DIMACS
Center for Discrete Mathematics &
Theoretical Computer Science

DIMACS EDUCATIONAL MODULE SERIES

MODULE 08-2
Reed-Solomon Codes: An Application of Linear Algebra
Date Prepared: May 2008

Steven Leonhardi
Winona State University, Winona, Minnesota 55987
sleonhardi@winona.edu

Lidia Luquet
Saint Mary's College of California, Moraga, CA 94575
lluquet@stmarys-ca.edu

Jim Sauerberg
Saint Mary's College of California, Moraga, CA 94575
jsauerbe@stmarys-ca.edu

DIMACS Center, CoRE Bldg., Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018
TEL: 732-445-5928 • FAX: 732-445-5932 • EMAIL: center@dimacs.rutgers.edu Web: http://dimacs.rutgers.edu/

Founded as a National Science Foundation Science and Technology Center and a Joint Project of Rutgers University, Princeton University, AT&T Labs - Research, Bell Labs, NEC Laboratories America and Telcordia Technologies with affiliated members Avaya Labs, Georgia Institute of Technology, HP Labs, IBM Research, Microsoft Research, Rensselaer Polytechnic Institute, and Stevens Institute of Technology.

Module Description Information

- **Title:**

  Reed-Solomon Codes: An Application of Linear Algebra

- **Author(s):**

1. Steven Leonhardi, Winona State University, Winona, Minnesota 55987
sleonhardi@winona.edu

2. Lidia Luquet, Saint Mary's College of California, Moraga, CA 94575
lluquet@stmarys-ca.edu

3. Jim Sauerberg, Saint Mary's College of California, Moraga, CA 94575
jsauerbe@stmarys-ca.edu

- **Abstract:**

This module provides a brief introduction to the theory of error-detecting and error-correcting codes, with special emphasis on the Reed-Solomon codes. Several different methods of defining codes and decoding are presented, and the parameters of these codes and their significance in practice are discussed. Simple, concrete examples are studied first before more general families of codes are considered. Throughout the module, concepts and results from Linear Algebra and (to a lesser degree) Abstract Algebra are used to define and analyze these codes.

- **Informal Description:**

In this module we discuss how one can use linear algebra and finite fields to define and analyze the performance of various error-detecting and error-correcting codes used in Coding Theory, with special attention given to Reed-Solomon codes. These codes are widely used in the production of compact discs (CDs) and digital video disks (DVDs), as well as in cell phone and satellite communications. The module presents examples of Reed-Solomon codes that have smaller dimensions than those used in industry, to simplify the computations and analysis, but which share the same properties as widely used codes. Each section of the module includes exercises for the reader to test his or her understanding and to further explore these codes.

- **Target Audience:**

This module is aimed at the undergraduate students who are studying Linear Algebra or Abstract Algebra, or who are studying Coding Theory as an independent project. Most students registered in these courses are at the sophomore, junior, or senior level.

- **Prerequisites:**

Students are assumed (after some review as provided in the appendix) to be familiar with basic concepts from a linear algebra course such as matrix operations, independent and dependent vectors, span, basis vectors, vector space, dimension, null space, column space, and rank. Finite fields are used, but no previous exposure to finite fields is required.

- **Mathematical Field:**

Linear Algebra, Abstract Algebra.

- **Application Areas:**

Coding Theory

- **Mathematics Subject Classification:**

MSC (2000): 11T71, 94B05, 94B20, 94B35

- **Contact Information:**

Steven Leonhardi, Winona State University, Winona, Minnesota 55987
sleonhardi@winona.edu

- **Other DIMACS modules related to this module:**

None

REED-SOLOMON CODES: AN APPLICATION OF LINEAR ALGEBRA

S. LEONHARDI, L. LUQUET, AND J. SAUERBERG

1. Linear Codes

Who uses linear algebra? Anyone who enjoys listening to compact disks, watching DVD's, talking over cellular telephones, or surfing the internet over a high-speed modem. The reliability of satellite communication, digital television, and all of the other storage and communication devices mentioned, depends in part on the use of error-correcting codes, whose design and usefulness in turn is founded on basic applications of linear algebra and abstract algebra. Check digits, which are used in UPC codes, ISBN numbers, and driver's licenses, also rely on concepts from coding theory.

The goal of algebraic coding theory is to reduce or eliminate any errors that are introduced in a message when that message is transmitted. Redundancy is added to the original message to produce an **encoded message** consisting of **codewords**. The transmission of the encoded message may (and usually does) introduce errors, caused, perhaps, by static electricity or a blot on the compact disc. As a consequence, the received word may be different from the sent codeword. A **decoder** then determines an estimate of the sent codeword, detecting and correcting errors to the extent made possible by the redundancy that was added to the message by the **encoder**. From the estimate of the sent codeword, the decoder can estimate the message that was originally sent. The sequence of steps in this process may be depicted as follows.

$$\text{Message} \to \text{Encoded Message} \to \text{Sent Codewords} \to \text{Received Words} \to \text{Estimate of Sent Codewords} \to \text{Estimate of Message}$$

The goal of this module is to give a very brief introduction to the subject of algebraic coding theory, with emphasis on Reed–Solomon codes, which are among the most commonly used codes. Section 1 gives an introduction to a few simple codes, including our first example of a Reed–Solomon code. Section 2 gives additional standard methods of defining codes and decoding received messages, and gives a more general method for defining Reed–Solomon codes. Section 3 gives a third approach to Reed–Solomon codes, and develops a deeper understanding of the structure of these codes by exploring their algebraic properties in greater detail.

The reader is assumed to be a student in a first course in Linear Algebra or Abstract Algebra who has seen vector spaces over the real numbers. The Appendix offers a review of the necessary prerequisite material, and introduces the concept of a finite field. The results stated for vector spaces over finite fields follow closely those for real vector spaces. The later sections require greater mathematical maturity, but all three sections are intended to be accessible to anyone who has progressed in a Linear Algebra course up to the concept of a linear transformation and who has reviewed the material in the Appendix.

1.1. **Linear Codes and Reed–Solomon Codes.** To create a **code** we must choose a set S of symbols and then specify the set C of **codewords** that will make up the code. If $S = \{0, 1, 2\}$, a code word could be 011201. When this codeword is transmitted through a **channel**, such as a telephone line, it may arrive as 011201 or it may arrive as 011221, i.e., with an **error**. In order to mathematically represent words that are sent or received, we consider them as vectors selected from a vector space V over a field F. The reader is probably most familiar with vector spaces that are defined over infinite fields such as the real numbers $\mathbf{R}$ or the rationals $\mathbf{Q}$. However, since the messages we work with will always be formed from a finite set of symbols, we will find it more appropriate to work over a finite field such as $\mathbf{Z}_2$ or $\mathbf{Z}_7$.

In general, a code C consists of a set of codewords. For example,

$$C_1 = \{\text{NN}, \text{NE}, \text{EE}, \text{SE}, \text{SS}, \text{SW}, \text{WW}, \text{NW}\}$$

is a code that uses the English alphabet as symbols, while $C_2 = \{0001, 0101, 1011, 0011, 1001\}$ is a code with symbols in $\mathbf{Z}_2$. Each word of the code is sent through a channel. Assume we have adopted C_2 and suppose the codeword 0101 is sent through. If the word that arrives at the other end of the channel, i.e., **the received word** is:

(1) 0101, we would conclude that the word sent was 0101, as this word is in our code. In other words, we are able to decode the message.
(2) 1111, we know there has been an error in the transmission because the word received is not in the code. We would conclude that the word sent was 1011 because it is the only word in the code that differs from 1111 in only one digit. (Here we are using an informal criterion to decode a message that has arrived with an error: the word sent was most likely the one in the code that differs from the word received in the smallest number of places.)
(3) 0111, we would know there has been an error in the transmission because that word is not in the code. We would not be able to decide whether the original word was 0101 or 0011 since both differ by a single digit from the word received. So we could not decode the received message. That is, although we would detect an error, we would not be able to correct it.

(4) 0011, we would mistakenly accept this as the sent word, since it is a codeword. Note that when using this code, we can detect any one-digit error, but cannot always detect a transmission with errors in two digits.

The goal of coding theory is to design codes that allow for the efficient detection, and sometimes correction, of transmission errors. In what follows we will consider only codes that are also vector spaces.

Definition 1.1. *A **linear code** C over the field $\mathbf{Z}_p$ is a subspace of the vector space $\mathbf{Z}_p^n$ for some n. If, as a subspace, C has dimension k, we say C is an $[n,k]_p$ code. The numbers n and k are the **length** and **dimension** of the code, respectively, and together constitute two **parameters** of the code C. A **codeword**, or more simply, a **word** in an $[n,k]_p$ code is a string of n characters taken from $\mathbf{Z}_p$.*

Intuitively, the reader may want to think of the n characters in a codeword as consisting of k characters of information and $(n-k)$ redundant characters that may be used to detect and possibly correct errors. Such a code is called a **systematic code**. In practice, some common codes are systematic, and other codes are more complicated.

Example 1.1: Consider the vector $(1,0,1)$ in $\mathbf{Z}_2^3$. From now on we will write such a vector as 101. Let C be the subspace generated by 101, i.e., $C = \{101, 000\}$. For this code we have $n = 3$, $k = 1$ and $p = 2$. Thus C is a $[3,1]_2$ code. ◇

Example 1.2: The $[5,1]_2$ code $C = \{00000, 11111\}$ is called the **repetition code of length five**, since each codeword consists of five copies of the same symbol. Only two words can be sent and it is easy to detect and correct up to two errors. If the message 11111 is sent and it arrives as 10110, we detect two errors, correct them and decode it as 11111. (It could be argued that the received word 10110 might have three errors and the original message was 00000. However, we assume that only two errors occurred, since a transmission with two errors is more likely than a transmission with three errors, for any useful channel. In general, we decode received words assuming that fewer errors are more likely than a greater number of errors.) ◇

Since a subspace is determined by a basis, rather than enumerating all the elements in a code C we may specify C by giving its basis.

Example 1.3: The $[7,3]_2$ code (i.e., the 3-dimensional subspace of $\mathbf{Z}_2^7$) generated by the vectors 0001111, 0110011 and 1010101. A convenient way to indicate this code is to create a matrix G whose rows are the given vectors,

$$G = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

for then the code is the **row space** of G, i.e., the subspace denoted $Row(G)$ of $\mathbf{Z}_2^7$ generated by the linear combinations of the rows. This matrix G has rank 3 (look at the determinant of the last three columns of G and use Theorem 4.6). Therefore the rows of G are linearly independent and this code is a subspace of dimension 3 over $\mathbf{Z}_2$. Section 4.3 in the Appendix tells us that this code has $p^k = 2^3 = 8$ codewords. One such codeword is $(0001111) + (1010101) = (1011010)$. ◇

Definition 1.2. *Let C be a linear code with basis B. The matrix G whose rows are the codewords in B is called a* ***generator matrix*** *for C.*

So G is a generator matrix for code C if and only if the rows of G span C, that is, $\mathrm{Row}(G) = C$. Just as a subspace may have many bases, a code may have many generating matrices: any two row equivalent matrices will generate the same code.

Finally we come to our main topic, **Reed–Solomon codes**, which have a generating matrix of a special form.

Example 1.4: The $[6, 4]_7$ Reed–Solomon code has generator matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 2 & 2 & 4 & 1 \\ 1 & 1 & 6 & 1 & 6 & 6 \end{pmatrix}.$$

(Note that when working over the finite field $\mathbf{Z}_7$, the powers in the third and fourth rows are all simplified modulo 7.) The determinant D of the matrix formed by the first four columns of G is a Vandermonde determinant, thus by Theorem 4.7 in the Appendix, $D = (4-3)(4-2)(4-1)(3-2)(3-1)(2-1) \neq 0$. This means the rank of G is 4, so this Reed–Solomon code is a subspace of $\mathbf{Z}_7^6$ of dimension 4 and consists of $p^k = 7^4 = 2401$ codewords. One such codeword is $4(123456) + (111111) + (116166) = (635463)$. ◇

This example may be generalized.

Definition 1.3. *Fix a prime number $p > 2$ and an integer k with $0 \leq k \leq p$. The $[p-1, k]_p$ Reed–Solomon code is the code given by the generator matrix*

$$G = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & p-1 \\ 1^2 & 2^2 & 3^2 & \dots & (p-1)^2 \\ 1^3 & 2^3 & 3^3 & \dots & (p-1)^3 \\ \vdots & \vdots & \vdots & & \vdots \\ 1^{k-1} & 2^{k-1} & 3^{k-1} & \dots & (p-1)^{k-1} \end{pmatrix}.$$

It has p^k codewords.

In the $[6,4]_7$ Reed–Solomon code of Example 1.4 if one of the 7^4 code words is sent, due to possible errors in transmission any of the $7^6 = 117,649$ words in $\mathbf{Z}_7^6$ could arrive. If the received word is not in the code, we compare it to the words in the code, looking for the codeword that differs from the received word in the least number of places. This leads to the following definition.

