STUDENT MATHEMATICAL LIBRARY
IAS/PARK CITY MATHEMATICAL SUBSERIES
Volume 7

Codes and Curves

Judy L. Walker

American Mathematical Society
Institute for Advanced Study

Contents

Preface

These notes summarize a series of lectures I gave as part of the IAS/PCMI Mentoring Program for Women in Mathematics, held May 17-27, 1999 at the Institute for Advanced Study in Princeton, NJ with funding from the National Science Foundation. The material included is not original, but the exposition is new. The booklet [**LG**] also contains an introduction to algebraic geometric coding theory, but its intended audience is researchers specializing in either coding theory or algebraic geometry and wanting to understand the connections between the two subjects. These notes, on the other hand, are designed for a general mathematical audience. In fact, the lectures were originally designed for undergraduates.

I have tried to retain the conversational tone of the lectures, and I hope that the reader will find this monograph both accessible and useful. Exercises are scattered throughout, and the reader is strongly encouraged to work through them.

Of the sources listed in the bibliography, it should be pointed out that [**CLO2**], [**Ga**], [**H**], [**L**], [**MS**], [**NZM**] and [**S**] were used most intensively in preparing these notes. In particular:

- Theorem 1.11, which gives some important properties of cyclic codes, can be found in [**MS**].

- The proof given for the Singleton Bound (Theorem 2.1) is from [**S**].
- The proofs given for the Plotkin Bound (Theorem 2.3), the Gilbert-Varshamov Bound (Theorem 2.4), and the asymptotic Plotkin Bound (Theorem 2.7) are from [**L**].
- Exercise 3.6, about finding points on a hyperbola, is taken from [**NZM**].
- The pictures and examples of singularities (as in Exercise 4.4) are from [**H**].
- The proof of the classification of finite fields outlined in the Exercises in Section B.3 is from [**CLO2**].

More generally, the reader is referred to [**L**], [**MS**], and [**S**] for more information on coding theory, [**H**], [**ST**], and [**CLO2**] for more information on algebraic geometry, and [**Ga**] for more background on abstract algebra. In particular, any results included in these notes without proofs are proven in these sources.

I would like to thank all of the people who contributed to the development of this monograph. In particular, special thanks go to: Chuu-Lian Terng and Karen Uhlenbeck, who organize the Mentoring Program and invited me to speak there; Kirstie Venanzi and especially Catherine Jordan, who provide the staff support for the program as well as for IAS/PCMI; Christine Heitsch, who did a great job coordinating problem sessions for my lectures; Graham Leuschke and Mark Walker, who proofread the various drafts of these notes; and, most importantly, the thirteen amazingly bright undergraduate women who participated in the program — Heidi Basler, Lauren Baynes, Juliana Belding, Mariana Campbell, Janae Caspar, Sarah Gruhn, Catherine Holl, Theresa Kim, Sarah Moss, Katarzyna Potocka, Camilla Smith, Michelle Wang, and Lauren Williams.

Judy L. Walker

Chapter 1

Introduction to Coding Theory

1.1. Overview

Whenever data is transmitted across a channel, errors are likely to occur. It is the goal of coding theory to find efficient ways of encoding the data so that these errors can be detected, or even corrected. Traditionally, the main tools used in coding theory have been those of combinatorics and group theory. In 1977, V. D. Goppa defined algebraic geometric codes [**Go**], thus allowing a wide range of techniques from algebraic geometry to be applied. Goppa's idea has had a great impact on the field. Not long after Goppa's original paper, Tsfasman, Vladut and Zink [**TVZ**] used modular curves to construct a sequence of codes with asymptotically better parameters than any previously known codes. The goal of this course is to introduce you to some of the basics of coding theory, algebraic geometry, and algebraic geometric codes.

Before we write down a rigorous definition of a code, let's look at some examples. Probably the most commonly seen code in day-to-day life is the International Standardized Book Number (ISBN) Code. Every book is assigned an ISBN, and that ISBN is typically displayed on the back cover of the book. For example, the ISBN for *The Theory of Error-Correcting Codes* by MacWilliams and Sloane

([**MS**]) is 0-444-85193-3. The first nine digits 0-444-85193 contain information about the book. The last "3", however, is a check digit which is chosen on the basis of the first nine. In general, the check digit a_{10} for the ISBN $a_1-a_2a_3a_4-a_5a_6a_7a_8a_9$ is chosen by computing $a_{10}' := (a_1 + 2a_2 + \cdots + 9a_9)$. If $a_{10}' \equiv i \pmod{11}$ for some i with $0 \leq i \leq 9$, we set $a_{10} = i$. If $a_{10}' \equiv 10 \pmod{11}$, we set a_{10} to be the symbol "X". The point is that every book is assigned an ISBN using the same system for choosing a check digit, and so, for example, if you are working in the Library of Congress cataloging new books and you make a mistake when typing in this number, the computer can be programmed to catch your error.

