Error Correcting Codes and Finite Projective Planes

Patrick Fleming
Mathematics and Computer Science Department
South Dakota School of Mines & Technology
Rapid City, SD 57702
Patrick.Fleming@sdsmt.edu

April 17 & 18, 2009

Abstract

The Hamming Code was invented in the 1940's by Richard Hamming while he worked for Bell Labs. As the story goes, he would use the Bell model V computer over the weekends. At that time punch cards were checked for error with a parity check digit. If an error was found the computer would skip over that job and move on to the next. Richard Hamming found that having to resubmit a job because of a single error was frustrating. He thought that if a computer is smart enough know that there is an error, why not find and fix the error so that the job can run. The rest is history.

Projective planes were first proposed by Bernhard Reimann as an alternative to the Euclidean geometries fifth postulate: Given a line ℓ and a point P not on that line there is a unique line ℓ' containing P which is Parallel to ℓ. Reimann's alternative was the following: For every two lines there is a unique point of intersection. (No parallel lines!) This non-Euclidean geometry is axiomatically simple, yet rich in theory and applications. One such application is the generation of error correcting codes.

The first error correction code, the Hamming Code, is intrinsic to the projective plane of order two. Many of the most recently developed error correction codes, the "Space Time Codes" are related to Finite Semifield Planes. In this talk I will begin by defining a projective plane and how it relates to a finite fields. Second, I will discuss the history of the Hamming Code and how it works. I will then explain the relationship between the Hamming Code and projective planes. Finally, I will explore one minor generalization of this relationship.

1 Projective Planes

The first major push for an axiomatic system in geometry was put forward by Euclid about 300 BC. For centuries most geometry was done according to his axioms. The Parallel Line Postulate however, was a source of contention for many. The postulate states:

> Given a line ℓ and a point p not on that line, then there is a unique line containing p which is Parallel to ℓ.

It was not until the 1800's that Bernhard Riemann suggested that there may not be any parallel lines at all. It was from this idea that projective plane geometry was developed.

A *projective plane* is a collection of points and lines which satisfy the following three properties:

1. For any two lines there is a unique point of intersection.

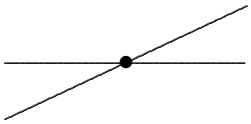


Figure 1: No parallel lines.

2. For any two points there is a unique line containing them.

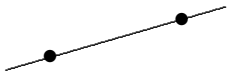


Figure 2: Two points determine a line.

3. There are 4 points of which no three are colinear.

Figure 3: Non-degeneracy condition.

One geometric way to describe a projective plane is to start with a 3-dimensional Euclidean space, and then to project through the origin. The 1-dimensional subspaces of the original 3-space are the points of the projective plane. The 2-dimensional subspaces of the original 3-space are the lines of the projective plane.

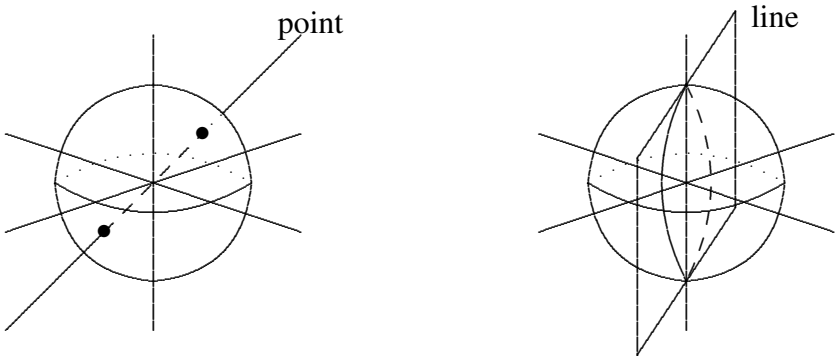


Figure 4: Geometric interpretation.

1.1 Fields

Projective planes can be given coordinates. A plane which has coordinates is called a coordinatized plane. For the planes discussed in this paper the coordinates come from an algebraic structure called a field.

A *Field* is a set R with binary operations $+, *$ s.t.
$a + b = b + a$ (addition is commutative).
$(a + b) + c = a + (b + c)$ (addition is associative)
$a + 0 = 0 + a = a$ (0 is the additive identity)
$a + (-a) = 0$ (every element has an additive inverse).
$a * b = b * a$ (multiplication is commutative).
$(a * b) * c = a * (b * c)$ (multiplication is associative)
$a * 1 = 1 * a = a$ (1 is the multiplicative identity)
$a * (a^{-1}) = 1$ (every element has an multiplicative inverse).
$a * (b + c) = a * b + a * c$ (multiplication is distributive).

