Modular numbers and Error Correcting Codes

- Introduction
- Modular Arithmetic
- Finite fields
- n-space over a finite field
- Error correcting codes
- Exercises

§ Introduction.

Data transmission is not normally perfect; errors can be introduced because of misread data or poor transmission conditions. The purpose of error-correcting codes is to eliminate these errors, allowing near-perfect transmission of data under imperfect conditions. For example, a scratched LP record, with no error correction, sounds poor. In contrast, a scratched CD still sounds perfect, because the CD player can tell that the scratched data is wrong and reconstruct the correct data as long as the scratch isn't too long. Other applications include satellite transmission, in which the signal is very weak and is likely to include errors; and computer networks, in which error correction allows for faster data transmission while avoiding the decrease in reliability that it would normally cause.

The theory of error-correcting codes uses techniques from many different areas of mathematics, but its basis is in linear algebra. To develop this theory, we need to use linear algebra over sets other than the real numbers $\mathbb{R}$. Except for the geometric transformations, the only properties of the real numbers we have used are addition, subtraction, multiplication, and division. Therefore, we can do linear algebra over any set on which we have these operations. Any such set is called a *field.*

§ Modular Arithmetic.

For every positive integer m larger than 1, we can form what is called a modular number system. In the simplest case of $m = 2$, the modular number system is nothing more than the notion of even and odd. This system of number is sometimes called the *binary numbers* with the even

numbers denoted as $\overline{0}$ and the odd numbers denoted as $\overline{1}$. We are all very familiar with the rules for addition and multiplication of binary numbers numbers. They are

$+$	$\overline{0}$	$\overline{1}$
$\overline{0}$	$\overline{0}$	$\overline{1}$
$\overline{1}$	$\overline{1}$	$\overline{0}$

$\bullet$	$\overline{0}$	$\overline{1}$
$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$

The idea of modular arithmetic is the following. We divide up all the integers into m groups according to what their remainder is when we divide by m. For instance, when $m = 3$, the three groups of integers are

the integers with remainder 1 $\{\ldots, -8, -5, -2, 1, 4, 7, 10, \ldots\}$,
the integers with remainder 2 $\{\ldots, -7, -4, -1, 2, 5, 8, 11, \ldots\}$,
the integers with remainder 0 $\{\ldots, -9, -6, -3, 0, 3, 6, 9, \ldots\}$

For convenience we label the three sets as $\overline{1}_3$, $\overline{2}_3$ and $\overline{0}_3$ and call these three sets the integers 1, 2 and 0 modulo 3. In this notation, the even integers would be $\overline{0}_2$ (the integers 0 modulo 2), while the odd integers would be $\overline{1}_2$ (the integers 1 modulo 2). The modular number system for the integer 3 consists of the sets $\overline{1}_3$, $\overline{2}_3$ and $\overline{0}_3$ with addition and multiplication defined by

$+$	$\overline{0}_3$	$\overline{1}_3$	$\overline{2}_3$
$\overline{0}_3$	$\overline{0}_3$	$\overline{1}_3$	$\overline{2}_3$
$\overline{1}_3$	$\overline{1}_3$	$\overline{2}_3$	$\overline{0}_3$
$\overline{2}_3$	$\overline{2}_3$	$\overline{0}_3$	$\overline{1}_3$

$\bullet$	$\overline{0}_3$	$\overline{1}_3$	$\overline{2}_3$
$\overline{0}_3$	$\overline{0}_3$	$\overline{0}_3$	$\overline{0}_3$
$\overline{1}_3$	$\overline{0}_3$	$\overline{1}_3$	$\overline{2}_3$
$\overline{2}_3$	$\overline{0}_3$	$\overline{2}_3$	$\overline{1}_3$

An example of what this means; if a is an integer with remainder 1 when divided by 3 (so a $\in \overline{1}_3$), and if b is an integer with remainder 2 when divided by 3 (so $b \in \overline{2}_3$), then the sum $a+b$ will be in the set $\overline{0}_3$, i.e. $a+b$ is divisible by 3. Similary, the product $a \cdot b$ will be in the set $\overline{2}_3$. The modular number system for 3 has just three elements. The modular number $\overline{0}_3$ is zero in the sense that when we add it to any other modular number a ($\overline{1}_3$, $\overline{2}_3$ or $\overline{0}_3$), we get a. In a likewise manner, the modular number $\overline{1}_3$ has the property that when we multiply any modular number b by $\overline{1}_3$, we get back

b. The addition and multiplication tables for the modular number system for $m = 4$ are

