CODING PERFORMANCE ON SATELLITE CHANNELS USING AX.25 PROTOCOL

by

William Augustus Jesser Jr.

209 Montvue Dr.

Charlottesville, VA 22901

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED:

T. Pratt

T. Pratt, Chairman

F. G. Gray

F. G. Gray

Brian D. Woerner

B. D. Woerner

May, 1993

Blacksburg, Virginia

Coding Performance on Satellite Channels

Using AX.25 Protocol

by

William Augustus Jesser Jr.

Committee Chairman: Timothy Pratt

Department of Electrical Engineering

(ABSTRACT)

A form of data transmission which is increasing in popularity is satellite communication. In order to insure that data is received correctly, certain error control strategies are employed. In packetized transmission, automatic-repeat-request (ARQ) schemes and error correcting codes have been employed. Currently a combination of both is not employed by the AX.25 protocol. The purpose of this project is to determine which error correcting code should be combined with ARQ to provide the optimum performance.

This project investigates the performance of the (7,4) Hamming code, (23,12) Golay code, Reed-Solomon, RS, codes over the Galois fields of 16, GF(16), and 256, GF(256), elements, and common rate convolutional codes of various constraint lengths. The codes are evaluated on three primary criteria, which include: throughput vs. input bit error rate, coding complexity, and burst error performance.

The class of Reed-Solomon codes over GF(256) was chosen to be superior due to the throughput and burst error

performance. The encoding complexity is small, but the decoding is more complicated. The conclusion is that error correcting codes should be employed in ARQ satellite systems. However, the error correcting strength of the code must be determined by observing the channel characteristics.

ACKNOWLEDGEMENTS

I would like to thank Dr. Timothy Pratt for taking his time to work with me on this thesis. Without his knowledge of satellites and packet radio, none of this would have been possible. I would also like to thank Dr. F.G. Gray for teaching and the Department of Electrical Engineering for offering a course in coding theory. Dr. Stephen Wilson was very helpful in tutoring me in coding over non-binary fields while I was attending the University of Virginia. He was also helpful in teaching some of the finer points of convolutional codes. Finally, I would like to thank my committee, Dr. F.G. Gray and Dr. Brian Woerner, for taking their time to read my thesis and schedule a defense.

TABLE OF CONTENTS

FIGURES

TABLES

CHAPTER 1

Introduction

Satellite communication is widely used for the transmission of data, television and voice. One of the main concerns of data transmission is that the information arrive exactly as it was sent. Many error control schemes have been invented to assure that communication is performed reliably. Since this form of communication is relatively new, some of these schemes are not optimized. The purpose of this project is to attempt to improve one of these error control schemes.

1.1 The AX.25 Protocol

Amateur radio enthusiasts have developed the AX.25 protocol for the transmission of data packets. This protocol was designed for terrestrial line of sight use. Terrestrial links have a low probability of bit error and are employed over short distances, usually less than 40 km. These facts led to the use of a go-back-N (ARQ) error control scheme.

1.2 The ARQ Error Control Scheme

In this scheme, redundant bits are sent with the information bits in order to detect any errors that may have occurred during transmission. If any errors are detected, the receiving station requests that the data be retransmitted.

The throughput of a system can be defined as the ratio of the number of information bits received correctly to the total number of bits transmitted. Note that the throughput can be affected by the addition of redundant check bits and the number of times data must be retransmitted.

The ARQ scheme is inefficient on noisy channels because a high percentage of packets will have errors. Each of these faulty packets must be retransmitted, and the throughput falls off quickly. If some of these errors could be corrected, through the use of coding, the throughput of the system could be increased dramatically. Figure 1.2.1 shows the throughput of a go-back-N ARQ system with and without coding. Note that there is over an order of magnitude improvement at the expense of maximum throughput. An ARQ scheme with error correction is referred to as a hybrid ARQ scheme.

1.3 ARQ Use in Satellite Communications

Because of the lack of standardized error control schemes, the AX.25 protocol has been adopted for use in some low earth orbit satellite (LEOSAT) systems. As the satellite descends toward the horizon, the probability of bit error rises. Consequently, the number of faulty packets increases to the point where the throughput of the system falls to zero. The transmission distance in a LEOSAT system is greater than 300 km and will develop delays between transmission and reception on the order of several ms. When one considers the high bit error rate and the long delay time, the ARQ scheme seems less appealing.

1.4 Types of Noise

In order to combat noise on the channel, the mechanisms which produce the noise must be understood. The most common type is thermal noise. Thermal noise is white Gaussian noise, found at a relatively constant background level. Occasionally small thermal spikes will change bits of data, which can be modeled as random occurrences at a certain rate. Thermal noise poses little problem for most error control codes.

One of the most difficult types of noise to combat is burst noise. This type of noise produces spikes that are both high in amplitude and long in duration, which can destroy a long string of bits. This type of noise is usually caused by sources such as lightning and man-made noise. Fading can also destroy a large number of consecutive bits. In order to combat burst noise, special techniques or codes must be used.

1.5 Project Goal

The goal of this project is to determine which type of error correcting code would be suitable for use within a packet radio system using the AX.25 system. There are two major types of error correcting codes, block and convolutional. Among the block codes to be tested, are the (7,4) Hamming code, the (23,12) Golay code, and several Reed-Solomon codes. Convolutional codes will also be tested with various rates and memory constraint lengths.

1.6 Evaluation Criteria

The criteria for evaluating each code must be specified. For each code, the output packet error rate vs. the input bit error rate will be found. This can be translated into throughput vs. input bit error rate, which is one of the criteria. Since the maximum throughput is the limited by rate of the code, this parameter is also a deciding factor. The simplicity of the encoding and decoding process will also figure heavily into the decision process. Another less heavily weighted criterion is the ability of the code to correct burst errors.

1.7 Format

This paper begins with an overview of the different ARQ schemes. The ARQ section describes the three different types of ARQ, how they are used, and their performance trade-offs. Chapter 3 gives an overview of error control coding, including both block and convolutional codes. Chapter 4 includes an overview of the AX.25 protocol, frame structure, and procedures. Chapters 5-8 give a description of the encoder and decoder and performance for each error correction codes. In Chapter 9, the codes are evaluated according to the criteria mentioned above. Recommendations are made as to which code should be employed for satellite systems transmitting packetized data, and future research is discussed. Chapter 10 will include recommendations on the implementation of a packet system with error control coding. Chapter 11 will include a summary of the results and recommendations for further research.

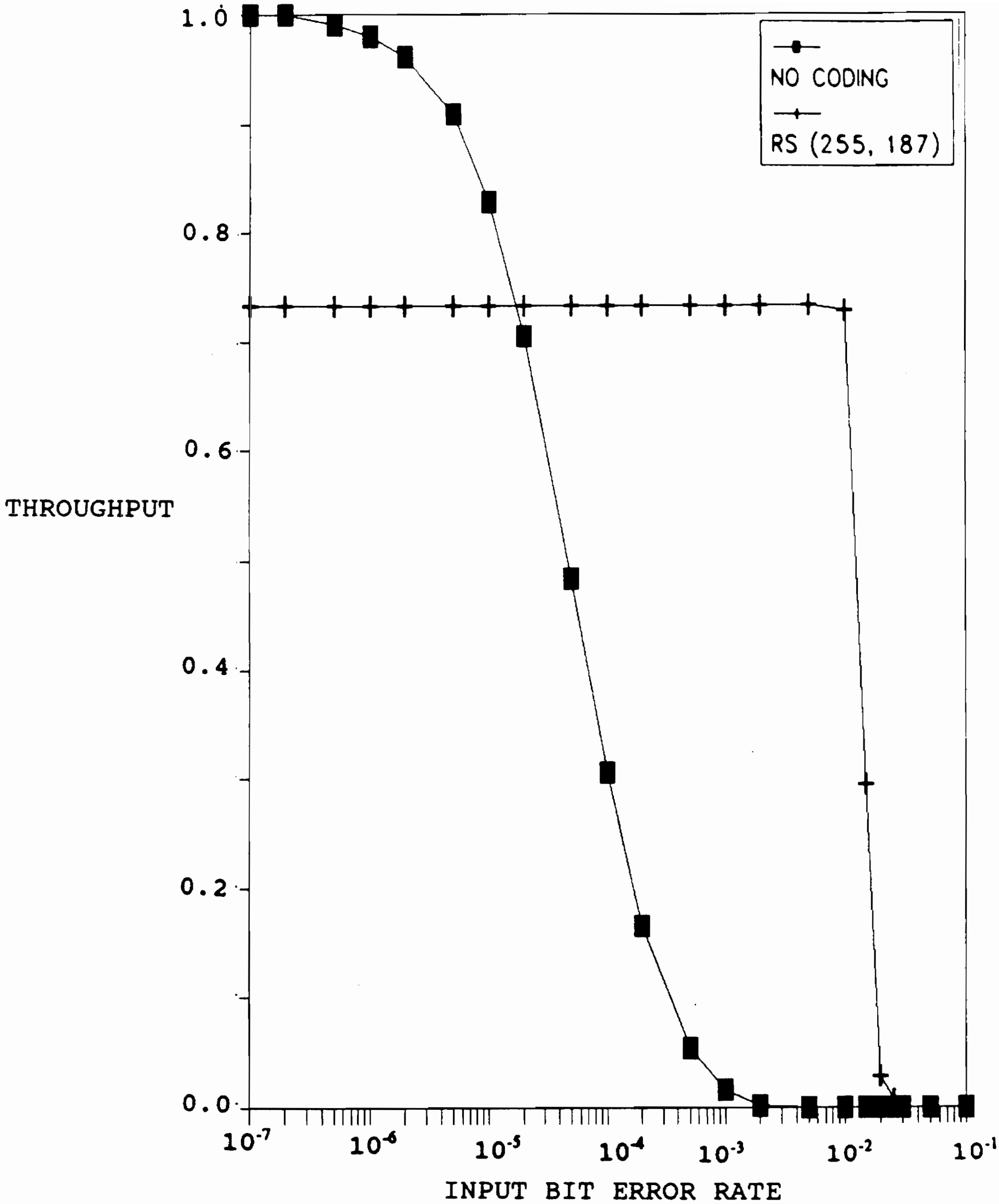


Figure 1.2.1 Throughput of a go-back-N, with N=10, automatic-repeat-request system using a (255,187) Reed-Solomon error correcting code and using no coding.