Definition 1.4. *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be words in* $\mathbf{Z}_p^n$*. The* ***Hamming distance*** *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined to be the number of places in which* $\mathbf{x}$ *and* $\mathbf{y}$ *differ, and is indicated by* $d(\mathbf{x},\mathbf{y})$*.*

Example 1.5: In the $[6,4]_7$ Reed–Solomon code let $\mathbf{x} = 253352$ and $\mathbf{y} = 251330$. Then $d(\mathbf{x},\mathbf{y}) = 3$. ◇

Something called a "distance" should satisfy certain properties, and the Hamming distance does. (Any binary function satisfying all three properties below is called a **metric**.)

We leave the proof of the following theorem to the reader.

Theorem 1.1. *If* d *is the Hamming distance, then for all* $\mathbf{x}$*,* $\mathbf{y}$ *and* $\mathbf{z}$ *in* $\mathbf{Z}_p^n$*, we have*

(1) $d(\mathbf{x},\mathbf{0}) > 0$ *for* $\mathbf{x} \neq 0$*;* $d(\mathbf{0},\mathbf{0}) = 0$*,*
(2) $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$*, and*
(3) $d(\mathbf{x},\mathbf{z}) \leq d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z})$*.*

In Example 1.2 the received word was 10110, which is not a codeword. The distance from it to the two words in the code are $d(10110, 00000) = 3$ and $d(10110, 11111) = 2$. It makes intuitive sense to decode the word as the codeword that was closest to the received message. This criterion for decoding is investigated in the next section.

1.2. **Minimum Distance Decoding.** So far we have associated with each linear code two parameters, n and k. We now introduce a third parameter that will allow us to describe a simple decoding method.

Definition 1.5. *Let* C *be a code with* $k > 0$*, i.e.,* C *has more than one word. The smallest distance* $d = d(C)$ *between different codewords of* C *is called the* ***minimum distance*** *of* C*. When the minimum distance* d *of a code is known, we indicate the parameters of the code as* $[n,k,d]_p$*.*

In the code from Example 1.2 there is only one pair of codewords to consider, and $d(00000, 11111) = 5$, so $d(C) = 5$. Thus the repetition code of length five is a $[5,1,5]_2$ code.

Example 1.6: The linear code $C = \{00000, 01101, 10100, 11001\}$ over $\mathbf{Z}_2$ has basis $\{01101, 10100\}$. Thus it is a $[5,2]_2$ code. To find the minimum distance of C using the definition, we first tabulate the values $d(\mathbf{x},\mathbf{y})$ for $\mathbf{x} \neq \mathbf{y}$ in C:

d	00000	01101	10100	11001
00000	0	3	2	3
01101	3	0	3	2
10100	2	3	0	3
11001	3	2	3	0

Since the smallest non-zero value in the table is 2, $d = 2$ and C is a $[5, 2, 2]_2$ code. $\diamond$

We will use the $[4, 1]_2$ repetition code $C = \{0000, 1111\} \subset \mathbf{Z}_2^4$ to introduce a decoding method. When a word from C is transmitted, the received word may be any word in $\mathbf{Z}_2^4$. A **decoding method** assigns to each (received) word in $\mathbf{Z}_2^4$ a (sent) word in C. We will make this assignment using the notion of distance. The **sphere** in $\mathbf{Z}_2^4$ with center 0000 and radius 1 is defined as usual:

$$S(0000, 1) = \{\mathbf{x} \in \mathbf{Z}_2^4 \mid d(\mathbf{x}, 0000) \leq 1\} \text{ , i.e.,}$$

$S(0000, 1) = \{0000, 10000, 0100, 0010, 0001\}$. Similarly, the sphere with center 1111 and radius 1 is

$$S(1111, 1) = \{1111, 0111, 1011, 1101, 1110\}.$$

These spheres are disjoint since $d(C) = 4$ and the radius is 1. There are 10 vectors in the union of the two spheres (5 vectors in each) and 16 vectors in $\mathbf{Z}_2^4$.

If a codeword $\mathbf{x} \in C$ is sent, it arrives as $\mathbf{x}' \in \mathbf{Z}_2^4$. If $\mathbf{x}'$ is in one of the two spheres, we decode it as the center of that sphere. If it is not in one of the spheres, we declare that we have detected an error, but are not able to decode it. For example,

Word received	Decode as	Reason
1011	1111	$1011 \in S(1111, 1)$
0101	?	$0101 \notin S(1111, 1) \bigcup S(0000, 1)$

Notice that there is a single codeword closest to 1011, allowing us to correct the received word 1011, but there are two different codewords at a distance of 2 from 0101, preventing us from correcting the received word 0101.

This example illustrates an important principle we will use in decoding: when a received word has a nearest codeword then we decode it as that codeword, while those received words that are equidistant from two or more codewords we say cannot be decoded. In general, to implement this method of **minimum distance decoding** we need to determine an integer r such that

(1) r is small enough so that the spheres of radius r centered at the codewords are disjoint, and
(2) r is as large as possible to be able to decode the largest possible number of received words.

In general, the best possible r can be shown to be $r = \lfloor \frac{d-1}{2} \rfloor$, where $d = d(C)$ is the minimum distance of the code and $\lfloor \cdot \rfloor$ denotes the floor function.

For the $[4, 1]_2$ repetition code above $d = 4$ and so $r = \lfloor \frac{4-1}{2} \rfloor = 1$. (Notice that this is best, for if $r = 2$ then 0101 would be in both spheres.) For the code in Example 1.6, $d = 2$ hence $r = \lfloor \frac{2-1}{2} \rfloor = 0$, and, indeed, if $r = 1$ then spheres centered around 01101 and 1101 would overlap.

Minimum distance decoding reveals part of the role played by d in decoding, but to decode this way requires that we calculate the distance of the received word to all the codewords before we decide if and how to decode it. A less time consuming decoding method is presented in the next subsection.

Since d plays a crucial role in any decoding algorithm we present an alternate way of looking at it.

Definition 1.6. *Let C be a code in $\mathbf{Z}_p^n$. The **weight** of a word $\mathbf{x}$ in C is denoted $w(\mathbf{x})$ and is the number of non-zero entries in $\mathbf{x}$. The minimum weight of all the non-zero codewords in C is called the **weight** of C and is denoted $w(C)$.*

Notice that the weight of a word $\mathbf{x}$ is the Hamming distance of $\mathbf{x}$ to the zero vector $\mathbf{0}$. For example, in $\mathbf{Z}_7^6$, $w(160034) = d(160034, 000000) = 4$. A more general relation between d and w follows.

Theorem 1.2. *If C is a linear code, then $d(C) = w(C)$.*

Proof. If $\mathbf{x}$ and $\mathbf{y}$ are vectors in C such that $d(\mathbf{x}, \mathbf{y}) = d(C)$, then $\mathbf{x} - \mathbf{y}$ is a vector in C with exactly $d(C)$ non-zero entries. Thus $w(\mathbf{x}-\mathbf{y}) = d(C)$ and consequently $w(C) \leq d(C)$. To obtain the reverse inequality, let $\mathbf{z}$ be a vector in C such that $w(\mathbf{z}) = w(C)$. Then $d(\mathbf{z}, \mathbf{0}) = w(C)$ and therefore $d(C) \leq w(C)$. □

One way of calculating d is to calculate the weight of all the vectors in a code and look for the minimum weight, as we did in Example 1.6. This can be slow since codes can be very large. For example, a $[11, 9]_7$ code has $7^9 = 40,353,607$ codewords. The Reed–Solomon family of codes have the advantage that one can follow a procedure to construct a code that will have an *a priori* specified value of d as its minimum distance (and hence also as its weight). This will be seen in Section 3.

Having considered a few different examples of codes leads us naturally to the question of what makes a code "good". Since transmission carries a cost in terms of time and money, we want as high a ratio of information to codeword length as is practical. So we would like k to be large relative to n. On the other hand, we would like our codewords to be far apart to better allow us to determine from which codeword each received word came. So we would like d to be large. If we suppose we have a fixed n, perhaps due

to the size of transmission channel, can we satisfy simultaneously both of these desires? A fundamental result in coding theory says that the answer is no. That is, there is an unavoidable tradeoff between the efficiency of information transmission (as measured by k) and error detecting/correcting capability (as dependent upon d).

Theorem 1.3. ***(The Singleton Bound)*** *Let C be a linear code of length n, dimension k, and having minimum distance d over a finite field. Then $d \leq n - k + 1$.*

Proof. Let C be a $[n, k, d]_p$ code. Then C has p^k codewords, each codeword $\mathbf{x}$ consisting of n integers in $\mathbf{Z}_p$:

$$\mathbf{x} = a_1 a_2 a_3 \dots a_{d-1} a_d \dots a_n.$$

If we delete the first $d - 1$ integers from each codeword, we obtain a shorter word $\mathbf{y} = a_d \dots a_n$. These shorter words are all different since $d(C) = d$. Hence there are p^k of them. Now the shorter words have length $n - (d - 1) = n - d + 1$, and the total number of words of this length is p^{n-d+1}. Consequently $p^k \leq p^{n-d+1}$, or $k \leq n-d+1$, i.e., $d \leq n-k+1$. □

Thus (for a fixed n) a larger k will result in a smaller bound for d. Thinking intuitively that an $[n, k, d]_p$ code carries k "information bits" and $n - k$ is "redundancy bits," the theorem indicates that to increase d we must increase the redundancy of the code and thus decrease the amount of information being sent, whereas increasing the information that can be sent reduces the code's ability to detect errors.

1.3. **Decoding by Standard Array.** The method of look-up decoding using spheres presented in the previous section does not make use of the linearity of the code and can even leave some received words undecoded. Ideally we would like a rule that allows us to decode *any* received word. If C is an $[n, k]_p$ code contained in $\mathbf{Z}_p^n$, then a sent codeword $\mathbf{x}$ must be an element of C while a received word $\mathbf{x}'$ can be any element in $\mathbf{Z}_p^n$. In the following example we show how to partition the entire $\mathbf{Z}_p^n$ into p^k disjoint (column) sets, each containing exactly one codeword in C. Each set will contain p^{n-k} vectors. A received word $\mathbf{x}'$ will be in exactly one set (i.e., column) of the partition and it will be decoded as the unique codeword in that set (the column heading). All words will be decoded.

Example 1.7: The vector space $\mathbf{Z}_2^5$ contains the linear $[5, 2]_2$ code

$$C = \{00000, 01101, 10100, 11001\}.$$

We arrange the vectors in $\mathbf{Z}_2^5$ in four columns, each headed by a vector in C. The columns are formed following a criteria that will be explained shortly.

00000	01101	10100	11001
00001	01100	10101	11000
00010	01111	10110	11011
00100	01001	10000	11101
01000	00101	11100	10001
10010	11111	00110	01011
01010	00111	11110	10011
00011	01110	10111	11010

This table is called a **standard array** and contains all possible received words. If, for example, we receive $\mathbf{x}' = 11111$, located in the second column, we decode it as $\mathbf{x} = 01101$, the codeword at the top of that column. While $\mathbf{y} = 11001$ is another codeword a distance 2 from $\mathbf{x}'$, the standard array makes it clear that $\mathbf{x}'$ is to be decoded as $\mathbf{x}$ and not as $\mathbf{y}$. $\diamond$

To construct a standard array for a code $C \subseteq \mathbf{Z}_p^n$:

(1) List the elements of the code $C = \{\mathbf{c}_1 = \mathbf{0}, \mathbf{c}_2, \mathbf{c}_3, \cdots, \mathbf{c}_{p^k}\}$ in the first row of the table, with $\mathbf{c}_1 = \mathbf{0}$ in the first column.
(2) Choose a word $\mathbf{e}_2 \in \mathbf{Z}_p^n$ of least weight that has not yet appeared in the table. Add $\mathbf{e}_2$ to each of the codewords in the first row to create the second row of the standard array. (In Example 1.7, $\mathbf{e}_2 = 00001$.)
(3) Create additional rows in this manner: to create the i-th row, choose a word $\mathbf{e}_i \in \mathbf{Z}_p^n$ of smallest weight that has not yet appeared in the table, and add it to each of the codewords from the first row.
(4) Repeat this process until all possible received words in $\mathbf{Z}_p^n$ have appeared in the table. Since there are p^k codewords and p^n words, there will be p^{n-k} rows and p^k columns in the array.

The resulting standard array has the form

$$A = \begin{pmatrix} 0 & c_2 & c_3 & \dots & c_{p^k} \\ e_2 & c_2 + e_2 & c_3 + e_2 & \dots & c_{p^k} + e_2 \\ \vdots & \vdots & \vdots & & \vdots \\ e_i & c_2 + e_i & c_3 + e_i & \dots & c_{p^k} + e_i \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_{p^{n-k}} & c_2 + e_{p^{n-k}} & c_3 + e_{p^{n-k}} & \dots & c_{p^k} + e_{p^{n-k}}. \end{pmatrix}$$

Each row of a standard array for C is called a **coset**, and the first entry in each row (an $\mathbf{e}_i$) is called a **coset leader**. While different choices of coset leaders may lead to different tables, it turns out that decoding is always possible, that is, a received word can always be decoded to some code word, no matter which standard array for the code is used.

Exercise 10 shows that different choices of coset leaders can lead to different standard arrays and hence different decodings of the same word. But any standard array will place each received word under a codeword that is as close as possible to the received word.