$+$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{1}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{0}$
$\overline{2}$	$\overline{2}$	$\overline{3}$	$\overline{0}$	$\overline{1}$
$\overline{3}$	$\overline{3}$	$\overline{0}$	$\overline{1}$	$\overline{2}$

$\bullet$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{2}$	$\overline{0}$	$\overline{2}$	$\overline{0}$	$\overline{2}$
$\overline{3}$	$\overline{0}$	$\overline{3}$	$\overline{2}$	$\overline{1}$

Here, we have removed the subscript 4 for convenience. The entries in the addition table are very regular. There is a cyclic shift as we move from row to row. The entries in the multiplication table are less predictable. Note that $\overline{2}$ times $\overline{2}$ is $\overline{0}$, so the $m = 4$ number system is somewhat strange in that two nonzero numbers can multiply up to zero.

Question. The modular numbers for $m = 12$ and $m = 24$ are very commonly used by people throughout the world. How?
Answer. Clocks 'run' on either the modular system for $m = 12$, twelve hours, or $m = 24$, 24 hour clocks.

The multiplication table for the modular number system for the integer $m = 7$ is

$\bullet$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$
$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$
$\overline{2}$	$\overline{0}$	$\overline{2}$	$\overline{4}$	$\overline{6}$	$\overline{1}$	$\overline{3}$	$\overline{5}$
$\overline{3}$	$\overline{0}$	$\overline{3}$	$\overline{6}$	$\overline{2}$	$\overline{5}$	$\overline{1}$	$\overline{4}$
$\overline{4}$	$\overline{0}$	$\overline{4}$	$\overline{1}$	$\overline{5}$	$\overline{2}$	$\overline{6}$	$\overline{3}$
$\overline{5}$	$\overline{0}$	$\overline{5}$	$\overline{3}$	$\overline{1}$	$\overline{6}$	$\overline{4}$	$\overline{2}$
$\overline{6}$	$\overline{0}$	$\overline{6}$	$\overline{5}$	$\overline{4}$	$\overline{3}$	$\overline{2}$	$\overline{1}$

For any modular number system, the familiar properties of

COMMUTATIVITY	$a + b = b + a$
	$a \cdot b = b \cdot a$
ASSOCIATIVITY	$a + (b + c) = (a + b) + c$

	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
ZERO	$a + 0 = a$
ONE	$a \cdot 1 = a$
DISTRIBUTIVITY	$a \cdot (b + c) = a \cdot b + a \cdot c$

are all valid. The *negative* of a modular number such as

$$\overline{2}_3 = \{\dots, -7, -4, -1, 2, 5, 8, 11, \dots\}$$

is the modular number which as a set is equal to

$$\overline{1}_3 = \{\dots, 7, 4, 1, -2, -5, -8, -11, \dots\}.$$

The biggest difference between modular numbers systems and the more familar real or complex number systems was already mentioned in the above paragraph. In some modular systems, eg those for the integers 4 or 6 or 8 or 9, it is possible for two nonzero numbers to multiply to zero. For example, in the modular system of the integer 15, we have $\overline{6}_{15} \cdot \overline{10}_{15} = \overline{0}_{15}$. In this setting, neither of the equations $\overline{6}_{15} \cdot x = \overline{1}_{15}$ nor $\overline{10}_{15} \cdot x = \overline{1}_{15}$ have solutions. In other words, both the modular numbers $\overline{6}_{15}$ and $\overline{10}_{15}$ do not have *inverses* in the modular 15 system. To see why this is so, suppose that $x = a$ is a solution to $\overline{6}_{15} \cdot x = \overline{1}_{15}$. Multiplying both sides of $\overline{6}_{15} \cdot \overline{10}_{15} = \overline{0}_{15}$ gives

$$\begin{aligned} a \cdot (\overline{6}_{15} \cdot \overline{10}_{15}) &= a \cdot \overline{0}_{15} = \overline{0}_{15} \\ (a \cdot \overline{6}_{15}) \cdot \overline{10}_{15} &= \overline{0}_{15} \\ \overline{10}_{15} &= \overline{0}_{15} \end{aligned}$$

This is not true, so the assumption that $\overline{6}_{15} \cdot x = \overline{1}_{15}$ has a solution ($\overline{6}_{15}$ has an inverse in the modular numbers) is wrong.