CHAPTER 2

Overview of Automatic-Repeat-Request Schemes

The primary concern in data communication is the transmission of error-free information. Channel noise often causes errors in the transmitted data. To combat this problem, an error control scheme is often implemented. The most commonly used method is an automatic-repeat-request (ARQ) scheme. The data are divided into information packets, which are encoded with an error detecting code. This code is used to detect any errors which may have occurred during transmission. If errors are detected, the receiving station signals the transmitting station to retransmit the faulty packet. By choosing an appropriate error detecting code, the probability that an error passes undetected can be made very small.

The AX.25 amateur radio system uses go-back-N ARQ. This chapter gives background for the system, and equations are given for channel throughput. The stop-and-wait and the selective-repeat types are included for completeness and to facilitate a better understanding of ARQ systems.

2.1. ARQ Schemes

2.1.1 Stop-and-Wait ARQ Scheme

Three basic ARQ schemes are commonly employed, including stop-and-wait, go-back-N and selective-repeat ARQ schemes. The simplest is the stop-and-wait scheme. The transmitting station sends the packet and awaits a positive acknowledgement (ACK) from the receiving station. If an ACK is received, the transmitter continues to send packets and wait for the proper acknowledgement. If errors are detected in the packet, the receiving station replies with a negative acknowledgement (NAK). The transmitting station retransmits the faulty packet until an ACK is received. This stop-and-wait procedure continues until all the packets in the queue have been transmitted. If the channel has long round-trip delays, such as on a satellite link, the throughput of this scheme falls off rapidly as more errors are detected. This disadvantage led to continuous transmission schemes such as the go-back-N and selective-repeat variations.

2.1.2 Go-Back-N ARQ Scheme

In the go-back-N scheme, packets are transmitted continuously and stored in a buffer until the appropriate ACK or NAK is received, illustrated in Figure 2.1.2.1. During the delay between transmission of the packet and receipt of the

ACK, N-1 packets are sent. If no ACK is received, the transmitting station begins retransmitting with the faulty packet and also retransmits the subsequent N-1 packets, which are rejected when a NAK is received. This total of N packets is retransmitted until the first packet is positively acknowledged. Only then will transmission of new packets resume. High data rates and long round-trip delays require N to be a large number. Consequently, the number of discarded packets will be quite large. In this case, the go-back-N ARQ scheme also becomes inefficient for high error rates. Therefore, the selective-repeat strategy was developed.

2.1.3 Selective-Repeat ARQ Scheme

In the selective-repeat scheme, packets are transmitted continuously, but only those for which a NAK is received are retransmitted. See Figure 2.1.3.1. Since only the faulty packets are discarded, this scheme represents the most efficient of the three and is employed most frequently in modern systems.

2.2 Throughput Efficiency

The throughput of an ARQ scheme can be defined as the ratio of the average number of information bits received correctly to the total number of bits transmitted. In the following equations[1], it is assumed that the return channel causes no errors. The throughput is based on the following parameters:

P_p = the probability that a received packet contains errors or the packet error rate

n = the number of bits per packet

k = the number of information bits per packet

r = the data rate of the transmitter

d = the delay between consecutive transmissions

N = the number of packets that must be retransmitted.

2.2.1 Throughput of the Stop-and-Wait Scheme

The average number of bits that are transmitted is given by

$$M_{SW} = (n+dr)/(1-P_p) \qquad (2.2.1.1)$$

The throughput is given by

$$T_{SW} = k/M_{SW} = (1-P_p)k/(n+dr) \qquad (2.2.1.2)$$

2.2.2 Throughput of the Go-Back-N Scheme

The average number of transmissions for a codeword to be received successfully is given by

$$M_{GBN} = 1 + NP_p/(1-P_p) \qquad (2.2.2.1)$$

The throughput is given by

$$T_{GBN} = k/nM_{GBN} = (1-P_p)k/n((1-P_p)+P_pN) \qquad (2.2.2.2)$$

2.2.3 Throughput of the Selective-Repeat Scheme

The average number of transmissions for a codeword to be received successfully is given by

$$M_{SR} = 1/(1-P_p) \qquad (2.2.3.1)$$

The throughput is given by

$$T_{SR} = k/M_{SR}n = (1-P_p)k/n \qquad (2.2.3.2)$$

These equations assume that the selective-repeat buffer is infinite.

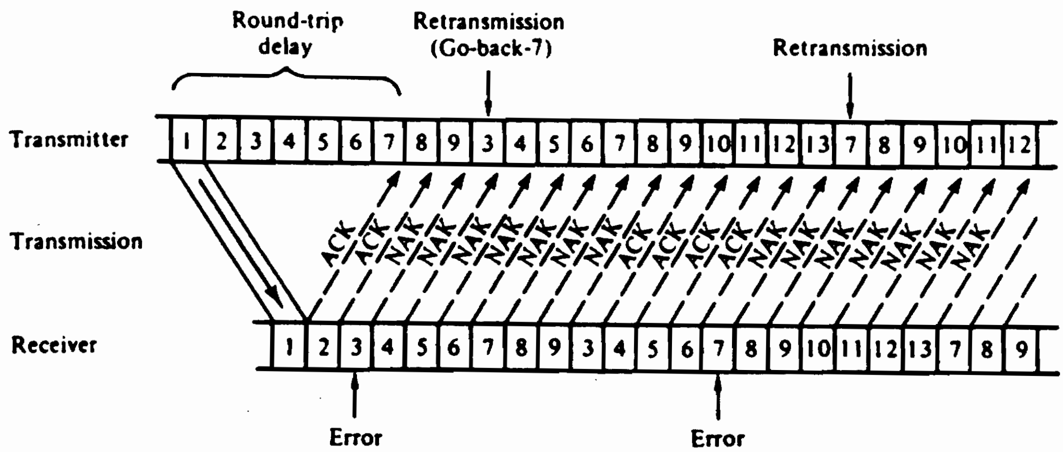


Figure 2.1.2.1 The go-back-N ARQ scheme with N=7. Note that 6 correct packets are retransmitted with the faulty packet. From Lin, Costello and Miller, "Automatic-Repeat-Request Error-Control Schemes," *IEEE Comm. Mag*. March, 1984.

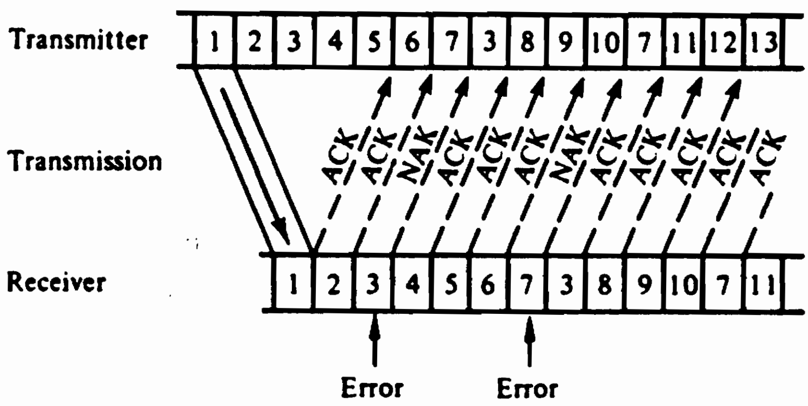


Figure 2.1.3.1 The selective-repeat ARQ system. Note that only the faulty packet is retransmitted. From Lin, Costello and Miller, "Automatic-Repeat -Request Error-Control Schemes," *IEEE Comm. Mag*. March, 1984.

CHAPTER 3

Overview of Error Control Coding

Error control coding is a powerful technique in digital communications. Through the addition of redundant data, errors that occur in the channel can be corrected. The channel encoder adds redundant bits to the information bits. The decoder uses these bits to detect or correct errors that may have been caused by the channel. There are two main classes of error control codes: block codes and convolutional codes. Each of these will be examined in this paper.

Through the use of error control coding, transmission of packetized data can be made more efficient. A coded system will correct some of the errors caused by the channel. Since these errors will be corrected, the system will not be required to retransmit as many faulty packets, and the throughput will increase.

Burst errors cause severe problems on some channels. Bursts are usually caused by highly correlated noise with a duration lasting more than that of a single bit, causing a sequence of bits to be corrupted. A burst is defined as the number of bits between 2 nonzero bits in the error pattern.[2] All of the bits in the burst may be in error or some of them may be in error. A code for packet radio should be able to correct burst errors.

