

Decimal Codes

To make error correcting codes easier to use and analyze, it is necessary to impose some algebraic structure on them. It is especially useful to have an alphabet in which it is possible to add, subtract, multiply and divide without restriction. In other words we wish to construct a *finite field*. Evarist Galois (1811-32), a French mathematician who died in a duel at the age of 20 introduced finite fields and proved that there exists a field of order q if and only if q is a prime power (i.e. $q = p^r$, where p is prime and r is a positive integer). Furthermore, there is, up to relabeling, only one field of that order. Finite fields of order q are also known as *Galois field* of order q and are denoted by $GF(q)$.

Let us now try to give $\mathbf{Z}_m = \{0, 1, 2, \dots, m-1\}$ the structure of a field. We define *addition* and *multiplication* in \mathbf{Z}_m by $a + b \equiv c \pmod{m}$ and $ab \equiv d \pmod{m}$. For example in \mathbf{Z}_{12} we have

$$8 + 4 \equiv 0 \pmod{12}, \quad 5 + 7 \equiv 11 \pmod{12}, \quad 8 + 9 \equiv 5 \pmod{12}, \quad 4 \times 5 \equiv 8 \pmod{12}, \quad 3 \times 4 \equiv 0 \pmod{12}.$$

Note that $3 \times 4 \equiv 0 \pmod{12}$, thus \mathbf{Z}_{12} is not a field. The following theorem characterizes \mathbf{Z}_m .

Theorem 1. \mathbf{Z}_m is a field if and only if m is a prime number.

Proof. Suppose m is not prime, then $m = ab$ for some integers a and b , both less than m . Thus

$$ab \equiv 0 \pmod{m}, \quad \text{with } a \neq 0 \text{ and } b \neq 0.$$

So, m must be prime.

Now suppose that m is prime. To prove that \mathbf{Z}_m is a field we only need to show that every nonzero member of \mathbf{Z}_m has a multiplicative inverse. Let $a \in \mathbf{Z}_m$ with $a \neq 0$, then $\{1a, 2a, \dots, (m-1)a\}$ must be distinct in \mathbf{Z}_m . If not then for some $i, j \in \mathbf{Z}_m$ with $i > j$ and

$$ia = ja \Rightarrow (i - j)a \equiv 0 \pmod{m} \Rightarrow m \text{ divides } a \text{ or } (i - j).$$

This is a contradiction with the fact that m is greater than both a and $i - j$. Thus \mathbf{Z}_m is a field.

According to this theorem \mathbf{Z}_{10} is not a field but \mathbf{Z}_{11} is a field. Although \mathbf{Z}_{10} is not a field but some of its members have an inverse, for example the inverse of 3 in \mathbf{Z}_{10} is 7.

♣ **The Extended Euclidean Algorithm.** This algorithm finds the inverse of a number x in \mathbf{Z}_m . It also shows if x has no inverse in \mathbf{Z}_m .

First we set $v_0 = x$ and $v_1 = n$. The quotient obtained at step k will be denoted by q_k . As we carry out each step of the Euclidean Algorithm, we will also calculate an auxiliary

number, p_k . For the first two steps, the value of this number is given: $p_0 = 0$ and $p_1 = 1$. For the remainder of the steps, we recursively calculate

$$p_k \equiv p_{k-2} - p_{k-1}q_{k-2} \pmod{n}.$$

Continue this calculation for one step beyond the last step of the Euclidean algorithm. The algorithm starts by dividing n by x .

Case 1. The last non-zero remainder occurs at step k , then if this remainder is 1, x has an inverse and it is p_{k+2} .

Case 2. The last non-zero remainder is not 1, then x does not have an inverse.

Example. Find the inverse of 15 mod 26.

First we set $x_0 = 15$ and $x_1 = 26$.