However, if we look at the multiplication table for both the modular 3 and modular 7 systems, we see that in these two number systems, every nonzero number has an inverse. In the modular 7 system, the numbers $\overline{2}_7$, $\overline{4}_7$ are inverses of each other. The numbers $\overline{3}_7$, $\overline{5}_7$ are inverse of each other. The numbers $\overline{1}_7$ and $\overline{6}_7$ are are their own inverses.

$$\overline{2}_7 \cdot \overline{4}_7 = \overline{1}_7 \qquad \overline{3}_7 \cdot \overline{5}_7 = \overline{1}_7 \qquad \overline{1}_7 \cdot \overline{1}_7 = \overline{1}_7 \qquad \overline{6}_7 \cdot \overline{6}_7 = \overline{1}_7$$

§ **Finite fields.**

A modular system in which every nonzero number has an inverse is called a *finite field*. A *field* is any number system in which the one can add and multiply two numbers with the commutativity, associativity, zero, one, distributivity properties and in which every nonzero number has an inverse. The familar real numbers are a field. The complex numbers are another example of a field. The real numbers are denoted $\mathbb{R}$ and the complex numbers are denoted $\mathbb{C}$. The rational numbers (denoted $\mathbb{Q}$) are yet another example of a field. The modular number system for the integer 3 is denoted $\mathbb{F}_3$, while the modular number system for the integer 7 is denoted $\mathbb{F}_7$. Modular number system for the integer 2 is also a field and denoted $\mathbb{F}_2$. The fields $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Q}$ are fields with infinitely many elements/numbers. The fields $\mathbb{F}_2$, $\mathbb{F}_3$ and $\mathbb{F}_7$ are fields which have only a finite number of elements/numbers in them.

Finite Field Theorem. *If p is any prime integer, the modular numbers for p is denoted $\mathbb{F}_p$. These modular numbers form a finite field with p elements/numbers.*

To see why the modular numbers $\mathbb{F}_p$ form a field, we must find a way to show that any nonzero modular number has an inverse. One easy way when the prime p is not too large is to write out the complete multiplication table for $\mathbb{F}_p$ and then check that every nonzero modular number has an inverse. When the prime p writing out the multiplication table is extremely tedious. Luckily it is not necessary. There is an algorithm called the Euclidean algorithm which can be used to determine the inverse. For more about this, see the appendix.

§ **Linear algebra over a finite field.**

Regular n-space (written in row form) $\mathbb{R}^n$ consists of the row 'vectors' $v = (x_1, \ldots, x_n)$. That is $\mathbb{R}^n$ is the set of all possible size n row vectors, with the coordinates x_i being real numbers. Complex n-space, written $\mathbb{C}^n$ is the set of all row vectors $w = (z_1, \ldots, z_n)$, where the coordinates z_i are complex numbers. Similary if the coordinates are just rational numbers, we speak of rational n-space. Having just described finite fields $\mathbb{F}_p$, we can now speak of n-space $\mathbb{F}_p^n$ over a finite field. It consists of all the row vectors

$v = (x_1, \ldots, x_n)$ with the condition that the coordinates belong to the finite field $\mathbb{F}_p$. Note that n-space over a finite field has just finitely many vectors. Indeed, each of the n coordinates can be any of p possible things; therefore $\mathbb{F}_p^n$ consists of p^n vectors. For example, 7-dimensional space over the binary field $\mathbb{F}_2$ has $2^7 = 128$ vectors. For any n-space over a field:

- We add two vectors v and v' by adding their corresponding coordinates.
- We multiply a vector v by a scalar α in the field by multiplying all the coordinates of v by α.