3.1. Block Codes

A diagram of a block code system is shown in Figure 3.1.1. Important parameters of block codes are given below.

n = the number of bits in a block.

k = the number of information bits.

n-k = the number of check bits.

d_{min} = the minimum distance of the code.

t = $(d_{min}-1)/2$ = the maximum number of correctable errors.

d = $d_{min}-1$ = the maximum number of detectable errors.

Groups of k binary information bits are entered into the encoder, which can be represented as a vector $\mathbf{u}=(u_0,u_1,\ldots,u_{k-1})$. The encoder transforms these k bits into n bits, n>k, which can be represented by the n-tuple $\mathbf{v}=(u_0,u_1,\ldots,u_{n-1})$. This code is comprised of 2^k codewords, out of 2^n possible binary combinations, and is described as an (n,k) code. The code rate is given by R=k/n, which is the ratio of the number of information bits to the number of total bits. The received n-tuple **r** is the original codeword

u plus the error pattern **e**. This error pattern will be non-zero in all places where an error has occurred and zero elsewhere. An error control code can correct a maximum of t errors and detect a maximum of d errors. If the received n-tuple is one of the original codewords, it is transformed back into a k-tuple **y** and passed to the user. If **r** is not one of the original codewords, the most likely codeword is estimated given the received vector, transformed to a k-tuple, and sent to the user. Block codes are good at correcting burst errors. A block code can correct any burst of t bits and detect any burst of k bits.[3]

Some special error control codes have desirable properties. The structure of codes is such that some codes can correct certain received vectors even though more than t errors are present. However, a code is guaranteed to correct all pattern that contain t errors. Codes which correct exactly t errors are called perfect codes. The probability of codeword error can be found exactly by:[4]

$$P_c = \sum_{i=t+1}^{n} [n!/(n-i)!*i!]*p^i(1-p)^{n-i} \qquad (3.1.1)$$

The Hamming codes and the Golay codes are important examples of these perfect codes.

A code which contains the k information bits somewhere in the block is called a systematic code. This property is desirable because to retrieve the information the n-k check bits are simply deleted. If each bit in a code is cyclically shifted one place, and a valid codeword results, this code is said to be cyclic. This property allows simplification in the encoding circuitry.

A code which has the largest possible value of d_{min} compared to any other code with the same n and k, is called a maximum distance code. No known non-trivial binary code is a maximum distance code. However, the non-binary Reed-Solomon class of codes is a maximum distance code. This implies that a given Reed-Solomon code will always have superior performance over other binary codes, such as a binary BCH code.

As mentioned previously, only 2^k acceptable codewords exist out of 2^n possible binary combinations. The fact that, in a cyclic code, these codewords can be formed by shifting a previous codeword and all linear combinations of these shifts is a highly desirable property. The encoding and decoding process can be greatly simplified if a code has this property.

3.2 Convolutional Codes

The encoder of a convolutional code also takes k bits and transforms them into an n-bit sequence. However, k and n are small integers usually less than 10. The output sequence depends on the previous L+1 blocks, where L is the constraint length. A convolutional code is described as an (n,k,L) code. Figure 3.2.1 gives an example of a convolutional encoder with k=1, n=2 and L=2. For each bit entered into the encoder, two will be output. The output will depend on the previous two input bits. A popular method of decoding finds the most likely error pattern and decodes accordingly.[5] This class of codes performs poorly for burst error correction.

3.3 Interleaving

An effective method of combatting burst errors is interleaving. If a burst B bits in length occurred during one codeword, up to B bits of the codeword would be in error. One would need a very powerful error control code to cope with the errors. Through the use of interleaving, the errors can be distributed throughout many codewords. Interleaving is best explained using an ixn matrix, shown in Figure 3.3.1, where i is the degree of interleaving. The blocks from the encoder are written into the matrix by rows and read into the channel by columns. If burst errors occur, a sequence of B consecutive bits will be destroyed. Upon reception, the data is written back into an nxi matrix by rows and read into the decoder by columns. The series of

B errors is now distributed over approximately i codewords, with the number of errors per codeword being approximately B/i. The error control code has a much better chance of correcting a small number of errors distributed over many codewords, than a large number in one codeword. In Figure 3.3.1b an error 23 bits in length will cause single bit errors in 17 codewords and double bit errors in 3 codewords. If interleaving were not employed, an entire codeword would be corrupted, as well as three bits of another codeword.

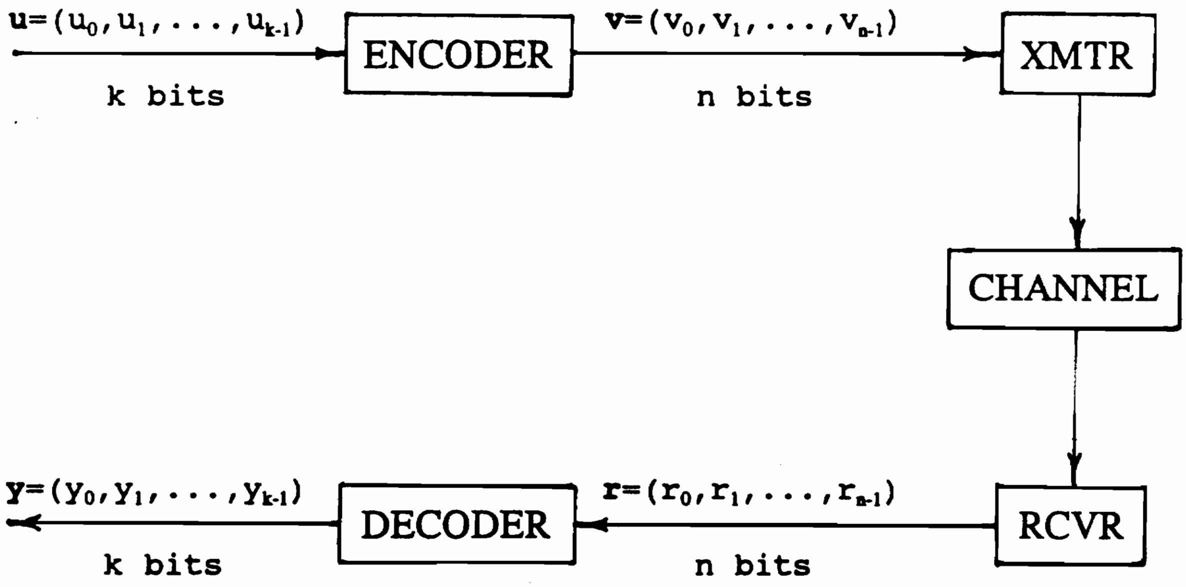


Figure 3.1.1 Block diagram of a typical block coded system.

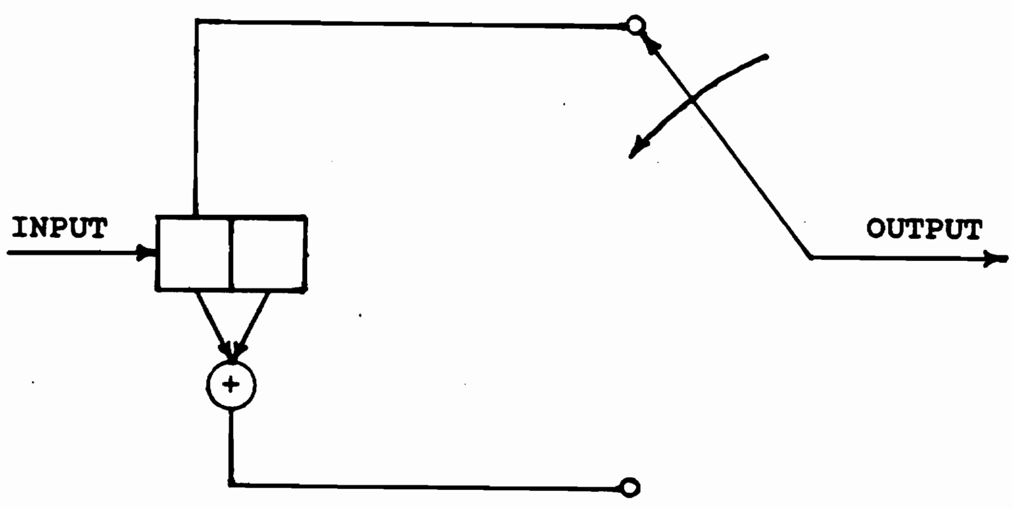
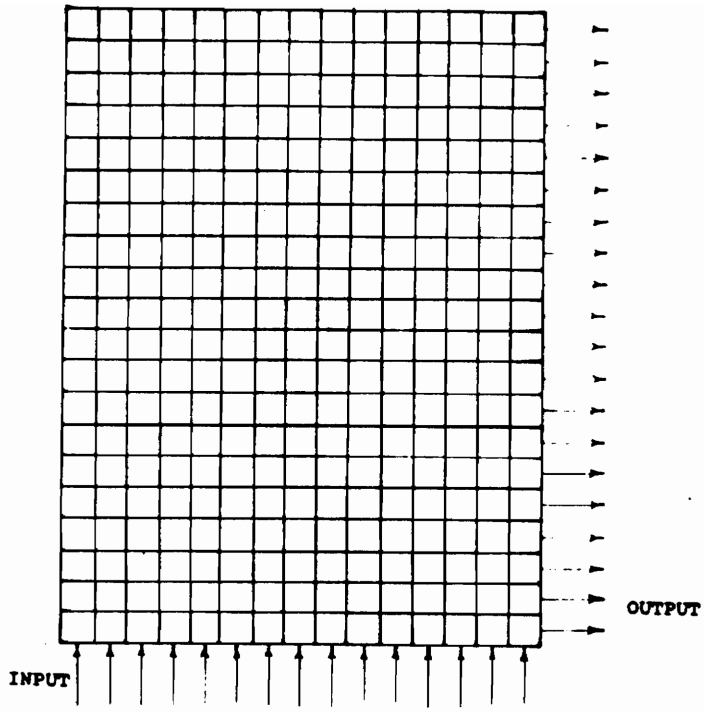
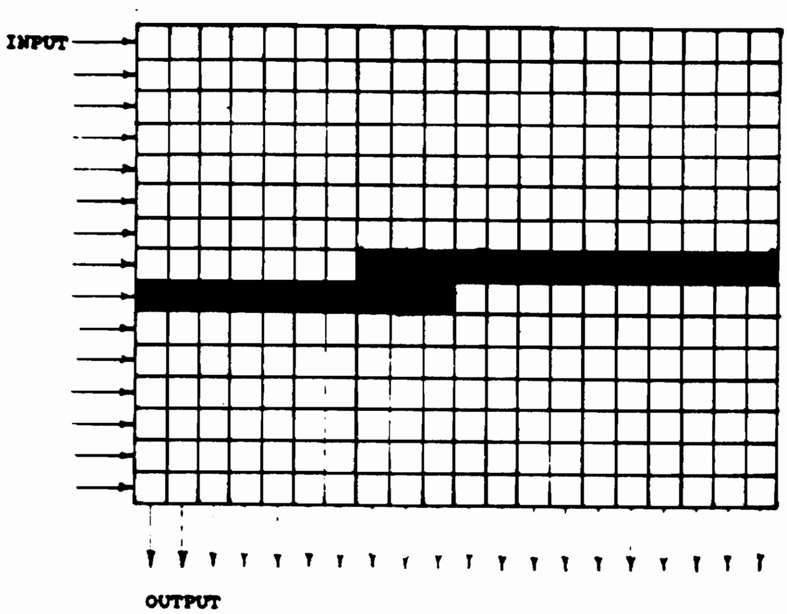


