \quad \Rightarrow \quad (p^m-1) = NJK$

where I and J are integers. Write $p^m = p^i\, p^{m-i}$ where $m-i \geq 0$ so then the last equation above becomes

$$(p^i\, p^{m-i} - 1) = NJK \quad .$$

But $p^i = NI$ so we have

$$(NI\, p^{m-i} - 1) = NJK$$
$$N(I\, p^{m-i} - JK) = 1$$
$$(I\, p^{m-i} - JK) = 1/N \quad .$$

Since the left side is an integer and the right side is a fraction, we have a contradiction, so N = 1. **QED**

Notice that the fact that p = prime was not necessary in the above Lemma proof. It never hurts to test a proof in Maple. Here we let p,m,i,K range from 1 to 25 and look for $GCD(p^i, (p^m-1)/K) \neq 1$:

```
n := 25: found := 0:
for p from 1 to n do
  for m from 1 to n do
    for K from 1 to n do
      for i from 0 to n do
        g := gcd(p^i,(p^m-1)/K);
        if g <> 1 then print(p,m,p^m-1,K,i,g); found := 1 fi;
        # print("p,m,p^m-1,K,i,g = ", p,m,p^m-1,K,i,g);
      od;
    od;
  od;
od;
if found = 0 then print( "no cases found for n = ", n) fi;
```

"no cases found for n = ", 25

Corollary 15: All roots of any minimum polynomial m(x) have the same order, see (5.17). In particular, all roots of a primitive polynomial have order q-1 since any primitive element has order q-1. (5.36)

Fact 16: Let $\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3}, \ldots \alpha^{p^{k-1}} \}$ be the conjugate set of α of some order $k \leq n$. Let $\beta \equiv \alpha^{p^r}$ be any one of these conjugates. Then the exact same set can be written $\{ \beta, \beta^p, \beta^{p^2}, \beta^{p^3}, \ldots \beta^{p^{k-1}} \}$. Thus any element of a conjugate set can be used to generate the conjugate set. (5.37)

Proof: Perhaps this seems obvious, but here is a proof. Start with

$$\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3}, \ldots \alpha^{p^{k-1}} \} \qquad \alpha^{p^k} = \alpha \qquad \text{k conjugates} \qquad k \leq m \qquad \textit{(5.16)}$$

Take α , which is first in the list, and move it to the end of the list, using $\alpha^{p^k} = \alpha$:

$$\{\alpha^{p}, \alpha^{p^2}, \alpha^{p^3}, \ldots \alpha^{p^{k-1}}, \alpha^{p^k}\} = \{\alpha^{p}, \alpha^{p^2}, \alpha^{p^3}, \ldots \alpha^{p^{1+k-1}}\}$$

Now move α^{p} to the right end and use $\alpha^{p^{k+1}} = \alpha^{p^k p} = (\alpha^{p^k})^{p} = \alpha^{p}$ to get

$$\{\alpha^{p^2}, \alpha^{p^3}, \ldots \alpha^{p^{k-1}}, \alpha^{p^k}, \alpha^{p^{k+1}}\} = \{\alpha^{p^2}, \alpha^{p^3}, \ldots \alpha^{p^{2+k-1}}\}$$

Doing this r times lets us list off the conjugate set as follows

$$\{\alpha^{p^r}, \alpha^{p^{r+1}} \ldots \alpha^{p^{r+k-1}}\} = \{[\alpha^{p^r}], [\alpha^{p^r}]^{p} \ldots [\alpha^{p^r}]^{k-1}\}$$ **QED**

Fact 17: There are several related parts to this Fact: (5.38)

(a) No two distinct conjugate sets can have a common element.
(b) Each non-zero element of GF(q) appears in exactly one conjugate set.
(c) The conjugate sets form a partition of the non-zero elements of GF(q)

In these claims we are including the trivial conjugate set { 1 } as a conjugate set.

Proof: (a) Consider these two conjugate sets

$\{\alpha, \alpha^{p}, \alpha^{p^2}, \alpha^{p^3}, \ldots \alpha^{p^{k-1}}\}$	$\alpha^{p^k} = \alpha$	k conjugates	$k \le m$
$\{\beta, \beta^{p}, \beta^{p^2}, \beta^{p^3}, \ldots \beta^{p^{k'-1}}\}$	$\beta^{p^{k'}} = \beta$	k' conjugates	$k' \le m$

Suppose these sets have some common element a. Then we must have, for some i and j,

$$a = \alpha^{p^i}$$
$$a = \beta^{p^j}$$

Therefore, assuming $i \ge j$, we write

$$\alpha^{p^i} = \beta^{p^j} \quad \Rightarrow \quad \beta = \alpha^{[p^i/p^j]} = \alpha^{p^{i-j}} = \alpha^{p^r} \quad \text{where } r \equiv i-j$$

But we know from Fact 16 that if $\beta = \alpha^{p^r}$, then the first conjugate set can be written as the second conjugate set, and therefore the two sets are the same and k' = k. If i < j, repeat the argument reversing the roles of α and β. **QED**

(b) In order to form all the conjugate sets, we consider all powers α^{s} for s = 1,2,3...q-1 and check to see if the conjugate set formed from α^{s} is distinct from all the previous conjugate sets. In theory, one could end up with q-1 different conjugate sets each with one power α^{s}. More likely, however, there will be some number n < q-1 of conjugate sets and some sets will contain multiple elements. According to (a), a given power α^{s} can end up in only one conjugate set, which is the claim of (b). Putting the q-1 powers a^{s} into these n conjugate sets is like putting q-1 numbered balls into $n \le q-1$ boxes. Since a ball can only go into one box, the set of numbered balls is naturally partitioned into the n boxes. Thus, the conjugate sets form a partition of the powers α^{s}, which is the claim of (c)

Corollary 1: Two distinct minimum polynomials can have no roots in common. (5.39)

Proof: Minimum polynomials have as their roots the elements of conjugate sets, as in (5.17), and according to (5.38) (a), those conjugate sets can have no elements in common.

(h) Cyclotomic Cosets

If α is a primitive element of GF(p^m), then {GF(p^m)-0-1} is enumerated by powers α^s for s = 1,2...p^m-1. Fact 8 (5.15) tells us that the first conjugate set will have m elements of the form { α, α^p, α^{p^2}, α^{p^3} , ... $\alpha^{p^{m-1}}$ }. Subsequent conjugate sets for powers α^s will all have some number $k_s \leq m$ elements, and the < can apply only in the case that α^s is not a primitive element of GF(q). Here is one of those subsequent sets,

$$\{ (\alpha^s), (\alpha^s)^p, (\alpha^s)^{p^2}, (\alpha^s)^{p^3}, ... (\alpha^s)^{p^{k-1}} \} \qquad k \leq m$$

$$= \{ \alpha^s, \alpha^{sp}, \alpha^{sp^2}, \alpha^{sp^3} \quad , ... \alpha^{sp^{k-1}} \} . \tag{5.40}$$

We can think of this as the s^{th} attempted conjugate set. It may end up the same as some other conjugate set. Consider the set of *exponents* of this attempted conjugate set

$$\text{exponents} = \{ s, sp, sp^2, sp^3, ... sp^{k-1} \}. \tag{5.41}$$

There is an obvious 1-to-1 correspondence between the conjugate sets and the sets of such exponents. We have already shown in (5.38) (c) that the set of distinct conjugate sets partitions the set of powers α^s. Therefore, the corresponding set of distinct sets of exponents partitions the set of exponents 1 to q-1.

The set of exponents above $C_s = \{ s, sp, sp^2, sp^3, ... sp^{k-1} \}$ is called a **cyclotomic coset mod p**. We have just noted that the cyclotomic cosets are in 1-to-1 correspondence with the conjugate sets. Just as the distinct conjugate sets partition the q-1 powers α^s, so the cyclotomic cosets partition the exponents of those powers which are just integers in the range 1 to q-1. We have just proven :

Fact 18: The cyclotomic cosets mod p of GF(p^m) partition the integers 1,2....p^m-1. (5.42)

Example: In (5.26) we saw for GF(2^4) how the conjugate sets partition the powers α^0 through α^{14}.

$$\underset{\text{prim}}{\{ \alpha, \alpha^2, \alpha^4, \alpha^8 \}}, \{ \alpha^3, \alpha^6, \alpha^{12}, \alpha^9 \}, \underset{\text{prim}}{\{ \alpha^5, \alpha^{10} \}}, \{ \alpha^7, \alpha^{14}, \alpha^{13}, \alpha^{11} \}, \{1\} \tag{5.26}$$

where {1} = {α^0} = {α^{15}}. The corresponding cyclotomic coset partitioning of integers 1 to 15 is

{1,2,4,8}, {3,6,12,9}, {5,10}, {7,14,13,11}, {15} .

Since $\alpha^{p^m} = \alpha^q = \alpha$ for a primitive element of GF(p^m), we know that $\alpha^{q-1} = \alpha^0$ so that the exponent q-1 is always equivalent to the exponent 0. Thus,

Corollary: The cyclotomic cosets mod p of GF(p^m) partition the integers 0,1....p^m-2. (5.43)

Example: {1,2,4,8}, {3,6,12,9}, {5,10}, {7,14,13,11}, {0} .

(i) Least Common Multiples of minimum polynomials

Suppose we multiply together the minimum polynomials of some *subset* of the elements of GF(q). It is quite possible that the same minimum polynomial will appear more than once in this product. Assume that we remove all duplicate copies and end up with a polynomial of this form, where all the m_i are distinct,

$$F(x) = m_1(x)m_2(x).....m_j(x) .$$

This is sometimes written in this notation

$$F(x) = LCM[m_1(x)m_2(x).....m_j(x)]$$

where LCM means **Least Common Multiple**, which just means duplicate m_i are removed. We know from Fact 17 that no root of GF(q) can appear more than once in this product. Therefore, the degree of F(x) must be ≤ q. Recall now the result of Big Theorem 2 (4.34) that the product of *all* the (x-α) factors of GF(q) is given by

$$(x^q - x) = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_q) .$$

Since each $m_i(x)$ includes some subset of these $(x-a_i)$ factors, if follows that F(x) includes some larger subset of the $(x-a_i)$ factors, and therefore both the $m_i(x)$ and F(x) must divide evenly into x^q - x. We have just proven this Fact:

Fact 18: If $F(x) = LCM[m_1(x)m_2(x).....m_j(x)]$ for some subset of the elements of GF(q), then the degree of F(x) is ≤ q, no root of GF(q) is repeated more than once in F(x), and F(x) divides evenly into x^q - x, as do each of the $m_i(x)$. (5.44)

Here is the same Fact restricted to non-zero elements of GF(q):

Fact 19: If $G(x) = LCM[m_1(x)m_2(x).....m_j(x)]$ for some subset of the *non-zero* elements of GF(q), then the degree of G(x) is ≤ q-1, no root of GF(q) is repeated more than once in G(x), and G(x) divides evenly into x^{q-1} - 1, as do each of the $m_i(x)$. (5.45)

Proof: According to Corollary 1 (5.39), no root of GF(q) is repeated more than once in G(x), just as with F(x) of Fact 18. The minimum polynomial of α = 0 is just (x-0) = x. If we divide this out of the product shown above, we get

$$(x^{q-1} - 1) = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_{q-1}) ,$$

where we have assumed that $a_q = 0$. Since there are now only q-1 roots available for G(x), we know that the degree of G(x) is ≤ q-1. Since all $m_i(x)$ as well as G(x) are constructed from subsets of the $(x-a_i)$ factors shown, we know that all these $m_i(x)$ as well as F(x) divide evenly into $x^{q-1} - 1$.

Corollary: If $F(x) = LCM[m_1(x)m_2(x).....m_j(x)]$ for *all* the elements of GF(q), then $F(x) = x^q - x$. If m(x) = x of the 0 element of GF(q) is omitted, then $F(x) = x^{q-1} - 1$. (5.46)

Proof: Each $m_i(x)$ includes a factor (x-α) for its own α. If we include all $m_i(x)$ in F(x), then all α are included in (x-α) factors, and thus each factor is included once and the product is then $x^q - x$. If F(x) is a product for only non-zero α $m_i(x)$, then m(x) = x is omitted, and F(x) is then $x^{q-1} - 1$.

(j) Order = Period Theorem for a minimum polynomial

First we set up the context for this theorem. Recall these earlier definitions:

order of β in GF(q) = smallest integer N such that $\beta^N = 1$ *(4.17) (k)*

period of minimum polynomial h(x) = smallest integer n such that $(x^n-1)/h(x) = g(x)$ *(5.9)*

Note: Peterson and Weldon use the term "exponent" instead of period. Here is a linguistic translation:

n is the period of h(x) ↔ n is the exponent to which h(x) belongs

A minimum polynomial h(x) for α in GF(q) of degree k has the form,

$h(x) = (x-a_1)(x-a_2)....(x-a_k)$

where α is one of these a_i and where the symbols $\{a_1, a_2....a_k\}$ form the conjugate set of h(x). We know from Fact 15 (5.35) that all elements of a conjugate set have the same order which we shall denote here as N. So here is our claimed theorem:

Fact: If h(x) is any minimum polynomial of GF(q) of degree k, then the *order* of any member of the conjugate set of h(x) equals the *period* of h(x). (5.47)

Proof: As just stated above:

(1) All members of a conjugate set have the same order, which we shall call N;

(2) The order of some β in GF(q) is the smallest integer N for which $\beta^N = 1$;

(3) The period, on the other hand, is the smallest integer n such that h(x) divides evenly into (x^n-1), so $(x^n-1)/h(x)$ = some polynomial g(x) with no remainder.

The k elements a_i of the conjugate set of h(x) comprise all the roots of h(x), so $h(a_i) = 0$ for any conjugate set element. Since these are *all* the roots of h(x), if one finds that h(β) = 0, then β must be one of the conjugate set elements. We know that h({x}) = 0 because we know {h(x)} = 0 since there is no remainder when h(x) is divided by h(x) in the representation $GF(p^m) = R/h(x)$. Since h({x}) = 0, we conclude that {x} must be one of those a_i conjugate set elements. Therefore, the order of {x} is N. Since

the elements of a conjugate set are really all on an equal footing, let's just call this a_1 and so we have then $\alpha = a_1 = \{x\}$. Now consider these steps:

$g(x)h(x) = (x^n-1)$ // period n is the smallest integer that makes this possible

$g(\{x\})\, h(\{x\}) = (\{x\}^n-1)$ // apply {...} to both sides and use the rules of (3.14)

$g(\alpha)h(\alpha) = (\alpha^n-1)$ // {x} has name α

$0 = (\alpha^n-1)$ // since $h(\{x\}) = h(\alpha) = 0$

$\alpha^n = 1$.

Thus, we see that n is a *candidate value* for N, the order of {x} = α and all other conjugate set elements. Since we are finding here that $\alpha^n = 1$ for period n, we certainly know that the order $N \leq n$ since N is supposed to be the smallest possible integer for which $\alpha^N = 1$. Thus we have ruled out N > n.

We shall now also rule out N < n and conclude therefore that N = n and our Fact is proven.

Suppose N = order of α = {x} and n = period of h(x) with N < n. Since the order is less than the period, we must get a non-zero remainder r(x) when (x^N-1) is divided by h(x),

$(x^N-1)/h(x) = q(x) + r(x)/h(x)$ or $(x^N-1) = q(x)h(x) + r(x)$ where $r(x) \neq 0$.

This says that $(\{x\}^N-1) = r(\{x\})$ or $(\alpha^N - 1) = r(\alpha)$ where α = {x}. But since N is the order of α, we know that $\alpha^N = 1$ so we then find that r(α) = 0, but $r(x) \neq 0$, and the degree of r(x) is < k.

We assumed that h(x) was the minimum polynomial for α. Thus, h(x) has this form:

$h(x) = (x-\alpha)(x- a_2)(x-a_3).....(x-a_k)$ h(x) in R, all a_i in GF(q), deg h(x) = k,

Since r(α) = 0, it has this form:

$r(x) = (x-\alpha)\,(x-b_1)(x-b_2).....$ r(x) in R, b_j = unknown but in GF(q), deg r(x) < k

Both h(x) and r(x) have the form for a minimum polynomial of α, but since r(x) has the lower degree, it must be the true minimum polynomial of α which then has degree < k. But this contradicts our original assumption that h(x) is the minimum polynomial of α and has degree k. Therefore we cannot have N < n.

The conclusion is that n = N so the period n of a minimum polynomial is the same as the order N of any of the elements of the conjugate set of that minimum polynomial. **QED**

Comment: If irreducible h(x) in R degree k < m is *not* a minimum polynomial of GF(q) (white region of Fig 5.5) , then it cannot be written in the usual factored form shown above, and therefore it cannot divide into $x^n - 1$ for any n, even for n = q-1. Such a polynomial therefore has no period whatsoever. Of course it

also has no conjugate set, so we cannot talk about the order of conjugate set elements. The point is just that if h(x) is not a minimum polynomial, this Fact (5.47) makes no sense at all.

Consistency Observation: We showed in (5.35) that all elements of a conjugate set of a minimum polynomial h(x) have the same order. In retrospect, now that we know that the order of an element α of the conjugate set of h(x) is the period of h(x), and since h(x) can only have one period, the elements of the conjugate set must have the same order.

(k) Maple code to compute all minimum and primitive polynomials for any GF(p^m)

Our method has been described in section 5 (c) 3 above. Here then is self-documented Maple code which carries out the method for an arbitrary Galois Field GF(p^m). We first show the code and then show the output for all the cases already treated above and a few new cases as well. This code produces output lines like [5,10], $(x-\alpha^5)(x-\alpha^{10})$. It does not multiply out the polynomial to get something like $x^2 + x + 1$. Doing that requires the selection of a specific primitive polynomial as shown later in Section 6 (d) 2. One could modify the code below allowing the user to enter a specific primitive polynomial and then each output line could have the form [5,10], $(x-\alpha^5)(x-\alpha^{10})$, $x^2 + x + 1$ and this could be put into a fancy spreadsheet. For now, we just want to see all the minimum polynomials in their factored form with the primitive ones marked. The code shown happens to have p = 2 and m = 6.

Program to compute all minimum polynomials of GF(p^m)
Output format is: conjugate set exponents, factored minimum polynomial, order=period

```
restart; with(numtheory): # numtheory only for Euler phi use.
PrimID := 1e12:  # use this to mark a prim poly so sorts last in list
p := 2: m:= 6:   # enter desired values here
q := p^m: printf("\n\np = %d, m = %d, q-1 = %d",p,m,q-1);
printf("Number of Primitive Polynomials is %d",phi(p^m-1)/m);

first := true:
for s from 1 to q-2 do  # for each GF(q) element (alpha)^s ...
  # create array e of m conjugate set exponents as shown in Sec 5 (c) 3
  for n from 1 to m do e[n] := (p^n*s) mod (q-1); od:
  # convert this array to a set so dupe exponents get thrown out (Fig 5.1)
  eset := convert(e,set);
  # test for primitive poly as per (4.32), if yes add PrimID to set
  if gcd(s,q-1) = 1 then eset := eset union {PrimID}; fi;
  # first time seta is something like {{3,6,7}}, a set of one set.
  if first then seta := {eset};first := false;
    # otherwise add a new set to set of sets e.g. {{3,6,7},{2,6,PrimID}}
    # this operation removes dupe conjugate sets (!) (Fig 5.2)
    else seta := seta union {eset};
  fi;
od:

# at this point we have seta = {{3,6,7},{2,6,PrimID}, ....}
printf("Number of Minimum Polynomials is %d",nops(seta));

for cs in seta do   # for each conjugate set cs in seta ...
  # convert set to a list so it can be sorted, then sort it
  cs1 := sort(convert(cs,list)):
  L := nops(cs1);  # number of elements in cs1
  if cs1[L] = PrimID then cs1[L] := Prim; L := L-1; fi;
  # construct the polynomial h(x) with exponents in cs1
  first := true;
  for n from 1 to L do
    if first then h := (x-a^cs1[n]); first := false;
    else h := h*(x-a^cs1[n]); fi:
  od:
  ord := (q-1)/gcd(cs[1],q-1): # per(minpoly)= order of any cs element (4.32a)
  print(cs1,h,ord);  # show the conj set exponent list, min poly, its period
od:
```

Here are some sample runs of the above code :

```
p = 2, m = 2, q-1 = 3
Number of Primitive Polynomials is 1
Number of Minimum Polynomials is 1
```

$$[1, 2, Prim], (x-a)(x-a^2), 3$$ // must be x^2+x+1 from Fig 5.29 (5.48)

```
p = 2, m = 3, q-1 = 7
Number of Primitive Polynomials is 2
Number of Minimum Polynomials is 2
```

$$[1, 2, 4, Prim], (x-a)(x-a^2)(x-a^4), 7$$

$$[3, 5, 6, Prim], (x-a^3)(x-a^5)(x-a^6), 7$$

// agrees with (5.24) (5.49)

```
p = 2, m = 4, q-1 = 15
Number of Primitive Polynomials is 2
Number of Minimum Polynomials is 4
```

$$[7, 11, 13, 14, Prim], (x-a^7)(x-a^{11})(x-a^{13})(x-a^{14}), 15$$

$$[5, 10], (x-a^5)(x-a^{10}), 3$$

$$[1, 2, 4, 8, Prim], (x-a)(x-a^2)(x-a^4)(x-a^8), 15$$

$$[3, 6, 9, 12], (x-a^3)(x-a^6)(x-a^9)(x-a^{12}), 5$$

// agrees with (5.27) (5.50)

```
p = 2, m = 5, q-1 = 31
Number of Primitive Polynomials is 6
Number of Minimum Polynomials is 6
```

$$[15, 23, 27, 29, 30, Prim], (x-a^{15})(x-a^{23})(x-a^{27})(x-a^{29})(x-a^{30}), 31$$

$$[1, 2, 4, 8, 16, Prim], (x-a)(x-a^2)(x-a^4)(x-a^8)(x-a^{16}), 31$$

$$[3, 6, 12, 17, 24, Prim], (x-a^3)(x-a^6)(x-a^{12})(x-a^{17})(x-a^{24}), 31$$

$$[5, 9, 10, 18, 20, Prim], (x-a^5)(x-a^9)(x-a^{10})(x-a^{18})(x-a^{20}), 31$$

$$[7, 14, 19, 25, 28, Prim], (x-a^7)(x-a^{14})(x-a^{19})(x-a^{25})(x-a^{28}), 31$$

$$[11, 13, 21, 22, 26, Prim], (x-a^{11})(x-a^{13})(x-a^{21})(x-a^{22})(x-a^{26}), 31$$

// agrees with (5.28) (5.51)

Here is a case we have not done before:

```
p = 2, m = 6, q-1 = 63
Number of Primitive Polynomials is 6
Number of Minimum Polynomials is 12
```

$$[21, 42], (x-a^{21})(x-a^{42}), 3$$
$$[15, 30, 39, 51, 57, 60], (x-a^{15})(x-a^{30})(x-a^{39})(x-a^{51})(x-a^{57})(x-a^{60}), 21$$
$$[23, 29, 43, 46, 53, 58, Prim], (x-a^{23})(x-a^{29})(x-a^{43})(x-a^{46})(x-a^{53})(x-a^{58}), 63$$
$$[13, 19, 26, 38, 41, 52, Prim], (x-a^{13})(x-a^{19})(x-a^{26})(x-a^{38})(x-a^{41})(x-a^{52}), 63$$
$$[11, 22, 25, 37, 44, 50, Prim], (x-a^{11})(x-a^{22})(x-a^{25})(x-a^{37})(x-a^{44})(x-a^{50}), 63$$
$$[9, 18, 36], (x-a^{9})(x-a^{18})(x-a^{36}), 7$$
$$[7, 14, 28, 35, 49, 56], (x-a^{7})(x-a^{14})(x-a^{28})(x-a^{35})(x-a^{49})(x-a^{56}), 9$$
$$[5, 10, 17, 20, 34, 40, Prim], (x-a^{5})(x-a^{10})(x-a^{17})(x-a^{20})(x-a^{34})(x-a^{40}), 63$$
$$[1, 2, 4, 8, 16, 32, Prim], (x-a)(x-a^{2})(x-a^{4})(x-a^{8})(x-a^{16})(x-a^{32}), 63$$
$$[3, 6, 12, 24, 33, 48], (x-a^{3})(x-a^{6})(x-a^{12})(x-a^{24})(x-a^{33})(x-a^{48}), 21$$
$$[31, 47, 55, 59, 61, 62, Prim], (x-a^{31})(x-a^{47})(x-a^{55})(x-a^{59})(x-a^{61})(x-a^{62}), 63$$
$$[27, 45, 54], (x-a^{27})(x-a^{45})(x-a^{54}), 7 \qquad (5.52)$$

Comment: As will be shown in Chapter 6 (d) item 3, it is an easy matter to display a minimum polynomial written in its normal form as a polynomial with coefficients in GF(p). For example, using the primitive polynomial $x^6+x^4+x^3+x+1$ for GF(2^6) taken from Fig 5.3, the last polynomial shown above becomes just $x^6+x^5+x^2+x+1$, appropriately of degree 6 :

```
alias(a=RootOf(x^6+x^4+x^3+x+1));
```
$$I, a$$
```
p := (x-a^11)*(x-a^22)*(x-a^25)*(x-a^37)*(x-a^44)*(x-a^50);
```
$$p := (x-a^{11})(x-a^{22})(x-a^{25})(x-a^{37})(x-a^{44})(x-a^{50})$$
```
sort(subs(a=alpha,simplify(p)) mod 2);
```
$$x^6+x^5+x^2+x+1$$

Here are some low-m runs with p = 3, 5 and 7. One can compare these results to section 5 (c) 4 above by identifying a primitive element, as shown in the last case below.

```
p = 3, m = 1, q-1 = 2
Number of Primitive Polynomials is 1
Number of Minimum Polynomials is 1
```
$$[1, Prim], x-a, 2 \qquad (5.53)$$

```
p = 3, m = 2, q-1 = 8
Number of Primitive Polynomials is 2
Number of Minimum Polynomials is 4
```

$$[5, 7, Prim], (x - a^5)(x - a^7), 8$$
$$[4], x - a^4, 2$$
$$[1, 3, Prim], (x - a)(x - a^3), 8$$
$$[2, 6], (x - a^2)(x - a^6), 4 \qquad (5.54)$$

```
p = 5, m = 1, q-1 = 4
Number of Primitive Polynomials is 2
Number of Minimum Polynomials is 3
```

$$[1, Prim], x - a, 4$$
$$[2], x - a^2, 2$$
$$[3, Prim], x - a^3, 4 \qquad (5.55)$$

```
p = 7, m = 1, q-1 = 6
Number of Primitive Polynomials is 2
Number of Minimum Polynomials is 5
```

$$[5, Prim], x - a^5, 6$$
$$[3], x - a^3, 2$$
$$[1, Prim], x - a, 6$$
$$[2], x - a^2, 3$$
$$[4], x - a^4, 3 \qquad (5.56)$$

For GF(7) we saw in 5 (c) 4 that " The minimum polynomials are (x-2), (x-3), (x-4), (x-5), (x-6) but only (x-3), (x-5) are primitive." and that 3 and 5 are primitive elements. Picking then α = 3 we find :

(x-α) = (x-3) Prim
(x-α^2) = (x-2) since $3^2 = 9 = 2$
(x-α^3) = (x-6) since $3^3 = 9*3 = 2*3 = 6$
(x-α^4) = (x-4) since $3^4 = 9*9 = 2*2 = 4$
(x-α^5) = (x-5) since $3^5 = 9*9*3 = 2*2*3 = 12 = 5$ Prim

Chapter 6: The GF(q) Enumeration Table

In Chapter 5 we spent much time developing the notion of a primitive polynomial. In this chapter, we show why primitive polynomials are useful. They let us make a concise table to enumerate all elements of GF(q) as both powers of some α, and as m-tuples. From this enumeration table, the • and + field operation tables can be immediately derived. We then show how Maple can be harnessed to create the enumeration table for GF(q) and to carry out various other Galois field related tasks. Although the enumeration table can then be used to compute the field + operation table, we do not carry out that task in Maple, we just indicate how it can be done.

(a) Development History of the Primitive Polynomial

It is now time for a glance back at the development of the previous chapters. We wish here to trace the thread of our voyage through the thick forest of Facts, Theorems, Lemmas, Definitions and Proofs. If the reader is not comfortable with the vocabulary appearing in this section (terms such as group, ring, order, period, cyclic, generator, primitive element, residue class ring, primitive and minimum and irreducible polynomials, etc.) then it is not yet time to proceed. In other words, this is a good time to do a solid review of the first five Chapters before the forest voyage continues. We are almost done with the Theory and we want to apply that theory to some Applications.

Chapter 1 provided the underpinning mathematical formalisms, especially the idea of a residue class ring formed as R/I where I is some ideal in a ring R.

In Chapter 2 the connection was made between GF(p) and Z_p , the field of integers modulo p. Once this connection was made, we knew at once how to construct the + and • operation tables for GF(p). However, some facts about GF(q) which applied to GF(p) as a special case were not yet developed.

We knew that the study of $GF(p^m)$ required the use of polynomials, so this was the main topic of Chapter 3. The grand connection was made that every extended Galois field could be identified with a certain residue class ring,

$$GF(p^m) = R/(f(x)) \qquad (3.17)$$

where f(x) was any irreducible polynomial in R of degree m. We know in retrospect that there are usually many choices for f(x), some of which are primitive polynomials, and some of which are not. We know that changing from one f(x) to another does not change the structure of $GF(p^m)$ (the • and + tables), it just alters the "basis", which is to say, it alters the way the elements are named. Formally speaking, the different versions of $GF(p^m)$ obtained by changing f(x) are all isomorphic to each other. The R/I structure with some f(x) is just a representation of the abstract $GF(p^m)$ Galois field.

Then came Chapter 4 where most of the "facts" about $GF(p^m)$ were painfully extracted and proved. It was like pulling teeth. The m-tuple notation was introduced as a convenient way to label the p^m remainder polynomials which are in effect the elements of $GF(p^m)$ in its R/I representation. It was noted that in the m-tuple basis, the + table is trivial to construct, but the • table is painful to construct, requiring many long divisions to find remainder polynomials.

We then considered the cyclic subgroups of GF(q) and were able to derive many facts about GF(q). For example, each element of {GF(q) - 0) is in a cyclic subgroup of some order n. We learned that there is always at least one primitive element which has order q-1 and whose powers can be used to enumerate the elements of {GF(q) -0}. In other words, we learned that {GF(q)-0,•} is cyclic.

This suggested another "basis" for labeling the elements of {GF(q)-0}, namely, the "powers of α" basis, where α is some primitive element. In this basis, we noted that the • table was trivial to write down, but the + table was painful to develop. We compared the two bases for GF(2^2) toward the end of Chapter 4.

We then learned that the little polynomial x^{q-1}- 1 can be exploded into a product of linear factors, each of which contains an element of GF(q). For the first time, this caused us to think about the idea that a polynomial might have roots in GF(q) due to these factors (x-a). Soon, we became comfortable with the idea of a polynomial having some roots in GF(q).

If we look back now at Chapter 3 on polynomials, there was no concept of a polynomial f(x) having an element α of GF(q) as a *root*, which is to say f(α) = 0. In Chapter 3, polynomials were what lived in the chart rows of the R/I residue class ring. The rows of this chart *were* the elements of GF(q), but we did not consider to write f(some row) = 0, where "some row" was a root of f. It was not reasonable to think of our defining f(x) as having roots in Chapter 3 because after all, f(x) was supposed to be irreducible over GF(p) which meant you could not factor it, so it seemed there were no roots at all. We later realized that any polynomial f(x) defined over the field GF(p) can be extended to have a definition over GF(q), and it is here that an "irreducible in GF(p)" polynomial might have roots. As will be seen below, it is perfectly reasonable to observe that f({x}) = 0 when f(x) is the R/I defining polynomial, and where {x} is the chart row containing the polynomial x.

In Chapter 5 we defined the minimum polynomial m(x) of some α in GF(q) to be the smallest set of (x-a) factors which contains the factor (x-α) and which has coefficients in the ground field GF(p). We then noted that m(x) has several roots in GF(q) -- the elements of the conjugate set of α. The exact number of roots is not known in general, it is some k ≤ m where m is the m of q = p^m.

Finally, we said that if α is a primitive element of GF(q), then minimum polynomial m(x) gets the special name of being a primitive polynomial of GF(q). We noted several facts about such primitive polynomials p(x) : they are monic and irreducible in GF(p), p(α) = 0 for some primitive element of GF(q), they are of degree m for GF(q=p^m), and they have period = q-1.

We considered briefly how one might search for the primitive polynomials p(x) of GF(p^m). We know for sure that there is *at least one* primitive polynomial for any choice of p and m, and there are generally more than one. This is simply because there is likely to be more than one cyclic generator of {GF(q)-0}. One got the impression in Chapter 5 that the process of searching for the p(x) for GF(q) could be automated in some way. Figure 5.3 listed one primitive polynomial for p=2 and m = 2 to 32, taken from a web reference that goes up to m = 1256.

Having done all this work, the reader must wonder: Why are primitive polynomials useful? Where is the payoff?

(b) Using a Primitive Polynomial as the f(x) in GF(q) = R/(f(x))

The material of this section was treated in an introductory fashion in Chapter 4 (b) for the field GF(2^2). Here we formalize that treatment. Our context is the identification noted above,

$$GF(p^m) = R/(f(x)) \qquad (3.17)$$

where f(x) is certainly allowed to be a primitive polynomial of GF(p^m), since any primitive polynomial p(x) is irreducible in R and has degree m. We shall now learn that there is a great advantage in making this selection for f(x). We shall show this by first *not* doing so.

f(x) *could be* a primitive polynomial

We know that we can represent the elements of GF(q) as m-tuples consisting of m integers, where each integer lies in Z_p = GF(p). And each such m-tuple is a shorthand notation for a GF(q) remainder polynomial of degree less than m (relative to the defining polynomial f(x)) which characterizes a row of the R/I chart, that is to say, which characterizes an element of GF(q). For example,

m-tuple = <1ab....> $\qquad$ {r(x)} = {1 + ax + b x^2 + ...} $\qquad$ 1,a,b $\in Z_p$.

Let us now single out one very simple m-tuple:

$$\alpha = \{x\} = <0100....> = ...00010 \qquad (6.1)$$

Thus, as defined right here, α is the element of GF(q) which corresponds to the R/I chart row {x} which contains the polynomial x. Assume that we use some generic degree-m irreducible f(x) to define our R/I chart. We can then start building a table of powers of α as follows: (assume p = 2, m = 4)

1	{1}	0001	
α	{x}	0010	
α^2	{x^2}	0100	
α^3	{x^3}	1000	
α^4	{x^4}	????	
.....			
α^{15}	???	????	(6.2)

How does this work? Recall from Section 1 (c) 2 that the product of two rows in the R/I chart gives a third row, and that the polynomials in that third row will be the products of the polynomials in the two rows, modulo remaindering which we ignore for the moment, thinking m is large. Thus, if we multiply row {x} which contains the polynomial x by itself, we get a new row, and that new row will be the one which contains x^2, which row is {x^2}. Therefore if row {x} has the name α, then {x^2} is {x}•{x} = {x}^{2} = α^2, where α^2 is field element α squared. Thus is built the above table, at least through {x^4} since we assumed m = 4 for illustration. See (3.14) for general {} usage.

When we reach the power 4, we have to do a remainder calculation. Assume that f(x) = 1 + x^2 + x^4, which happens *not* to be a primitive polynomial for GF(2^4). Then Rem(x^4/f) = 1 + x^2,

```
Rem(x^4,1+x^2+x^4,x) mod 2;
```

$$x^2+1$$

and this lets us continue with the chart:

α^4 $\{x^4\} = \{x^2 + 1\}$ 0101 (6.3)

The next row will be $\alpha^5 = \alpha\,(\alpha^4) = \{x\}\bullet\{1+x^2\} = \{x + x^3\}$, so

α^5 $\{x^5\} = \{x^3 + x\}$ 1010 (6.4)

In this manner, we can build up all powers through α^{15}. The job of building the chart is really quite simple and fast. In fact, one does not have to compute any remainders due to the following fact, which may not have been obvious before now:

Fact 1: $f(\alpha) = 0$ for $\alpha = \{x\}$. { α *may* or may *not* be a primitive element of GF(q) } (6.5)

Proof: Assume that $f(x) = 1 + x^2 + x^4$ is the polynomial of degree 4 which defines our R/I residue class ring. Then write $\{f(x)\} = \{ 1 + x^2 + x^4 \}$. Since the polynomial $1 + x^2 + x^4$ has no remainder when divided by itself, we must have that $\{ 1 + x^2 + x^4 \} = \{0\}$, which is the first row of the chart (3.13) which contains all polynomials with zero remainder. This row is identified with element **0** of GF(q). Thus we have shown that $\{f(x)\} = \{0\} = \mathbf{0}$. Now, according to (3.14) (f), we know that $\{f(x)\} = f(\{x\})$. Therefore we have shown that

$f(\{x\}) = \{f(x)\} = \{0\} = \mathbf{0}$.

If we use the name $\alpha = \{x\}$, this says

$f(\alpha) = \{0\} = \mathbf{0} = 0$ where $\alpha = \{x\}$ **QED**

It is understood that 0 means **0**, the 0 element of the base field GF(p), so we stop bolding it. Perhaps to be more consistent we should be saying $\mathbf{f}(\alpha) = \mathbf{0}$ and $f(\alpha) = 0$ as the two notational choices. And then since α is an element of GF(q), one might choose to write **α** instead. The reader was warned earlier about our planned inconsistent use of bolded-font objects.

Given this Fact, for our $f(x) = 1 + x^2 + x^4$, we have $f(\alpha) = 1 + \alpha^2 + \alpha^4 = 0$, so $\alpha^4 = -\alpha^2-1$ which is the same as $\alpha^4 = \alpha^2 + 1$, since + and - are the same in GF(2). If $\alpha = \{x\}$, then this tells us that

$\{x^4\} = \{x^2\} + \{1\} = \{x^2 + 1\}$

which gives (6.3) with no remainder calculations needed. Similarly $\alpha^5 = \alpha^3 + \alpha$ gives (6.4)

$\{x^5\} = \{x^3\} + \{x\} = \{x^3 + x\}$ // which is (6.4)

In this way we can quickly finish the table (6.2) started above.

Question: Have we in this manner enumerated all elements of GF(q) - 0 ?

Answer: No! At least there is no guarantee. How do we know that we "hit" all q-1 = 15 distinct m-tuples by this construction? If α as defined above happens to be a primitive element of GF(q), then all the m-tuples will be hit, because by definition, α would then be a generator of the full group {GF(q) - 0,•}. This is our motivation.

f(x) *is* a primitive polynomial

Fact 2: If we choose f(x) to be a *primitive* polynomial for GF(q), then we can enumerate all elements of the set {GF(q)-0} using any root of f(x), including the root {x}. This is because all roots of f(x) are primitive elements of GF(q). (6.6)

Proof: Saying that f(x) is a primitive polynomial says that f(x) is the minimum polynomial for some primitive element α of GF(q), meaning α is a generator of {GF(q)-0}. We know from Big Theorem 3 of (5.17) that all the roots of a minimum polynomial are in the conjugate set of α. But from Fact 8 (b) of (5.15) we know that all elements of the conjugate set of α are primitive elements. Therefore, all the roots of f(x) are primitive elements and any can be used to enumerate {GF(q)-0}. Since f({x}) = 0 by Fact 1, we know that {x} is a primitive element and can therefore be used to enumerate {GF(q)-0}.

In general, there is no reason one would imagine that the (row of the) remainder function r(x) = x and its m-tuple representation ...000010 would be a primitive element of GF(q). This would be an amazing coincidence, one would think offhand. It turns out, as shown above, that by using a primitive polynomial as the defining f(x), this is exactly what happens.

We have now proved the following claim, which summarizes the content of this section:

Fact 3: If the GF(q) = R/(f(x)) defining polynomial f(x) of degree m is selected to be p(x), a primitive polynomial of GF(q), then the remainder function r(x) = x which belongs to chart row {x}, and which is represented by the m-tuple ...000010 , labels a primitive element α of GF(q). All other elements of {GF(q) - 0} can therefore be represented as powers of this primitive element α. (6.7)

Constructing the Enumeration Table for GF(q)

We now return to the details of constructing the table. Going back to the general GF(p^m), we know that our primitive polynomial p(x) is monic and of degree m, so write it as follows,

$$p(x) = x^m + c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + ... + c_3x^3 + c_2x^2 + c_1x + c_0 \quad (6.8)$$

where the coefficients c_i are elements of GF(p). Note that $c_0 \neq 0$ because otherwise x could be factored out and p(x) would not be irreducible. We know from (6.5) that p(α) = 0, where α equals {x}, so

$$\alpha^m + c_{m-1}\alpha^{m-1} + c_{m-2}\alpha^{m-2} + ... + c_3\alpha^3 + c_2\alpha^2 + c_1\alpha + c_0 = 0,$$

and we know from (6.6) that α is a primitive element of GF(p^m). Solving the above for α^m,

$$\alpha^m = -\,[\, c_{m-1}\alpha^{m-1} + c_{m-2}\alpha^{m-2} + \ldots + c_3\alpha^3 + c_2\alpha^2 + c_1\alpha + c_0]$$

$$= (p-c_{m-1})\,\alpha^{m-1} + (p-c_{m-2})\,\alpha^{m-2} + \ldots + (p-c_1)\,\alpha + (p-c_0)\,. \qquad (6.9)$$

In the second line we have used the fact (1.17) that -a = (p-a) for a lying in GF(p). Thus, we are able to *replace* α^m with a sum of lower powers of α. This equation exists in the space GF(q), not GF(p). We are writing the GF(q) element α^m as a sum of other elements of GF(q). The coefficients are still in GF(p). To get α^{m+1} multiply both sides of (6.9) by α, then in the first term on the right use (6.9) to replace α^m. Keep going in this manner to obtain all the powers of α. For any power we will therefore obtain a result of the form

$$\alpha^n = A_{m-1}\,\alpha^{m-1} + A_{m-2}\alpha^{m-2} + \ldots + A_1\alpha + A_0 = \Sigma_{i=0}^{m-1}\, A_i\alpha^i\,. \qquad (6.10)$$

So now the construction of a complete table for GF(p^m) is a straightforward and fast mechanical procedure once some primitive polynomial p(x) is identified. There are no remainder division computations needed, one just keeps re-using (6.9) to reduce things to degree less than m.

Reminder: Equation (6.9) is the statement p(α) = 0 for some known primitive polynomial p(x).

Here then is a summary of how to compute an enumeration table for GF(q):

- find a primitive element α and its minimum polynomial p(x). You then know the c_i in (6.9).
- create a table with powers α^n going down the left edge
- the powers $\alpha, \alpha^2 \ldots \alpha^{m-1}$ stand by themselves
- use (6.9) to reduce higher powers of α to linear combinations of the first m table entries

The case GF(2^2) is worked out in section (e) below. It seems more useful here to start with a more substantial case.

Example GF(2^3)

From the primitive polynomial list in Fig 5.3 we find $p(x) = 1 + x + x^3$. We proved this is a primitive polynomial for GF(8) in (5.30). In terms of (6.8) we have $c_3 = c_1 = c_0 = 1$. Then (6.9) says that
$\alpha^3 = (2-c_2)\alpha^2 + (2-c_1)\alpha + (2-c_0) = (2)\alpha^2 + \alpha + 1 = \alpha + 1$ since $p\boldsymbol{\alpha} = 0$, (4.44). So the reduction rule (6.9) states that $\alpha^3 = \alpha + 1$. This is of course totally obvious just setting p(α) = 0 for the above p(x), remembering + and - are the same, but we wanted to see how (6.9) gets the right answer. So we can build the entire enumeration table in 1 minute. We now start using the m-tuple notation <...> which puts the highest power on the left.

0	000	0		
$\alpha^0 = 1$	001	1		
$\alpha^1 = \alpha$	010	2		
α^2	100	4		
$\alpha^3 = \alpha + 1$	011	3		
$\alpha^4 = \alpha^2 + \alpha$	110	6	$\alpha^4 = \alpha\,\alpha^3 = \alpha(1+\alpha) = \alpha + \alpha^2$	
$\alpha^5 = \alpha^2 + \alpha + 1$	111	7	$\alpha^5 = \alpha\,\alpha^4 = \alpha^2 + \alpha^3 = \alpha^2 + \alpha + 1$	
$\alpha^6 = \alpha^2 + 1$	101	5	$\alpha^6 = (\alpha^3)^2 = (1+\alpha)^2 = 1 + \alpha^2$	(6.11)

Notice that all 3-tuples are "hit", as claimed above. We have added some octal/hex numbers to make this fact more obvious. Also we know that $\alpha^7 = 1$, since the non-zero elements of GF(8) form a cyclic group under • for a primitive element like α (which is a root of primitive polynomial p(x)) . This fact $\alpha^7 = 1$ was not needed to build the above table, but it is needed to construct the • table.

Question: Why is this construction so useful?

Answer: The reason is that it provides an immediate + addition table. Recall from our earlier discussion that in the powers-of-α basis, the • table was trivial, but the + table was difficult to compute. With the above construction, the • table is still trivial, but the + table is now almost as trivial! Here are some sample computations for GF(2^3) :

• table: $\alpha^5 \bullet \alpha^6 = \alpha^{11} = \alpha^4$ and so on, not much more to say, $\alpha^7 = 1$

The • table for GF(2^m) could be constructed using this trivial function,

```
mult := proc(n1,n2,m) alpha^(n1+n2 mod (2^m-1)) end:
mult(5,6,3);
```

$$\alpha^4$$

+ table:
$\alpha^5 + \alpha^6 = (\alpha^2 + \alpha + 1) + (\alpha^2 + 1) = \alpha$ // remember that 2**g** = 0 for **g** in GF(2^m)
$\alpha^5 + \alpha^4 = (\alpha^2 + \alpha + 1) + (\alpha^2 + \alpha) = 1$
$\alpha^5 + \alpha^3 = (\alpha^2 + \alpha + 1) + (\alpha + 1) = \alpha^2$
$\alpha^5 + \alpha^2 = (\alpha^2 + \alpha + 1) + (\alpha^2) = 1 + \alpha = \alpha^3$
etc.

So by choosing a primitive polynomial p(x) to develop the extension field,

$GF(p^m) = R / (p(x))$,

we can easily construct both the • and + tables for GF(p^m) where elements are listed as powers or α, the primitive element for which p(α) = 0.

Note: We are using the word "table" to refer to two different objects. One object is the addition or multiplication table just discussed, such as appear in (4.41) for GF(2^2). The other object is the enumeration table of GF(q) which puts information like (6.11) into a table as shown below.

Comment: Each element of a field is supposed to have an additive inverse in the field. When p = 2, this is a very boring thing to prove. For example, $-\alpha^3 = -1-\alpha = 1+\alpha = \alpha^3$ and in general $-\alpha^i = \alpha^i$. This situation applies in fact to any GF(2^m). For the case GF(3^2) treated below, thing are less trivial.

It is traditional to put the data of (6.11) into a table. The entire table is based on a particular choice of a primitive polynomial, which in our example above is $p(x) = 1 + x + x^3$, which implies $\alpha^3 = \alpha + 1$, which is (6.10), the "replacement rule". So here is the table version of (6.11):

Table for GF(2^3) $\quad \alpha^7 = 1 \quad$ (6.9) m=3 $\alpha^3 = \alpha + 1$ $\quad$ (6.10) m=3 $\alpha^n = A_2\alpha^2 + A_1\alpha + A_0$

n	α^n	A_2	A_1	A_0	hex	
1	α^1	0	1	0	2	
2	α^2	1	0	0	4	
3	α^3	0	1	1	3	// $\alpha + 1$
4	α^4	1	1	0	6	// $\alpha^2 + \alpha$
5	α^5	1	1	1	7	// $\alpha^2 + \alpha + 1$
6	α^6	1	0	1	5	// $\alpha^2 + 1$
7,0	α^7	0	0	1	1	// 1

(6.12)

The columns $A_2A_1A_0$ contain the <m-tuple> (here a 3-tuple) notation for each GF(8)-0 element. The column "hex" is the m-tuple represented as a hexadecimal digit (in this example it is also octal and decimal). Once the table is constructed, one can rewrite the table sorted on the hex value since that is sometimes useful. Since the columns α^n and hex are redundant, they are usually not shown.