In a similar manner, we can talk about an $m \times n$ matrix with entries in a finite field. An $m \times n$ matrix A is a linear transformation. If we think of our m-space as being row vectors, then A takes row vectors in $\mathbb{F}_p^m$ to row vectors in $\mathbb{F}_p^n$. If we think of our n-space as being column vectors, then A takes column vectors in $\mathbb{F}_p^n$ to column vectors in $\mathbb{F}_p^m$. Over the finite field $\mathbb{F}_{11}$, the matrix

$$A = \begin{bmatrix} \overline{4} & \overline{1} \\ \overline{6} & \overline{7} \\ \overline{9} & \overline{10} \end{bmatrix}$$

is the linear transformation which would take the the column vector $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ to

$$A \cdot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \overline{4} \cdot y_1 + \overline{1} \cdot y_2 \\ \overline{6} \cdot y_1 + \overline{7} \cdot y_2 \\ \overline{9} \cdot y_1 + \overline{10} \cdot y_2 \end{bmatrix}.$$

As another, example of matrices over finite fields, we can check that in the finite field $\mathbb{F}_5$

$$\begin{bmatrix} \overline{1} & \overline{2} \\ \overline{3} & \overline{4} \end{bmatrix} \cdot \begin{bmatrix} \overline{3} & \overline{1} \\ \overline{4} & \overline{2} \end{bmatrix} = \begin{bmatrix} \overline{1 \cdot 3 + 2 \cdot 4} & \overline{1 \cdot 1 + 2 \cdot 2} \\ \overline{3 \cdot 3 + 4 \cdot 4} & \overline{3 \cdot 1 + 4 \cdot 2} \end{bmatrix} = \begin{bmatrix} \overline{1} & \overline{0} \\ \overline{0} & \overline{1} \end{bmatrix}.$$

that is the two matrices on the left side are each other's inverses.

§ Error correcting codes (The first Hamming code)

As an example of the use of matrices over finite field, we discuss the *first Hamming code*. Data transmitted or stored electronically is usually done so in packets of 1's and 0's. The packet (word) size is usually of fixed size. For instance, a word size of 4 would be able to take on 16 word states ranging from 0000, 0001, 0010, ... to 1111. Imagine now sending words of length 4 over a serial communications channel and that there is some probability that each bit we transmit will be changed either from 0 to 1 or vice verse. If the probability is .95 that a bit is transmitted successfully over a communication channel, then the probabilty that a word consisting of 4 bits is transmitted successfully is

$$\text{sending 4 bits with no errors} = .95 \cdot .95 \cdot .95 \cdot .95 = .8145$$

One way to improve things is to add an extra check or parity bit. Adding a check bit means sending a 5th bit which is the sum (in the binary field $\mathbb{F}_2$ of the original 4 bits. It allows the receiver to check whether or not an error has occurred during transmission. If the received 5 bits do not have the property that they add up to 0, then at least one bit, possibly more, was altered during transmission. Note that this scheme has the drawback that if 2 bits were changed the receiver would be unable to recognize this error. The probability of various possibilities for the transmission of 5 bits (assuming the .95 as before) are

$$\begin{aligned}
&\text{sending 5 bits with no errors } .95^5 = 0.7738\\
&\text{sending 5 bits with at most one error } \ .95^5 + 5 \cdot .95^4 \cdot .05 = .9774\\
&\text{sending 5 bits with two or more errors } \ 1 - .9774 = .0226
\end{aligned}$$

While adding a parity check bit allows a receiver to *detect* some errors, it does not allow the receiver to fix things. The first Hamming code is a way of coding a 4 bit word into a 7 bit word so that if just one error occurs then the receiver will not only be able to detect the error but the receiver will be able to correct the error. Such a method/code is called an *error correcting* code. Since

$$\begin{aligned}
&.8145 = \text{probablity of transmitting 4 bits with error}\\
&.9556 = \text{probablity of transmitting 7 bits with at most one error}
\end{aligned}$$

It is more reliable to code a 4 bit word into a 7 bit word and then send the 7 bit word. In order to measure the ability of a code to correct errors, we define the *Hamming distance* between two vectors $\vec{v}_1$ and $\vec{v}_2$ in $\mathbb{F}_2^n$ to be the number of bits which are different in the two vectors. This is thus the number of errors which would need to be made for a signal to be transmitted as $\vec{v}_1$ and read as $\vec{v}_2$. There are $2^7 = 128$ possible 7 bits words, or equivalently, there are vectors in 128 vectors in the 7-dimensional vector space $\mathbb{F}_2^7$. The number of 7 bit words within a Hamming distance of 1 or 0 from a word is 1+7=8.