Steps	$x_{k+1} = q_k(x_k) + r_k$	$p_k \equiv p_{k-2} - p_{k-1}q_{k-2} \pmod{n}$
Step 0	$26 = 1(15) + 11$	$p_0 = 0$
Step 1	$15 = 1(11) + 4$	$p_1 = 1$
Step 2	$11 = 2(4) + 3$	$p_2 \equiv 0 - 1(1) \pmod{26} = 25$
Step 3	$4 = 1(3) + 1^\dagger$	$p_3 \equiv 1 - 25(1) \pmod{26} \equiv -24 \pmod{26} = 2$
Step 4	$3 = 3(1) + 0$	$p_4 \equiv 25 - 2(2) \pmod{26} = 21$
Step 5	The inverse is found	$p_5 \equiv 2 - 21(1) \pmod{26} \equiv -19 \pmod{26} = 7$

$\dagger r_3 = 1$, so the inverse of 15 modulo 26 exists.

Exercises. In \mathbf{Z}_{34} , find the inverse (if there exists) of

- (a) 21,
- (b) 26.

♣ Decimal Codes. In applications involving people, numbers with decimal digits are often used (because humans have 10 fingers). Humans and machines processing these numbers make decimal mistakes in them. The most common mistakes that humans make are to alter a single digit or to transpose two adjacent digits. It is also common to drop a digit or to add an extra digit. Machines can misread a digit, though they would not transpose digits.

We shall discuss some interesting decimal codes, that is, codes over the alphabet \mathbf{Z}_{10} . Since \mathbf{Z}_{10} is not a field, to work around this problem, we first define codes over \mathbf{Z}_{11} , and then “reduce” them to decimal codes.

First we construct a single-error-detecting decimal code known as *ISBN* code, then we shall construct single-error-correcting and double-error-correcting decimal codes of length 10 and determine efficient algorithms for decoding them. We shall explain how to generalize this construction for a family of *t-error-correcting* codes over finite fields $GF(q)$,

where $2t + 1 < q$. These codes are particular examples of *BCH* codes (*BCH* codes were discovered independently by Hocquenghem (1959) and by Bose and Ray-Chahudrri (1960)) or Reed-Solomon codes.

Decoding of these codes depends on solving a certain system of simultaneous non-linear equation, for which coding theorists have devised some clever methods of solution. Surprisingly, such a system of equations was first solved by Ramanujan (1912) in seemingly little-known paper in the *Journal of the Indian Mathematical Society*. The decoding algorithm is based on Ramanujan's method, which is easy to understand and makes use of the method of partial fractions.

♣ **The Single-Error-Detecting *ISBN* Code.** Every recently published book has a number associated with it, known as its *International Standard Book Number*, or *ISBN* (This number generally appears on the back cover of the book.) *ISBN* is a 10-digit codeword assigned by the publisher. For example, a book might have the *ISBN*

$$0 - 13 - 562901 - 2$$

although the hyphens may appear in different places and are in fact unimportant. The first digit of an *ISBN* indicates the language of the book, which in this case is 0, for English. The next group of digits (13) stand for the publisher, which in this case is Prentice-Hall. The next group of numbers (562901) is the book number assigned by the publisher (in this case, *Matrices and Matlab*, a tutorial by Marvin Marcus). The final digit is a *redundant check digit*, designed to detect errors. This digit is chosen to make the whole 10-digit number $v = v_1v_2 \dots v_{10}$ satisfy

$$v_1 + 2v_2 + 3v_3 + \dots + 9v_9 + 10v_{10} \equiv 0 \pmod{11}.$$

The left-hand side of the above equation is called the *weighted check sum* of the number $v = v_1v_2 \dots v_{10}$. In fact, since $-10 \equiv 1 \pmod{11}$, then v_{10} is define by

$$v_{10} \equiv v_1 + 2v_2 + 3v_3 + \dots + 9v_9 \pmod{11}$$

For instance, in the *ISBN* given above, the check digit is

$$1 + 2 \times 1 + 3 \times 3 + 4 \times 5 + 5 \times 6 + 6 \times 2 + 7 \times 9 + 8 \times 0 + 9 \times 1 \equiv 2 \pmod{11}$$

The publisher is forced to allow a symbol X ("ten" in Roman numerals), in the final position if the check digit v_{10} turns out to be a '10'; for example, *Impressionism and Post-Impressionism, The Hermitage, Leningrad, the Pushkin Museum of Fine Arts, Moscow, and the National Gallery of Art, Washington* has *ISBN* 0 - 517 - 66562 - X .