At this point we still have to look up each received word in the table in order to decode it. We will see in Section 2 a method of decoding, called **syndrome decoding**, that applies the theory of vector spaces to greatly speed the process.

Example 1.8: In the vector space $\mathbf{Z}_3^4$, consider the code C with generating matrix

$$G = \begin{pmatrix} 2 & 1 & 0 & 1 \\ 1 & 2 & 2 & 0 \end{pmatrix}.$$

$\mathbf{Z}_3^4$ has a total of 81 vectors. Since the two rows of G are linearly independent, C has $3^2 = 9$ vectors, enumerated in the first row of the table below. The entire table constitutes one possible standard array for C.

0000	2101	1202	1220	0021	2122	2110	1211	0012
0001	2102	1200	1221	0022	2120	2111	1212	0010
0100	2201	1002	1020	0121	2222	2210	1011	0112
1000	0101	2202	2220	1021	0122	0110	2211	1012
0002	2100	1201	1222	0020	2121	2112	1210	0011
0200	2001	1102	1120	0221	2022	2010	1111	0212
2000	1101	0202	0220	2021	1122	1110	0211	2012
1100	0201	2002	2020	1121	0222	0210	2011	1112
1010	0111	2212	2200	1001	0102	0120	2221	1022

For example, with this array 1122 is decoded as 2122. ◇

Example 1.9: Let $C = \{0000, 1111, 2222\}$ be a one dimensional code in $\mathbf{Z}_3^4$. Then C has $3^3 = 27$ cosets, each containing three codewords. Can you give a very simple description of coset decoding for this particular case? ◇

Standard arrays have the following properties.

Proposition 1.1. *Let C be a linear $[n, k]_p$ code with standard array A. Then*

(i) The coset leader of each row has minimum weight among all words in its coset.

(ii) Any word $\mathbf{w}$ in a standard array is the vector sum of the codeword $\mathbf{c}$ heading its column and the coset leader $\mathbf{e}$ that leads its row.

(iii) Every word $\mathbf{w} \in \mathbf{Z}_p^n$ appears in A exactly once.

(iv) The number of rows of A is p^{n-k}.

(v) Two words $\mathbf{w}, \mathbf{v} \in \mathbf{Z}_p^n$ *lie in the same coset iff their difference* $\mathbf{w} - \mathbf{v}$ *is a codeword.*

(vi) Every word $\mathbf{w} \in \mathbf{Z}_p^n$ *appears in a column headed by a codeword* $\mathbf{c} \in C$ *such that for all* $\mathbf{r} \in C$, $d(\mathbf{w}, \mathbf{c}) \leq d(\mathbf{w}, \mathbf{r})$.

Proof. Parts (i) and (ii) follow directly from the construction.

To prove part (iii), first note that clearly each word occurs at least once in the array, otherwise we would not be finished with the construction of the array.

To show that each word in $\mathbf{Z}_p^n$ occurs at most once in the array, suppose for the sake of contradiction the same word $\mathbf{x}$ occurs in at least two different places in the array. Then we have $\mathbf{x} = \mathbf{b} + \mathbf{e} = \mathbf{c} + \mathbf{f}$ for some codewords $\mathbf{b}, \mathbf{c} \in C$ and some coset leaders $\mathbf{e}, \mathbf{f}$, with either $\mathbf{b} \neq \mathbf{c}$ or $\mathbf{e} \neq \mathbf{f}$. Since $\mathbf{b} = \mathbf{c}$ if and only if $\mathbf{e} = \mathbf{f}$, we know that both $\mathbf{b} \neq \mathbf{c}$ and $\mathbf{e} \neq \mathbf{f}$. Without loss of generality, suppose that $\mathbf{e}$ was chosen as a coset leader earlier in the construction than $\mathbf{f}$.

Since C is closed under subtraction, $\mathbf{y} = \mathbf{b} - \mathbf{c} = \mathbf{f} - \mathbf{e}$ is also in C. Hence $\mathbf{f} = \mathbf{y} + \mathbf{e}$, with $\mathbf{y} \in C$. This means f would have appeared in the row with coset leader $\mathbf{e}$ and could not have been chosen as a coset leader for a later row. This contradiction completes the proof of part (iii).

The proof of parts (iv), (v) and (vi) are left for Exercise 12. □

If we are interested in codes with a large number of codewords, relative to the length n, then we want to know for given values of n, d and p how large p^k can be. For the moment, let us restrict ourselves to binary vectors, i.e., $p = 2$. Let $A(n, d)$ denote the largest possible size of any subset of $\mathbf{Z}_2^n$ such that $d(x, y) \geq d$ for all pairs of distinct vectors $\mathbf{x}, \mathbf{y}$ taken from the subset.

Example 1.10: If $n = 3$ there are 8 binary vectors of length 3. Let $d = 2$. Then we find that any two vectors in the set

$$\{000, 011, 110, 101\}$$

are at a mutual distance of 2. Therefore $A(3, 2) \geq 4$. Further inspection of all possible subsets of $\mathbf{Z}_2^3$ shows that $A(3, 2) = 4$. ◇

Example 1.11: While $A(17, 4)$ is not known, it is known that

$$2720 \leq A(17, 4) \leq 3276.$$

Therefore a $[17, k, 4]_2$ code, linear or not, cannot have more than 3276 codewords. ◇

Finding the value of $A(n, d)$ is a fundamental problem in coding theory that remains open (that is, unsolved) for many values of n and d.

1.4. **Exercises for Section 1.**

(1) Give an example of a $[7, 1, 7]_2$ code, and an example of a $[7, 1, 7]_3$ code.
(2) (a) Enumerate two vectors in the code given in Example 1.3 different from those given in the example.
    (b) What is the redundancy of this code? How many information bits does it carry?
(3) Enumerate five vectors in the $[6, 4]_7$ Reed–Solomon code of Example 1.4 different from those in the example.
(4) Find a vector in $\mathbf{Z}_7^6$ that is orthogonal to (that is, has zero dot product with) the $[6, 4]_7$ vector
    a. 111111
    b. 123456
(5) Find the minimum distance of the linear code $C = \{00000, 10202, 20101\}$ in $\mathbf{Z}_3^4$.
(6) In Example 1.6,
    (a) Find $S(01101, 1)$ and $S(11001, 1)$.
    (b) Verify that the spheres are not disjoint.
    (c) Explain in your own words how $d(C)$ is related to the overlapping of the spheres and how the overlapping of the spheres renders the minimum distance decoding method ambiguous.
(7) Calculate the minimum weight of the code $C = \{00000, 01101, 10100, 11001\}$ in $\mathbf{Z}_2^5$.
(8) For the code and the standard array of Example 1.8, decode the received message:

$$2010 \qquad 1211 \qquad 1220 \qquad 0220.$$

(9) For the code of Example 1.7
    (a) Complete the indicated rows in the following standard array

00000	01101	10100	11001
01000			
10000			
00010			
00001			
00110			

    (b) Decode 11111 using the standard array in a).
    (c) Compare the result of part b) to the decoding of 11111 in Example 1.7. What does this comparison tell you about the method of standard arrays?
(10) Write out a generating matrix for a $[12, 4]_{13}$ Reed–Solomon code. How many codewords does the code have? How many cosets are there in a standard array?
(11) Complete the proof of Proposition 1.1. (Hint: For part (iv), compare the number of vectors in $\mathbf{Z}_p^n$ to the number of vectors in C.)
(12) It is known that $A(9, 4) = 20$. What does this mean?

(13) It is known that $2560 \leq A(21, 6) \leq 4096$.
  (a) Is it possible to find a binary code of length 21, minimum distance 6 or more, and 500 codewords?
  (b) State an open problem in Coding Theory.
(14) It is known that the $[6, 4]_7$ Reed–Solomon Code of Example 1.4 has $d = 3$.
  (a) How many vectors are there in this code?
  (b) What can be said about the mutual distance between the vectors in this code?
  (c) What inequality can we conclude for $A(6, 3)$ for codes in $\mathbf{A}_7^6$?
(15) Find the minimum distance of the linear code C generated by $G = \begin{pmatrix} 2 & 1 \end{pmatrix}$ in $\mathbf{Z}_3^2$.

2. The Parity Check Matrix and Syndrome Decoding

2.1. **The Parity Check Matrix.** We have described two different ways of defining a linear code: (1) listing the elements of a code as a subset $C \subseteq \mathbf{Z}_p^n$ that satisfies certain linearity properties, and (2) using a generating matrix G. We now describe a third way of defining a linear code: (3) using a parity check matrix H. We will use this parity check matrix to introduce the method of decoding called syndrome decoding; to present a second definition of Reed–Solomon codes; and to understand some of the general properties of codes.

Definition 2.1. *A **parity check matrix** for a $[n, k]_p$ code C is an $(n - k) \times n$ matrix H such that $C = \{\mathbf{x} \in \mathbf{Z}_p^n : H\mathbf{x}^T = \mathbf{0}\}$, i.e., $C^T = \{\mathbf{x}^T : \mathbf{x} \in C\} = Null(H)$, where $Null(H)$ is the nullspace[1] of H and $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$. If matrix G is a generating matrix for a code C, and if H is a parity check matrix for code C, we also say that H is a **parity check matrix for the matrix** G.*

Two quick remarks are in order. First, any code defined by a parity check matrix is necessarily a linear code, i.e., C will be a subspace of $\mathbf{Z}_p^n$. Second, for any given linear code C we can find a (non-unique) parity check matrix H_C; conversely, any matrix H determines a (unique) linear code $C_H = [\text{Null}(H)]^T$.

Example 2.1: We show how to calculate a parity check matrix H for the code defined in Example 1.6. Recall that $C = \{00000, 01101, 10100, 11001\}$ over $\mathbf{Z}_2$ has basis $\{01101, 10100\}$, and is thus a $[5, 2]_2$ code with the (non-unique) generating matrix

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Since the rows of H must be orthogonal to exactly those vectors that are in C, we must solve the **parity check equations** resulting from the matrix equation $G\mathbf{x} = \mathbf{0}$. Written

[1]The nullspace is sometimes called the kernel, and consists of the vectors $\mathbf{x}$ such that $\mathbf{x}H = \mathbf{0}$.

out explicitly, we need

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

which results in the parity check equations

$$x_1 + x_3 = 0 \text{ and } x_2 + x_3 + x_5 = 0.$$

Hence $x_1 = x_3$ and $x_2 = x_3 + x_5$, with x_3, x_4, and x_5 all free variables. (Recall that we are working over Z_2, so that $-x_3 = x_3$, and so on.) In matrix form, we have

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = x_3 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + x_5 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Therefore, the solution space for this matrix equation (that is, the null space of the matrix G) has basis

$$\left\{ \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

Transposing these vectors gives the rows of a parity check matrix: $H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$. $\diamond$

To understand why this method of calculating H from G works correctly, given a linear code $C \subseteq \mathbf{Z}_p^n$ we define a new vector subspace of $\mathbf{Z}_p^n$, i.e., a new code.

Definition 2.2. *The **dual code** to code C is $C^\perp = \{\mathbf{y} \in \mathbf{Z}_p^n : \mathbf{x} \cdot \mathbf{y} = \mathbf{0}$ for all $\mathbf{x} \in C\}$.*

This corresponds exactly to the definition of the **orthogonal complement** of any vector subspace W within any finite-dimensional vector space V, and, like the orthogonal complement of a subspace, C and $C^\perp$ share elegant relations.

Proposition 2.1. *(i) If C is a linear code, then $C^\perp$ is also a linear code.*

(ii) Suppose that $C \subseteq \mathbf{Z}_p^n$ is a linear code with a spanning set of vectors $S \subseteq C$ and $\mathbf{x} \in \mathbf{Z}_p^n$. Then $\mathbf{x} \in C^\perp$ if and only if $\mathbf{x} \cdot \mathbf{s} = \mathbf{0}$ for every $\mathbf{s} \in S$. In particular, if G is a generating matrix for code C, then $\mathbf{x} \in C^\perp$ if and only if $\mathbf{x} \cdot \mathbf{g}_i = \mathbf{0}$ for every row $\mathbf{g}_i$ of G.

(iii) For any matrix A, $(Row(A))^\perp = Null(A)$. In particular, if G is a generating matrix for a code, then $(Row(G))^\perp = Null(G)$.

(iv) If G is a generating matrix for code C, then G is a parity check matrix for $C^\perp$.

(v) If H is a parity check matrix for C, then H is a generating matrix for $C^\perp$.

(vi) If $dim(C) = k$, then $dim(C^\perp) = n - k$.

This proposition shows that calculating a basis for the null space of any generating matrix G for C is equivalent to calculating a basis for the dual code $C^\perp$. The matrix H whose row space equals $C^\perp$ then serves as a parity check matrix for C. The reader is asked to prove this proposition in the exercises.

Before exploring further the connections between H and C_H we will need the following definition.