Here then is how the above table and its hex-sorted version appears in Bussey, where

our n→λ $\quad$ our α→i $\quad$ our $A_2,A_1A_0 \rightarrow \alpha,\beta,\gamma$

$p(x) = x^3 + x + 1$ [as used above]

$GF[2^3]$, $i^3 \equiv i + 1$, modulo 2. $\quad i^\lambda = \alpha i^2 + \beta i + \gamma$.

FIRST TABLE.

λ	α	β	γ
1		1	0
2	1	0	0
3		1	1
4	1	1	0
5	1	1	1
6	1	0	1
7			1

SECOND TABLE.

λ	α	β	γ
7			1
1		1	0
3		1	1
2	1	0	0
6	1	0	1
4	1	1	0
5	1	1	1

(6.13)

The tables of course get larger as $q = p^m$ increases.

Example GF(2^4)

$$p(x) = x^4 + x + 1 \quad [\text{ this is } p_1(x) \text{ of (6.21) }]$$

$GF[2^4]$, $i^4 \equiv i + 1$, modulo 2. $i^\lambda = \alpha i^3 + \beta i^2 + \gamma i + \delta$.

FIRST TABLE.

λ	α	β	γ	δ
1			1	0
2		1	0	0
3	1	0	0	0
4			1	1
5		1	1	0
6	1	1	0	0
7	1	0	1	1
8		1	0	1
9	1	0	1	0
10		1	1	1
11	1	1	1	0
12	1	1	1	1
13	1	1	0	1
14	1	0	0	1
15				1

SECOND TABLE.

λ	α	β	γ	δ
15				1
1			1	0
4			1	1
2		1	0	0
8		1	0	1
5		1	1	0
10		1	1	1
3	1	0	0	0
14	1	0	0	1
9	1	0	1	0
7	1	0	1	1
6	1	1	0	0
13	1	1	0	1
11	1	1	1	0
12	1	1	1	1

(6.14)

Example GF(2^5)

$$p(x) = x^5 + x^3 + x^2 + x + 1 \quad [\text{ this is } p_1(x) \text{ of (6.22) }]$$

$GF[2^5]$, $i^5 \equiv i^3 + i^2 + i + 1$, modulo 2. $i^\lambda = \alpha i^4 + \beta i^3 + \gamma i^2 + \delta i + \epsilon$.

FIRST TABLE.

λ	α	β	γ	δ	ε
1				1	0
2			1	0	0
3		1	0	0	0
4	1	0	0	0	0
5		1	1	1	1
6	1	1	1	1	0
7	1	0	0	1	1
8		1	0	0	1
9	1	0	0	1	0
10		1	0	1	1
11	1	0	1	1	0
12				1	1
13			1	1	0
14		1	1	0	0
15	1	1	0	0	0
16	1	1	1	1	1
17	1	0	0	0	1
18		1	1	0	1
19	1	1	0	1	0
20	1	1	0	1	1
21	1	1	0	0	1
22	1	1	1	0	1
23	1	0	1	0	1
24			1	0	1
25		1	0	1	0
26	1	0	1	0	0
27			1	1	1
28		1	1	1	0
29	1	1	1	0	0
30	1	0	1	1	1
31					1

SECOND TABLE.

λ	α	β	γ	δ	ε
31					1
1				1	0
12				1	1
2			1	0	0
24			1	0	1
13			1	1	0
27			1	1	1
3		1	0	0	0
8		1	0	0	1
25		1	0	1	0
10		1	0	1	1
14		1	1	0	0
18		1	1	0	1
28		1	1	1	0
5		1	1	1	1
4	1	0	0	0	0
17	1	0	0	0	1
9	1	0	0	1	0
7	1	0	0	1	1
26	1	0	1	0	0
23	1	0	1	0	1
11	1	0	1	1	0
30	1	0	1	1	1
15	1	1	0	0	0
21	1	1	0	0	1
19	1	1	0	1	0
20	1	1	0	1	1
29	1	1	1	0	0
22	1	1	1	0	1
6	1	1	1	1	0
16	1	1	1	1	1

(6.15)

Example GF(3^2)

This is our first example with p > 2. It turns out that $p(x) = x^2 + x + 2$ is one of two possible primitive polynomials for this field (the other is $x^2 + 2x + 2$). For GF(p) the symbols can be taken as the integers mod p. And -n = (p-n). So for p = 3, we have 4 = 1, -1 = 2, -2 = 1, 3 = 0 and so on. It is then easy to construct the table :

$\alpha^2 = -\alpha - 2 = 2\alpha + 1$ // from the primitive polynomial
$\alpha^3 = \alpha^2\alpha = (2\alpha + 1)\alpha = 2\alpha^2 + \alpha = 2(2\alpha+1)+\alpha = 4\alpha + 2 + \alpha = 2\alpha+2$
$\alpha^4 = \alpha^3\alpha = (2\alpha+2)\alpha = 2\alpha^2 + 2\alpha = 2(2\alpha + 1) + 2\alpha = 6\alpha + 2 = 2$
and so on.

Here then is the fully constructed power table for GF(3^2), using primitive polynomial $x^2 + x + 2$:

α	10
$\alpha^2 = 2\alpha+1$	21
$\alpha^3 = 2\alpha+2$	22
$\alpha^4 = 2$	02
$\alpha^5 = 2\alpha$	20
$\alpha^6 = \alpha+2$	12
$\alpha^7 = \alpha + 1$	11
$\alpha^8 = 1$	01

(6.16)

The fact that all elements are distinct confirms that $x^2 + x + 2$ is primitive.

Since we have never done this before for p > 2, we shall verify that the additive inverse of each element in the list is in the list :

$-\alpha = 2\alpha = \alpha^5$
$-\alpha^2 = 2\alpha^2 = 2(2\alpha+1) = 4\alpha+2 = \alpha+2 = \alpha^6$
etc.

Note that $-1 = 2 = \alpha^4$ and of course -0 = 0. Completing this list we find

$-\alpha = \alpha^5$	$-\alpha^6 = \alpha^2$
$-\alpha^2 = \alpha^6$	$-\alpha^7 = \alpha^3$
$-\alpha^3 = \alpha^7$	$-1 = \alpha^4$
$-\alpha^4 = 1$	-0 = 0
$-\alpha^5 = \alpha$	

(6.17)

and indeed, the additive inverse of each element of GF(3^2) lies in GF(3^2).

If one takes instead the other primitive polynomial, $x^2 + 2x + 2$, then $x^2 = -2x - 2 = x+1$ and one then obtains the following alternate table from Bussey,

$$p(x) = x^2\text{-}x\text{-}1 \;= x^2 + 2x + 2$$

$GF[3^2]$, $i^2 \equiv i + 1$, modulo 3. $i^\lambda = \alpha i + \beta$.

FIRST TABLE.

λ	α	β	λ	α	β
1	1	0	5	2	0
2	1	1	6	2	2
3	2	1	7	1	2
4		2	8		1

SECOND TABLE.

λ	α	β	λ	α	β
8		1	7	1	2
4		2	5	2	0
1	1	0	3	2	1
2	1	1	6	2	2

(6.18)

(c) Using Maple to build any GF(q) Enumeration Table

The code shown here uses the "RootOf" capability of Maple. The last example uses p = 3.

Normally in Maple if one has functions $f = \alpha^6$ and $\alpha = w + z$, it knows that $f = (w+z)^6$. However, if $f = \alpha^6$ and you want to tell Maple that $\alpha^3 = \alpha + 1$ and you want Maple to use this rule to simplify α^6, you cannot naively inform Maple of your rule by making a normal assignment,

```
> alpha^3 := 1 + alpha;
Error, invalid lhs of assignment
> alpha := (1 + alpha)^(1/3):
Warning, recursive definition of name
```

Here is a sample calculation we want Maple to do,

$$\alpha^6 = \alpha^3\alpha^3 = (1+\alpha)^2 = 1 + 2\alpha + \alpha^2 \rightarrow\; 1 + \alpha^2 \qquad \text{since } 2(\text{anything}) = 0$$

and here is how Maple can do it,

```
alias(a = RootOf(z^3-z-1)):
subs(a=alpha,simplify(a^6));
```

$$2\,\alpha + \alpha^2 + 1$$

```
subs(a=alpha,simplify(a^6)) mod 2;
```

$$\alpha^2 + 1$$

In the first line a is an "alias" and we inform Maple that 'a' should be a root of $p(x) = x^3\text{-}x\text{-}1$, meaning 'a' is a solution of $a^3\text{-}a\text{-}1 = 0$, which means we are telling Maple that $a^3 = a + 1$. The second line is a kludge to get what we want, and the third line is the second line told to throw out things like 2α.

Here then is code to generate the table for **GF(2^3)** with p(x) = x^3+x+1 :

```
t:=array(1..7):
alias(a=RootOf(x^3+x+1)):
for n from 1 to 7 do
  t[n]:=sort(subs(a=alpha,simplify(a^n)) mod 2);
od:
print(t);
```

$$[\alpha, \alpha^2, \alpha+1, \alpha^2+\alpha, \alpha^2+\alpha+1, \alpha^2+1, 1]$$

n	α^n
1	α
2	α^2
3	$\alpha+1$
4	$\alpha^2+\alpha$
5	$\alpha^2+\alpha+1$
6	α^2+1
7	1

Fig 6.4

With a little extra fiddling the result can be displayed in a Maple "spreadsheet" as on the right, which can then be compared with (6.11) or (6.13) above.

Here is the same thing for **GF(2^4)** where we use p(x) = x^4+x+1 as the primitive polynomial, so in this case $\alpha^4 = \alpha + 1$:

```
> t := array(1..15):
> alias(a=RootOf(x^4+x+1)):
> for n from 1 to 15 do
    t[n]:=sort(subs(a=alpha,simplify(a^n)) mod 2);
  od:
> print(t);
```

$$[\alpha, \alpha^2, \alpha^3, \alpha+1, \alpha^2+\alpha, \alpha^3+\alpha^2, \alpha^3+\alpha+1, \alpha^2+1, \alpha^3+\alpha, \alpha^2+\alpha+1,$$
$$\alpha^3+\alpha^2+\alpha, \alpha^3+\alpha^2+\alpha+1, \alpha^3+\alpha^2+1, \alpha^3+1, 1]$$

n	α^n
1	α
2	α^2
3	α^3
4	$\alpha+1$
5	$\alpha^2+\alpha$
6	$\alpha^3+\alpha^2$
7	$\alpha^3+\alpha+1$
8	α^2+1
9	$\alpha^3+\alpha$
10	$\alpha^2+\alpha+1$
11	$\alpha^3+\alpha^2+\alpha$
12	$\alpha^3+\alpha^2+\alpha+1$
13	$\alpha^3+\alpha^2+1$
14	α^3+1
15	1

Fig 6.5

which can be compared with (6.14) above.

Next, here is the Table for **GF(2^5)** where $p(x) = x^5 + x^3 + x^2 + x + 1$:

```
> t := array(1..31):
> alias(a=RootOf(x^5+x^3+x^2+x+1)):
> for n from 1 to 31 do
    t[n]:=sort(subs(a=alpha,simplify(a^n)) mod 2);
  od:
> print(t);
```

$$[\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^3+\alpha^2+\alpha+1, \alpha^4+\alpha^3+\alpha^2+\alpha, \alpha^4+\alpha+1, \alpha^3+1, \alpha^4+\alpha, \alpha^3+\alpha+1,$$
$$\alpha^4+\alpha^2+\alpha, \alpha+1, \alpha^2+\alpha, \alpha^3+\alpha^2, \alpha^4+\alpha^3, \alpha^4+\alpha^3+\alpha^2+\alpha+1, \alpha^4+1, \alpha^3+\alpha^2+1,$$
$$\alpha^4+\alpha^3+\alpha, \alpha^4+\alpha^3+\alpha+1, \alpha^4+\alpha^3+1, \alpha^4+\alpha^3+\alpha^2+1, \alpha^4+\alpha^2+1, \alpha^2+1, \alpha^3+\alpha,$$
$$\alpha^4+\alpha^2, \alpha^2+\alpha+1, \alpha^3+\alpha^2+\alpha, \alpha^4+\alpha^3+\alpha^2, \alpha^4+\alpha^2+\alpha+1, 1]$$

n	α^n
1	α
2	α^2
3	α^3
4	α^4
5	$\alpha^3+\alpha^2+\alpha+1$
6	$\alpha^4+\alpha^3+\alpha^2+\alpha$
7	$\alpha^4+\alpha+1$
8	α^3+1
9	$\alpha^4+\alpha$
10	$\alpha^3+\alpha+1$
11	$\alpha^4+\alpha^2+\alpha$
12	$\alpha+1$
13	$\alpha^2+\alpha$
14	$\alpha^3+\alpha^2$
15	$\alpha^4+\alpha^3$
16	$\alpha^4+\alpha^3+\alpha^2+\alpha+1$
17	α^4+1
18	$\alpha^3+\alpha^2+1$
19	$\alpha^4+\alpha^3+\alpha$
20	$\alpha^4+\alpha^3+\alpha+1$
21	$\alpha^4+\alpha^3+1$
22	$\alpha^4+\alpha^3+\alpha^2+1$
23	$\alpha^4+\alpha^2+1$
24	α^2+1
25	$\alpha^3+\alpha$
26	$\alpha^4+\alpha^2$
27	$\alpha^2+\alpha+1$
28	$\alpha^3+\alpha^2+\alpha$
29	$\alpha^4+\alpha^3+\alpha^2$
30	$\alpha^4+\alpha^2+\alpha+1$
31	1

Fig 6.6

This data will be needed in the next section and may be compared with (6.15).

Finally, here is the Table for GF(3^2) where we replace mod 2 by mod 3 in the code:

```
t := array(1..8):
alias(a=RootOf(x^2+x+2)):
for n from 1 to 8 do
  t[n]:=sort(subs(a=alpha,simplify(a^n)) mod 3);
od:

print(t);
```

$$[\alpha, 2\alpha+1, 2\alpha+2, 2, 2\alpha, \alpha+2, \alpha+1, 1]$$

n	α^n
1	α
2	$2\alpha+1$
3	$2\alpha+2$
4	2
5	2α
6	$\alpha+2$
7	$\alpha+1$
8	1

Fig 6.7

Since we used $p(x) = x^2 + x + 2$, this matches our (6.16) rather than Bussey's (6.18).

(d) Using the GF(q) Table to multiply out polynomials factored in GF(q)

In Big Theorem 3 (5.17) we showed how to obtain all minimum polynomials of some GF(q) in factored form, where α is some element of GF(q),

$$m(x) = (x - \alpha) \bullet (x - \alpha^p) \bullet (x - \alpha^{p^2}) \bullet (x - \alpha^{p^3}) \ldots \ldots (x - \alpha^{p^{k-1}}) \qquad \alpha^{p^k} = \alpha \qquad k \le m \ . \qquad (5.17)$$

In the examples of Chapter 5 (c) we then explicitly listed all minimal polynomials for $GF(2^m)$ for m = 3,4,5. Here we shall show how the GF(q) table constructed above can be used to expand the m(x) into a form which explicitly has coefficients in GF(2), that is, polynomials like $x^2 + x + 1$. We shall first do an example by hand, then show how to automate the process using Maple.

1. Expansion of a factored minimum polynomial: an example

For $GF(2^3)$ one of the minimum polynomials (in fact a primitive one) was found to be

$$p_1(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4) \ . \qquad (5.24)$$

The first step is to expand in the obvious manner,

$$p_1(x) = x^3 + [\alpha + \alpha^2 + \alpha^4]x^2 + [\alpha^3 + \alpha^5 + \alpha^6]x + [1]\ 1 \ .$$

The important information in the above $GF(2^3)$ table is this

$$\alpha^3 = 1 + \alpha$$
$$\alpha^4 = \alpha + \alpha^2$$
$$\alpha^5 = 1 + \alpha + \alpha^2$$
$$\alpha^6 = 1 + \alpha^2$$

which we then use to evaluate each of the non-obvious coefficients

$$\text{coeff of } x^2 = \alpha + \alpha^2 + \alpha^4 = \alpha + \alpha^2 + (\alpha + \alpha^2) = 2\alpha + 2\alpha^2 = 0 + 0 = 0 \qquad \text{// } 2y=0 \text{ for any } y$$

$$\text{coeff of } x = \alpha^3 + \alpha^5 + \alpha^6 = (1 + \alpha) + (1 + \alpha + \alpha^2) + (1 + \alpha^2) = 3 + 2\alpha + 2\alpha^2 = 1$$

Thus, we find

$$p_1(x) = x^3 + x + 1 \qquad (6.19)$$

This just happens to be the very primitive polynomial with which we built the table. But this method works with any factored polynomial, and we will do them all below.

2. Maple expansion of factored minimum polynomials for GF(2^3), GF(2^4), GF(2^5)

Using the Maple "alias" method described at the start of Chap 6 (c), we can now expand *all* the factored minimum polynomials obtained back in Chap 5 (c), primitive p(x) and non-primitive m(x). In each case we copy the polynomials from Chap 5 (c), do the calculation, and then write back the results to the right of the copied equations. If one makes an error entering an exponent, the result blows out into some long polynomial of α, so just getting simple results with coefficients in GF(2) is a good test on the accuracy of one's Maple code entry! One should keep in mind that which factored polynomials go with which expanded polynomials depends on the primitive polynomial p(x) one uses to build the Table for GF(q), which is to say, depends on one's choice of the primitive element α.

In **Appendix H** we prove the following claims regarding order-reversed polynomials:

Fact 1: If h(x) is irreducible in R, then so is its order-reversed partner H(x). *(H.2)*

Fact 2: If h(x) is a minimum polynomial in R, then so its order-reversed partner H(x). *(H.6)*

Fact 3: If h(x) is a primitive polynomial in R, then so is its order-reversed partner H(x). *(H.15)*

Fact 4 Corollary: Except for GF(2), GF(2^2) and GF(3), no primitive polynomial is symmetric and therefore primitive polynomials always come in pairs as per Fact 3. *(H.18)*

One will find all these rules respected in the following polynomial lists. In particular, primitive polynomial pair m-tuples are shown in matching color.

GF(2^3)

$p_1(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4)$ $= x^3 + x + 1$ 1011

$p_3(x) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^5)$ $= x^3 + x^2 + 1$ 1101 (6.20)

```
alias(a=RootOf(x^3+x+1)):          // x^3 + x + 1 used as the defining prim poly

p1 := (x-a)*(x-a^2)*(x-a^4);
                        p1 := (x - a) (x - a^2) (x - a^4)
subs(a=alpha,simplify(p1)) mod 2;
                                  x^3 + x + 1
p3 := (x-a^3)*(x-a^6)*(x-a^5);
                        p3 := (x - a^3) (x - a^6) (x - a^5)
subs(a=alpha,simplify(p3)) mod 2;
                                  x^3 + 1 + x^2
```

GF(2^4)

$p_1(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8) = x^4 + x + 1$ 10011

$p_7(x) = (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{13})(x - \alpha^{11}) = x^4 + x^3 + 1$ 11001

$m_3(x) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^9) = x^4 + x^3 + x^2 + x + 1$ 11111

$m_5(x) = (x - \alpha^5)(x - \alpha^{10}) = x^2 + x + 1$ 111 (6.21)

```
alias(a=RootOf(x^4+x+1)):
```
// $x^4 + x + 1$ used as the defining prim poly

```
p1 := (x-a)*(x-a^2)*(x-a^4)*(x-a^8);
subs(a=alpha,simplify(p1)) mod 2;
p7 := (x-a^7)*(x-a^14)*(x-a^13)*(x-a^11);
subs(a=alpha,simplify(p7)) mod 2;
m3 := (x-a^3)*(x-a^6)*(x-a^12)*(x-a^9);
subs(a=alpha,simplify(m3)) mod 2;
m5 := (x-a^5)*(x-a^10);
subs(a=alpha,simplify(m5)) mod 2;
```

$$p1 := (x-a)(x-a^2)(x-a^4)(x-a^8)$$

$$x^4 + x + 1$$

$$p7 := (x-a^7)(x-a^{14})(x-a^{13})(x-a^{11})$$

$$x^4 + 1 + x^3$$

$$m3 := (x-a^3)(x-a^6)(x-a^{12})(x-a^9)$$

$$x^4 + x^3 + x^2 + 1 + x$$

$$m5 := (x-a^5)(x-a^{10})$$

$$x^2 + 1 + x$$

In this example, the non-primitive minimum polynomials are symmetric and so don't appear as "pairs".

GF(2^5) : [Note that these agree with Fig 5.4.]

$p_1(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)(x-\alpha^{16}) = x^5 + x^3 + x^2 + x + 1$ 101111
$p_3(x) = (x-\alpha^3)(x-\alpha^6)(x-\alpha^{12})(x-\alpha^{24})(x-\alpha^{17}) = x^5 + x^4 + x^3 + x + 1$ 111011
$p_5(x) = (x-\alpha^5)(x-\alpha^{10})(x-\alpha^{20})(x-\alpha^9)(x-\alpha^{18}) = x^5 + x^2 + 1$ 100101
$p_7(x) = (x-\alpha^7)(x-\alpha^{14})(x-\alpha^{28})(x-\alpha^{25})(x-\alpha^{19}) = x^5 + x^4 + x^2 + x + 1$ 110111
$p_{11}(x) = (x-\alpha^{11})(x-\alpha^{22})(x-\alpha^{13})(x-\alpha^{26})(x-\alpha^{21}) = x^5 + x^3 + 1$ 101001
$p_{15}(x) = (x-\alpha^{15})(x-\alpha^{30})(x-\alpha^{29})(x-\alpha^{27})(x-\alpha^{23}) = x^5 + x^4 + x^3 + x^2 + 1$ 111101

(6.22)

```
alias(a=RootOf(x^5+x^3+x^2+x+1)): // x^5 + x^3 + x^2 + x + 1 used as the defining prim poly
p1 := (x-a)*(x-a^2)*(x-a^4)*(x-a^8)*(x-a^16);
subs(a=alpha,simplify(p1)) mod 2;
p3 := (x-a^3)*(x-a^6)*(x-a^12)*(x-a^24)*(x-a^17);
subs(a=alpha,simplify(p3)) mod 2;
p5 := (x-a^5)*(x-a^10)*(x-a^20)*(x-a^9)*(x-a^18);
subs(a=alpha,simplify(p5)) mod 2;
p7 := (x-a^7)*(x-a^14)*(x-a^28)*(x-a^25)*(x-a^19);
subs(a=alpha,simplify(p7)) mod 2;
p11 := (x-a^11)*(x-a^22)*(x-a^13)*(x-a^26)*(x-a^21);
subs(a=alpha,simplify(p11)) mod 2;
p15 := (x-a^15)*(x-a^30)*(x-a^29)*(x-a^27)*(x-a^23);
subs(a=alpha,simplify(p15)) mod 2;
```

$$p1 := (x-a)(x-a^2)(x-a^4)(x-a^8)(x-a^{16})$$
$$x^5 + x^3 + x^2 + x + 1$$
$$p3 := (x-a^3)(x-a^6)(x-a^{12})(x-a^{24})(x-a^{17})$$
$$x^4 + x + 1 + x^5 + x^3$$
$$p5 := (x-a^5)(x-a^{10})(x-a^{20})(x-a^9)(x-a^{18})$$
$$1 + x^5 + x^2$$
$$p7 := (x-a^7)(x-a^{14})(x-a^{28})(x-a^{25})(x-a^{19})$$
$$x^4 + x + 1 + x^5 + x^2$$
$$p11 := (x-a^{11})(x-a^{22})(x-a^{13})(x-a^{26})(x-a^{21})$$
$$1 + x^5 + x^3$$
$$p15 := (x-a^{15})(x-a^{30})(x-a^{29})(x-a^{27})(x-a^{23})$$
$$x^4 + 1 + x^5 + x^3 + x^2$$

3. Factoring polynomials in Maple

Having done all the above, we now show how Maple can work in the opposite direction. For a given polynomial that is factorizable in GF(q), Maple can factor it. Here is an example for GF(2^5). From (6.22) we know that

$p_1(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)(x-\alpha^{16}) = x^5 + x^3 + x^2 + x + 1$. *(6.22)*

So we enter $x^5 + x^3 + x^2 + x + 1$ and ask Maple if it can factor this into a product of $(x-r_i)$ factors where the roots r_i all lie in GF(2^5) . We continue to use alias(a=RootOf(x^5+x^3+x^2+x+1) from earlier.

```
Factor(x^5+x^3+x^2+x+1,a) mod 2;
```

$$(x+a^3+1)(x+a^4+a^3+a^2+a+1)(x+a^4)(x+a)(x+a^2)$$

If we just want the roots,

```
Roots(x^5+x^3+x^2+x+1,a) mod 2;
```

$$[[a^4+a^3+a^2+a+1, 1], [a^4, 1], [a, 1], [a^3+1, 1], [a^2, 1]]$$

α^{16} α^4 α α^8 α^2

In each pair, the second index gives the root multiplicity and here each root occurs only once. Looking in the GF(2^5) table in Fig 6.6 we find that this root list contains the values added above in black, and these roots agree with those of the factored $p_1(x)$ shown above. If Maple is asked to factor something that is not factorable in GF(q) it will go as far as it can,

```
Factor(x^5+x^4 +1,a) mod 2;
```

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

but there are no roots in GF(2^5) so the Roots function returns a null set

```
Roots(x^5+x^4 +1,a) mod 2;
```

$$[\,]$$

As shown above, $x^5 + x^4 + 1$ is reducible into the product of two factors $(x^3+x+1)(x^2+x+1)$. Each of these factors is irreducible but is not a minimum polynomial since each cannot be factored into a product of GF(2^5) elements (otherwise Maple would have done it). Thus, these two polynomials fit into the white region on the right side of Fig 5.5. To be specific, for x^3+x+1 we get

```
Factor(x^3+x+1,a) mod 2;
```

$$x^3+1+x$$

```
Roots(x^3+x+1,a) mod 2;
```

$$[\,]$$

Notice the alias 'a' second argument of the Factor and Roots commands above. If one simply wants to factor a polynomial within GF(2) and not GF(q), this argument should be omitted. For example, back in (5.31) we eliminated various polynomials from a list because they were not irreducible. There we had this partially factored result done by hand,

$x^4 + x^3 + x + 1 \qquad = (x^3+1)(x+1) =$ reducible

Maple can check to see if $x^4 + x^3 + x + 1$ is factorable in GF(2) as follows,

```
Factor(x^4 + x^3 + x + 1) mod 2;
```

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

Comparing these two results, it must be that $(x^3+1) = (x+1)(x^2+x+1)$. We check that below.

4. Multiplying GF(2) polynomials in Maple

We have already seen in section 2 above how to multiply polynomials in GF(q) using the alias 'a' argument. If one wants to do the same within the base field GF(2), this argument is omitted. For example, one can verify the claim made just above. First we do it by hand,

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

and then we ask Maple to do it for us

```
expand((x+1)*(x^2+x+1)) mod 2;
```

$$x^3 + 1$$

5. Finding GF(2) quotients and remainders in Maple

Maple also knows how to compute quotients and remainders in the GF(2) world. For example, we know that

$$p_1(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)(x-\alpha^{16}) = x^5 + x^3 + x^2 + x + 1 \quad . \qquad (6.22)$$

is a minimum polynomial and therefore must divide evenly into $x^{31} - 1$:

```
Quo(x^31-1,x^5+x^3+x^2+x+1,x) mod 2;
```

$$x^{26} + x^{24} + x^{23} + x^{21} + x^{19} + x^{15} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^4 + x + 1$$

```
Rem(x^31-1,x^5+x^3+x^2+x+1,x) mod 2;
```

$$0$$

On the other hand, our little non-factorable $x^5 + x^4 + 1$ does not divide into $x^{31} - 1$ evenly,

```
Quo(x^31-1,x^5+x^4 +1,x) mod 2;
```

$$x^{26} + x^{25} + x^{24} + x^{23} + x^{22} + x^{20} + x^{18} + x^{15} + x^{14} + x^{10} + x^5 + x^4 + x^3 + x^2 + x$$

```
Rem(x^31-1,x^5+x^4 +1,x) mod 2;
```

$$x^4 + x^3 + x^2 + x + 1$$

6. Finding all Irreducible and Minimum Polynomials of GF(2^m)

The simple code below works for GF(2^m) but could easily be generalized for GF(p^m). The example given is GF(2^4). We select a GF(2^4) primitive polyomial p(x) = x^4+x+1 as shown in (6.21) and set "a" using the alias method discussed above. Then we exhaust all possible polynomials of degree 4 and less which have constant 1 and have coefficients in GF(2) (0 or 1) and we attempt to factor each one:

```
alias(a = RootOf(z^4+z+1)):          # appears in the Factor call below
for n from 1 to 2^5-1 by 2 do        # only generate polys of R having a +1
  c := convert(n,base,2):            # for example, n=6 makes c = [0,1,1]
  L := nops(c):                      # L = number of bits in c, eg, 3
  p := sum(c[i]*x^(i-1),i=1..L):     # generate poly p(x) with c as coeffs
  f := Factor(p,a) mod 2:            # attempt to factor p(x) in GF(p,m)
  print(p,"       ",f);              # print p(x) and then its factored form
od:
```

For each non-trivial polynomial there are four possibilities:

(A) factors into unique (x-a_i) factors where all $a_i \neq 1$	irreducible and min poly (maybe prim)
(B) has at least one (x+1) factor	reducible hence not min poly
(C) has a single but non-linear-in-x factor	irreducible and not min poly
(D) has multiple factors with at least one non-linear-in-x	reducible hence not min poly
(E) case A but at least one factor appears more than once	reducible hence not min poly

The irreducible polynomials are all A + C. In terms of Fig 5.5 which we repeat here, the A items go in the gray area (some of them will be primitive, some will not), while the C items go in the white area.

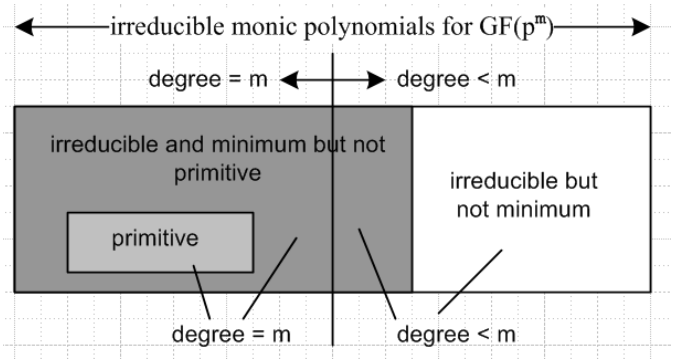


Fig 5.5

Here is the output of the above code where each case is labeled A,B,C,D. The four A items are in (6.21). See Fig 6.5 to find that $\alpha^2+\alpha = \alpha^5$ and $\alpha^2+\alpha+1 = \alpha^{10}$, so the first "A" item is $(x-\alpha^5)(x-\alpha^{10})$ which we know from (6.21) is a non-primitive minimum polynomial.

$1,"\quad ",1$

$1+x,"\quad ",1+x$ **B**

$1+x^2,"\quad ",(1+x)^2$ **B**

$1+x+x^2,"\quad ",(x+a^2+a)(x+a^2+a+1)$ **A**

$1+x^3,"\quad ",(x+1)(x+a^2+a+1)(x+a^2+a)$ **B**

$1+x+x^3,"\quad ",1+x+x^3$ **C**

$1+x^2+x^3,"\quad ",1+x^2+x^3$ **C**

$1+x+x^2+x^3,"\quad ",(x+1)^3$

$1+x^4,"\quad ",(x+1)^4$ **B**

$1+x+x^4,"\quad ",(x+a+1)(x+a)(x+a^2)(x+a^2+1)$ **A**

$1+x^2+x^4,"\quad ",(x+a^2+a+1)^2\ (x+a^2+a)^2$ **E**

$1+x+x^2+x^4,"\quad ",(x+1)(1+x^2+x^3)$ **D**

$1+x^3+x^4,"\quad ",(x+a^3+a^2+1)(x+a^3+a^2+a)(x+a^3+a+1)(x+a^3+1)$ **A**

$1+x+x^3+x^4,"\quad ",(x+1)^2(x+a^2+a+1)(x+a^2+a)$ **B**

$1+x^2+x^3+x^4,"\quad ",(x+1)(x^3+1+x)$ **D**

$1+x+x^2+x^3+x^4,"\quad ",(x+a^3)(x+a^3+a)(x+a^3+a^2)(x+a^3+a^2+a+1)$ **A**

One could of course automate the process of identifying all polynomials of any particular case. For example, Appendix C of Weldon and Peterson enumerates all those of type A and C (that is, all irreducible polynomials) for GF(2^m) with m = 1 to 16, as described in the following section.

7. Connection with Peterson and Weldon Appendix C

Here are the results from (6.21) above for GF(2^4), where we have added an *octal* notation on the right.

$$
\begin{array}{lll}
p_1(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8) & = x^4+x+1 & 10011 = 23_8 \\
p_7(x) = (x-\alpha^7)(x-\alpha^{14})(x-\alpha^{13})(x-\alpha^{11}) & = x^4+x^3+1 & 11001 = 31_8 \\
m_3(x) = (x-\alpha^3)(x-\alpha^6)(x-\alpha^{12})(x-\alpha^9) & = x^4+x^3+x^2+x+1 & 11111 = 37_8 \\
m_5(x) = (x-\alpha^5)(x-\alpha^{10}) & = x^2+x+1 & 111 = 7_8 \qquad (6.21)
\end{array}
$$

Knowing how to form a conjugate set given any element of the set, we might decide to make the following list of minimum polynomials for GF(2^4), where we choose the lowest power of α in each minimum polynomial as a marker:

GF(2^4): 1 23_8 prim 3 37_8 5 7_8 7 31_8 prim

Knowing that primitive polynomials always come in order-reversed pairs according to Fact 3 (H.15), we might save space in the table by not displaying the last item since we know how to order-reverse the first item to obtain the last item. Then we have just

GF(2^4): 1 23_8 prim 3 37_8 5 7_8

This is the basic notation used in the Peterson and Weldon Appendix C,

DEGREE 4 1 23F 3 37D 5 07

Primitive polynomials of degree m get suffix letters E,F,G,H while non-minimum polynomials of degree m get suffix letters A,B,C,D and polynmials of degree less than 4 get no suffix. The tables show full results for GF(2^m) with m = 1 through 16. Reduced information is then given form m = 17 through 34. The number of entrees of course can get very large. For example

number or primitive polynomials for GF(2^{16}) = $\varphi(2^{16}-1)/16 = 2048$

of which 1024 appear in the table, along with many non-primitive polynomials.

(e) Construction of the + and • tables for GF(2^2)

To illustrate the method outlined above, we take this very simple case GF(2^2). We need the GF(2^2) enumeration table, which we now obtain by mimicking the GF(2^3) example above:

" From the primitive polynomial list in Fig 5.3 we find $p(x) = 1 + x + x^2$. We proved this is a primitive polynomial for GF(2^2) in (5.29). In terms of (6.8) we have $c_2 = c_1 = c_0 = 1$. Then (6.9) says that $\alpha^2 = (2-c_1)\alpha + (2-c_0) = \alpha + 1$. So the reduction rule (6.9) states that $\alpha^2 = \alpha + 1$. This is of course totally obvious just setting $p(\alpha) = 0$ for the above p(x), remembering + and - are the same, but we wanted to see how (6.9) gets the right answer. So we can build the entire enumeration table in 1 minute.

0	00	0	
$\alpha^0 = 1$	01	1	$\alpha^3 = \alpha^2\alpha = (\alpha+1)\alpha = \alpha^2+\alpha = \alpha+1+\alpha = 2\alpha+1 = 1$
α	10	2	
$\alpha^2 = \alpha + 1$	11	3	(6.23)

Notice that all 2-tuples are "hit", as claimed above. We have added some octal/hex numbers to make this fact more obvious. We know that $\alpha^3 = 1$, since the non-zero elements of GF(8) form a cyclic group under • for a primitive element like α (which is a root of primitive polynomial p(x)). This fact is verified as shown above on the right. This fact $\alpha^3 = 1$ was not needed to build the above enumeration table, but it is needed to construct the • table. "

The addition table was already worked out in (4.10) in the m-tuple basis so we just quote that result here,

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

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

(6.24)

Recall that the table on the left is trivially computed by doing GF(2) addition (XOR) independently on each position of the 2-tuples being added as shown in (4.8). This table can be written in the power basis using (6.23), where on the right we use $\alpha + 1$ in place of α^2,

+	0	1	α	α^2
0	1	1	α	α^2
1	1	0	α^2	α
α	α	α^2	0	1
α^2	α^2	α	1	0

+	0	1	α	$\alpha+1$
0	1	1	α	$\alpha+1$
1	1	0	$\alpha+1$	α
α	α	$\alpha+1$	0	1
$\alpha+1$	$\alpha+1$	α	1	0

(6.25)

As noted above, the multiplication table is trivial to construct in the power basis using $\alpha^3 = 1$ as shown on the left,

•	0	1	α	α^2
0	0	0	0	0
1	0	1	α	α^2
α	0	α	α^2	1
α^2	0	α^2	1	α

•	0	1	α	$\alpha+1$
0	0	0	0	0
1	0	1	α	$\alpha+1$
α	0	α	$\alpha+1$	1
$\alpha+1$	0	$\alpha+1$	1	α

(6.26)

On the right we have just replaced α^2 by $\alpha + 1$. Using the enumeration table above, we can write this multiplication table in these alternate m-tuple basis,

•	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

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

(6.27)

Chapter 7: Linear Block Codes

There are many good texts on the theory of error-correcting codes; two are mentioned in the References. It has been our experience that most of the difficulty encountered in reading these texts is not so much with the coding theory itself, but with the underlying theory of Galois Fields. Since we have just finished a long exposition of Galois Field theory, it seems appropriate to make at least a preliminary connection between block codes and Galois Fields. This connection will become much tighter in Chapter 8 on Cyclic Codes.

(a) The Basics

For our purposes, a **symbol**, or **code symbol**, is an element of some Galois field GF($q=p^m$). For example, if p = 2 and m = 1, a symbol is an element of GF(2). The only elements of GF(2) are 0 and 1, and a symbol that takes these two values is called a **bit**. If p = 2 and m = 8, a symbol is an element of GF(2^8), an 8-tuple of bits known as a **byte,** which symbol can take 256 values. We shall try to keep the discussion as general as possible by using the term symbol in place of bit or byte or some other special case.

Vector Space comment

Recall from (1.14b) that the elements of GF(q) form a vector space over the field GF(p). Thus, our coding symbols are elements of this vector space, which let us call $\mathcal{V}$. One can then think of an n-tuple of coding symbols like $\mathbf{d} = (d_1, d_2, d_3)$ as being an element of a vector space $\mathcal{V}^3 \equiv \mathcal{V} \oplus \mathcal{V} \oplus \mathcal{V}$ which means $\mathcal{V}^3$ is a "direct sum" of three copies of the vector space $\mathcal{V}$. When we refer below to the vector spaces like D^k, C^k and V^k, we are referring to direct sum spaces of the type $\mathcal{V}^k$.

A direct sum of vector spaces is always itself a vector space because all the axioms of a vector space are fulfilled, as the reader can verify. For example, if $\mathbf{d} = (d_1, d_2, d_3)$ and $\mathbf{e} = (e_1, e_2, e_3)$, then the addition of two vectors in the $\mathcal{V}^3$ space is defined in the very obvious manner:

$$\mathbf{d} + \mathbf{e} = (d_1, d_2, d_3) + (e_1, e_2, e_3) = (d_1+e_1, d_2+e_2, d_3+e_3), \tag{7.1}$$

and of course a sum of code *symbols* (elements of GF(q)) like d_2+e_2 is provided by the + table for GF(q).

A vector space in general allows for vectors to be added as above. It says nothing about the possible multiplication of two vectors. We shall in effect be using the following definition of the product of two vectors in $\mathcal{V}^3$,

$$\mathbf{d} \bullet \mathbf{e} = (d_1, d_2, d_3) \bullet (e_1, e_2, e_3) = (d_1 \bullet e_1, d_2 \bullet e_2, d_3 \bullet e_3)\ , \tag{7.2}$$

where terms like $d_2 \bullet e_2$ are provided by the $\bullet$ table for GF(q).

Notice that the three spaces $\mathcal{V}$ which comprise $\mathcal{V}^3$ do not interact with each other in any way. A familiar example is a point $\mathbf{r} = (x,y,z)$ in 3D Cartesian space $R^3 = R \oplus R \oplus R$ where R is the real numbers. In linear algebra, a direct sum matrix like $\mathcal{V}^{(3)}$ can be thought of as being a large matrix in which the matrices being direct-summed appear as square blocks on the diagonal, surrounded by a sea of zeros. In this situation, a corresponding large vector has a section associated with each smaller matrix, and each section is affected only by its matrix block on the large matrix diagonal.

The direct sum concept is different from the notion of a direct product. In our example above, the direct product space $\mathcal{V} \otimes \mathcal{V}$ would have elements of the form $[\mathbf{d} \text{ x } \mathbf{e}]_{ij} = d_i e_j$ and $\mathcal{V} \otimes \mathcal{V} \otimes \mathcal{V}$ elements like $[\mathbf{d} \text{ x } \mathbf{e} \text{ x } \mathbf{f}]_{ijk} = d_i e_j f_k$. This is not what we are dealing with in our current context.

The basic plan of a **block code** is to take **data words** of k symbols each, add n-k **parity check symbols** which are dependent on (computed from) the k data symbols, and end up with **code words** of n symbols each. Thus, data is encoded in finite **blocks** consisting of n symbols. Such a code is denoted as **(n,k)**. In some sense the efficiency of a code is indicated by the ratio of data bits k to total bits n, known as the **code rate,**

k/n = the code rate of an (n,k) code .

One might be willing to tolerate a lower code rate if the code is good at detecting and/or correcting a certain desired number of errors in the received code words.

A subset of block codes are the **linear block codes**, where the code words are generated by the application of a linear operator onto the data words. Since we are limiting our interest to situations where k and n are finite numbers, such a linear operator is simply a **matrix**. If we think of data words and code words as **row vectors** (as opposed to column vectors), then a linear block code is defined by a matrix G (known as the **generator matrix**) acting to the left on data words **d** to generate code words **c**,

$$\mathbf{c} = \mathbf{d}G \, . \tag{7.3}$$

The data word **d** is a k-component row vector, the code word **c** is an n-component row vector, and therefore G is a matrix with n columns and k rows.

It is useful to go immediately to a simple example. For k = 2 and n = 3, so we write $\mathbf{d} = (d_1 \; d_2)$ and $\mathbf{c} = (c_1 \; c_2 \; c_3)$ and G as shown [commas in our row vectors are optional]

$$(c_1 \; c_2 \; c_3) = (d_1 \; d_2) \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} . \tag{7.4}$$

The vector $\mathbf{d} = (d_1 \; d_2)$ lies in a 2 dimensional vector space (called D^2, **the data space**) whose basis vectors we could take to be (1 0) and (0 1). Here, $D^2 = \mathcal{V} \oplus \mathcal{V}$.

When these basis vector data words are mapped into the code space we get

$$(a \; b \; c) = (1 \; 0) \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \qquad\qquad (d \; e \; f) = (0 \; 1) \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} . \tag{7.5}$$

Since (1 0) and (0 1) are legal data words, the triplets (a b c) and (d e f) are legal code words. These code words exist in a 3 dimensional vector space (V^3) spanned by (1 0 0), (0 1 0) and (0 0 1). We want the two code word vectors (a b c) and (d e f) to be linearly independent so that these vectors span a 2 dimensional subspace (called C^2, or **the code space**) of the 3 dimensional space V^3. For that reason, we want the k = 2 rows of the matrix G to be linearly independent. In matrix language, this means that the matrix G has **rank** k. Since the number of rows k is ≤ the number of columns n, k is the maximum rank that matrix G can possibly have.

If one makes arbitrary linear combinations of (a b c) and (d e f), one exhausts all possible code vectors in V^3. There are of course q^2 such code words in our example (q from GF(q)) because there are q^2 data words we can start with.

In general there will be q^k such code words in C^k if the symbols all lie in GF(q) and data words have k symbols. Of course q^k is also the total number of possible data words in the data space D^k. There are three vector spaces of interest. Think of a word or vector as a "point" in a vector space:

data space = D^k (dim = k) which has q^k points, all of which are legal data words

code space = C^k (dim = k) which has q^k legal code word points, spanned by the k rows of G

embedding space = V^n (dim = n) which has q^n points, many of which are not legal code word points

code space C^k is a k dimensional subspace of V^n which is an n dimensional vector space (7.6)

The code space C^k is the same as the **row space** of matrix G (the vector space spanned by the rows of G) and some authors use the term row space instead of code space, but we shall refer to C^k always as the code space, which is a subspace of V^n .

Since the code points in V^n all live in the C^k subspace of V^n which has k spanning vectors, we can think of *the rest* of V^n as being spanned by some set of n-k basis vectors all of which are orthogonal to the k basis vectors which are the rows of G, like (a b c) and (d e f) in the example. We might call this $(C^k)^\perp$ where symbol $^\perp$ means this space is orthogonal or "perpendicular" to C^k. So we add to our list,

perp space = $(C^k)^\perp$ (dim = n-k) which has $q^n - q^k$ points, all of which are NOT legal code points (7.7)

The implication of course is that

$$V^k = C^k + (C^k)^\perp \tag{7.8}$$

where in this one equation + means the union of the two sets of points C^k and $(C^k)^\perp$. Since the legal code words are all points in the code space C^k we can say

linear block code (n,k) = the code space = C^k = the k-dimensional row space of matrix G . (7.9)

The components of all vectors referred to above are symbols = elements of GF(q). The elements of the G matrix are symbols, not real numbers. Even the components of vectors in $(C^k)^\perp$ are symbols. It happens that they are combinations of symbols which don't form legal code words. We sometimes like to add the word "legal" when talking about code words, but it is redundant. A code word is a legal code word. The implication is that elements of $(C^k)^\perp$ are illegal code words, meaning they are *not* code words. As we shall see, noise in data transmission can convert a legal code word into an illegal one (an error).

(b) Notational Remarks

(1) The transpose $^{\mathbf{T}}$ symbol (some people use other symbols) swaps the columns and rows of a matrix of any dimension n x m. Here are some examples (≡ means "is defined as")

$$\mathbf{d} \equiv (d_1\ d_2) \quad \Rightarrow \quad \mathbf{d}^{\mathbf{T}} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \tag{7.10}$$

$$G \equiv \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \quad \Rightarrow \quad G^{\mathbf{T}} = \begin{pmatrix} a & d \\ b & e \\ c & f \end{pmatrix} \tag{7.11}$$

$$\mathbf{d} \equiv \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \quad \Rightarrow \quad \mathbf{d}^{\mathbf{T}} = (d_1\ d_2)\ . \tag{7.12}$$

Traditionally **linear algebra texts** use column vectors like the **d** on the last line above, but we are perversely using row vectors as on the first line. Coding theory texts often do this as a space-saving measure, since one wastes less vertical line-inches writing out a row vector than a column vector. So far we have indicated row vectors like (a b c) without commas, but (a,b,c) means the same thing. Notice that there are no commas in a column vector or a matrix.

Therefore, we could write (7.3) **c** = **d**G in the following completely equivalent transposed form

$$\mathbf{c}^{\mathbf{T}} = G^{\mathbf{T}}\mathbf{d}^{\mathbf{T}} \tag{7.13}$$

where one can easily show that $(ABC...)^{\mathbf{T}} = ...C^{\mathbf{T}}B^{\mathbf{T}}A^{\mathbf{T}}$. Of course if we had decided to start off using column vectors instead of row vectors, we would have redefined everything so this equation would have said **c** = G**d**.