Figure 3.2.1 Encoder for a (2,1,2) convolutional code.



1a



1b

Figure 3.3.1 Decoding interleaved codes can be visualized as using an nxi matrix.
(a). The received data is read into the matrix in rows and read out by columns. If a burst error occurs, it is distributed over many codewords (b).

CHAPTER 4

Overview of the AX.25 Amateur Packet Radio Protocol

Packet radio has been used by amateur radio enthusiasts for years. Files of information are transmitted from station to station until they reach their specified destination. These files are then stored by the receiver. In this fashion, users can communicate or send data over long distances. In order to standardize the format, the AX.25 protocol has been established. This protocol specifies the frame structure, procedural structure, and operation.[6]

The AX.25 protocol uses an ARQ system with no error correction. With the addition of error correction, the throughput at high bit error rates should increase dramatically.

4.1 Frame Structure

Information and control data in a packet is divided into blocks called frames. These frames have substructures called fields. A flag field is found at the beginning and end of each frame. Within the frame are found the address, control, protocol identifier, information, and frame check sequence fields. The two different types of frames are shown in Figure 4.1.1. The frame in 4.1.1a is used for supervisory and unnumbered frames. The supervisory frames are used for acknowledging or requesting retransmission. The unnumbered frames are used for establishing or terminating the link or to maintain the link when it is idle. The frame in 4.1.1b contains information to be transmitted.

4.1.1 Flag Field

The flag field is found at the beginning and end of each frame, and is always 01111110. Consecutive frames may share a flag. This sequence is disallowed anywhere else through the use of bit stuffing. Since this sequence must be detected in order for reception to begin, it cannot be encoded with the rest of the packet.

4.1.2 Address Field

This field is used to identify the source, destination, and, if a repeater is used, repeater address. The length varies from 112, for just a source and destination, to 560, for a source, destination and eight repeaters. Figure 4.1.2.1 shows an example of non-repeater and repeater fields.

FLAG	ADDRESS	CONTROL	FCS
01111110	112 BITS	8 BITS	16 BITS

1a Structure of supervisory and unnumbered frames.

FLAG	ADDRESS	CONTROL	PID	INFO	FCS
01111110	112 BITS	8 BITS	8 BITS	N*8 BITS	16 BITS

1b Structure of information frames.

Figure 4.1.1 The structure of the two different types of frames in the AX.25 protocol.

Octet	ASCII	Binary Data
A1	K	10010110
A2	8	01110000
A3	I	10010010
A4	M	10011010
A5	F	10001100
A6	space	01000000
A7	SSID	XXXXXXXX
A8	W	10101110
A9	K	10010110
A10	4	01101000
A11	J	10010100
A12	M	10010110
A13	O	10011110
A14	SSID	XXXXXXXX

1a Address field with no repeaters.

Octet	ASCII	Binary Data
A1	K	10010110
A2	8	01110000
A3	I	10010010
A4	M	10011010
A5	F	10001100
A6	space	01000000
A7	SSID	XXXXXXXX
A8	W	10101110
A9	K	10010110
A10	4	01101000
A11	J	10010100
A12	M	10010110
A13	O	10011110
A14	SSID	XXXXXXXX
A15	K	10010110
A16	4	01101000
A17	B	10000100
A18	F	10001100
A19	O	10011110
A20	space	01000000
A21	SSID	XXXXXXXX

1b Address field with 1 repeater.

Figure 4.1.2.1 Examples of address fields without (a) and with (b) repeaters.

In the field without repeaters, the destination and source amateur call signs are found in octets A1-A6 and A8-A13 respectively. The call signs are encoded as upper-case ASCII characters, except that the last bit is always 0. The Secondary Station Identifier (SSID), octets A7 and A14, is given in case more than one station is operating under the same call sign.

The field with repeaters is encoded in a similar fashion as the field without repeaters. The destination and source addresses are followed by the repeater addresses, where the first repeater listed is the first in the chain. Up to eight repeaters can be used under this protocol. Note that for use in a satellite only communications system, no repeaters will be used.

4.1.3 Control Field

The control field is 8 bits in length. This field is used to identify the frame type and convey commands and responses through the link.

4.1.4 Protocol Identifier Field

These 8 bits identify the type of protocol used on a different level of the system.

4.1.5 Information Field

The data in this field are the actual information to be transmitted from station to station. This field contains an integral number of octets, up to 256.

4.1.6 Frame Check Sequence

This 16 bit sequence is the redundant bits of an error control code used to detect any errors that may have occurred during transmission.

4.2 AX.25 Procedures

The AX.25 protocol is an automatic repeat request (ARQ) system. This system is a modified go-back-N type of ARQ.

To begin the interaction process, the transmitter sends a request for connection to the receiver. The receiver will reply with a receive ready (RR) or a receive not ready (RNR). After the RR response is received, the link is established, and the transmitter may begin sending data.

When the first information (I) frame is sent, the T1 timer is started. The T1 timer is set to the transmission time of 3 packets plus the round-trip delay time. The timer is reset at the transmission of each subsequent I frame. Up to seven I frames can be outstanding at any time. After each frame is sent, the receiver will attempt to detect any errors. If no errors are detected, the receiving station will acknowledge with a command, usually an RR. If any errors are detected, the receiving station will not acknowledge the packets, and the T1 timer will expire. When the T1 timer expires the transmitting station must transmit a command, usually RR, and wait for the appropriate response from the receiving station. If no response is received, the station will continue transmitting the RR command until the maximum number of tries is exceeded or the station acknowledges. If the correct response is received, the station may resume transmitting with the last unacknowledged I packet.

In order to calculate the throughput of the system, the effective number of retransmitted packets, N, must be calculated. These calculations are based on packets of 2040 bits in length. If one assumes that an infinite number of frames are transmitted, the transmitter will continue to transmit the maximum number of frames that can be outstanding at one time, which is 7 or 14280 bytes. Time must also be allowed for the T1 timer to expire, which is the amount of time for three packets, 6120 bits, to be sent plus the round-trip delay. The round-trip delay time is approximately 3 ms, which at 9600 baud is only 30 bits. Time must also be allowed to transmit an RR command and receive the proper response. These frames are only 144 bits long, and 30 bits are added for the delay time. The number of bits which need to be retransmitted is approximately 20,700 or about 10 packets. Therefore, to be modeled as a go-back-N system, N is chosen to be 10 on the average. Some recommendations have been to make the T1 timer delay 6 seconds for satellite communications, in which case N would be made about 31.[7] In such a case, throughput would fall off very rapidly as the bit error rate increased.

When all transactions have been completed, the transmitting station will send a disconnect (DISC) command, and the link will be terminated. If at any time a response to a command is not received, the station will repeat the command until either a response is received or the maximum number of retries is exceeded.

CHAPTER 5

The (7,4) Hamming Code

The (7,4) Hamming code is one of the simplest linear block codes.[8] All Hamming codes have a $d_{min}=3$ and $t=1$, which means that they can correct any single bit error. Because the code is short, it can be encoded by simple table look-up. The code is also a perfect code, and the probability of error can be found through a simple expression. Any coding book will have a section on the (7,4) Hamming code. It was one of the first codes found and is often used to introduce the subject of error control coding. There are many excellent texts on coding which thoroughly cover block codes.[9-13]

5.1 Encoding

To simplify the decoding process, the data is encoded systematically. Normally, the 4 information bits are represented by a 4-tuple and multiplied by a generator matrix to produce the codewords. However, since this code is short, only 16 different codewords exist. The encoding can therefore be implemented by table look-up. The information bits and codewords are shown in Table 5.1.1a and the generator matrix in 5.1.1b.[14]

5.2 Performance

Since this is a perfect code the codeword error rate can be computed exactly by:

$$P_c=\sum_{i=t+1}^{n} n!/((n-i)!*i!)*p^i*(1-p)^{n-i} \qquad (5.2.1)$$

where, n=block length

t=error correction capacity

p=bit transition probability.