Definition 2.3. *A $k \times n$ matrix G is said to be in **row-reduced echelon form** if*

1. *The first non-zero entry of each non-zero row of G is a 1.*
2. *Each leading 1 is to the right of the leading 1's above it.*
3. *A column containing a leading 1 consists of 0's otherwise.*
4. *Any rows of 0 occur at the bottom of G.*

Example 2.2: The matrix $G = \begin{pmatrix} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ in $\mathbf{Z}_3$ is not is row-reduced echelon form, but can be put into this form by adding the third row to the first. ◇

In general, a matrix not in row-reduced echelon form may be transformed into one by performing elementary row operations on it.

We now consider examples that further illustrate the connection between H and C_H.

Example 2.3: Consider the code $C \subseteq \mathbf{Z}_3^4$ that was defined in Example 1.8 using the generating matrix $G = \begin{pmatrix} 2 & 1 & 0 & 1 \\ 1 & 2 & 2 & 0 \end{pmatrix}$. To find H, instead of solving the matrix equation $G\mathbf{x} = \mathbf{0}$, we solve $G'\mathbf{x} = \mathbf{0}$, where $G' = \begin{pmatrix} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 2 \end{pmatrix}$ is the row-reduced echelon form of G. This is easily done (see Example 2.1) and produces $H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$ as a parity check matrix for this code. ◇

Now suppose that a parity check matrix H is given and we want to use it to find a generating matrix for the code C_H. If we solve the matrix equation $H\mathbf{x}^T = \mathbf{0}$, then $C_H = \left\{\mathbf{x} \in \mathbf{Z}_3^4 : \mathbf{x}^T \text{ is a solution to } H\mathbf{x}^T = \mathbf{0}\right\}$.

Example 2.4: We continue with Example 2.2, but assume this time that the parity check matrix $H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$ is given and we must find a generating matrix G_H. We start by finding a basis for the solution space of the matrix equation $H\mathbf{x}^T = \mathbf{0}$:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Solving the parity check equations, we find

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = x_1 \begin{pmatrix} 1 \\ 2 \\ 2 \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix}.$$

Thus, a generating matrix for this code is $C_H = \begin{pmatrix} 1 & 2 & 2 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix}$, which is row equivalent to the generating matrix G given for this code originally. ◇

As we have seen, finding the parity matrix for a given code, and finding the code determined by a given parity check matrix, are both straightforward but tedious tasks. The following proposition gives us a faster method for finding H when G is in the form prescribed in the following proposition.

Proposition 2.2. *If G is a $k \times n$ matrix with row-reduced echelon form $[I_k|A]$ then $H = \left[-A^T|I_{n-k}\right]$ is a parity check matrix for G. (Here $[B|C]$ is the matrix whose first columns are those of B and whose last columns are those of C.)*

Proof. Let $H = \left[-A^T|I_{n-k}\right]$. Since A is $k \times (n-k)$, A^T is $(n-k) \times k$, so H is $(n-k) \times n$. Further

$$\begin{aligned} HG^T &= [-A^T|I_{n-k}][I_k|A]^T \\ &= [-A^T|I_{n-k}] \left[\frac{I_k}{A^T}\right] \\ &= -A^T + A^T \\ &= [0], \end{aligned}$$

where [0] is the zero matrix. So G is in the null space of H. Finally, since H has rank $(n-k)$, as it has $n-k$ leading 1's, and G has rank k, G must equal the null space of H. Therefore $H = [-A^T|I_{n-k}]$ is indeed a parity check matrix for G. □

So to find H we do not need to solve a system of equations, but in general we do need to row reduce G to put it into the form $[I_k|A]$. We call $[I_k|A]$ the **standard form** of G.

Example 2.5: In Example 2.1 above, G is already in standard form: $G = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} = [I_k|A]$, where $k = 2$ and $A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$. Here $n-k = 5-2 = 3$ and $-A^T = \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$. By Proposition 2.2, this code has as a parity check matrix $H = \left[-A^T|I_{n-k}\right] = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$. This agrees exactly with the matrix H we found earlier by solving the parity check equations directly. (In general, the two different methods may yield two different parity check matrices, but any two such parity check matrices for the same code must be row equivalent, and therefore have the same null space.) ◇

Example 2.6: We use Proposition 2.2 to quickly find a parity check matrix for the code in Example 1.4. Working in $\mathbf{Z}_7$, we had $G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 2 & 2 & 4 & 1 \\ 1 & 1 & 6 & 1 & 6 & 6 \end{pmatrix}$ as a generating matrix for a $[6,4]_7$ Reed–Solomon code. The row reduced form of G is $G' = \begin{pmatrix} 1 & 0 & 0 & 0 & 6 & 3 \\ 0 & 1 & 0 & 0 & 4 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 4 & 3 \end{pmatrix}$, which is I_4 followed by $A = \begin{pmatrix} 6 & 3 \\ 4 & 1 \\ 1 & 1 \\ 4 & 3 \end{pmatrix}$. To find an H we transpose A and find the negative (in $\mathbf{Z}_7$), which is $-A^T = \begin{pmatrix} 1 & 3 & 6 & 3 \\ 4 & 6 & 6 & 4 \end{pmatrix}$. So $H = \begin{pmatrix} 1 & 3 & 6 & 3 & 1 & 0 \\ 4 & 6 & 6 & 4 & 0 & 1 \end{pmatrix}$. ◇

2.2. **Syndrome Decoding.** For codes that are subsets of $\mathbf{Z}_p^n$ with p^n relatively small the standard array introduced in Section 1 can be used to decode a received message by simply looking up each received word in the array. However, even a moderately-sized code such as the $[6,4]_7$ Reed-Solomon code will have a standard array with $p^n = 7^6 = 117,649$ entries

($p^{n-k} = 49$ rows and $p^k = 2,401$ columns), and the $[255, 223]_2$ Reed–Solomon code, a version of which is commonly used to correct errors on compact disks, has a standard array with 2^{255}, or approximately 6×10^{76} entries! Clearly we need a more efficient decoding method.

The key idea of **syndrome decoding** is quite simple. Consider a received word $\mathbf{r}$, and let $\mathbf{e}$ be its coset leader in a standard array for C and $\mathbf{c} \in C$ the codeword heading $\mathbf{r}$'s column. Then $\mathbf{r} = \mathbf{c} + \mathbf{e}$. So if H is a parity matrix for C then

$$H\mathbf{r} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = \mathbf{0} + H\mathbf{e} = H\mathbf{e}.$$

In other words, if $\mathbf{e}$ is the coset leader of $\mathbf{r}$, then $H\mathbf{r} = H\mathbf{e}$. If we had a list of all the vectors $H\mathbf{e}$, we could then find $H\mathbf{r}$ in that list, and would then know that $\mathbf{r} - \mathbf{e} = \mathbf{c}$ is the correct decoding of $\mathbf{r}$. This motivates the following definition.

Definition 2.4. *Given a linear code $C \subseteq \mathbf{Z}_p^n$ and a parity check matrix H for C, the **syndrome** of any (received word) $\mathbf{r} \in \mathbf{Z}_p^n$ is $syn(\mathbf{r}) = [H\mathbf{r}]^T$.*

The proof of the following proposition is straightforward.

Proposition 2.3. *Suppose that $\mathbf{r} \in \mathbf{Z}_p^n$ and that $C \subseteq \mathbf{Z}_p^n$ is a linear code. Then*

(i) $syn(\mathbf{r}) \in \mathbf{Z}_p^{n-k}$.

(ii) $\mathbf{r} \in C$ if and only if $syn(\mathbf{r}) = \mathbf{0}$.

(iii) If $\mathbf{r} = \mathbf{c} + \mathbf{e}$ and $\mathbf{c} \in C$, then $syn(\mathbf{r}) = syn(\mathbf{e})$.

Using syndrome decoding is relatively simple, but there is a bit of overhead as beforehand we must, once and for all, agree on a **syndrome table**. This table has two columns: a syndrome column, and a coset leader column. The syndrome column will consist of every possible syndrome $\mathbf{s} \in \mathbf{Z}_p^{n-k}$.

If we happen to have a list of coset leaders already available (for instance, from a standard array) we can simply compute the syndromes of those coset leaders, and hence have the table. Otherwise we take the following steps to form the table row by row.

(1) The first row is the zero vector, the coset leader for the coset consisting of the codewords of C, and its syndrome is $\mathbf{0} \in \mathbf{Z}_p^{n-k}$.
(2) Of the remaining unsampled vectors in $\mathbf{Z}_p^n$, choose one $\mathbf{v}$ of least weight and compute $syn(\mathbf{v})$. If $syn(\mathbf{v})$ does not appear in the syndrome column declare $\mathbf{v}$ to be a coset leader and add a row consisting of $\mathbf{v}$ and its syndrome to the table.
(3) Continue step (2) until we have found a coset leader for all of the p^{n-k} possible syndromes.

Once having created the syndrome table, syndrome decoding is simple. For each codeword $\mathbf{r} \in \mathbf{Z}_p^n$ that is received, compute $syn(\mathbf{r}) = H\mathbf{r}^T$ and from the syndrome table find the coset leader $\mathbf{e}$ such that $syn(\mathbf{e}) = syn(\mathbf{r})$. Then $\mathbf{c} = \mathbf{r} - \mathbf{e}$ is the correct decoding of $\mathbf{r}$.

Example 2.7: Let us create a syndrome table for the $[5, 2]_2$ code $C \subseteq \mathbf{Z}_2^5$ that may be listed explicitly as $C = \{00000, 01101, 10100, 11001\}$. In Example 1.7 we found a standard array for this code, so we may read off the coset leaders directly from the first column of the standard array. In Example 2.1 we found $H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$ is a parity check matrix for this code. Multiplying each coset leader by H allows us to create a syndrome table.

$\mathbf{e}$	$syn(\mathbf{e})^T$
00000	000
00001	001
00010	010
00100	100
01000	101
10010	110
01010	111
00011	011

Now suppose that we receive $\mathbf{r} = 10011$. We compute $syn(\mathbf{r}) = H\mathbf{r}^T = (1, 1, 1)^T$. Since $(1, 1, 1)^T$ is the syndrome of $\mathbf{e} = 01010$, we decode the received word $\mathbf{r}$ as $\mathbf{c} = \mathbf{r} - \mathbf{e} = 10011 - 01010 = 11001$. (Referring back to the standard array in Example 1.7, we see that $\mathbf{r} = 10011$ is indeed found in the array column headed by codeword $\mathbf{c} = 11001$.) Notice that we only had to look up eight entries in the syndrome table versus thirty-two in the standard array, so even in this small example, syndrome decoding was faster. ◇

As the code get larger, this advantage increases. The $[6, 4]_7$ Reed-Solomon code mentioned at the beginning of this subsection has a standard array of $7^6 = 117,649$ but its syndrome table (with two columns of $7^2 = 49$ rows each) has only 98 entries. Clearly the syndrome table is easier to deal with. Similarly the $[255, 223]_2$ Reed-Solomon code has a syndrome table of two columns with 2^{32} rows. While using a table of 2^{33} entries (approximately 8.6 billion) to decode may seem cumbersome, it far, far faster than using the standard array, which has $2^{255} \approx 6 \times 10^{76}$ entries!

2.3. **A Second Presentation of Reed–Solomon Codes.** So far we have seen examples of $[6, 4]_7$ and $[4, 2]_5$ Reed–Solomon codes, each defined by a generating matrix. We now describe a more general "recipe" that allows us to consider the Reed–Solomon codes as a

whole family of codes $RS(k,p)$, where, as usual, we fix a prime p and an integer k with $1 \leq k \leq p-1$.

Definition 2.5. *The Reed–Solomon code $RS(k,p)$ is the subspace of $\mathbf{Z}_p^{p-1}$ given by*

$$RS(k,p) = \left\{ \big(f(1), f(2), \ldots, f(p-1)\big) : f(x) \in \mathbf{Z}_p[x], \mathit{deg}\big(f(x)\big) < k \right\}.$$

Informally, we will think of an $RS(k,p)$ code as the image in $\mathbf{Z}_p^{p-1}$ of all polynomials with coefficients modulo p and degree less than k via a special type of linear transformation.

We now proceed to give examples implementing this definition. (We will present the linear transformation in Example 4.15.)

Example 2.8: Suppose we have fixed $p = 7$ and $k = 4$. Then

$$RS(4,7) = \{\big(f(1), f(2), \ldots, f(6)\big) : f(x) \in \mathbf{Z}_7[x], \deg\big(f(x)\big) < 4\}.$$

To give an example of a codeword we first need a polynomial over F_7 of degree less than 4, such as $f = x^3 + 4x^2 + 3x + 6$. The corresponding codeword in $RS(4,7)$ is then $\big(f(1), f(2), \ldots, f(6)\big) = (0,0,5,0,5,5)$.

$RS(4,7)$ was introduced in Example 1.4 via the generating matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \end{pmatrix}.$$

So since the two definitions are equivalent, 005055 should be obtained as a linear combination of the rows of G. And, indeed, we have

$$\mathbf{u} \times G = \begin{pmatrix} 6 & 3 & 4 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 5 & 0 & 5 & 5 \end{pmatrix}.$$

The vector $\mathbf{u}$ is obtained by listing the coefficients of f when its terms are written in order of increasing degree. Thus using the third column in the product above yields

$$6 \cdot 1 + 3 \cdot 3 + 4 \cdot 3^2 + 1 \cdot 3^3 = f(3) = 5$$

and similarly for the remaining entries of $\mathbf{u} \times G$. ◇

What are the parameters of $RS(k,p)$? Clearly $n = p-1$. The following proposition shows that the dimension of $RS(k,p)$ is precisely k, so that $RS(k,p)$ is a $[p-1,k]_p$ code.

