# ELEMENTARY CODING THEORY INCLUDING HAMMING AND REED-SOLOMON CODES WITH MAPLE AND MATLAB

Richard Klima
Department of Mathematical Sciences
Appalachian State University
Boone, North Carolina 28608
klimare@appstate.edu

Neil Sigmon
Department of Mathematics
Radford University
Radford, Virginia 24142
npsigmon@radford.edu

As our society continues to become more reliant on digital and computing technology for communication, the ability to transmit information in a reliable fashion continues to increase in importance. *Coding theory* is the study of methods for efficient and accurate transmission of information. Techniques in coding theory are currently used to ensure effective communication, including between, for example, wireless devices and satellites, in the encoding of music and information on compact discs, and in numerous other areas.

A *code* is a set of messages, called *codewords*, which can be transmitted electronically between two parties. An *error correcting* code is a code for which it is sometimes possible to detect and correct errors that occur during transmission of the codewords. Many of the current techniques in coding theory are applications of topics typically studied by students in undergraduate linear and abstract algebra courses. The calculations involved with these techniques can be tedious by hand or with only a hand-held calculator, thus making it difficult for students to see realistic examples that demonstrate their importance. However, this problem can be alleviated through the use of computing software such as Maple or MATLAB. Many techniques in coding theory can easily be implemented using Maple and MATLAB, and can thus provide students with opportunities to work with realistic examples of several different types of codes.

The purpose of this paper is to demonstrate how three well-known error correcting codes, Reed-Muller, Hamming, and Reed-Solomon codes, can be implemented using Maple or MATLAB. We have successfully taught each of these topics in courses involving applications of linear and abstract algebra. The use of Maple and MATLAB to perform the necessary computations has been an essential and integral part of the process.

**Reed-Muller Codes**

Reed-Muller codes were developed in 1954 by I. Reed and D. Muller, and were among the first error correcting codes with the ability to correct multiple transmission errors. Reed-Muller codes were used extensively in the Mariner 9 space probe when it transmitted photographs back to Earth after entering into an orbit around Mars.

Reed-Muller codes can be constructed using *Hadamard* matrices. An $n \times n$ matrix $H$ is called a Hadamard matrix if the entries in $H$ are all 1 or $-1$, and $HH^T = nI$, where $I$ is the $n \times n$ identity matrix. A *normalized* Hadamard matrix is one in which the first row

and column of $H$ contain only positive ones. The only normalized Hadamard matrices of orders one and two are $H_1 = [1]$ and $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Additional normalized Hadamard matrices can be constructed using the block matrix formula $H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$.

Beginning with a normalized Hadamard matrix $H$ of order $4k \geq 8$, a Reed-Muller code is constructed by the following steps:

1. Delete the first row and first column from $H$, and change all negative ones in $H$ into zeros. Call the resulting $(4k-1)\times(4k-1)$ matrix $A$.
2. Interchange all zeros and ones in $A$. Call the resulting $(4k-1)\times(4k-1)$ matrix $B$.
3. Attach a column of ones to the left of $A$, and a column of zeros to the left of $B$. Call the resulting $(4k-1)\times(4k)$ matrices $A$ and $B$, respectively.
4. Stack the matrices $A$ and $B$ together to form a new matrix, and include at the bottom of this new matrix an extra row of all zeros, and an extra row of all ones. Call the resulting $(8k)\times(4k)$ matrix $C$.

The rows of $C$ are the codewords in the Reed-Muller code. Suppose a codeword $c$ in this code is transmitted, and we receive a vector $r$ of the same length as $c$. If $r$ is not a codeword in the code, then we would assume that the fewest possible number of errors occurred during transmission, and correct $r$ (by trial and error) to the codeword from which it differs in the fewest positions. As long as fewer than $k$ errors occurred during transmission, then we are guaranteed that $c$ will be the only codeword from which $r$ differs in fewer than $k$ positions. As such, the code is said to be $(k-1)$-*error correcting*.