P_c=probability of codeword error

In order to properly compare the different codes, they are normalized to a packet length of approximately 2040 bits. If 292 7-bit codewords are used, a length of 2044 bits will result. This implies that the value for q in equation 5.2.2 is 292. The following equation will give the packet error probability:

$$P_p=1-(1-P_c)^q \qquad (5.2.2)$$

where, q=the number of blocks in the packet.

This probability of output packet error rate vs. input bit error rate is shown in Figure 5.2.1. The equation for throughput of the go-back-N ARQ is given by equation 2.2.3.2:

$$T_{GBN}=k(1-P_p)k/n((1-P_p)+P_pN) \qquad (2.2.2.2)$$

A plot of throughput vs. bit error rate for go-back-N, with N=7, is shown in Figure 5.2.2. With the interleaving of 292 codewords within a packet, this code can always correct a burst of length 292 bits in a single packet.

Table 5.1a The data can be encoded using table look-up. The encoder table is shown here.

DATA	CODE WORD
0000	0000000
0001	0001011
0010	0010110
0011	0011101
0100	0100111
0101	0101100
0110	0110001
0111	0111010
1000	1000101
1001	1001110
1010	1010011
1011	1011000
1100	1100010
1101	1101001
1110	1110100
1111	1111111

Table 5.1b The generator matrix of the systematic (7,4) Hamming code.

$$G = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{matrix}$$

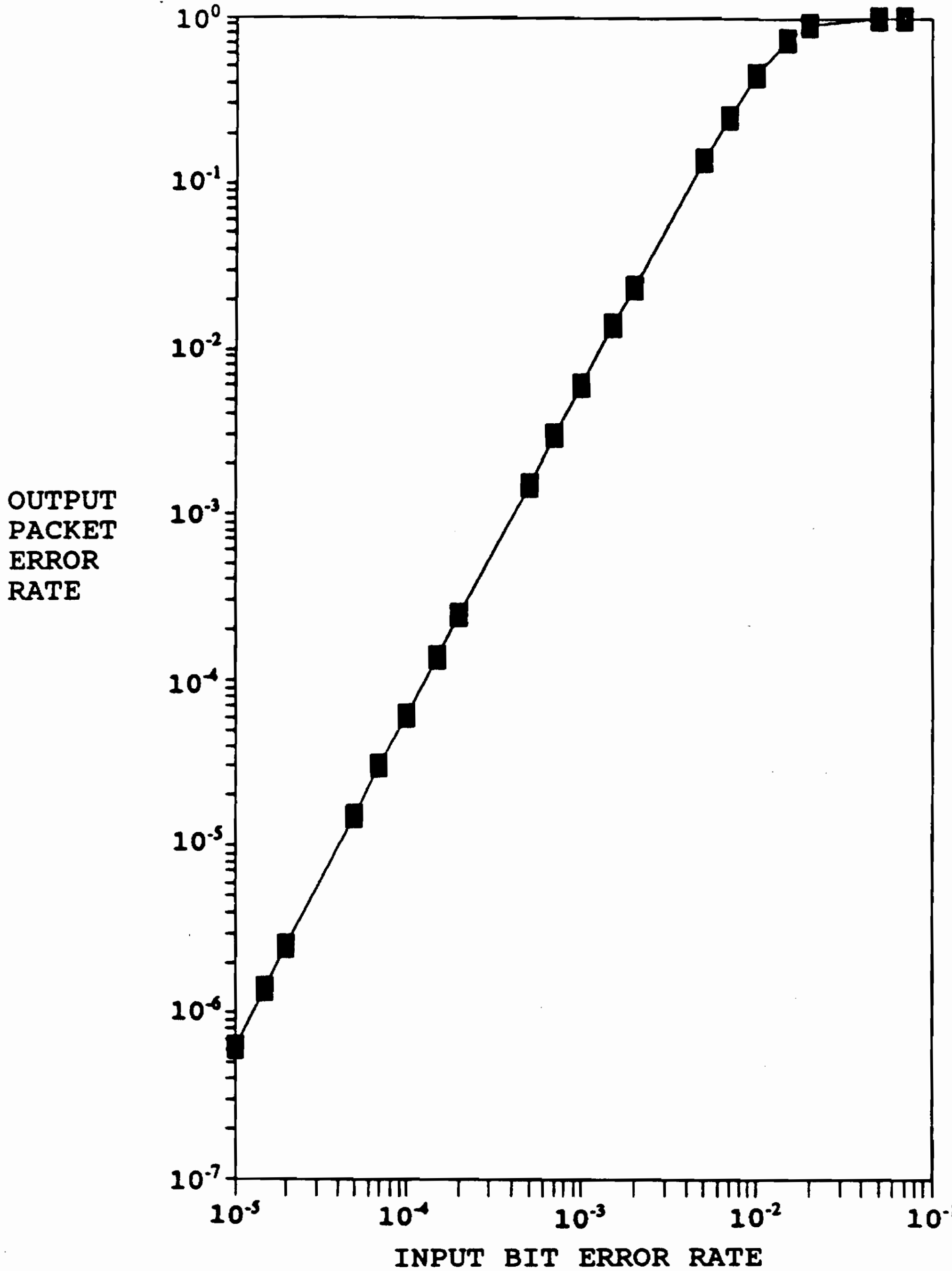


Figure 5.2.1 Output packet error rate (2044 bit packet) vs. input bit error rate for the (7,4) Hamming code.

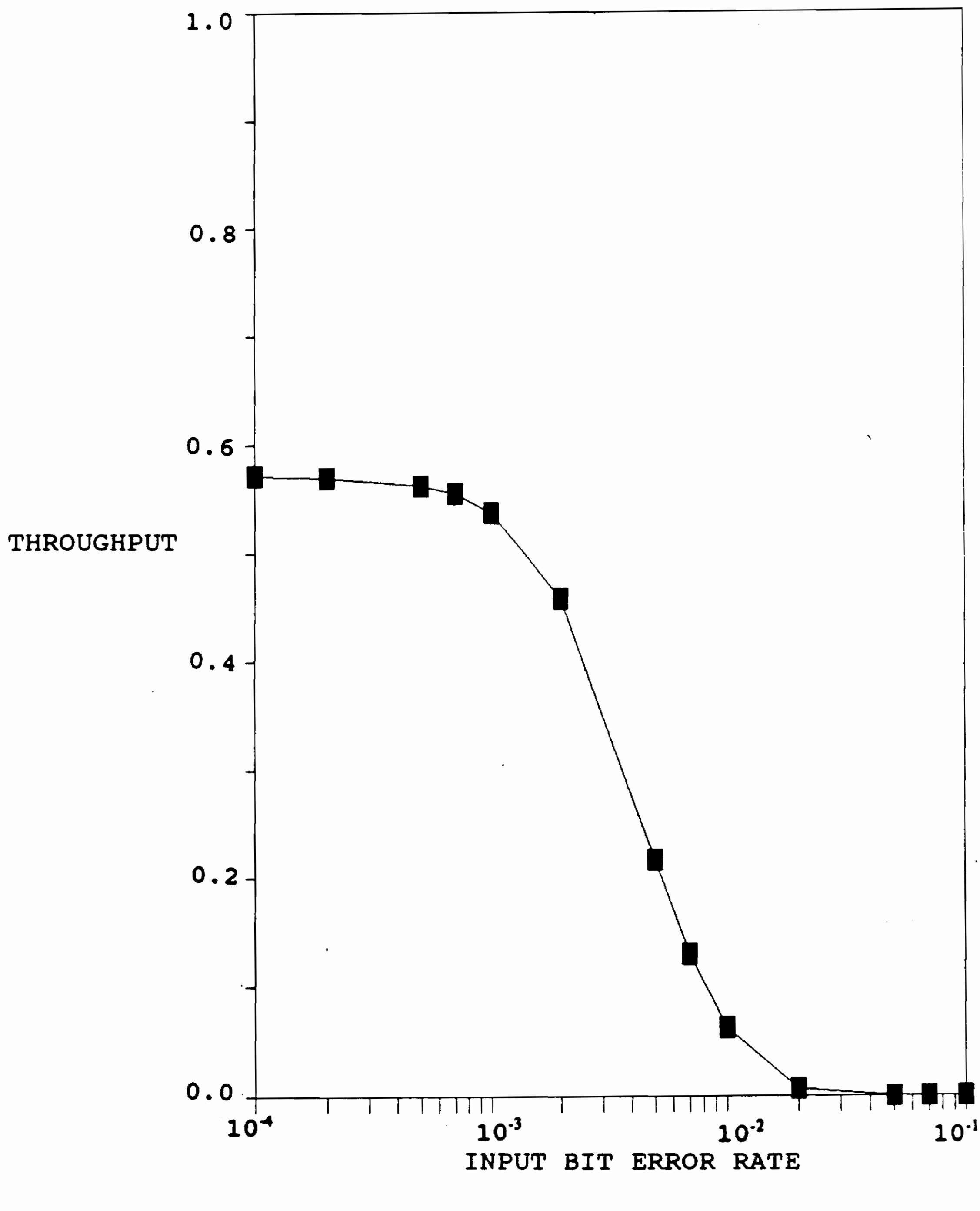


Figure 5.2.2 Throughput vs. input bit error rate for the (7,4) Hamming code.