Proposition 2.4. *The dimension of $RS(k,p)$ is k.*

Proof. Letting $L_p^{k-1} = \{f \in \mathbf{Z_p}[x] : \deg(f) < k\}$, then L_p^{k-1} is a subspace of $\mathbf{Z}_p[x]$ of dimension k, and

$$RS(k,p) = \left\{(f(1), f(2), \ldots, f(p-1)) : f \in L_p^{k-1}\right\}.$$

(Note that the superscript $k-1$ reminds us of the maximum possible degree for f.)

Define $T : L_p^{k-1} \to {\mathbf{Z}_p}^{p-1}$ by

$$T(f) = (f(1), f(2), \ldots, f(p-1)).$$

Then T is a linear transformation (by Example 4.15) whose image in ${\mathbf{Z}_p}^{p-1}$ is precisely $RS(k,p)$. Thus

$$\dim(RS(k,p)) = \dim(\text{image}(T)) \leq \dim\left(L_p^{k-1}\right) = k.$$

To show that $\dim(RS(k,p))$ is indeed k we need to show that T is one-to-one. So suppose $T(f) = T(g)$. Then $T(f-g) = \mathbf{0}$, i.e.,

$$T(f-g) = ((f-g)(1), (f-g)(2), \ldots, (f-g)(p-1)) = (0, 0, \ldots, 0).$$

This means $f-g$ is a polynomial of degree at most $k-1$ with at least $p-1$ roots. Now, just as over $\mathbf{R}$ and $\mathbf{C}$, the number of roots of a non-zero polynomial with coefficients in $\mathbf{Z}_p$ must be no larger than its degree. So if $f-g$ is non-zero we must have $p-1 \leq k-1$. But by definition of $RS(k,p)$, $1 \leq k \leq p-1$, or $k-1 \leq p-2 < p-1$. Thus $f-g$ cannot be non-zero, i.e., $f-g$ is the zero-polynomial, so $f = g$ and T is one-to-one. This makes $RS(k,p)$ isomorphic to L_p^{k-1} and so both spaces must have the same dimension, k. □

As stated in Section 1, on one hand we would like both k and d to be large (as this increases the amount of information we can send and number of errors we can detect), while on the other hand the Singleton Bound (Theorem 1.3) states that for any linear code $k+d \leq n-1$. So the best we can do is to have $d = n-k-1$. Codes for which this equality holds are called **Maximum Distance Separable** or **MDS** codes. One of the attractive properties of Reed–Solomon codes is that it is an MDS code.

Theorem 2.1. *A Reed–Solomon Code is an MDS code.*

Proof. In view of the Singleton Bound, we only need to show that if d is the minimum distance of $RS(k,p)$, then $d \geq n-k+1$ or $k-1 \geq n-d$.

If d is the minimum distance, it is also the minimum weight of $RS(k,p)$. Let $f \in L_p^{k-1}$ be such that $T(f)$ is of minimum weight in $RS(k,p)$. Then $T(f) = (f(1), f(2), \ldots, f(p-1))$ has exactly d non-zero entries, or, equivalently, $n-d$ zero entries. The polynomial f therefore has at least $n-d$ roots. This means f has degree at least $n-d$, and since the polynomials in L_p^{k-1} have degree at most $k-1$, we have

$$n-d \leq \deg(f) \leq k-1.$$

Therefore, $n-k+1 \leq d$ and the theorem is proved. □

2.4. **Exercises for Section 2.**

(1) Show that if C is a code for which matrix H is a parity check matrix, then C must be a linear code, that is, a vector subspace of $\mathbf{Z}_p^n$.

(2) The parity check matrix for a code C is sometimes defined as a matrix H such that $C = \{\mathbf{x} \in \mathbf{Z}_p^n : \mathbf{x}H^T = \mathbf{0}\}$. Explain why this is equivalent to our definition.

(3) Verify that for the parity check matrix H that we found in Example 2.1, and for every vector $\mathbf{b}$ in the basis given for that code, the equation $H\mathbf{b}^T = \mathbf{0}$ holds true. Why does this imply that $H\mathbf{x}^T = \mathbf{0}$ holds true for every $\mathbf{x} \in C$?

(4) Consider in both parts of this question the Hamming $[7, 4]_2$ code $C \subseteq \mathbf{Z}_2^7$ with basis $\mathbf{B} = \{1000110, 0100101, 0010111, 0001011\}$.
   (a) Find a parity check matrix H for this code by solving the system of parity check equations directly.
   (b) Write down a generating matrix G for this code, and then use Proposition 2.2 to find a parity check matrix H for this code.

(5) (a) Suppose that a code $C \subseteq \mathbf{Z}_3^5$ has generating matrix $G = \begin{pmatrix} 1 & 2 & 1 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 \end{pmatrix}$. Find a parity matrix H for code C.
   (b) Suppose that a code $D \subseteq \mathbf{Z}_3^5$ has parity check matrix $P = \begin{pmatrix} 1 & 2 & 1 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 \end{pmatrix}$. Find a generating matrix M for code D.
   (c) Compare matrix H from part (a) with matrix M from part (b). What does this tell you about codes C and D?

(6) Find a generating matrix G for the code C in $\mathbf{Z}_5^5$ whose parity matrix is $H = \begin{pmatrix} 1 & 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 3 & 4 \end{pmatrix}$. *Hint:* find a basis for Null(H), and then define G to have the basis vectors for Null(H) as its rows. You should explain why doing this works.

(7) Prove all parts of Proposition 2.1. Hint: you may find it helpful to refer to a linear algebra textbook to prove part (iii), looking up *orthogonal complement* in the index.

(8) Suppose that $G = \begin{pmatrix} 1 & 3 & 5 & 0 & 2 \\ 0 & 2 & 4 & 6 & 1 \end{pmatrix}$ is a generating matrix for code $C \subset \mathbf{Z}_7^5$. Find at least three other matrices G_1, G_2, and G_3 that are also generating matrices for this same code C.

(9) Show that if the matrix G' is formed from the matrix G by elementary row operations, then the code generated by G' is the same as the code generated by G.

(10) Suppose that the vectors

$$00010, 10220, 01001, 20100, 02022, 11211, 21121, 12202, 22112$$

constitute one row of a standard array for a code C (with 00010 being the coset leader). Find the complete set of all codewords in C, and explain how you know that your answer is correct.

(11) A syndrome table for the Hamming code $H_2(3)$ defined in Example 1.3 is given below.

(a) Use this table to decode the following received words: 1100110, 1010011, 1101001, 0010111.

(b) Verify that 0010100 has the same syndrome as 1000001. Note that these words both have a weight of two. What does this tell us about the maximum number of errors that we can correct using this code?

$\mathbf{e}$	Transpose of $syn(\mathbf{e}) = H\mathbf{e}^T$
0000000	0000
0000001	0001
0000010	0010
0000011	0011
0000100	0100
0000101	0101
1100000	0110
0001000	0111
0010000	1000
1000100	1001
1001000	1010
0100000	1011
1000001	1100
1000000	1101
1000010	1110
1000010	1111

(12) Follow the outline below to create a syndrome table for the code $C \subseteq \mathbf{Z}_3^4$ defined in Example 1.8. We saw in Example 2.2 that a parity check matrix for C is $H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$.

(a) How many rows should the standard array (and hence the syndrome table) have? Explain your calculations.

(b) List all possible errors $\mathbf{e}$ with weight 0 or 1. How many are there?

(c) For each possible error you found in part (b), compute the syndrome for that error. Note that if two possible errors $\mathbf{e}, \mathbf{f}$ have the same syndrome, that means that they are in the same coset (i.e., $\mathbf{e} + C = \mathbf{f} + C$).

(d) Continue computing syndromes of possible errors in order of ascending weight until you have found the number of rows with distinct syndromes that we are looking for. You will need to choose a few possible errors of weight 2 in addition to computing the syndromes for possible errors of weight 0 or 1.

(13) Find the codeword in $RS(4, 7)$ generated by polynomial $g = 2x^3 + x^2 + 5x + 3$. Write out the corresponding matrix equation showing how this codeword could also

be generated using the generating matrix for this code given earlier. (Hint: Follow Example 2.7.)

(14) Prove Proposition 2.3.

(15) (For those who have studied group theory.) Show that
   (a) If V is a vector space, then $\langle V, + \rangle$ forms a group, where the operation $+$ is defined as vector addition.
   (b) If W is a vector subspace of V, then $\langle W, + \rangle$ forms a normal subgroup of $\langle V, + \rangle$. Note that this allows us to define the quotient group $\langle V/W, \oplus_p \rangle$. Explain how a "coset" in coding theory is related to a "coset" in group theory.

3. Generating Polynomials

3.1. **Reed–Solomon Codes Generated by One Polynomial.** We have seen two ways of defining Reed–Solomon codes: using a generating matrix (in Section 1) and using a set of polynomials (in Section 2). We now give a third presentation, one which has the advantage of allowing the user to determine the parameter values in advance of defining the code. This gives the code designer greater control over the desired balance of efficiency of information transmission (as measured by k) and the error detecting and correcting capability (as dependent upon d) that was discussed prior to the statement of Theorem 1.3, The Singleton Bound.

First, a definition. We say that an element α of $\mathbf{Z}_p$ is a **primitive root** if multiplying α by itself enough times produces all the non-zero elements of $\mathbf{Z}_p$. For example, in $\mathbf{Z}_7$, 3 is a primitive root since $3^1 = 3$, $3^2 = 2$, $3^3 = 6$, $3^4 = 4$, $3^5 = 5$ and $3^6 = 1$, whereas 2 is not a primitive root. (As our concerns are elsewhere, you will be given the primitive root whenever necessary.)

Now we show how to construct a $[p-1, p-\delta]_p$ Reed-Solomon code. Choose an integer δ, $1 \leq \delta \leq p-1$, and choose $\alpha \in \mathbf{Z}_p$ that is a primitive root. Then set

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)\cdots(x-\alpha^{\delta-1}),$$

so $g(x) \in \mathbf{Z}_p[x]$ is a polynomial of degree $\delta - 1$ that has the first $\delta - 1$ powers of α as its roots. Multiplying out this product to see g as a polynomial, we have

$$g(x) = a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3 + \cdots + a_{\delta-2} x^{\delta-2} + x^{\delta-1}$$

for some $a_0, a_1, \cdots, a_{\delta-2} \in \mathbf{Z}_p$. We then define a code $RS_p(\alpha, \delta)$ by choosing the $(p-\delta) \times (p-1)$ generating matrix

$$G = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 & \cdots & a_{\delta-2} & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & a_0 & a_1 & a_2 & a_3 & \cdots & a_{\delta-2} & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & a_0 & a_1 & a_2 & a_3 & \cdots & a_{\delta-2} & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & a_0 & a_1 & a_2 & a_3 & \cdots & a_{\delta-2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \ddots & & \ddots & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_0 & a_1 & a_2 & \cdots & a_{\delta-2} & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_0 & a_1 & a_2 & \cdots & a_{\delta-2} & 1 \end{pmatrix}, \tag{1}$$

a matrix with $p-1$ entries in each row. (Although G may look complicated, each of its rows is simply a shift of the row above. Thus this form of Reed–Solomon is a special case of so-called "cyclic" codes, codes with generating matrices formed by such cyclic shifts.)

Example 3.1: Fix $p = 7$ and take $\alpha = 3$ as the primitive root. If we choose $\delta = 3$, then $\delta - 1 = 2$ so $g(x) = (x-3)(x-3^2) = (x-3)(x-2) = x^2 - 5x + 6 = 6 + 2x + x^2$. The coefficients of g are $(6, 2, 1)$ and the matrix G should have dimension $(p-\delta) \times (p-1) = (7-3) \times (7-1) = 4 \times 6$. So

$$G = \begin{pmatrix} 6 & 2 & 1 & 0 & 0 & 0 \\ 0 & 6 & 2 & 1 & 0 & 0 \\ 0 & 0 & 6 & 2 & 1 & 0 \\ 0 & 0 & 0 & 6 & 2 & 1 \end{pmatrix}$$

is a generating matrix for the code $RS_7(3, 3)$. ◇

Example 3.2: Fix $p = 11$ and take $\alpha = 2$ as the primitive root. If δ is choosen to be 6, then

$$\begin{aligned} g(x) &= (x-2)(x-2^2)(x-2^3)(x-2^4)(x-2^5) \\ &= (x-2)(x-4)(x-8)(x-5)(x-10) \\ &= 1 + 9x + 2x^2 + 8x^3 + 4x^4 + x^5. \end{aligned}$$

The dimensions of the matrix G are $(p-\delta) \times (p-1) = 5 \times 10$. The first row of G is $(1, 9, 2, 8, 4, 1, 0, 0, 0, 0)$ and the subsequent rows are wrapped until all the 0's have moved to the front.

$$G = \begin{pmatrix} 1 & 9 & 2 & 8 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 9 & 2 & 8 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 9 & 2 & 8 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 9 & 2 & 8 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 9 & 2 & 8 & 4 & 1 \end{pmatrix}.$$

Then $RS_{11}(2, 6)$ is generated by G. ◇

Since a_0 is non-zero, each of the rows will have a leading 1 when G is put into row-reduced form. So the generating matrices produced this way have rank equal to $p-\delta$. Thus $RS_p(\alpha,\delta)$ is a linear code in $\mathbf{Z}_p^{p-1}$ of dimension $p-\delta$, i.e., a $[p-1,p-\delta]_p$ code. As we will see in Proposition 3.1 below, the presence of a fair number of zeros in G facilitates the calculation of the parity check matrix H of $RS(\alpha,\delta)$. Once we have H, we can implement the syndrome decoding method.