(2) Now what happens when we multiply a row vector and a column vector together? There are two ways to do this, known as an **inner and outer product**,

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix}(e_1\ e_2) = \begin{pmatrix} d_1e_1 & d_1e_2 \\ d_2e_1 & d_2e_2 \end{pmatrix} \qquad \text{// outer product , result is a 2x2 matrix}$$

$$(e_1\ e_2)\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = e_1d_1 + e_2d_2 \qquad \text{// inner product, result is a 1x1 matrix} \tag{7.14}$$

The outer product is an example of the direct product concept mentioned above.

In our perverse row vector notation of (7.10) these lines would be written

$$\mathbf{d}^{\mathbf{T}}\mathbf{e} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}(e_1\ e_2) = \begin{pmatrix} d_1e_1 & d_1e_2 \\ d_2e_1 & d_2e_2 \end{pmatrix} \qquad \text{// outer product , result is a 2x2 matrix}$$

$$\mathbf{e}\mathbf{d}^{\mathbf{T}} = (e_1\ e_2)\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = e_1d_1 + e_2d_2 \qquad \text{// inner product, result is a 1x1 matrix}$$

$$\equiv \mathbf{e} \bullet \mathbf{d} = \text{"dot product"} = (e,d) = \text{inner product} = \text{scalar product}\ . \tag{7.15}$$

but in normal linear algebra notation using vectors as in (7.12) one would have

$$\mathbf{d}\mathbf{e}^{\mathbf{T}} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}(e_1\ e_2) = \begin{pmatrix} d_1e_1 & d_1e_2 \\ d_2e_1 & d_2e_2 \end{pmatrix}$$

$$\mathbf{e}^{\mathbf{T}}\mathbf{d} = (e_1\ e_2)\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = e_1d_1 + e_2d_2$$

$$\equiv \mathbf{e} \bullet \mathbf{d} = \text{"dot product"} = (e,d) = \text{inner product} = \text{scalar product} . \qquad (7.16)$$

In either notation the inner product is thought of as the "dot product" of two vectors as shown, normally written with some dot symbol and bolded vectors. For a very brief description of inner products (as well as norms, distance, metrics, and Hilbert Spaces) see Section 4 of the author's tensor analysis reference.

(3) Looking at (7.4) above, we would write, for example,

$$c_2 = d_1b + d_2e$$

and in (7.16) we wrote $\mathbf{e}^{\mathbf{T}}\mathbf{d} = e_1d_1 + e_2d_2$. One must **keep in mind** that, since all these symbols are elements of GF(q), the meaning of addition and multiplication is that given by the GF(q) + and • tables. Recall that we showed how to construct such tables in Chapter 6. As a reminder of this, one *might* write

$$c_2 = d_1 \bullet b + d_2 \bullet e \qquad (7.17)$$

One must be careful not to use ordinary real-number-field + and • operators. In the matrix and vector formalism, these operators are sort of hidden away, as in the vector/matrix equation (7.3), $\mathbf{c} = \mathbf{d}G$.

(4) In earlier chapters, we sometimes used **bold font** to denote elements of GF(q), to distinguish such elements from normal scalars. For example we had (with p = integer, **g** = field element, $q = p^m$) ,

Fact: $p\mathbf{g} = \mathbf{0}$ for any element **g** of GF($q=p^m$). *(4.5)*

In the current context, we are now using bolded font to indicate vectors in $\mathcal{V}^k$ type vector spaces, so we shall discontinue randomly using bold font for field elements. The above Fact now says

Fact: pg = 0 for any element g of GF($q=p^m$). *(4.5)*

(5) For all our vectors we have starting the **component numbering** with 1, as in $\mathbf{d} = (d_1\ d_2\ d_3)$ just because this seems the simplest thing to do. Obviously one could have written $\mathbf{d} = (d_0\ d_1\ d_2)$ as an alternate notation. Later when we start talking about vector components being the coefficients of polynomials, it will be much easier to start the labeling with 0, then we will have for example $d(x) = d_0 + d_1x + d_2x^2$ so the coefficient subscript will match the power. The reader should be aware of which convention is being used in a given section below.

(6) There is a semantic question about **which symbol is first**. When we write $\mathbf{d} = (d_0\ d_1\ d_2)$, we say that the d_0 symbol is first just because it is on the left and we read left to right. However, in the next chapter when we come to polynomials like $d(x) = d_0 + d_1x + d_2x^2$, we shall find that it is most convenient in the transmission of data to send the most significant symbol first, so if the coefficients of this polynomial

were transmitted from A to B, the temporal ordering of the symbols would be d_2, d_1, d_0. The reason for this ordering is that it greatly simplifies the hardware of both encoders and decoders, as we shall see.

(c) The Perp Space and the Parity Check Matrix H

We have seen that all data vectors in our data space D^k get mapped by G into the code space C^k, which is a subspace of V^n. Within V^n there exists another useful subspace besides C^k. It is the space $(C^k)^\perp$ discussed above formed by the set of all n-tuples in V^n which are **orthogonal** to the code vectors of C^k. In other words, an n-tuple row vector **h** in $(C^k)^\perp$ has this property relative to *any* code word **c**,

$$\mathbf{ch}^T = \mathbf{c} \bullet \mathbf{h} = 0 \qquad \text{// } \mathbf{c} \text{ and } \mathbf{h} \text{ are "orthogonal" or "perpendicular"} \tag{7.18}$$

Such a vector **h** is "an illegal code word" since it lies in V^n but outside C^k. In (7.18) $\mathbf{h}^T$ is a column vector, and $\mathbf{ch}^T$ is then an inner product exactly as shown in (7.15). Note, by the way, that this 0 is a symbol, an element of GF(q). That is because the components of **c** and $\mathbf{h}^T$ are symbols.

As an example, let's take the code with k = 2 and n = 5 so that (7.4) looks like

$$(c_1\ c_2\ c_3\ c_4\ c_5) = (d_1\ d_2\ d_3)\begin{pmatrix} a & b & c & d & e \\ a' & b' & c' & d' & e' \\ a'' & b'' & c'' & d'' & e'' \end{pmatrix}. \tag{7.19}$$

Since vector **h** in $(C^k)^\perp$ has 5 components (like all vectors within V^5) we write $\mathbf{h} = (h_1\ h_2\ h_3\ h_4\ h_5)$ and then (7.18) looks like

$$(c_1\ c_2\ c_3\ c_4\ c_5)\begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{pmatrix} = 0\ . \tag{7.20}$$

It is not hard to show that the set of such orthogonal vectors **h** does in fact form a vector space $(C^k)^\perp$, and that the dimension of this vector space is n-k. Within $(C^k)^\perp$, we can therefore find (n-k) basis vectors **h** which satisfy (7.18) for all code vectors **c**. We can then consider these **h**'s to be the *rows* of a matrix H which has n-k rows and n columns. Then we can write:

$$\mathbf{c}H^T = \mathbf{0} \quad \text{or alternatively, applied to column vectors } \mathbf{c}^T, \quad H\mathbf{c}^T = \mathbf{0} \tag{7.21}$$

where here **0** on the left is a row vector containing n-k 0's, each of which is the 0 symbol of GF(q). The **0** on the right is the corresponding all-zeros column vector.

Since $H\mathbf{c}^T = 0$ for all **c** in C^k, one would say in linear algebra parlance that the column vector code words $\mathbf{c}^T$ inhabit the **nullspace** of the linear operator H, which nullspace is just C^k. This nullspace space consists

of all the points in V^n which are hit by the mapping **c** = **d**G for all data vectors **d**. Thus, the **range** of operator G (all points it hits) = the nullspace of operator H = C^k.

In our example, let us call our set of n-k = 5-3 = 2 **h** vectors by the names **h** and **h**'. Then (7.21) reads

$$\mathbf{c}H^T = (c_1\ c_2\ c_3\ c_4\ c_5)\begin{bmatrix} h_1 & h'_1 \\ h_2 & h'_2 \\ h_3 & h'_3 \\ h_4 & h'_4 \\ h_5 & h'_5 \end{bmatrix} = (0\ 0) \qquad H = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 & h_5 \\ h'_1 & h'_2 & h'_3 & h'_4 & h'_5 \end{pmatrix} \tag{7.22}$$

In coding theory, the matrix H is called the **parity check matrix** of the (n,k) code. We can "check" that a vector **c** really is a code word by computing $\mathbf{c}H^T$ and seeing if the result is **0**. If a received code word flunks this test, we know that there has been a transmission error. But all we know, when an error is detected by $\mathbf{c}H^T \neq \mathbf{0}$, is that one or more of the n symbols in that code word are wrong.

Since the rows of G are a set of code vectors, it trivially follows from the above that:

$$GH^T = 0 \qquad \text{or} \qquad HG^T = 0 \tag{7.23}$$

where now each 0 is an all-zeros matrix.

(d) The Dual Code generated by H

Since matrix H has rank n-k, it can *itself* be considered as the generator matrix of a new code having data words containing n-k symbols. That is, we can define a code:

$$\mathbf{c}' = \mathbf{d}'H \tag{7.24}$$

where **d**' is a (n-k)-tuple and **c**' is an n-tuple. This code is (n,n-k) and is known as the **dual code** to (n,k). All code vectors **c** of the code (n,k) are orthogonal to all code vectors **c**' of the dual code (n,n-k). Here is a little proof:

$$\mathbf{c}'\mathbf{c}^T = (\mathbf{d}'H)(\mathbf{d}G)^T = (\mathbf{d}'H)(G^T\mathbf{d}^T) = \mathbf{d}'\,(HG^T)\mathbf{d}^T = \mathbf{d}'\,(0)\,\mathbf{d}^T = 0\ .$$

The following drawing illustrates the above discussion:

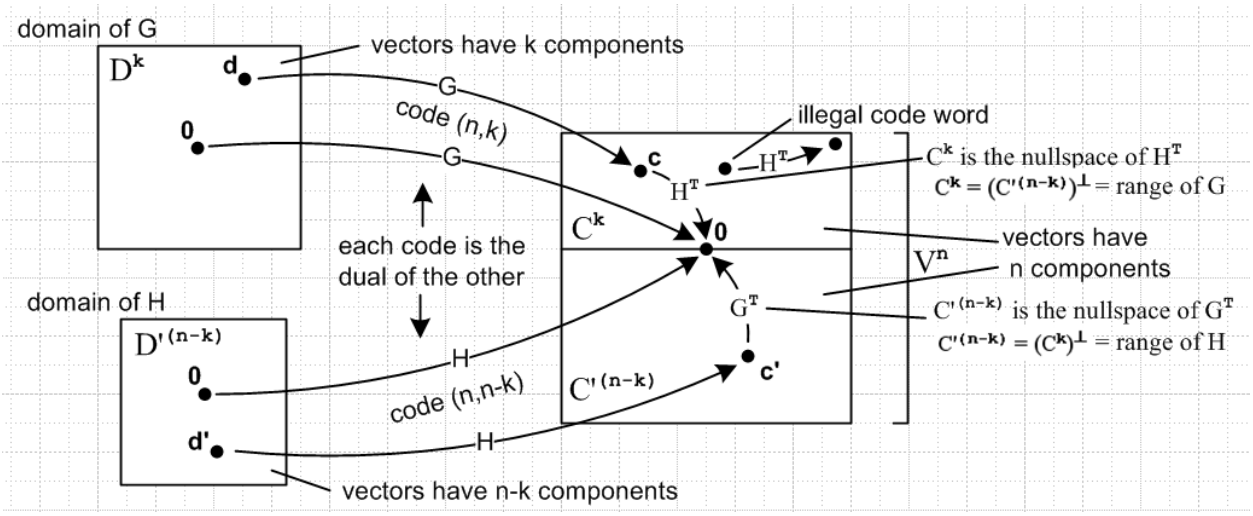


Fig 7.1

(e) The Systematic Basis

A standard manner of expressing the G matrix is

$$G = [P \mid I_k] \,. \tag{7.25}$$

Recall that G has n columns and k rows. The first n-k columns contain some submatrix P, and the last k columns contain a k x k identity matrix I_k. When this G is applied to a data vector **d**, as in **c** = **d**G, the first n-k components of **c** are the parity check symbols obtained by multiplying the vector **d** by the first n-k columns of G, where the P matrix data is located. The last k components of **c** are exactly the components of **d**. This form of matrix G is called the **systematic basis**, and we shall refer to it again in reference to cyclic codes. Here is an example of G in this basis for our case n = 5 and k = 3:

$$\mathbf{c} = \mathbf{d}G: \qquad (c_1\ c_2\ c_3\ c_4\ c_5) = (d_1\ d_2\ d_3)\begin{pmatrix} a & d & 1 & 0 & 0 \\ b & e & 0 & 1 & 0 \\ c & f & 0 & 0 & 1 \end{pmatrix} = (*\ *\ d_1\ d_2\ d_3)\,. \tag{7.26}$$

Note: The term "first" at this point means leftmost.

One might wonder if, given the above matrix G, one can construct a viable H. The answer is yes.

Fact 0: If $G = [P \mid I_k]$, then $H = [I_{n-k} \mid -P^T]$. (7.27)

Before proof, here is the H so constructed from the systematic-basis G shown in (7.26),

$$H = \begin{pmatrix} 1 & 0 & -a & -b & -c \\ 0 & 1 & -d & -e & -f \end{pmatrix}\,. \tag{7.28}$$

Proof: In this proof, the first vector component is labeled with a 0, see remark (b) 5 above.

Notice that H in (7.27) has the right dimensions: n columns and n-k rows. To prove (7.27) we need to show (7.23) that $GH^T = 0$. In other words, we need to show that

$\Sigma_{j=0}^{n-1} G_{ij} H_{sj} = 0$ for all legal values of i and s .

The first index of a matrix M_{ij} denotes the row i, the second j the column. The upper left corner element of matrix M_{ij} is M_{00} (not M_{11}) due to our current labeling convention.

The proof is easy once we find a good way to codify the matrices. First, here is a picture showing the layout of the two matrices G and H,

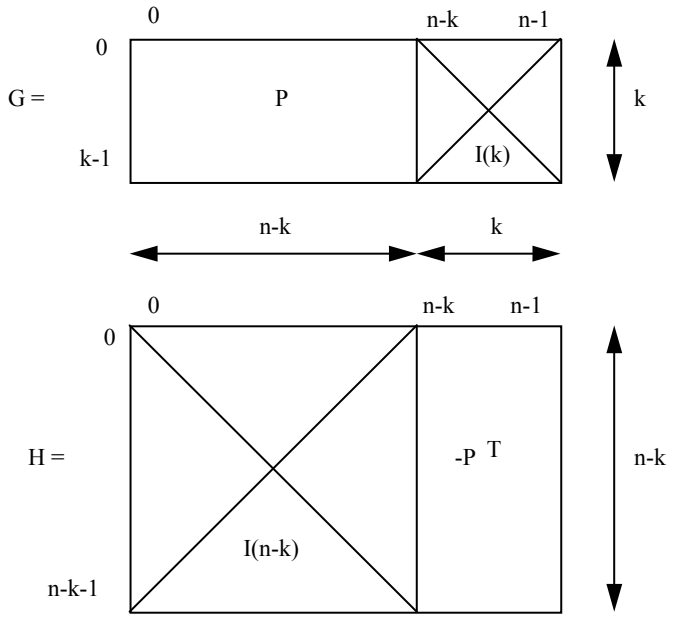


Fig 7.2

We define θ(Boolean) which is 1 for Boolean = true and 0 for Boolean = false. Then staring at the above pictures, and defining x ≡ n-k, one finds

$$G_{ij} = P_{ij} \;\; \theta(j<x) \; + \; \delta_{i,j-x} \;\; \theta(j \geq x)$$

$$H_{ij} = \delta_{i,j} \; \theta(j<x) \; - \; P^{T}_{i,j-x} \; \theta(j \geq x) \; . \qquad (7.29)$$

For example, suppose i = 1 and j = n-k+1 = x+1. This marks the location of the second 1 element on the diagonal of the identity matrix I_k which is part of G. Since j ≥ x, the second term only contributes $\delta_{1,1}$ = 1, as desired. Note that G_{ij} and H_{ij} have a very similar structure in (7.29). From the second line we get, using $P^{T}_{ab} = P_{ba}$ and setting i = s,

$$H_{sj} = \delta_{s,j} \; \theta(j<x) - \; P_{j-x,s} \; \theta(j \geq x) \; .$$

Then

$$\Sigma_{j=0}^{n-1} G_{ij} H_{sj} = \Sigma_{j=0}^{n-1} \left[P_{ij} \, \theta(j<x) \; + \; \delta_{i,j-x} \;\; \theta(j \geq x) \right] \left[\delta_{s,j} \; \theta(j<x) - \; P_{j-x,s} \; \theta(j \geq x) \right] .$$

There are four sums here, but the two "cross term" sums vanish due to θ(j<x) θ(j ≥x) = 0. Thus,

$$\Sigma_{j=0}^{n-1} G_{ij} H_{sj} = \Sigma_{j=0}^{n-1} \{ P_{ij} \delta_{s,j} \theta(j<x) - \delta_{i,j-x} P_{j-x,s} \theta(j \geq x) \} .$$

In the first term j gets pinned to s by $\delta_{s,j}$, while in the second we have j pinned to i+x by $\delta_{i,j-x}$. Thus we find

$$\begin{aligned} \Sigma_{j=0}^{n-1} G_{ij} H_{sj} &= P_{is} \theta(s<x) - P_{(i+x)-x,s} \theta(i+x \geq x) \\ &= P_{is} \theta(s<x) - P_{is} \theta(i+x \geq x) \\ &= P_{is} \theta(s<x) - P_{is} \theta(i \geq 0) . \end{aligned}$$

Since $i \geq 0$ for any i index on G_{ij}, we have $\theta(i \geq 0) = 1$.
Since $s < n-k$ for any index s on H_{sj} (recall s = 0,1,...n-k-1 since H has n-k rows), we have $\theta(s<x) = 1$.
Thus we get

$$\Sigma_{j=0}^{n-1} G_{ij} H_{sj} = P_{is} - P_{is} = 0 . \qquad \textbf{QED}$$

Thus, we have shown that if $G = [P \mid I_k]$ and $H = [I_{n-k} \mid -P^T]$ we get $GH^T = 0$. Taking the transpose of this equation, we also have shown that $HG^T = 0$.

In our example, we can verify that the product really is zero by visual inspection,

$$GH^T = \begin{pmatrix} a & d & 1 & 0 & 0 \\ b & e & 0 & 1 & 0 \\ c & f & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -a-d & \\ -b-e & \\ -c-f & \end{pmatrix} = \begin{pmatrix} a-a & d-d \\ b-b & e-e \\ c-c & f-f \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} . \qquad (7.30)$$

(f) The notion of Distance between code words: Error Correction

We have noted that the code space C^k is a subspace of V^n. There are many n-tuples in V^n which are not code words. Here is a suggestive picture, where the large dots are code words in C^k, and the little dots are other points in V^n -- points which are "illegal" code words. The picture is arranged so adjacent pairs of dots differ by one symbol. Recall that any V^n point has n symbols in its n-tuple (v_1 v_2 .... v_n).

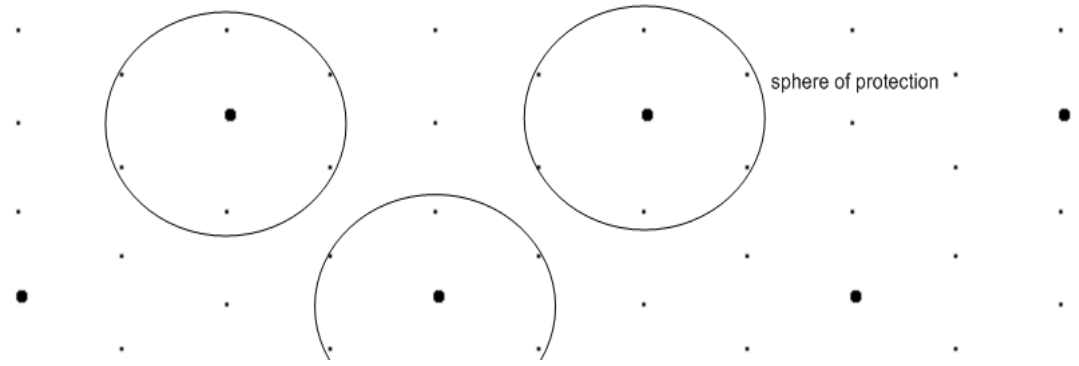


Fig 7.3

Suppose an *error* occurs during the transmission of a code word (•), and a code word is converted into an unused vector (˙). It seems reasonable to assume that such an erroneous code word is a mutation of the "nearest" good code word (•). By "error" we mean that one or more symbols of the code word got hammered during transmission. If the number of such symbol alterations in a code word is t, we say that there was a **t-symbol random error** in that code word.

As the picture above suggests, if the bad code word (˙) is an immediate neighbor of a good word (•), it seems likely that the bad code word was (•) when it was transmitted, so (˙) should be corrected to (•). Conceptually, each legal code word (•) is surrounded by a sphere of protection which contains no other legal code words. Any bad code word in a sphere is corrected to the good code word at the center of the sphere. The radius of the protective sphere is basically t units and in the above drawing t = 1.

In the above symbolic picture, the **distance** between code words is 3. That is, 3 symbols must be changed to get from one (•) to another (•). Thus, if we get some bad code word (˙), it is more likely that it represents a 1-symbol error than a 2-symbol error, so the best thing to do is correct it to the nearest neighboring (•). This is illustrated in the next drawing. The transmitted code word is the (•) in the center of the red hexagon. The vertices of this red hexagon show all possible 1-symbol errors (in this schematic example). Suppose one symbol is changed and we end up at the (˙) at the end of the 1 arrow. An error in some other symbol of the code word could take us to any point on the perimeter of the dashed blue hexagon, and so on. Three symbols of the code word must get changed in this example to arrive at the good code word at the end of the 3 arrow. Such an error would not detected by the parity matrix method of (7.22) and following discussion.

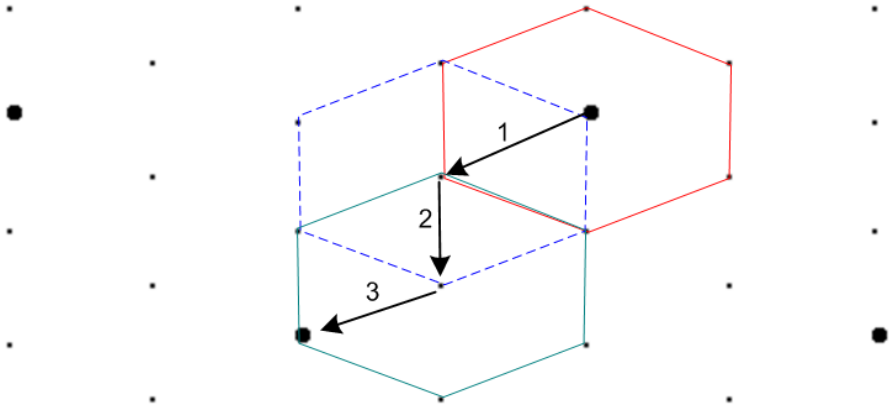


Fig 7.4

In summary, a code word (•) has some "private region" surrounding it, and all bad code words (˙) in this region most likely are mutations of the (•). The larger n is relative to k, and the better "quality" the code has, the larger is this private region surrounding each code word, so more erroneous symbols can be corrected. Of course the larger n is relative to k, the worse the code rate k/n becomes.

Not all code words necessarily have the same size private region. The *smallest* distance between *any* two code words of a code is called d.

d = smallest distance between any two code words of a code

Suppose you wish to be able to "correct" all random errors of t or fewer symbols.

t = number of random errors (in symbols per codeword) that a code can "correct"

We use the term **random error** to indicate errors that occur independently on each symbol of a code word during transmission. Another class of errors are **burst errors**, where there is a coherent destruction of many sequential symbols by a single event (lightning stroke, scratch on a CD). Burst errors are important in coding theory, but we shall have nothing to say about them other than that they exist, and one can calculate their probability and analyze the quality of a code in terms of how well it can correct burst errors. Burst errors can be "spread out" by the use of a transmitting interleaver and a receiving deinterleaver.

If $d \geq 2t+1$, then one can make a clear presumption that a non-codeword vector that results from some random error is closer to one codeword than to any other, and it would best be corrected to this code word. For example, if there were a 2-symbol error, and if the distance between code words were at least 5 symbols (meaning 5 symbols must be changed to convert one codeword into another), then you would have a choice of making a 2-symbol correction to get to codeword A, or a 3-symbol correction to get to codeword B. Probability favors codeword A. Thus, "error correction" is really a probabilistic process, not a foolproof one. Moreover, it is possible that an error (a 5-symbol error in this example) could convert one code word into another. This would be an **undetectable error**.

If the probability of a random error in a symbol of a code word is very small, like 10^{-6} , then it is very reasonable to assume that the a (t+1)-symbol random error is less likely than a t-symbol error:

Probability of (t+1)-symbol error = $(10^{-6})^{(t+1)}$
Probability of (t)-symbol error = $(10^{-6})^{(t)}$ ratio = 10^{-6}

Thus, in this case, the t+1 symbol error is a million times less likely than a t symbol error.

The **distance** between any two code words is the number of symbols that *differ* between the two code words. This is the same as the number of non-zero symbols in the code word which is the difference of those two code words (the difference code word is a code word because C^k is a vector space). Usually this distance is called the **Hamming distance**, but we shall just call it the distance.

The **minimum distance of a code** is defined to be the minimum of the distances between all pairs of code words in the code.

The **weight** of a code word is the number of non-zero symbols in a code word (**Hamming weight**).

Fact 1: The minimum distance d of a code is the minimum of the weights of all the code words. (7.31)

To get a large d, you want code word weights to be large, so you don't want lots of 0's in code words.

Proof: Imagine we take a survey of all N^2 pairs of code words in our code which contains N code words. Suppose we find that the minimum distance between any pair of code words is d, and this occurs for code words c_1 and c_2. The difference code word $c_3 = c_1 - c_2$ will therefore have d non-zero symbols, and that means c_3 will have weight d. In general, in our list of pairs, we will find that the distance is always $\geq$ d, which means the weight of the difference code word will be $\geq$ d. Thus, of all the difference code words generated, the minimum weight will be d. But *all* code words are difference code words for *some* pair of code words (just let the second of the pair be the 0 code word). Since all difference code words have weight $\geq$ d, that means that all code words have weight $\geq$ d, and that means that d is the minimum weight of all code words.

Fact 2: If the minimum distance of a code is d, then the maximum number t of symbol errors per code word that the code can correct (but of course not perfectly) is given by:

$$t = \text{Int}[(d-1)/2] \qquad (\text{Int} = \text{Integer Part}) \tag{7.32}$$

Proof by example:

(d odd) If d = 5, then you have (• · · · · • · · · · •), and any bad code word can be unambiguously mapped to a good code word if the error is less than t=2 symbols. If there were a t $\geq$ 3 symbol error, your correction would fix it as if it were t $\leq$ 2 symbol error, and the result would be wrong.

(d even) If d = 6, then you have (• · · · · · • · · · · · •), and you can still only unambiguously correct t $\leq$ 2 bit errors. An error of 3 symbols which puts you at a middle (·) cannot be corrected without a 50% chance of doing the wrong thing (in our schematic model).

Corollary: The relation between t and d is d = 2t+1 if d is odd, and d = 2t+2 if d is even. (7.33)

Fact 3: There is an upper bound on d or t which applies to all block codes. It is this:

$$d \leq n-k+1 \qquad \text{or} \qquad t \leq \text{Int}[\,(n-k)/2\,]\ . \tag{7.34}$$

Proof: Consider the data unit vector d = (1 0 0 ...). In the systematic basis of (7.26), the weight of the corresponding code vector is at most 1 + (n-k), because at most you might have all n-k parity check symbols non-zero. The minimum weight over all code vectors must certainly be $\leq$ the maximum weight of this one code vector. Thus, we arrive that the fact that the minimum weight $\leq$ 1+n-k . According to Fact 1 (7.31), we set the minimum weight equal to the code distance d to get

$$d \leq (n-k+1) \qquad t \leq \text{Int}[(n-k)/2]$$

The rightmost inequality follows from Fact 2 (7.32) above.

Obviously, the closer you can come to this bound, the happier you are, since you can correct more errors t for a given number n-k of parity check symbols, that is, for a given code rate k/n. Amazingly, certain high quality codes hit this bound exactly. A code which satisfies the above bound as an equality is called **maximum-distance-separable.** The Reed-Solomon codes are examples.

Now a set of columns of a matrix are **linearly dependent** if there exists a linear combination of them that adds to zero. Another way to say this is that any of the columns of a linearly dependent set can be written as a linear combination of the others in the set. For example, three columns C_1, C_3 and C_6 are linearly dependent if $\alpha C_1 + \beta C_3 + \gamma C_6 = 0$ for some non-zero α,β,γ. Then $C_3 = -(\alpha/\beta)C_1 - (\gamma/\beta)C_6$. If two columns are linearly dependent, either can be written as a multiple of the other. We are now ready for:

Fact 4: The minimum distance d of a code is equal to the minimum number of columns of the matrix H which are linearly dependent. (7.35)

This should not be confused with the rank of H, which is n-k, and which is the maximum number of columns which are linearly *in*dependent.

Proof by Example: Suppose the minimum distance of a code is d =3. According to Fact 1 (7.31), the minimum weight of all code words is 3. Suppose 3 columns of H are linearly dependent. Linear dependence of 3 columns means that there is a linear combination of the these 3 columns which adds up to 0. According to our known fact (7.21) that $H\mathbf{c}^T = \mathbf{0}$, we can *interpret* the coefficients in any such linear combination as the coefficients of some code word **c**. For example, in our n=5, k=2 example, suppose the first 3 columns of H are the ones which are linearly dependent. Then we must have

$$\mathbf{0} = H\mathbf{c}^T = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 & h_5 \\ h'_1 & h'_2 & h'_3 & h'_4 & h'_5 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ 0 \\ 0 \end{pmatrix} = c_1 \begin{pmatrix} h_1 \\ h'_1 \end{pmatrix} + c_2 \begin{pmatrix} h_2 \\ h'_2 \end{pmatrix} + c_3 \begin{pmatrix} h_3 \\ h'_3 \end{pmatrix} = \mathbf{0}$$

so the code word **c** has weight 3 (three non-zero symbols). Fine. Now suppose there were 2 columns of H which were linearly dependent, perhaps the last two. Then we would have, for some code word **c**,

$$\mathbf{0} = H\mathbf{c}^T = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 & h_5 \\ h'_1 & h'_2 & h'_3 & h'_4 & h'_5 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ c_4 \\ c_5 \end{pmatrix} = c_4 \begin{pmatrix} h_4 \\ h'_4 \end{pmatrix} + c_5 \begin{pmatrix} h_5 \\ h'_5 \end{pmatrix} = \mathbf{0}\,.$$

But this implies the existence of a code word of weight 2. This contradicts Fact 1 (7.31) which said the minimum weight must be our d=3. Therefore there cannot be any 2 columns which are linearly dependent. Thus in this case the minimum number of dependent columns is 3 which matches the minimum distance of the code. One can repeat the proof for arbitrary d, so the minimum code word weight is d. If one tries to have < d independent columns, one gets a code word of weight < d, which is a contradiction.

Corollary 1: If a column of H is all zeros, then d = 1. (7.36)

Proof: In this case we can write, for example, $\alpha \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \mathbf{0}$, which says that the column $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ all by itself is a linearly dependent set, according to the above linear-combination definition of linear dependence. Since the minimum linear dependent set has 1 column, Fact 4 (7.35) says d = 1.

Corollary 2: If no columns are zeros, but some column i is a multiple of another j, then d = 2. (7.37)

Proof: In this case we have $\alpha \begin{pmatrix} h_i \\ h'_i \end{pmatrix} + \beta \begin{pmatrix} h_j \\ h'_j \end{pmatrix} = 0$, so the minimum linear dependent set has 2 columns and d = 2.

Corollary 3: Suppose H has rank n-k. This is the most the rank of H can be since it has n-k rows. In this case, any set of n-k+1 (or more) columns must be linearly dependent, that is what rank means. The *minimum* number of dependent columns must therefore be ≤ n-k+1. But Fact 4 (7.35) says that the minimum number of dependent columns is d, the minimum distance of the code. Therefore d ≤ n-k+1. Thus then is an alternate derivation of Fact 3 (7.34). (7.38)

(g) What does the real code space picture look like?

In the above discussion, we used "suggestive" pictures like Fig 7.3 to show the relationship between code points (•) and non-code points (˙) in the embedding space V^n. In order to draw a true picture, we have to start out with a lattice of non-code points (˙) in an n-dimensional space, and then we have to mark off those lattice points which are code points with our heavy dots (•). We don't really know how to draw a lattice with n > 3, but we can try to do this with n = 2 and n = 3 and see what can be learned.

Case n = 2 and k = 1

The code generating equation (7.4) has this very simple form

$$(c_1\ c_2) = (d_1)\ (a\ b)$$
$$\mathbf{c} \quad = \quad \mathbf{d} \quad G$$

Let's assume the code symbols lie in GF(2^8) so d_1 can take 256 values so we at least have a lattice of some large size, and we can then draw a piece of it near the origin. If we take (a,b) = (3,1), we can in fact make a reasonable drawing of the V^2 space:

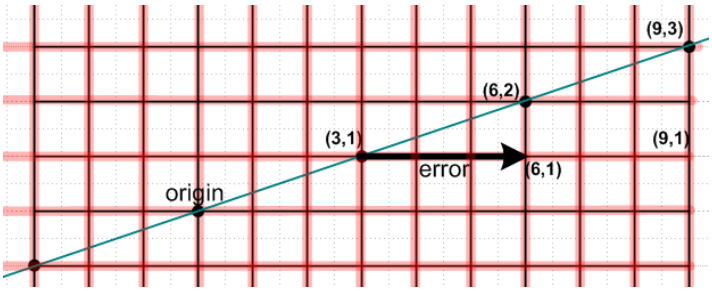


Fig 7.5

The thin black lines are the lattice and their intersections are the lattice points, mostly illegal code words (·). The legal code words lie on the green line at the large dots (•). The pink bands show the effect of single-symbol errors. For example, suppose the legal code word (3,1) was received as the illegal code word (6,1) due to a single-symbol error in the first symbol of the code word (c_1 c_2). Since we *don't know* which symbol is wrong, c_1 or c_2 (all we know is that $\mathbf{c}H^T = \mathbf{s} \neq 0$), this received code word could have started life as either (3,1) or (6,2). Both of these legal code words differ from the bad code word (6,1) by a single-symbol error. We then don't know whether to correct this code word to (3,1) or to (6,2). These two code words are separated by distance d = 2, and from (7.32) this means t = 0 and this code cannot correct *all* single-symbol errors. If the error vector above happened to be one box shorter in length, then the code could conclude that (3,1) was the proper corrected code word.

If a code has some t > 0, that code must be able to correct *all* t-symbol errors. So to get anything reasonable in terms of error correction capability, we have to turn to n > 2. So next we try n = 3.

Case n = 3 and k = 1

The code generating equation (7.4) now has this form,

$$(c_1\ c_2\ c_3) = (d_1)\ (a\ b\ c) \ .$$
$$\mathbf{c} \quad = \quad \mathbf{d} \quad G$$

The reader must now imagine a 3D lattice in which the valid code words lie on a skewed green line similar to that shown above. Perhaps we select (a,b,c) = (1,2,3). It is painful to attempt a full rendering of the V^3 space as done above, but we can draw a very small piece of this space which shows only two code words (1,2,3) and (2,4,6) lying on the skewed green line. These (•) code words lie on opposite corners of the lattice box shown. Not all lattice lines are drawn.

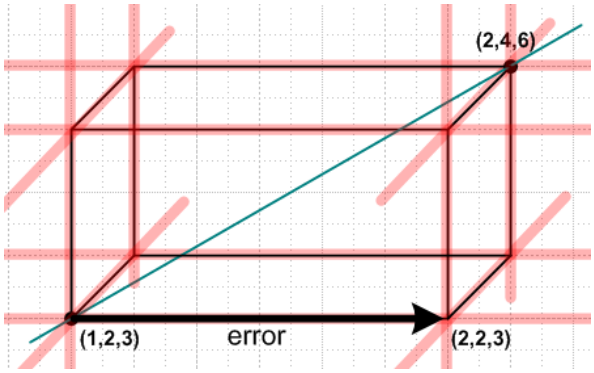


Fig 7.6

Suppose there is an error in the first symbol of the code word (1,2,3) so it is received as (2,2,3). This then is a single-symbol error. If we could correct it, what would we correct it to? The point (2,2,3) lies on the intersection of three single-symbol-error pink bands. Because our V^n space has more dimension now than it did in the previous example, we have a new situation. If we probe along the vertical pink band through (2,2,3) we find that it hits no code points. The same is true if we probe along the front-to-back pink band. Only the horizontal pink band through (2,2,3) goes through a code word, namely, (1,2,3). So in this example, if we receive the bad code word (2,2,3), and if we assume there has been only a 1-symbol error,

then we *know* we should correct (2,2,3) to be (1,2,3). In this example we have d = 3 since any two code words differ in all three symbols. Then from (7.32) we have t = Int[(d-1)/2] = Int[2/2] = 1 which agrees with our conclusion just developed.

In this example we require a 3-symbol error to get from one legal code word to another. If we pile up several copies of the above box following generally along the green line, we get,

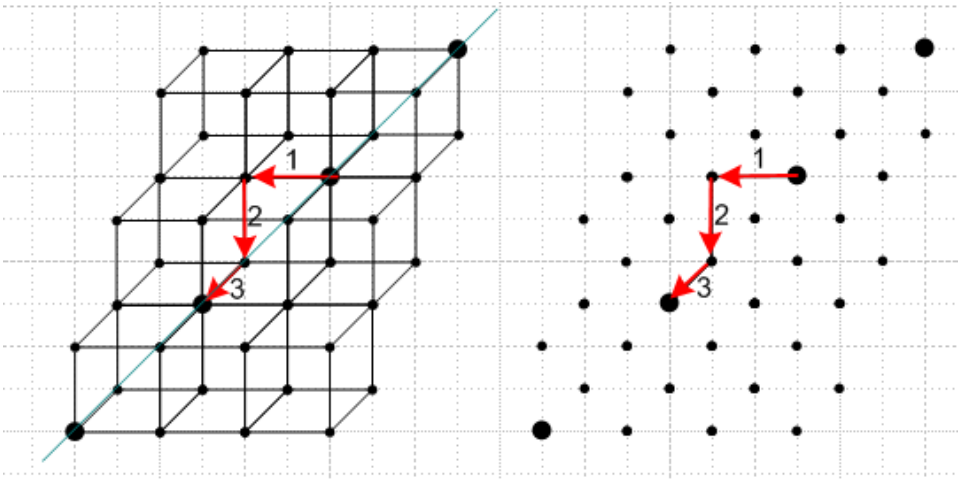


Fig 7.7

where on the right the V^n grid segments have been deleted. The picture on the right at least bears a dim topological resemblance to our symbolic Figures 7.3 and 7.4. The notion of the three numbered arrows in Fig 7.4 (each a single symbol error) to get from one code word to another is also born out in the above drawing. One difference between Fig 7.4 and Fig 7.7 is that the code points (•) are collinear in the latter figure, but in the general case discussed next, the code points will lie on a hyperplane of dimension more than 1, and the code points will then no longer be collinear, and Fig 7.4 becomes more "accurate".

General n and k

Now we have to first imagine an n-dimensional lattice of (·) points for V^n. In this lattice we then imagine a skewed hyperplane of dimension k (this is C^k) which intersects some of the lattice points which we then mark as (•). Through each such valid code point (•) we then drawn n pink bands parallel to the V_n axes, showing the effect of single-symbol errors. Assume for our selected code that $t \geq 2$. This code, of the form (7.4), determines the orientation of that hyperplane in V^n. We will find that, if we start at some bad code word lattice point, and if we trace all *double* paths first onto one pink band and then from that onto any second pink band, only one such double path will lead to a code word point (•), and that is the code word we want to correct to. In order to get to any other (•) from our bad (·), we will have to travel on at least three different pink bands. These facts are a challenge to illustrate in a drawing.

Hopefully this discussion explains why we grudgingly accept schematic representations of the V^n embedding space such as Figure 7.3 and 7.4. The theory of block codes must really be done algebraically without the assistance of precise graphical aids.

(h) Encoders and Decoders: The Syndrome

An "encoder" is a piece of hardware or software which in effect applies the matrix G to data words to generate codes words, **c** = **d**G as in (7.3). It normally does this in the systematic basis, so G has the form (7.25). The data symbols are passed through and n-k parity check symbols are added, as in (7.26).

A "decoder" is a piece of hardware or software which receives the code words, and tries to determine if there have been errors, and may try to correct some errors.

When a bad code word is received, that is to say, some $\mathbf{c}' \neq \mathbf{c}$ because an error occurred in transmission, the so-called **syndrome s** is the result of the application of H^T to **c**'. Recall from (7.21) that $\mathbf{c}H^T = \mathbf{0}$ for good code words. Thus, the syndrome is defined as

$$\mathbf{s} \equiv \mathbf{c}' H^T . \tag{7.39}$$

One can express the bad code word as a good code word plus an error code word, so $\mathbf{c}' = \mathbf{c} + \mathbf{e}$. Then

$$\mathbf{s} = \mathbf{c}' H^T = (\mathbf{c} + \mathbf{e}) H^T = \mathbf{e} H^T . \tag{7.40}$$

Since H^T has n-k columns, the syndrome row vector **s** has n-k components (**s** = 0 example in (7.22)). A non-vanishing syndrome vector **detects** an error; good code words always have zero syndrome. An error correcting decoder attempts to look up the most likely **e** for a given **s**, and then corrects the error by subtracting the error back out of **c**' to recover the good code word,

$$\mathbf{c} = \mathbf{c}' - \mathbf{e} \quad \text{where} \quad \mathbf{s} = \mathbf{e} H^T \tag{7.41}$$

Exactly how this is done for various specific codes is the subject of error correction texts such as Peterson and Weldon or Rhee. If it happens that the damage to **c** is enough to knock **c**' into another good code word, then $\mathbf{e} \equiv \mathbf{c}' - \mathbf{c} \neq 0$ but **s** = 0 and the error goes undetected.

(i) Important codes, code history, and other kinds of codes

Linear block codes are characterized by their values of n and k, by their d (which is really their ability to correct t-symbol random errors), and by the complexity/expense of the required encoders and decoders. Another factor is the ability to correct burst errors. All these things are measures of the **performance** of a code. Some of the names of famous block codes are Hamming, Golay, Hadamard, Reed-Muller, Fire, BCH, and Reed-Solomon. There are many more.

The study of error-correcting codes is a relatively recent development on the time scale of the supporting math. Although Galois died in 1832, Shannon's famous paper was published in 1948. Golay codes arrived in 1949, then the Hamming codes were discovered in 1950. Fire codes came in 1959, especially formulated to correct long burst errors. In 1960 came the large class of BCH codes (discussed below) that could correct multiple random symbol errors. The Reed-Solomon codes also appeared in 1960. These are BCH codes applied to code symbols in $GF(2^m)$ and are thus finding much current use since digital

hardware uses "buses" m bits wide. They have good burst-error-correcting capabilities and are used, for example, in CD recorders and players.

Convolution codes are in a completely different class from block codes, and were developed in the 1960's. You produce n code symbols based on k data symbols, but also based on some number m of *previous* data symbols, so there is no clearly defined "block" that is the basic encoded unit. These systems use encoders which look somewhat like polynomial multipliers in that they store some number of previous data symbols in flip-flops and have lots of feed-forward adders. There is a definite state-machine flavor here, due to the memory of previous symbols. Convolution codes are more complex to analyze than linear block codes, but implementation can be very efficient. Topological methods such as tree or trellis diagrams are used to understand them. One popular decoding method is the Viterbi algorithm associated with Markov chains that appeared in 1967. In contrast, the block codes and especially the cyclic block codes have a strong "algebraic" flavor. For example, they might involve the algebra of Galois Field elements. According to Rhee, the 1970's were spent mostly finding codes of longer length and better performance, while the 1980's concentrated on practical applications. Block and convolution codes were combined in various ways, always seeking better performance. In 1993 **turbo codes** were discovered which allow very close approach to the Shannon channel-capacity limit in information theory. These codes have somewhat supplanted the older codes in both research and application.

Chapter 8: Cyclic Codes

In this section, component numbering of data and code words begins with 0 instead of 1.

Cyclic codes are a subset of the linear block codes. The name arises from the fact that, in a cyclic code, any cyclic permutation of a code word is also a code word. For example, with n = 3, if $\mathbf{c} = (c_0\ c_1\ c_2)$ is a code word, then $\mathbf{c}^{(1)} \equiv (c_2\ c_0\ c_1)$ and $\mathbf{c}^{(2)} \equiv (c_1\ c_2\ c_0)$ are also code words.

We shall define a cyclic code in (8.1) below. This definition makes no mention of anything being cyclic, but later in section (f) we shall prove that a code defined by (8.1) is in fact a cyclic code.

Example : Suppose k = 2 and n = 3 and $q = p^m = 2^1 = 2$. The embedding space of the code words V^3 must contain $2^n = 2^3 = 8$ vectors, but only $2^k = 2^2 = 4$ of these vectors are code words. We can arrange the 3-tuples of V^3 into rows where the elements of each row are cyclically related:

(0 0 0)	(0 0 0)	(0 0 0)	1
(1 0 0)	(0 1 0)	(0 0 1)	3
(1 1 0)	(0 1 1)	(1 0 1)	3
(1 1 1)	(1 1 1)	(1 1 1)	1

Here you see the 8 distinct elements of V^3. We know that a (3,2) code only has $2^k = 2^2 = 4$ code words, and we know that (0 0 0) must be the code word corresponding to **d** = (0 0). Thus, if this is a cyclic code, the code must consist of (0 0 0) and just one of the middle two rows.

(a) Definition of a Cyclic Code: The Cyclic Basis

A cyclic code is constructed (and therefore defined) by the sequence of instructions given below. All polynomials referred to have coefficients in some field GF(q=p^m). The three key polynomials are these :

$g(x) = \Sigma_{i=0}^{n-k} g_i x^i$	degree = n-k	#coefficients = n-k+1	"generator"
$d(x) = \Sigma_{i=0}^{k-1} d_i x^i$	degree ≤ k-1	#coefficients = k	"data word"
$c(x) = \Sigma_{i=0}^{n-1} c_i x^i$	degree ≤ n-1	#coefficients = n	"code word"

The code word $\mathbf{c} = (c_0, c_1 ... c_{n-1})$ is embedded into the code word polynomial c(x), and similarly **d** is embedded into d(x). Since there is a 1-to-1 relation between the vectors and the polynomials, we sometimes loosely refer to c(x) as a "code word" and d(x) as a "data word"

Definition of a Cyclic Code (n,k) (8.1)

(a) Select an integer value for n, the desired length of the code words.

(b) Find a pair of polynomials g(x) and h(x) such that

$$g(x)h(x) = x^n - 1$$

where the degree of h(x) is k and the degree of g(x) is n-k. The polynomial g(x) is the **generator** of the (n,k) cyclic code, and h(x) is the generator of the dual code (n,n-k).

(c) Let a set of k data symbols (in GF(q)) be the coefficients of a polynomial d(x) of degree ≤ k-1.

(d) The code words are then the n coefficients of polynomials c(x) where

$$c(x) = d(x)\, g(x) \quad .$$

Since d(x) has degree ≤ k-1, and g(x) has degree n-k, c(x) is of degree ≤ n-1, which means it has n coefficients or symbols in GF(q). In fact, [n-k ≤ degree of c(x) ≤ n] since d(x) has degree n-k.

Comment**:** One might wonder just how one goes about finding polynomials g(x) which divide x^n-1 and whether in fact there are any such polynomials for a given n, and if so, what degrees these polynomials have. One way to find viable g(x) is to do a blind search of all possible g(x) of degree < n. This method is illustrated in **Appendix E**. The main effort of Appendix E, however, is to demonstrate that for any integer n > 1, there does exist at least one g(x) that divides x^n-1 and this g(x) happens to be irreducible over GF(p). It turns out that this g(x) is a minimum polynomial of GF(p^m) where m = φ(n), Euler's totient function mentioned back in Chap 5 (d) (and also in Appendix G).

