



Investigation of Hamming, Reed-Solomon, and Turbo Forward Error Correcting Codes

by Gregory Mitchell

ARL-TR-4901

July 2009

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Adelphi, MD 20783-1197

ARL-TR-4901

July 2009

Investigation of Hamming, Reed-Solomon, and Turbo Forward Error Correcting Codes

Gregory Mitchell

Sensors and Electron Devices Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) July 2009		2. REPORT TYPE Final		3. DATES COVERED (From - To) FY09	
4. TITLE AND SUBTITLE Investigation of Hamming, Reed-Solomon, and Turbo Forward Error Correcting Codes				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Gregory Mitchell				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: RDRL-SER-E 2800 Powder Mill Road Adelphi, MD 20783-1197				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-4901	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT As data rate requirements for communications applications continue to increase and power requirements continue to fall, it becomes increasingly difficult to provide error-free communications over a noisy channel. Shannon's Limit provides the maximum error-free data rate achievable over an additive white Gaussian noise (AWGN) channel for a given signal to noise ratio (SNR). Forward error correcting (FEC) codes provide algorithms for encoding and decoding data bits, and help achieve data rates closer to Shannon's Limit. Three FEC codes are presented and their improvements in data rate are compared to tradeoffs in complexity and decoding lag. Different types of modulation are used to make comparisons in the performance of each FEC code.					
15. SUBJECT TERMS Hamming codes, RS codes, turbo codes, bit error rate, signal to noise ratio					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 24	19a. NAME OF RESPONSIBLE PERSON Gregory Mitchell
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code) (301) 394-2322

Contents

List of Figures	iv
1. Introduction	1
2. Linear Block Codes	1
2.1 Hamming Codes	2
2.2 Reed Solomon Codes	4
3. Turbo Codes	7
4. Bit Error Rate Analysis	10
5. Conclusions	15
6. References	16
List of Symbols, Abbreviations, and Acronyms	17
Distribution List	18

List of Figures

Figure 1. Block diagram of a generic encoding process for turbo codes (7).....	8
Figure 2. Block diagram for a $\frac{1}{2}$ rate RSC encoder (7).	9
Figure 3. Block diagram of the decoding process for turbo codes (7).....	9
Figure 4. BER vs. SNR for the (7,4) Hamming code modulated by BPSK, QPSK, and 8QAM modulations.	12
Figure 5. BER vs. SNR for the (31,16) Reed-Solomon code modulated by BPSK, QPSK, and 8QAM modulations.....	13
Figure 6. BER vs. SNR for multiple iterations of the turbo code decoding process using BPSK modulation (7).....	14
Figure 7. BER performance of turbo codes for various code rates modulated with BPSK. Fourteen iterations were used (7).....	15

1. Introduction

Wireless data transmission has become an essential part of many applications. Whether it is a wireless internet connection, cell phone conversation, radio broadcast, or some military application, the need for data to be decoded error-free across a wireless medium is vital. Over a channel characterized by Gaussian noise, a limit exists that tells us the maximum channel capacity, or error free transmission rate, possible for a given signal-to-noise ratio (SNR) and channel bandwidth. Known as Shannon's Limit, developed by Claude Shannon in his paper "A Mathematical Theory of Communication," the equation is (1)

$$C = B \log_2(1 + SNR) \quad (1)$$

The problem is that noise within the channel can cause errors in the data during transmission. To combat this problem, we use forward error correcting (FEC) codes to detect and correct these potential errors at the receiver. FECs add redundancy to data in the form of parity bits, and in general, the more parity bits generated per data bit, the more potential errors can be detected and corrected. Different FECs use different methods to check and correct errors in an attempt to close in on the limit proposed by Shannon for data transmission. Here, Hamming codes, Reed-Solomon codes, and turbo codes are examined, and their differences are briefly discussed, along with the pros and cons of each approach.

2. Linear Block Codes

Linear block codes (LBC) are often referred to as (n,k) codes, where n is the length of the block, and k is the number of information bits in the case of a binary code. This means that to transmit data, an LBC uses 2^k codewords, which make up a subset of 2^n possible codewords (1). The general form of an LBC is

$$C = DG \quad (2)$$

where C is the codeword, D is the k -bit message, and G is the generator matrix that creates the parity check bits from the data bits (2).

The code rate of a code is the ratio $R = \frac{k}{n}$, or the number of input bits over the number of output bits of an FEC encoder. This ratio can be thought of as the percentage of time used to send actual information bits through a channel. The code rate improves with larger data block lengths, but larger block lengths have a proportionally larger number of expected errors.

Improving the code rate increases the chance of decoding error because with more information bits and fewer parity bits, more undetectable errors are expected.

The optimal detector for a block code, given a binary symmetric channel, finds the codeword that is closest to the received block of n bits. The Hamming distance between two words measures the number of coordinates in which they differ. For example, the distance between 1110010 and 0110101 is four.