Question. Is there some clever way/process to code a 4 bit word into a 7 bit word?
Answer. The first Hamming code.

The first Hamming code is a 4-dimensional subspace W of $\mathbb{F}_2^7$ with the property that the Hamming distance between any two vectors in W is at least 3. The description of the subspace W is that it is **both** the kernel of the 'testing' matrix

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

as well as the image of the 'encoding' matrix

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

The columns of the encoding matrix E are obtained by finding a basis for the kernel of T via reduced row echelon form.

To code a 4 bit word $\vec{v} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ into a 7 bit vector in the Hamming code (subspace W) we multiply $\vec{v}$ into the encoding matrix E to get the

encoded/transformed vector $\vec{w} = E(\vec{v}) = E \cdot \vec{v}$. We then transmit $\vec{w}$ via our communication channel.

At the other end of the communication channel, a person who receives a 7 bit word encoded $\vec{w}$ in this fashion, can first multiply $\vec{w}$ into the testing matrix T to check if there have been any errors in transmission.

- If $\vec{w}$ is received with no errors, then $T \cdot \vec{w}$ will be equal to $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ since the Hamming subspace is the kernel of the testing matrix T.
- If $\vec{w}$ has just one error in it, then the size three column vector $T \cdot \vec{w}$ has the remarkable property, which we cannot fully explain here, that it tells us (in binary) which column the error is located.

To decode an encoded vector, one uses a decoding transformation $D : \mathbb{F}_2^7 \longrightarrow \mathbb{F}_2^4$ which changes the 7 bit encoded word back to the original 4 bit word. The decoding matrix must satisfy the property that

$$D \cdot E = \text{Identity } 4 \times 4 \text{ matrix}$$

For the first Hamming code, the decoding matrix is

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

As an exercise, check that $D \cdot E = I_{4\times 4}$.

Example: Suppose we want to transmit the 4 bit word $\vec{v} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$. Computation of $\vec{w} = E(\vec{v})$ gives $\vec{w} = E \cdot \vec{v} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$. If during transmission, the code

word is altered to $\vec{w}' = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when we multiply it into the testing matrix T we get

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

We now read the column vector $\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ as the base two number $110 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 4 + 2 = 6$, which means the 6th bit has been altered. Changing the 6th bit corrects the error caused in transmission.

Error-correcting codes are of practical use because they greatly decrease the probability of an error without greatly lengthening the transmission. The design of error correcting codes depends very much on linear algebra as well as much more sophisticated mathematics beyond the scope of this course. An example of a much more sophisticated code is a code known as the Golay code. The Golay code is a 12 dimensional subspace of 23 dimensional binary space $\mathbb{F}_2^{23}$. In analogy with the first Hamming code, there is an encoding 23×12 matrix E which encodes a 12 bit word into a 23 bit word. The image of the matrix E in $\mathbb{F}_2^{23}$ is the 12 dimensional Golay code. There is also a testing matrix T whose kernel is the Golay code as well as a decoding matrix (a linear transformation from $\mathbb{F}_2^{23}$ to $\mathbb{F}_2^{12}$). The Golay code, which is used both in satellite as well as spacecraft data transmission, has the ability to detect up to 6 bit changes in a code word and to correct code words which have been altered in 3 bits or less.

Suppose that each bit has a .01 chance of being transmitted incorrectly.