Fact 0: When one carries out the polynomial multiplication c(x) = d(x) g(x), the coefficients c_s of c(x) can be expressed in terms of those of d(x) and g(x) according to,

$$c_s = \sum_{j = \max(0,s-k+1)}^{\min(s,\, n-k)} d_{s-j}\, g_j \quad \text{for } s = 0,1...n-1 \qquad c(x) = \Sigma_{s=0}^{n-1} c_s\, x^s \tag{8.2}$$

where we have used the result of Appendix C with A = k-1 and B = n-k and A+B = n-1.

We will show in the next section how it is possible to simplify the messy convolution of (8.2) by replacing the polynomials c(x) and d(x) with certain C(x) and D(x), a transformation known as going to the **systematic basis** in the cyclic world. We have already seen the systematic basis at work in the matrix world as in (7.25) and (7.26),

$$\mathbf{c} = \mathbf{d}G \qquad G = [P \mid I_k] \qquad (c_0\, c_1\, c_2\, c_3\, c_4) = (d_0\, d_1\, d_2) \begin{pmatrix} a & d & 1 & 0 & 0 \\ b & e & 0 & 1 & 0 \\ c & f & 0 & 0 & 1 \end{pmatrix} = (pc_0\, pc_1\, d_0\, d_1\, d_2)$$

In this form, or basis, the n-symbol code words c_i have their left n-k symbols being parity check symbols pc_i, and their right k symbols being the data symbols. Recall from Chapter 7 (b) 6 that the symbols are transmitted from right to left, so in the above example d_2 is sent first.

In the **cyclic basis** with c(x) = d(x)g(x) these symbols are convoluted together in the n-bit code words according to (8.2). We shall soon show how the systematic and cyclic bases are related.

For the moment, imagine working in the cyclic basis. It is not hard (section (j) below) to build a piece of hardware that computes c(x) = d(x)g(x). The circuit is a **polynomial multiplier**.

The code word c(x) is then transmitted over a channel. The receiver, to nobody's great surprise, tries to recover d(x) by dividing c(x) by g(x): d(x) = c(x)/g(x). The hardware for doing this is called a **polynomial divider.**

Example: In a Reed-Solomon code, the code symbols are carried on parallel groups of bits, so the R-S polynomial multipliers and dividers have data paths that are multiple bits wide. They look very much like digital filters, with one significant difference which we will describe in Chapter 9 (e) below.

(b) The Systematic Basis versus the Cyclic Basis

In practice, one tends to use hardware (or software) which works in the systematic basis rather than the cyclic basis. The reason is that error detection and correction are much easier in the systematic basis, since one knows where the data and parity check symbols are in the code words. We are soon going to show that the two bases have a 1-to-1 relationship. This means that the set of polynomials { c(x) } is simply a rearrangement of a set of polynomials { C(x) }. The theory of cyclic codes (including a heavy dose of Galois Field theory) uses the cyclic basis, whereas the practice of cyclic codes uses the systematic basis.

According to our polynomial Division Algorithm (3.6), we can write the following expansion,

$$x^{n-k}\, d(x) = D(x)g(x) - \gamma(x)\ . \tag{8.3}$$

In other words, we multiply d(x) [which carries our k data symbols as coefficients] by a power, and then divide this product by the generator polynomial g(x) which, recall, has degree n-k. Then D(x) is the quotient polynomial, and - γ(x) is the remainder. Whatever γ(x) is, it is of degree less than n-k and thus has n-k coefficients. Since the LHS has degree ≤ (n-k) + (k-1) = (n-1), and since g(x) has degree (n-k), D(x) must have degree ≤ (k-1), the same as d(x).

Suppose we now define our code word to be

$$C(x) = D(x)g(x) \tag{8.4}$$

$$= \gamma(x) + x^{n-k}\, d(x)\ . \tag{8.5}$$

If you write out the coefficients of C(x) (from low to high power), you find from (8.5) that the first n-k of them are the coefficients of $\gamma(x)$, and the last k of them are the data bits of d(x). For example, if n=5 and k=3,

$$C(x) = (\gamma_0+\gamma_1 x) + x^2(d_0+d_1 x+d_2 x^2) = \gamma_0+\gamma_1 x+d_0 x^2+d_1 x^3+d_2 x^4 \rightarrow (\gamma_0, \gamma_1, d_0, d_1, d_2) \ . \qquad (8.6)$$

Therefore, the coefficients of $\gamma(x)$ are precisely the n-k parity check symbols which one combines with the k data symbols to get an n-symbol code word. Thus, we are now in the systematic basis. In the cyclic basis, we had code words c(x) = d(x)g(x), whereas in the systematic basis, we have C(x) = D(x)g(x).

Comparing the $(\gamma_0, \gamma_1, d_0, d_1, d_2)$ of (8.6) to the $(pc_0\ pc_1\ d_0\ d_1\ d_2)$ of our example below (8.2), it seems clear that one could come up with a G matrix of the form $G = [P \mid I_k]$ which would give the same result as C(x) = D(x)g(x), so there is an alignment between the "systematic basis" **c** = **d**G of the matrix world and the "systematic basis" C(x) = D(x)g(x) of the polynomial world.

The systematic basis plan will be to multiply polynomials in our transmitting encoder C(x) = D(x)g(x), and to divide polynomials in our receiving decoder D(x) = C(x)/g(x). We have already noted that the coefficients of a polynomial are transmitted most significant symbol first. We certainly know from our pre-calculator experience with long division (see (3.7)) that we start with the most significant digit of a dividend, here C(x). A hardware polynomial divider is no different. Thus, in terms of time ordering, when we think of C(x) as described above, the most significant symbols are the data symbols that come first in time, then the least significant symbols are the parity check symbols which come last in a data stream of coefficients which represents C(x). Also, within each group (data symbols, parity check symbols), the time ordering is most significant first..

In either cyclic or systematic basis, the code word is the product of some sort of data polynomial of degree ≤ k-1 with the generator polynomial of degree n-k to give a code word polynomial of degree ≤ n-1. The construction which led to C(x) = D(x)g(x) is what one needs do to untangle the convolution (8.2) inherent in the cyclic basis method c(x) = d(x)g(x).

(c) Implementation of Encoders and Decoders

If C(x) is sent through a channel, how do we recover d(x) at the receiving end? We just grab the first (in time) k symbols of C(x) and these make up d(x), according to (8.5) and (8.6).

How do we know if there was an error? We set up a polynomial divider to compute C(x)/g(x). The quotient is D(x), and looking at (8.4) the remainder is supposed to be zero! [Remember that remainder $\gamma(x)$ comes from a different division, $x^{n-k}d(x)$ by g(x).] That is, true code words C(x) are multiples of g(x), so there should be no remainder. If there is a non-zero remainder, then there has been an error. This remainder is a form of the *syndrome* referred to earlier. Thus, at the receiver we have:

$$C'(x) = D'(x)\, g(x) + s(x) \qquad s(x) = \text{Rem}[C'(x)/g(x)] \qquad (8.7)$$

where now s(x) is a syndrome polynomial. Here we indicate by C'(x) a received code word polynomial that has an error. D'(x) and s(x) are the quotient and remainder of C'(x)/g(x). If there was no transmission error, then D'(x) = D(x), C'(x) = C(x) and s(x) = 0. If $s(x) \neq 0$, then we have *detected* an error.

If $s(x) \neq 0$, then the coefficients of s(x) can be used to *correct* the error, provided that the size of the error was less than t symbols and the selected code can correct t-symbol errors. A large fraction of any book on error-correcting codes deals with the details of how this is done for various codes, and this is where implementations can become quite complex.

Obviously, it is the fact that there are extra parity check symbols in the code which allows an error to be detected and possibly corrected.

Let us take one more look at the encoding and decoding equations for a cyclic code in the systematic basis:

encode $C(x) = D(x)\, g(x) = \gamma(x) + x^{n-k}\, d(x)$ (8.8a)

decode $C'(x) = D'(x)\, g(x) + s(x) \qquad s(x) = \text{Rem}[C'(x)/g(x)]$ (8.8b)

error polynomial: $E(x) \equiv C'(x) - C(x)$ (8.8c)

In this notation, $E(x) \neq 0$ means there was an error, while $s(x) \neq 0$ means the error was detected. In the case that C'(x) is another legal code word, we will have $E(x) \neq 0$ and $s(x) = 0$ so the error won't be detected. The dcoder only knows s(x), it doesn't know E(x).

We wish to stress the extreme simplicity of implementing these functions in hardware, apart from the error correction. The encoder is a two-stage process. In the first stage, we just bypass through the k data symbols into "the channel" (most significant first), since these are the higher coefficients of C(x) as in (8.6). At the same time, we run these same symbols into a polynomial divider set up do compute $[\, x^{n-k}\, d(x)\,] \,/\, g(x)$. The factor x^{n-k} is a trivial complication in the implementation of such a divider (it turns out). We then throw out the quotient D(x); it goes into a bit bucket. When the division is done, the flip-flops of the divider contain the remainder $-\gamma(x)$ as shown in (8.8a). From this $\gamma(x)$ is obtained and the coefficients of $\gamma(x)$ are then shifted out into the channel and we are done. C(x) is built and sent.

A non-correcting decoder is also a two-stage process. The first k symbols of C'(k) are the data symbols we want. They are received and parked in local memory within the decoder. At the same time, all n symbols of C'(x) are run through a polynomial divider which computes C'(x)/g(x). The quotient is again thrown out, and the remainder is the syndrome s(x), again as shown in (8.8b). If $s(x) \neq 0$, an error has been detected. Error correction logic can then attempt to correct the data word that has been temporarily parked in memory before it is sent out from the decoder to the rest of the receiving system.

Drawings of hardware polynomial mulitipliers and dividers are shown in Fig 8.2 and Fig 8.3 below.

In a practical transmission system, data transmitted may be interleaved just before transmission and then deinterleaved by the receiver, in order to disperse the effect of a possible burst error in the channel. The same system is used on an audio CD to spread out the effect of a surface scratch. The encoder might also encrypt the data being sent, requiring decryption at the receiving end.

(d) Cyclic Redundancy Check (CRC)

If we ignore error correction, the circuit described above is exactly how a CRC system works. There is some generator polynomial g(x). The parity check symbols are computed as shown above and are tacked onto the end of the data stream to form C(x). At the receiver, the data symbols are siphoned off as needed. The entire incoming packet (coefficients of C'(x)) is run through a divider and the syndrome remainder is computed. If it is not zero, there was a "CRC error".

Of course the number of data symbols k can be somewhat larger than one usually thinks of in a "code". One might have a block of k = 100,000 data symbols followed by some parity check symbols. Although we associate CRC with a cyclic code, it is not required that g(x) divide $x^n - 1$ for an (n,k) code used for CRC purposes. If g(x) does not divide $x^n - 1$, the code words won't be cyclic, but we don't care. Thus, if one first selects a CRC checksum length n - k, which is also the degree of g(x), one can then set either n or k arbitrarily. It is shown below, however, that in order to detect double-bit errors, g(x) must have a period $\geq$ n for the selected value of n.

The probability of detecting an error increases with the degree of g(x) which is the number of parity check symbols n-k. Typical numbers used are n-k = 8, 16 or 32.

In **Appendix F** we derive some classic "facts" about CRC error detection. Here we just quote those facts. The requirement $g(0) \neq 0$ just means g(x) is not of the form $x^i f(x)$ and g(x) $\neq$ a constant.

The first two Facts are not particularly impressive since they can be achieved for GF(2) by adding a single parity check bit to the end of a data stream.

Fact 1: If $g(0) \neq 0$, a cyclic code generated by g(x) over GF(p) detects all single-symbol errors. *(F.1)*

Fact 2: If g(x) = (x-1)f(x), a cyclic code generated by g(x) over GF(2) detects all odd-number-bit errors. *(F.2)*

Fact 3: If $g(0) \neq 0$ and g(x) has period $\geq$ n, a cyclic code generated by g(x) over GF(2) detects all double-bit errors and all single-bit errors. *(F.3)*

Fact 4: If $g(0) \neq 0$ *and* g(x) = (x-1)f(x) *and* g(x) has period $\geq$ n, a cyclic code generated by g(x) over GF(2) detects all single-bit, double-bit, triple-bit and all other odd-number-bit errors. *(F.4)*

Definition: A **burst error** of length r refers to an arbitrary amount of damage done to the symbols of a transmitted code word provided this damage is limited to some extent of width r. For example, if symbol c_4 is the first damaged symbol and c_9 is the last damaged symbol, the following illustrates a burst error of length r = 6. Any or all of the symbols c_5 through c_8 might also be in error.

$$c' = (c_0, c_1, c_2, c_3, \mathbf{c_4, c_5, c_6, c_7, c_8, c_9}, c_{10}, c_{11}, c_{12}, c_{13}, c_{14} \ldots\ldots)$$

The following two facts state quite impressive capabilities of CRC error detection:

Fact 5: If $g(0) \neq 0$, a cyclic code generated by g(x) over GF(p) detects all burst errors of length n-k or smaller. *(F.5)*

Fact 6: The cyclic code of Fact 5 detects all burst errors of length $r \le n\text{-}k$. It does not detect all burst errors for $r > n\text{-}k$, but statistically it can detect a lot of them. For GF(2), here are the conclusions:

$$\begin{aligned} &\text{fraction of burst errors NOT detected for } r = (n\text{-}k) + 1 \;=\; 1/2^{(n-k-1)} \\ &\text{fraction of burst errors NOT detected for } r > (n\text{-}k) + 1 \;=\; 1/2^{(n-k)} \end{aligned} \qquad (F.6)$$

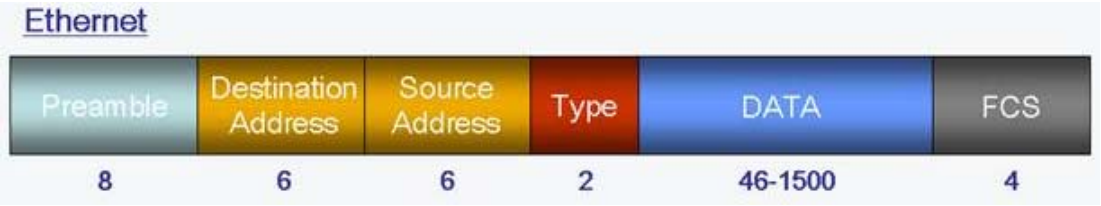
For example, if n-k = 32, then $1/2^{(n-k-1)} = 1/2^{31} \sim 10^{-9}$. For such a code, only one burst error out of a billion goes undetected even if the burst extent is the entire length-n packet! This means that almost all multi-symbol errors of any arrangement will be detected. And *all* burst errors of extent 32 or less are detected.

An error goes undetected if the error pattern converts the transmitted code word into another code word, since the syndrome will then be zero. The above facts dimly suggest that there is a Hamming distance d ~ 3 between the legal codewords and that complicated burst errors within an n-k extent cannot convert one code word into another.

The CRC idea was concocted in 1961 by Peterson and Brown (see Refs). This is the same W.W. Peterson (1924-2009) of the Peterson and Weldon classic text on error-correcting codes (first edition also 1961). The authors note that the CRC codes they discuss are equivalent to the Hamming codes which can correct all 1-bit errors, t = 1 and d = 3. These Hamming codes are discussed in (9.23).

Example: Ethernet CRC-32

A CRC scheme (called CRC-32) is used to protect Ethernet packets (called frames) during transmission. The generator g(x) is of degree n-k = 32 and so there are 32 parity check symbols (bits) appended to the data packet. These check bits are packaged into four bytes which is then called the Frame Checksum Sequence or FCS. The CRC-32 scheme benefits from the impressive burst error detection capability noted just above. Here is an Ethernet frame in one of the standard formats, where numbers are bytes :



Normally the "data" part is 1500 bytes (called the payload or MTU), so the packet without the CRC is then 1522 bytes or k = 12,176 bits. Then n = k + (n-k) = 12,176 + 32 = 12,208 bits. The CRC-32 generator polynomial g(x) is this,

$$g := x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

which the Maple command factor(g) mod 2 shows is irreducible in GF(2). In order to have the double-bit detection capability mentioned above, g(x) must have a period ≥ 12,208. We don't know how to find the

period of the above g(x), but it is likely to be a very large number, much greater than 12,208 . We tested using Maple and were able to show period ≥ 1,600 but then things slow down too much to be useful.

```
for n from 0 to 1600 do
  if (n mod 50) = 0 then print("counter",n); fi;
  r := rem(x^n-1,g,x) mod 2;
  if type(r,numeric) and r = 0 then print(n); else print(n, "no go") ;fi
od:
```

If g(x) were a primitive polynomial of GF(2^{32}), the period would be 2^{32}-1 ~ 4 billion. However, it is known that g(x) is not a primitive polynomial. Presumably g(x) is a minimum polynomial of GF(2^m) for some m ≥ 32 and then the period of g(x) is one of the divisors of 2^m- 1.

(e) The 1-to-1 relationship between the Cyclic Basis and the Systematic Basis

In the cyclic basis, the code words of a cyclic code are formed as c(x) = d(x)g(x), whereas in the systematic basis, the code words are C(x) = D(x) g(x), see (8.1d) and (8.4).

Fact 1: Given any data word polynomial d(x), we can find D(x) of the systematic code. (8.9)

Proof: The connection between data polynomial d(x) and D(x) is given by (8.3),

$$x^{n-k}\, d(x) = D(x)\, g(x) - \gamma(x). \qquad (8.3)$$

For any d(x) of degree s ≤ k-1, the Division Algorithm (3.6) says there exists a unique D(x) of degree s ≤ k-1 which is the quotient of the above expansion over g(x) which has degree n-k. The remainder -γ(x) is also unique, and of degree < n-k.

Fact 2: Given any D(x) of the systematic code, we can find it's data word polynomial d(x). (8.10)

Proof: Assume D(x) has degree s ≤ k-1. Form the product D(x)g(x) which then has degree s+n-k. Expand D(x)g(x) using the Division Algorithm (3.6) over the function x^{n-k}. This yields unique quotient q(x) and remainder r(x) :

$$D(x)g(x) = q(x)\, x^{n-k} + r(x)$$

q(x) has degree s ≤ k-1
r(x) has degree less than n-k

Rearrange to get first line below, and recall (8.3) as the second line,

$$x^{n-k}\, q(x) = D(x)g(x) - r(x)$$
$$x^{n-k}\, d(x) = D(x)\, g(x) - \gamma(x). \qquad (8.3)$$

Thus we find these candidate polynomials for d(x) and γ(x):

$$d(x) = q(x) \qquad \gamma(x) = r(x)$$

Fact 3: The mapping between d(x) and D(x) is one-to-one. Thus, one can regard either set of polynomials as a rearrangement of the other set of polynomials. (8.11)

Proof: Suppose different $d_1(x)$ and $d_2(x)$ both map into the same D(x). Then

$$x^{n-k}\, d_1(x) = D(x)\, g(x) + \gamma_1(x)$$
$$x^{n-k}\, d_2(x) = D(x)\, g(x) + \gamma_2(x)$$

Subtract to get:

$$x^{n-k}\, [\, d_1(x) - d_2(x)\,] = \gamma_1(x) - \gamma_2(x)\ .$$

The LHS is a polynomial of degree $\geq$ n-k. The RHS is a polynomial of degree < n-k. This is a contradiction, so one cannot have two different d's which map into the same D.

Similarly, suppose different $D_1(x)$ and $D_2(x)$ map into the same d(x). Then write

$$x^{n-k}\, d(x) = D_1(x)\, g(x) + \gamma_1(x)$$
$$x^{n-k}\, d(x) = D_2(x)\, g(x) + \gamma_2(x)$$

Subtract to get:

$$[\, D_2(x) - D_1(x)\,]\, g(x) = \gamma_1(x) - \gamma_2(x)$$

We make the same argument as above: The LHS is a polynomial of degree $\geq$ n-k. The RHS is a polynomial of degree < n-k. This is a contradiction, so one cannot have two different D's which map into the same d.

Thus, the mapping between the d(x) and D(x) is a 1-to-1 mapping.

Fact 4: The mapping between c(x) and C(x) is one-to-one. Thus, one can regard either set of polynomials as a rearrangement of the other set of polynomials. (8.12)

Proof: According to Fact 3 (8.11), the mapping between the d(x) and the D(x) is one-to-one. Then, since

$$c(x) = g(x)d(x) \qquad\qquad C(x) = g(x)D(x)$$

we can make a corresponding one-to-one mapping between the c(x) and the C(x).

Corollary: The set of code words of a code in the cyclic basis is a rearrangement of the set of code words of the same code in the systematic basis. (8.13)

Comment: The Corollary is true whether or not the generator g(x) divides $x^n - 1$. Nowhere in this section have we made use of this fact. However, this fact is the basis of the next sections.

(f) Why Cyclic Codes are Cyclic

In the above discussion of cyclic codes, it was noted that the name arises because all cyclic permutations of code words are also code words. Here we will prove this is so.

If we take the coefficients of a polynomial f(x) and "rotate them m places to the right" to form a new polynomial, this new polynomial is called a **cyclic permutation** of the original polynomial by m places, and is denoted by $f^{(m)}(x)$.

Example: (a rotation of one place)

$$f(x) = f_0 + f_1x + f_2x^2 + f_3x^3 + \ldots f_{n-2}x^{n-2} + f_{n-1}x^{n-1}$$

$$f^{(1)}(x) = f_{n-1} + f_0x + f_1x^2 + f_2x^3 + f_3x^4 + \ldots + f_{n-2}x^{n-1} \qquad (8.14)$$

Math Lemma 1: Claim that: $f^{(1)}(x) = \text{Rem}[(x\, f(x))/(x^n - 1)]$. (8.15)

Proof: Let $f(x) = f_0 + f_1x + f_2x^2 + \ldots f_{n-1}x^{n-1}$. Then

$$xf(x) = f_0x + f_1x^2 + f_2x^3 + \ldots f_{n-1}x^n$$

$$= [\, f_{n-1} + f_0x + f_1x^2 + \ldots f_{n-2}x^{n-1} \,] + f_{n-1}(x^n - 1) = [\, f^{(1)}(x) \,] + f_{n-1}(x^n - 1).$$

If we were to expand xf(x) over (x^n-1) using the Division Algorithm (3.6), we would recognize the coefficient f_{n-1} as the quotient polynomial, and $[\, f^{(1)}(x) \,]$ as the remainder polynomial. **QED**

Math Lemma 2: Claim that:

$$f^{(m)}(x) = \text{Rem}[(x^m f(x))/(x^n - 1)]$$

or (8.16)

$$x^m f(x) = q(x)\,(x^n - 1) + f^{(m)}(x) \qquad \text{for some quotient polynomial } q(x)$$

Proof: Math Lemma 1 shows this is true for m=1. Then do an induction proof. Assume true for m=k and show that true for m = k+1. For m=k we have the equivalent equations,

$$f^{(k)}(x) = \text{Rem}[(x^k f(x))/(x^n - 1)] \qquad (*)$$

$$x^k f(x) = q(x)\,(x^n - 1) + f^{(k)}(x)\,. \qquad (**)$$

Lemma 1 with $f = f^{(k)}(x)$ says that

$$[f^{(k)}(x)]^{(1)} = \text{Rem}[(x\, f^{(k)}(x))/(x^n - 1)]\,.$$

But $f^{(k)}$ shifted one to the right is $f^{(k+1)}$. Then insert $f^{(k)}$ from (**) on the right to get

$$f^{(k+1)}(x) = \text{Rem}[(x\ \{\ x^k f(x) - q(x)\ (x^n - 1)\})/(x^n - 1)]$$

$$= \text{Rem}[(x\ \{\ x^k f(x)\ \})/(x^n - 1)] \quad = \text{Rem}[(x^{k+1} f(x))/(x^n - 1)]$$

which is (*) for m = k+1. **QED**

Fact 5: For a cyclic code, all cyclic permutations of any code word are also code words. Thus, the k-dimensional vector space C^k spanned by the code words is a **cyclic subspace** of V^n. (8.17)

Proof: We do this proof in the cyclic basis. It then also applies to the systematic basis, since we have just shown above in Corollary (8.13) that the set of code words is the same in either basis (they are just rearranged).

Apply Math Lemma 2 (8.16) to f(x) = c(x), a polynomial of degree $s \leq n-1$ corresponding to a legal codeword,

$$x^m c(x) = q(x)\ (x^n - 1) + c^{(m)}(x) \ .$$

Make the replacement c(x) = d(x)g(x) to get

$$c^{(m)}(x) = x^m d(x)g(x) - q(x)(x^n - 1) \ .$$

Now make the further replacement $(x^n - 1) = h(x)g(x)$ from (8.1) (b),

$$c^{(m)}(x) = [\ x^m d(x) - q(x)h(x)\ ]\ g(x) \ .$$

Since g(x) has degree n-k, and since $c^{(m)}(x)$ has degree $s \leq n-1$, the item in brackets [ ] must have degree s-(n-k). But since $s \leq n-1$ we have

$$s-(n-k) \leq (n-1) - (n-k) = k-1 \ .$$

Thus, the degree of $[\ x^m d(x) - q(x)h(x)\ ]$ is $\leq k-1$, so this is just some data polynomial we will call $d_m(x)$,

$$d_m(x) = [\ x^m d(x) - q(x)h(x)\ ] \ .$$

Thus,

$$c^{(m)}(x) = d_m(x)\ g(x).$$

So, the rotated (cyclically permuted) code word $c^{(m)}(x)$ results from encoding the data word $d_m(x)$, so $c^{(m)}(x)$ must therefore *be* a code word. **QED**

(g) A Cyclic Code as an Ideal of the Ring $A_n = R_q / (x^n - 1)$

In the Observation at the end of Chapter 3 (a) we discussed the commutative ring R of polynomials having coefficients in some arbitrary ring-with-identity $\mathcal{R}$ having operations $\oplus$ and $\otimes$. The sums and products of polynomials in such a ring R are indicated by operations + and •. Here we consider the case where $\mathcal{R}$ = GF(q), and we shall call the corresponding polynomial ring R_q. Consider then the residue class ring,

$$A_n \equiv R_q / (x^n - 1) \ . \tag{8.18}$$

Here (x^n - 1) refers to an ideal consisting of all polynomials which are multiplies of x^n-1. We studied this type of R/I structure in detail in Chapter 3 (d). The elements of A_n are the rows of a chart, and each row can be labeled as {r(x)} where r(x) is a polynomial of degree < n obtained by dividing some polynomial in R_q by x^n - 1. Thus, we can think of the elements of A_n (rows of the chart) as corresponding to the remainder polynomials of degree < n, and we know there are q^n such polynomials in R_q. Because x^n- 1 is reducible, this residue class ring is not a field, it is just a ring. We have in mind that n is the same n that appears in the (n,k) code designation.

A remainder r(x) could for example be a data polynomial d(x), or a code word polynomial c(x), or our cyclic code generator g(x). If we consider the chart row labeled by d(x) and the row labeled by g(x), the ring element product of these two rows must contain d(x) • g(x) [= d(x)g(x)]. In other words, we must have

$$\{ d(x) \bullet g(x) \} = \{ d(x) \} \bullet \{ g(x) \}$$

where the curly bracket notation {...} was discussed in (3.14). But d(x)• g(x) = c(x) so this says

$$\{ c(x) \} = \{ d(x) \} \bullet \{ g(x) \} \ . \tag{8.19}$$

If we draw a picture of the chart for A_n similar to (3.13), we get

$$\begin{array}{ll} i_1(x)=0 & i_2(x) = q(x)\bullet(x^n-1) \ \text{ for all possible } q(x) \\ r_1(x) & r_1(x) + i_2(x) = q(x)\bullet(x^n-1) + r_1(x), \text{ for all possible } q(x) \\ r_2(x) & r_2(x) + i_2(x) = q(x)\bullet(x^n-1) + r_2(x), \text{ for all possible } q(x) \\ \text{more rows like the above} & \end{array} \tag{8.20}$$

We might imagine enumerating the q^n remainders $r_i(x)$ by increasing degree in some manner, so that the first row in the chart has the remainder $r_i(x) = 0$ which corresponds to the ideal (x^n-1). Then the second chart row might have remainder x, and so on. Since c(x) = d(x)g(x), code word polynomials have degree in the range (n-k,n-1), where n-k occurs for d(x) = constant and n-1 for d(x) of maximal degree k-1. Thus, the code word polynomials are dispersed throughout the lower region of the chart, that region where remainders have degree n-k up to n-1. They are dispersed amidst many other chart rows which also correspond to remainders in this range (all corresponding to "illegal" code words). The point is that in the A_n chart sorted in this manner, the rows corresponding to code polynomials c(x) are not adjacent to each

other but are widely dispersed. Of course there is always the exceptional code word c(x) = 0 which goes in the first row of the chart.

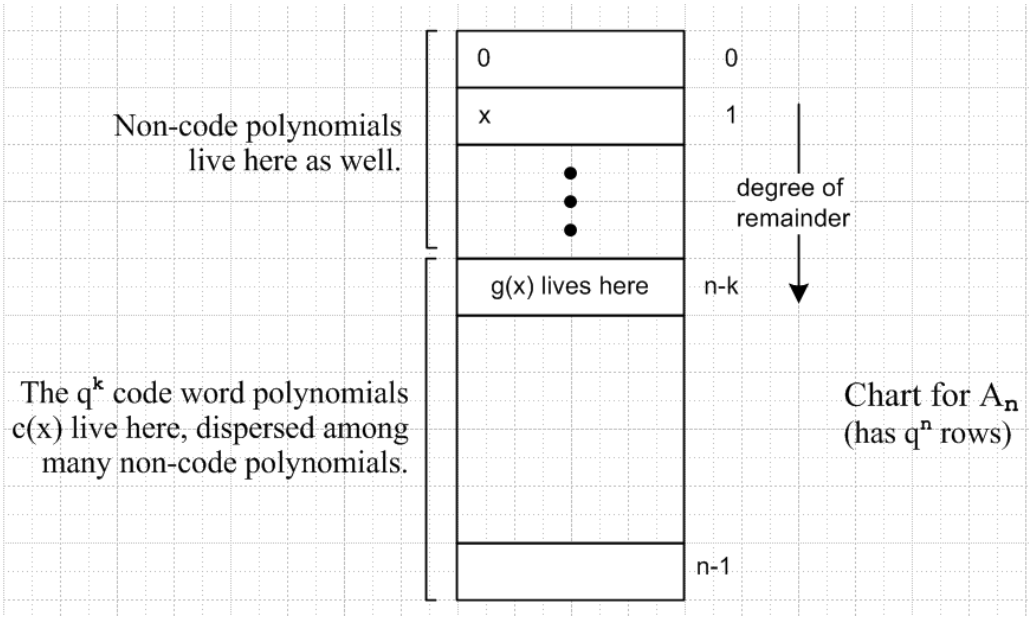


Fig 7.8

Recall the embedding vector space V^n of (7.6) which has q^n points and which contains the subspace C^k which contains the q^k code words. Each row of the chart A_n, meaning each element of A_n , corresponds to one point in V^n. The actual vector in V^n is formed by taking the m-tuple of the coefficients of a polynomial in A_n. So we can regard Figure 7.8 as a picture of V^n in which the code vectors are dispersed in line with Fig 7.3 showing the (•) in a sea of (·).

Fact 6: If g(x) divides $x^n - 1$, the chart rows {c(x)} form an *ideal* (g(x)) within A_n. (8.21)

In other words, the code word polynomials of a cyclic code comprise an ideal within the set of polynomials of degree < n. We shall denote this ideal as (g(x)), meaning rows { f(x) } of A_n where f(x) is any multiple of g(x).

Proof: An ideal I of a ring R was defined in (1.22). First of all, the set {c(x)} is an additive subgroup of A_n, as required. The set contains the additive identity 0 from c(x) = 0 and the other required properties including closure under + :

$$\{c_1(x)\} + \{c_2(x)\} = \{c_1(x) + c_2(x)\} = \{c_3(x)\} \ . \qquad \text{// closed under } A_n \text{ + operation}$$

We then have to show that r•I = I. This means that {r(x)}•{i(x)} = {i'(x)} for any {r(x)} in A_n and any {i(x)} in the ideal (g(x)). Since our candidate ideal is I = {c(x)}, the set of code word polynomials, i(x) is some code word polynomial, call it c(x). We then have to show that {r(x)}•{c(x)} = {c'(x)} where c'(x) is some other code word polynomial, where r(x) is some arbitrary element of A_n. Setting c(x) = d(x)g(x), this is what we need to show

$\{r(x)\} \bullet \{ d(x)g(x)\} = \{c'(x)\}$ where c'(x) is a code word polynomial .

Here is a proof pathway that looks promising, but does not succeed. Rewrite the above as

$\{r(x)d(x)\} \bullet \{g(x)\} = \{c'(x)\}$.

If we knew that $\{r(x)d(x)\} = \{d_1(x)\}$ for some data $d_1(x)$ (degree < k), we could write the left side as

$\{r(x)d(x)\} \bullet \{g(x)\} = \{d_1(x)\} \bullet \{g(x)\} = \{d_1(x)g(x)\} = \{c_1(x)\}$

where $c_1(x) \equiv d_1(x)g(x)$ is a code word polynomial, and then our proof that {c(x)} is an ideal would be concluded with $c' = c_1$. But we *don't* know that such a $d_1(x)$ exists with degree < k, so this approach does not work.

Here is a viable proof. We know we can write

$\{r(x)\} \bullet \{d(x)g(x)\} = \{f(x)\}$

where f(x) has degree < n and is the remainder of r(x)d(x)g(x) divided by x^n-1. But this does not tell us that f(x) is a code word. Now we add the assumption that $g(x)h(x) = (x^n-1)$. Multiply both sides of the above equation by h(x) and rearrange

$\{r(x)d(x)\} \bullet \{g(x)\, h(x)\} = \{f(x)\, h(x)\}$

or

$\{r(x)d(x)\} \bullet \{x^n-1\} = \{f(x)\, h(x)\}$

or

$\{r(x)d(x)\} \bullet \mathbf{0} = \{f(x)\, h(x)\}$

or

$\{f(x)\, h(x)\} = \mathbf{0}$.

This says that

$$\text{Rem}\left[\frac{f(x)\, h(x)}{x^n-1}\right] = 0$$

or

$$f(x)h(x) = q(x)(x^n-1) .$$
$$<n \quad k \qquad <k \quad n$$

The degree of quotient q(x) must be < k in order to balance the powers on the two sides of this last equation. Therefore q(x) can be regarded as a data polynomial. Rewrite the above as

$f(x)h(x) = q(x)g(x)h(x)$

or

$f(x) = q(x)g(x).$

Since q(x) is a data polynomial, f(x) must be a code polynomial. **QED**

Fact 5 Revisited: If g(x) divides $x^n - 1$, then any cyclic permutation of a code word of the cyclic code generated by g(x) is also a code word. (8.22)

Proof: Here we are giving an "alternate proof" of Fact 5 (8.17). Actually, it is the same proof as above, only here it is expressed in the language of rings and ideals.

A code word c(x) is a polynomial of degree < n. Thus, according to Math Lemma 2 (8.16),

$$c^{(m)}(x) = \text{Rem}[\, (x^m c(x))/ (x^n - 1)] \,,$$

where $c^{(m)}(x)$ is a cyclic permutation of c(x) by m places. Since c(x) is of degree < n, we know that $\{ c(x) \} \in A_n$. If m < n, then $\{ x^m \} \in A_n$. Thus, we can rewrite the above equation as

$$\{ c^{(m)}(x) \} = \{ x^m \} \bullet \{ c(x) \} \,. \qquad \text{(has } q^n \text{ rows)}$$

Because {c(x)} is an element of the ideal (g(x)), and because $\{ x^m \} \in A_n$, it follows from the definition of an ideal that $\{ c^{(m)}(x) \}$ is also in the ideal (g(x)). Thus, $\{ c^{(m)}(x) \}$ must be some code word. **QED**

So, we have an interesting new way to view a cyclic code:

Fact 7 : A cyclic code with symbols in GF(q), and which is generated by some g(x) which divides x^n -1, forms an ideal (g(x)) of the ring $A_n = R_q / (x^n - 1)$. This ideal consists of the set of q^k code words generated by multiplying the q^k data polynomials d(x) by g(x), $\{ c(x) \} = \{ d(x) \} \bullet \{ g(x) \}$. The full A_n chart has q^n rows since there are q^n remainder polynomials of degree < n.

(8.23)

(h) The Standard Array and Cyclic Code Error Correction

We now take the q^n rows of the A_n chart shown in Fig 7.8 above and sort them into bins. To start off, we grab all q^k code word polynomials and put them into one bin. We realize that this bin must be all the polynomials in A_n which have remainder 0 when divided by g(x), since c(x) = d(x)g(x). We just showed in Fact 6 (8.21) that this bin is an ideal of A_n. We shall regard this bin as the first row of a *new* chart for the residue class ring $A_n/(g(x))$ which is built on top of our previous residue class ring $A_n = R_q/(x^n - 1)$. The other, non-ideal bins correspond to non-zero remainders of the A_n remainders upon division by g(x). Since g(x) has q^{n-k} remainders, there must then be q^{n-k} of these bins. The number of bins times the population of each bin gives $q^{n-k}\, q^k = q^n$, which is the total number of A_n rows. This is consistent with the fact that the order of A_n must be integrally divisible by the order of any ideal within A_n, see (1.27). This set of bins forms our new residue class ring of A_n with respect to (g(x)), and this set of bins has a special name: the **standard array** :

$$\text{Standard Array} = A_n / (g(x)) = [\, R_q / (x^n - 1)] \;/\; (g(x)) \,, \qquad (8.24)$$

which has this row structure,

bin leader	contents of the bin
$r_0(x)=0$	$c_i(x) = d_i(x) \bullet g(x)$ for all possible $d_i(x)$
$r_1(x)$	$r_1(x) + c_i(x) = d_i(x) \bullet g(x) + r_1(x)$, for all possible $d_i(x)$
$r_2(x)$	$r_2(x) + c_i(x) = d_i(x) \bullet g(x) + r_2(x)$, for all possible $d_i(x)$
...	
$r_j(x)$	$r_j(x) + c_i(x) = d_i(x) \bullet g(x) + r_j(x)$, for all possible $d_i(x)$
...	

(8.25)

In this next drawing, each horizontal row is one of the bins of A_n / (g(x)) :

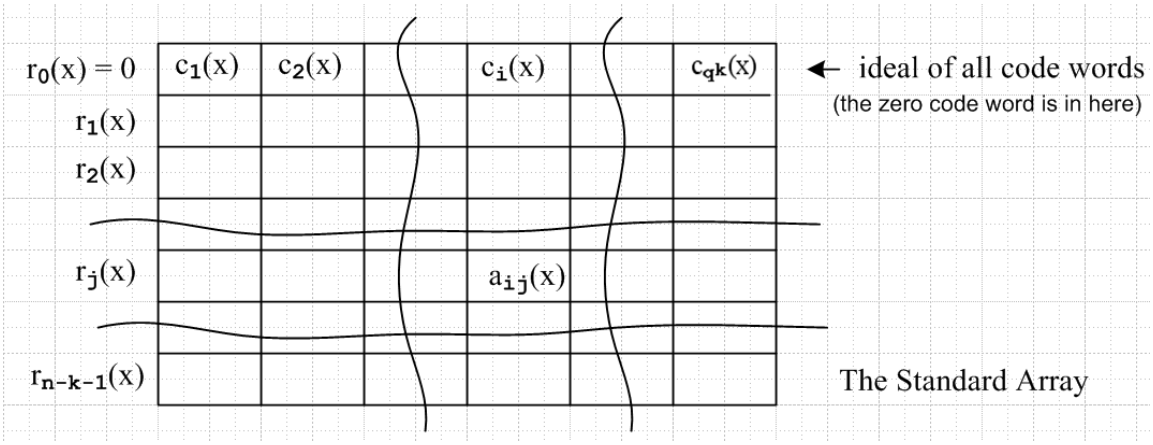


Fig 7.9

The standard array is a particular rearrangement of the q^n elements of A^n , so it is a rearrangement of the points in the vector space V^n. In this new arrangement, all the "legal" code words appear in the first row of Fig 7.9. Thus, the entries in all other rows must be "illegal" code words.

Now comes the Big Payoff of all this work involving A_n and the standard array.

Recall that in a data transmission, a transmitted code word c(x) might get damaged and might then be received as an illegal code word. The receiving decoder has full knowledge of the standard array shown above. Since the array includes all points in V^n, it includes all possible damaged code words. Suppose then a received and damaged code word c'(x) is found to match $a_{ij}(x)$ in the standard array. The decoder knows from the structure of the array that

$$a_{ij}(x) = c_i(x) + r_j(x) \; . \tag{8.26}$$

Therefore, the decoder *knows* that the code word which was actually transmitted was $c_i(x)$!!! Thus, the decoder knows how to do "error correction" on this received and damaged code word. (But see caveat coming below.)

The algorithm then for error correction is as follows. The decoder examines an incoming and possibly damaged code word and it determines in which row of the standard array that word lies. It can do this

since it knows all about the standard array. It then obtains the $r_j(x)$ for that row and subtracts it from the word that came in,

$$c_i(x) = a_{ij}(x) - r_j(x) \tag{8.27}$$

which, when written as V^n vectors (coefficient m-tuples), says

$$\mathbf{c}_i = \mathbf{a}_{ij} - r_j. \tag{8.28}$$

We may compare this with our earlier equation,

$$\mathbf{c} = \mathbf{c}' - \mathbf{e} \quad \text{where} \quad \mathbf{s} = \mathbf{e}\,H^T = \mathbf{c}'H^T \tag{7.41}$$

Thus, the *correcting vector* is $-\mathbf{e} = -\mathbf{r}_j$. Notice that all bad code words in the row with $\mathbf{a}_{ij}$ have the *same* correcting vector $-\mathbf{r}_j$ due to the fact that the standard array is a residue class ring. Thus, all bad code words in that row will generate the same syndrome $\mathbf{s}_j = \mathbf{r}_j H^T$. As shown in the example (7.22), there are q^{n-k} possible syndromes $\mathbf{s}_j$, and this corresponds to the number of remainders $\mathbf{r}_j$ so there is a 1-to-1 relation between the $\mathbf{r}_j$ and the $\mathbf{s}_j$. In its initialization sequence (or perhaps this is precomputed into a ROM memory), the decoder can build a lookup table as follows. For each possible $\mathbf{r}_j$ it computes the corresponding syndrome $\mathbf{s}_j = \mathbf{r}_j H^T$. The lookup table then provides $\mathbf{r} = F(\mathbf{s})$. Then in operation, when an incoming word $\mathbf{c}'$ arrives, the decoder computes $\mathbf{s} = \mathbf{c}'H^T$, looks up the proper $\mathbf{r}$ in this little table, and does the addition $\mathbf{c} = \mathbf{c}' - \mathbf{r}$. That's it! If the incoming word had no error, the syndrome will be $\mathbf{s} = 0$, and the lookup table puts out $\mathbf{r} = 0$, and this is benignly added to the incoming word.

It almost sounds too good to be true. One gets the impression that the standard array can correct any number of symbol errors in an incoming code word, but this is a false impression. The technique can only correct errors for which the error pattern is one of the r_i class leaders in the standard array! Since the code words have n symbols, there are $q^n - q^k$ possible bad code words (all rows of the standard array below the first row), but there are only q^{n-k} correction vectors r_i used as residue class leaders. In general $q^{n-k} < q^n - q^k$ so there are not enough r_i to correct all possible errors. So in Fig 7.9, the received word $a_{ij}(x)$ *might* have come from an alteration of some other codeword $c_r(x) \neq c_i(x)$ due to an error pattern which does not appear among the listed r_i.

Recall that in the construction of a coset decomposition or a residue class ring, there is always freedom in choosing the row "leaders". The leaders, which in our case are the $\mathbf{r}_i$, can be taken as any element of the row that the leader leads. One wants then to select the leaders to represent error patterns that are most likely to occur, and that means patterns with the largest possible number of zeros, or minimum weight. There are nq patterns with all zeros except a single symbol (weight 1) and such patterns would represent single-symbol errors which are the most probable errors to occur. So the best use of the standard array is to select the leaders r_j to be patterns representing r-symbol errors for small r. Recall from Chapter 7 (f) that an r+1 symbol error might be 10^{-6} times less likely than an r-symbol error. When an error pattern does occur which is not among the r_j, the correction process of course goes ahead, but the corrected code word is wrong.

This concludes our minimal discussion of "error correction" and the reader is directed to the references for more information.

(i) Galois-Induced Cyclic Codes and the Parity Check Matrix H

So far in this Chapter we have already made a modest connection between cyclic codes and Galois Fields GF(q). We have suggested that code symbols -- the coefficients of the polynomials c(x), d(x), and g(x) -- can be thought of as elements of GF(q).

Now we are going to make a much stronger connection between such cyclic codes and the Galois Fields GF(q). This connection is motivated by the following two observations:

(1) According to the definition given in (8.1), a cyclic code (n,k) is defined by any polynomial g(x) (degree n-k) which divides evenly into $x^n - 1$.

(2) According to Big Theorem 2 (4.34) item 5, the polynomial $x^{q-1} - 1$ can be fully factored into linear factors each of which contains an element of Galois Field GF(q). We write this as:

$$(x^{q-1} - 1) = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_{q-1}) \quad . \tag{8.29}$$

These two observations scream out the following suggestion:

Suggestion: Why not take g(x) to be any subset of n-k of these factors in (8.29). Such a g(x) will then generate a cyclic code (n,k) = (q-1,k) = (p^m - 1,k). We shall call this a Galois-induced cyclic code.

Let g(x) then have the following form, where the α_i are taken from the set $\{a_r\}$ shown above,

$$g(x) = (x - \alpha_1)\bullet(x - \alpha_2)\bullet(x - \alpha_3) ..... (x - \alpha_{n-k}) \quad .$$

What can we say about this particular g(x)? (8.30)

(1) it has n-k *roots* in GF(q), that is, $g(\alpha_1) = g(\alpha_2) = g(\alpha_3) ... = 0$
(2) it has degree n-k
(3) the coefficients of g(x) lie in GF(q)

Are there any cyclic codes with polynomial coefficients in GF(q) which are *not* Galois-induced cyclic codes? Such a code would involve a g(x) which divides x^n-1 which g(x) was *not* formed from a subset of the factors in (8.29). But we know that x^n-1 fully factors within GF(q=n+1) as shown in (8.29) (GF(q) is a "splitting field" for x^n - 1), so the only g(x) with coefficients in GF(q) which can divide x^n- 1 *must* be formed from a subset of the (8.29) factors. In other words, if we have

$$x^n\text{-}1 = h(x) \bullet g(x)$$

and

$$x^n\text{-}1 = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_n)$$

then

$$h(x) \bullet g(x) = (x - a_1)\bullet(x - a_2)\bullet(x - a_3)\bullet(x - a_4)\bullet.....(x - a_n) \ .$$

Since h(x) and g(x) are polynomials in x, it must be that both h(x) and g(x) are formed from sets of (x-a) factors which partition the set of all (x-a) factors. Thus, the answer to the question is no, and we conclude that all cyclic codes having coefficients in GF(q) are in fact Galois-induced cyclic codes.

According to the definition of a cyclic code, the code word polynomials will be formed as

$$c_i(x) = d_i(x)g(x) \quad i = 1,2,3.... \; q^k \; .$$

It therefore follows that all the code word polynomials of such a cyclic code will have at least the same GF(q) roots α_i that g(x) has. It also follows that the code symbols, being coefficients of $c_i(x)$, will lie in GF(q). If we happen to want these coefficients to lie in the ground field GF(p), we would have to take combinations of factors $(x - \alpha_i)$ that form the "minimum polynomials" defined in (5.3), and this will be done soon.