The ISBN Code is a very simple code. It is not hard to see that it *detects* all single-digit errors (a mistake is made in one position) and all transposition errors (the numbers in two positions are flipped). It cannot *correct* any single-digit or transposition errors, but this is not a huge liability, since one can easily just type in the correct ISBN (re-send the message) if a mistake of this type is made. Further, the ISBN code is efficient, since only one non-information symbol needs to be used for every nine-symbol piece of data.

The so-called Repetition Codes provide an entire class of simple codes. Suppose, for example, every possible piece of data has been assigned a four bit string (a string of zeros and ones of length four), and suppose that instead of simply transmitting the data, you transmit each piece of data three times. For instance, the data string 1011 would be transmitted as 1011 1011 1011. If one error occurs, then that error would be contained in one of the three blocks. Thus the other two blocks would still agree, and we would be able to detect and correct the error. If we wanted to be able to correct two errors, we would simply transmit each piece of data five times, and in general, to correct t errors, we would transmit the data $2t + 1$ times.

The Repetition Codes have an advantage over the ISBN Code in that they can actually correct errors rather than solely detect them. However, they are very inefficient, since if we want to be able to correct just one error, we need to transmit a total of three symbols for every information symbol.

We are now in a position to make some definitions.

Definition 1.1. A *code* C over an *alphabet* A is simply a subset of $A^n := A \times \cdots \times A$ (n copies).

In this course, A will always be a finite field, but you should be aware that much work has been done recently with codes over finite rings; see Project C.6. Appendix B discusses finite fields, but for now, you may just think of the binary field $\mathbb{F}_2 := \{0, 1\}$, where addition and multiplication are done modulo 2. More generally, for any prime p, we have a field $\mathbb{F}_p := \{0, 1, \ldots, p-1\}$ with addition and multiplication modulo p.

Definition 1.2. Elements of a code are called *codewords*, and the *length* of the code is n, where $C \subseteq A^n$. If A is a field, C is called a *linear code* if it is a vector subspace of A^n, and in this case the *dimension* k of C is defined to be the dimension of C as a vector space over A. Notice that if $A = \mathbb{F}_q$ is the finite field with q elements, and C is a linear code over A, then $k = \log_q(\#C)$, where $\#C$ is the number of codewords in C. Together with the *minimum distance* $d_{\min}$ of C which we define below, n and k (or n and $\#C$ in the nonlinear case) are called the *parameters* of C.

If C is a linear code of length n and dimension k over A, we can find k basis elements for C, each of which will be a vector of length n. We form a $k \times n$ matrix by simply taking the basis elements as the rows, and this matrix is called a *generator matrix* for C.

Notice that if G is a generator matrix for C, then C is exactly the set $\{\mathbf{u}G \mid \mathbf{u} \in A^k\}$. For example, the matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

is a generator matrix for a linear code of length 3 and dimension 2.

Definition 1.3. For $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_n) \in A^n$, we define the *Hamming distance* from $\mathbf{x}$ to $\mathbf{y}$ to be

$$d(\mathbf{x}, \mathbf{y}) := \#\{i \mid x_i \neq y_i\}.$$

For $\mathbf{x} \in A^n$, we also define the *Hamming weight* of $\mathbf{x}$ to be $wt(\mathbf{x}) = d(\mathbf{x}, (0, 0, \ldots, 0))$.

Exercise 1.4. Show that the Hamming distance in fact defines a metric on A^n. In other words, show that for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in A^n$, we have:

a) $d(\mathbf{x}, \mathbf{y}) \geq 0$, with $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$,

b) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$, and

c) $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$.

Definition 1.5. The *minimum distance* of C is

$$d_{\min} := d_{\min}(C) = \min\{d(\mathbf{x}, \mathbf{y}) \,|\, \mathbf{x}, \mathbf{y} \in C \text{ and } \mathbf{x} \neq \mathbf{y}\}$$

If the meaning is clear from context, we will often drop the subscript and simply write d for the minimum distance of a code.

Exercise 1.6. Show that if C is a linear code then the minimum distance of C is $\min\{wt(\mathbf{x}) \,|\, \mathbf{x} \in C \text{ and } \mathbf{x} \neq (0, 0, \ldots, 0)\}$. In other words, show that for linear codes, the minimum distance is the same as the *minimum weight*.