The chance that four bits will be transmitted correctly is thus $.99^4 = .9606$. If we encode them with the first Hamming code, the chance is $.99^7 = .9320$ that all seven bits will be correct, and $7 \cdot .01 \cdot .99^6 = .0659$ that there is only one error (since any one of seven bits can be the wrong one), for a chance of .9979 that four bits will be transmitted correctly. This is 19 times more reliable. The probability of no error in 23 bits is $.99^{23} = .793614$, The probabililty of one error is $23 \cdot .01 \cdot .99^{22} = .184375$. If there are two errors, there are 23 possible bits which can be one wrong bit, and 22 possibilities which are another wrong bit, but each pair of wrong bits is being counted twice (bits 2 and 8 being wrong is the same as bits 8 and 2 being wrong), so there are $23 \cdot 22/2$ ways to have two wrong bits, and the total probability of two wrong bits is $(23 \cdot 22/2) \cdot .01^2 \cdot .99^{21} = .020486$. Similarly, the probability of three wrong bits is $(23 \cdot 22 \cdot 21/6) \cdot .01^3 \cdot .99^{20} = .001449$. Adding these together gives a probability of .999924. For comparison, the chance is .9939 that the same 12 bits will be correct if they are transmitted as three groups of four using the first Hamming code; so the Golay code cuts the errors by an additional factor of 80 using only two more bits (23 bits for the Golay code vs 21 bits for three copies of the first Hamming code).

§ **Exercises.**

1. Write out the multiplication table for the **modular numbers** (finite field) $\mathbb{F}_7$
2. Solve the equation $\overline{4}x = \overline{5}$ in the finite field $\mathbb{F}_7$
3. Find the inverse matrix of

$$A = \begin{bmatrix} 3 & 3 \\ 4 & 2 \end{bmatrix}$$

where all entries are in the modular numbers $\mathbb{F}_7$. Solve the linear system

$$3x_1 + 3x_2 = 4$$
$$4x_1 + 2x_2 = 1$$

for numbers x_1 and x_2 in $\mathbb{F}_7$.

4. Choose a 4-bit word $\vec{v}$ in $\mathbb{F}_2^4$ code it via the encoding matrix E into a 7-bit Hamming code word $\vec{w}$. Verify that $\vec{w}$ lies in the kernel of the testing matrix T. Then change one of the bits of $\vec{w}$ and verify that the testing matrix T allows one to determine which bit was changed.
5. Suppose you have received the code vector (encoded as a 7-bit Hamming code word)

$$\vec{w} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Find the transmitted vector $\vec{v}$, correcting an error if necessary, and then find the original 4 bit vector $\vec{u}$.

6. Write out the multiplication table for the **modular numbers** $\mathbb{F}_5$ and determine a matrix E (entries in $\mathbb{F}_5$) whose image is equal to the kernel of the matrix

$$T = \begin{bmatrix} 2 & 0 & 0 & 1 & 1 \\ 3 & 1 & 0 & 1 & 1 \\ 4 & 2 & 1 & 1 & 1 \end{bmatrix}$$

7. How many vectors are there in the vector space $\mathbb{F}_7^5$? If W is a subspace of $\mathbb{F}_7^5$ of dimension 3 how many vectors are in W?

§ Appendix. Euclidean Algorithm

The Euclidean algorithm is a procedure for finding the greatest common divisor $\gcd(a, b)$ of two nonzero integers a and b. When the $\gcd(a, b) = 1$, the steps of the Euclidean algorithm can be 'reversed' to find inverse of the number $\bar{b}$ in the modular system of the number a.

Euclidean Algorithm. *If $a > b$ are two positive integers, then their greatest common divisor $gcd(a, b)$ can be obtained as follows. Divide b into a to get a quotient and remainder term*

$$a = q_1 b + r_1 \qquad \text{the remainder } r_1 \text{ is between } 0 \text{ and } b - 1$$

If $r_1 = 0$, then b divides a and $gcd(a, b) = b$. If $r_1 > 0$ then we divide r_1 into b to get

$$b = q_2 r_1 + r_2 \qquad \text{the remainder } r_2 \text{ is between } 0 \text{ and } r_1 - 1$$

If $r_2 = 0$, then r_1 divides a and $gcd(a, b) = r_1$. If $r_2 > 0$ then we divide r_2 into r_1 to get

$$r_1 = q_3 r_2 + r_3 \qquad \text{the remainder } r_3 \text{ is between } 0 \text{ and } r_2 - 1$$

If $r_3 = 0$, then r_2 divides a and $gcd(a, b) = r_3$. If $r_3 > 0$ then we continue on. The process will eventually stop since the remainders are getting smaller ($b > r_1 > r_2 > \cdots > r_k$) but still larger than 0.