Decoding errors will occur when noise in the channel changes the transmitted codeword so that its Hamming distance is closer to one of the other 2^k codewords. For this reason, the distance between codewords is relevant to the probability of a decoding error. The minimum Hamming distance of a code is the smallest distance between any of its 2^k codewords. If a code consists of two codewords that are a distance of $d_{\min}$ apart, a word has to be within distance $\frac{(d_{\min} - 1)}{2}$ of a codeword in order to guarantee that it is mapped to that codeword. Therefore it follows that a code can correct up to $t = \frac{(d_{\min} - 1)}{2}$ errors (2).

Since the number of errors a code can correct is directly related to $d_{\min}$, it is advantageous to find the code of a given size with the largest $d_{\min}$. Such a code best utilizes the Hamming space, the total set of 2^n possible words. There is an inequality called the Hamming bound, which states that if there is an (n, k) code with an alphabet of q elements and $d_{\min} = 2t + 1$ then (1),

$$q^n \geq q^k \sum_{i=0}^t \binom{n}{i} (q-1)^i \quad (3)$$

For each of the q^k codewords there are exactly $\binom{n}{i} (q-1)^i$ codewords that are exactly distance i from it. Therefore, the number on the right of the inequality is the total number of words that are at most a distance of t from a codeword. Since the total number of words cannot exceed q^n , the inequality follows immediately. For a binary code, (3) simplifies to (1)

$$2^{n-k} \geq \sum_{i=0}^t \binom{n}{i} \quad (4)$$

Codes that attain equality in the Hamming bound are called perfect codes. A perfect t -error correcting code has the property that every word in the Hamming space lies within a distance of t from exactly one codeword.

2.1 Hamming Codes

Hamming codes are the earliest codes capable of actually correcting an error detected at the receiver. However, by definition they are limited so that they can detect and correct only a single error. This arises from the fact that $d_{\min} = 3$, so that if more than one bit is corrupted then

the erroneous word is the same as one of the other 2^k codewords, and the decoding algorithm will assume no error (3). The (7,4) Hamming code is a famous single error correcting code because it belongs to the class of perfect codes.

The equations for block decoding error probability, P_e , and bit decoding error probability, P_b , are (4)

$$P_e = \sum_{j=2}^n \binom{n}{j} p^j (1-p)^{(n-j)} \quad (6)$$

$$P_b = \left(\frac{1}{n}\right) \sum_{j=2}^n \beta_j \binom{n}{j} p^j (1-p)^{(n-j)} \quad (7)$$

where p is the probability a bit is received in error, and β_j is the average number of bit errors in the decoded words which is approximated as j . P_e is the probability that two or more bits are received in error for a codeword of length, n , and P_b is a weighted version that translates block error into bit error for linear block codes.

Hamming Codes use syndrome decoding such that

$$\vec{s} = H\vec{r} \quad (8)$$

determines the syndrome where H is a parity check matrix and

$$\vec{r} = \vec{b} + \vec{e} \quad (9)$$

is the received bit stream where $\vec{e}$ is an error vector created by the noisy channel, and $\vec{b}$ is the transmitted codeword.

H is created by a linear combination of the k message bits. For the (7,4) Hamming code, if $b_1b_2b_3b_4$ are the information bits, then the parity bits are (3)

$$\begin{aligned} b_5 &= b_1 + b_3 + b_4 \\ b_6 &= b_1 + b_2 + b_4 \\ b_7 &= b_2 + b_3 + b_4 \end{aligned} \quad (10)$$

Then H is generated by

$$\begin{aligned} b_5 + b_5 &= b_1 + b_3 + b_4 + b_5 = 0 \\ b_6 + b_6 &= b_1 + b_2 + b_4 + b_6 = 0 \\ b_7 + b_7 &= b_2 + b_3 + b_4 + b_7 = 0 \end{aligned} \quad (11)$$

$$H = \begin{bmatrix} 1011100 \\ 1101010 \\ 0111001 \end{bmatrix} \quad (12)$$

If $\vec{s} = 0$, then $\vec{r}$ is a known codeword, and the decoder determines there were no errors upon transmission. If $\vec{s} \neq 0$, then the value of $\vec{s}$ gives the bit that is in error for a single error, and the decoder knows to flip this bit. If there are two errors, then $\vec{s}$ gives the sum of the two rows that were in error. Since $\vec{s} \neq 0$, the code can detect when two errors occur but there is no way to know which bits are in error (3).

An advantage of Hamming codes is that encoding and decoding are easy to implement. They would be effective as a simple and efficient code over a channel where it is known that errors are burst-free and tend to be widely dispersed. Disadvantages of Hamming codes are that they are very ineffective for low SNR, where the received signal level is very low. These types of conditions will tend to cause more frequent errors. Also, Hamming codes do very poorly against the bursts of errors caused by fading over a channel.