The real numbers $\mathbb{R}$ and the complex numbers $\mathbb{C}$ are two example infinite fields. Finite fields exist for prime powers p^k. A finite field $\mathbb{F}_p = \{0, 1, 2, \ldots .p - 1\}, +, *$ has arithmetic done $mod\ p$. Binary arithmetic is done over the field of order 2, which is denoted $\mathbb{F}_2$.

1.2 Projective Plane Examples

Example: $PG(2, 2)$

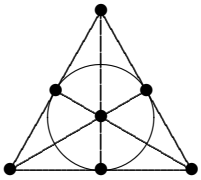


Figure 5: Fano Plane.

Notice that $PG(2, 2)$ has 7 points and 7 lines. There are no parallel lines; in fact, each pair of lines share a unique point, and each pair of points are contained by a unique line. Each line contains 3 points and each point lies on 3 lines. This plane, sometimes called the Fano Plane, is coordinatized by $\mathbb{F}_2$ (the field of order 2). It is also worth noting that the sum of two points is the third point on that line.

A Coordinatized Fano Plane:

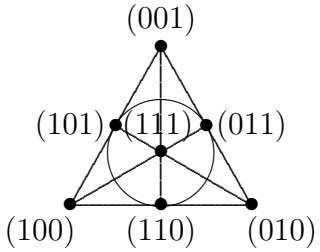


Figure 6: Coordinatized Fano Plane.

Example: $PG(3, 2)$

$PG(3, 2)$ is the projective plane which can be coordinatized by the field of order 3. This plane contains 13 points and 13 lines. Every pair of lines intersect at a unique point. Linear combinations of points on a line are, again, points on that line.

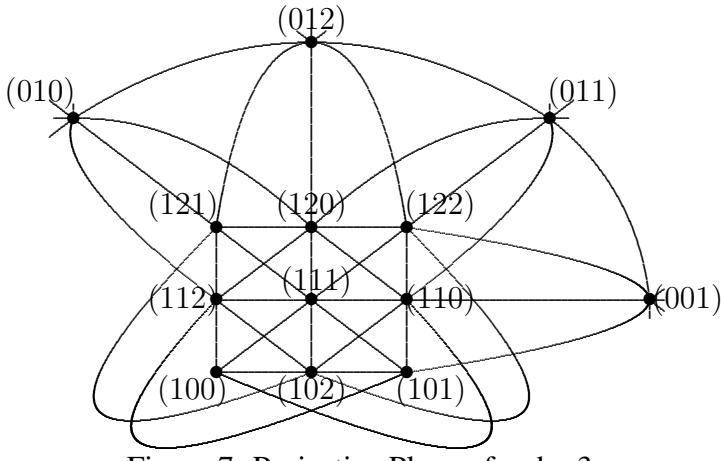


Figure 7: Projective Plane of order 3.

For any prime power $q = p^k$ there is a projective plane of order q. This Plane has q^2+q+1 points and $q^2 + q + 1$ lines. Each pair of points are contained on a unique line, and each pair of lines share a unique point. This plane can be coordinatized by $\mathbb{F}_q$ (the field of order q). There may be planes of order q which cannot be coordinatized by $\mathbb{F}_q$; however such planes do not always exist. This is a much longer story and is not discussed in this paper. All of the information above on projective planes can be found in Hughes and Piper's book "Projective Planes" [1].

2 The Coding Problem

The coding problem, simply stated, is to detect and correct errors in messages transmitted over a channel containing some form of static. The original error detection scheme was called a parity-check digit. In the 1940's, computer punch cards were used to program a computer to perform a given task. The static in the transmission channel was human error. To insure that each statement was intentional, every word was to have an even number of $1's$. Think of having a 4-digit command consisting of 0's and 1's and a single check-digit chosen so that the entire code work is even. For example: Say the word "yes "was to be (1,1,1,0) and the word "no"was to be (1,1,0,0). If when creating the punch cards an inopportune 1 was overlooked, the program would not do what was expected. To avoid this problem, "yes" would become (1,1,1,0,1) and the word "no" would become (1,1,0,0,0). Both words have an even number of $1's$ and so if either word is encoded with a single mistake, the mistake will be noted. When a computer receives the word(1,1,1,0,0) it will know that there is an error, although it will not know which digit is wrong.