**Reed-Muller Codes with MATLAB**

Through the matrix functions that are readily available within MATLAB, it is easy to construct and correct errors in Reed-Muller codes. We demonstrate by generating the Reed-Muller code that was used in the Mariner 9 space probe when it transmitted photographs back to Earth after entering into an orbit around Mars. The code was constructed beginning with the normalized Hadamard matrix $H_{32}$ (with $k = 8$). Thus, the code consisted of 64 codewords, each of length 32 positions, and could correct up to 7 transmission errors. Before being transmitted, photographs taken by the probe were broken down into a collection of pixels. Each pixel was assigned one of 64 levels of grayness, and then encoded into one of the 64 codewords in the code. Due to space limitations, the MATLAB commands that we used to construct and correct errors in this code cannot be displayed here. However, the MATLAB M-file **reedmuller.m**, in which we give an interactive demonstration of how to construct and correct errors in this code, can be downloaded from [1].

**Hamming Codes**

Hamming codes were developed in 1950 by R. Hamming, and have been used extensively in the telecommunications industry and in the transmission of data between computers. Hamming codes are only able to correct one transmission error. However, Hamming codes are *perfect*, meaning that every vector that can possibly be received (of the correct length) is guaranteed to be uniquely correctable to the codeword that was originally transmitted (assuming no more than one transmission error occurred).

Codewords in Hamming codes are constructed as follows. Let $W$ be the vector space consisting of all binary $k$-tuples, and let $V$ be the vector space consisting of all binary $n$-tuples with $k < n$. Also, let $G$ be a $k \times n$ binary matrix of full row rank. The codewords in a Hamming code are then the vectors in the set $C = \{v \in V \mid v = wG \text{ for some } w \in W\}$, which is a subspace of $V$ of dimension $k$ and order $2^k$. The matrix $G$ used to construct the code is called a *generator* matrix for the code, and can be formed from the null space of a matrix $H$ known as the *parity check* matrix for the code. A convenient method for constructing $H$ is to let $n = 2^m - 1$ and $k = 2^m - 1 - m$ for some integer $m > 1$. With these parameters, $H$ can be the matrix of size $m \times (2^m - 1)$ whose columns form an ordered list of the binary expressions of the integers $1, 2, \ldots, 2^m - 1$, which, with leading zeros, yields all nonzero binary vectors of length $m$. For example, the following is the $4 \times 15$ parity check matrix for the Hamming code with $m = 4$.

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

The resulting generator matrix $G$ will be a matrix of size $(2^m - 1 - m) \times (2^m - 1)$, or $11 \times 15$ for the parity check matrix shown above, whose rows form a basis for the null space of $H$. Thus, if $c = wG$ is a codeword in a Hamming code, it follows that $Hc^T = 0$. This fact provides a convenient method for checking if a received vector is a codeword.

To demonstrate how transmission errors can be corrected in Hamming codes, suppose a Hamming codeword $c$ is transmitted, and we receive a vector $r$ of the same length as $c$. In addition, suppose $Hr^T \neq 0$, so we know $r$ is not a codeword. Assuming that exactly one transmission error occurred, say in the $i^{th}$ position, we can correct $r$ to $c$ as follows. Note first that $r = c + e_i$ for the error vector $e_i$ that contains a zero in every position except for a single one in the $i^{th}$ position. As a result, $Hr^T = He_i^T$, and thus $Hr^T$ and $He_i^T$ produce the same column in $H$. Therefore, to correct $r$ to $c$, we can compute $Hr^T$, and compare the result to the columns in $H$. The column in $H$ that is identical to $Hr^T$ will be in the same position in $H$ as the position of the single transmission error in $r$.

## Hamming Codes with Maple

Through the matrix functions that are readily available within Maple, it is easy to construct and correct errors in Hamming codes. Due to space limitations, the Maple commands that we used to construct and correct errors in Hamming codes cannot be displayed here. However, the Maple 10 file **hamming.mw**, in which we give an interactive demonstration of how to construct and correct errors in Hamming codes, can be downloaded from [1].

## Reed-Solomon Codes

Reed-Solomon codes were developed in 1960 by I. Reed and G. Solomon, and are claimed to be the most frequently used digital error correcting codes in the world. They are used extensively in the encoding of music on compact discs, have played an integral role in the development of high-speed supercomputers, and will be an important tool in the future for dealing with complex communication and information transfer systems. They were also famously used in the Voyager 2 satellite when it transmitted photographs back to Earth as it passed by the outer planets in our solar system (see Figure 1).