5.3 Decoding

Decoding of the (7,4) Hamming code is accomplished using syndrome decoding. The syndrome is a 3-tuple, $\mathbf{s}=(s_0,s_1,s_2)$.

The equations to compute the coefficients of **s** are:

$$s_0=r_0+r_1+r_2+r_4$$
$$s_1=r_1+r_2+r_3+r_5 \qquad (5.3.1)$$
$$s_2=r_0+r_1+r_3+r_6$$

The computed syndrome will give the most probable error pattern. The pairing of syndromes and error patterns are given in Table 5.3.1. If the syndrome is (0,0,0), no error was detected, and the information can be recovered by deleting the last 3 bits of each code word. If the syndrome is non-zero, the corresponding error pattern is added to the received codeword. If this new code vector is one of the original codewords, the information can be recovered as mentioned above. This decoder has very little complexity and is simple to implement.

Table 5.3.1 Syndromes and the corresponding maximum likelihood error pattern.

Syndrome	Error Pattern
000	0000000
001	0000001
010	0000010
100	0000100
011	0001000
110	0010000
111	0100000
101	1000000

CHAPTER 6

The (23,12) Golay Code

The (23,12) Golay code is another one of the few perfect linear block codes.[15] This code has $d_{min}=7$ and $t=3$. This code can be encoded with a linear feedback shift register, and can be decoded in a similar fashion. In order to form a rate 1/2 code, this code is often extended 1 bit to (24,12) and $d_{min}=8$. Most texts have limited information on the Golay code[16][17]. The latter reference is more mathematically rigorous than the previous.

6.1 Encoding

The cyclic Golay code can be encoded with a simple linear feedback shift register. Figure 6.1.1 shows a generalized circuit for encoding. The g_i's are the coefficients of the generator polynomial, which in this case is $g(x)=1 + x + x^5 + x^6 + x^7 + x^9 + x^{11}$.[18]

6.2 Performance

This code is a perfect code, and therefore the probability of codeword error is given by equation 5.2.1.

$$P_c=\sum_{i=t+1}^{n} n!/((n-i)!*i!)*p^i*(1-p)^{n-i} \qquad (5.2.1)$$

where, n=23 and t=3.

This rate is then normalized to 2047 bits using equation 5.2.2 with q=89.

$$P_p=1-(1-P_c)^q \qquad (5.2.2)$$

The probability of output packet error rate vs. input bit error rate is plotted in Figure 6.2.1. The throughput is given by equation 2.2.2.2.

$$T_{GBN}=(1-P_p)k/n((1-P_p)+P_pN) \qquad (2.2.2.2)$$

The throughput vs. bit error rate is given in Figure 6.2.2. The rate, k/n, of this code is 0.52. Without the use of interleaving, this code can always correct bursts of 3 errors. However, when interleaving is employed, with a degree of 89, up to 267 errors can always be corrected.

6.3 Decoding

Decoding can be accomplished by the Kasami error-trapping technique.[19] The circuit diagram of the decoder can be found in Figure 6.3.1. The decoding process is somewhat complex. For a complete description of the decoding process, see section 5.3 of Lin and Costello's *Error Control Coding: Fundamentals and Applications*.

In order to give a sense of the complexity, some of the steps are described. The received code vector is read into the buffer and the syndrome register. When the error pattern is found in the syndrome registers, certain gates are opened or closed, and the error pattern is shifted out of the syndrome register. In order to determine if the error pattern is in the syndrome register, a complex series of tests are performed, T_0-T_2.

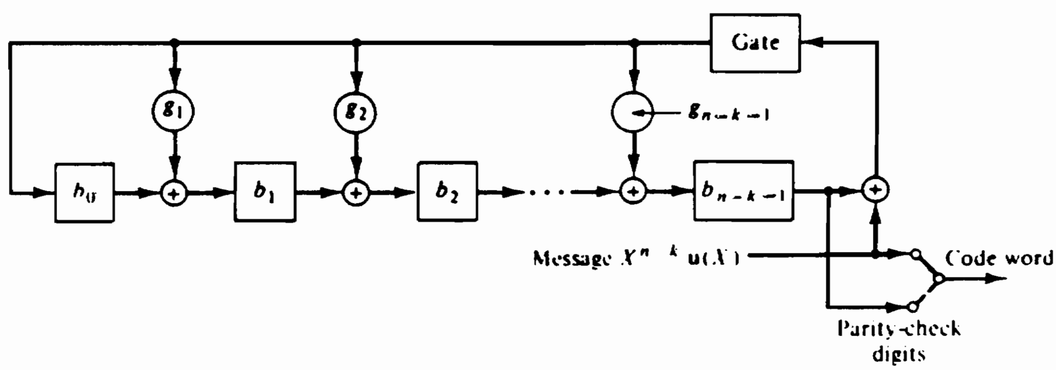


Figure 6.1.1 Encoding circuit for the (23,12) cyclic Golay code. The g_i's are the generator polynomial coefficients, and the b_i's are the parity check bits. From Lin and Costello *Error Control Coding: Fundamentals and Applications,* 1983, p. 95.

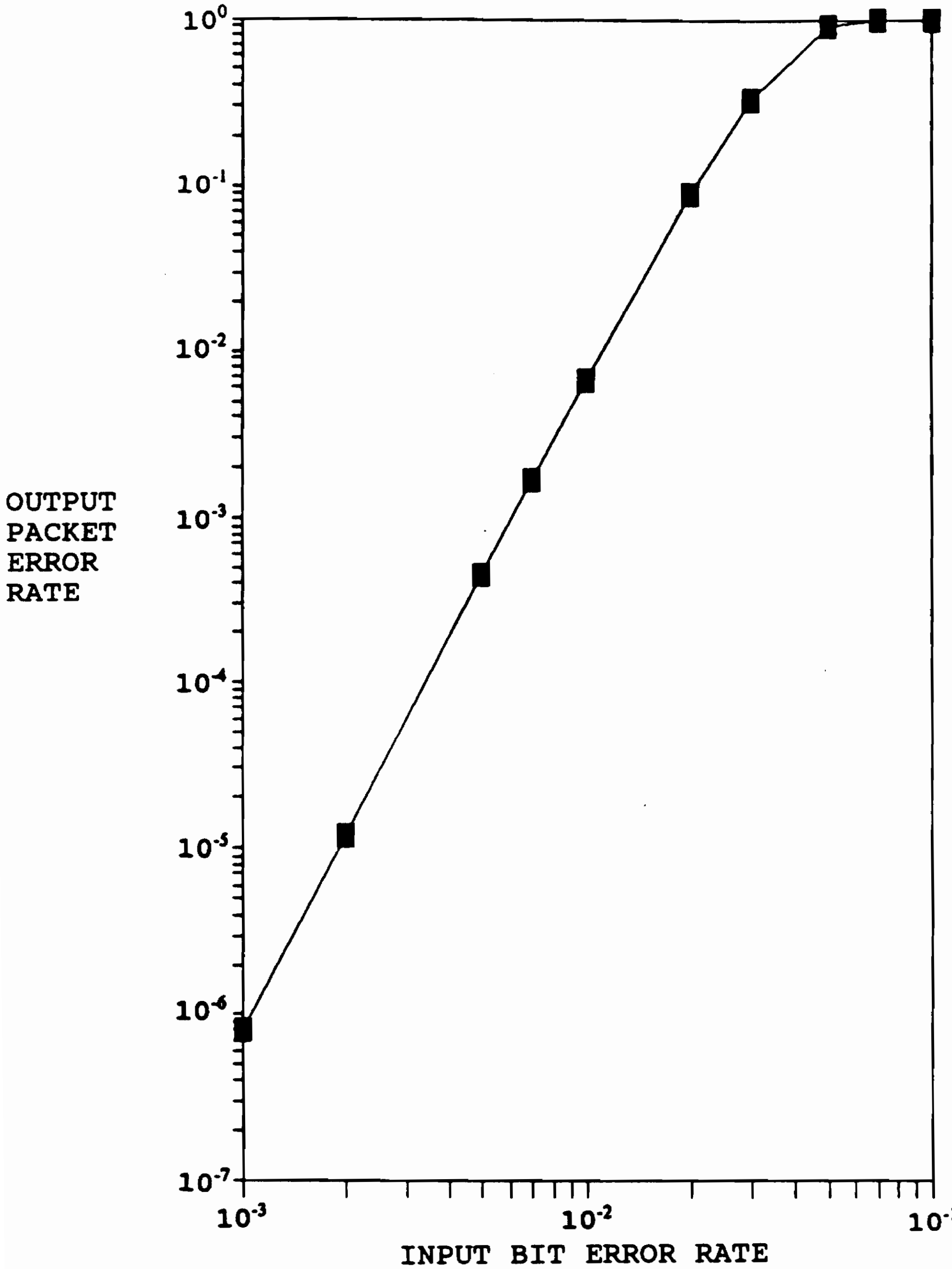


Figure 6.2.1 Output packet error rate vs. input bit error rate for the (23,12) Golay code.