Let's now return to our examples. The ISBN Code is a code of length 10 over $\mathbb{F}_{11}$ (where the symbol X stands for the element $10 \in \mathbb{F}_{11}$). It is a nonlinear code since the X can never appear in the first nine positions of the code. It has 10^9 codewords, and the minimum distance is 2. Our Repetition Code is a linear code over $\mathbb{F}_2$ of length $4r$, where r is the number of times we choose to repeat each piece of data. The dimension is 4, and the minimum distance is r.

Why are the dimension (or number of codewords) and minimum distance of a code important? Suppose C is a linear code over an alphabet A which has length n, dimension k, and minimum distance d. We may think of each codeword as having k information symbols and $n - k$ checks. Thus, we want k large with respect to n so that we are not transmitting a lot of extraneous symbols. This makes our code efficient. On the other hand, the value of d determines how many errors our code can correct. To see this, for $\mathbf{x} \in A^n$ and a positive integer t, define $B_t(\mathbf{x})$ to be the ball of radius t centered at $\mathbf{x}$. In other words, $B_t(\mathbf{x})$ is the set of all vectors in A^n which are Hamming distance at most t away from $\mathbf{x}$. Since C has minimum distance d, two balls of radius $\lfloor \frac{d-1}{2} \rfloor$ centered at distinct codewords cannot intersect. Thus, if at most $\lfloor \frac{d-1}{2} \rfloor$ errors are made in transmission, the received

word will lie in a unique ball of radius $\lfloor \frac{d-1}{2} \rfloor$, and that ball will be centered at the correct codeword. In other words, a code of minimum distance d can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors, so we want d large with respect to n as well.

The question now, of course, is: If we say that a linear code is good if both k and d are large with respect to n, then just how good can a code be?

As a partial answer to this question, let's turn now to the Reed-Solomon codes. Let $\mathbb{F}_q$ be the field with q elements. For any non-negative integer r, define $L_r := \{f \in \mathbb{F}_q[x] \mid \deg(f) \leq r\} \cup \{0\}$. Note that L_r is a vector space over the field $\mathbb{F}_q$.

Exercise 1.7. Show that $\dim_{\mathbb{F}_q}(L_r) = r + 1$ by finding an explicit basis.

Definition 1.8. Label the $q-1$ nonzero elements of $\mathbb{F}_q$ as $\alpha_1, \ldots, \alpha_{q-1}$ and pick $k \in \mathbb{Z}$ with $1 \leq k \leq q-1$. Then the *Reed-Solomon Code* $RS(k,q)$ is defined to be

$$RS(k,q) := \{(f(\alpha_1), \ldots, f(\alpha_{q-1})) \mid f \in L_{k-1}\}.$$

Notice that $RS(k,q)$ is a subset of $\mathbb{F}_q^{q-1} := \mathbb{F}_q \times \cdots \times \mathbb{F}_q$ ($q-1$ copies), so $RS(k,q)$ is a code over the alphabet $\mathbb{F}_q$. Further, since the map $\epsilon : L_k \to \mathbb{F}_q^{q-1}$ given by $\epsilon(f) = (f(\alpha_1), \ldots, f(\alpha_{q-1}))$ is a linear transformation (see Definition A.21) and $RS(k,q)$ is its image, $RS(k,q)$ is a linear code. What are the parameters of $RS(k,q)$? Certainly the length is $n = q-1$ and the dimension is at most $\dim L_{k-1} = k$. If $\epsilon(f) = \epsilon(g)$, then $f - g$ has at least $q-1$ roots, so by Exercise B.10, $f - g$ has degree at least $q-1$. But $f - g \in L_k$, which implies $f = g$. Thus C has dimension exactly k. To find the minimum distance, we'll use Exercise 1.6 and find the minimum weight instead. So, suppose $f \in L_{k-1}$ and $wt(\epsilon(f)) = d = d_{\min}$. Then f has at least $n - d$ zeros, so it has degree at least $n - d$ (again using Exercise B.10). Since $f \in L_{k-1}$, this means that $n - d \leq k - 1$, or, equivalently, $d \geq n - k + 1$.

In Chapter 2.1, we will show that, in fact, we have $d = n - k + 1$.

1.2. Cyclic Codes

Before we move on, we should spend a little time on cyclic codes. This class of codes is very important. In particular, some of the codes given as possible project topics in Appendix C are cyclic codes.

Definition 1.9. A linear code C is called a *cyclic code* if it has the following property: whenever $(c_0, c_1, \ldots, c_{n-1}) \in C$, it is also true that $(c_1, c_2, \ldots, c_{n-1}, c_0) \in C$.