Fact 8: The viable parity check matrix H for such a cyclic code has the following form:

$$H = \begin{vmatrix} 1 & (\alpha_1) & (\alpha_1)^2 & (\alpha_1)^3 & ... & (\alpha_1)^{n-1} \\ 1 & (\alpha_2) & (\alpha_2)^2 & (\alpha_2)^3 & ... & (\alpha_2)^{n-1} \\ 1 & (\alpha_3) & (\alpha_3)^2 & (\alpha_3)^3 & ... & (\alpha_3)^{n-1} \\ ... & ... & ... & ... & ... & ... \\ 1 & (\alpha_{n-k}) & (\alpha_{n-k})^2 & (\alpha_{n-k})^2 & ... & (\alpha_{n-k})^{n-1} \end{vmatrix} \qquad (8.31)$$

Proof: First of all, this H has the right number of rows (n-k) and columns (n). Recall from (7.21) that H is a matrix which kills all code vectors, $H\mathbf{c}^T = \mathbf{0}$. Think of the components of $\mathbf{c}^T$ as the coefficients of the polynomial c(x). If we multiply the i^{th} row of the above H matrix by column vector $\mathbf{c}^T$, we get:

$$c_0 + c_1 \; (\alpha_i) + c_2 \; (\alpha_i)^2 + c_3 \; (\alpha_i)^3 + ... \; c_{n-1} \; (\alpha_i)^{n-1} \; . \qquad (8.32)$$

But this is exactly $c(\alpha_i)$ which we know is 0 since α_i is a root of g(x) and c(x) = d(x)g(x). Thus, the above matrix H has the right property that $H\mathbf{c}^T = \mathbf{0}$ for any codeword **c**. The elements of H are symbols, elements of GF(q). And H is the generator of the dual code (n,n-k) as in (7.24).

There are $\binom{q-1}{n-k}$ ways to construct a g(x) of degree n-k from the factors of (8.29). For $q = 2^m = 2^8 = 256$ this represents a *very* large number of possible codes with relatively small n-k. For example, for codes with n = q-1 = 255 and with n-k = 10 parity check bytes there are $\binom{255}{10}$ = 267,934,565,633,045,025 Galois induced codes. It turns out that some of these codes are much more interesting than others because they have better "performance". In particular, as we shall see in the next section, there is a theorem which lets us pick out the codes which have the best random error correction capability. These codes form the so-called BCH code family, which happens to contain the narrow-sense BCH codes as well as the Reed-Solomon codes. A subset of the narrow-sense BCH codes consists of the Hamming Codes. Here is the general relationship:

BCH codes			(8.33)
The narrow-sense BCH Codes	$t \geq 0$	GF(q)	
The Hamming Codes	$t = 0,1$	GF(q)	
The Reed-Solomon codes	$t \geq 0$	GF(q)	

The t column indicates the amount of random error correction possible, and the last column shows the space of the code symbols. Most useful cases have $q = p^m$ with p=2, since that is what digital computers deal with most easily (a flip-flop can be 0 or 1).

(j) Motivation for g(x) to have coefficients in GF(p)

In our Chapter 5 discussion of minimum polynomials the reader may have wondered why it is useful for the coefficients to lie in GF(p) instead of in GF($q=p^m$). In Chapter 9 we shall see codes in which g(x) is formed from certain products of minimum polynomials, and therefore the g(x) so formed have coefficients in GF(p). Although Galois theory is always done with a general prime number p as the characteristic, engineers always have in mind the world of binary digital logic where p = 2. In digital logic, a physical wire is either high or low. In early times, a wire was "high" if it was 5 volts, and was "low" if it was 0 volts. The number 5 continues to decrease with the progression of Moore's Law and die shrinkage. Regardless, a traditional digital wire carries an element of GF(2). [It *could* carry an element of GF(p) with p discrete voltage levels, but we assume p = 2.]

In this section, we want to show why it is useful in practice to have the coefficients of g(x) lie in GF(2). We think here in terms of hardware implementations. By "hardware" we mean an implementation that involves a set of elements which run separate "programs" concurrently.

Let's start with the regular **block code** situation of (7.3) and (7.4) where we had

$$\mathbf{c} = \mathbf{d}G: \qquad (c_1\ c_2\ c_3) = (d_1\ d_2)\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}. \tag{8.34}$$

The elements of vectors **c** and **d** as well as the matrix elements of G were all assumed to be in GF(q). Consider the computation of c_2,

$$c_2 = b \bullet d_1 + e \bullet d_2. \tag{8.35}$$

An encoder has to actually *compute* things like $b \bullet d_1$ using the GF(q) multiplication table. In a hardware implementation, this is a fairly expensive proposition. One needs a black box with two inputs and one output which "looks up" the product of two elements of GF(q). In the real world of digital hardware, we are going to have $q = p^m = 2^m$ for some power m. Let's just take m = 8 so the symbols of GF(2^8) are then normal computer bytes of 8 bits each. A typical m-tuple is then [11001010]. In this case, the computation of $b \bullet d_1$ could be implemented in a read-only memory (ROM) with two 8-bit inputs going to a total of 16 address lines, and 8 data outputs. In other words, this is a 2^{16} x 8 = 64K x 8 ROM. This method would be "fast" because the result could be developed in a single "clock". A more practical alternative would be to have the black box compute the GF(2^8) product in some number of clocks using less hardware than the

ROM requires. If one has a lot of a • b activity going on, one might need many of these black boxes to do things fast, or share black boxes and have things be slower.

But *suppose* we could arrange for the elements of the G matrix to lie in GF(p) = GF(2), the only elements of which are 0 and 1. Then look at $b \bullet d_1$. If b = 0, the product is 0, and if b = 1, the product is d_1. The need to actually *do* multiplication goes away. For each term in (8.35), we either add in one copy of the byte or we don't. Thus, a computation like (8.35) is very easy to do in hardware. Suppose the computation (8.35) had k terms,

result $= \Sigma_{i=1}^{k} C_i d_i$.

Here is a simple hardware circuit which computes this result in k clocks :

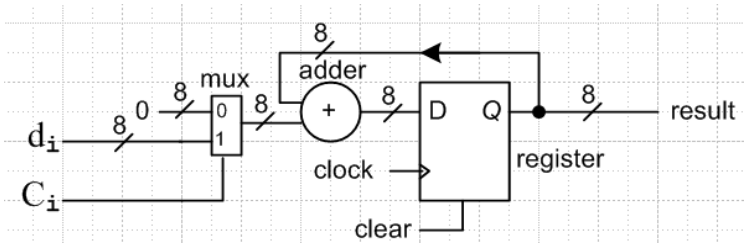


Fig 8.1

A line is a wire carrying a 1 or 0 (high or low), but a line marked by 8 is a bus of 8 wires. Such a bus carries an element of GF(q=2^8) -- a byte. The register is first cleared. Then on each clock, a new d_i and C_i are presented on the left. The multiplexor, based on C_i, selects either the d_i input byte or a 0 byte to be added to the current contents of the register which here is being use as an "accumulator". The result is ready in k clocks. Very cheap and very fast.

Moreover, the "adder" is a GF(2^8) adder, not a Z_{256} adder normally used to add integers. Looking back at the m-tuple notation of Chapter 4 (b) one recalls from (4.8) that the individual "digits" of an m-tuple representing an element of GF(q) are added independently in Z_p = GF(p), and here those digits are bits since p=2. There is no "carry" between digits. Thus, the adder shown above is just a set of eight cheap XOR gates, each having 2 bits in and 1 bit out. This XOR gate implements the + function of GF(2), and in particular, 1+1 = 0. To distinguish GF(2) addition from normal integer addition one can use that $\oplus$ symbol, so then $1 \oplus 1 = 0$.

In the world of **cyclic codes**, a similar simplification is obtained if the coefficients of g(x) lie in GF(2) and not in GF(2^8). In the cyclic-basis product c(x) = d(x) g(x) we found the need to do the following computation in (8.2),

$$c_s = \sum_{j = \max(0,s-k+1)}^{\min(s,\, n-k)} d_{s-j}\, g_j \quad \text{for } s = 0,1 \ldots n-1 \qquad c(x) = \Sigma_{s=0}^{n-1} c_s\, x^s \qquad (8.2)$$

In this sum, if the g_j are elements of GF(2), then once again each term in this sum is a situation of either you add d_{i-j} or you don't, there is no need for actual multiplication. A circuit similar to that shown above could compute the c_s coefficients. In the systematic basis with C(x) = D(x) g(x), things are more difficult. In order to form the code word C(x) = γ(x) + x^{n-k} d(x) in which the data coefficients are "visible", as in (8.5) and (8.6), one must compute the parity check polynomial γ(x) according to (8.3)

- $\gamma(x) = \text{Rem}[x^{n-k}\, d(x)/\, g(x)\,]$

The product $x^{n-k}\, d(x)$ is obtained by effectively advancing the signal d(x) by n-k clocks which is easily done with a front-end delay line or other means, but then we are faced with polynomial division by g(x). Similarly, from (8.7) the decoder must compute the syndrome polynomial s(x) as

$s(x) = \text{Rem}[\, C'(x)/g(x)\,]$

Both these remainder calculations can be done by a hardware polynomial divider of the type shown below, and again the need to do multiplications in GF(q) is avoided when g(x) has coefficients in GF(2).

Multiplying or dividing a polynomial by a fixed polynomial

In the grade-school method of multiplying two numbers, one does "shift, multiply and add" several times to work out the product. When the multiplier's symbols are in GF(2), the "multiply" part is "either take it or not" at each stage of the multiplication process. Long division as in the example of (3.7) is a process of "multiply, subtract and shift" where again the "multiply" part is "either take it or not" if the divisor has coefficients in GF(2). And subtract is the same as add for GF(2) bits. The upshot is that a "polynomial divider" in hardware is really no more complicated than a "polynomial multiplier", and both are quite simple, cheap and fast.

Without a detailed explanation, just for the reader's possible interest, here are logic circuits for a polynomial multiplier and a polynomial divider. In our application the h_i would be our GF(2) g_i coefficients. Note that these coefficients are "fixed" by the code and don't change. They might be loaded into the circuit at some initialization time. In both pictures, all the wires are m bit buses which carry GF($q=2^m$) symbols. The buses with a cross indicate that the symbol at the arrow tail is multiplied by some h_i and the product is the output of the arrow. As just noted, since the each $h_i = 0$ or 1, the adders are just "take it or not" circuits. These drawings are taken from the author's "scrambler" document, to be put online sometime soon. Chapter 7 of Peterson and Weldon is devoted to this general topic.

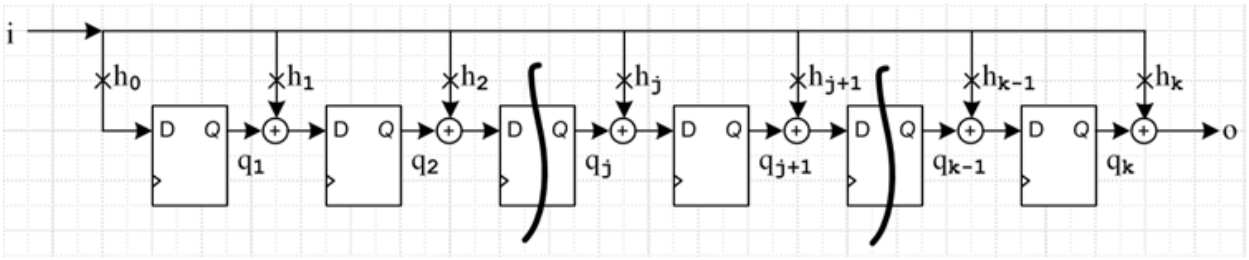


Figure 1.10: Type B polynomial multiplier.

Fig 8.2

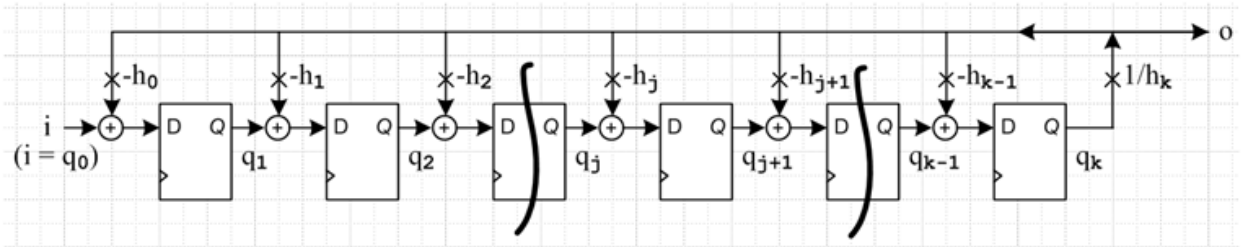


Figure 1.5. Type B polynomial divider.

Fig 8.3

These drawings show one of two standard configurations for such circuits. The Type B divider has the desirable feature that when a division has completed, the division remainder is left in the registers.

Having said all this, we shall learn in Chapter 9 that the Reed-Solomon codes have g(x) coefficients which do *not* lie in GF(2) but instead lie in GF(2^m) along with all the other coefficients. This willingness to give up "simple hardware" is due to the strong error correcting capability of such codes. The above two circuits could still be used, but the multiplication crosses are then full GF(2^8) multipliers. For each, one could in principle use a ROM as noted above, or some kind of combinatoric shifting circuit. Such a multiplier circuit in effect has to compute c(x) = Rem[(a(x)•b(x))/f(x)] as in (3.14) (h), and this means the circuit has to know about f(x), usually referred to as "the irreducible polynomial". The general subject of efficient hardware designs for Galois Field multipliers has been quite active in the last 15 years, see for example Kitsos, Ahlquist, Savas, and Xilinx in References. One focus is on the area of cryptography.

(k) The Code Word Exhaustion-by-Rotation Theorem

We first set the context for this theorem. In the construction GF(q=p^k) = R/(h(x)) according to (3.17), let h(x) some minimum polynomial of degree k. We know from the definition of a minimum polynomial that such an h(x) factors into a product of k ($x-a_i$) factors which form a subset of the factors of (x^n - 1) as in (4.25), where n = q-1 = p^k- 1 and where the a_i lie in GF(q) but the coefficients of h(x) lie in GF(p). Therefore we know that:

(1) such an h(x) divides evenly into (x^n - 1) to produce some polynomial g(x) = (x^n - 1)/h(x) of degree n-k which has coefficients in GF(p).

(2) This g(x) generates a cyclic (n,k) code in which the data words have k coefficients and code words have n = p^k- 1 coefficients.

Since there are p^k-1 non-zero data words, there are also p^k-1 non-zero code words because c(x) = d(x)g(x). Since a code word **c** has n coefficients, and since any rotation of a code word is a code word, any given code word yields a total set of n code words doing all possible rotations. If it happened that each non-trivial rotation of some code word **c** resulted in a *new* code word (with none repeated), then we could enumerate all p^k-1 non-zero code words of the code simply by rotating any one code word **c** all n = p^k-1 ways.

We will show that in the special case that h(x) is a *primitive* polynomial, each non-trivial rotation of any non-zero code word **c** produces a *different* code word. This then is the claim of this following Theorem.

Exhaustion-by-Rotation Theorem. In the context of the above paragraph, if h(x) is a primitive polynomial, then: (1) each non-trivial rotation of a non-zero code word yields a new and different code word and thus (2) the code words obtained by rotating any given non-zero code word exhaust the set of non-zero code words of the code associated with h(x). (8.36)

Proof: Let c(x) be some code word polynomial, and let $c^{(m)}(x)$ be the code word polynomial obtained by rotating the coefficients of c(x) m symbols to the right. It is understood that in such a rotation or "cyclic permutation", symbols wrap around as shown in (8.14). By a non-trivial rotation we mean that $0 < m < n$. From Math Lemma 2 of (8.16) we know that

$$c^{(m)}(x) = \text{Rem}[(x^m\, c(x))/(x^n - 1)]$$

or *(8.16)*

$$x^m\, c(x) = q(x)\,(x^n - 1) + c^{(m)}(x) \qquad \text{for some quotient polynomial } q(x)$$

$m \quad n-1 \quad m-1 \quad n \qquad\qquad < n$

where we have indicated the degree of the various polynomial players. So assume for the moment that the m^{th} rotated codeword polynomial $c^{(m)}(x)$ is identical to the original code word polynomial c(x),

$$c^{(m)}(x) = c(x) \qquad \text{for some } m \text{ in range } (1,n-1)\ .$$

Then (8.16) just above becomes

$$x^m\, c(x) = q(x)\,(x^n - 1) + c(x) \qquad \text{for some quotient polynomial } q(x)$$

or

$$c(x)\,(x^m - 1) = q(x)\,(x^n - 1) \qquad 1 \le m \le n-1\ . \qquad (8.37)$$

$n-1 \quad m \qquad m-1 \quad n$

Installing into this $c(x) = d(x)g(x)$ and $(x^n - 1) = h(x)g(x)$ gives

$$d(x)g(x)\,(x^m - 1) = q(x)\,h(x)g(x)$$

or

$$d(x)\,(x^m - 1) = q(x)\,h(x)\ . \qquad (8.38)$$

$k-1 \quad m \qquad m-1 \quad k$

Now using the rules shown in (3.14), we put our special Galois brackets {...} around the above equation and drill them inward to get

$$d(\{x\})\,(\{x\}^m - 1) = q(\{x\})\,\{h(x)\} \qquad (8.39)$$

where recall that {x} represents some element (call it α) of GF(q). Writing {x} = α and recalling that in our remainder construction we must have {h(x)} = 0 (this is the remainder of dividing h(x) by h(x)), we have

$$d(\alpha)\,(\alpha^m - 1) = 0 \qquad (8.40)$$

where the 0 and 1 are the 0 and 1 elements of GF(q). Since we assume that d is some non-zero data word, we know that $d(\alpha) \neq 0$ and we conclude that

$$\alpha^{m} = 1 \qquad \text{where } \alpha = \{x\} \quad \text{where } 1 \leq m \leq n\text{-}1 \ . \tag{8.41}$$

Since $\{h(x)\} = 0$, we know also that $h(\{x\}) = 0$ and thus $h(\alpha) = 0$. We know that the k a_i for which $h(x) = \Pi (x\text{-}a_i)$ [and for which therefore $h(a_i) = 0$] form the conjugate set of any of these a_i.

Now we said that h(x) was a *primitive* polynomial for GF(q). As noted in (5.35), all elements of the conjugate set of a primitive polynomial are primitive elements of GF(q). Since $h(\alpha) = 0$, we know that our $\alpha = \{x\}$ is one of these primitive elements, call it a_j. Thus, we may regard h(x) as a primitive polynomial for element $\alpha = \{x\}$ of GF(q). We know from (4.31) and elsewhere that if $\alpha^{m} = 1$ for a primitive element α of GF(q) then m must be a multiple of q-1, so m = N(q-1). But q-1 is the symbol n of the discussion above, so that we can only have m = Nn for some integer N. But this then rules out all values of m in the range $1 \leq m \leq n\text{-}1$ shown in (8.41) above, so we arrive at a contradiction. Therefore our supposition that it was possible to have some $c^{(m)}(x) = c(x)$ for primitive h(x) was wrong, so we must then have $c^{(m)}(x) \neq c(x)$ if h(x) is a primitive polynomial.

Now consider two rotations of c(x), call them $c^{(m1)}(x)$ and $c^{(m2)}(x)$ and assume $m_2 > m_1$. Define

$$C(x) \equiv c^{(m1)}(x) \ .$$

Then

$$C^{(m2-m1)}(x) = c^{(m2)}(x) \ .$$

In this last line we rotate C(x) m_1 symbols to the left (undoing $c^{(m1)}(x)$ to c(x)) and then we rotate it m_2 symbols to the right to get $c^{(m2)}(x)$. If we now apply our proven Theorem above to C(x) instead of c(x), we conclude that we must have $C^{(m2-m1)}(x) \neq C(x)$ which then says $c^{(m2)}(x) \neq c^{(m1)}(x)$. Thus we know that each non-trivial rotation of c(x) gives a *new* code word polynomial! Then, as noted in the opening paragraph, since all the rotations of c(x) are different, they exhaust the entire set of code word polynomials and thus their coefficients exhaust all the code words **c** of our cyclic code.

Chapter 9: A Small Survey of a few Standard Code Families

(a) The BCH Codes

BCH are the initials of Bose and Ray-Chaudhuri (1960) and independently Hocquenghem (1959), see References.

The philosophy here is a little different than that of the last two Chapters. There we selected arbitrary values n and k for our (n,k) code and then constructed a code. For a linear block code, that meant selecting a kxn matrix G which had k linearly independent rows -- rank k. For a cyclic code as defined in (8.1) we had to select an h(x) of degree k and g(x) of degree n-k such that their product was x^n-1. In contrast, in the BCH discussion below, we specify certain desirable properties of the code (error correcting ability) and then we try to find values for n and k for which such desirable properties are realized. We are forced to a certain set of values for n (usually $n = p^m - 1$) and for each n we are forced to some set of values for k. We end up then with a family of (n,k) BCH codes.

Let us return to the Suggestion of Chapter 8 (i). We select g(x) of the form (8.30) so that g(x) has the set of roots α_1 , α_2, ... α_{n-k} . According to Big Theorem 1 (4.30) we know that GF(q) is cyclic under •, and we can therefore represent all its non-zero elements as powers α^k of a primitive element α. Thus, we can rewrite the (8.30) list of roots as:

$$\{ \alpha_1, \alpha_2, \alpha_3, ... \alpha_{n-k} \} = \{ \alpha^{e_1} , \alpha^{e_2} , \alpha^{e_3} , ... \alpha^{e_{n-k}} \} \qquad (9.1)$$

where the e_i are whatever exponents produce α_1 , α_2, ... α_{n-k}.

We now wish to quote a very strange theorem about the minimum distance, hence error correcting ability, of this Galois-induced cyclic code. You will have to read this very carefully.

The BCH Bound

Context: Any element α of GF(q) has *some* specific order n, which is the minimum power for which $\alpha^n = 1$. Recall that such an α carves out a cyclic subgroup under • within GF(q) of order n. We first pick an α, and then use its order n to be the length of our cyclic (n,k) code.

BCH Bound Theorem. Given some α in GF(q) of order n, the minimum distance d of the cyclic code (n,k) whose generator g(x) has some set of n-k roots in GF(q), namely $\{ \alpha^{e_1}, \alpha^{e_2}, \alpha^{e_3}, ... \alpha^{e_{n-k}} \}$, is *greater than* the largest number of *consecutive* integers in the power list: $\{ e_1, e_2, e_3, ...e_{n-k}\}$. (9.2)

This theorem provides a *lower* bound on the minimum distance d of the code, implying a lower limit on t, the number of errors per codeword that can be corrected. Recall from (7.33) that d ~ 2t+1. This suggests that we might be able to find some codes that will correct large numbers of errors in a code word.

Proof: Recall Fact 4 of (7.35) that the minimum distance d of a code is equal to the minimum number of columns of the matrix H which are linearly dependent. One proof of the BCH bound theorem involves examining the H matrix shown above in (8.31), and showing that d = (the minimum number of linearly

dependent columns) exceeds the strange count noted in the theorem. This proof is presented in Peterson and Weldon Section 9.1 p 269 and we won't repeat it here since it is a bit involved.

From this bound, we are immediately motivated to select powers that are in sequence, rather than some set of random powers of α, due to the word "consecutive" appearing in the theorem. Also, we might like to have n = the order of α be as large as possible, since this increases the size of the pool of roots from which we may select the roots of g(x), and this would increase the chances of having a longer consecutive power sequence. For any GF(q), if we pick α = a primitive element, its order will be n = q-1 which is the most it can be. Of course in this case the code length n must be set to $n = q-1 = p^m - 1$, which is a restriction on the possible values of the code length n. For the binary p=2, one could have n = 3, 7, 15, 31, 63, 127, 255 and so on.

Question: Why can't one just pick the roots $\{ \alpha^{e_1}, \alpha^{e_2}, \alpha^{e_3}, ... \alpha^{e_{n-k}} \}$ to be $\{ \alpha, \alpha^2, \alpha^3 .... \alpha^{n-k}\}$ for some general n and k, and then the theorem says d > n-k which we can write as d ≥ n-k+1. This would be an interesting situation since we know from (7.34) that d ≤ n-k+1, and therefore we would have to have d = n-k+1, which would be the best d possible. The reason one can't in general do this is that, for n < q-1, the elements of $\{ \alpha, \alpha^2, \alpha^3 .... \alpha^{n-k}\}$, although roots of $x^{q-1}-1$ according to (4.34), probably won't all be roots of $x^n - 1$ and therefore can't be used as the roots of g(x) as required by (8.1) $g(x)h(x) = x^n-1$. But if α is taken as a primitive element, then its order is n = q-1 and *all* powers α^s are roots of x^n-1 and then we *can* select the sequence $\{ \alpha, \alpha^2, \alpha^3 .... \alpha^{n-k}\}$ and this *does* give d = n-k+1. The only problem here is that $g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)....(x-\alpha^{n-k})$ likely won't have coefficients in GF(p). For example, this is not the obvious form of a minimum polynomial (5.14) which does have coefficients in GF(p). In Chapter 8 (j) we explained *why* we want the coefficients of g(x) to be in GF(p): hardware simplification. We have said earlier that k is not yet determined. As we reduce the number k for a given n, the number of roots of g(x) increases. As we add more (x-a) factors to g(x) in this way, maybe at some point g(x) will become a minimal polynomial or a product of minimum polynomials, and then we know g(x) will have GF(p) coefficients. This is precisely the plan that will be pursued below, and this then explains why the value of k gets "forced" on the code designer. In our final section on Reed-Solomon codes, however, we will go ahead and allow g(x) to have coefficients in GF(q), despite the hardware cost, and this will result in the best distance d possible, as suggested above.

BCH Bound Corollary : If we can select the roots of g(x) to include $\{ \alpha^a, \alpha^{a+1}, \alpha^{a+2}, ... \alpha^{a+r}\}$ for some a and r, then, since this list has r+1 consecutive powers, the minimum code distance will be d > r+1. If we take α as a primitive element, then the order of α is n = q-1, and then all terms shown in the list will be roots of x^n-1 and therefore allowable roots for g(x). Such codes are generally called **BCH codes.** (9.3)

At this point then we shall select n = q-1, but we have to hold off on finding a value of k for (n,k) because that is going to be a forced value. The order of g(x) is n-k, but we don't yet have a value for k so we don't yet know the actual order of g(x).

Fact: For a BCH code defined on GF(q = p^m), the n of (n,k) is set to $n = q-1 = p^m - 1$.

Comment: The Corollary above says that the minimum of (the minimum code distance d) is r+2.

(b) The Narrow-Sense BCH Codes

The **narrow-sense BCH codes** are those BCH codes for which a=1, α is a primitive element of GF(q), g(x) includes the roots { α, α^2 , α^3, ...α^N } so d > N (Corollary above with N = r+1), *and* g(x) has coefficients in GF(p).

This last requirement usually means that g(x) will have many more roots than the N listed above, as we shall see below. Nevertheless, the BCH Bound Corollary above still applies. According to the Corollary, this code has d ≥ N+1. Thus, we can define a so-called designed minimum distance of the code d_{des} = N+1, and then d ≥ d_{des}. This says that the true minimum distance of one of the BCH codes is at least as large as d_{des} = N+1. To summarize:

Narrow-Sense BCH Codes: (9.4)

α = a primitive element of GF(q) n = (order of α) = q-1 = p^m - 1

degree of g(x) = n-k but k is not yet known

roots of g(x) include { α, α^2 , α^3, ...α^N } for selected N, so Corollary says d > N

g(x) includes whatever *other* roots are needed such that g(x) has coefficients in GF(p).
When this is known, we then know the order of g(x) and then we know the k of n-k.

$d \geq d_{des}$ where d_{des} = N+1

$t \geq t_{des}$ where t_{des} = Int(N/2) // t = Int[(d-1)/2] from (7.32)

From our work in Chapter 5, we know immediately how to construct a generator g(x) which has coefficients in GF(p) and which includes the roots { α, α^2 , α^3, ...α^N } in GF(q). Recall that the minimum polynomial m(x) of α has α as a root, and has coefficients in GF(p). Therefore, an obvious candidate for g(x) is to take the product of the minimum polynomials of all the roots in the list { α, α^2 , α^3, ...α^N }. Each minimum polynomial includes not just its labeling root α, but also all the conjugate roots of α as seen in (5.14) and (5.17). Due to the conjugates, it may happen that several roots have the same m(x). In this case, we want to keep only one copy of this m(x). Thus we propose the following g(x) for the narrow-sense BCH code:

$$g_N(x) = \text{LCM}[\, m_1(x)m_2(x)m_3(x)m_4(x) \ldots m_N(x)] \qquad \text{// arbitrary p} \qquad (9.5)$$

where LCM = Least Common Multiple just means any duplicate $m_i(x)$ are thrown out. It was shown in Fact 19 (5.45) that no root of GF(q) appears more than once in this $g_N(x)$, that the degree of $g_N(x)$ is some s ≤ q-1 = n, and that $g_N(x)$ divides evenly into x^{q-1} - 1 = x^n -1, as required by cyclic code definition (8.1) item (b). If we define k ≡ n-s, then the degree of $g_N(x)$ is s = n-k, and we have then obtained the required k for our (n,k) code.

Polynomial $m_1(x)$ includes $(x-\alpha)$ and, since α is assumed to be a primitive element, $m_1(x)$ is always a primitive polynomial. The other m_i appearing in $g_N(x)$ may or may not be primitive depending on whether α^i is primitive.

Note that the order s of g(x) (and hence the value of k) is highly dependent on the roots α^i.

The case N = 1 gives $g(x) = m_1(x)$ all by itself. These result in the Hamming Codes which we discuss in a separate section below.

In Chapter 5 we noted that $m_1(x)$ contains all roots of the conjugate set of α, which consists of the distinct elements of this set:

$$\{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3} \quad , ... \alpha^{p^{m-1}} \} \qquad \alpha^{p^m} = \alpha^q = \alpha \qquad \text{m elements} \qquad (5.14)$$

Here then are the various conjugate lists for the elements α^i of our list $\{ \alpha, \alpha^2, \alpha^3, ...\alpha^N \}$

$$\begin{array}{lll}
\{ \alpha, \ \alpha^p, \ \alpha^{p^2}, \ \alpha^{p^3}, ... \ \alpha^{p^{m-1}} \} & \alpha & \text{roots of } m_1(x) \\
\{ \alpha^2, \alpha^{2p}, \alpha^{2p^2}, \alpha^{2p^3}, ... \alpha^{2p^{m-1}} \} & \alpha^2 & \text{roots of } m_2(x) \\
\{ \alpha^3, \alpha^{3p}, \alpha^{3p^2}, \alpha^{3p^3}, ... \alpha^{3p^{m-1}} \} & \alpha^3 & \text{roots of } m_3(x) \\
..... & & \\
\{ \alpha^p, \alpha^{pp}, \alpha^{pp^2}, \alpha^{pp}, ... \ \alpha^{pp^{m-1}} \} & \alpha^p & \text{roots of } m_p(x) \\
..... & &
\end{array} \qquad (9.6)$$

and so on. Therefore, as you build up the $g_N(x)$ there will be repeats among the $m_i(x)$. In general, every p^{th} $m_i(x)$ will be a duplicate of some earlier $m_i(x)$, so every p^{th} one can be thrown out. For example, $m_p(x)$ contains the root α^p, but so does $m_1(x)$, as shown above. Recall from (5.38) that no root can appear in two distinct conjugate sets, so these must be the same conjugate set and one gets crossed out.

For the special case p=2, this means that every *even* $m_i(x)$ can be thrown out, so we are left with:

$$g_{N+1}(x) = g_N(x) = \text{LCM}[\ m_1(x)m_3(x)m_5(x)m_7(x) \ ... \ m_N(x)] \qquad // \ p = 2, \ N=\text{odd} \qquad (9.7)$$

In this case, $g_2 = g_1$, $g_4 = g_3$ and so on.

The general procedure for developing narrow-sense BCH codes is as follows. (9.8)

(1) Select p and m, so you have $n = q-1 = p^m - 1$. You have now selected a particular Galois Field $GF(p^m)$, as described in much detail in Chapter 4, and you have selected n of (n,k).

(2) Construct the enumeration table for $GF(p^m)$ exactly as described in Chapter 6. As noted there, this listing implicitly contains the entire algebra of the field, that is, it implies the complete • and + field operation tables. You then know how to multiply or add any two elements of the field $GF(p^m)$.

(3) for each power α^i construct the minimal polynomial $m_i(x)$. Chapter 5 showed exactly how this is done.

(4) Then for each value of N = 1,2,3,.... compute g(x) exactly as shown above,

$$g_N(x) = \text{LCM}[\, m_1(x)m_2(x)m_3(x)m_4(x) \ldots m_N(x)] \ .$$

For each N, you find that g(x) has some degree n-k, so for each N, you get a value of k.

From the (9.4) root listing $\{ \alpha, \alpha^2, \alpha^3, \ldots \alpha^N \}$ we know that $N \le n$, since N cannot be larger than the number of non-zero elements in GF(q). Thus also from (9.4) $t_{des} \le \text{Int}(n/2)$. So,

$$N \le n \qquad t_{des} \le \text{Int}(n/2) \tag{9.9}$$

Example: GF(32 = 2^5). We found all the minimum polynomials in (5.28), then we expanded them later on in (6.22). Since 2^5-1 = 31 is prime, they are all primitive polynomials of degree m = 5. Here they are. Here we call them $m_i(x)$ instead of $p_i(x)$,

$$\begin{aligned}
m_1(x) &= (x-\alpha)\,(x-\alpha^2)\,(x-\alpha^4)\,(x-\alpha^8)\,(x-\alpha^{16}) &&= x^5 + x^3 + x^2 + x + 1 \\
m_3(x) &= (x-\alpha^3)\,(x-\alpha^6)\,(x-\alpha^{12})\,(x-\alpha^{24})\,(x-\alpha^{17}) &&= x^5 + x^4 + x^3 + x + 1 \\
m_5(x) &= (x-\alpha^5)\,(x-\alpha^{10})\,(x-\alpha^{20})\,(x-\alpha^9)\,(x-\alpha^{18}) &&= x^5 + x^2 + 1 \\
m_7(x) &= (x-\alpha^7)\,(x-\alpha^{14})\,(x-\alpha^{28})\,(x-\alpha^{25})\,(x-\alpha^{19}) &&= x^5 + x^4 + x^2 + x + 1 \\
m_{11}(x) &= (x-\alpha^{11})\,(x-\alpha^{22})\,(x-\alpha^{13})\,(x-\alpha^{26})\,(x-\alpha^{21}) &&= x^5 + x^3 + 1 \\
m_{15}(x) &= (x-\alpha^{15})\,(x-\alpha^{30})\,(x-\alpha^{29})\,(x-\alpha^{27})\,(x-\alpha^{23}) &&= x^5 + x^4 + x^3 + x^2 + 1 \ . \qquad \textit{(6.22)}
\end{aligned}$$

We can consider the possibilities $\{ \alpha, \alpha^2, \alpha^3, \ldots \alpha^N \}$ one at a time:

$$\begin{array}{llll}
N = 1 & \{\alpha\} & g_1(x) = \text{LCM}[m_1(x)] & = m_1(x) \\
N = 2 & \{\alpha, \alpha^2\} & g_2(x) = \text{LCM}[m_1(x)m_2(x)] & = m_1(x) \\
N = 3 & \{\alpha, \alpha^2, \alpha^3\} & g_3(x) = \text{LCM}[m_1(x)m_2(x)m_3(x)] & = m_1(x)\, m_3(x) \\
N = 4 & \{\alpha, \alpha^2, \alpha^3, \alpha^4\} & g_4(x) = \text{LCM}[m_1(x)m_2(x)m_3(x)m_4(x)] & = m_1(x)\, m_3(x) \\
N = 5 & \{\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5\} & g_5(x) = \text{LCM}[m_1(x)m_2(x)m_3(x)m_4(x)m_5(x)] & = m_1(x)\, m_3(x)\, m_5(x) \\
\ldots\ldots & & & \qquad (9.10)
\end{array}$$

- As just noted, we always have $g_{N+1}(x) = g_N(x)$ for any odd N since p = 2.
- Since all the $m_i(x)$ are of degree 5, all the $g_i(x)$ will have a degree that is a multiple of 5. Eventually as N increases, all 6 of the m_i form g and then g has degree 30, the most it can be.
- For each value of N we know t_{des} = Int(N/2), we can compute s = n-k, the order of g(x), and thus k, and the code is (31,k).

We then arrive at this table, where the $g_i(x)$ are expressed in a recursive manner:

N	t(des)	generator g(x)	n-k = degree of g(x)	k	BCH code (n,k)	
1	0	$g_1(x) = m_1(x)$	5	26	(31,26)	
2	**1**	$\mathbf{g_2(x) = g_1(x)}$	**5**	**26**	**(31,26)**	
3	1	$g_3(x) = g_1(x)m_3(x)$	10	21	(31,21)	
4	**2**	$\mathbf{g_4(x) = g_3(x)}$	**10**	**21**	**(31,21)**	
5	2	$g_5(x) = g_3(x)m_5(x)$	15	16	(31,16)	
6	**3**	$\mathbf{g_6(x) = g_5(x)}$	**15**	**16**	**(31,16)**	
7	3	$g_7(x) = g_5(x)m_7(x)$	20	11	(31,11)	
8	4	$g_8(x) = g_7(x)$	20	11	(31,11)	
9	4	$g_9(x) = g_7(x)$	20	11	(31,11)	
10	**5**	$\mathbf{g_{10}(x) = g_9(x)}$	**20**	**11**	**(31,11)**	
11	5	$g_{11}(x) = g_{10}(x)m_{11}(x)$	25	6	(31,6)	
12	6	$g_{12}(x) = g_{11}(x)$	25	6	(31,6)	
13	6	$g_{13}(x) = g_{11}(x)$	25	6	(31,6)	
14	**7**	$\mathbf{g_{14}(x) = g_{13}(x)}$	**25**	**6**	**(31,6)**	
15	7	$g_{15}(x) = g_{13}(x)m_{15}(x)$	30	1	(31,1)	
16	8	$g_{16}(x) = g_{15}(x)$	30	1	(31,1)	
17	8	$g_{17}(x) = g_{15}(x)$	30	1	(31,1)	
.....		.....	30	1	(31,1)	
31	**15**	$\mathbf{g_{31}(x) = g_{29}(x)}$	**30**	**1**	**(31,1)**	(9.11)

In general, we see that several sequential values of N all generate the same code (n,k). We put in bold the bottommost line in each group, since this line displays the maximum possible t_{des} for that code. We can therefore compress the above table:

N	t(des)	generator g(x)	n-k = degree of g(x)	k	BCH code	
1-2	1	$g_2(x) = g_1(x)$	5	26	(31,26)	
3-4	2	$g_4(x) = g_3(x)$	10	21	(31,21)	
5-6	3	$g_6(x) = g_5(x)$	15	16	(31,16)	
7-10	5	$g_{10}(x) = g_9(x)$	20	11	(31,11)	
11-14	7	$g_{14}(x) = g_{13}(x)$	25	6	(31,6)	
15-31	15	$g_{31}(x) = g_{29}(x)$	30	1	(31,1)	(9.12)

Thus, for p=2, m=5, we can identify a set of six narrow-sense BCH codes of length n=31. They are able to correct at least 1,2,3,5,7 and 15 errors. For all these codes, a coding symbol is an element of $GF(2^5)$ which can be regarded as an extended nibble of 5 bits.

Recall from just below Fig 7.8 that, since q = n+1, there are $q^k = (n+1)^k$ code words in space C^k which is embedded in the larger space V^n which has $q^n = (n+1)^n$ points. For example, in the (31,1) code listed above, there are a mere $(31+1)^1 = 32$ code words (each code block has only k=1 GF(32) data symbol) living in the monstrous space V^{31} which has $(32)^{31} \sim 5 \times 10^{46}$ points, all but 32 of which represent illegal code words. For this code, d = 31 and each legal code word is surrounded by a huge protective sphere of radius t ~ 15 units which does not include any other legal code words. Any bad code

word in this sphere is corrected to the center point legal code word, and that is why such a code can correct even a 15 symbol error in the 31 symbol code word. Since this code has the horrible code rate k/n = 1/31 and is overkill on correction, it is often omitted from listings of BCH codes. The (31,6) code puts $(32)^6$= 1,073,741,824 code words in the same huge V^{31} space and has code rate k/n = 6/31 which is six times better than that of the (31,1) code, and it can still correct up to 7 symbol errors in a 32 symbol code word.

The specific generators of the length 31 BCH codes appearing in the above table are

$$g_2(x) = m_1(x)$$
$$g_4(x) = m_1(x)\, m_3(x)$$
$$g_6(x) = m_1(x)\, m_3(x)\, m_5(x)$$
$$g_{10}(x) = m_1(x)\, m_3(x)\, m_5(x)\, m_7(x)$$
$$g_{14}(x) = m_1(x)\, m_3(x)\, m_5(x)\, m_7(x)\, m_{11}(x)$$
$$g_{31}(x) = m_1(x)\, m_3(x)\, m_5(x)\, m_7(x)\, m_{11}(x)\, m_{15}(x)\ . \qquad (9.13)$$

We can have Maple compute each of these GF(2) polynomials:

```
m1  := x^5+x^3+x^2+x+1:
m3  := x^5+x^4+x^3+x+1:
m5  := x^5+x^2+1:
m7  := x^5+x^4+x^2+x+1:
m11 := x^5+x^3+1:
m15 := x^5+x^4+x^3+x^2+1:
g2  :=  sort(expand(m1) mod 2);
g4  :=  sort(expand(m1*m3) mod 2);
g6  :=  sort(expand(m1*m3*m5) mod 2);
g10 := sort(expand(m1*m3*m5*m7) mod 2);
g14 := sort(expand(m1*m3*m5*m7*m11) mod 2);
g31 := sort(expand(m1*m3*m5*m7*m11*m15) mod 2);
```

(9.14)

$$g2 := x^5 + x^3 + x^2 + x + 1$$
$$g4 := x^{10} + x^9 + x^4 + x^3 + 1$$
$$g6 := x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^4 + x^3 + x^2 + 1$$
$$g10 := x^{20} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x + 1$$
$$g14 := x^{25} + x^{21} + x^{20} + x^{19} + x^{17} + x^{15} + x^{12} + x^{10} + x^9 + x^8 + x^7 + x^4 + x^3 + x + 1$$
$$g31 := x^{30} + x^{29} + x^{28} + x^{27} + x^{26} + x^{25} + x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

A table of all narrow-sense BCH codes for GF(2^m) with m = 1 to 10 appears on pages 274-5 of Pederson and Weldon.

The significance of the BCH codes is that they provide clear-cut codes for correcting multi-symbol errors. The Hamming codes for p=2 discussed below only correct single bit errors. We have seen that the BCH codes are completely based on Galois Field theory.

(c) Narrow-Sense BCH Codes with N = 1 and Hamming Codes

The generator g(x) for a narrow-sense BCH code with N = 1 is simply (see (9.5))

$$g(x) = g_1(x) = m_1(x) \ . \qquad (9.15)$$

Here $m_1(x)$ is the minimal polynomial of some primitive element α of GF(q). Since this makes $m_1(x)$ a primitive polynomial, $m_1(x)$ has degree m (see (5.15) (a) and elsewhere).

For any narrow-sense BCH code we have n = q-1 and k is determined by n-k being the degree of the generator g(x). When N = 1, the generator is $m_1(x)$ which has degree n-k = m, so k = n-m. Thus for the N=1 case we have

$$n = q-1 = p^m - 1 \qquad (p = \text{prime}, m = \text{integer}) \qquad \text{// as for all narrow-sense BCH codes}$$
$$n-k = m$$
$$k = n-m = p^m - 1 - m \quad \Rightarrow \ (n,k) = (p^m-1, p^m-1-m) \qquad (9.16)$$

From (9.4) with N = 1, we obtain $t_{des} = \text{Int}(N/2) = \text{Int}(1/2) = 0$, which is not particularly promising. There is hope though since t_{des} is a lower bound and then $t \geq 0$ is at least possible.

1. The H matrix for the Narrow-Sense BCH Codes with N = 1

In (8.31) the parity check matrix H for any Galois-induced cyclic code is shown in terms of α_i which are the n-k roots of g(x) as shown above (8.30),

$$H = \begin{vmatrix} 1 & (\alpha_1) & (\alpha_1)^2 & (\alpha_1)^3 & \dots & (\alpha_1)^{n-1} \\ 1 & (\alpha_2) & (\alpha_2)^2 & (\alpha_2)^3 & \dots & (\alpha_2)^{n-1} \\ 1 & (\alpha_3) & (\alpha_3)^2 & (\alpha_3)^3 & \dots & (\alpha_3)^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & (\alpha_{n-k}) & (\alpha_{n-k})^2 & (\alpha_{n-k})^2 & \dots & (\alpha_{n-k})^{n-1} \end{vmatrix} \qquad (8.31)$$

For the N=1 narrow-sense BCH codes we have $g(x) = m_1(x)$. From Big Theorem 3 (5.17), and from (5.20) which says k = m , we know that $m_1(x)$ can be written this way,

$$m_1(x) = (x - \alpha) \bullet (x - \alpha^p) \bullet (x - \alpha^{p^2}) \bullet (x - \alpha^{p^3}) \dots \dots (x - \alpha^{p^{m-1}}) \qquad \alpha^{p^m} = \alpha \qquad (5.17)$$

Thus we make this identification of the roots of g(x) (recall that for N=1 case, n-k = m)

$$\{ \alpha_1, \alpha_2 \dots\dots \alpha_{n-k} \} = \{ \alpha, \alpha^p, \alpha^{p^2}, \alpha^{p^3} \quad , \dots . \ \alpha^{p^{m-1}} \} \quad \text{or} \quad \alpha_i = \alpha^{p^{i-1}} \qquad (9.17)$$

so the H matrix takes this form, $n = q-1$

$$H = \begin{vmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \\ 1 & (\alpha^p) & (\alpha^p)^2 & (\alpha^p)^3 & \dots & (\alpha^p)^{n-1} \\ 1 & (\alpha^{p^2}) & (\alpha^{p^2})^2 & (\alpha^{p^2})^3 & \dots & (\alpha^{p^2})^{n-1} \\ 1 & (\alpha^{p^3}) & (\alpha^{p^3})^2 & (\alpha^{p^3})^3 & \dots & (\alpha^{p^3})^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & (\alpha^{p^{i-1}}) & (\alpha^{p^{i-1}})^2 & (\alpha^{p^{i-1}})^3 & \dots & (\alpha^{p^{i-1}})^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & (\alpha^{p^{m-1}}) & (\alpha^{p^{m-1}})^2 & (\alpha^{p^{m-1}})^3 & \dots & (\alpha^{p^{m-1}})^{n-1} \end{vmatrix} \qquad (9.18)$$

As usual, there are n-k (=m) rows and n columns, and the matrix elements are symbols in GF(q=p^m).

Fact: The first row of H contains all q-1 non-zero elements of GF(q).

Proof: Since α is a primitive element of GF(q), it enumerates all non-zero elements.

Fact: Each column of H is a conjugate set of GF(q).

Proof: See (5.14) and (5.16). Typically most columns include duplicate elements such as the first. The only column that is guaranteed to have m distinct elements is the column headed by α since the conjugate set of a primitive element α always has m distinct elements, according to (5.15). Typically a conjugate set will be repeated multiple times among the columns of H with different ordering of the elements.

We know from (3.10) that the elements of GF(q=p^m) can be represented by remainder polynomials of degree less than m whose coefficients lie in GF(p), and which coefficients can be represented by an m-tuple as in (4.6). Since each element of the above H matrix (9.18) is an element of GF(q), we can *think* of each single matrix element (like α^2) as an m-tuple column vector. From this point of view, the H matrix then has m^2 rows and n columns. This view will come into play in the next section.

Recall from (8.1) that c(x) = d(x)g(x) where d(x) is a data polynomial and c(x) is a code word polynomial.

Fact: The i^{th} row of H corresponds to c(β) = 0 for $\beta = \alpha^{p^{i-1}}$ where c(x) is any code polynomial.