As an example of the process, we find the greatest common divisor between the integers 319 and 100. We have

$$\begin{aligned} 319 &= 3 \cdot 100 + 19 \\ 100 &= 5 \cdot 19 + 5 \\ 19 &= 3 \cdot 5 + 4 \\ 5 &= 1 \cdot 4 + 1 \\ 4 &= 4 \cdot 1 \end{aligned}$$

So, the greatest common divisor of 319 and 100 is 1. In this case (when the greatest common divisor is 1), the Euclidean algorithm provides us a way of

determining the inverse of the modular number $\overline{100}_{319}$. We just *reverse* the steps of the Euclidean algorithm starting with the last ($r = 1$) remainder. We have

$$\begin{aligned}
1 &= 5 - 4 \\
&= 5 - (19 - 3 \cdot 5) = -19 + 4 \cdot 5 \\
&= -19 + 4 \cdot (100 - 5 \cdot 19) = 4 \cdot 100 - 21 \cdot 19 \\
&= 4 \cdot 100 - 21 \cdot (319 - 3 \cdot 100) \\
&= -21 \cdot 319 + 67 \cdot 100
\end{aligned}$$

In the modular system 319, this becomes

$$\begin{aligned}
\overline{1}_{319} &= \overline{-21}_{319} \cdot \overline{319}_{319} + \overline{67}_{319} \cdot \overline{100}_{319} \\
\overline{1}_{319} &= \overline{0}_{319} + \overline{67}_{319} \cdot \overline{100}_{319}
\end{aligned}$$

The inverse of $\overline{100}_{319}$ in the $m = 319$ modular system is $\overline{67}_{319}$.

For a prime p, the process of the Euclidean algorithm can be used to find the inverse of any nonzero numbers in a modular number system. It is why the modular number system for a prime p is a field.

Many of the things we do in the real field $\mathbb{R}$ can be done in a finite field as well. For instance, in the finite field $\mathbb{F}_7$, we can solve system of linear equations

$$\begin{array}{rcrcl}
\overline{3}x_1 & + & \overline{3}x_2 & = & \overline{4} \\
\overline{4}x_1 & + & \overline{2}x_2 & = & \overline{1}
\end{array}$$

by Gaussian elimination. We need to divide eliminate the variable x_1 from the 2nd equation by adding a multiple of the first equation. To do this we need to find a modular number $\overline{?}$ so tha $\overline{?} \cdot \overline{3} = \overline{4}$. From the multiplication table for $\mathbb{F}_7$, or via the Euclidean algorithm, we find $\overline{5} \cdot \overline{3} = \overline{1}$, so it must be the case that $\overline{4} \cdot \overline{5} = \overline{20} = \overline{6}$ is the modular number we need to take for $\overline{?}$. Doing the elimination gives

$$\begin{array}{rcrcl}
\overline{3}x_1 & + & \overline{3}x_2 & = & \overline{4} \\
(\overline{4} - \overline{6} \cdot \overline{3})x_1 & + & (\overline{2} - \overline{6} \cdot \overline{3})x_2 & = & \overline{1} - \overline{6} \cdot \overline{4}
\end{array}$$

which is

$$\begin{array}{rcrcl}
\overline{3}x_1 & + & \overline{3}x_2 & = & \overline{4} \\
 & & \overline{5}x_2 & = & \overline{5}
\end{array}$$

We see that $x_2 = \overline{1}$. If we now perform the back substitution of $x_2 = \overline{1}$ into the first equation, we get $\overline{3}x_1 = \overline{1}$ and so $x_1 = \overline{5}$. As a check, we substitute $x_1 = \overline{5}$ and $x_2 = \overline{1}$ into the original two equations. We have

$$\begin{array}{rcl} \overline{3} \cdot \overline{5} + \overline{3} \cdot \overline{1} & = & \overline{1} + \overline{3} \quad = \quad \overline{4} \\ \overline{4} \cdot \overline{5} + \overline{2} \cdot \overline{1} & = & \overline{6} + \overline{2} \quad = \quad \overline{1} \end{array}$$

§ **Exercises.**

1. Find the inverse of $\overline{105}$ in the finite field $\mathbb{F}_{1997}$ using the Euclidean algorithm.