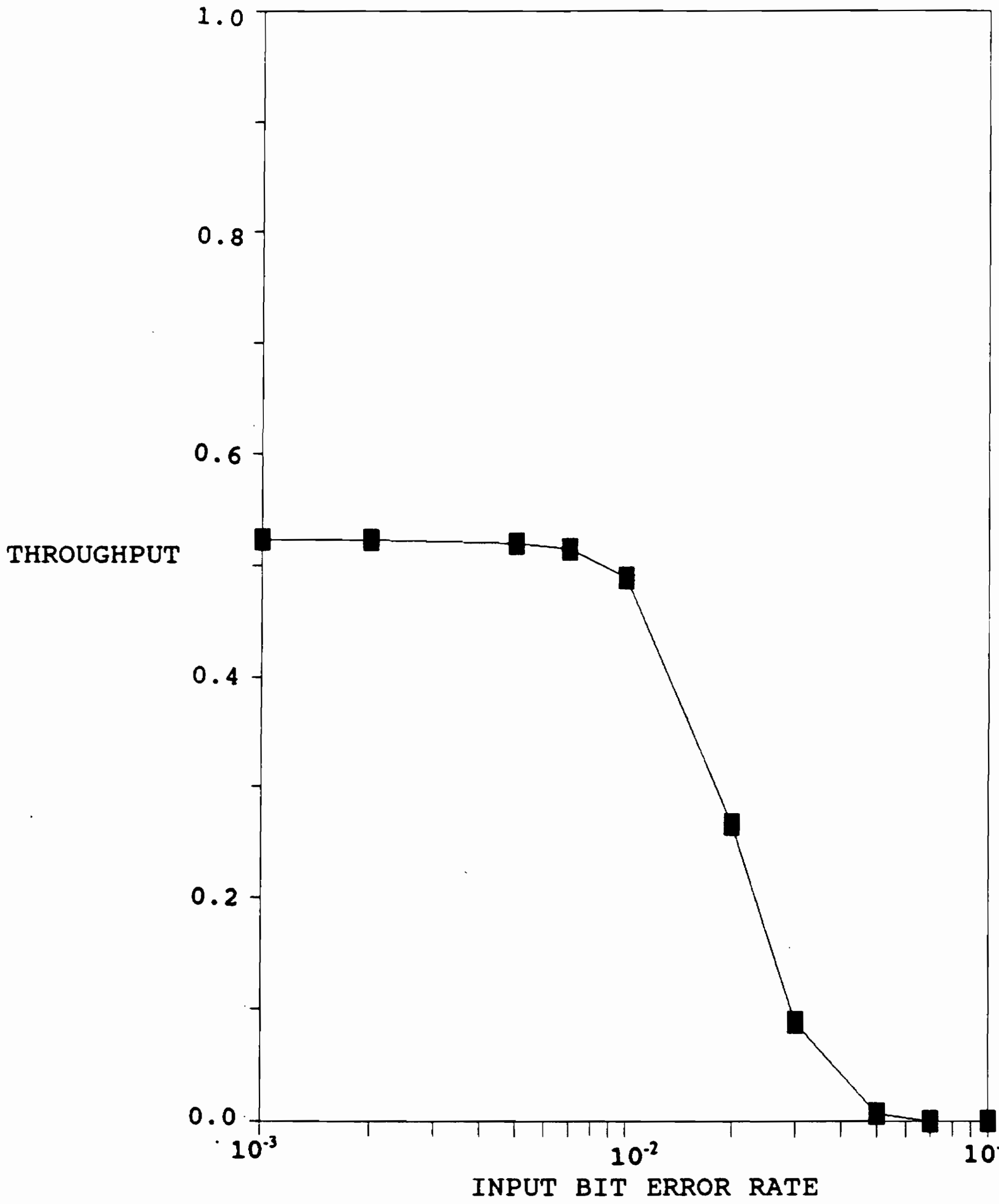


Figure 6.2.2 Throughput vs. input bit error rate for the (23,12) Golay code using go-back-10 ARQ.

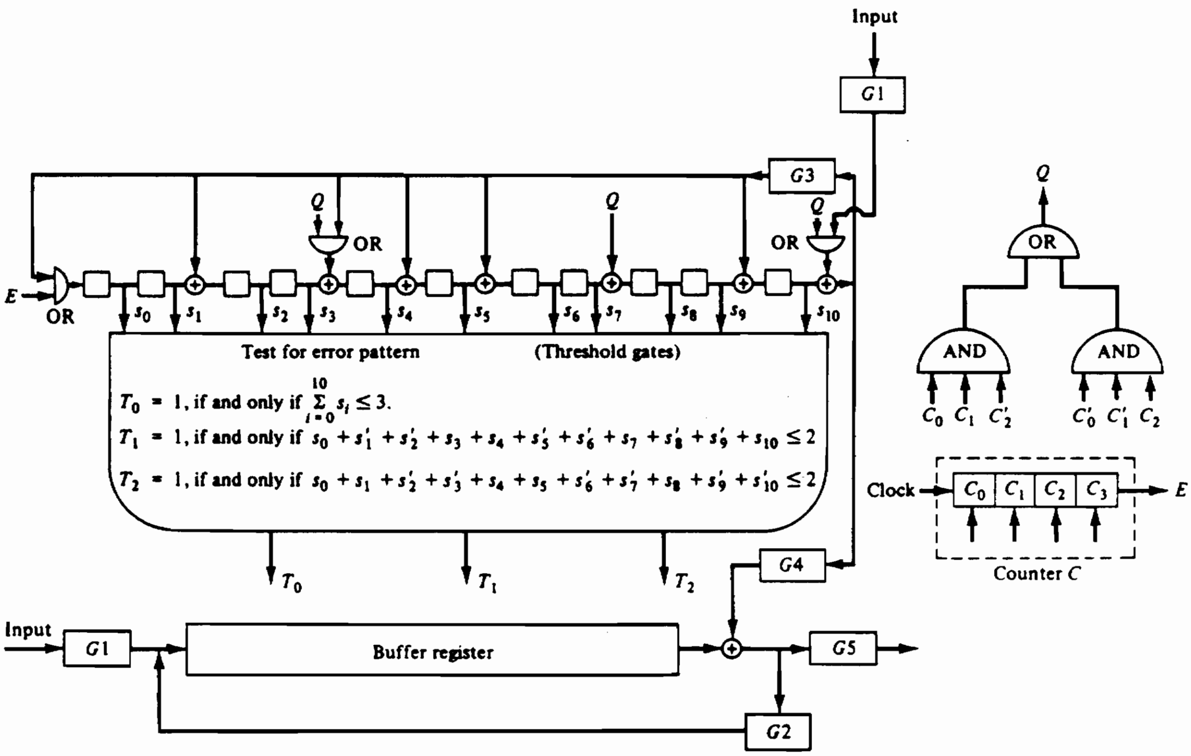


Figure 6.3.1 Decoder circuit for the (23,12) Golay code. From Lin and Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall 1983. p. 136.

CHAPTER 7

The Reed-Solomon Codes

The class of Reed-Solomon (RS) codes is one of the most powerful.[20] These codes are a non-binary sub-class of BCH codes. Reed-Solomon codes are some of the few codes which have maximum separable distance. In other words, no other code can have a greater minimum distance for a given n and k. Because of the ease of implementation, RS code performance was found through computer simulation using a frequency domain technique. However, actual hardware implementation would be performed using a different coding technique. Most coding texts have a section on RS codes. Blahut's text has a good section on RS codes, as well as sections on using the frequency domain techniques.[21]

7.1 Parameters of the Reed-Solomon Codes

This class of codes operates over a non-binary alphabet. Several bits are used to form each symbol. Therefore, when the code is specified as an (n,k) code, the n and k are the number of symbols, not bits. Note that 255 symbols, each 8 bits in length, will yield a packet length of 2040 bits, which is why packets were normalized to this length. The following standardized parameters describe the RS codes:

m = the number of bits per symbol.

$n = 2^m-1$ = the block length in symbols.

k = the number of information symbols.

$n-k = 2t$ = the number of parity check symbols.

$d_{min} = n-k+1$ = the minimum distance in symbols.

$t = (d_{min}-1)/2$ = the maximum number of correctable symbol errors.

7.2 Encoding

Encoding and decoding can be implemented by a number of methods, two of which will be described here.[22] The first will use a linear feedback shift register (LFSR) and would be the method used for hardware implementation. The second will use frequency domain coding, which is the method used for the software simulation. Codes generated by these methods may not be identical but are equivalent. In other words, they have the same properties, including n, k, t, and d_{min}.

7.2.1 Hardware Implementation

If the RS code is encoded cyclicly and systematically, the encoding circuit will be simpler. A LFSR will perform the encoding for such a code, which is shown in Figure 7.2.1.1. In the figure, the g_i's are the coefficients of the generator polynomial, and the b_i's are the parity check symbols. The message is shifted into the register one symbol at a time. Once the entire block has been shifted through the register, the parity check symbols will be left in the storage buffers. These symbols can then be concatenated with the information symbols to yield a systematic code.

7.2.2 Software Implementation

In order to make the coding process faster and simpler, a frequency domain approach was used. Miller et al. have shown that frequency domain coding can be up to 7 times faster than conventional time domain coding.[23] Figure 7.2.2.1 shows a block diagram of such a system. The k information symbols are loaded into a buffer. A sequence of 2t symbols are concatenated to the beginning of the information symbols. These 2t symbols are simply the zero element in the number field being used. An inverse discrete Fourier transform (DFT) is performed on the block of data. The data are then transmitted over the channel.