Proof: From (7.21) we know that the H matrix kills good code words meaning $H\mathbf{c}^T = \mathbf{0}$ where $\mathbf{c}^T$ is a column vector with coefficients c_i of polynomial c(x). When the i^{th} row of H in (8.31) is multiplied by $\mathbf{c}^T$ the result is 0, as was shown in (8.32),

$$c_0 + c_1\ (\alpha_i) + c_2\ (\alpha_i)^2 + c_3\ (\alpha_i)^3 + \dots\ c_{n-1}\ (\alpha_i)^{n-1} = 0 \qquad (8.32)$$

Since c(x) = d(x)g(x), $g(\alpha_i) = 0 \Rightarrow c(\alpha_i) = 0$, and this is exactly what (8.32) says: all the α_i roots of g(x) are passed through to c(x). In the current context we have the same idea,

$$c_0 + c_1\ (\alpha^{p^{i-1}}) + c_2\ (\alpha^{p^{i-1}})^2 + c_3\ (\alpha^{p^{i-1}})^3 + \dots\ c_{n-1}\ (\alpha^{p^{i-1}})^{n-1} = 0 \qquad (9.19)$$

which says $c(\alpha^{p^{i-1}}) = 0$. So, each of the m rows of H in (9.18) corresponds to a code word c(x) vanishing at some root $\beta = \alpha^{p^{i-1}}$.

2. The H_1 Matrix and Hamming Codes

Consider just the first row of the H matrix of (9.18), corresponding to c(x) vanishing at the primitive element α ,

$$H_1 = [\ 1, \alpha, \alpha^2, \alpha^3, \ldots. \alpha^{n-1}\ ] . \qquad (9.20)$$

Taking each element like α^2 to be an m-tuple column vector as described above, we can regard H_1 as a matrix with m rows and n columns. It is certainly true that $H_1\mathbf{c}^T = 0$ which then appears this way

$$\left[\begin{pmatrix} 1 \\ 0 \\ 0 \\ . \\ . \end{pmatrix} \begin{pmatrix} * \\ * \\ * \\ . \\ . \end{pmatrix} \begin{pmatrix} * \\ * \\ * \\ . \\ . \end{pmatrix} \ldots \begin{pmatrix} * \\ * \\ * \\ . \\ . \end{pmatrix} \right] \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ . \\ . \\ c_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ . \\ . \end{pmatrix} \quad \text{or} \quad H_1\mathbf{c}^T = 0 \qquad (9.21)$$

where each * is some element in GF(p). The **c** column vector has n components, while all the other column vectors have m components.

In any narrow-sense BCH code, we know that the coefficients of the generator g(x) lie in GF(p). In general, the coefficients of the data polynomial d(x) lie in GF(q) and since c(x) = g(x) d(x), the coefficients of c(x) also lie in GF(q). We can, however, restrict our interest to data words with coefficients in GF(p), in which case the code polynomials c(x) will also have coefficients limited to GF(p). When this restriction is made, the code falls into a general class called **Hamming codes**.

We have shown that there are two related Pictures going on here.

The **Full Picture** has $H\mathbf{c}^T = \mathbf{0}$ for the full matrix H in (9.18). This H matrix is the parity check matrix for an N=1 narrow-sense BCH code which inputs d(x) data polynomials with coefficients in GF(q), and outputs c(x) code word polynomials also with coefficients in GF(q), although the generator matrix has elements in GF(p). Matrix H has m rows and n columns. Matrix H generates a dual code (n,n-k) which is dual to the (n,k) N=1 narrow-sense BCH code.

The **Hamming Picture** has $H_1\mathbf{c}^T = \mathbf{0}$ for the matrix H_1 in (9.20) and (9.21). This H_1 matrix is the parity check matrix for a Hamming code which inputs d(x) data polynomials with coefficients in GF(p), and outputs c(x) code word polynomials also with coefficients in GF(p). It uses the same generator g(x) used in the Full Picture. Matrix H_1 has m rows and n columns. Matrix H_1 generates a dual code (n,n-k) which is dual to the (n,k) Hamming code.

Obviously the Hamming code is a subset or subcode of the full N=1 narrow-sense BCH code. It is the restriction of the full code to data polynomials with coefficients in GF(p).

A **Hamming Code** (as we define it) is therefore a narrow-sense N=1 BCH cyclic code which has the coefficients of d(x) restricted to lie in GF(p). That is to say, the data vectors and code vectors are in a vector spaces defined over the field GF(p).

Some authors might further restrict Hamming codes to having p = 2 so they are then binary codes -- the data coefficients and code coefficients are bits. In the case from above we have

$$n = q-1 = 2^m - 1 \qquad (m = \text{integer}) \qquad \text{// binary Hamming Code}$$
$$n-k = m$$
$$k = n-m = 2^m - 1 - m \quad \Rightarrow \quad (n,k) = (2^m-1,\ 2^m-1-m) \tag{9.22}$$

We might call this a **binary Hamming code**. For m=3 one has (n,k) = (7,4) and this is the specific code that Richard Hamming invented and used in 1950 to deal with punched card reader errors (References).

Comment: When one first sees the Hamming Codes, the fact that (n,k) = (2^m-1, 2^m-1-m) seems very strange. One has no clue as to where these peculiar numbers are coming from. In retrospect, these numbers fall out very naturally from the concept of Galois-induced cyclic codes. The 2^m-1 is just the number of non-zero elements in GF(2^m), and the generator $m_1(x)$ has degree m, so n-k = m and k = n-m.

Fact: (9.23)

A Hamming code with p = 2 can correct single-symbol errors (t=1).
A Hamming code with p > 2 cannot correct any errors (t=0).

Proof: This Fact is proved in the following section.

3. Error Correction capabilities of Hamming Codes

The Hamming parity check matrix H_1 gives us some insight as to why t=0 when p>2. We know that all q-1 elements of {GF(q)-0} must appear in the first row of H, called H_1 in (9.20), and that means that every non-zero m-tuple (components in GF(p)) must appear as a column of H_1 in (9.21).

For p = 2, no m-tuple is a multiple of another m-tuple simply because there *are* no multiple m-tuples in the set of m-tuples for p = 2. Thus, no pair of columns in H_1 is linearly dependent. Since a column of zeros does not appear, no single column is linearly dependent. We know for sure that many columns can be written as a linear combination of two other columns. Thus, the minimum number of dependent columns in H will be 3. According to Fact 4 (7.35), the code must have a minimum distance of d = 3, and then (7.32) says that t = Int[(d-1)/2] = 1. Thus, the p = 2 Hamming code can correct single-symbol errors.

When p > 2, we still have all q-1 m-tuples appearing as the columns of H_1, but of course q is larger now, being p^m instead of 2^m. This set of m-tuples includes many multiples, such as (2,0,0...) = 2(1,0,0...). Since at least one pair of columns has one being a multiple of the other (there will be many such pairs), the minimum number of dependent columns is 2, and then (7.35) says that d = 2 and t = Int[(d-1)/2] = 0. That is why Hamming codes for p > 2 cannot correct even single symbol errors.

4. A Modified Hamming Code

There are many kinds of modified Hamming codes. Here we describe one which has interesting algebraic properties. It is mentioned on page 221 of Peterson and Weldon. We derive this code using two Lemmas which are then proven below.

It was noted in the previous section that the non-binary Hamming codes (p > 2) cannot correct even 1-symbol errors due to the existence of sets of columns in the H_1 matrix which are multiples of each other. This problem can be repaired by eliminating the extra columns and the result is a modified Hamming Code that can in fact correct single-symbol errors for p > 2.

Toward this end, we reconsider the (3.16) representation $GF(p^m) = R / (f(x))$ where we have a chart whose rows are labeled by the remainders of polynomials over GF(p) divided by some irreducible f(x). In the resulting chart, one could combine together into one row all rows whose remainders are multiples of each other. For example, the rows {x}, {2x}....{(p-1)x} = {x}, 2{x}....(p-1){x} would be combined into a single row of a new chart which of course then has fewer rows. If we ignore the first row {0}, the original chart has p^m -1 non-zero rows, and the new chart then has $(p^m-1)/(p-1)$ non-zero rows.

As shown in **Lemma 1**, there is no integer m' such that, given m and p, $(p^m-1)/(p-1) = p^{m'}-1$. Thus, the order of the new chart (including the zero row) is not of the form $p^{m'}$, so the rows in the new chart do not represent a field, since all finite fields have order of the form $p^{m'}$. The elements of this new chart only form a ring, which we might loosely write as $GF(p^m)/Z_p$. So we have constructed

$$GF(p^m)/Z_p \qquad r \equiv (p^m-1)/(p-1) = \text{the number of non-zero elements} \tag{9.24}$$

Notice that $r = p^{m-1} + p^{m-2}..... + 1$ is of course an integer. Each row of the new chart can be represented by a polynomial p(x) which represents not just itself, but all integer multiples of itself by an integer in Z_p. If a polynomial is multiplied by k, its remainder is multiplied by k, and so it is associated with the same row of the new chart.

In terms of the m-tuple notation for polynomial coefficients, we can regard the non-zero chart rows as being represented by a set of m-tuples none of which are multiples of each other. For example, (2,0,0) = 2(1,0,0) is in the same row as (1,0,0). Since each original m-tuple had m digits, each of the new set of m-tuples obviously also has m digits. Recall that m is the degree of f(x) in $GF(p^m) = R /(f(x))$.

If α is a primitive element of $GF(q=p^m)$, then the order of α is q-1 and we can enumerate the non-zero elements of GF(q) as powers of α. If we define $\beta \equiv \alpha^{p-1}$, we find that β is *not* a primitive element because it's order is less than q-1. In fact, its order is r = (q-1)/(p-1), which is easily shown since $\alpha^{q-1} = 1$:

$$\beta^r = (\alpha^{p-1})^{[(q-1)/(p-1)]} = \alpha^{q-1} = 1 \ .$$

In **Lemma 2** we show that each element of the subgroup $\{1, \beta, \beta^2....\beta^{r-1}\}$ can be associated with a non-zero row of our new chart which defines $GF(p^m)/Z_p$. Thus, each power of β can be represented by one of the m-tuples which label the rows of the new chart. Recall that no m-tuples of this set are multiples of each other. We then define a Hamming type parity check matrix in this way

$$H'_1 \equiv \ [1, \beta, \beta^2....\beta^{r-1}] \ . \tag{9.25}$$

The space of code words is the nullspace of H'_1, which is to say $H'_1\mathbf{c}^T = 0$. The H'_1 matrix has m rows and r = (q-1)/(p-1)columns. This is similar to the Hamming matrix H_1 which had m rows but n = q-1 columns. The difference is that all the "multiple columns" in H_1 have been removed in H'_1. Therefore, in H'_1 no pair of columns is a multiple of each other, and then the minimum number of dependent columns is 3. According to (7.35) this new code has minimum distance d = 3 and then t = Int[(d-1)/2] = 1.

Thus, we arrive at a **modified Hamming code** which can correct single-symbol errors even when p > 2. Based on the size of the H'_1 matrix, this code has

$$n = r \text{ = number of columns in } H'_1$$
$$n-k = m \text{ = number of rows in } H'_1 \quad \Rightarrow \quad (n,k) = (r,r-m) \quad \text{where } r = (p^m-1)/(p-1) \tag{9.26}$$

We now state and prove the two lemmas used above.

Lemma 1: There is no integer n such that, given m and p>1, $(p^m-1)/(p-1) = p^n-1$. (9.27)

Proof: This equality would require that $(p^m-1) = (p-1)(p^n-1)$. The only way to even have a chance is if m > n. So write m = n+s where integer s > 0 would give a solution. Then

$$(p^{n+s}-1) = (p-1)(p^n-1) \quad ? \quad \text{for some integer set } n,s > 0 \text{ and } p > 1$$

If we can show that $(p^{n+s}-1) > (p)(p^n-1)$ then it would certainly be true that $(p^{n+s}-1) > (p-1)(p^n-1)$ and then there can be no equality solution. We can in fact show that $(p^{n+s}-1) > (p)(p^n-1)$:

$$p^{n+s}-1 > p(p^n-1) \text{ for all } s>0 \quad ?$$

$$p^{n+s}-1 > p^{n+1}-p \text{ for all } s>0 \quad ?$$

If s =1, this is certainly true since -1 > -p. For larger s it is even more true. Thus there is no integer s > 0 which solves the problem and the Lemma is proved.

Lemma 2: Show the correspondence between powers of β and the rows of $GF(p^m)/Z_p$. (9.28)

We know that $\alpha = \{x\}$ is a primitive element of GF(q). Then $\beta = \alpha^{p-1} = \{x\}^{p-1} = \{x^{p-1}\}$. The powers of β are then of the form $\beta^k = \{x^{p-1}\}^k = \{x^{k(p-1)}\}$, so this shows the connection between powers of β and rows of the new chart. Our only concern is that two different powers k and k' in $1 \le k,k' < r$ might put us into the same chart row and then the set of β powers does not have r distinct elements. This would require that for $k \neq k'$ we get $\{ x^{k(p-1)}\} = \{ x^{k'(p-1)}\}$ or $\{x\}^{k(p-1)} = \{x\}^{k'(p-1)}$. That in turn would require that

$$k(p-1) = k'(p-1) + N(q-1) \quad \text{since } \{x\}^{q-1} = \alpha^{q-1} = 1, \; N = \text{integer}$$

where q-1 is the smallest power for which $\{x\}^s = 1$. Then we would need to have

$$k = k' + N(q-1)/(p-1) = k' + N r \quad \Rightarrow \quad k - k' = N r$$

But two integers smaller than r cannot differ by a multiple of r other than for N = 0. Thus, there is a 1-to-1 correspondence between the r powers β^k and the r non-zero elements of $GF(p^m)/Z_p$.

(d) The Reed-Solomon Codes

We have seen that the narrow-sense BCH codes (9.4) are a special case of the general class of BCH codes (9.3). The Reed-Solomon codes are another special case. Recall that all BCH codes have n = q-1.

The reason we chose the minimum polynomials in (9.5) for the narrow-sense BCH codes was to make sure g(x) had coefficients in the field GF(p). Suppose we consider generators g(x) that have coefficients in GF(q) = GF(p^m). As noted in Chapter 8 (j), this means we give up a significant amount of design simplicity in coding hardware. In this case, there is no need to fool around with the minimum polynomials, and we have a much simpler expression for a generator containing the desired roots. As with the narrow-sense BCH codes, we set a = 1 in (9.3) and select α to be a primitive element of GF(q). Thus, our candidate g(x) in GF(q) is simply this:

$$g_N(x) = (x - \alpha)(x-\alpha^2)(x-\alpha^3).....(x-\alpha^N) \qquad g(x) \text{ has coefficients in GF}(p^m) \tag{9.29}$$

Notice that there is no uncertainty about the degree of $g_N(x)$ as there is in the narrow-sense BCH codes. The degree here is n-k = N.

If g(x) has coefficients in GF(p^m), then so do the code polynomials c(x) = g(x)d(x) of (8.1). There is then no motivation to restrict the coefficients of the data polynomials d(x) to GF(p), so they can be in GF(q) as well.

The error correcting capability of the above code is exactly the same as for the corresponding narrow sense BCH code expressed at the end of (9.4), and for the same reason: an application of the BCH Bound Corollary (9.3). Thus we have,

$$d \geq d_{des} \quad \text{where} \quad d_{des} = N+1$$

$$t \geq t_{des} \quad \text{where} \quad t_{des} = \text{Int}(N/2) \qquad // \; t = \text{Int}[(d-1)/2] \text{ from (7.32)} \tag{9.30}$$

Example: p=2 and m=8, n = q-1 = 2^8-1 = 255. The data symbols and code symbols are then *bytes*. Each code word contain 255 bytes. If N=even, then t_{des}= N/2 and we know that at least N/2 erroneous bytes in such code words can be corrected. Of course k = n-N = 255-N is the size of the data word.

Codes with code symbols in p^m and with g(x) as shown above are known as **Reed-Solomon codes** (1960). α is a primitive element of GF(p^m) and the code can correct at least t = Int(N/2) bad symbols errors per code word. Thus for even N, the code can correct half the number of parity check symbols (n-k). Note that a single bad symbol error might involved multiple bad bits in that symbol.

For the special case p=2, the parameters of a Reed Solomon code are as follows:

$$n = 2^m - 1$$
$$n-k = N$$
$$k = 2^m - 1 - N = n-N \tag{9.31}$$

Recall from (7.34) the upper bound on error correction capability of any code:

$$d \le n-k+1 \qquad \text{or} \qquad t \le \text{Int}[\,(n-k)/2\,] \ . \tag{7.34}$$

For the arbitrary-p Reed-Solomon codes we have

$$d = (n-k)+1$$
$$t = \text{Int}(N/2) = \text{Int}[(n-k)/2] \ . \tag{9.32}$$

Thus, all Reed-Solomon codes saturate this bound -- the have "the best d possible". This is a very strong selling point for R-S codes in general. There can exist no other codes which can correct more errors for a given n-k. Reed Solomon codes are **maximum-distance-separable**, where the phrase evokes Fig 7.3.

In terms of the (n,k) notation, we have the Reed-Solomon codes specified as:

$$(n,k) = (n,n-N) \qquad \text{where } n = p^m - 1 \tag{9.33}$$

Here is a list of the R-S codes for p=2 and m=3 (code symbol = 3-bit nibble)

m	n	N	t(des)	k	(n,k)	
3	7	1	0		6	(7,6)
3	**7**	**2**	**1**		**5**	**(7,5)**
3	7	3	1		4	(7,4)
3	**7**	**4**	**2**		**3**	**(7,3)**
3	7	5	2		2	(7,2)
3	**7**	**6**	**3**		**1**	**(7,1)**

(9.34)

There seems little reason to use a code with odd N, since the next lower even N code has the same t_{des} and one gets one more data symbol and needs one less parity symbol. For this reason, one often sets N = 2t and summarizes the codes as:

$$(n,k) = (n,n-2t) \qquad \text{where } n = p^m - 1. \tag{9.35}$$

Obviously, there are a *lot* of Reed-Solomon Codes. And of course there are various *modified* Reed-Solomon codes. Codes can be "punctured" or "expurgated" or "shortened" or tortured in other ways. An (n,k) code can, for example, be "shortened" to any (n_1,k_1) with $n_1 \le n$ and $k_1 \le k$ as long as $n_1-k_1 = n-k$. In this case $n-n_1$ symbols are treated as zeros and are never transmitted, and the receiver knows to regenerate them, so there is no bandwidth waste. Sometimes the minimum distance d is written as a third code argument, so one might see (n_1,k_1,d). The letters RS are sometimes prefixed, as in RS(n,k).

Example: When data is encoded for recording onto CDs, a rather complicated scheme called CIRC is used, which involves two Reed-Solomon codes usually expressed as (32,28) and (28,24). Each of these is a shortened version of the (255,251) standard RS code where data symbols are bytes. Since these codes are maximum distance codes, all three have d = 5, so one might see the notation (32,28,5) and (28,24,5) for the two RS codes used in CIRC.

CIRC stands for Cross-Interleaved Reed-Solomon Coding. As mentioned earlier, the interleaving aspect shuffles the data so that a physical burst error is widely dispersed into the logical data stream. It is often claimed that CIRC can correct a burst of up to 3500 sequential bad bits which could be caused by a 1/10" wide scratch on a CD. It is impressive that a full CIRC decoder can be implemented in a $9 Walmart portable CD player.

(e) Galois Field GF(2^m) Math compared to Digital Filter Math

Another selling point of p=2 Reed-Solomon codes is that one can work with polynomial multipliers, dividers and related circuits that have data paths that are m-bits wide. Hardware is well geared for doing this and a few circuit examples were shown in Chapter 8 (j).

Circuits used in digital filters and related digital applications have a very similar appearance, but there is one major difference that we wish to point out.

In an integer-math digital filter with data paths m-bits wide, the + and • tables are those of modulo-q arithmetic where $q = 2^m$. The field in which the numbers reside is Z_q. For $q=2^8$, an adder does addition then within Z_{256} and there is carry between the bit lines inside the adder.

In a Reed-Solomon polynomial divider or multiplier with data paths m-bits wide (as one might encounter in an encoder or decoder), the + and • tables are those of the field GF($q=2^m$). In this case an adder is just a set of m XOR gates and there is no carry between bit lines within the adder. All addition is done within Z_2.

Earlier it was stressed that the multiplication table for Z_4 is different from that of GF(4). This is true in general for any $q = 2^m$ (except m=1). In both Z_q and GF(q) there is an interaction between all the bit lines during a multiplication.

Thus, in building Reed-Solomon circuits, one can use XOR gates for addition independently on the individual bit lines, but multiplication must be done with some kind of lookup or combinatoric/state machine circuit which correctly computes products of GF(2^m) symbols. As in a digital filter, multiplication cannot be done independently on the individual bit lines.

Chapter 10: Matrix Representation of a Galois Field

Appendix D provides a small collection of matrix facts that the reader might wish to peruse.

(a) How to construct a matrix representation for GF(q)

In Chapter 3 we obtained the following representation of $GF(p^m)$:

$$GF(p^m) = R / (d(x)) \qquad \begin{array}{l} R = \text{ring of polys with coefficients in } Z_p = GF(p) \\ d(x) = \text{degree m, irreducible in R} \quad . \end{array} \qquad \textit{(3.17)}$$

Here we have replaced our usual f(x) by notation d(x) so we can free up f(x) to represent an arbitrary polynomial in the ring R.

R was the set of polynomials f(x) with coefficients in Z_p and with variable x in an unspecified set. The nature of variable x never played much of a role in things, the powers of x just served as inert carriers for the coefficients which were in Z_p.

The above equation is in effect a triple isomorphism in the following sense. On the left we have a field whose elements are the abstract elements of $GF(q=p^m)$, which were referred to in earlier chapters by symbols such as a_i, α, β, g. On the right, we have a "chart" defined by the ring/ideal structure whose rows are isomorphic with GF(q). Finally, each row of the chart is associated with a remainder polynomial r(x) that results when an arbitrary polynomial f(x) in R is divided by d(x). So,

$$\text{abstract elements of GF(q)} \;\leftrightarrow\; \text{rows of the R/I chart} \;\leftrightarrow\; \text{remainders r(x) of polys f(x)/d(x)} \qquad (10.1)$$

One says that either the rows of the chart or the set of remainder polynomials form a "representation" of the abstract field GF(q). Using the symbol $\doteq$ as a sort of isomorphic equality, we would say that

$$\underset{\text{abstract field element}}{a_i} \;\doteq\; \underset{\text{chart row}}{\{ r_i(x) \}} \;\doteq\; \underset{\text{remainder polynomial}}{r_i(x) = \text{Rem}(f_i^{(k)}(x)/d(x))} \qquad (10.2)$$

where $f_i^{(k)}$ just indicates that there are many polynomials in R whose remainder is $r_i(x)$.

One row of interest in the R/I chart is the first row (the ideal) which has 0 remainder,

$$\underset{\text{abstract field element}}{0} \;\doteq\; \underset{\text{chart row}}{\{ 0 \}} \;\doteq\; \underset{\text{remainder polynomial}}{0 = \text{Rem}([q(x)d(x)]/d(x))} \qquad (10.3)$$

So, the polynomials of the form q(x)d(x) in R are associated with abstract field element 0,

$$\underset{\text{polys in R}}{q(x)d(x)} \;\doteq\; \underset{\text{abstract element of GF(q)}}{0} \qquad (10.4)$$

This set of polynomials in R of course forms the ideal I of the R/I chart structure.

Now in what follows we are going the take the unspecified polynomial variable x and specify it to be a square matrix in a certain space of matrices. To stress that x is a matrix, we will write it as X. If we take any polynomial f(x) and evaluate it at x = X, that polynomial evaluates to a matrix, where we assume that the constant term in the polynomial is that constant times the identity matrix I. For example:

$$f(x) = a + bx + cx^2 \qquad \rightarrow \qquad f(X) = aI + bX + cX^2 = \text{a matrix}$$

We wish to select a certain set of matrices to be a "representation" of the field GF(q). Thus, we write (10.4) as

$$\underset{\text{polys in R}}{q(x)d(x)} \doteq \underset{\text{abstract element of GF(q)}}{0} \doteq \underset{\text{matrix in the matrix representation}}{q(X)d(X)} \qquad (10.5)$$

We would like the "all-zeros matrix" of the matrix representation to represent the abstract element 0 of the field GF(q). We can achieve this goal for all q(x) by requiring that d(X) = 0. Thus we are led to our first key equation,

Fact 1: d(X) = 0 (the all-zeros matrix) (10.6)

Another row of interest in the R/I chart is this one, where the remainder is $r_1(x) = x$,

$$\underset{\text{abstract field element}}{\alpha} \doteq \underset{\text{chart row}}{\{ x \}} \doteq \underset{\text{remainder polynomial}}{x} \qquad (10.7)$$

Recall that if d(x) is a primitive polynomial for element α, then d(x) contains (x-α) as a factor, d(α) = 0, and α is a primitive element of GF(q) whose powers enumerate the non-zero elements of GF(q). We will assume from now on that d(x) is in fact a primitive polynomial for some primitive element α, which in the chart representation we identify with {x} and also with remainder polynomial x. Recall from (6.5) that since d(x) is a primitive polynomial, d(α) = 0 for α = {x} *and* α is a primitive element of GF(q). We now add our matrix representation item to (10.7) and say

$$\underset{\substack{\text{primitive abstract}\\ \text{field element}}}{\alpha} \doteq \underset{\text{chart row}}{\{ x \}} = \underset{\substack{\text{remainder}\\ \text{polynomial}}}{x} \doteq \underset{\substack{\text{matrix}\\ \text{representation}}}{X} \qquad (10.8)$$

Therefore we have the following:

Fact 2: The matrix X is a primitive element in the matrix representation of GF(q) and the powers of matrix X enumerate the non-zero elements of GF(q) in the matrix representation. (10.9)

This enumeration capability of a primitive element results from our realization in Big Theorem 1 (4.30) that {GF(q) - 0, •} is a cyclic group, and from (1.10) which says that, by the definition of a cyclic group, there is always some element of the group (called a generator) which enumerates the group. Such a generator is called a primitive element and the minimal polynomial of a primitive element is a primitive polynomial.

Fact 3: (Matrix Representation) If we can find a square matrix X such that d(X) = 0 where d(x) is a primitive polynomial for GF(q) of degree m, then in the matrix representation of GF(q=p^m) the matrix X is a primitive element and its powers generate all the non-zero elements of the matrix representation of GF(q). The identity of GF(q) is represented by the identity matrix ($I=X^0$), and the zero element of GF(q) is represented by the all-zeros matrix. This set of q matrices then forms a matrix representation of GF(q) which is isomorphic to the abstract field GF(q). (10.10)

Notice that nothing has been said so far about the size of the matrix X.

At this point the reader may be a bit dubious that we can willy-nilly replace the scalar polynomial variable x used in all previous chapters with a square matrix X and have all our "Facts" carry over into this matrix world. The reason things *do* carry over is that hardly any "property" of x was utilized in the earlier chapters. As just noted, the variable x and its powers were just inert carriers of the coefficients of Z_p. The variable x was not specified to be in any set or space whatsoever. All we needed was a notion of x^n and nx in order to talk about polynomials in the variable x. Here we make a more specific statement that X is in a certain space of matrices, but the old Facts still work since their derivation made no use of the properties of x beyond those just mentioned. Of course X^n and nX have clear meanings for a matrix X. Basically, scalar additions and multiplications get replaced by matrix additions and multiplications.

Consider for example the Division Algorithm (3.6) which says n(x) = q(x)d(x) + r(x). In the matrix world this is replaced by n(X) = q(X)d(X) + r(X) where now q(X)d(X) is the product of two matrices. One might wonder about dividing two matrices as in n(X)/d(X), but one only need look at the example in (3.7) with scalar x replaced by matrix X and everything works fine. Scalar addition and multiplication is replaced by matrix addition and multiplication at each stage of the long division process.

Here is another example of the change from x to X:

$$x^{q-1} - 1 = (x - \mathbf{1}) \bullet (x - a_2) \bullet (x - a_3) \bullet (x - a_4) \bullet \ldots\ldots (x - a_{q-1}) \; . \qquad (5.1)$$

$$X^{q-1} - 1 = (X - I)(X - A_2)(X - A_3)(X - A_4)\ldots\ldots(X - A_{q-1}) \; . \qquad (5.1)_{matrix}$$

In the first line x is the dummy variable, **1** is the abstract • identity element of GF(q), the a_i are non-zero abstract elements of GF(q), and • indicates the • operator of GF(q). In the second line, both sides of the equation are matrices. X is our square matrix, I is the identity matrix, the A_i are the matrix representations of the non-zero elements of GF(q), and multiplication of the factors is standard matrix multiplication.

As a last example, assuming that X is found such that $0 = d(X) = \Sigma_{i=0}^{m} d_i X^i$, we get the matrix reduction rule ($-d_i = p - d_i$ in Z_p)

$$X^m = (p-d_{m-1})\, X^{m-1} + (p-d_{m-2})\, X^{m-2} + \ldots + (p-d_1)\, X + (p-d_0) \qquad (10.11)$$

which may be compared to (6.9)

$$\alpha^m = (p-c_{m-1})\, \alpha^{m-1} + (p-c_{m-2})\, \alpha^{m-2} + \ldots + (p-c_1)\, \alpha + (p-c_0) \; . \qquad (6.9)$$

Nevertheless, for the dubious reader we provide the next section which makes a detailed specification of the new ring of polynomials R_p^m which is appropriate for our new matrix isomorphism $GF(p^m) = R_p^m/d(X)$. Everything about this ring R_p^m is just an extension of the Chapter 3 (a) scalar ring R where scalar addition and multiplication is replaced by matrix addition and multiplication.

Rather than hold the reader in suspense, we shall right here finish off the logic flow to show exactly how a matrix representation of GF(q) can be constructed in actual practice. Then the following sections can be considered "support material". In the final section we will use Maple to execute our algorithm for creating matrix representations for some sample GF(q) fields.

Looking at Fact 3 (10.10) above, we are faced with this problem: given a primitive polynomial d(x) for GF(q) (which we can find from a table, or which we can figure out by some method outlined earlier), how do we find a square matrix X such that d(X) = 0? If we can solve this problem, then Fact 3 gives complete instructions for constructing a matrix representation of GF(q).

The following famous theorem comes to the rescue:

Theorem: (Cayley-Hamilton). Every square matrix satisfies its own characteristic equation. (10.12)

As reviewed in Appendix D, if X is an mxm matrix, then

(a) $d(x) \equiv \det(xI - X)$ is the characteristic polynomial of matrix X, and it has degree m.

(b) the characteristic (secular) equation says d(x) = 0. Normally this equation is solved to find the eigenvalues of matrix X.

What Cayley-Hamilton says is simply this: for any square matrix X, d(X) = 0.

Proof: See any good linear algebra text. Birkhoff and MacLane give a one page proof on page 341. Wiki has several proofs. Here is a *non*-proof which involves an equation which does *not* "carry over" when x goes to matrix X :

$$d(x) = \det(xI - X) = 0 \quad \Rightarrow? \quad d(X) = \det(XI - X) = 0$$

On the right, d(X) = det(XI - X) is completely meaningless since d(X) is a matrix while any determinant is just a number. The proof takes a little more work than that.

In order that $d(x) \equiv \det(xI - X)$ be a polynomial of degree m, the square matrix X must be an mxm matrix. Then Cayley-Hamilton says d(X) = 0. So if we use this plan to find a matrix X such that d(X)= 0, then our X must be an mxm matrix if we are building a matrix representation for $GF(p^m)$.

We are now faced with a new problem: given a primitive polynomial d(x) of degree m and which we associate with the characteristic polynomial of matrix X, how to we actually construct a matrix X which has this d(x) as its characteristic polynomial? If we can construct such an X, then d(X) = 0, and then Fact 3 is happy.

As shown in the second section following, there is a standard method of constructing a matrix X which has any desired characteristic polynomial. Such a matrix X is called a **companion matrix** to that

polynomial. It is then trivial to write down a matrix X that works, and we are done. In the third section following, we shall have Maple construct some matrix representations using Fact 3 above.

(b) Specification of the ring R_p^m of matrix polynomials with coefficients in Z_p

Let $\mathcal{X}_p^m$ denote the set of mxm matrices whose elements lie in Z_p. There are exactly p^{m^2} matrices in this set. Let X and Y be elements of $\mathcal{X}_p^m$. With set $\mathcal{X}_p^m$ we associate + and • operations defined as follows. First the + operator:

$$Z = X + Y \qquad \Leftrightarrow \qquad Z_{rs} \equiv X_{rs} \oplus Y_{rs} \tag{10.13}$$

where, as before, $\oplus$ and $\otimes$ refer to the operators of Z_p. Thus, addition in $\mathcal{X}_p^m$ is normal matrix addition, but matrix elements are added in Z_p. Now the • operator :

$$Z = X \bullet Y = XY \qquad \Leftrightarrow \qquad Z_{rs} \equiv \Sigma^{\oplus}_{t=1}{}^{m} \, (X_{rt} \otimes Y_{ts}) \, . \tag{10.14}$$

Here X_{rt} and Y_{ts} are elements of Z_p, so $(X_{rt} \otimes Y_{ts})$ is a well-defined element of Z_p and we are adding m such elements with the $\oplus$ operator, all within Z_p. This is the meaning of $\Sigma^{\oplus}$.

Notice that everything is normal matrix addition and matrix multiplication except the only numbers that ever appear lie in Z_p. Examples of the above are:

$$Z_{rs} = X_{rs} \oplus X_{rs} = 2X_{rs} \qquad \Rightarrow \qquad Z = X + X = 2X \qquad \text{// assuming } p > 2 \tag{10.15}$$

$$Z_{rs} \equiv \Sigma^{\oplus}_{t=1}{}^{m} \, (X_{rt} \otimes X_{ts}) = (X^2)_{rs} \; \Rightarrow \qquad Z = X \bullet X = X^2 \, . \tag{10.16}$$

Thus any multiple of X or power of X lies in $\mathcal{X}_p^m$ and as usual all numbers come out lying in Z_p. In particular we have

Fact 1: pX = 0: (10.17)

Proof: $pX = X + X + ... + X \Leftrightarrow pX_{rs} = X_{rs} \oplus X_{rs} \oplus ... \oplus X_{rs} = 0$ since p**a** = 0 for any **a** in Z_p .

With the + and • operations of (10.13) and (10.14), the set $\mathcal{X}_p^m$ forms a ring with identity, the identity being the usual identity matrix with all 1's on the diagonal. The additive identity is the all-zeros matrix. One can go through all the ring properties just as we did in Chapter 3 (a) with the + and • definitions shown here and verify each one. Therefore we have:

Fact 2: Let $\mathcal{X}_p^m$ denote the set of mxm matrices whose elements lie in Z_p, and let the + and • operations for $\mathcal{X}_p^m$ be as given in (10.13) and (10.14). Then $\mathcal{X}_p^m$ is a ring with identity having p^{m^2} elements. (10.18)

Now, let R_p^m be the set of polynomials f(X) whose coefficients lie in Z_p and whose variable X lies in $\mathcal{X}_p^m$. It is easy to show that any f(X) is in fact a matrix lying in $\mathcal{X}_p^m$. Consider:

$$f(X) = \Sigma_i\, a_i\, X^i \qquad \Leftrightarrow \qquad [f(X)]_{rs} = \Sigma_i\, a_i\, [X^i]_{rs} \quad . \tag{10.19}$$

We know that $a_i X^i = X^i + X^i + ...$ is in fact a matrix in $\mathcal{X}_p^m$ having elements in Z_p. And then $\Sigma_i\, a_i\, X^i$ is just a sum of matrices in $\mathcal{X}_p^m$, as defined above. Thus we conclude that f(X) is itself a matrix in $\mathcal{X}_p^m$. Since adding and multiplying polynomials in R_p^m is the same as adding and multiplying matrices in $\mathcal{X}_p^m$, the + and • operations for R_p^m are the same as those for $\mathcal{X}_p^m$. For example,

$$a(X) + b(X) = (\Sigma_i a_i X^i) + (\Sigma_i b_i X^i) \equiv \Sigma_i\, (a_i \oplus b_i) X^i \tag{10.20}$$

$$a(X) \bullet b(X) = (\Sigma_i a_i X^i) \bullet (\Sigma_j b_j X^j) \equiv \Sigma_{i,j}\, (a_i \otimes b_j)\, X^{i+j} \quad . \tag{10.21}$$

In the second line, $X^{i+j} = X^i \bullet X^j$ is a product of two matrices X^i and X^i using (10.14), so X^{i+j} then has elements in Z_p. Then $(a_i \otimes b_j)$ is some integer multiple of $X^i X^j$ and we know what that means in $\mathcal{X}_p^m$.

Fact 3: Let R_p^m be the set of polynomials f(X) whose coefficients lie in Z_p and whose variable X lies in $\mathcal{X}_m$. Let the + and • operations for R_p^m be those of $\mathcal{X}_p^m$. Then R_p^m is a ring with identity having an infinite number of elements. (10.22)

Proof: The identity polynomial for R_p^m is just f(X) = I, the identity matrix of $\mathcal{X}_p^m$. One can go through each of the ring definition items exactly as done in Chapter 3 (a) with x → X and one finds that each item verifies.

Fact 4: The rings R_m and $\mathcal{X}_p^m$ are different, although they have the same + and • operations. (10.23)

Proof: R_p^m has an infinite number of elements, while $\mathcal{X}_p^m$ has a finite number of elements p^{m^2}. R_p^m has an infinite number of elements because it contains polynomials of arbitrarily high degree.

Fact 5: The mapping from R_p^m to $\mathcal{X}_p^m$ defined by evaluating a polynomial in R_p^m is many-to-one. (10.24)

Proof: We know each element of R_p^m evaluates to some element of $\mathcal{X}_p^m$ and we know that R_p^m has a lot more elements than $\mathcal{X}_p^m$ has, so the mapping has to be many-to-one.

Fact 6: The ring R_p^1 is exactly the ring R studied above in Chapter 3 (a) if we let x of that section lie in Z_p. (10.25)

(c) The Companion Matrix

In this section we shall explicitly construct forms of the matrix X which satisfy the needs of (10.10).

Let d(x) be some *monic* polynomial of degree m,

$$d(x) = d_0 + d_1 x + ... + d_{m-1} x^{m-1} + x^m. \qquad (10.26)$$

Consider the following mxm matrix :

$$X = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & .. & 0 & -d_0 \\ 1 & 0 & 0 & 0 & 0 & .. & 0 & -d_1 \\ 0 & 1 & 0 & 0 & 0 & .. & 0 & -d_2 \\ 0 & 0 & 1 & 0 & 0 & .. & 0 & -d_3 \\ .. & & & & & & & \\ .. & & & & & & & \\ .. & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & .. & 1 & -d_{m-1} \end{vmatrix} \qquad (10.27)$$

Note that this matrix X is mxm, where m is the degree of d(x). There are 1's just below the main diagonal all the way through the matrix. The rightmost column contains the negatives of all but the leading coefficient of d(x), which is 1, since d(x) was assumed monic. Notice that

$$tr(X) = -d_{m-1} \quad \text{and} \quad det(X) = (-1)^{m-1} (-d_0) = (-1)^m d_0. \qquad (10.28)$$

The X matrix elements can be written using Kronecker deltas. With rows and columns going 1 to m,

$$X_{ij} = \delta_{i,j+1} (1) + \delta_{j,m} (-d_{i-1}) \qquad \text{// for example, } X_{2m} = \delta_{m,m}(-d_1) = -d_1 \qquad (10.29)$$

The $\delta_{i,j+1}$ says that to get the 1, the row i has to be one more than the column j, while the $\delta_{j,m}$ term says that, in the last column, row i gets coefficient $-d_{i-1}$.

This matrix X is called a **companion matrix** of the monic polynomial d(x). We now show that:

Fact 1: The characteristic polynomial of matrix X is precisely d(x). (10.30)

Proof: There are various well-known operations one can perform on rows or columns of a matrix which do not change the determinant, such as adding some multiple of one row to another row. Here then is the basic plan:

start with matrix $M_1 = xI - X$ (10.31)
do $row_{m-1} := row_{m-1} + x \bullet row_m$ to make matrix M_2
do $row_{m-2} := row_{m-2} + x \bullet row_{m-1}$ to make matrix M_3
....
do $row_1 := row_1 + x \bullet row_2$ to make matrix M_m

At each level, we add x times some row to the row above. What this does is cancel out the various x elements of the rows of M_1, and causes the right column to build up to our polynomial f(x). What you end up with is a matrix with an easily computed determinant.

Here is an example for m=4 so $M_1 = xI - X$ is a 4x4 matrix:

$$M_1 = \left|\begin{array}{cccc} x & 0 & 0 & d_0 \\ -1 & x & 0 & d_1 \\ 0 & -1 & x & d_2 \\ 0 & 0 & -1 & d_3 + x \end{array}\right| \quad \text{add x times this row to row above}$$

$$M_2 = \left|\begin{array}{cccc} x & 0 & 0 & d_0 \\ -1 & x & 0 & d_1 \\ 0 & -1 & 0 & d_2 + d_3\, x + x^2 \\ 0 & 0 & -1 & d_3 + x \end{array}\right| \quad \text{add x times this row to row above}$$

$$M_3 = \left|\begin{array}{cccc} x & 0 & 0 & d_0 \\ -1 & 0 & 0 & d_1 + d_2\, x + d_3\, x^2 + x^3 \\ 0 & -1 & 0 & d_2 + d_3\, x + x^2 \\ 0 & 0 & -1 & d_3 + x \end{array}\right| \quad \text{add x times this row to row above}$$

$$M_4 = \left|\begin{array}{cccc} 0 & 0 & 0 & d_0 + d_1\, x + d_2\, x^2 + d_3\, x^3 + x^4 \\ -1 & 0 & 0 & d_1 + d_2\, x + d_3\, x^2 + x^3 \\ 0 & -1 & 0 & d_2 + d_3\, x + x^2 \\ 0 & 0 & -1 & d_3 + x \end{array}\right|$$

At this point, the determinant is

$$(-1)^{4-1}\,[\, d_0 + d_1\, x + d_2\, x^2 + d_3\, x^3 + x^4\,]\, \det(-I_3)$$

$$= (-1)^{4-1}\, d(x)\, \det(-I_3) = (-1)^{4-1}\, d(x)\, (-1)^3 \det(I_3) = (-1)^{4-1}\, d(x)\, (-1)^3 = d(x)\,.$$

In the general case, where X is an mxm matrix,

$$d(x) = \det(xI-X) = \det(M_1) = \det(M_m) = (-1)^{m-1}\, d(x)\, \det(-I_{m-1})$$

$$= (-1)^{m-1}\, d(x)\, (-1)^{m-1} \det(I_{m-1}) = (-1)^{m-1}\, d(x)\, (-1)^{m-1} = d(x)\,. \qquad \textbf{QED}$$

Fact 2: (a) The eigenvalues of companion matrix X are the roots of d(x) (10.32)
(b) The sum of the eigenvalues of companion matrix X is $-d_{m-1}$
(c) The product of the eigenvalues of companion matrix X is $(-1)^m\, d_0$

Proof: (a) By Fact 1 (10.30), d(x) is the characteristic polynomial of matrix X. Thus, the equation d(x) = 0 determines the eigenvalues of X. Thus, the eigenvalues of X are the roots of d(x).
(b) the eigenvalue sum is the trace of X (sum of diagonal elements) which from (10.28) is $-d_{m-1}$

(c) the eigenvalue product of X is det(X) which from (10.28) is $(-1)^m d_0$

Fact 3: The characteristic polynomial d(x) = det(xI - X) remains unaltered if the matrix X is reflected in either or both diagonals. (10.33)

Comment: This is a Fact that would have interested Évariste Galois. We ponder the symmetry group of a physical object in the context of an equation relating to the object.

Proof: (1) Consider det(B). Think of the numbers which comprise matrix B as a square sheet of paper aligned with this page. If this piece of paper is rigidly rotated in any way which leaves it aligned with the page, the determinant is unchanged. These transformations include 180° rotation about the x or y axis, or about either diagonal, or various 90° rotations about an axis perpendicular to the page. The reason these operations leave det(B) unchanged is that these transformations by rotations result in $B' = RBR^{-1}$ where R is a rotation, and $\det(B') = \det(RBR^{-1}) = \det(BR^{-1}R) = \det(B)$, as in (D.6). Matrix B' is how matrix B appears in a frame of reference rotated by R from the frame of reference in which B is observed. A similar statement can be made about any tensor of rank n, and this happens to be the statement for the rank 2 tensor B.

(2) Not all of these rigid transformations leave the identity matrix I unchanged. The following do: rotation about either diagonal, and thus rotation about one diagonal followed by rotation about the other. This last is the same as a perp-to-page rotation by 180°. The diagonal rotations are the same as reflections in the diagonals. The reflection in the main diagonal gives the transpose of the original matrix. The other two transformations give matrices which don't have standard names.

(3) Since the three transformations R just noted leave I intact, we conclude that

$$\det(xI - X) = \det\{ R (xI - X) R^{-1}\} = \det\{ x RIR^{-1} - RXR^{-1}\} = \det\{ (xI - X'\}$$

Thus, these three transforms do not alter the characteristic polynomial. **QED**

Fact 4: Based on Fact 3, we can now write down three alternative forms of the companion matrix. All four forms, including X above, have the same characteristic polynomial: (10.34)

To get X_1, reflect X in the / diagonal:

$$X_1 = \begin{vmatrix} -d_{m-1} & -d_{m-2} & -d_{m-3} & -d_{m-4} & -d_{m-5} & .. & -d_1 & -d_0 \\ 1 & 0 & 0 & 0 & 0 & .. & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & .. & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & .. & 0 & 0 \\ .. & & & & & & & \\ .. & & & & & & & \\ .. & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & .. & 1 & 0 \end{vmatrix} \quad (10.35)$$

To get X_2, reflect X in the \ diagonal (transpose)

$$X_2 = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 & .. & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & .. & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & .. & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & .. & 0 & 0 \\ .. & & & & & & & \\ .. & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & .. & 0 & 1 \\ -d_0 & -d_1 & -d_2 & -d_3 & -d_4 & .. & -d_{m-2} & -d_{m-1} \end{vmatrix} \qquad (10.36)$$

To get X_3, reflect X_2 in the / diagonal:

$$X_3 = \begin{vmatrix} -d_{m-1} & 1 & 0 & 0 & 0 & .. & 0 & 0 \\ -d_{m-2} & 0 & 1 & 0 & 0 & .. & 0 & 0 \\ -d_{m-3} & 0 & 0 & 1 & 0 & .. & 0 & 0 \\ -d_{m-4} & 0 & 0 & 0 & 1 & .. & 0 & 0 \\ .. & & & & & & & \\ .. & & & & & & & \\ -d_1 & 0 & 0 & 0 & 0 & .. & 0 & 1 \\ -d_0 & 0 & 0 & 0 & 0 & .. & 0 & 0 \end{vmatrix} \qquad (10.37)$$

Fact 5: If we *start* with some primitive polynomial d(x) for GF(q), then the companion matrix X of d(x) (or any of its variations) satisfies d(X) = 0. (10.38)

Proof: Fact 1 (10.30) says that d(x) is the characteristic polynomial of such an X. Cayley-Hamilton (10.12) then says that d(X) = 0, which says X solves its own secular equation.