2.2 Reed Solomon Codes

Reed-Solomon (RS) codes are non-binary codes; that is, a codeword is defined in terms of multi-bit symbols rather than bits. Such codes can achieve a very large Hamming distance. The RS decoder corrects the entire symbol, whether the error was caused by one bit being corrupted or by all of the bits being corrupted. Thus, if a symbol is wrong, it might as well be wrong in all of its bit positions. This gives RS codes tremendous burst-noise advantages over binary codes. Burst-noise is relatively common in wireless communication due to fading. The code minimum distance for RS code is given by

$$d_{\min} = n - k + 1 \quad (13)$$

where k is now the number of data symbols being encoded, and n is the length of the codeword. The code can correct up to t symbol errors, where t is given by

$$t = \frac{n - k}{2} \quad (14)$$

This equation shows that a codeword needs $2t$ parity symbols to correct t errors (1).

The RS bit error probability, P_E , in terms of the channel bit-error probability, p , is given by (5)

$$P_b \approx \frac{1}{2^m - 1} \sum_{j=t+1}^{2^m-1} j \binom{2^m-1}{j} p^j (1-p)^{2^m-1-j} \quad (15)$$

where t is the maximum number of symbol errors that can be corrected, and the symbols are made up of m bits each.

In order to understand RS encoding and decoding, finite fields or Galois Fields (GF) must be used, and the following description of the encoding and decoding process is heavily based on those outlined in (6).

For any prime number p there exists a finite field, $GF(p^m)$, where m is a positive integer. There are numbers in the finite field denoted by the power of α other than 0 and 1. Finite fields only contain 2^m elements, and the field is closed under multiplication. The condition that closes the set of field elements under multiplication is shown by the following irreducible polynomial

$$\alpha^{(2^m-1)} = 1 = \alpha^0. \quad (16)$$

Thus, any field element that has a power equal to or greater than $2^m - 1$ can be reduced to a power less than $2^m - 1$ by the following equation

$$\alpha^{(2^m+n)} = \alpha^{(2^m-1)}\alpha^{n+1} = \alpha^{n+1}. \quad (17)$$

RS codes are usually written in the following form

$$R - S(n, k) = (2^m - 1, 2^m - 1 - 2t), \quad (18)$$

where, k is the number of data symbols being encoded, n is the length of the codeword, t is the maximum error correction capability, and the symbols are made up of m bits each.

The generator polynomial for an RS code takes the following form:

$$g(X) = g_0 + g_1X + g_2X^2 + \dots + g_{2t-1}X^{2t-1} + X^{2t}. \quad (19)$$

The degree of the generator polynomial is the number of parity symbols, which is $2t$. Therefore, there are exactly $\alpha = 2t$ roots of the polynomial.

The above process follows the multiplication and addition rules of the finite field. Also, in a binary field $+1 = -1$, so we can express $g(X)$ as

$$g(X) = \alpha^3 + \alpha^1X + \alpha^0X^2 + \alpha^3X^3 + X^4 \quad (20)$$

Thus, the resulting codeword polynomial $U(X)$ can be written as

$$U(X) = p(X) + X^{n-k}m(X) \quad (21)$$

where $m(X)$ is the message polynomial, multiplying by X^{n-k} right shifts the polynomial by $n - k$ positions, and the parity polynomial is

$$p(X) = X^{n-k}m(X) \bmod g(X) \quad (22)$$

Consider a (7,3) codeword for the message polynomial given by equation 23. To generate a codeword for the following three symbol message, multiply $m(X)$ by $X^{n-k} = X^4$ giving

$$m(x) = \underbrace{010}_{\alpha^1} \underbrace{110}_{\alpha^3} \underbrace{111}_{\alpha^5}$$

$$m(x)X^{n-k} = \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6 \quad (23)$$

Divide the shifted message by the generator polynomial to get the parity polynomial

$$p(X) = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 \quad (24)$$

Adding the shifted message with the parity polynomial yields the codeword

$$U(X) = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6 \quad (25)$$

If $U(X)$ gets corrupted by two errors due to channel noise, then the error polynomial for this example is shown by

$$e(X) = \sum_{n=0}^6 e_n X^n \quad (26)$$

The received bit stream becomes

$$r(X) = U(X) + e(X) \quad (27)$$

In RS codes, there are twice as many unknowns than in binary coding. In binary coding, one only needs to find the error location and then flip the bit. In non-binary coding, one must know the error value to correct the error.