3.2. **A Parity Check Matrix for $RS(\alpha,\delta)$.** We saw in Section 2.1 that any linear code with generating matrix G (of size $k\times n$) has a parity check matrix H, an $(n-k)\times n$ matrix such the row space of G is the transpose of Null(H). When the code has no special properties, to find H we must solve $G\mathbf{x}=\mathbf{0}$, which involves first row reduction and then solving the resulting system of equations. Proposition 2.2 gave a simpler method of finding H that does not require solving any systems of equations but does require that we row reduce G and thereby put it into standard form. We now show that when G is given by a generating polynomial even this row reduction is unnecessary.

Proposition 3.1. *A parity check matrix for $RS_p(\alpha,\delta)$ is the $(\delta-1)\times(p-1)$ Vandermonde matrix*

$$H=\begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^i & \cdots & \alpha^{p-2} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \cdots & \alpha^{2i} & \cdots & \alpha^{2(p-2)} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \cdots & \alpha^{3i} & \cdots & \alpha^{3(p-2)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 1 & \alpha^{\delta-1} & \alpha^{2(\delta-1)} & \alpha^{3(\delta-1)} & \cdots & \alpha^{i(\delta-1)} & \cdots & \alpha^{(p-2)(\delta-1)} \end{pmatrix}.$$

In Section 1 the Reed–Solomon codes came from generating matrices that are Vandermonde matrices (with rows that are powers of the elements of $\mathbf{Z}_p$). In this generating polynomial presentation of Reed–Solomon it is the parity check matrices that are Vandermonde matrices (with rows that are the powers of a primitive root).

Proof. To begin showing that the transpose of each row of the generating matrix G for $RS_p(\alpha,\delta)$ is in Null(H), write

$$\mathbf{g}_1=(g_0,g_1,g_2,\ldots,g_{p-2})=(a_0,a_1,\ldots,a_{\delta-1},1,0,\ldots,0)$$

for the first row of G. We need to show that $H{\mathbf{g}_1}^T=\mathbf{0}$. Let

$$\mathbf{h}_1=(h_0,h_1,h_2,\ldots,h_{p-1})=(1,\alpha,\alpha^2,\ldots,\alpha^{p-2})$$

be the first row of H. Then

$$\begin{aligned}\mathbf{h}_1{\mathbf{g}_1}^T=\mathbf{g}_1\cdot\mathbf{h}_1&=g_0h_0+g_1h_1+g_2h_2+\cdots+g_{p-1}h_{p-1}\\&=a_0+a_1\alpha+a_2\alpha^2+\cdots+a_{\delta-1}\alpha^{\delta-1}\\&=g(\alpha).\end{aligned}$$

Since g was chosen to have α as a root, this is 0. Hence $\mathbf{h}_1\cdot\mathbf{g}_1^T=0$.

Similarly, $\mathbf{h}_2\mathbf{g}_1^T = g(\alpha^2)$, since $\mathbf{h}_2$ is the even powers of α, and this is 0, again by choice of g. More generally, $\mathbf{h}_j\mathbf{g}_1^T = g(\alpha^j) = 0$, for $1 \leq j \leq \delta - 1$. Thus $\mathbf{g}_1^T$ is in Null(H).

Next, notice that the second row, $\mathbf{g}_2$, of G consists of the coefficients of $xg(x)$ followed by 0's. So $\mathbf{h}_1\mathbf{g}_2^T = \alpha^2 g(\alpha) = 0$, and, similarly, $\mathbf{h}_j\mathbf{g}_2^T = \alpha^2 g(\alpha^j) = 0$ for $1 \leq j \leq \delta - 1$. Finally, i-th row of G consists of the coefficients of $x^{i-1}g(x)$ followed by 0's, so by dealing carefully with the subscripts and exponents we can show $\mathbf{h}_j\mathbf{g}_i^T = (\alpha^j)^i g(\alpha^j) = 0$. Therefore the transpose of every row of G is in Null(H), and so H is a parity check matrix for $R_p(\alpha, \delta)$. This shows that Row$(G)^T \subseteq$ Null(H). Finally, by dimension considerations, we have Row$(G)^T =$ Null(H). □

Example 3.3: Using the parameters from Example 3.1, $p = 7$, $\alpha = 3$ and $\delta = 3$,

$$H = \begin{pmatrix} 1 & 3 & 3^2 & 3^3 & 3^4 & 3^5 \\ 1 & 3^2 & 3^4 & 3^6 & 3^8 & 3^{10} \end{pmatrix} = \begin{pmatrix} 1 & 3 & 2 & 6 & 4 & 5 \\ 1 & 2 & 4 & 1 & 2 & 4 \end{pmatrix}$$

is a parity check matrix for $RS_7(3, 3)$. ◇

Example 3.4: In Example 3.2 we had $p = 11$, $\alpha = 2$ and $\delta = 6$, so

$$H = \begin{pmatrix} 1 & 2 & 2^2 & 2^3 & 2^4 & 2^5 & 2^6 & 2^7 & 2^8 & 2^9 \\ 1 & 2^2 & 2^4 & 2^6 & 2^8 & 2^{10} & 2^{12} & 2^{14} & 2^{16} & 2^{18} \\ 1 & 2^3 & 2^6 & 2^9 & 2^{12} & 2^{15} & 2^{18} & 2^{21} & 2^{24} & 2^{27} \\ 1 & 2^4 & 2^8 & 2^{12} & 2^{16} & 2^{20} & 2^{24} & 2^{28} & 2^{32} & 2^{36} \\ 1 & 2^5 & 2^{10} & 2^{15} & 2^{20} & 2^{25} & 2^{30} & 2^{35} & 2^{40} & 2^{45} \end{pmatrix} = \begin{pmatrix} 1 & 2 & 4 & 8 & 5 & 10 & 9 & 7 & 3 \\ 1 & 4 & 5 & 9 & 3 & 1 & 4 & 5 & 9 \\ 1 & 8 & 9 & 6 & 4 & 10 & 3 & 2 & 5 \\ 1 & 5 & 3 & 4 & 8 & 1 & 5 & 3 & 4 \\ 1 & 10 & 1 & 10 & 1 & 10 & 1 & 10 & 1 \end{pmatrix}$$

is a parity check matrix for $RS_{11}(2, 6)$. ◇

Corollary 3.1. *Syndrome decoding may be performed on $RS_p(\alpha, \delta)$ without having to first row reduce G.*

3.3. **The Minimum Distance of $RS_p(\alpha, \delta)$.** In Section 3.1 we used the polynomial $g(x)$ whose roots are the first $(\delta - 1)$'st powers of α to create a $[p - 1, p - \delta]_p$ code, that is, a linear subspace of $\mathbf{Z}_p^{p-1}$ of dimension $p - \delta$. What is the minimum distance of this code?

We know the minimum distance is at most δ, since, by Theorem 1.2, for linear codes the minimum distance and the minimum weight agree and G is made up of rows with at most δ non-zero entries. Why is the minimum distance exactly δ? Perhaps it is simplest to answer this question by first proving a result that is interesting in its own right.

Proposition 3.2. *Suppose C is a linear code and H is a parity matrix for C. Then the minimum weight of C is d if and only if H has d linearly dependent columns and every choice of $d - 1$ columns from H gives a linearly independent set.*

Proof. The key idea of the proof is that if H is a $(n-k) \times n$ matrix, then for *any* vector $\mathbf{v}$ of $\mathbf{Z}_p^n$, $H\mathbf{v}^T$ produces a linear combination of columns of H, while conversely, any linear combination of the columns H can be expressed as $H\mathbf{w}^T$ for some vector $\mathbf{w} \in \mathbf{Z}_p^n$. Hence, a linear dependence of, say, e columns of H corresponds to a vector in C with e non-zero coefficients, that is, a vector of weight e.

With this understanding, and recalling that C^T is the null space of H, "H has d linearly dependent columns" is equivalent to "there is an element of C of weight d," while "every choice of $d-1$ columns from H gives a linearly independent set" is equivalent to "C contains no non-zero elements of weight less than d". (Note that the weight of a vector is not changed by taking its transpose.) □

Proposition 3.2 tells us that to compute d for $RS_p(\alpha, \delta)$ we must study the linear dependence/independence of the columns of a matrix of the form appearing in Proposition 3.1. First, every set of δ columns is linearly dependent because each column is in $\mathbf{Z}_p^{\delta-1}$. To see that every choice of $\delta - 1$ columns produces a linearly independent set, let H' be a $(\delta-1) \times (\delta-1)$ matrix consisting of some choice of columns. The top row of H' consists of various powers of α, with the remainder of each column consisting of powers of that first entry. So if we factor out of each column its top term, we produce a square Vandermonde matrix. Calling this new matrix K, then $\det(H') = \pi\det(K)$, where π is the product of the elements in the first row of H', and $\det(K) \neq 0$. Therefore $\det(H') \neq 0$. Remembering where we started, this means that every choice of $\delta-1$ columns from H produces a linearly independent set. By Proposition 3.2, this shows that the weight of $RS(\alpha, \delta)$ is δ. We have proven

Corollary 3.2. *The minimum distance of $RS_p(\alpha, \delta)$ is δ. That is, for $RS_p(\alpha, \delta)$, we have $d = \delta$.*

Example 3.5: In the code from Example 3.2 we had

$$H = \begin{pmatrix} 1 & 2 & 2^2 & 2^3 & 2^4 & 2^5 & 2^6 & 2^7 & 2^8 & 2^9 \\ 1 & 2^2 & 2^4 & 2^6 & 2^8 & 2^{10} & 2^{12} & 2^{14} & 2^{16} & 2^{18} \\ 1 & 2^3 & 2^6 & 2^9 & 2^{12} & 2^{15} & 2^{18} & 2^{21} & 2^{24} & 2^{27} \\ 1 & 2^4 & 2^8 & 2^{12} & 2^{16} & 2^{20} & 2^{24} & 2^{28} & 2^{32} & 2^{36} \\ 1 & 2^5 & 2^{10} & 2^{15} & 2^{20} & 2^{25} & 2^{30} & 2^{35} & 2^{40} & 2^{45} \end{pmatrix}.$$

There are five rows, so any choice of six or more column will produce a linearly dependent set. Now suppose we pick exactly five columns, say the 2nd, 3rd, 4th, 7th and 9th. As in the proof let H' consist of these columns: $H' = \begin{pmatrix} 2 & 2^4 & 2^5 & 2^7 & 2^9 \\ 2^2 & 2^8 & 2^{10} & 2^{14} & 2^{18} \\ 2^3 & 2^{12} & 2^{15} & 2^{21} & 2^{27} \\ 2^4 & 2^{16} & 2^{20} & 2^{28} & 2^{36} \\ 2^5 & 2^{20} & 2^{25} & 2^{35} & 2^{45} \end{pmatrix}$. Then

$\det(H') = 2 \cdot 2^4 \cdot 2^5 \cdot 2^7 \cdot 2^9 \det(K)$, where $K = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2^4 & 2^5 & 2^7 & 2^9 \\ 2^2 & 2^8 & 2^{10} & 2^{14} & 2^{18} \\ 2^3 & 2^{12} & 2^{15} & 2^{21} & 2^{27} \\ 2^4 & 2^{16} & 2^{20} & 2^{28} & 2^{36} \end{pmatrix}$. Clearly K is a Vandermonde matrix, and so it is invertible, and the same holds for H'. ◇

Corollary 3.2 also provides an alternative proof of Theorem 2.1 from Section 2.