More generally, the *automorphism group* $\mathrm{Aut}(C)$ of a code C is the set of permutations $\sigma \in S_n$ such that $\sigma(\mathbf{c}) \in C$ for all $\mathbf{c} \in C$, where $\sigma(c_0, \ldots, c_{n-1}) = (c_{\sigma(0)}, \ldots, c_{\sigma(n-1)})$. In other words, the code C is cyclic if and only if the permutation $\sigma = (0, 1, 2, \ldots, n-1)$ is in $\mathrm{Aut}(C)$.

There is a very nice algebraic way of looking at cyclic codes which we will now investigate. Let C be a cyclic code over the field $\mathbb{F}_q$. As in Appendix A, we set $R_n := \mathbb{F}_q[x]/\langle x^n - 1\rangle$. We can think of elements of R_n as polynomials of degree at most $n-1$ over $\mathbb{F}_q$, where multiplication is done as usual except that $x^n = 1$, $x^{n+1} = x$, and so on (see Exercise A.17). Thus, we can identify C with

$$I_C := \{\mathbf{c}(x) := c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in R_n \mid \mathbf{c} := (c_0, c_1, \ldots, c_{n-1}) \in C\}.$$

(This is the reason for indexing the coordinates of a cyclic code beginning with 0 rather than 1.)

Exercise 1.10. Let C be a cyclic code. Show that I_C is an ideal of R_n.

Exercise A.13 shows that every ideal of $\mathbb{F}_q[x]$ is principal, generated by the unique monic polynomial of smallest degree inside the ideal. The next Theorem first shows that the same is true for ideals of R_n, then gives some important properties of that polynomial.

Theorem 1.11. *Let I be an ideal of R_n and let $g(x) \in I$ be a monic polynomial of minimal degree. Let $\ell = \deg(g(x))$. Then*

a) *$g(x)$ is the only monic polynomial of degree ℓ in I.*

b) *$g(x)$ generates I as an ideal of R_n.*

c) $g(x)$ *divides* $x^n - 1$ *as elements of* $\mathbb{F}_q[x]$.

d) *If* $I = I_C$ *for some cyclic code* C, *then* $\dim C = n - \ell$.

Proof. Suppose first that $f(x) \in I$ is monic of degree ℓ. If $f(x) \neq g(x)$, then $f(x) - g(x)$ is a polynomial of degree strictly less than ℓ in I. Multiplying by an appropriate scalar yields a monic polynomial, which contradicts the minimality of ℓ, proving (a).

To prove (b), let $c(x)$ be any element of I. Lifting to $\mathbb{F}_q[x]$, we can use the division algorithm to write $c(x) = f(x)g(x) + r(x)$ for polynomials $f(x)$ and $r(x)$ with $r(x)$ either 0 or of degree strictly less than ℓ. Since $c(x)$, $g(x)$ and $r(x)$ all have degree less than n, it must also be true that $f(x)$ has degree less than n, so this equation makes sense in R_n as well. But then we have $r(x) = c(x) - f(x)g(x) \in I$, which means $r(x) = 0$ by minimality of ℓ.

For (c), use the division algorithm in $\mathbb{F}_q[x]$ to write $x^n - 1 = q(x)g(x) + r(x)$ with $r(x)$ either 0 or having degree strictly less than ℓ. Passing to R_n, we have $r(x) = -q(x)g(x) \in I$, which implies $r(x) = 0$ in R_n by minimality of ℓ. Thus $r(x) = 0$ in $\mathbb{F}_q[x]$ as well since otherwise $x^n - 1$ divides $r(x)$, which makes $r(x)$ have degree at least $n > \ell$.

Finally, let $\mathbf{c} \in C$ be any codeword. Then $\mathbf{c}(x) \in \langle g(x) \rangle \subset R_n$, so there is some $f(x) \in R_n$ with $\mathbf{c}(x) = f(x)g(x)$. In $\mathbb{F}_q[x]$, then, we have $\mathbf{c}(x) = f(x)g(x) + e(x)(x^n - 1)$ for some polynomial $e(x) \in \mathbb{F}_q[x]$. Using (c), we have $\mathbf{c}(x) = g(x)(f(x) + e(x)q(x))$, where $g(x)q(x) = x^n - 1$. Setting $h(x) = f(x) + e(x)q(x)$, we have $\mathbf{c}(x) = g(x)h(x)$, where $\deg(h(x)) \leq n - \ell - 1$. Thus the codewords of C, when thought of as elements of $\mathbb{F}_q[x]$, are precisely the polynomials of the form $g(x)h(x)$, where $h(x) \in L_{n-\ell-1}$, so $\dim C = \dim L_{n-\ell-1} = n - \ell$. This proves (d). □

Because of the importance of this generator of the ideal I_C, we give it a special name.

Definition 1.12. If C is a cyclic code, we define the *generator polynomial* for C to be the unique monic polynomial $g(x) \in I_C$ of minimal degree.