To decode the received bit stream, a syndrome S will be made of $n-k$ symbols. Since the codeword is made by multiplying the message polynomial by the generator polynomial, the roots of the generator polynomial must be the roots of the transmitted codeword. Evaluating $r(X)$ at the roots of the generator polynomial should yield a syndrome of zero with no errors and a nonzero syndrome if errors exist. The calculation of syndrome is shown by

$$S_i = r(X) \Big|_{X=\alpha^i} = r(\alpha^i) \quad i = 1, \dots, n-k \quad (28)$$

The error polynomial can be written as

$$e(X) = e_{j1} X^{j1} + e_{j2} X^{j2} + \dots + e_{jv} X^{jv} \quad (29)$$

where, e_{jl} is the error value, X^{jl} is the error location, and v is the number of errors. The syndrome calculation provides the $2t$ error polynomial equations. We have t error values and t error locations, which create $2t$ simultaneous equations. The error locator polynomial is needed to find the location of errors, and is given by

$$\sigma(X) = (1 + \alpha^1 X)(1 + \alpha^2 X) \dots (1 + \alpha^v X) \quad (30)$$

$$\sigma(X) = 1 + \sigma_1 X + \sigma_2 X^2 + \dots + \sigma_v X^v \quad (31)$$

The roots of $\sigma(X)$ are inverses of the error location numbers of the error pattern, $e(X)$.

To find the error locations, use the autoregressive technique. Autoregressive means using the first t syndromes to predict the next syndrome. Test the roots of this equation with the field elements. Any element X that yields $\sigma(X) = 0$ is a root, which reveals the error.

The elements of the error vector are given as

$$E^i = \frac{\omega(\alpha^{-i})}{\sigma'(\alpha^{-i})} \quad (32)$$

where $\omega(x)$ is the evaluator polynomial and $\sigma'(x)$ is the inverse of the i^{th} root of the locator polynomial such that

$$\sigma(x)S(x) = \omega(x) \pmod{x^r} \quad (33)$$

where r is the length of the syndrome.

Now that the error locations and their corresponding error values are known, the codeword can be corrected as

$$C^i = r^i - E^i \quad (34)$$

RS codes are mainly non-binary block codes that are especially effective against burst-errors. The coding efficiency mainly increases with code length. RS codes can be used for long block lengths with less decoding time than other codes because RS codes work with symbol-based arithmetic. This makes RS coding harder to implement, but it provides better throughput.

3. Turbo Codes

Turbo codes are part of a class known as convolutional codes. Convolutional codes generate codewords as the convolution of the incoming message data with the impulse response of a shift register. As an incoming bit propagates through the shift register, it influences several output bits thereby spreading the information over several adjacent output bits. This means an error in any one bit can be overcome at the receiver without any information being lost (2).

Turbo Codes are well suited for long distance and low power wireless communications because they achieve a very low bit error rate (BER) at very low SNR. This means that data is still transmitted nearly error-free with a low energy signal, and this characteristic has led turbo

codes to be implemented in applications such as deep space communications and third-generation cellular standards.

The encoding process is what helps turbo codes achieve such dramatic results, and the following description of both the encoding and decoding of turbo codes is heavily based on that in (7). By using two independent encoders separated by an interleaver, as shown in figure 1, turbo codes actually use two unique codewords instead of just one to encode data. The interleaver scrambles the data bits prior to input into the second encoder in order to generate a second codeword that differs from the first. By generating two distinct codewords, turbo encoding increases the probability of generating a codeword of high Hamming weight. This means that the codeword has a large number of ones relative to the number of zeros, making it more distinctive to the decoder. Although both codewords are not guaranteed to have high Hamming weight, by generating two codewords, one of them is likely to be of high Hamming weight. This is one of the main features that make turbo codes so powerful.

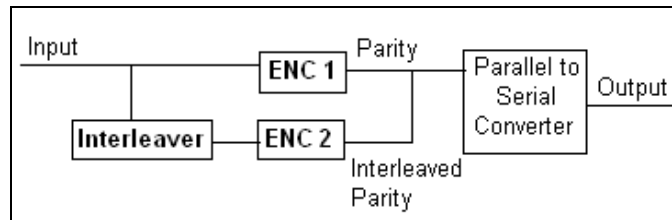


Figure 1. Block diagram of a generic encoding process for turbo codes (7).

Turbo encoding uses Recursive Systematic Convolutional (RSC) encoders. These encoders are systematic because they regenerate an input bit as one of the outputs, and recursive because they feed one of the two parity outputs back to the input. The RSC encoder depicted in figure 2 works by performing an exclusive OR of the input, with a subset of the bits stored in the D flip-flop registers. Since this RSC encoder has a code rate of one-half, the amount of information bits per second transmitted over a Gaussian channel will be half the size of the total codeword.

The decoding process for turbo codes must also be more complex to decode the codewords generated by the encoder. A turbo decoder uses an iterative decoding algorithm that requires two parallel Soft-Input Soft-Output (SISO) processors. The upper SISO functions on the received output from first encoder and the lower SISO functions on the received output from the second encoder of figure 1. A soft input or output is one that can take any value, and is not limited only to integers like the codeword input stream. The lower SISO processor takes the interleaved output from SISO one, as well as the output from the second encoder shown in figure 3. This information sharing makes the decoding process iterative and improves the decoder's overall performance.