Corollary 3.3. *$RS(\alpha, \delta)$ is a MDS code. To be specific, $RS(\alpha, \delta)$ is a linear code with parameters $n = p - 1$, $k = p - \delta$, and $d = \delta$ that satisfy $d + k = n - 1$.*

3.4. **Exercises for Section 3.**

(1) (a) Show that 2 is a primitive root in $\mathbf{Z}_{13}$ but 3 is not.
    (b) Show that 9 is a primitive root in $\mathbf{Z}_{11}$ but 4 is not.
(2) Given $p = 5$, $\alpha = 2$, and $\delta = 3$.
    (a) Calculate a generating matrix G for $RS(2, 3)$.
    (b) Calculate a parity check matrix H for $RS(2, 3)$. Verify that $\mathbf{h}_2\mathbf{g}_2^T = 0$.
    (c) Calculate the parameters of $RS(2, 3)$.
(3) Given $p = 7$, $\alpha = 4$, and $\delta = 4$.
    (a) Calculate a generating matrix G for $RS(4, 4)$.
    (b) Calculate a parity check matrix H for $RS(4, 4)$. Verify that $\mathbf{h}_3\mathbf{g}_4^T = 0$.
    (c) Calculate the parameters of $RS(4, 4)$.
(4) Suppose that in Example 3.1 we defined G' to be the $3 \times 5 = (p - \delta - 1) \times (p - 2)$ matrix $G' = \begin{pmatrix} 6 & 2 & 1 & 0 & 0 \\ 0 & 6 & 2 & 1 & 0 \\ 0 & 0 & 6 & 2 & 1 \end{pmatrix}$ and let C be the linear code generated by G'.
    (a) Find a parity check matrix H' for C.
    (b) What is the minimum distance for C.
    (c) Can you generalize the results of this exercise? That is, suppose δ has been chosen to be strictly less than $p-1$ and G' is defined to be the the $(p-\delta-1) \times (p-2)$ matrix built from cyclic shifts of the coefficients of $g(x)$. (Alternatively, G' is found by deleting the last row and last column of G.) Then what are the parameters for the linear code generated by G'?
(5) Suppose that in Example 3.1 we defined $g_2(x) = (x-2^2)(x-2^3)(x-2^4)(x-2^5)(x-2^6)$, and let G_2 be the 4×6 matrix generated by the coefficients of k. Like $g(x)$, $g_2(x)$ has consecutive powers of α for its roots, however these powers start with the second power, not the first.
    (a) Compute the generating matrix G_2.
    (b) Compute a parity matrix H_2 corresponding to G_2.

(c) Can you generalize the results of this exercise? That is, suppose $g_k(x)$ has α^k, α^{k+1}, $\cdots$, $\alpha^{\delta-1+k}$ for its roots, and G_k is the matrix constructed from the coefficients of g_k. What are the parameters of the linear code generated by G_k?

4. Appendix

We assume that the reader is familiar with the theory of vector spaces over the field of real numbers, as presented in a first course in Linear Algebra. We begin this appendix by defining arithmetic modulo a prime number p in order to introduce finite fields and vector spaces over these fields. We conclude with a result on linear transformations and a remark on the cardinality of vector spaces over finite fields. The later theorems in the appendix are used in the earlier sections of the module. Any standard Linear Algebra text contains proofs of these theorems for real vector spaces. These proofs generalize to vector spaces over finite fields without complication.

4.1. **Vector Spaces Over Finite Fields.**

Definition 4.1. *If a and b are integers, we say that a is **congruent to** b **modulo** p, and write $a \equiv b \pmod{p}$, if $a - b$ is divisible by p, i.e., there is some integer k such that $a = b + kp$.*

Example 4.1: $33 \equiv 5 \pmod{7}$ since $33 = 5 + 4 \times 7$. Likewise, $203 \equiv 58 \pmod{5}$ since $203 = 58 + 29 \times 5$. ◇

Consider the set of integers $\mathbf{Z}_p = \{0, 1, 2, \ldots, p-1\}$. Using the congruence relation, let us define addition and multiplication for this set.

Definition 4.2. *If a and b are elements of $\mathbf{Z}_p$, their **sum** is $a \oplus b = c$, where c is the (unique) element in $\mathbf{Z}_p$ that is congruent to $(a + b)$ modulo p, and their **product** is $a \otimes b = d$, where d is the (unique) element in $\mathbf{Z}_p$ that is congruent to $a \times b$ modulo p.*

Example 4.2: If $p = 7$, $\mathbf{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\}$. Then $4 \oplus 5 = 2$ since $4 + 5 = 9 \equiv 2 \pmod{7}$, and $4 \otimes 5 = 6$ since $4 \times 5 = 20 \equiv 6 \pmod{7}$. The full addition and multiplication tables for $\mathbf{Z}_7$ are as follows:

$\oplus$	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

$\otimes$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	1	2	3	4	5

.

When we wish to be very clear, we indicate the set $\mathbf{Z}_7$ together with its two operations by $(\mathbf{Z}_7, \oplus, \otimes)$. $\diamond$

$(\mathbf{Z}_7, \oplus, \otimes)$ is very rich in properties. In fact, it satisfies all the properties listed in the following definition.

Definition 4.3. *A **field** $(\mathbf{F}, \oplus, \otimes)$ is a set $\mathbf{F}$ together with two binary operations, $\oplus$ and $\otimes$ called **addition** and **multiplication**, defined on $\mathbf{F}$ such that the following axioms are satisfied for all elements a, b and c in $\mathbf{F}$.*

- F_1: *$\mathbf{F}$ is closed with respect with addition and multiplication, i.e., $a \oplus b$ and $a \otimes b$ are elements of $\mathbf{F}$.*
- F_2: *Addition and multiplication are commutative.*
- F_3: *Addition and multiplication are associative.*
- F_4: *$\mathbf{F}$ contains an element e, called the identity element for addition, such that $a \oplus e = a$.*
- F_5: *For each a in $\mathbf{F}$ there exists an element a' in $\mathbf{F}$, called the additive inverse of a, such that $a \oplus a' = e$.*
- F_6: *$\mathbf{F}$ contains an element u, called the identity element for multiplication, such that $a \otimes u = a$.*
- F_7: *For each $a \neq e$ in $\mathbf{F}$ there exists an element a^* in $\mathbf{F}$, called the multiplicative inverse of a, such that $a \otimes a^* = u$.*
- F_8: *Multiplication is distributive with respect to addition, i.e., $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$.*

This definition looks complicated so we provide two examples. The first should be very familiar, and the second is fundamental for this module.

Example 4.3: The set $\mathbf{R}$ of real numbers, with usual addition and multiplication is a field. In this case $e = 0$ and $u = 1$. $\diamond$

Example 4.4: $(\mathbf{Z}_7, \oplus, \otimes)$ is a field with $e = 0$ and $u = 1$. It is evident from the tables that axioms F_1 and F_2 are satisfied. Let us verify one case of each of the remaining axioms.

F_3: Since $6 \oplus (2 \oplus 3) = 6 \oplus 5 = 4$, and $(6 \oplus 2) \oplus 3 = 1 \oplus 3 = 4$. This verifies that addition is associative in this case.
F_3: $4 \otimes (5 \otimes 2) = 4 \otimes 3 = 5$, and $(4 \otimes 5) \otimes 2 = 6 \otimes 2 = 5$. This verifies that multiplication is associative in this case.
F_5: Since $3 \oplus 4 = 0$, 4 is the additive inverse of 3.
F_7: The multiplicative inverse of 2 is 4 since $2 \otimes 4 = 1$.
F_8: Since $5 \otimes (2 \oplus 6) = 5 \otimes 1 = 5$, and $(5 \otimes 2) \oplus (5 \otimes 6) = 3 \oplus 2 = 5$, the distributive law holds in this case.

◇

For any prime number p we may construct tables of addition and multiplication as we did for $\mathbf{Z}_7$ and the resulting structure is also a field.

Theorem 4.1. *$(\mathbf{Z}_p, \oplus, \otimes)$ is a field for any prime number p.*

For a proof, see [1], page 267.

Since we will only work with fields $\mathbf{Z}_p$, we will henceforth indicate the elements e and u with 0 and 1, respectively. The definition of vector space over the real numbers $\mathbf{R}$ can now be generalized to that of a vector space over a field $\mathbf{Z}_p$. Just as the typical vector space over the real numbers consists of n-tuples of real numbers, so the typical vector space over the field $\mathbf{Z}_p$ consists of n-tuples of elements in $\mathbf{Z}_p$. For example, a vector space of dimension five over $\mathbf{Z}_3$ will contain vectors such as $(1, 0, 0, 2, 1)$.

Definition 4.4. *A **vector space** V over $\mathbf{Z}_p$ is a set of elements called vectors, together with two operations, addition and scalar multiplication. The addition of two vectors satisfies the following axioms for all vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ in V.*

V_1: *V is closed with respect to addition.*
V_2: *Addition is commutative and associative.*
V_3: *There exists a vector $\mathbf{0}$ such that $\mathbf{x} + \mathbf{0} = \mathbf{x}$.*
V_4: *For each vector $\mathbf{x}$ there exists a vector $\mathbf{x}'$ such that $\mathbf{x} + \mathbf{x}' = \mathbf{0}$.*

The scalar multiplication of an element in $\mathbf{Z}_p$, called a scalar, by each element in V satisfies the following axioms, for any elements a and b in $\mathbf{Z}_p$ and any elements $\mathbf{x}$ and $\mathbf{y}$ in V.

V_5: *$a\mathbf{x}$ is an element of V.*
V_6: $a(b\mathbf{x}) = (ab)\mathbf{x}$.
V_7: $(a + b)\mathbf{x} = a\mathbf{x} + b\mathbf{x}$.
V_8: $a(\mathbf{x} + \mathbf{y}) = a\mathbf{x} + a\mathbf{y}$.
V_9: $1\mathbf{x} = \mathbf{x}$.

The following examples of vector spaces are of importance in coding theory.

Example 4.5: We will indicate with $\mathbf{Z}_7[x]$ the set of polynomials in the variable x with coefficients in $\mathbf{Z}_7$. The set of polynomials with coefficients in $\mathbf{Z}_7$ and degree at most 5, say, is indicated by L_7^5, so

$$L_7^5 = \{f \in \mathbf{Z}_7[x] : \deg(f) \leq 5\}.$$

Addition and scalar multiplication in L_7^5 are defined by adding and multiplying coefficients modulo 7. So if $f(x) = 6x^2 + 4x^3 + x^5$ and $g(x) = x^2 + 5x^3 + 3x^4$ are two elements in L_7^5, then $(f+g)(x) = f(x) + g(x) = (6x^2 + 4x^3 + x^5) + (x^2 + 5x^3 + 3x^4) = 2x^3 + 3x^4 + x^5$ and $(5f)(x) = 5f(x) = 2x^2 + 6x^3 + 5x^5$. With these operations, L_7^5 is a vector space over $\mathbf{Z}_7$. $\diamond$

Example 4.6: Let $\mathbf{Z}_3^6$ indicate the set of six-tuples of elements in $\mathbf{Z}_3$. Define addition of six-tuples coordinate-wise, modulo 3. For example, $(1,0,2,2,0,2) + (2,2,0,1,1,2) = (1 \oplus_p 2, 0 \oplus_p 2, 2 \oplus_p 0, 2 \oplus_p 1, 0 \oplus_p 1, 2 \oplus_p 2) = (0,2,2,0,1,1)$. Likewise, define scalar multiplication modulo 3. For example, $2(2,1,1,0,2,1) = (2 \otimes_p 2, 2 \otimes_p 1, 2 \otimes_p 1, 2 \otimes_p 0, 2 \otimes_p 2, 2 \otimes_p 1) = (1,2,2,0,1,2)$. With these operations, $\mathbf{Z}_3^6$ is a vector space over $\mathbf{Z}_3$. $\diamond$

These examples generalize as follows.

Theorem 4.2. *Let $\mathbf{Z}_p^n$ be the set of n-tuples of elements of $\mathbf{Z}_p$. Define addition of n-tuples coordinate-wise modulo p, i.e.,*

$$(x_1, x_2, \ldots, x_n) + (y_1, y_2, \ldots, y_n) = (x_1 \oplus_p y_1, x_2 \oplus_p y_2, \ldots, x_n \oplus_p y_n).$$

For a in $\mathbf{Z}_p$ define scalar multiplication also coordinate-wise modulo p, i.e.,

$$a(x_1, x_2, \ldots, x_n) = (a \otimes_p x_1, a \otimes_p x_2, \ldots, a \otimes_p x_n).$$

Then $\mathbf{Z}_p^n$ is a vector space over $\mathbf{Z}_p$.

Theorem 4.3. *Let L_p^n be the set of polynomials of degree at most n with coefficients in $\mathbf{Z}_p$. Define addition and scalar multiplication of polynomials as addition and scalar multiplication modulo p on the corresponding coefficients. Then L_p^n is a vector space over $\mathbf{Z}_p$.*

The proofs of these theorems are fairly straightforward.

4.2. **Subspaces, Bases and Dimension.** As in vector spaces over $\mathbf{R}$, the subsets of a general vector space that are closed under addition and scalar multiplication deserve special attention.

Definition 4.5. *A subset C of a vector space V over the field $\mathbf{Z}_p$ is called a **subspace** of V if*

(i) whenever $\mathbf{x}$ and $\mathbf{y}$ are elements of C, then $\mathbf{x} + \mathbf{y}$ is an element of C, and

(ii) whenever $a \in \mathbf{Z}_p$ is a scalar and $\mathbf{x}$ is in C, then $a\mathbf{x}$ is an element of C.

It follows from this definition that a subspace satisfies all the axioms in the definition of a vector space. Therefore a subspace is a vector space over the field $\mathbf{Z}_p$.

Example 4.7: The set $C = \{(0,0,0),(1,1,1)\}$ is a subspace of $\mathbf{Z}_2^3$. It is easy to verify that the two axioms are satisfied since $(1,1,1)+(1,1,1) = (0,0,0)$ and the only scalars are 0 and 1. ◇

Just as in the case of vector spaces over $\mathbf{R}$, linear combinations of vectors can be used to construct subspaces of V.