Corollary: X (or any variation) is then a viable candidate for X in Fact 3 (10.10). (10.39)

We have just proven this restatement of (10.10):

Big Fact 6: In order to construct a matrix representation for GF(q), first select a primitive polynomial d(x) for GF(q) from Fig 5.3 or elsewhere. Then select for primitive element matrix X any of the companion matrix forms shown above which contain the coefficients of d(x).
Then: the GF(q) representation is GF(q) = { 0, I, X, X^2, X^3, ... X^{q-2}}. (10.40)

If we use the first form of X given in (10.27), we may construct X in this simple manner from (10.29),

$$X_{ij} = \delta_{i,j+1} (1) + \delta_{j,m} (-d_{i-1}) \quad \text{where } d(x) = \Sigma_{k=0}^{m} d_k x^k . \qquad (10.41)$$

Example: For GF(2^3) a primitive polynomial from Fig 5.2 is d(x) = x^2+x+1 so d_2=d_1=d_0=1. Then

$$X_{ij} = \delta_{i,j+1} (1) + \delta_{j,m} (-1) = \delta_{i,j+1} - \delta_{j,m} .$$

(d) Maple program to construct the matrix representation of GF(p^m)

The code is self-documented. The p,m values selected in this code are for GF(2^3).

```
with(linalg): # invoke the linear algebra library
```

Procedure to replace elements of mxm matrix M with mod p elements

```
> matrixp := proc(M::array,m,p)
    local i,j;
    for i from 1 to m do
      for j from 1 to m do
        M[i,j] := M[i,j] mod p;
      od;
    od;
    evalm(M);
  end:
```

Enter p, m and the coefficients of a primitive polynomial

```
>   p := 2: m := 3: d[2] := 0: d[1] := 1: d[0] := 1:
> # p := 3: m := 2:            d[1] := 1 :d[0] := 2:
```

Cosmetic labels for output

```
> Xpow := seq(X^k,k=1..p^m-1):
```

Construct the primitive element matrix X using (10.41) and verify the characteristic polynomial

```
> X_ := matrix(m,m):
> for i from 1 to m do
    for j from 1 to m do
      X_[i,j] := delta(i,j+1) - delta(j,m)*d[i-1] mod p;
    od;
  od;
> charpoly(X_,x) mod p;
```

$$x^3 + x + 1$$

Display the entire matrix representation for GF(p^m)

```
>
> for k from 1 to p^m-1 do
     print(Xpow[k],matrixp(evalm(X_^k),m,p));
  od;
  print("0 " ,matrix(m,m,[0$m^2]));
```

Example: GF(2^3) This example uses the numbers shown in the above code. The primitive polynomial d(x) = 1 + x + x^3 is the same one used in the example below (6.10). Here are the 8 matrices of our matrix representation of GF(2^3),

$$X,\begin{bmatrix}0&0&1\\1&0&1\\0&1&0\end{bmatrix}\quad X^3,\begin{bmatrix}1&0&1\\1&1&1\\0&1&1\end{bmatrix}\quad X^5,\begin{bmatrix}1&1&1\\1&0&0\\1&1&0\end{bmatrix}\quad X^7,\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$
$$X^2,\begin{bmatrix}0&1&0\\0&1&1\\1&0&1\end{bmatrix}\quad X^4,\begin{bmatrix}0&1&1\\1&1&0\\1&1&1\end{bmatrix}\quad X^6,\begin{bmatrix}1&1&0\\0&0&1\\1&0&0\end{bmatrix}\quad \text{"0"},\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}$$

As expected, X^7 = I and I does not appear as some lesser power of X. There are $2^{3^2} = 2^9 = 512$ possible matrices containing 1's and 0's (the ring $\mathcal{X}_2{}^3$), and we have used 8 of them to represent GF(2^3). Since GF(2^3) has q-1 = 7 which is prime, any power of X can also serve as the primitive element, and then the matrices above get reordered. Also, the primitive element X can be replaced by any of its reflected versions as noted above in (10.35-37), which set includes the simple transpose of X.

Example: GF(3^2) The primitive polynomial f(x) = x^2 + x +2 is the same one quoted below (6.15). Here are the 9 matrices of our matrix representation of GF(3^2),

$$X,\begin{bmatrix}0&1\\1&2\end{bmatrix}\quad X^4,\begin{bmatrix}2&0\\0&2\end{bmatrix}\quad X^7,\begin{bmatrix}1&1\\1&0\end{bmatrix}$$
$$X^2,\begin{bmatrix}1&2\\2&2\end{bmatrix}\quad X^5,\begin{bmatrix}0&2\\2&1\end{bmatrix}\quad X^8,\begin{bmatrix}1&0\\0&1\end{bmatrix}$$
$$X^3,\begin{bmatrix}2&2\\2&0\end{bmatrix}\quad X^6,\begin{bmatrix}2&1\\1&1\end{bmatrix}\quad \text{"0"},\begin{bmatrix}0&0\\0&0\end{bmatrix}$$

As expected, X^8 = I and I does not appear as some lesser power of X. Notice that X^4 = 2I which is not the same as I. There are $3^{2^2} = 81$ possible matrices containing 0,1,2 (the ring $\mathcal{X}_3{}^2$), and we have used 9 of them to represent GF(3^2).

Example: GF(2^9) For this example we use the Fig 5.3 primitive polynomial $x^9 + x^4 + 1$:

```
m := 9: d[8] := 0: d[7] := 0: d[6] := 0: d[5] := 0:
p := 2: d[4] := 1: d[3] := 0: d[2] := 0: d[1] := 0:d[0] := 1:

charpoly(X_,x) mod p;
```

$$1 + x^4 + x^9$$

In this case brute force calculation should give $X^{511} = I$. Maple cheerfully does the job:

```
print(Xpow[1],matrixp(evalm(X_^1),m,p));
print(Xpow[511],matrixp(evalm(X_^511),m,p));
```

$$X, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad X^{511}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Furthermore, we can verify that the identity matrix does not occur anywhere along the way. Here we compare X^k to I for k = 1 to 512 and take note of any X^k that equals I:

```
ID := array(1..m,1..m,identity):
for k from 1 to p^m do
  if equal( matrixp(evalm(X_^k),m,p),ID) then print("yes", k) fi;
od;
```

"yes", 511

One could use this code as a *very* inefficient way to search for primitive polynomials.

Appendix A: Proof of Fact 10 (4.29)

Theorem: The order of any cyclic subgroup in {GF(q)-0,•} divides the exponent e.

This proof is a fleshed-out version of a 3-column-inch proof given by Bobrow and Arbib, their Lemma 7 of Chapter 8. We include this proof because it is the essence of the proof that GF(q) is cyclic, which is the single most important fact about GF(q). Facts 3,4 and 5 from our Chapter 4 will be used:

Fact 3: Let $g \in \{GF(q) - 0, \bullet\}$. If "g has order n" , then $g^n = 1$ and n is the least integer for which $g^n = 1$. *(4.18)*

Fact 4: Let $g \in \{GF(q) - 0, \bullet\}$. If $g^k = 1$, the "order of g" divides k. . *(4.19)*

Fact 5: If β is any element of the cyclic subgroup (α,n,•), then $\beta^n = 1$. *(4.20)*

Recall that the exponent e is defined to be the order of the largest cyclic subgroup in {GF(q)-0,•}. Let e' be the order of *some other* subgroup (α,e',•). We want to show that e' divides e.

We will assume that e' does *not* divide e and show that there is a contradiction. The proof has many steps, so they will be numbered. Warning: the proof is a bit tedious.

Step 1. If e' does not divide e, then e and e' can be written as follows:

$e = Rn$	$R = p^r$	$s > r \geq 0$	$S>R>1$	r,s,m,n are integers
$e' = Sm$	$S = p^s$	p = some prime number		p does not divide n or m

Proof. Expand both e and e' in terms of powers of primes using the Factorization Theorem (1.42). The p_i are the primes in some standard order, and the exponents are all integers ≥ 0.

$e = (p_1)^{r_1} (p_2)^{r_2} (p_3)^{r_3} \ldots$

$e' = (p_1)^{r_1'} (p_2)^{r_2'} (p_3)^{r_3'} \ldots$

If e' divides e, then we know that $r_i \geq r_i'$ for all i. In other words, we have to have e' divides e in each prime power component. Therefore, if e' does *not* divide e, there must be at least one prime where this is not true. Thus, let p = this p_i, let $r_i = r$ and $r_i' = s$, so that s > r, and r ≥ 0 since it is an exponent. Write:

$e = (p)^r \bullet$ [all the other primes to their powers] = p^r [n] = Rn $\quad R \equiv p^r$

s>r S>R

$e' = (p)^s \bullet$ [all the other primes to their powers] = p^s [m] = Sm $\quad S \equiv p^s$

The bracketed quantities are of course integers which do not contain any powers of the prime p.

Step 2. Consider G = (g,e,•) and G' = (g',e',•). That is, g is a generator of G of order e, and g' is a generator of G' of order e'. Then construct these two elements:

$g_1 = (g)^R$	in G	Claim that order of g_1= n.	$(g_1, n, \bullet) = G_1$
$g_2 = (g')^m$	in G'	Claim that order of g_2 = S.	$(g_2, S, \bullet) = G_2$

Proof: Consider these facts:

$$(g_1)^n = (g)^{nR} = (g)^e = 1$$
$$(g_2)^S = (g')^{Sm} = (g')^{e'} = 1$$

In each line, the last equality is due to fact that e and e' are the orders of the two subgroups G and G'. This is an application of Fact 5. Since $(g_1)^n = 1$, according to Fact 4 we know that the order of g_1 divides n. So we write both:

$$(g_1)^n = 1 \quad \Rightarrow \quad \text{(order of } g_1\text{) divides n} \quad \Rightarrow \quad \text{(order of } g_1\text{)} = n/d_1$$
$$(g_2)^S = 1 \quad \Rightarrow \quad \text{(order of } g_2\text{) divides S} \quad \Rightarrow \quad \text{(order of } g_2\text{)} = S/d_2 \ .$$

We now argue that $d_1 = d_2 = 1$. If this is not so, then for example get:

$$(g_1)^{n/d_1} = (g)^{nR/d_1} = (g)^{e/d_1} = 1$$
$$(g_2)^{S/d_2} = (g')^{Sm/d_2} = (g')^{e'/d_2} = 1$$

The top last equality here says (order of g) divides (e/d_1) for $d_1>1$. This implies that (order of g) < e. This is a contradiction since we assumed at the start that (order of g) = e. Thus $d_1 = 1$. Similarly for the lower line. Therefore:

$$d_1 = 1 \quad \Rightarrow \quad \text{(order of } g_1\text{)} = n$$
$$d_2 = 1 \quad \Rightarrow \quad \text{(order of } g_2\text{)} = S$$

So now we know the order of our two constructed elements g_1 and g_2.

Step 3. Now construct element $g_3 = g_1 g_2$. Let d = (order of g_3) as shorthand, so we have $G_3 = (g_3, d, \bullet)$.
Claim: 1. $(g_3)^d = 1$
2. d divides nS.

Proof: (1) This follows from Fact 3.
(2) Using above facts, we find that:

$$(g_3)^{nS} = [(g_1)^n]^S [(g_2)^S]^n = 1^S 1^n = 1 \ .$$

Therefore, from Fact 4, we know that (order of g_3) divides nS, so d divides nS. This result will again be used at the very end of this proof.

Step 4. Define $h = (g_1)^d$. Claim that h lies in both G_1 and G_2 .

Proof: We know that $(g_3)^d = 1$. Therefore, $(g_1^d)(g_2^d) = 1$. Therefore $(g_1^d) = (g_2^d)^{-1}$, the inverse element of (g_2^d). But the inverse of any element in G_2 is in G_2, since it is a group. Thus, $h = (g_1^d)$ lies in G_2. Of course it also lies in G_1 since it is a power of g_1.

Step 5: Now consider the subgroup (h, I, •) generated by h, and let I be its order. We claim that

I divides n
I divides S

Proof: Since h is in G_1, we know from Fact 5 that $h^n = 1$ since n is the order of G_1. Since I is the order of h, we know from Fact 4 that I divides n. Similarly, since h is in G_2, we know from Fact 5 that $h^S = 1$ since S is the order of G_2. Since I is the order of h, we know from Fact 4 that I divides S.

Step 6: Claim the following sequence of results:

1. I = 1
2. h = 1
3. $(g_1^d) = 1$ and $(g_2^d) = 1$.
4. n divides d
5. S divides d

Proof: (1) We know that $S = p^s$ for some prime p, and that n has no powers of p. This means that GCD(n,S) = 1, n and S are "relatively prime". If a number divides into both n and S, that number must be 1. Therefore, from Step 5, I = 1. Thus, our subgroup (h, I, •) has order I = 1; (2) I=1 means that $h^1 = 1$, or simply h = 1. (3) Since $(g_1^d)(g_2^d) = 1$ and $h = (g_1)^d$, the last results follows. (4) Since order of g_1 is n, and since $(g_1^d) = 1$, we know from Fact 4 that n divides d. (5) Since order of g_2 is S, and since $(g_2^d) = 1$, we know from Fact 4 that S divides d.

Step 7: Claim that nS divides d.

Proof: We have already noted that n and S are relatively prime. Thus, if these two numbers both divide into a third number, that third number must be a multiple of nS. From Step 6's 4 and 5 above, we know that n and S each divide d, and therefore d must be a multiple of nS. Thus, nS divides d.

Step 8: Claim that d = nS.

Proof: From Step 3 above, we learned that d divides nS. From Step 7 we learned that nS divides d. The conclusion is that d = nS.

Step 9: This leads to the contradiction that d > e, and therefore e' divides e.

Proof: Recall that d is the order of field element g_3 in $G_3 = (g_3, d, •)$. We have just shown that d = nS. Since S > R, we know that d > nR. But nR = e, so we have d > e. This is a contradiction because the order of any subgroup is supposed to be less than e -- this was how e was defined.

Recap of the above proof

The idea is, based on the assumption that e' does not divide e, to identify some subgroup which has order greater than e, and this is then a contradiction since e is supposed to be the maximal subgroup order. The too-large subgroup is G_3 of order d generated by $g_3 = g_1 g_2$. The proof is really finished at Step 7 where it is found that nS divides d, so that $d \geq nS > nR = e$.

Step 1: Relate (e = order of G) and (e' = order of G') to the four integers R, S, n, m.
Step 2A: Let G = (g,e,•), define $g_1 = g^R$, show order of $g_1 = n$, so define $G_1 = (g_1, n, \bullet)$
Step 2B: Let G' = (g',e',•), define $g_2 = g'^m$, show order of $g_2 = S$, so define $G_2 = (g_2, S, \bullet)$
Step 3: Consider $G_3 = (g_3, d, \bullet)$, where $g_3 = g_1 g_2$, order d. Since $(g_3)^{nS} = 1$, find that d divides nS.

Step 4: Let $h = g_1^{d'}$ order I. Show that $h \in G_1$ and $h \in G_2$
Step 5: I divides n and S
Step 6: I = 1, so $h = g_1^d = 1$; n and S divide d
Step 7: nS divides d

Step 8: d = nS
Step 9: d > e, which is contradiction.
Therefore, e' divides e.
Therefore, the order of any cyclic subgroup in {GF(q)-0,•} must divide e.

Appendix B: The Nature of the Conjugate Set of α

Overview: We show first that, in general, a conjugate sequence of elements (B.1) of some element α of GF(q) maps into a sequence of elements in a certain cyclic set (B.2) in a repeating cycle. This "mapping" is simply a rewriting of the conjugate elements as their lowest equivalent powers of α. If this cycle repeats just one element as the conjugate elements are mapped left to right, then of course not all the conjugate elements are distinct. Most of this Appendix is concerned with showing the nature of the possible repeating cycle. At the very end we then show that, if α is a primitive element of GF(q), then there is no repeat at all, and thus all m conjugates of α are distinct. Then if α is not a primitive element, it is possible to get some repeats, and then not all m elements of the conjugate set are distinct.

Here we wish to examine what happens as one keeps raising the conjugate exponent in the series

$$\{ \alpha, \alpha^{p}, \alpha^{p^2}, \alpha^{p^3} \quad , ... \alpha^{p^{m-1}} \} \qquad \text{conjugate sequence, m elements} \tag{B.1}$$

It is useful to think for the moment of m being very large, so there are many terms to worry about.

According to Fact 2 (4.17), any such α is a generator of *some* cyclic group of GF(q) of *some* order n. Thus, we write

$$\{1, \alpha, \alpha^2, \alpha^3, ... \alpha^{n-1} \} \qquad \text{"the landing zone", n elements} \quad \alpha^n = 1 \quad \alpha^{n+1} = \alpha \tag{B.2}$$

Let's call n the **repeat index** of this cyclic subgroup, since $\alpha^n = 1$.

Imagine that this set (B.2) is relatively small, while the set of conjugates (B.1) is relatively large. As we step sequentially through the conjugates, each one maps onto (hits, lands on, is equal to) some element in this little cyclic group (B.2). This is entirely controlled by the fact that $\alpha^n = \alpha$. In fact, here is how the mapping works:

$$\text{exponent of } \alpha \text{ in the landing zone for } \alpha^{p^k} = \text{Rem}(p^k/n) \equiv r_k \, . \tag{B.3}$$

so the mapping from the conjugate sequence to the landing zone is

$$\alpha^{p^k} \rightarrow \alpha^{r_k} \qquad \text{which means} \qquad \alpha^{p^k} = \alpha^{r_k} \, . \tag{B.4}$$

Think of the conjugates of (B.1) being loaded one at a time into a cannon which fires them into the landing zone. Each conjugate lands somewhere in that landing zone.

In (B.3) we are just taking the conjugate's exponent and figuring it modulo n. As the conjugate sequence starts off with k = 0,1,2... we hit the elements α^{r_0}, α^{r_1}, α^{r_2}.... in the landing zone. Since $r_0 = 1$, the first element hit in the landing zone is α, but the sequence of landing spots after that is dependent on the values of p and n according to (B.3). Some landing spots might never be hit at all.

Notice that all the r_k would vanish if n were a power of p, in which case the cannonballs would all land on element $\alpha^0 = 1$ in the landing zone, meaning all conjugates are 1. We now show that this never happens.

Fact: Order n is not a power of p. (B.5)

Proof: The cyclic group shown in (B.2) has n elements and is certainly a subgroup of { GF(q) - 0} which has q-1 elements. From (1.9d) then we know that n must divide evenly into q-1 = p^m - 1. Suppose n were some power of p, n = p^k. This value of n does not divide evenly into p^m - 1 and in fact there is always a remainder of -1. Thus, the order n cannot be a power of p.

Let us now assume there is some minimum k = k_0 > 0 such that r_{k0} = Rem(p^{k0}/n) = 1. This k_0 would mark the second cannonball that lands on α (the first was marked with k = 0). For example, if n = 5 and p = 2, we get k_0 = 4. If there is no k_0 that is less than m (the max number of conjugates), then α is hit only once by the limited set of conjugates shown above. So assume k_0 does exist.

Fact: We now claim this to be true: Rem[$\frac{p^{k_0+s}}{n}$] = r_s, where r_s is as in (B.3). (B.6)

We will prove this below, but first, what is this saying? Well, for k = 0,1,2...k_0-1, our α^{p^k} land on $\alpha^{r_0} = \alpha, \alpha^{r_1}, \alpha^{r_2}, \ldots \alpha^{r_{k0-1}}$ in the landing zone, according to (B.4). Then when k=k_0 we land on $\alpha^{r_{k0}} = \alpha^1 = \alpha$ again. Then as s increments and we have k = k_0 + s for s = 1,2,... and then our α^{p^k} land on $\alpha^{r_1}, \alpha^{r_2}, \ldots$ all over again, a second time! This is what that above Facts says. So if we have a large number of conjugates (B.1) and a small cyclic group (B.2), this says that as the conjugates are mapped in sequence, they are sprayed into the landing zone in a sequence of landing spots that repeats again and again in the same order. As noted above, this sequence might never hit one or more of the landing spots.

Proof: Apply Little Lemma 2 (1.30) ,

$$\text{Rem}[\frac{xy}{n}] = \text{Rem}[\frac{y\ \text{Rem}(x/n)}{n}] \qquad \textit{(1.30a)}$$

with x = p^{k_0} and y = p^s and xy = p^{k_0+s} to get,

$$\text{Rem}[\frac{p^{k_0+s}}{n}] = \text{Rem}[\frac{p^s\ \text{Rem}(p^{k_0}/n)}{n}] = \text{Rem}[\frac{p^s\ 1}{n}] = r_s \ . \qquad \textbf{QED}$$

Now consider this enhanced version of the previous Fact:

Fact: Rem[$\frac{p^{Nk_0+s}}{n}$] = r_s, the same r_s numbers given above, for N = 0,1,2... (B.7)

If we have a large number of conjugates (B.1) and a small cyclic group (B.2), this Fact says that as the conjugates are mapped in sequence, they are sprayed into the landing zone in a sequence of landing spots that repeats again and again in the same order! The first spray sequence has N = 0 and then the Fact just restates (B.3). The second spray sequence has N = 1, which we studied in (B.6). Then N = 2,3,4.. are

further repeats of the same spray sequence. As noted above, this sequence might never hit one or more of the landing spots.

Proof: We prove this by induction. Assume it is true for some N (we know N=0 and N=1 work), and show true for N+1. Our proof makes two uses of Little Lemma 2 (1.30). In the first use we have

$$\text{Rem}[\frac{xy}{n}] = \text{Rem}[\frac{y\ \text{Rem}(x/n)}{n}] \qquad \textit{(1.30a)}$$

with $x = p^{Nk_0+s}$ and $y = p^{k_0}$ and $xy = p^{(N+1)k_0+s}$ to get

$$\text{Rem}[\frac{p^{(N+1)k_0+s}}{n}] = \text{Rem}[\frac{p^{k_0}\text{Rem}(p^{Nk_0+s}/n)}{n}]$$

$$= \text{Rem}[\frac{p^{k_0}\ r_s}{n}] \qquad \text{// induction proof assumes Fact (B.7) is valid for N = N}$$

$$= \text{Rem}[\frac{r_s\ \text{Rem}[p^{k_0}/n]}{n}] \qquad \text{// this is (1.30a) applied again with } x = p^{k_0} \text{ and } y = r_s$$

$$= \text{Rem}(r_s 1\ /n) \qquad \text{// from Fact (B.6)}$$

$$= r_s \qquad \text{// Rem}(r_s/n) = r_s$$

Thus we have shown that $\text{Rem}[\frac{p^{(N+1)k_0+s}}{n}] = r_s$ which is our Fact (B.7) for N+1. **QED**

So the question of whether all m conjugates are distinct in the original set boils down to the question of whether there exists a k_0 such that $\text{Rem}(p^{k_0}/n) = 1$, where $k_0 < m$. This in turn depends on p and on the order n of the cyclic subgroup which contains element α. If there *is* some $k_0 < m$, then we get some number of "repeat hits" in the landing zone, which means that some of the conjugate set elements are identical.

If α is a *primitive* element of GF(q), then we know n = q-1 because such an α is a generator of {GF(q)-0} and since this group is cyclic, all group elements can be enumerated as powers of α, see (4.31). In this case, it is easy to see that all m conjugates in the sequence (B.1) are distinct, because there is no $k_0 < m$ such that $\text{Rem}(p^{k_0}/n) = 1$ since,

$$\text{Rem}[\frac{p^{k_0}}{n}] = \text{Rem}[\frac{p^{k_0}}{q-1}] = \text{Rem}[\frac{p^{k_0}}{p^m-1}] = p^{k_0} \neq 1 \ .$$

For the last steps, since $k_0 < m$, the fraction $\frac{p^{k_0}}{p^m-1}$ is a proper fraction, so the remainder is p^{k_0}. Since we assume $p \geq 2$ and $k_0 > 0$, we cannot have $p^{k_0} = 1$. Thus we have proven:

Fact: If α is a primitive element of GF(q), then all m conjugates in the conjugate set of α are distinct. Conversely, if α is not a primitive element, it is possible that only the first k_0 of the conjugates are in fact distinct, for some $k_0 < m$. (B.8)

Appendix C: Evaluation of a(x)b(x)

Consider this product of two polynomials

$$c(x) = a(x)\, b(x) \tag{C.1}$$

where

$$\begin{aligned} a(x) &= \Sigma_{i=0}^{A}\, a_i x^i \\ b(x) &= \Sigma_{j=0}^{B}\, b_j x^j \end{aligned} \tag{C.2}$$

and where a_i and b_j lie in Z_p whose operations are $\otimes$ and $\oplus$.

Fact: We will show that

$$c(x) = \Sigma_{s=0}^{A+B}\, c_s\, x^s \qquad \text{where } c_s = \sum_{j\,=\,\max(0,s-A)}^{\min(s,\,B)} (a_{s-j} \otimes b_j) \ . \tag{C.3}$$

Proof: Inserting the expansions (C.2) we get

$$\begin{aligned} a(x)\, b(x) &= (\Sigma_{i=0}^{A}\, a_i x^i)\, (\Sigma_{j=0}^{B}\, b_j x^j) \\ &= \Sigma_{j=0}^{B}\, \Sigma_{i=0}^{A}\, (a_i \otimes b_j)\, x^{i+j} \ . \end{aligned} \tag{C.4}$$

Let $s \equiv i+j$ to get

$$\begin{aligned} &= \Sigma_{j=0}^{B}\, \Sigma_{s-j=0}^{A}\, (a_{s-j} \otimes b_j)\, x^s \\ &= \Sigma_{j=0}^{B}\, \Sigma_{s=0}^{A+j}\, (a_{s-j} \otimes b_j)\, x^s \end{aligned} \tag{C.5}$$

We can see that max(s) = A+B, so $s \le A+B$. If we define a function θ(Boolean) such that $\theta = 1$ if the Boolean is true and $\theta = 0$ if it is false, then we are free to insert a factor $\theta(s \le A+B)$ into our sum to get

$$= \Sigma_{j=0}^{B}\, \Sigma_{s=0}^{A+j}\, \theta(s \le A+B)\, (a_{s-j} \otimes b_j)\, x^s \ . \tag{C.6}$$

We can use this same θ function to make explicit the bounds of the subscripts on a_{s-j} and b_j

$$\begin{aligned} a_{s-j} &= \{\theta(s-j \ge 0)\, \theta(s-j \le A)\}\, a_{s-j} \ = \ \{\theta(s \ge j)\, \theta(s \le A+j)\}\, a_{s-j} \\ b_j &= \{\theta(j \ge 0)\, \theta(j \le B)\}\, b_j \end{aligned} \tag{C.7}$$

then we have

$$a(x)\, b(x) = \Sigma_{j=0}^{B}\, \Sigma_{s=0}^{A+j}\, \{\theta(s \ge j)\, \theta(s \le A+j)\}\, \{\theta(j \ge 0)\, \theta(j \le B)\}\, \theta(s \le A+B)\, (a_{s-j} \otimes b_j)\, x^s \ .$$

Since the upper limits of the j and s sums are now pinned by θ functions $\theta(j \le B)$ and $\theta(s \le A+j)$, we can formally raise both these endpoints to infinity. Having done that, we are then free to swap the order of the two summations, since each has fixed endpoints,

$$a(x)\, b(x) = \Sigma_{j=0}^{\infty}\ \Sigma_{s=0}^{\infty}\ \{\theta(s \ge j)\ \theta(s \le A+j)\}\{\theta(j \ge 0)\ \theta(j \le B)\}\ \theta(s \le A+B)\ (a_{s-j} \otimes b_j)\ x^s$$

$$= \Sigma_{s=0}^{\infty}\ \Sigma_{j=0}^{\infty}\ \{\theta(s \ge j)\ \theta(s \le A+j)\}\{\theta(j \ge 0)\ \theta(j \le B)\}\ \theta(s \le A+B)\ (a_{s-j} \otimes b_j)\ x^s$$

$$= \Sigma_{s=0}^{\infty}\ \theta(s \le A+B)\ \ \Sigma_{j=0}^{\infty}\ \{\theta(s \ge j)\ \theta(s \le A+j)\}\{\theta(j \ge 0)\ \theta(j \le B)\}\ (a_{s-j} \otimes b_j)\ x^s$$

$$= \Sigma_{s=0}^{A+B}\ \Sigma_{j=0}^{\infty}\ \{\theta(s \ge j)\ \theta(s \le A+j)\}\{\theta(j \ge 0)\ \theta(j \le B)\}\ (a_{s-j} \otimes b_j)\ x^s \quad . \qquad \text{(C.8)}$$

Now the four remaining θ functions can be regarded as limits on the j sum. They say :

$$j \le s \quad \text{and} \quad j \le B \qquad \Rightarrow \qquad j \le \min(s,B)$$

$$j \ge s-A \ \text{ and } j \ge 0 \qquad \Rightarrow \qquad j \ge \max(0,s-A) \qquad \text{(C.9)}$$

Thus we have

$$a(x)\, b(x) = \Sigma_{s=0}^{A+B}\ \{\ \sum_{j\,=\,\max(0,s-A)}^{\min(s,\,B)} (a_{s-j} \otimes b_j)\ \}\ x^s\ =\ \Sigma_{s=0}^{A+B}\ c_s\ x^s \qquad \text{(C.10)}$$

where

$$c_s\ =\ \sum_{j\,=\,\max(0,s-A)}^{\min(s,\,B)} (a_{s-j} \otimes b_j) \quad . \qquad \text{(C.11)}$$

QED

Appendix D: A Small Collection of Matrix Facts

If the reader is anything like the author, all these facts are familiar but don't suffer from being refreshed from time to time. The reader may consult any book on matrices to find proofs of things unproven here.

All the following facts deal with square matrices. We use an mxm matrix A as our prototype. It is of course assumed that the reader has a basic knowledge of matrices, such as how to multiply them and how to compute a determinant.

It is assumed in this section that the elements of the matrix A are just "numbers", meaning complex numbers. More generally, the matrix elements are elements of some field F. In our application in Chapter 10, that field will be Z_p= GF(p).

It is amazing how the theory of matrices seems to infiltrate every field of human interest in which there is some attempt to calculate something. There are probably a thousand "facts" about matrices which could be listed in an exhaustive treatise, which this small review is not.

Convention: For the definitions which follow, we shall assume that the "first" row or column or a matrix is labeled by 1. Labels increase top to bottom for rows, and left to right for columns.

If A is a matrix, and its elements are A_{ij} , the first index i is the **row index**, and the second j is the **column index**. Thus, the first row of this matrix has elements A_{11}, A_{12}, A_{13}, ... The row index is constant across a row, and serves to define a row. Similarly for the column index.

The **transpose** of a matrix A we shall call A^T. It is a new matrix formed by interchanging the rows and columns of A. We have then $(A^T)_{ij} = A_{ji}$. (D.1)

A **tensor of rank n** is a "matrix" with n subscripts instead of 2. For example, A_{ijk} is a rank-3 tensor. A matrix is a rank-2 tensor, a vector is a rank-1 tensor. This is a simplification of the meaning of a tensor; for detail on this subject see the author's reference on tensor analysis or elsewhere. (D.2)

The completely antisymmetric tensor $\varepsilon_{abcde\ldots}$ of rank n is defined as follows. If the subscripts are an even permutation of 123...n, it is (+1). If the subscripts are an odd permutation, it is (-1). If the subscripts are not a permutation, this means some subscripts are repeats, and in this case it is (0). This object is totally antisymmetric because it changes sign when any two indices are swapped. It is also called **the permutation tensor**. (D.3)

Comment: The actual tensor nature of $\varepsilon_{abcde\ldots}$ is an interesting subject, see the author's tensor analysis reference, Section 7 (h) and Appendix D. The $\varepsilon_{abcde\ldots}$ object is a tensor density of weight -1.

Example : The rank 6 totally antisymmetric tensor. Some sample elements:

$\varepsilon_{123456} = 1$ $\varepsilon_{132456} = -1$ $\varepsilon_{132465} = 1$ $\varepsilon_{123345} = 0$

The **determinant** of an mxm matrix A is indicated by det(A) or |A| and is defined as:

$$\det(A) = \varepsilon_{abcde\ldots}A_{1a}A_{2b}A_{3c}A_{4d}\ldots \quad (= \varepsilon_{abcde\ldots}A_{a1}A_{b2}A_{c3}A_{d4}\ldots) \tag{D.4}$$

where ε is the totally antisymmetric tensor of rank m defined above. In the above formula, each repeated index is implicitly summed from 1 to m. There are a total of m such indices, hence there are a total of m factors of A.

This form of the determinant is convenient for making some points below. It is assumed that the reader is familiar with the more normal "recursive" method of computing a determinant in terms of sub-determinants by starting with any row or column.

Example: Determinant of a 2x2 matrix:

$$\det(A) = \varepsilon_{ab}A_{1a}A_{2b} = \varepsilon_{12}A_{11}A_{22} + \varepsilon_{21}A_{12}A_{21} = A_{11}A_{22} - A_{12}A_{21}$$

Fact 1: Adding a multiple of one row to another row does not change the determinant of a matrix, although it certainly changes the matrix itself. The same applies for columns. (D.5)

Fact 2: $\det(A) = \det(A^T)$
$\det(AB) = \det(A)\det(B)$
$\det(ABC) = \det(A)\det(B)\det(C) = \det(BCA) = \det(CAB)$
$\det(RR^{-1}) = \det(1) = 1 = \det(R)\det(R^{-1})$ (D.6)

Fact 3: If A is an mxm matrix, and α is a constant, then $\det(\alpha A) = \alpha^m \det(A)$. (D.7)

If A is a square matrix, then the **cofactor** of a matrix element A_{ij} is denoted by $[\mathrm{cof}(A)]_{ij}$ and is equal to $(-1)^{i+j}$ times the determinant of the submatrix obtained by crossing out the row and column which contains A_{ij}. Thus, the cofactor is just some number. Since there is one cofactor for each element of a matrix, we can combine all these cofactors into a new matrix called the **cofactor matrix** cof(A).

Fact 4: The inverse of a square matrix A is given by

$$A^{-1} = [\mathrm{cof}(A)]^T / \det(A) \quad (D.8)$$

Corollary 4: The inverse A^{-1} exists if and only if $\det(A) \neq 0$. (D.9)

If $\det(A) = 0$ so this A^{-1} does not exist, A is said to be **singular**.

The matrix $[\mathrm{cof}(A)]^T$ is sometimes called the **adjoint** matrix of A and denoted adj(A). That is , the adjoint is the transpose of the cofactor matrix of A. We will not use this adjoint terminology.

The **trace** of a square matrix A is denoted tr(A) and is defined as the sum of the diagonal elements. (D.10)

Peculiar Fact 5: If A is a square matrix, then $\det(e^A) = e^{\mathrm{tr}(A)}$. (D.11)

The **rank** of an mxm matrix A is denoted r(A) and is the number of linearly independent rows or columns. It is also the size of the largest non-zero sub-determinant. The size m of a square matrix is called its **order**. The quantity (order - rank) is called the **degeneracy** or the **nullity** of A, denoted by n(A). Thus, $n(A) = m - r(A)$. (D.12)

If $A\mathbf{x} = \mathbf{0}$, the space of solutions **x**, called the **nullspace** of A, has dimension n(A). If matrix A has full rank m, then $n(A) = m-m = 0$ and the nullspace contains only the trivial element $\mathbf{x} = \mathbf{0}$. This is the case in which the inverse A^{-1} exists and of course then $\mathbf{x} = A^{-1}\mathbf{0} = \mathbf{0}$. (D.13)

The rank of a matrix has nothing to do with the rank of a tensor mentioned above. It just happens that the same word is used for both concepts.

Corollary 5: If an mxm matrix A is invertible, it has full rank m. (D.14)

Proof: From Fact 4, A invertible means $\det(A) \neq 0$, which means rank = m.

Peculiar Fact 6: Consider the equation C = AB for three square matrices. It turns out that $n(c) \geq n(a),n(b)$ but $n(c) \leq n(a) + n(b)$. This "triangle rule" is called **Sylvester's Law of Nullity**. (D.15)

The **characteristic polynomial** d(x) of a square matrix A is

$$d(x) \equiv \det(xI - A). \quad (D.16)$$

Here, I is the identity matrix.

Fact 7: The characteristic polynomial of an mxm matrix A is of degree m. Two of the coefficients of d(x) are known at once: $d_0 = (-1)^m \det(A)$, and $d_m = 1$. From this last, d(x) is a monic polynomial, meaning the coefficient of the highest power is 1. (D.17)

Proof: From the expression (D.4) given for a determinant, it is clear that one term in the determinant is the product of the diagonal elements of the matrix with a (+1) coefficient. If the matrix is (xI-A), then this term has the form $(x - A_{11}) (x-A_{22})\ldots. (x - A_{mm}) = x^m + \ldots .$ This shows that the degree is m and the leading coefficient is $d_m = 1$. As for the constant term, $d(0) = d_0 = \det(0I-A) = \det(-A) = (-1)^m \det(A)$.

The equation $\det(xI - A) = 0$ [$d(x) = 0$] is called the **characteristic equation** or sometimes the **secular equation**. The roots of the characteristic polynomial are thus the solutions to the secular equation. These roots/solutions are called **eigenvalues** and are often denoted λ_i. Thus, $d(\lambda_i) = 0$ for any eigenvalue λ_i. The eigenvalues of A are the solutions to its secular equation. (D.18)

Comment: The characteristic equation arises in the calculation of long-term variations in the motion of the planets. Such long term variations are called "secular motions". OED2 gives these distinct definitions of the word secular: I. Of or pertaining to the world (as opposed to the spiritual realm); II Of or pertaining to an age or long period.

The significance of eigenvalues is that one often encounters problems in which one is solving for a vector **v** in the equation

$$A\mathbf{v} = \lambda\mathbf{v} \; . \qquad \text{(D.19)}$$

Since $\lambda\mathbf{v} = \lambda I\mathbf{v}$, by writing the above as $(\lambda I - A)\mathbf{v} = 0$ one sees that if $(\lambda I - A)$ has an inverse, the equation has only the trivial solution $\mathbf{v} = 0$, since then $\mathbf{v} = (\lambda I - A)^{-1}\, 0 = 0$. However, if $\lambda = \lambda_i$, a root of $d(\lambda)$, then

$$d(\lambda_i) = 0 = \det(\lambda_i I - A) \qquad \Rightarrow \qquad (\lambda_i I - A) \text{ has no inverse according to (D.9)}$$

In this case, there might be (and usually is) some non-zero solution $\mathbf{v}_i$ corresponding to eigenvalue λ_i . If this is the case, then $\mathbf{v}_i$ is called the **eigenvector** corresponding to the eigenvalue λ_i. (D.20)

Comment: If the vector space of interest is an infinite dimensional space of normalizable functions on some interval (a,b), the eigenvectors are called **eigenfunctions** since the vectors in such a space are functions. For example, the Legendre polynomials $P_n(z)$ for n = 0 to ∞ form basis vectors on (-1,1).

Example: In quantum mechanics, a matrix of interest is called the Hamiltonian H, the eigenvectors **v** are called stationary states ψ, the eigenvalues λ_i are the energies E_i of these states, and $A\mathbf{v} = \lambda\mathbf{v}$ is written $H\psi = E\psi$ and is called the time-independent Schrodinger equation. In most quantum mechanics problems, the vector space is a function space and the eigenvectors are eigenfunctions called wavefunctions.

Fact 8: The trace of a matrix tr(A) is equal to the sum of its eigenvalues. Recall that the trace was defined as being the sum of the diagonal elements of A. (D.21)

Fact 9: The determinant of a matrix det(A) is equal to the product of its eigenvalues. (D.22)

A **triangular** matrix is one which has nothing but zeros on one side of the diagonal. (D.23)

Fact 10: The eigenvalues of a triangular matrix are its diagonal elements. (D.24)

Proof: It is easy to show this by the usual method of recursively computing a determinant.

Corollary 10: The eigenvalues of a diagonal matrix are its diagonal elements. (D.25)

Example: Let A = I, the identity matrix. Then $d(x) = \det(xI - I) = \det[(x-1)I] = (x-1)^m \det(I) = (x-1)^m$. There are m roots of d(x) , they are all 1. This matrix has m eigenvalues equal to 1. These are its diagonal elements.

Fact 11: If any eigenvalue of an mxm matrix A is 0, then A is singular. (D.26)

Proof: (D.22) says then that det(A) = 0 which says A is singular.

A **similarity transformation** of a square matrix A is defined by

$$A' = S\,A\,S^{-1} \tag{D.27}$$

where S is any square matrix such that S^{-1} exists and thus $\det(S) \neq 0$. One says that A and A' are **similar**. If we define $Q \equiv S^{-1}$, then $A' = Q^{-1}AQ$, another form of a similarity transformation.

Observation: Saying that A and A' are similar is like saying that a rotated set of axes (x',y',z') is similar to an unrotated set (x,y,z). Generally speaking, a similarity transformation does not change the nature of what is going on, it just changes how things are labeled. It is a change of basis. The following fact should impress upon the reader how little a similarity transformation changes the nature of a matrix.

Fact 12: Similar matrices have the same determinant, the same trace, the same rank, the same characteristic polynomial, and the same eigenvalues. In addition, any equation involving matrices retains its same form after application of a similarity transformation. Such an equation is said to be covariant. (D.28)

Example: If AB=C, then apply S .. S^{-1} to get $(S\,A\,S^{-1})(S\,B\,S^{-1}) = (S\,C\,S^{-1})$ or A'B' = C' .

Comment: There are certain interesting classes of matrices (Hermitian, real symmetric) which can be brought to diagonal form by a similarity transformation (unitary, real orthogonal). If one can find this similarity transformation, then one at once knows all the eigenvalues of the original matrix, since they are just the diagonal elements of the transformed matrix and since eigenvalues are preserved under any similarity transformation. The process of bringing a matrix to diagonal form is called **diagonalization**.

A notable absence from this Appendix is the Cayley-Hamilton theorem. Since this plays such a key role in Chapter 10, it is stated there in (10.12).

Appendix E: Existence of g(x) which divides $x^n - 1$.

In our definition (8.1) of a cyclic code, we required that generator g(x) divide $x^n - 1$ where n is the length of the (n,k) code. It was not required that n be the period of g(x), so n might not be the smallest s for which g(x) divides $x^s - 1$. Nor was it required that g(x) be irreducible. In particular, g(x) need not be the primitive polynomial of some Galois Field.

The purpose of this Appendix is to demonstrate that for any positive integer n which is not a multiple of the p of GF(p), there does indeed exist at least one g(x) in ring R which divides $x^n - 1$. It happens that this g(x) is irreducible with respect to GF(p). Recall that ring R [Chap 3 (a)] is just the set of polynomials which have coefficients in GF(p). When applied to GF(2), this says that for any odd integer n, an irreducible g(x) exists in R over GF(2) which divides $x^n - 1$. Our demonstration does not say precisely what the degree of that existent g(x) is, though it could be figured out. The degree is $\leq \varphi(n)$ as we shall see. This totient function $\varphi(n)$ is defined and studied in Appendix G.

The degree of g(x) is n-k. It would perhaps be nice if one could find a g(x) having *any* desired degree less than the selected value of n, the code length. One might wonder whether for the special and practical case GF(2) this might even be possible. This would lead to an (n,k) cyclic code for any n and k < n.

Consider this little Maple program:

```
n := 8;
for N from 1 to 2^(n+1)-1 do   # go only as far as needed
 # compute polynomial for this value of N
  c := convert(N,base,2):
  p := sum(c[i]*x^(i-1),i=1..nops(c)):
# see if it has a remainder
  r := Rem(x^n-1,p,x) mod 2;
  if type(r,numeric) and r = 0 then print("N=",N,"degree=",nops(c)-1,"p(x)=",p, r) fi;
od:
```

You enter a value of n for $(x^n - 1)$ and it tries *all possible* polynomials in search of a g(x) that divides $x^n - 1$. These polynomials are generated by setting N = 1.2.3,... and converting each integer N into binary and regarding that bit pattern as the coefficients of a polynomial in R over GF(2). Here is an encouraging sample run with n = 8 as shown:

$$n := 8$$

"N=", 1, "degree=", 0, "p(x)=", 1, 0

"N=", 3, "degree=", 1, "p(x)=", $1 + x$, 0

"N=", 5, "degree=", 2, "p(x)=", $1 + x^2$, 0

"N=", 15, "degree=", 3, "p(x)=", $1 + x + x^2 + x^3$, 0

"N=", 17, "degree=", 4, "p(x)=", $1 + x^4$, 0

"N=", 51, "degree=", 5, "p(x)=", $1 + x + x^4 + x^5$, 0

"N=", 85, "degree=", 6, "p(x)=", $1 + x^2 + x^4 + x^6$, 0

"N=", 255, "degree=", 7, "p(x)=", $1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7$, 0

"N=", 257, "degree=", 8, "p(x)=", $x^8 + 1$, 0

For this value of n, we find a viable g(x) of every degree less than n. Viable just means we could create a cyclic code based on that g(x) (the case g(x) = 1 excluded).

Unfortunately, this is true for some values of n and not true for others. There is doubtless some theorem that explains which values of n work and which don't. Here are some values of n which are found not to have g(x) of every possible degree: n = 5,7,9,10,14,15. Some of these are prime, some not, some even, some odd. The run for n = 9 looks like this

$$n := 9$$

"N=", 1, "degree=", 0, "p(x)=", 1, 0

"N=", 3, "degree=", 1, "p(x)=", $1 + x$, 0

"N=", 7, "degree=", 2, "p(x)=", $1 + x + x^2$, 0

"N=", 9, "degree=", 3, "p(x)=", $1 + x^3$, 0

"N=", 73, "degree=", 6, "p(x)=", $1 + x^3 + x^6$, 0

"N=", 219, "degree=", 7, "p(x)=", $1 + x + x^3 + x^4 + x^6 + x^7$, 0

"N=", 511, "degree=", 8, "p(x)=", $1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8$, 0

"N=", 513, "degree=", 9, "p(x)=", $x^9 + 1$, 0

so that no g(x) exists of degree 4 or 5. The reason a degree is missing can be analyzed by writing the equation g(x)h(x) = x^n-1 where g and h are allowed arbitrary coefficients a,b,c.... For bad values of n, one is led to a set of equations for these coefficients which have no solution. For example, one might end up with the requirement that b + b + 1 = 0 and no b in GF(2) solves that equation.

With the above as introduction, we now start into a set of Facts, many of which are quite interesting in their own right. They lead to the final conclusion that some g(x) which divides x^n-1 does exist.

Fact 1: For any integer n not a multiple of prime p, one can find integer m such that n is a divisor of (p^m - 1). That is, we can find m such that (p^m-1) mod n = 0 which is the same as p^m mod n = 1. (E.1)

Proof: Since we have assumed n is not a multiple of p, and that p is a prime number, we know that GCD(p,n) = 1, so p and n are coprime (relatively prime). After all, p cannot divide n, and nothing smaller than p can divide p but 1.

Suppose n > p. Euler's Theorem (G.17) states that, for any element "a" of Mod(n) = Z_n that is coprime to n, there exists an integer φ(n) such that $a^{\varphi(n)}$ mod n = 1. The integer φ(n) is just the number of integers < n that are coprime to n. Since our p < n lies in Z_n and is coprime to n, we can say $p^{\varphi(n)}$ mod n = 1. Thus, we have found an integer m = φ(n) which solves the problem.

Suppose n < p. From (G.11) we know that GCD(p mod n, n) = GCD(p,n) = 1. Thus, p' ≡ p mod m is an element of Z_n which is coprime with n, so we can apply Euler's Theorem to get $(p')^{\varphi(n)}$ mod n = 1. But (1.31c) says

(p*p*p + ...) mod n = ([p mod n] * [p mod n] * [p mod n] + ...) mod n = (p'*p'*p' + ...) mod n

which then says $(p)^{\varphi(n)}$ mod n = $(p')^{\varphi(n)}$ mod n. Thus, $(p)^{\varphi(n)}$ mod n = 1 and we are done. **QED**

Verification: Here we check the above Fact for the first 20 prime numbers and all n < 1000:

```
primes := seq(ithprime(i),i=1..20);
```

primes := 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71

```
found := false:
for p in [primes] do
  for n from 1 to 1000 do
    if n mod p = 0 then next; fi;   #ignore n being multiple of p
    m := phi(n);
    r := p^m-1 mod n;
    if r <> 0 then found := true; print("exception found", p, m, n) ;fi;
  od;
od;
if found = false then print("no exceptions were found"); fi;
```

"no exceptions were found"

Fact 2: If integer D is a divisor of q-1, then GF(q) has at least one element of order D. (E.2)

Proof: We know from (4.21) that [order(α^k)] = (q-1)/GCD(k,q-1) where α is any primitive element of GF(q) and k any positive integer. Since D is a divisor of q-1 we can write DM = q-1 where M is the other factor. M then also divides q-1. Setting k = M,

[order(α^M)] = (q-1)/GCD(M,q-1) .

But GCD(M,q-1) = M. The reason is that M is a candidate GCD since it divides M and q-1, and no larger integer can divide M, so M is it. Thus we have shown that [order(α^M)] = (q-1)/M = D. Thus we have found an element of GF(q), namely α^M , which has order D, for any D among the divisors of (q-1).