In the language of coding theory, we construct the *ISBN* code as follows. Let C be the linear code in \mathbf{Z}_{11}^{10} with parity-check matrix H , where

$$H^t = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \]$$

California State University, East Bay

This code is a $[10,9,2]$ -code over \mathbf{Z}_{11} of size 11^9 . Let \mathcal{I} be the code obtained from C by removing all codewords that have “10” in any position, except possible the last. The codewords in \mathcal{I} are precisely the ISBNs and so we refer to \mathcal{I} as the ISBN code. Note that $c_1 = (0, 1, 9, 8, 5, 3, 8, 0, 4, 9)$ and $c_2 = (0, 7, 6, 4, 1, 5, 2, 1, 4, 9)$ are both in \mathcal{I} , but their sum $c_1 + c_2 = (0, 8, 4, 1, 6, 8, 10, 1, 8, 7)$ is not in \mathcal{I} . Thus \mathcal{I} is not a linear code.

We now show that ISBN code \mathcal{I} can detect any single error, as well as any transposition of two digits in a codeword.

(a) Suppose the received word $w = w_1w_2 \dots w_{10}$ is the same as the codeword $v = v_1v_2 \dots v_{10}$ except that the digit v_j is received as $w_j = v_j + a$ with $a \neq 0$. This means that the error pattern $u = w - v = ae_j$, where e_j is the j^{th} row of the 10×10 identity matrix I_{10} . The syndrome of w is

$$s(w) = s(u) = s(ae_j) = ae_jH \equiv j \times a \pmod{11}$$

which is the product of two nonzero element of the field \mathbf{Z}_{11} and is therefore nonzero. Thus a single error is detected by the presence of a nonzero syndrome.

(b) Suppose the received word $w = w_1w_2 \dots w_{10}$ is the same as the codeword $v = v_1v_2 \dots v_{10}$ except that digits w_i and w_j (assume that $i < j$) have been transposed. Hence the error pattern is

$$u = w - v = (v_j - v_i)e_i + (v_i - v_j)e_j = 0 \dots 0(v_j - v_i)0 \dots (v_i - v_j)0 \dots 0$$

and the syndrome of w is

$$s(w) = s(u) = (v_j - v_i)e_i + (v_i - v_j)e_jH = i \times (v_j - v_i) + j \times (v_i - v_j) \equiv (v_j - v_i)(i - j) \pmod{11}$$

But since $i \neq j$ and $v_i \neq v_j$, again we have a nonzero syndrome in \mathbf{Z}_{11} . Thus a transposition of digits is also detected by means of a nonzero syndrome.

The ISBN error cannot be used to correct an error unless we know that just a given digit is in error or the codeword has been received with one of the digit unreadable. For example, suppose we receive the word w

$$0 - 201 - 1 \blacksquare - 502 - 7$$

We choose the sixth digit as an x , then we have

$$1 \times 0 + 2 \times 2 + 3 \times 0 + 4 \times 1 + 5 \times 1 + 6 \times x + 7 \times 5 + 8 \times 0 + 9 \times 2 + 10 \times 7 \equiv 0 \pmod{11}$$

Hence

$$6x + 136 \equiv 0 \pmod{11} \Rightarrow 6x + 4 \equiv 0 \pmod{11} \Rightarrow 6x \equiv -4 \pmod{11}$$

Since $-4 \equiv 7 \pmod{11}$ and $6^{-1} \equiv 2 \pmod{11}$, we conclude that

$$x \equiv 7 \times 2 \equiv 14 \equiv 3 \pmod{11}$$

Thus the *ISBN* codeword is

$$0 - 201 - 13 - 502 - 7$$

Exercises. (a) Check whether the following are *ISBN*s.

1. 0 - 550 - 10206 - X
2. 0 - 387 - 94704 - 5
2. 0 - 7641 - 9115 - 3

(b) The following *ISBN*s have been received with smudges. What are the missing digits?