Definition 4.6. *If $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are vectors in V, and $a_1, a_2, \ldots, a_k$ are scalars, the vector $a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \ldots + a_k\mathbf{x}_k$ is called a **linear combination** of the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. The **span** of a set B of vectors over the field F is the set of all linear combinations of the vectors in B using coefficients from that field. That is, if $B = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ then*

$$Span(B) = \{a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \ldots + a_k\mathbf{x}_k : a_i \in F\}.$$

Theorem 4.4. *Let $B = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ be a set of vectors in the vector space V and let $C = Span(B)$. Then C is a subspace of V.*

If C is as in Theorem 4.4, we say that C is **generated** by the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$, or, equivalently, by the set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$.

Example 4.8: The set of all linear combinations of the vectors $(1,2,3,4,5,6)$, $(1,4,2,2,4,1)$ and $(1,1,6,1,6,6)$ is a subspace of $\mathbf{Z}_7^6$. Other vectors in this subspace include

$$5(1,4,2,2,4,1) + 3(1,1,6,1,6,6) = (1,2,0,6,3,2)$$

and

$$(1,2,3,4,5,6) + 6(1,2,3,4,5,6) = (0,0,0,0,0,0).$$

◇

The definitions of linear independence and linear dependence of a set of vectors in a vector space over a finite field are identical to those for vector spaces over $\mathbf{R}$. One must, however, be careful to carry out the operations modulo the appropriate prime number.

Definition 4.7. *A set of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ in a vector space V is said to be **linearly independent** if whenever $a_1\mathbf{x}_1 + a_1\mathbf{x}_2 + \ldots + a_k\mathbf{x}_k = \mathbf{0}$ for scalars $a_i \in F$, $i = 1, 2, \ldots, k$, it follows that $a_i = 0$ for all $i = 1, 2, \ldots, k$. If a set of vectors is not linearly independent, we say that it is **linearly dependent**.*

Example 4.9: In the vector space $\mathbf{Z}_5^3$, the set of vectors $S = \{(4,3,0),(0,1,1)\}$ is linearly independent because if $a(4,3,0) + b(0,1,1) = (0,0,0)$, then $(4a, 3a+b, b) = (0,0,0)$ or

$4a = 0$, $3a+b = 0$, and $b = 0$. Consequently $a = b = 0$. On the other hand, the set of vectors $T = \{(2,4,3),(1,2,4)\}$ is not linearly independent because $3(2,4,3) + 4(1,2,4) = (0,0,0)$. Thus the set $\{(2,4,3),(1,2,4)\}$ is linearly dependent. ◇

Given a vector subspace C, we often want to find a minimal set of vectors that generate C. Such minimal sets turn out to be linearly independent.

Example 4.10: Let C be the subspace of $\mathbf{Z}_5^3$ generated by the vectors $(4,3,0)$, $(0,1,1)$ and $(3,2,1)$. This set of vectors is linearly dependent since $2(4,3,0) + (0,1,1) = (3,2,1)$. So we may also describe C as the subspace generated by $(4,3,0)$ and $(0,1,1)$. This set is minimal in the sense that no single vector generates C. In fact any two of the three vectors above will form a minimal set generating C. ◇

The concepts of basis and dimension of a vector space of $\mathbf{Z}_p$ can now be formulated.

Definition 4.8. *Let V be a vector space. A set of vectors in V is called a **basis** of V if it is linearly independent and it generates V.*

Theorem 4.5. *Any two bases of V have the same number of elements.*

Definition 4.9. *The number of elements in any basis of V is called the **dimension** of V.*

The following examples illustrate these concepts.

Example 4.11: For L_7^5 is as in Example 4.5, two bases are:

$$B_1 = \{1, x, x^2, x^3, x^4, x^5\} \text{ and } B_2 = \{x-1, x^2-x, x^3-x^2, x^4-x^3, x^5-x^4, 1-x^5\}.$$

Thus L_7^5 has dimension six. The subspace of L_7^5 generated by the set of vectors $\{x-1, x^2-x, x^2-1\}$ has dimension two and a basis is $\{x-1, x^2-x\}$. ◇

Example 4.12: In Example 4.10, C is a subspace of dimension 2 of $\mathbf{Z}_5^3$, while the vector space $\mathbf{Z}_5^3$ itself has dimension 3, since a basis is provided by the set $\{(1,0,0),(0,1,0),(0,0,1)\}$. ◇

In general, searching for a basis by trial and error can be cumbersome. Given the set of vectors $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ that generates a subspace C, we would like to find a linearly independent subset of S that also generates C. This is equivalent to finding the largest subset of S that is linearly independent. The following theorem provides a quick way to find such a largest subset for the purposes of this module.

Theorem 4.6. *Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ be vectors in $\mathbf{Z}_p^n$ and let M be the matrix whose rows are the entries of the given vectors. If M has a $j \times j$ sub-matrix with a non-zero determinant, then the corresponding j rows of the matrix are linearly independent vectors.*

Example 4.13: Consider the vectors $(4,3,0)$, $(0,1,1)$ and $(3,2,1)$ in $\mathbf{Z}_5^3$. Then $M = \begin{pmatrix} 4 & 3 & 0 \\ 0 & 1 & 1 \\ 3 & 2 & 1 \end{pmatrix}$. Since $\det(M) \equiv 0 \pmod 5$, the three row vectors are linearly dependent over $\mathbf{Z}_5$. On the other hand, the 2×2 sub-matrix $\begin{pmatrix} 4 & 3 \\ 0 & 1 \end{pmatrix}$ has a non-zero determinant. So the set of vectors $\{(4,3,0),(3,2,1)\}$ is linearly independent. $\diamond$

Definition 4.10. *Given a matrix M, the size of the largest set of linearly independent row vectors is called the **rank** of M.*

Since determinants can be used to calculate the rank of a matrix, it will facilitate our work to know a family of matrices with non-zero determinants.

Definition 4.11. *A determinant D is a **Vandermonde determinant** if it is the determinant of a matrix of one of the following forms*

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ a_1 & a_2 & \dots & a_n \\ a_1^2 & a_2^2 & \dots & a_n^2 \\ \vdots & \vdots & & \vdots \\ a_1^{n-1} & a_2^{n-1} & \dots & a_n^{n-1} \end{pmatrix} \quad or \quad \begin{pmatrix} 1 & a_1 & a_1^2 & \dots & a_1^{n-1} \\ 1 & a_2 & a_2^2 & \dots & a_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & a_n & a_n^2 & \dots & a_n^{n-1} \end{pmatrix}$$

where the a_i are elements of a field.

Theorem 4.7. *A Vandermonde determinant has value* $\displaystyle\prod_{0<i<j\leq n} (a_i - a_j)$.

Note that if $a_i = a_j$ for $i \neq j$ then $D = 0$, while if all the a_i are distinct then D is non-zero.

Example 4.14: In the vector space $\mathbf{Z}_7^6$, consider the subspace generated by the vectors that are enumerated as rows of the following matrix.

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \end{pmatrix}$$

Since G has four rows, its rank is at most four. To see that it is four, consider the sub-matrix formed by the last four columns of G. Its determinant is not zero because it is a Vandermonde determinant with $n = 4$ and $a_i \neq a_j$ for $i, j = 1, 2, 3, 4$. Consequently the rows of G are linearly independent and they generate a four dimensional subspace of $\mathbf{Z}_7^6$. $\diamond$

4.3. **A Theorem on Linear Transformations.** A function from one vector space to another allows us to compare vector spaces. We will work with functions that respect the operations of addition and scalar multiplication in the sense that is made precise in the following definition.

Definition 4.12. *Let V and W be vector spaces over the field $\mathbf{Z}_p$. A function T from V to W is said to be a* ***linear transformation*** *if*
i) $T(\mathbf{x}+\mathbf{y}) = T(\mathbf{x}) + T(\mathbf{y})$ *for any* $\mathbf{x}$ *and* $\mathbf{y}$ *in* V*, and*
ii) $T(a\mathbf{x}) = aT(\mathbf{x})$ *for any* $\mathbf{x}$ *in* V *and any* a *in* $\mathbf{Z}_p$*.*

An important example for coding theory deals with a transformation from a vector space of polynomials to a vector space of n-tuples.

Example 4.15: Let $V = \mathbf{L}_7^3 = \{f \in \mathbf{Z}_7[x] : \deg(f) \leq 3\}$ and let $W = \mathbf{Z}_7^6$. Each f in V is of the form $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, where the a_i are in $\mathbf{Z}_7$. Define the function T by $T : \mathbf{L}_7^3 \to \mathbf{Z}_7^6$ where $T(f) = \big(f(1), f(2), f(3), f(4), f(5), f(6)\big)$. Then T is a linear transformation since

$$\begin{aligned} T(f+g) &= \big((f+g)(1), (f+g)(2), (f+g)(3), (f+g)(4), (f+g)(5), (f+g)(6)\big) \\ &= \big(f(1)+g(1), f(2)+g(2), f(3)+g(3), f(4)+g(4), f(5)+g(5), f(6)+g(6)\big) \\ &= T(f) + T(g), \end{aligned}$$

and

$$\begin{aligned} T(af) &= \big((af)(1), (af)(2), (af)(3), (af)(4), (af)(5), (af)(6)\big) \\ &= \big(a(f(1)), a(f(2)), a(f(3)), a(f(4)), a(f(5)), a(f(6))\big) \\ &= a\big(f(1), f(2), f(3), f(4), f(5), f(6)\big) \\ &= aT(f). \end{aligned}$$

◇

One method of defining a Reed–Solomon code makes use of a relation between linear transformations and subspaces that we present in the next theorem.

Theorem 4.8. *Let V and W be vector spaces over the field $\mathbf{Z}_p$ and $T : V \to W$ be a linear transformation. Then $T(\mathbf{V}) = \{\mathbf{w} \in W : \mathbf{w} = T(\mathbf{v})$ for some $\mathbf{v} \in V\}$ is a subspace of W.*

Example 4.16: With V, W and T as in Example 4.15, the set $T(V)$ is

$$\begin{aligned} T(V) &= \{\mathbf{w}_W : \mathbf{w} = T(\mathbf{v}) \text{ for some } \mathbf{v} \in V\} \\ &= \big\{\mathbf{x} \in \mathbf{Z}_7^6 : \mathbf{x} = \big(f(1), f(2), f(3), f(4), f(5), f(6)\big) \text{ for some } f \in \mathbf{L}_7^3\big\}. \end{aligned}$$

$T(V)$ is a subspace of $\mathbf{Z}_7^6$ in view of Theorem 4.8. ◇

4.4. **A Remark on Cardinality.** The cardinality of a set is the number of elements in the set. To count the number of elements in vector spaces and subspaces over a finite field, we use the following principle.

Theorem 4.9. ***Fundamental Principle of Counting (Multiplication Rule)****. Suppose we have to make a sequence of k choices, where the i-th choice can be made in n_i different ways, for $i = 1, 2, \ldots, k$. Then the number of possible sequences of choices is the product $n_1 n_2 \cdots n_k$.*

To choose a vector in $\mathbf{Z}_p^n$ we need to make n choices where each choice can be made in p different ways. Thus $\mathbf{Z}_p^n$ has exactly p^n elements. Let C be a subspace of $\mathbf{Z}_p^n$ and let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ be a basis of C. The vectors in C are exactly those of the form $a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \ldots + a_k\mathbf{x}_k$, where the a_i are in $\mathbf{Z}_p$. A standard result is the theory of vector spaces states that the representation of a vector in terms of a given basis is unique. Thus to choose a vector in C we must make k choices where each choice can be made in p different ways. This means C has p^k vectors.

Example 4.17: Let C be a 7 dimensional subspace of $\mathbf{Z}_{11}^{10}$. C has $11^7 = 19,487,171$ vectors and the cardinality of $\mathbf{Z}_{11}^{10}$ is $11^{10} \approx 2.5937 \times 10^{10}$. ◇

References

[1] Fraleigh, John B., A First Course in Linear Algebra, Addison Wesley Longman, 2002.
[2] Gallian, Joseph A., Contemporary Abstract Algebra, 5th Ed., Houghton Mifflin, Boston, MA, 2005.
[3] Pless, Vera, Introduction to the theory of Error-Correcting Codes, 3rd Ed., John Wiley, New York, NY, 1998.
[4] Pretzel, Oliver, Error-Correcting Codes and Finite Fields, Oxford University Press, Student Edition, Oxford, United Kingdom, 1996.
[5] Riley, Martyn and Richardson, Iain, An introduction to Reed–Solomon codes: principles, architecture, and implementation, 4i2i Communications Ltd., http:www.4i2i.comreed_solomon_codes.htm, copyrighted 1998, accessed 6/18/03.
[6] Roman, Steven, Introduction to Coding and Information Theory, Springer-Verlag, Renselaer, NY, 1997.
[7] Van Lint, J. H., Introduction to Coding Theory, Springer-Verlag, New York, NY, 1982.
[8] Walker, Judy L., Codes and Curves, Student Mathematical Library, American Mathematical Society, Providence, RI, 2000.