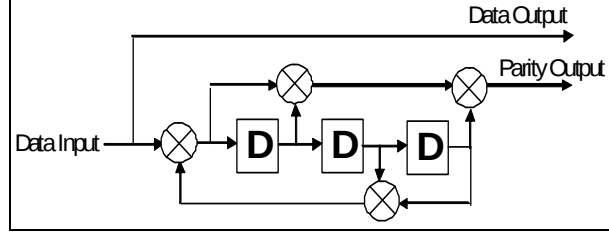


Figure 2. Block diagram for a $\frac{1}{2}$ rate RSC encoder (7).

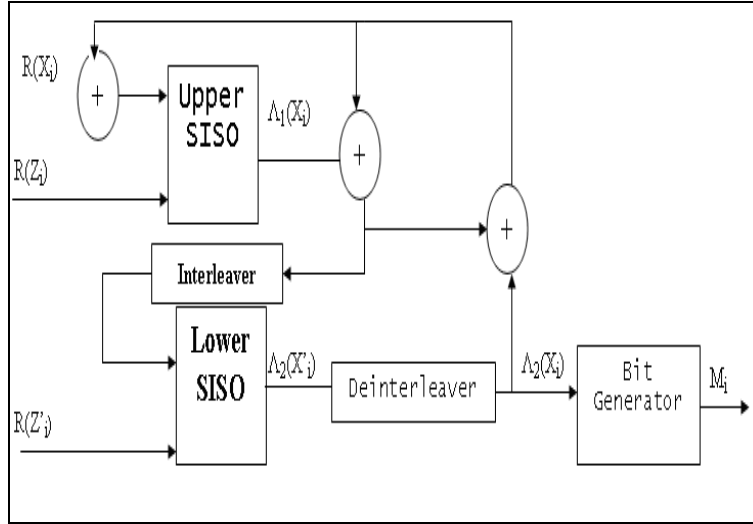


Figure 3. Block diagram of the decoding process for turbo codes (7).

The inputs and outputs of the SISO processors take the form of Log Likelihood Ratios (LLR), which is a ratio of probabilities. The LLR of the inputs will be

$$R(U_i) = \ln \left(\frac{P(Y_i | U_i = 1)}{P(Y_i | U_i = 0)} \right) \quad (35)$$

where U represents the transmitted bit and Y represents the received bit. The numerator is the conditional probability that the input bit is received if the output bit from the transmitter was a one, and the denominator is the conditional probability that the input bit is received if the output bit was a zero.

The LLR at the output of the SISO processors will be

$$\lambda(X_i) = \ln \left(\frac{P(X_i | Y_1 \cdots Y_n)}{P(X_i | Y_1 \cdots Y_n)} \right) \quad (36)$$

where X is the information bit and Y is the received bit. The numerator is the conditional probability that the information bit is a one, given all the received inputs from the codeword, and

the denominator is the conditional probability that the information bit is a zero, given all the received inputs from the codeword. The deinterleaver will reorganize the SISO outputs into the right order, and the bit generator can determine whether the input bit is a one or zero based on the value of $\Lambda_2(X_i)$. If $\Lambda_2(X_i) > 0$, then the input bit is a one, and if $\Lambda_2(X_i) < 0$, then the input bit is a zero.

The superior performance of turbo codes is due mainly to the unique encoding process, but also to the fact that the decoding process is iterative. However, the increase in iterations follows a law of diminishing returns. Every time the number of iterations is increased, the improvement over the BER achieved decreases. Also, continually increasing the number of iterations causes decoder delay, and this in turn decreases the throughput that turbo codes are able to achieve; because with this added delay, fewer bits can be decoded per second (7).

Another way to improve the BER is to add more parity bits to each input bit. As the code rate decreases, the BER decreases at an even lower SNR because more errors can be corrected by adding more parity bits. However, decreasing the code rate by too much will significantly decrease the number of information bits a channel can transmit per second because the channel will now be filled with more parity bits. In short, there will always be a tradeoff between BER and data throughput (7).

Though turbo codes seem to be an immensely powerful FEC code, there are issues that make them non-ideal in certain situations. The presence of two turbo codes increases the probability of high Hamming weight codewords, but will also double the number of low Hamming weight codewords present. Although the BER decreases dramatically for low SNR, the presence of low-weight codewords begins to dominate the performance of turbo codes at higher SNR, causing the BER curve to flatten out at these values. Though the encoding and decoding processes outlined previously lowers the BER, implementing these practices is costly, and may not be desirable. Furthermore, the decoding algorithm is complex, and more iterations require greater amounts of memory, causing the implementation of turbo codes to be more expensive than linear block codes. Lastly, in order to calculate the LLRs necessary for the decoder, the channel's characteristics must be known, which means that you must be familiar with the channel over which the data is being transmitted (7).