1. 0 - 7858 - 0■28 - 5
2. 0 - 201 - 1344■ - 9

♣ **A Single-Error-Correcting Code.** When we think of a single-error-correcting code, we think of Hamming codes — so let us start with the Hamming code $H_{11}(2)$, which is an 11-array linear $[12, 10, 3]$ -code with parity-check matrix H_{11} , where

$$H_{11}^t = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

Now we shorten this code in the first and last coordinates to obtain a $(10, 8, 3)$ linear code C with a parity-check matrix H , where

$$H^t = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

Let D be the decimal code obtain from C by removing all codewords that have a 10 in any position. As we explained in the *ISBN* code, this code is not linear. The minimum distance of this code is 3 over \mathbf{Z}_{10} .

We now show that D can correct any single error, as well as detect any transposition of two digits in a codeword.

(a) Suppose the received word $w = w_1 w_2 \dots w_{10}$ is the same as the codeword $v = v_1 v_2 \dots v_{10}$ except that the digit v_j is received as $w_j = v_j + a$ with $a \neq 0$. This means that the error pattern $u = a e_j$. The syndrome of w is

$$s(w) = s(u) = s(a e_j) = a e_j H = [a, (j-1)a] \equiv [s_1, s_2] \pmod{11}$$

If $s_1 + s_2 = 0 \pmod{11}$, then there is no error. If not, then we see immediately that the magnitude a of the error is the first component s_1 of s . To determine the position j , of

the error, first we find s_1^{-1} in \mathbf{Z}_{11} , then we solve the equation $s_2 = a(j-1)$ for j . Thus $j = s_1^{-1}s_2 + 1$ and $u = a e_j$. When $j = 0$, then $s_1 + s_2 = 0 \pmod{11}$ which makes w a codeword.

Note. The fact that 11 is a prime number, $s_1^{-1}s_2 \neq 0$, thus $j = s_1^{-1}s_2 + 1 \neq 1$. Hence an error in the first digit will never be corrected.

Example. Suppose that $w = 1\ 2\ 7\ 4\ 2\ 3\ 5\ 1\ 1\ 2$ is received. The syndrome of w is $s(w) = wH = 6\ 4$ and so $s_1 = 6$, $s_2 = 4$, and

$$s_1^{-1}s_2 + 1 = (6)^{-1} \times 4 + 1 = 2 \times 4 + 1 = 8 + 1 = 9 \pmod{11}.$$

Hence, the error pattern is $6e_9$ and the codeword

$$v = w - 6e_9 = 1274235112 - 0000000060 = 1274235162 \pmod{11}$$

(b) Suppose the received word $w = w_1w_2 \dots w_{10}$ is the same as the codeword $v = v_1v_2 \dots v_{10}$ except that digits w_i and w_j (assume that $i < j$) have been transposed. Hence the error pattern is

$$u = w - v = (v_j - v_i)e_i + (v_i - v_j)e_j = 0 \dots 0(v_j - v_i)0 \dots (v_i - v_j)0 \dots 0$$

and the syndrome of w is

$$\begin{aligned} s(w) = s(u) &= [(v_j - v_i)e_i + (v_i - v_j)e_j]H = [v_j - v_i + v_i - v_j, (v_j - v_i)(i - j)] \\ &= [0, (v_j - v_i)(i - j)] \equiv [s_1, s_2] \pmod{11} \end{aligned}$$

Note that the first component of the syndrome is zero but the second component is not zero, we may conclude that a transposition error has occurred or the first digit is wrong. But since $i \neq j$ and $v_i \neq v_j$, again we have a nonzero syndrome in \mathbf{Z}_{11} . Thus a transposition of digits is also detected by means of a nonzero syndrome.

Note. The facts that the entry $H(1,2) = 0$ and any number multiplied by zero is zero, imply that we could have $s(w) = [0, s_2]$ with no occurrence of a transposition error.