During the late 1940's at Bell laboratories, Richard Hamming decided that a better system was needed. As folklore has it, Richard Hamming was working for Bell Labs. He was allowed to use the computer for research over the weekends. He would put together his punch cards during the week and submit them to be run over the weekend. This would work great as long as his punch cards were completely error-free; however, a single error would cause the computer to pass the job over and move on to the next. He would have to make corrections and resubmit his program at a later time. Richard Hamming thought that if the computer was smart enough to know that there was a mistake, why not have the computer find the mistake, correct it and continue running the program. He then created the first error correction code, the Hamming Code.

2.1 The Hamming Code

The Hamming Code has a vocabulary consisting of 16 words. Instead of adding a single check digit the Hamming Code adds three additional bits of redundancy producing code words of length 7. The Hamming Code consists of two matrices: The Hamming matrix and the Generator matrix.

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

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

To use the Hamming Code we start with one of the 16 words in $\mathbb{F}_2^4$ (binary strings of length 4), say w and encode it by multiplying on the left by the generator matrix G. We now have a codeword of length 7, $c = Gw$. It is this codeword c that we will transmit. When that message has been received, it may have been altered by static. The received word r is equal to the codeword $c + error$. To determine where the error has occurred we simply multiply by H.

2.2 A Hamming Code Example:

Lets say Alice wants to send a message to Bob which says "no". Alice will start with the numeric word for "no"; $w = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$.

She will then encode the word w with G to get the codeword c for the word "no".

$$Gw = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = c.$$

Alice will then send the codeword c over the channel.

$$c = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \rightarrow [static] \rightarrow \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = r = c + error.$$

Bob will receive the message r which may have an error. Bob will multiply r by H to determine if there is an error in the message r.

$$Hr = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Notice that $\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ is the third Column of H. This tells Bob that the third entry in the received word r is wrong. He therefore knows that her intended message was $c = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ which is the codeword for $w = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ which is the numeric word for "no".

For computational purposes, one may choose to order the columns of H according to their binary numeric value. In such a case, Hr is the binary representation for the number of the entry which contains the error. Note that G would have to be adjusted accordingly. We would also loose the nice block form of both H and G which allows us to see why the code works.

3 A Connection Between Planes and Codes

The Hamming Code and the projective plane of order the $PG(2,2)$ are closely related.

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \; G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \leftrightarrow$$

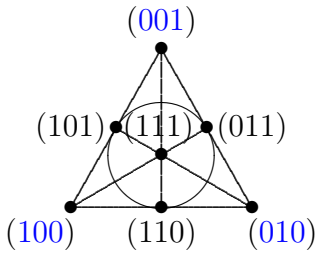
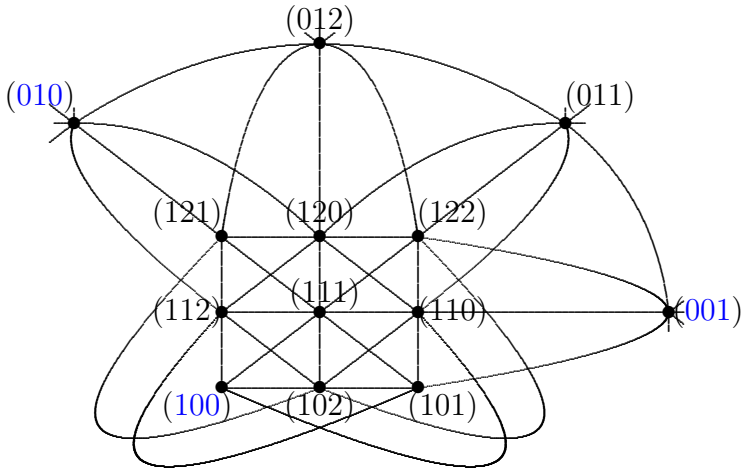


Figure 8: The Hamming matrix relates to the projective points.

The points of the projective plane make up the columns of the Hamming Matrix H. For convenience, we arrange H in block form. Let I be the 3×3 identity matrix. Let P be the 3×4 matrix with the remaining points of $PG(2,2)$ as its columns. $H = [P \;\; I]$. Therefore G is of the form $G = \left[\begin{smallmatrix} I \\ -P \end{smallmatrix}\right]$. Note that in G, I is the 4×4 identity matrix.