4. Bit Error Rate Analysis

So far the encoding and decoding process for Hamming, RS, and turbo codes has been presented, along with the advantages and disadvantages of each approach. Now the performance of each code in terms of BER for different levels of SNR will be examined. Equations 7 and 15 are used in order to calculate BER performance for Hamming and RS codes equations, respectively. The probability that a bit is flipped during transmission, p , in equations 7 and 15, is determined by

the type of modulation used on the channel. The following probability of error for BPSK, QPSK, and 8QAM modulations assume a binary symmetric additive white Gaussian noise (AWGN) channel. The following are derived by John Proakis in (8) and make use of the following relationship

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{t^2}{2}} dt \quad (37)$$

BPSK:

$$P_e = Q\left(\sqrt{\frac{2\mathcal{E}_b}{N_o}}\right) \quad (38)$$

where $\frac{\mathcal{E}_b}{N_o}$ is the SNR ratio per bit.

QPSK:

$$P_e = 2Q\left(\sqrt{\frac{2\mathcal{E}_b}{N_o}}\right) \left[1 - \frac{1}{2}Q\left(\sqrt{\frac{2\mathcal{E}_b}{N_o}}\right)\right] \quad (39)$$

8QAM:

$$P_{\sqrt{8}} = 2\left(1 - \frac{1}{\sqrt{8}}\right)Q\left(\sqrt{\frac{3\mathcal{E}_b}{N_o(8-1)}}\right) \quad (40)$$

$$P_e = 1 - (1 - P_{\sqrt{8}})^2 \quad (41)$$

For equations 7 and 15, use the substitution $P_e = p$ to calculate the BER versus SNR curves.

Figure 4 shows the improvement in bit error probability between an uncoded message and a message using the (7,4) Hamming code. Even this very rudimentary code shows a significant improvement in the achievable BER for a given SNR for all three types of modulation. For a BER of 10^{-6} , the improvement of SNR is roughly 3.5 dB, which means less than half the signal strength is needed to achieve the same reliability in decoding error. This factor of improvement means less transmission power is needed over a given distance, or that the transmission range can be increased and still achieves the same BER as at a much closer range without the code.

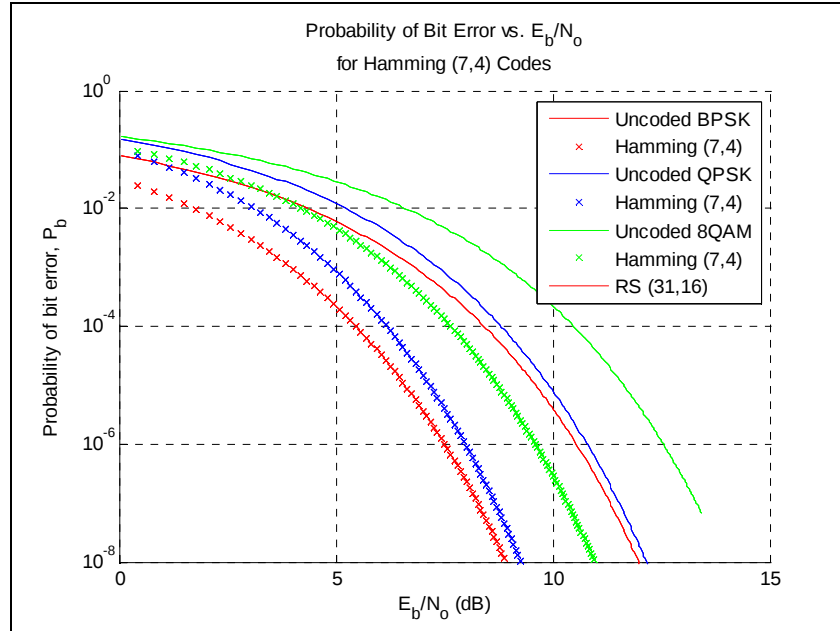


Figure 4. BER vs. SNR for the (7,4) Hamming code modulated by BPSK, QPSK, and 8QAM modulations.

Figure 5 shows an even greater improvement in SNR for the (31,16) RS code in comparison to a message transmitted without any form of FEC code. For a BER of 10^{-6} , the improvement in SNR is 6.5 dB. This almost doubles the improvement seen by the (7,4) Hamming code. Since RS codes are extremely effective against burst-errors and offer significant improvements in the SNR needed to achieve a given BER, the code is very powerful yet still relatively simple to implement in terms of hardware. RS codes may not compare to turbo codes in their performance in the very low SNR values encountered—in deep space missions, for example—but for many other applications such as data storage and internet data transmission, they are a very attractive option.