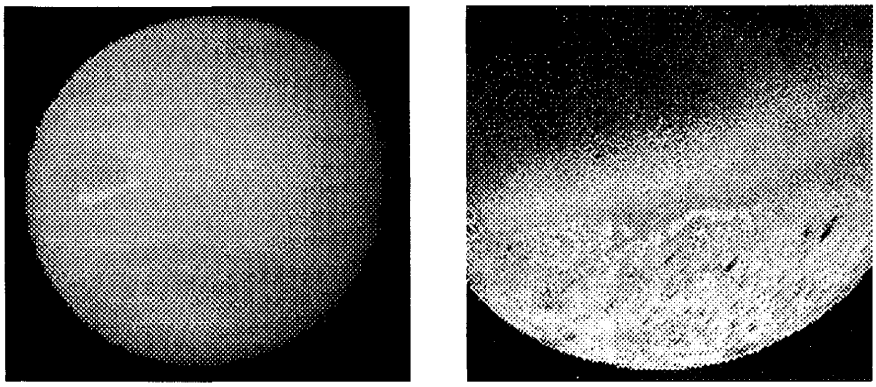

**Figure 1: Photographs taken by Voyager 2 of Neptune and one its moons (courtesy NASA/JPL)**

In the following discussion, we will describe how to construct and correct errors in Reed-Solomon codes using the code that was actually implemented in the Voyager 2 satellite.

Reed-Solomon codes are constructed using finite fields. The Reed-Solomon code that was implemented in Voyager 2 was constructed using a finite field $F$ containing $2^8 = 256$ elements formed using the primitive polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$. The codewords in this code are the polynomials that are obtained as multiples of the *generator* polynomial $g(x) = (x-a)(x-a^2)\cdots(x-a^{32})$ over $F$, where $a$ represents the primitive element used to generate the elements in $F$. That is, the codewords in the code are the polynomials $c(x)$ that are obtained by computing $c(x) = m(x)g(x)$ over $F$, where $m(x)$ represents any polynomial of degree less than $2^8 - 32 - 1 = 223$ with coefficients in $F$. Each codeword $c(x)$ is represented by a polynomial of degree less than $2^8 - 1 = 255$

with coefficients in $F$. The information transmitted in a codeword $c(x)$ is located in the list of coefficients of $c(x)$, which can be viewed as the elements in a vector of length 255 over $F$. Each polynomial coefficient in the codeword can be viewed as a binary polynomial of degree less than $\deg(p(x))=8$, and can be represented by a unique element in $F$. In the Voyager 2 satellite, full color images were digitized into binary vectors, which were expressed in blocks of length 8 and converted into coefficients for a codeword $c(x)$. For example, the binary block $(10111001)$ can be represented as the element $1+a^2+a^3+a^4+a^7$ in $F$, which can be used as a coefficient for a codeword $c(x)$. This technique can also be used in reverse to convert a codeword into a binary vector of length $\deg(p(x))\cdot(2^8-1)=2040$.

To correct errors that occur in this code, we can use the fact that for a codeword $c(x)$ in the code, $c(a^i)=0$ for $i=1,2,\ldots,32$. If $c(x)$ is transmitted, and we receive a polynomial $r(x)$ of an allowable degree (less than 255) that is not a codeword, an error polynomial $e(x)$ can be determined (assuming a given error-correction capability is not exceeded) that yields $c(x)=r(x)+e(x)$ over $F$. Determining $e(x)$ requires computations involving the Euclidean algorithm. The Reed-Solomon code used in Voyager 2 could correct up to $32/2=16$ polynomial coefficient errors, or up to 121 binary errors.

**Reed-Solomon Codes with Maple**

The extensive computations that are involved with implementing Reed-Solomon codes can be completed relatively quickly using Maple. Due to space limitations, the Maple commands that we used to construct and correct errors in Reed-Solomon codes cannot be displayed here. However, the Maple 10 file **reedsolomon.mw**, in which we give an interactive demonstration of how to construct and correct errors in Reed-Solomon codes, can be downloaded from [1].

**Conclusion**

We have shown how Maple and MATLAB can be used to assist in implementing several types of error correcting codes, each of which would be appropriate as an application in undergraduate linear or abstract algebra. The use of Maple or MATLAB would allow the instructor to use realistic examples without the overhead of tedious hand computations.

**References**

[1] R. E. Klima and N. P. Sigmon. Web site for ICTCM Maple and MATLAB files. Available at http://www.radford.edu/~npsigmon/ICTCM2006/downloads.htm.

[2] R. E. Klima, N. P. Sigmon, and E. L. Stitzinger. *Applications of Abstract Algebra with Maple and MATLAB, Second Edition.* CRC Press, LLC, to appear in Summer 2006.