Example. Suppose that $w = 1534232110$ is received, then the syndrome is $s(w) = wH = 07$. Since the first component of the syndrome is zero but the second component is 7, we conclude that a transposition error may have occurred.

Apart from the *ISBN* code, modulus 11 decimal codes are now widely used, mainly for error detection rather than correction. One of the earliest uses was in the allocation of registration numbers to the entire population of Norway in a scheme devised by Selmer (cf. 1967).

Every person in Norway has an 11-digit decimal registration number x_1x_2, \dots, x_{11} , where $x_1, x_2 \dots x_6$ is the date of birth, $x_7x_8x_9$ is a personal number and x_{10} and x_{11} are check digits defined by the parity-check matrix:

$$H^t = \begin{bmatrix} 3 & 7 & 6 & 1 & 8 & 9 & 4 & 5 & 2 & 1 & 0 \\ 5 & 4 & 3 & 2 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}$$

This code will detect all double error except when the error-pattern is in the form

$$u = (0, 0, 0, \lambda, 0, 0, 0, 0, -\lambda, 0) \pmod{11}$$

This is caused by the fact that the fourth and tenth columns of H^t are equal! (Not very clever). This causes the syndrome $s(u) = uH = [\lambda - \lambda, 2\lambda - 2\lambda] = [0, 0]$.

♣ **The U.S. Banking Check Scheme.** The U.S. banks place an 8-digit processing number on bank checks and they add a check digit at the end. The check equation is as follows:

$$(a_0 + 3 * a_1 + 7 * a_2 + a_3 + 3 * a_4 + 7 * a_5 + a_6 + 3 * a_7 + 7 * a_8) \pmod{10} = 0$$

It is essentially the same scheme to repeat the coefficients 1, 3, and 7 indefinitely. This scheme catches all single errors and all adjacent transpositions of digits that do not differ by 5. Thus 88.888% of adjacent transposition errors are caught (80 out of 90). (Computer programs later in this section verify this fact; it is also easy to prove mathematically.)

♣ **The IBM Check Scheme.** The most common check scheme now in use, for example on credit card numbers, is often called the "IBM" check." Here is the check equation:

$$(a_0 + 2\#a_1 + a_2 + 2\#a_3 + a_4 + 2\#a_5 + a_6 + \dots + 2\#a_{2i} + a_{2i+1} + \dots) \pmod{10} = 0$$

where $2\#a_i$ means to multiply a_i by 2 and add the decimal digits

$$2\#a_i = [(2 * a_i) \div 10] + (2 * a_i) \pmod{10}.$$

For example, if the account number is 5 4 9 9 6, then the check equation without the check digit is:

$$\begin{aligned} (2\#5 + 4 + 2\#9 + 9 + 2\#6) \pmod{10} &= ((1 + 0) + 4 + (1 + 8) + 9 + (1 + 2)) \pmod{10} \\ &= (1 + 4 + 9 + 9 + 3) \pmod{10} = 26 \pmod{10} = 6 \end{aligned}$$

so that the check digit must equal 4 to make the check equation true. Actual credit cards currently have 16 digits and place the check digit on the right, but they treat the other digits as above, so that the first (leftmost) digit is acted on by $2\#$, while the final 16th digit (the rightmost digit, which is also the check digit) is just added in to the check sum. For example, if a VISA card has number

$$4270 \ 7100 \ 1591 \ 2024$$

then the rightmost 4 is the check digit, chosen so that the check equation will work out to zero:

$$\begin{aligned} (2\#4 + 2 + 2\#7 + 0 + 2\#7 + 1 + 2\#0 + 0 + 2\#1 + 5 + 2\#9 + 1 + 2\#2 + 0 + 2\#2 + 4) \pmod{10} &= \\ (8 + 2 + 5 + 0 + 5 + 1 + 0 + 0 + 2 + 5 + 9 + 1 + 4 + 0 + 4 + 4) \pmod{10} &= 0 \end{aligned}$$

This scheme detects all single-digit errors as well as all adjacent transpositions except for 0 9 and 9 0. Thus it catches 97.777% of all adjacent transposition errors (88 out of 90).