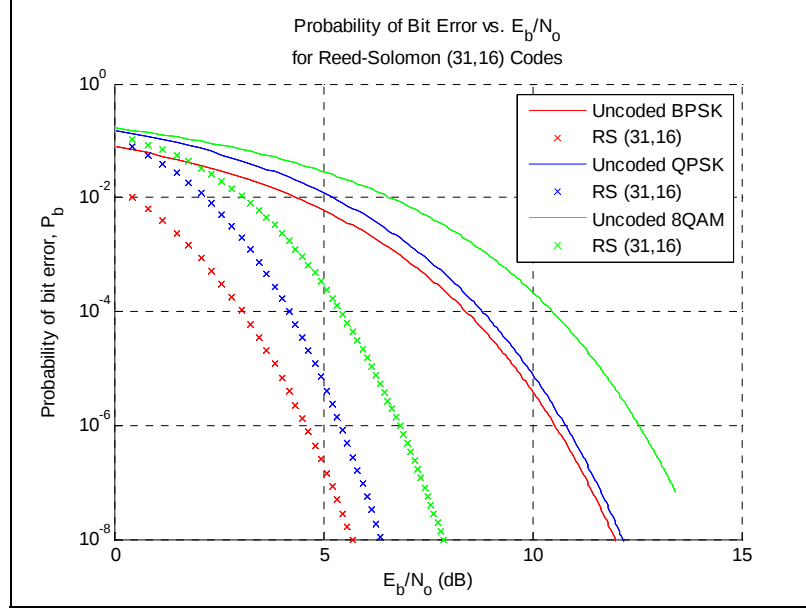


Figure 5. BER vs. SNR for the (31,16) Reed-Solomon code modulated by BPSK, QPSK, and 8QAM modulations.

Figure 6 shows achievable BER for turbo codes using different numbers of decoding iterations. By performing more iterations during decoding, the BER drops dramatically, even for extremely low SNR values. However, the law of diminishing returns is clearly evident, as the number of iterations is increased. Figure 6 shows that a bit error rate (BER) of 10^{-5} can be achieved for an SNR of 0.8 after 10 iterations by the decoder. This is a very good BER for a signal level that is, in fact, weaker than the noise over an AWGN channel. In fact, to achieve a BER of 10^{-6} , a SNR of 1 dB is all that is needed for 10 iterations, where even the (31,16) RS code needed a SNR of 4.5 dB. The ability to achieve such low BER for extremely low SNR is what makes turbo codes so attractive for applications such as deep space satellites. This type of application also provides the liberty to introduce as much delay as is needed at the decoder because it is not a real-time data application. Therefore, more iterations could be performed to achieve a low BER for signals coming in under 1 dB SNR.

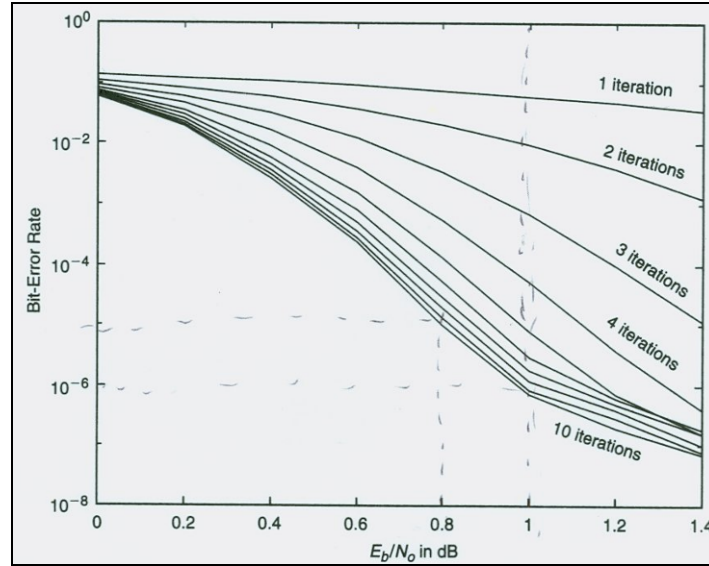


Figure 6. BER vs. SNR for multiple iterations of the turbo code decoding process using BPSK modulation (7).

Another way to improve the achievable BER for low SNR is to decrease the code rate. This means using more parity bits and less information bits in a codeword. When the code rate is decreased, the information throughput is sacrificed because there are less information bits being sent per transmission. But in an application like deep space satellites, this may be an acceptable tradeoff to increase BER at astronomical distances. Figure 7 shows how decreasing the code rate can achieve even better results for turbo codes than are depicted in figure 6. At a code rate of one-fifth a BER of 10^{-6} is achievable at an SNR of 0.6 dB for 14 decoding iterations. At this point a receiver can receive a transmission whose power is far below the noise floor and still achieve an acceptable BER.

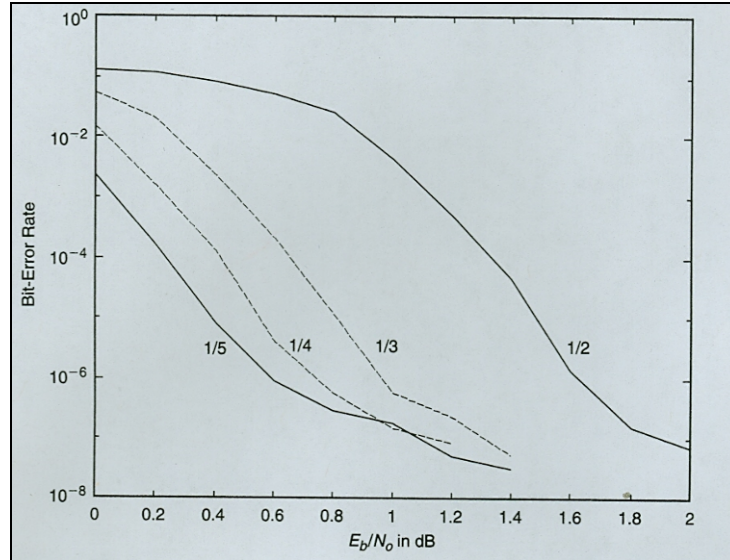


Figure 7. BER performance of turbo codes for various code rates modulated with BPSK. Fourteen iterations were used (7).