Example: For GF(2^4) we have q-1 = 15 which has divisors 1,3,5 and 15. If α is a primitive element, then we know $\alpha^{15} = 1$ since such an element has order q-1.

D	1	3	5	15
M	15	5	3	1
$\beta^{order}=1$	$(\alpha^{15})^1= 1$	$(\alpha^5)^3= 1$	$(\alpha^3)^5= 1$	$(\alpha^1)^{15}= 1$
order β	1	3	5	15

Here an element β of GF(q) has been found for each divisor of q-1. For D = 1, β = 1 and $1^1 = 1$ so the identity always has an order to match D = 1. For D = 15 we just get the statement that α is a primitive element $\alpha^{15}= 1$. It is the interior columns that are important.

Fact 3: If integer D is a divisor of q -1, then GF(q) has at least one minimum polynomial of period D. (E.3)

Proof: From Fact 2, we know that GF(p^m) has at least one element (call it α) of order D. This element has a minimum polynomial m(x) of the form (x-α)(other factors). From (5.47) we know that the period of a minimum polynomial is the same as the order of the elements in its conjugate set (here including α), so since α has order D, there exists m(x) of period D. **QED**

Example: Recall this enumeration (6.21) of the minimum polynomials of GF(2^4), where we have added the two trivial minimum polynomials shown in (5.24) which are always present :

$p_1(x)$	$= (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8)$	$= x^4 + x + 1$	10011
$p_7(x)$	$= (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{13})(x - \alpha^{11})$	$= x^4 + x^3 + 1$	11001
$m_3(x)$	$= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^9)$	$= x^4 + x^3 + x^2 + x + 1$	11111
$m_5(x)$	$= (x - \alpha^5)(x - \alpha^{10})$	$= x^2 + x + 1$	111
$m_0(x)$	$= (x - \alpha^0) = (x-1)$;		
m_{zero}	$= (x - 0) = x$;		

(6.21)

We shall now compute the period of each. First the polynomials are entered:

```
p1 := x^4+x+1:
p7 := x^4+x^3+1:
m3 := x^4+x^3+x^2+x+1:
m5 := x^2+x+1:
m0 := x-1:
mzero := x:
```

The following procedure scans n downward to find the period of polynomial f(x) :

```
period := proc(f,q,p) local per, n;
  per := infinity;
  for n from q by -1 to 1 do if Rem(x^n-1,f,x) mod p = 0 then per := n; fi; od;
  RETURN(per);
end:
```

And here are the resulting periods:

```
period(p1,15,2);
                        15
period(p7,15,2);
                        15
period(m3,15,2);
                         5
period(m5,15,2);
                         3
period(m0,15,2);
                         1
period(mzero,15,2);
                        ∞
```

As claimed in the Fact, there exists a minimum polynomial of period D for each D that divides p^m-1. The function x can never divide any $x^n - 1$ so the procedure sets period = ∞.

Fact 4: For any n not a multiple of p, there exists a g(x) which divides $x^n - 1$ and which can therefore be used to construct a cyclic code of length n. This g(x) is irreducible in GF(p). (E.4)

Proof: To find a suitable g(x) with coefficients in GF(p), we search through extension fields $GF(p^m)$ for m = 1,2,3 ... . We know from Fact 3 that, for any divisor D of $p^m - 1$, $GF(p^m)$ has a minimum polynomial m(x) of period D, so that m(x) divides x^D-1. If we can find a value of m such that $p^m - 1$ has a divisor D=n then m(x) divides $x^n - 1$ and this m(x) is a viable g(x). But Fact 1 says that we *can* find m such that n is a divisor of $p^m - 1$, namely, m = φ(n). Thus, the field $GF(p^{\varphi(n)})$ has a suitable minimum polynomial that can be used for g(x).

Corollary: For any n not a multiple of p one can construct a cyclic code of length n whose polynomials have coefficients in GF(p). (E.5)

Reminder: It is likely that there are many viable g(x) that divide x^n-1 and there will be such viable g(x) even when n *is* a multiple of p as our examples with GF(2) earlier show (see n = 8 case). It just happens that the g(x) we find here in our existence proof happens to be a minimum polynomial of some Galois extension field GF(q) over GF(p). This g(x) is of course irreducible in GF(p), and may or may not be a primitive polynomial of GF(q).

Appendix F: Cyclic Code Error Detection Theorems (CRC)

We shall identify the term Cyclic Redundancy Check (CRC) with any error detection system based on an (n,k) cyclic code. The Facts presented in this section are all stated and proved in the original 1961 CRC paper by Peterson and Brown noted in References. The notation used in their paper is the same as ours except for the following differences:

	us	CRC paper
generator	g(x)	P(X)
data polynomial	D(x)	G(X)
code polynomial	C(x)	F(X)
semantics	g(x) has period n	P(X) belongs to exponent n

Our proofs are only slightly more general in some cases than those of the paper which deals only with GF(2). We always assume the code length is n and number of data symbols is k.

Our definition of a cyclic code includes item (8.1) (b) requiring that g(x) divides $x^n - 1$. If this requirement is ignored, then a "cyclic code" so obtained won't really be cyclic because the proof of Chap 8 (f) fails. Also, the set of code words won't form an ideal in the ring A_n, and much of the theory of cyclic codes falls apart. Still, the claims below still apply if g(x) does not divide $x^n - 1$. However, in order to detect all double-bit errors, g(x) must divide some $x^s - 1$ for some $s \geq n$.

Fact 1: If $g(0) \neq 0$, a cyclic code generated by g(x) over GF(p) detects all single-symbol errors. (F.1)

Proof. Suppose C'(x) and C(x) differ in one symbol position. Then $E(x) \equiv C'(x) - C(x) = \alpha x^i$ where i = 0,1..n-1 and α is some non-zero element of GF(p) = Z_p. We need to show that in this case the syndrome s(x) will be non-zero so the error will then be detected. But :

$$s(x) = \text{Rem}[C'(x)/g(x)] = \text{Rem}[\{C(x) + E(x)\}/g(x)] = \text{Rem}[\{D(x)g(x) + E(x)\}/g(x)]$$

$$= \text{Rem}[E(x)/g(x)] = \text{Rem}[\alpha\, x^i/g(x)]$$

Now:

- For i = 0, $\text{Rem}[\alpha x^i/g(x)] = \text{Rem}[\alpha/g(x)] = \alpha \neq 0$. Note that $g(0) \neq 0$ rules out g(x) = constant ≠ 0.
- For i > 1, $\text{Rem}[\alpha x^i/g(x)] \neq 0$ unless $g(x) = \alpha x^n$ for $n \leq i$, but $g(0) \neq 0$ rules out such a g(x).

Thus, $s(x) \neq 0$ and the error is detected. Notice that it was not required that g(x) divide evenly into $x^n - 1$.

Fact 2: If g(x) = (x-1)f(x), a cyclic code generated by g(x) over GF(2) detects all odd-number-bit errors. (F.2)

Comment: x-1 is the same as x+1 for GF(2) since -1 = +1 in GF(2).

Proof: In this case, E(x) = C'(x) - C(x) is a polynomial with an odd number of terms. We must show that in this case the syndrome s(x) does not vanish so the error will be detected. But:

$s(x) = \text{Rem}[E(x)/g(x)] = \text{Rem}[(\text{poly with odd of terms})/g(x)]$.

Assume that this remainder is 0. Then we would have some quotient $q(x) = E(x)/g(x)$ and

$\text{poly with odd of terms}(x) = g(x)\,q(x) = (x-1)f(x)\,q(x)$.

Evaluate at x = 1 to get

odd sum of 1's $= 1 = (1-1)f(1)q(1) = 0$

Since $1 \neq 0$ we have a contradiction, so it must be that $s(x) \neq 0$ and the error is detected.

Notice that it was not required that g(x) divide evenly into $x^n - 1$.

Fact 3: If $g(0) \neq 0$ and g(x) has period $\geq n$, a cyclic code generated by g(x) over GF(2) detects all double-bit errors and all single-bit errors. (F.3)

Proof: We already know from Fact 1 that single-bit errors are detected. For double-bit errors we have $E(x) = x^i + x^j$ for some $0 \leq i \leq j \leq n-1$ We need to show that the syndrome does not vanish in this case.

$s(x) = \text{Rem}[E(x)/g(x)] = \text{Rem}[(x^i+x^j)/g(x)] = \text{Rem}[x^i(x^{j-i}+1)/g(x)]$.

Since $g(0) \neq 0$, we know that g(x) has no factors of x so the x^i cancels nothing in g(x), so the only way for s(x) to be 0 is if

$\text{Rem}[(x^{j-i}+1)/g(x)] = 0$

which we rewrite in GF(2) as

$\text{Rem}[(x^{j-i} - 1)/g(x)] = 0$. (*)

The maximum value of j-i is (n-1)-(0) = n-1. If the period of g(x) is $\geq n$, then (*) cannot be true, since the period t of g(x) is the smallest t such that (x^t-1) is divisible by g(x). Then we get $s(x) \neq 0$ and the double-bit error will be detected.

Notice that it was not required that g(x) divide evenly into $x^n - 1$. However, it *was* required that g(x) have a period $\geq n$, the length of the code.

Fact 4: If g(0) ≠ 0 *and* g(x) = (x-1)f(x) *and* g(x) has period ≥ n, a cyclic code generated by g(x) over GF(2) detects all single-bit, double-bit, triple-bit and all other odd-number-bit errors. (F.4)

Proof: Such a g(x) meets the criteria for Facts 1,2 and 3 so we can take the union of the detections of each.

Definition: A **burst error** of length (extent) r means any number of symbol errors occurring within a *sequential* set of powers x^i to x^{i+r-1}. For example, if a,b,c,d are non-zero elements of GF(p), the error pattern $E(x) = ax^3 + bx^5 + cx^9$ would be the error polynomial for a burst error of length r = 7. Other E(x) of length 7: $ax^4 + bx^{10}$, $ax^2 + bx^4 + cx^6 + dx^8$. For GF(2) all coefficients are of course 1.

Fact 5: If g(0) ≠ 0, a cyclic code generated by g(x) over GF(p) detects all burst errors of length n-k or smaller. (F.5)

Proof: The error pattern can be written $E(x) = x^i f(x)$ where f(x) has degree r-1 < n-k and f(x) ≠ 0. The syndrome is

$$s(x) = \text{Rem}[E(x)/g(x)] = \text{Rem}[x^i f(x)/g(x)] .$$

As before, x^i cancels nothing in g(x) if g(0) ≠ 0, so the only way to get s(x) = 0 is if

$$\text{Rem}[f(x)/g(x)] = 0 .$$

But since g(x) has degree n-k and f(x) has degree < n-k, this remainder is just f(x) ≠ 0. **QED**

Notice that it was not required that g(x) divide evenly into $x^n - 1$.

Fact 6: The cyclic code of Fact 5 detects all burst errors of length r ≤ n-k. It does not detect all burst errors for r > n-k, but statistically it can detect a lot of them. For GF(2), here are the conclusions:

fraction of burst errors NOT detected for r = (n-k) + 1 = $1/2^{(n-k-1)}$

fraction of burst errors NOT detected for r > (n-k) + 1 = $1/2^{(n-k)}$. (F.6)

For example, if n-k = 32 as in CRC-32, then $1/2^{(n-k-1)} = 1/2^{31} \sim 10^{-9}$. For such a code, only one burst error out of a billion goes undetected even if the burst extent is the entire length-n packet! This means that almost all multi-symbol errors of any arrangement will be detected. And *all* burst errors of extent 32 or less are detected.

Proof: See Theorem 6 of the original Peterson and Brown CRC paper appearing in the References.

Appendix G: GCD, mod n, totient φ(n), Euler Theorem, Fermat's Little Theorem

Here we take a brief tour of "number theory" with the goal being to arrive at the endpoints of Euler's Theorem (Fact 11) and Fermat's Little Theorem (Fact 12). More trees in the forest. The entire thread presented here (excluding a few references to items in the main document) is completely self-contained, and each Fact has a proof. In our selected pathway, every Fact in this section is required to get the train to the station. No doubt there are shorter paths, but all these Facts are worth stating and proving. If the GCD and Mod concepts are old hat to you, all these Facts will all be familiar.

Euler's Theorem is used in Appendix E to prove (E.1).

The two Facts 13 and 14 show how the number of primitive elements and the number of primitive polynomials of a Galois Field are related to the totient function φ defined below in (G.15). The final Fact 15 examines the smallest-x solution of the Diophantine equation ax = bx.

Fact 0: These fundamental modulo arithmetic rules were stated and derived in (1.31c) :

$$(x+y+z+\ldots) \bmod n = ([x \bmod n] + [y \bmod n] + [z \bmod n] + \ldots) \bmod n$$

$$(x*y*z+\ldots) \bmod n = ([x \bmod n] * [y \bmod n] * [z \bmod n] + \ldots) \bmod n \qquad \textit{(1.31c)}$$

Fact 1: Let d = GCD(c,m). Then by the meaning of "common divisor" there must exist integers N_1 and N_2 such that $c/d = N_1$ and $m/d = N_2$. The fact claims that $GCD(N_1,N_2) = 1$. (G.1)

Proof: Suppose $GCD(N_1,N_2) = K > 1$. Then there must exist integers M_1 and M_2 such that

$N_1/K = M_1$	$\Rightarrow N_1 = M_1K$	$\Rightarrow c/d = M_1K$	$\Rightarrow c/(dK) = M_1$
$N_2/K = M_2$	$\Rightarrow N_2 = M_2K$	$\Rightarrow m/d = M_2K$	$\Rightarrow m(dK) = M_2$.

The equations on the right show that then dK > d is a common divisor of c and m which is a contradiction since d is supposed to be the largest common divisor. **QED**

Fact 2: Suppose $GCD(N_1,N_2) = 1$ and $N_1A = N_2B$. Then N_2 divides A and N_1 divides B. (G.2)

Proof:

- If $N_1 = 1$ then $A = N_2B$ so N_2 divides A and of course $N_1 = 1$ divides B.
- If $N_2 = 1$ then $B = N_1A$ so N_1 divides B and of course $N_2 = 1$ divides A.
- If $N_1>1$ and $N_2> 1$ and if $N_1A = N_2B$, then N_2 divides N_1A (quotient B). But if $GCD(N_1,N_2) = 1$, N_2 cannot divide N_1 because if it did then $N_2 > 1$ would be a common divisor of N_1 and N_2. Thus, if N_2 divides N_1A then N_2 must be dividing A. Similarly, N_1 must divide B.

Notation: The notation a = b mod m is a *shorthand* for a mod m = b mod m, or (a-b) mod m = 0. It means that a and b are congruent mod m. The notation is a little misleading since it seems to say for example 100 = 100 mod 4 = 0 so 100 = 0, but that is not what it means. It means that 100 is congruent with 0 mod 4. A better notation might be a = b (mod m), but that adds clutter. Note then that for a and b integers in Z, there exists an integer I such that

$$a = b \bmod m \Leftrightarrow a \bmod m = b \bmod m \Leftrightarrow (a-b) \bmod m = 0 \Leftrightarrow (a-b) = I\,m \Leftrightarrow a = b + I\,m \qquad (G.3)$$

Fact 3: $ca = cb \bmod m \Leftrightarrow a = b \bmod \frac{m}{GCD(c,m)}$ (G.4)

Proof: Let $d \equiv GCD(c,m)$. As in **Fact 1**, there exist N_1 and N_2 such that

$$c = N_1 d$$

$$m = N_2 d \Rightarrow \frac{m}{GCD(c,m)} = \frac{m}{d} = N_2 = \text{an integer (which is promising!)}$$

Proof in the $\Rightarrow$ direction:

$$ca = cb \bmod m \Rightarrow ca = cb + em \Rightarrow c(a-b) = em \Rightarrow N_1 d(a-b) = em \Rightarrow a-b = \frac{em}{N_1 d} = \frac{e}{N_1}\frac{m}{d}.$$

If we can show that $\frac{e}{N_1}$ is an integer, then we have shown that $a = b \bmod \frac{m}{d}$ as claimed. Go back to

$$N_1 d(a-b) = em \Rightarrow N_1(a-b) = e\frac{m}{d} \Rightarrow N_1(a-b) = N_2\, e\,.$$

Since $GCD(N_1,N_2) = 1$ by **Fact 1**, **Fact 2** says that N_2 divides (a-b). Now write the last equation as

$$\frac{e}{N_1} = \frac{a-b}{N_2}$$

Since we just showed that N_2 divides a-b, we have shown that $\frac{e}{N_1}$ is an integer. **QED**

Proof in the $\Leftarrow$ direction:

If $a = b \bmod N_2$ then $a = b + kN_2$ and $ca = cb + ckN_2$. But

$$ckN_2 = (N_1 d)k\frac{m}{d} = (N_1 k)m$$

Thus $ca = cb + (N_1 k)m$ and so $ca = cb \bmod m$. **QED**

Corollary 3. If GCD(c,m) = 1 then [ca = cb mod m $\Leftrightarrow$ a = b mod m] . (G.5)

In this special case of **Fact 3**, if we see ca = cb, we can divide both sides by c to get a = b mod m.

Fact 4: (Linear Congruence Theorem).
If ax = b mod m and GCD(a,m) = 1, there is a unique mod-m solution x. (G.6)

Comment: If a = b mod m, then a and b are said to be **congruent** mod m. The set of integers a which are congruent to b mod n is called a **congruence class**. The set of such values was called a residue class in our residue class ring discussion of Chap 1 (c), with an example shown in (1.32). In this Fact our equation to be solved is ax = b which is a linear equation and ax is congruent to b mod m, hence the theorem name.

History: An equation of two of more variables, like $ax^2 + by + cy^3 + dz = e$, in which all coefficients and variables are restricted to be integers, is called a **Diophantine equation**, named after a Greek algebra guy Diophantus circa 250 BC. An equation like ax + by = c is a linear Diophantine equation. We saw an example of this in (1.43), Bezout's Identity $d = x\, n_1 + y\, n_2$ where $d = GCD(n_1,n_2)$ and we want to solve for integers x and y. In our current Fact 4, we have ax = b mod m which is a "linear Diophantine equation mod m". Hilbert's 10th Problem (1900) was to find an algorithm which could determine whether a given Diophantine equation has a solution or not. The work of several people from 1944 to 1970 showed that such an algorithm does not exist for the general case.

Proof of Fact 4: We want to solve ax = b mod m for x. Consider the linear Diophantine equation

ay - mz = 1 1 = GCD(a,m) variables y,z

Since this is the **Bezout Identity (1.43)**, we know there exists an integer solution for y and z. So we have a specific value of y. Multiply by b to get

a(by) = b + m(bz) => a(by) = b mod m

Thus our solution is x = by. Suppose there were another solution x'. Then

ax = b mod m
ax' = b mod m => ax = ax' mod m

Since GCD(a,m) = 1, **Corollary 3** says that x = x' mod m. Thus our solution x = by is the only solution mod m. **QED**

Corollary 4: Element a of Z_m has an inverse if GCD(a,m) = 1. (G.7)

Proof: In **Fact 4** we showed that in this case ax = b mod m had a unique solution. Setting b = 1 we find that ax = 1 has a unique solution. But that solution is a^{-1}. [Z_m means Mod(m, +, •)].

Fact 5: Given a,b > 0, and if x and y exist such that ax+by = 1, then GCD(a,b) = 1. (G.8)

Proof: Suppose GCD(a,b) = Q > 1. Then

$a = QN_1$
$b = QN_2 \quad \Rightarrow \quad 1 = ax+by = (QN_1)x + (QN_2)y = Q(N_1x + N_2y) \Rightarrow 1/Q = N_1x + N_2y$.

But since Q > 1, this says fraction = integer which is a contradiction, so Q = 1.

Fact 6: If GCD(a,m) = 1 and GCD(b,m) = 1, then GCD(ab,m) = 1. (G.9)

Proof: From the if conditions we know from **Bezout's Identity (1.43)** that x,y,x',y' exist such that

$1 = ax + my$ and $1 = bx' + my'$.

Thus,

$1 = (ax + my)(bx' + my') = ab(xx') + m(ybx'+axy'+myy') = ab\,x'' + m\,y''$.

From **Fact 5**, since $(ab)\,x'' + m\,y'' = 1$ we conclude that GCD(ab,m) = 1.

Fact 7: For any integer I, GCD(a + Im, m) = GCD(a,m). (G.10)

Proof: (1) Let s be some common divisor of a and m. Clearly s divides a+Im. Thus, s is a common divisor of the pair a+Im and m. So

$$\{ \text{common divisors of a and m} \} \subset \{ \text{common divisors of a+Im and m} \} . \qquad (*)$$

(2) Let s be some common divisor of a+Im and m. Does s divide a? To find out, consider

$a+Im = s\,N_1$
$m = s\,N_2$

$$N_1 = \frac{a+Im}{s} = \frac{a}{s} + I\frac{m}{s} = \frac{a}{s} + I\,N_2 \qquad \Rightarrow \qquad \frac{a}{s} = N_1 - I\,N_2$$

Thus, $\frac{a}{s}$ is an integer so yes, s does divide a. Thus, s is a common divisor of m and a. So

$$\{ \text{common divisors of a+Im and m} \} \subset \{ \text{common divisors of a and m} \} \qquad (**)$$

(3) We conclude from (*) and (**) that the two common divisor sets are the same set, call it Q.

Suppose this set is Q = {1, 5, 17}. Then GCD(a + Im, m) = 17 and GCD(a,m) = 17. In general, we find that both GCD(a + Im, m) and GCD(a,m) will be the largest element of Q, so GCD(a + Im, m) = GCD(a,m). **QED**

Fact 8: GCD(a mod m, m) = GCD(a,m) (G.11)

Proof: We know that a mod m = a + Im for some integer I (I < 0 if a > m). Thus from **Fact 7** we have that GCD(a mod m, m) = GCD(a + Im, m) = GCD(a,m). **QED**

Fact 9: For a,b in Z_m (with + and •), GCD(a•b, m) = GCD(ab mod m, m) = GCD(ab,m). (G.12)

Proof: The meaning of a•b is that this product is some element c within Z_m which is closed under •. The rule for finding this element c is c = ab mod m, so a•b = ab mod m. Then the last equality shown follows from **Fact 8**.

Definition: Integers n and m are **coprime** (relatively prime) iff GCD(n,m) = 1. (G.13)

Definition: The set of integers < n which are coprime with n are called the **totatives** of n.
We shall refer to this set as G_n in what follows. (G.14)

Example: The totatives of n = 12 are {1,5,7,11} .

Definition: **Euler's totient function** φ(n) is the number of totatives of n. (G.15)

Examples: φ(12) = 4 since the set {1,5,7,11} has four elements.
φ(7) = 6 since the set {1,2,3,4,5,6} has 6 elements.
φ(p) = p-1 if p is prime

Comment: It happens that the sum of the totatives of n is (n/2)φ(n), but we shall not prove it since we don't need it. In our example with n = 12, 1+5+7+11 = 24 and (n/2)φ(n) = 6*φ(12) = 6*4 = 24.

Here are the first 100 values of φ(n) in the form

Fact 10: The set G_n of the totatives of n forms an abelian group under • within Z_n of order φ(n). (G.16)

Proof: The properties of a group are shown in (1.1). We know that G is closed under • from **Fact 6** which reads " if a and b are in G_n, then ab is in G_n ". We know from **(1.29)** that Z_n forms an abelian ring with identity, so any subset of Z_n which includes 1 has the commutative, associative, and identity-exists properties for operator •. What is missing is the existence of a • inverse. When n is a prime p, we showed such an inverse always exists and Z_p is then a field GF(p). But for general Z_n some inverses do not exist.

However, for our set G_n inverses always exist due to **Corollary 4** above. Thus G_n is a group of order φ(n).

Example: $Z_8 = \{0,1,2,3,4,5,6,7\} = \text{Mod}(8,+,\bullet)$

$G_8 = \{1,3,5,7\}$ with • operation of Z_8. Note that φ(8) = 4.

$3 \bullet 3 = 9 \bmod 8 = 1 \in G_8$ $\quad 3^{-1} = 1$ $\quad 5 \bullet 5 = 15 \bmod 8 = 7 \in G_8$ $\quad 5^{-1} = 7$

$3 \bullet 5 = 15 \bmod 8 = 7 \in G_8$ $\quad 5 \bullet 7 = 35 \bmod 8 = 3 \in G_8$

$3 \bullet 7 = 21 \bmod 8 = 3 \in G_8$ $\quad 7 \bullet 7 = 49 \bmod 8 = 1 \in G_8$ $\quad 7^{-1} = 1$

Thus, G_8 is closed under • and all elements have an inverse.

Fact 11: (Euler's Theorem). For any a in G_n, $a^{\varphi(n)} = 1 \bmod n$. (G.17)

Proof: **Fact (1.9e)** says that for any a in group G, $a^n = 1$ where n is the order of the group. What that means here is a•a•a...•a = 1 where we have φ(n) factors of a since from **Fact 10** the order of G_n is φ(n). The operation • is that of Z_n of which G_n is a subset. For elements in Z_n we know that a•b= ab mod n. Thus, a•a•a...•a $= a^{\varphi(n)} \bmod n$ if we interpret $a^{\varphi(n)}$ as aaaa..a (integer multiplication). Thus, we have Euler's Theorem:

$a \bullet a \bullet a ... \bullet a = a^{\varphi(n)} \bmod n = 1$ or $a^{\varphi(n)} = 1 \bmod n$ // see (G.3)

This theorem was proved by Euler in 1763. The theorem and the totient function φ(n) play a major role in current day RSA public key cryptography.

Fact 12: (Fermat's Little Theorem). If p is prime and a is any integer, then $a^p = a \bmod p$. (G.18)

Proof: For n = prime p, $G_p = \{1,2,3, ... p-1\} = Z_p$ and φ(p) = p-1. **Fact 11** then says $a^{p-1} = 1 \bmod p$ for any a in G_p. Since any a in G_p and p are coprime, meaning GCD(a,p) =1, we can apply **Corollary 3** and multiply both sides of $a^{p-1} = 1 \bmod p$ by a to get $a^p = a \bmod p$ for any a in $G_p = Z_p$. We can then extend the theorem to any integer a using **Fact 0** to say,

$a^p \bmod p = \{[a \bmod p] * [a \bmod p] ...\} \bmod p = [a \bmod p]^p \bmod p = 1$

where the last equality follows since a mod p lies in G_p. There is no restriction to positive integers, since every negative integer is congruent to a positive integer, such as (-19) mod 7 = (-5) mod 7 = 2 mod 7 = 2. Then $[(-19)^7 - (-19)] \bmod 7 = (2^7 - 2) \bmod 7 = 126 \bmod 7 = 0$. Also, for a = 0, $0^p = 0 = 0 \bmod p = 0$.

Comments: This theorem was stated by Fermat in 1640 without proof (as was his custom) and was later proved by Euler in 1736. In the Fermat primality test, if one can find an integer a such that $a^{p-1} \bmod p \neq 1$, one knows that p is not prime. The converse of the theorem is not true: that $a^p = a \bmod p$ for all a => p is prime. This converse fails for integers known as Carmichael numbers (Fermat pseudoprimes), the

smallest of which is 561 (found in 1910 by Carmichael). Here we test 561 = 3*11*17 for a = the first 10,000 integers:

```
p := 561: ifactor(561);
                                                    (3) (11) (17)
found := false:
for a from 1 to 10000 do
  if (a^p-a) mod p <> 0 then print(a);found := true; fi;
od;
if found = false then print("no exceptions found"); fi;
                                                    "no exceptions found"
```

There are an infinite number of these pseudoprimes.

Fermat's "big" theorem was his Last Theorem which says that the Diophantine equation $x^n + y^n = z^n$ has no solutions for n > 2 and x,y,x > 0. This theorem was conjectured by Fermat in 1637 in a margin note and was first proved by Andrew Wiles in 1995, some 358 years after the conjecture was made.

Fact 13: The number of primitive elements in Galois Field GF(q=p^m) is φ(p^m-1). (G.19)

Proof: First we quote (4.32):

Fact 12: A power α^k of a known primitive element α of GF(q) is itself a primitive element if and only if GCD(k,q-1) = 1. *(4.32)*

Since powers of α enumerate all of {GF(q) - 0, • }, we can just try all powers α^k for k = 1 to q-2 and see whether or not the condition GCD(k,q-1) = 1 is valid for that power. If the condition is true, α^k is a primitive element. Then the number of primitive elements of GF(q) is the number of integers k < q-1 which are coprime to q-1. But this is exactly the set G_{q-1} of totatives of q-1 discussed above, a set we showed has φ(q-1) elements. Thus, the number of primitive elements of GF(q) is just φ(q-1). **QED**

Fact 14: The number of primitive polynomials in Galois Field GF(p^m) is φ(p^m -1)/m. (G.20)

Proof: We know from (5.15) that a primitive polynomial has coefficients which lie in a conjugate set which contains m elements, all of which are primitive elements. The primitive polynomials for all m elements in such a conjugate set are the same primitive polynomial since they have the same roots of GF(q). Thus, we have to divide the total count of primitive elements by m to get the total number of distinct primitive polynomials for GF(q). QED.

(continued next page)

Fact 15: Consider the Diophantine equation ax = by where a,b > 0. Consider the set of solutions (x,y) in which x,y > 0. The solution with the smallest value of x is (x = b/d , y = a/d) where d = GCD(a,b).

(G.21)

Proof: Let N_1, N_2 be the positive integers N_1 = a/d, N_2 = b/d. From **Fact 1**, GCD(N_1,N_2) = 1. Also, we see that ax = by $\Leftrightarrow$ $N_1x = N_2y$. The solutions (x,y) of these two equations are the same, so the smallest-x solution will be the same. We shall now determine the smallest-x solution of $N_1x = N_2y$.

According to **Fact 2** with A = x and B = y, we know, since GCD(N_1,N_2) = 1 and $N_1x = N_2y$, that $x/N_2 = y/N_1 = I$, a positive integer. Thus, solutions of $N_1x = N_2y$ are $x = IN_2$ and $y = IN_1$ where I is a positive integer. The smallest-x solution must then be $x = N_2$ and $y = N_1$. This is then also the smallest-x solution of equation ax = by. Thus, the smallest-x solution of ax = by is x = b/d and y = a/d. **QED**

Appendix H: Order Reversal Theorems for Irreducible Polynomials

An order-reversed polynomial is one in which the order of the coefficients is reversed. For example, here H(x) is the order-reversed version of h(x),

$$\begin{aligned} h(x) &= c_0 + c_1x + c_2x^2 + c_3x^3 + \ldots c_{k-1}x^{k-1} + c_kx^k \quad = \{c_0,c_1,c_2.....c_{k-1},c_k\} \\ H(x) &= c_k + c_{k-1}x + c_{k-2}x^2 + c_{k-2}x^3 + \ldots + c_1x^{k-1} + c_0x^k \quad = \{c_k, c_{k-1}.....c_2,c_1, c_0\} \end{aligned} \tag{H.1}$$

Peterson and Weldon refer to such polynomials as being reciprocal. If h(x) and H(x) are the same polynomial, we call it a symmetric polynomial (self-reciprocal).

We shall now develop four Facts relating to order-reversed polynomials.

Fact 1: If h(x) is irreducible in R, then so is its order-reversed partner H(x). (H.2)

Corollary: Irreducible polynomials thus come in pairs as long as h(x) is not symmetric. Often tables of irreducible polynomials only state one of the pair in order to save space.

Proof of Fact 1: Let h(x) of degree k be irreducible for GF(p^m). We can write ($c_0 \neq 0$ and $c_k \neq 0$)

$$h(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + \ldots + c_kx^k \qquad \text{degree } k \leq m \qquad GF(p^m) .$$

If c_0 were 0, we could factor out x and h(x) would be reducible. If c_k were 0, we would not have degree k which we want for this proof. Now consider the order-reversed polynomial,

$$H(x) = c_k + c_{k-1}x + c_{k-2}x^2 + c_{k-2}x^3 + \ldots + c_1x^{k-1} + c_0x^k \qquad \text{degree } k .$$

Notice that

$$\begin{aligned} H(1/x) &= c_k + c_{k-1}x^{-1} + c_{k-2}x^{-2} + c_{k-2}x^{-3} + \ldots + c_1x^{-k+1}\ c_0x^{-k} \\ &= x^{-k}\,[c_0 + c_1x + c_2x^2 + c_3x^3 + \ldots + c_kx^k] \\ &= x^{-k}\,h(x) . \end{aligned} \tag{H.3}$$

Assume h(x) is irreducible and H(x) is reducible. We will find a contradiction and conclude that if h(x) is irreducible, then the order-reversed H(x) must also be irreducible.

If H(x) is reducible then it can be written as the product of two polynomials in R whose degrees add up to the degree of H(x) which is k. So write,

$$H(x) = [g_n(x)][f_{k-n}(x)] \tag{H.4}$$

where the g_n is of degree n and f_{k-n} is of degree k-n. Values of n range from 1 to k-1, so each factor is at least linear in x, not just a constant. We then find that

$$h(x) = x^{k} H(1/x) = x^{n}x^{k-n} [g_{n}(1/x)][f_{k-n}(1/x)]$$

$$= \{ x^{n} g_{n}(1/x) \} \{ x^{k-n} f_{k-n}(1/x) \}$$

$$= \{ G_{n}(x) \} \{ F_{k-n}(x)\} \qquad (H.5)$$

where $G_{n}(x)$ and $F_{k-n}(x)$ are both polynomials in R with $1 \leq n \leq k-1$ so neither factor is a constant. Then h(x) is reducible, and this is our contradiction. **QED**

Fact 2: If h(x) is a minimum polynomial in R, then so its order-reversed partner H(x). (H.6)

Corollary: Minimum polynomials thus come in pairs as long as h(x) is not symmetric. Often tables of minimum polynomials only state one of the pair in order to save space.

Proof of Fact 2: If h(x) is a a minimum polynomial in R, we know it is irreducible and it can be factored as

$$h(x) = (x-a_{1})(x-a_{2}) \ldots\ldots (x-a_{k}) \qquad (H.7)$$

where the a_{i} are all elements of $GF(p^{m})$ and are elements of a conjugate set as defined in Section 5 (c). We already know that H(x) is irreducible from Fact 1, but we need to show that H(x) is a minimum polynomial. From (H.3) ,

$$H(1/x) = x^{-k} h(x) . \qquad \textit{(H.3)}$$

so that

$$H(x) = x^{k} h(1/x) = x^{k}(1/x-a_{1})(1/x-a_{2}) \ldots\ldots (1/x-a_{k})$$

$$= (1-a_{1}x) (1-a_{2}x) \ldots\ldots (1-a_{k}x)$$

$$= [(-a_{1})(x-a_{1}^{-1})] [(-a_{2})(x-a_{2}^{-1})] \ldots [(-a_{k})(x-a_{k}^{-1})]$$

$$= \{(-a_{1}) (-a_{2}) \ldots (-a_{k}) \} \{(x-a_{1}^{-1}) (x-a_{2}^{-1}) \ldots (x-a_{k}^{-1}) .$$

Since any product of field elements is a field element, we can call the first factor γ in GF(q). Then

$$H(x) = \gamma (x-a_{1}^{-1}) (x-a_{2}^{-1}) \ldots (x-a_{k}^{-1}) . \qquad (H.8)$$

According to following Lemma, we know that the inverses of the elements of a conjugate set form a conjugate set, and therefore H(x) is a minimum polynomial. **QED**

Lemma 1: The set formed by inverting the elements of a conjugate set is a conjugate set. (H.9)

Proof: If α is a primitive element of GF(q), then we can enumerate GF(q) this way from (4.31),

$$\{ 0, 1, \alpha, \alpha^2, \alpha^3, \ldots\ldots \alpha^{q-2} \} \qquad \alpha^{q-1} = 1 \qquad \alpha^q = \alpha \quad . \tag{4.31}$$

The general form of a conjugate set from (5.40) is

$$\{ \alpha^s, \alpha^{sp}, \alpha^{sp^2}, \alpha^{sp^3} \quad , \ldots. \alpha^{sp^{k-1}} \} , \tag{5.40}$$

where α is a primitive element of GF(q) and α^s is some other element of GF(q) where s is an integer which we normally think of as lying in the range 1 to q-1. If we set s = q-2, we get this conjugate set

$$\{ \alpha^{(q-2)}, \alpha^{(q-2)p}, \alpha^{(q-2)p^2}, \alpha^{(q-2)p^3} \quad , \ldots. \alpha^{(q-2)p^{k-1}} \} . \tag{H.10}$$

If α^{-1} is the inverse of α, then $\alpha\ \alpha^{-1} = 1$. On the other hand, we know $\alpha\ \alpha^{q-2} = \alpha^{q-1} = 1$. Therefore we can identify

$$\alpha^{-1} = \alpha^{(q-2)} \tag{H.11}$$

and we can write the above conjugate set as

$$\{ \alpha^{-1}, (\alpha^{-1})^p, (\alpha^{-1})^{p^2}, (\alpha^{-1})^{p^3} \quad , \ldots. (\alpha^{-1})^{p^{k-1}} \} . \tag{H.12}$$

We know that in general

$$(\alpha^n)^{-1} = (\alpha\ \alpha\ \alpha \ldots)^{-1} = \alpha^{-1}\ \alpha^{-1}\ \alpha^{-1} \ldots = (\alpha^{-1})^n \tag{H.13}$$

so we can write the above conjugate set as

$$\{ \alpha^{-1}, (\alpha^p)^{-1}, (\alpha^{p^2})^{-1}, (\alpha^{p^3})^{-1} \quad , \ldots. (\alpha^{p^{k-1}})^{-1} \} \tag{H.14}$$

Since this is a conjugate set, we have shown that the inverses of the elements of a conjugate set form a conjugate set (in general a different one with the same number of elements). **QED**

Fact 3: If h(x) is a primitive polynomial in R, then so is its order-reversed partner H(x). (H.15)

Corollary: Primitive polynomials thus come in pairs as long as h(x) is not symmetric (see Fact 4 below). Often tables of primitive polynomials only state one of the pair in order to save space.

Proof of Fact 3: If h(x) is a primitive polynomial of GF(p^m) then it is of degree m and from (5.35) all members of its conjugate set a_i are primitive elements of GF(q). From (H.10) $\alpha^{-1} = \alpha^{(q-2)}$. According to (4.32), if α is primitive, $\alpha^{-1} = \alpha^{(q-2)}$ is also primitive since GCD(q-2,q-1) = 1 (see Lemma 1 below). Now recall from Fact 2 the forms for h(x) and its order-reversed H(x),

$$h(x) = (x-a_1)(x-a_2)..... (x-a_k) \qquad (H.7)$$

$$H(x) = \gamma\ (x-a_1^{-1})\ (x-a_2^{-1})\ ...\ (x-a_m^{-1})\ . \qquad (H.8)$$

Letting $\alpha = a_1$, a primitive element of h(x), we have just shown that $\alpha^{-1} = a_1^{-1}$ is also a primitive element and therefore H(x) is a primitive polynomial (and all the a_i^{-1} are primitive elements of GF(q)).

Lemma 1: The GCD of two sequential integers is 1. (H.16)

Proof: Assume GCD(n,n+1) = N > 1. Then there exist integers I and J such that

$$n/N = I \quad => \quad n = IN$$
$$(n+1)/N = J \quad => \quad n+1 = JN \quad => \quad IN + 1 = JN \quad => \quad N(J-I) = 1\ .$$

We end up then with (J-I) = 1/N which is impossible unless N = 1. **QED**

Fact 4: A primitive polynomial h(x) for $GF(p^m)$ *cannot* be symmetric if p and m are in these ranges:

$p = 2$ with $m \geq 3$
$p = 3$ with $m \geq 2$
$p > 3$ (H.17)

Corollary: Thus, the only fields for which h(x) *can* be symmetric are GF(2), $GF(2^2)$ and GF(3). This means there are no symmetric primitive polynomials of degree greater than 2 for any GF(q). As was shown in Chapter 5 (c), for GF(2) and GF(3) the only primitive polynomial is 1+x, and for $GF(2^2)$ the only one is $1 + x + x^2$ as we found in (5.29). Thus, in the only cases in which h(x) *can* be symmetric, it *is* symmetric (Murphy's Law). (H.18)

Proof of Fact 4: The conjugate set of h(x) with respect to $GF(p^m)$ looks like this, where α is a primitive element of $GF(p^m)$:

$$\{\ \alpha,\ \alpha^p,\ \alpha^{p^2},\ \alpha^{p^3}\ ,\ ....\ \alpha^{p^{m-1}}\ \} \qquad \alpha^{p^m} = \alpha \qquad \text{m conjugates} \qquad (5.16)$$

The conjugate set of H(x) we found from (H.10) is, with k = m now since h(x) is primitive,

$$\{\ \alpha^{(q-2)},\ \alpha^{(q-2)p},\ \alpha^{(q-2)p^2},\ \alpha^{(q-2)p^3}\ ,\ ....\ \alpha^{(q-2)p^{m-1}}\ \}\ . \qquad (H.10)$$

In order to have h(x) = H(x), these two conjugate sets have to be the same. One must be at worst a rearrangement of the other. But we shall now show that the element α is *missing* from the second set under the conditions stated above, and therefore the two sets cannot be the same, and therefore we cannot have H(x) = h(x) and therefore h(x) cannot be symmetric.

In order to have α be present in the second set we would have to have

$$\alpha = \alpha^{(q-2)p^i} \qquad \text{for some i in range 0 to m-1}$$

Multiply both sides by α^{p^i} to get

$$\alpha^{p^{i+1}} = \alpha^{(q-1)p^i} = [\alpha^{q-1}]^{p^i} = [1]^{p^i} = 1 \quad .$$

Since α is a primitive element of GF(p^m), we know that $\alpha^{q-1} = 1$. In order for the above equation to be true, we must have p^i+1 be a multiple of $q-1 = p^m - 1$ for some i in (0,m-1):

$$p^i+1 = K\,(p^m - 1) \qquad K = \text{integer} \qquad \text{(H.19)}$$

Obviously K = 0 does not work. We try K = 1 next. But for (p,m) in the ranges stated below, we will show that in fact $p^i+1 < (p^m - 1)$, so K = 1 does not work nor does any K > 1 work.

In order to show that $p^i+1 < (p^m - 1)$ for all i in our range 0 to m-1, if suffices to show this for the largest exponent i = m-1, for then it will be true as well for all smaller i. So we want to show that, for a certain range of p and m, we have

$$p^{m-1}+1 < (p^m - 1)$$

or

$$p^m - p^{m-1} > 2$$

or

$$p^{m-1}(p-1) > 2$$

Lemma 2 below shows that this inequality is true for p and m in these ranges

p = 2	m ≥ 3		
p = 3	m ≥ 2		
p > 3	m ≥ 1	or just	p > 3

These then are the ranges for which no K exists in (H.18) and therefore α which is in the conjugate set of h(x) is NOT in the conjugate set of H(x). These sets are then different, so h(x) ≠ H(x) which means h(x) cannot have symmetric coefficients for p and m as shown above.

Lemma 2: The inequality $p^{m-1}(p-1) > 2$ is true for the following values of p and m:

p = 2	true for m ≥ 3	
p = 3	true for m ≥ 2	
p > 3	true for m ≥ 1	(H.20)

Proof: Start with p = 2

$$2^{m-1} * 1 > 2 \qquad ?$$

For m ≥3 this is clearly true, but it is not true for m = 1 or m = 2.

Now consider p = 3.

$$3^{m-1}(2) > 2 \qquad ?$$

This is true for $m \geq 2$. Finally, consider $p > 3$.

$p^{m-1}(p-1) > 2$?

In this case $(p-1) > 2$ all by itself and so this is valid for $m \geq 1$. **QED**

References

There are many modern books on the subjects of Galois fields (finite fields) and error-correcting codes, as a quick scan of Amazon will attest. Below we list only a few, including some "classics". The list below is alphabetical by first author. Bad links can be often be replaced by a web search on title or author.

G.C. Ahlquist, B. Nelson and M. Rice, "Optimal Finite Field Multipliers for FPGAs", Lecture Notes in Computer Science Vol. 1673, 1999, pp 51.60. http://splish.ee.byu.edu/docs/ffmult.fpl99.pdf

BBC: The link below contains a fascinating 14 minute podcast on the short life of Évariste Galois. http://www.bbc.co.uk/podcasts/series/maths

G. Birkhoff and S. MacLane, *Survey of Modern Algebra, 4th. Ed* (Macmillan, New York, 1977). This is an excellent and classic text. Earlier editions were 1941, 1953, and 1965. A paperback printing was published by A.K. Peters in 2008.

L.S. Bobrow and M.A. Arbib, *Discrete Mathematics: Applied Algebra for Computer and Information Science* (W.B. Saunders, Philadelphia, 1974).

R.C. Bose and D.K. Ray-Chaudhuri "On a class of error-correcting binary codes", Information and control, 3, (1960), 68–79. Also "Further results on error-correcting binary group codes", Information and control, 3, (1960), 279–290.

W.H. Bussey, "Galois Field Tables for $p^n \leq 169$", Bull. Amer. Math. Soc, Vol 12, Num 1 (1905), 22-38. For each Galois Field with $p^n \leq 169$, the author provides one primitive polynomial along with a table of the type we show in (6.12). This paper is freely available as a PDF at http://projecteuclid.org .

W.H. Bussey, "Tables of Galois Fields of order less than 1000", Bull. Amer. Math. Soc, Vol 16, Num 4 (1910), 188-206. This paper is an extension of the previous one and is also available at Project Euclid.

R.W. Hamming, *Error Detecting and Error Correcting Codes,* Bell System Technical Journal, Vol XXIX No.2 April 1950. (Try www.lee.eng.uerj.br/~gil/redesII/**hamming.pdf** .)

A. Hocquenghem. "Codes correcteurs d'erreurs", Chiffres (Paris), 2:147–156, September 1959. Try http://kom.aau.dk/~heb/kurser/NOTER/KOFA02.pdf for the first few pages of this paper.

P. Kitsos, G. Theodoridis, and O. Koufopavlou, " An efficient reconfigurable multiplier architecture for Galois field GF(2^m) ", Microelectronics Journal 34 (2003) 975-980.
http://dsmc.eap.gr/members/pkitsos/papers/Kitsos_j02.pdf

R. Lidi and H. Niederreiter, *Finite Fields*, *2nd Ed.*, Volume 20 of The Encyclopedia of Mathematics and Its Applications (Cambridge University Press, 2008). The original Volume 20 was published in 1983, and a slightly reduced separate edition appeared in 1986. The Encyclopedia now has 142 volumes!

P. Lucht, *Tensor Analysis and Curvilinear Coordinates* (2012, http://user.xmission.com/~rimrock/). This document is segmented into two PDF files, the second containing a set of Appendices. If link is bad, search on "Phil Lucht Documents".

M. Olofsson (Linkoping University, Sweden), lists of primitive polynomials over GF(2).
http://www.commsys.isy.liu.se/en/staff/mikael/polynomials/primpoly

W.W. Peterson and D.T. Brown, "Cyclic Codes for Error Detection", Proc. IRE, 49, 228-235 (1961). This classic paper can be found online as well.

W.W. Peterson and E.J. Weldon, *Error Correcting Codes, 2nd Ed.* (The MIT Press, Cambridge, 1972).

M.Y. Rhee, *Error Correcting Coding Theory* (McGraw-Hill, New York, 1989).

E. Savas, A.F. Tenca and C.K. Koc, "A Scalable and Unified Multiplier Architecture for Finite Fields GF(p) and GF(2^m) ", Lecture Notes in Computer Science Vol. 1965, 2000, pp 277-292.
http://cryptocode.net/docs/c19.pdf

E.J. Watson, "Primitive Polynomials (Mod 2)", Math. Comp. 16 (1962), 368-369.
Try www.ams.org/journals/...16.../S0025-5718-1962-0148256-1.pdf

Xilinx Corp., "CoolRunner-II CPLD Galois Field GF(2^m) Multiplier", XAPP371 (v1.0) Sept. 26, 2003.
www.xilinx.com/support/documentation/application_notes/xapp371.pdf