3.1 A code from $PG(3,2)$

$PG(3,2)$ consist of 13 points and 13 lines over the field of order 3. To construct a code from $PG(3,2)$, let I be the 3×3 identity matrix which corresponds to three of the points of the plane. Then let P be the 3×10 matrix with the remaining points of $PG(3,2)$ as its columns.



$$H = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 2 & 1 & 2 & 1 & 2 & 0 & 1 & 0 \\ 1 & 2 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 \end{bmatrix} = [P \;\; I]\,.$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 0 & 0 & 2 & 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 0 & 0 & 2 & 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} I \\ -P \end{bmatrix}.$$

Figure 9: $PG(3,2)$ Relates to the Matrix H.

In G, the matrix I is the 10×10 identity matrix. H and G provide a code with a vocabulary of size 3^{10} consisting of words which are strings of length 10 with an alphabet from $\mathbb{F}_3$. The codewords are length 13, and are attained by multiplication by G: $c = Gw$. The code can determine the location and scalar value of a single error.

At this point we can generalize this to codes arising from any projective plane which can be coordinatized by a field $\mathbb{F}q$. The points of the projective plane make up the columns of the Hamming Matrix H. Again for convenience, we arrange H in block form $H = [P \;\; I]$; therefore, G is of the form $G = \left[\begin{smallmatrix} I \\ -P \end{smallmatrix}\right]$.

In this form we can see how the encoding and error detection work. We encode a word$w$ with multiplication by G: $c = Gw = \left[\begin{smallmatrix} I \\ -P \end{smallmatrix}\right] w = \left[\begin{smallmatrix} w \\ -Pw \end{smallmatrix}\right]$. We determine the position and scalar value of a single error with multiplication by H.

Case 1, (no error): $r = c$

$$H \begin{bmatrix} w \\ -Pw \end{bmatrix} = [P \ \ I] \begin{bmatrix} w \\ -Pw \end{bmatrix} = [Pw + (-P)w] = [0] .$$

Case 2, (one error): $r = c + e_i = \left[\begin{smallmatrix} w \\ (-P)w \end{smallmatrix}\right] + s\,[e_i]$ where s is a scalar from $\mathbb{F}_3$ and $[e_i]$ is the vector with a 1 in the i^{th} position and 0's elsewhere.

$$H \left(\begin{bmatrix} w \\ -Pw \end{bmatrix} + s\,[e_i] \right) = [P \ \ I] \left(\begin{bmatrix} w \\ -Pw \end{bmatrix} + s\,[e_i] \right) = [P(w) + (-P)w] + sH\,[e_i] = sH\,[e_i] .$$

Since $sH\,[e_i]$ is a multiple of the i^{th} column of H, we know that if there is at most one error in r and Hr is a multiple s of the i^{th} column of H, that the error is in the i^{th} position in r and the scalar of the error is s. The codeword can now be recovered, $c = r - se_i$. Note that the block form of $H = [P \ \ I]$ and $G = \left[\begin{smallmatrix} I \\ -P \end{smallmatrix}\right]$ help to demonstrate how the encoding and error detection work; however, these block forms are not necessarily ideal for the error correction. In practice, we want to choose an ordering which will make it easy to compute s and e_i from sHe_i.

From these examples, we can see how to produce single error correcting codes of this form from projective planes over any Finite Field. The codes constructed in this way are called *projective codes*. If we want larger error correction capabilities, we do not have to leave the world of finite geometry; we may, however, need to look to larger dimensional spaces. I believe that there are many good undergraduate problems of this type. These problems can be theoretical or practical, geometric or strictly coding theoretic.

There are many good books about error correcting codes. Most coding theory texts will cover Hamming codes. J.H van Lint's "Introduction to Coding Theory" [3] and Steve Roman's "Coding and Information Theory" [2] are just a couple at the graduate level.

References

[1] HUGHES,D.R. AND PIPER. *Projective Planes*. Springer, Berlin-Heidelberg-New York, 1973.

[2] ROAMAN STEVEN. *Coding and Information Theory*. Springer-Verlag, Berlin-Heidelberg-New York, 1992.

[3] VAN LINT, J.H. *Introduction to Coding Theory*. Springer-Verlag, Berlin-Heidelberg-New York, 1992.