5. Conclusions

FEC codes are a powerful tool in combating transmission errors caused by a noisy channel. Using FEC codes allows communications to achieve the same level of transmission reliability, quantified by the BER, at lower output power levels. This could potentially give a large cost benefit because it reduces the amount of power needed for the transmission of data.

There are tradeoffs for every code that have to be weighed against the application they are being used for. This report investigated Hamming Codes, RS codes, and turbo codes, in the areas of both implementation and performance. There is no single FEC code that is an optimum solution for every application, and many factors must be weighed before a decision is made on which code to use. In general, the more effective a FEC code is at combating transmission errors, the more expensive it is to implement in terms of hardware, and the more complicated its encoding and decoding process become. Things like decoding delay and decreased throughput must also be considered when choosing between the different FEC codes that are available.

6. References

1. McEliece, Robert. *The Theory of Information and Coding*; Student edition; Cambridge University Press, 2004.
2. Pratt, Timothy; Bostian, Charles; Allnutt, Jeremy. *Satellite Communications*; 2nd ed.; John Wiley and Son, Inc., 2003.
3. Leon-Gracia, Alberto; Widjaja, Indra. *Communication Networks*; McGraw-Hill, 2004.
4. Xiong, Wenhui; Matolak, David. Performance of Hamming Codes in Systems Employing Different Code Symbol Energies. *IEEE Wireless Communications and Networking Conference*, 2005.
5. "Block FEC Coding for Broadcast Applications." 6 May 2005.
http://www.digitalradiotech.co.uk/fec_coding.htm (accessed 2009).
6. Wicker, Stephen; Bhargava, Vijay K. *Reed-Solomon Codes and Their Applications*; IEEE Press, 1994.
7. Dowla, Farid. *Handbook of RF and Wireless Technologies*; Butterworth-Heinemann, October 2003.
8. Proakis, John. *Digital Communications*; 4th ed.; McGraw-Hill, 2001.

List of Symbols, Abbreviations, and Acronyms

AWGN	additive white Gaussian noise
BER	bit error rate
FEC	forward error correcting
GF	Galois Fields
LBC	Linear block codes
LLR	Log Likelihood Ratios
RS	Reed-Solomon
RSC	Recursive Systematic Convolutional
SISO	Soft-Input Soft-Output
SNR	signal to noise ratio

NO OF. COPIES	ORGANIZATION	NO OF. COPIES	ORGANIZATION
1 ELEC	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218	1	US ARMY RSRCH LAB ATTN RDRL CIM G T LANDFRIED BLDG 4600 ABERDEEN PROVING GROUND MD 21005-5066
1	DARPA ATTN IXO S WELBY 3701 N FAIRFAX DR ARLINGTON VA 22203-1714	7	US ARMY RSRCH LAB ATTN IMNE ALC HRR MAIL & RECORDS MGMT ATTN RDRL CIM L TECHL LIB ATTN RDRL CIM P TECHL PUB ATTN RDRL SER E R DEL ROSARIO ATTN RDRL SER E G MITCHELL (3 COPIES) ADELPHI MD 20783-1197
1 CD	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) THE PENTAGON WASHINGTON DC 20301-3080	TOTAL: 16 (1 ELEC, 1 CD, 14 HCS)	
1	US ARMY RSRCH DEV AND ENGRG CMND ARMAMENT RSRCH DEV AND ENGRG CTR ARMAMENT ENGRG AND TECHNLGY CTR ATTN AMSRD AAR AEF T J MATTS BLDG 305 ABERDEEN PROVING GROUND MD 21005-5001		
1	PM TIMS, PROFILER (MMS-P) AN/TMQ-52 ATTN B GRIFFIES BUILDING 563 FT MONMOUTH NJ 07703		
1	US ARMY INFO SYS ENGRG CMND ATTN AMSEL IE TD A RIVERA FT HUACHUCA AZ 85613-5300		
1	COMMANDER US ARMY RDECOM ATTN AMSRD AMR W C MCCORKLE 5400 FOWLER RD REDSTONE ARSENAL AL 35898-5000		
1	US GOVERNMENT PRINT OFF DEPOSITORY RECEIVING SECTION ATTN MAIL STOP IDAD J TATE 732 NORTH CAPITOL ST NW WASHINGTON DC 20402		