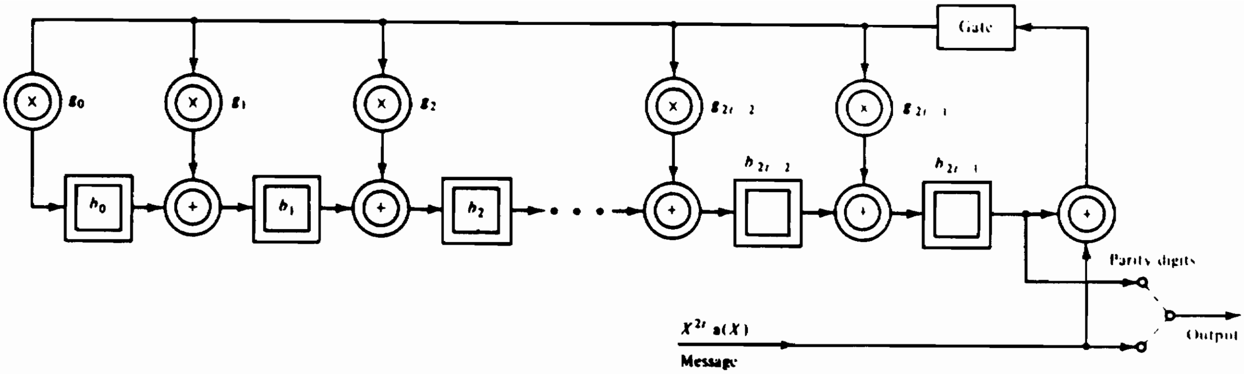


Figure 7.2.1.1 The encoding circuit for Reed-Solomon codes. This circuit will yield a systematic, cyclic code. From Lin and Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall. 1983 p. 172.

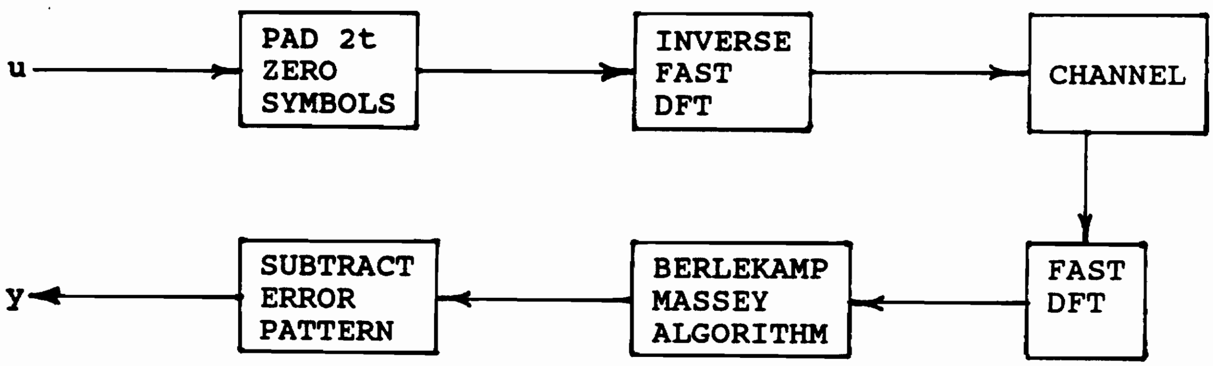


Figure 7.2.2.1 Block diagram of the frequency domain coding technique.

7.3 Performance

Because the Reed-Solomon codes are powerful, codes with several values of n and k were simulated. Among these were the (15,13), (15,11), (15,7), (255,221), (255,187), and (255,119) codes. Note that the codes with n=15 must have 4 bits per symbol, while the codes with n=255 must have 8. This fact aids in the error correcting properties of the n=255 code. Consider this situation. If 5 errors occur over the duration of 8 bits, two of the n=15 symbols will be corrupted, whereas only one of n=255 symbols may be in error.

As previously mentioned, the performance of these codes was tested using computer simulations. A block of symbols was sent through the encoder. A random number generator was used to corrupt these symbols at a certain rate, the bit error rate. The corrupted block was then sent through the decoder. If the decoder failed, the block was flagged. The number of flagged blocks was recorded. In order to reduce the effect of randomness in the results, this process was repeated until at least 10 blocks had been flagged. It may be of interest to note that at low bit error rates and code rates, some simulations required a week of continuous computer time, on a 12 MHz 286 type machine.

Plots of input bit error rate vs. output packet error rate for the (15,k) codes can be found in Figure 7.3.1.

These results were normalized to 2040 bits using equation 5.2.2 with q=34 and P_c=the codeword error rate.

$$P_p=1-(1-P_c)^q \qquad (5.2.2)$$

Plots for the (255,k) codes are in Figure 7.3.2. A comparison of the two codes is in Figure 7.3.3. Plots for the throughput, found by equation 2.2.2.2 with N=10, for the (15,k), (255,k), and both are found in Figures 7.3.4, 7.3.5, and 7.3.6 respectively.

$$T_{GBN}=(1-P_p)k/n((1-P_p)+P_pN) \qquad (2.2.2.2)$$

Reed-Solomon codes perform well on burst errors. Without interleaving, the (15,k) codes can always correct 1, 5 and 13 errors, for the (15,13), (15,11) and (15,7) codes, respectively. Interleaving with a degree of 34 would increase these numbers to 133, 269, and 541, respectively. The (255,k) codes can always correct 129, 265 and 537 errors, for the (255,221), (255,187) and (255,119) codes, respectively. Since the packet size is limited to 2040 bits, no intra-packet interleaving can be performed. However, inter-packet interleaving would increase these numbers.

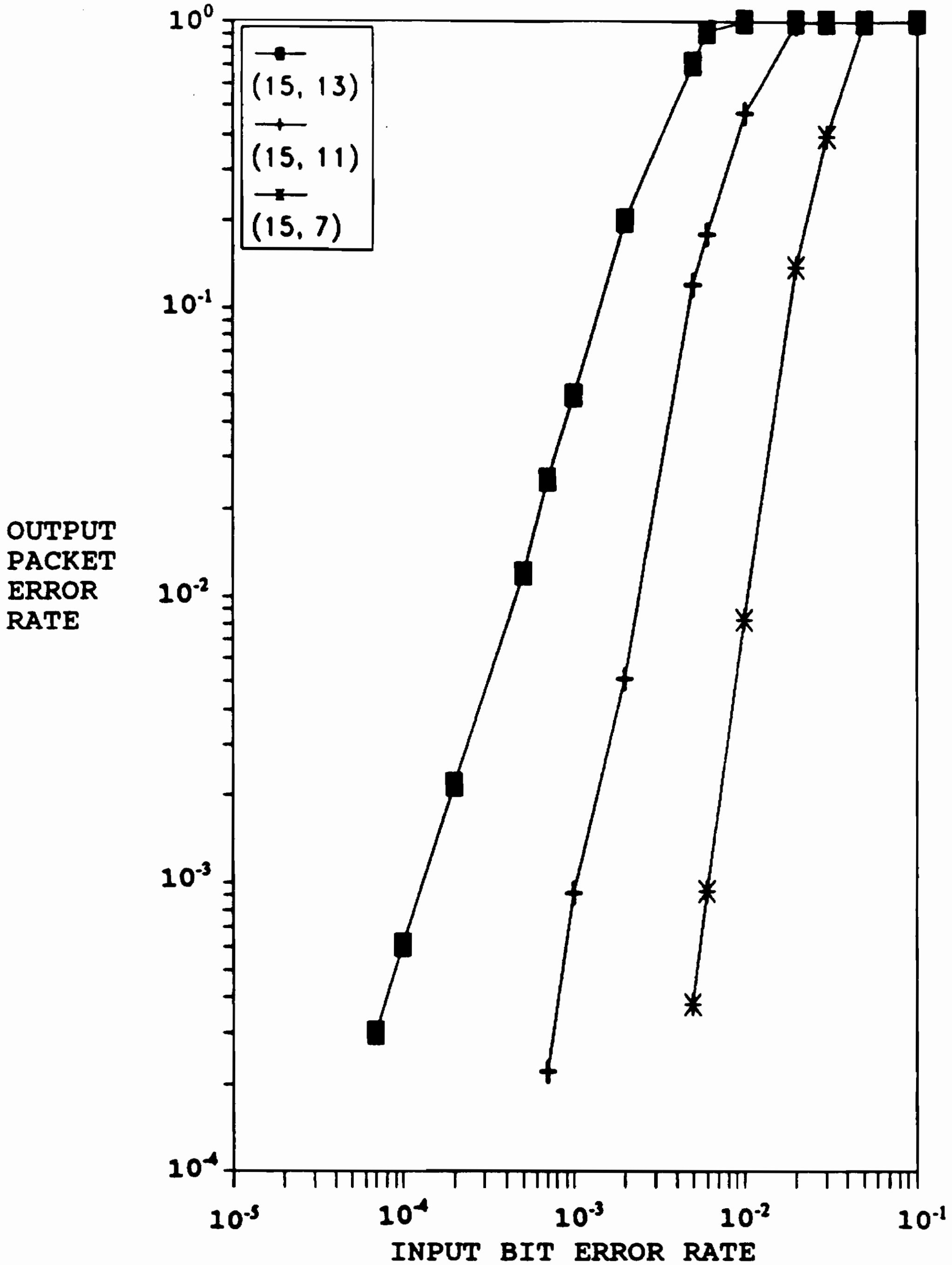


Figure 7.3.1 The output packet error rate vs. the input bit error rate for the (15,k) Reed-Solomon codes.

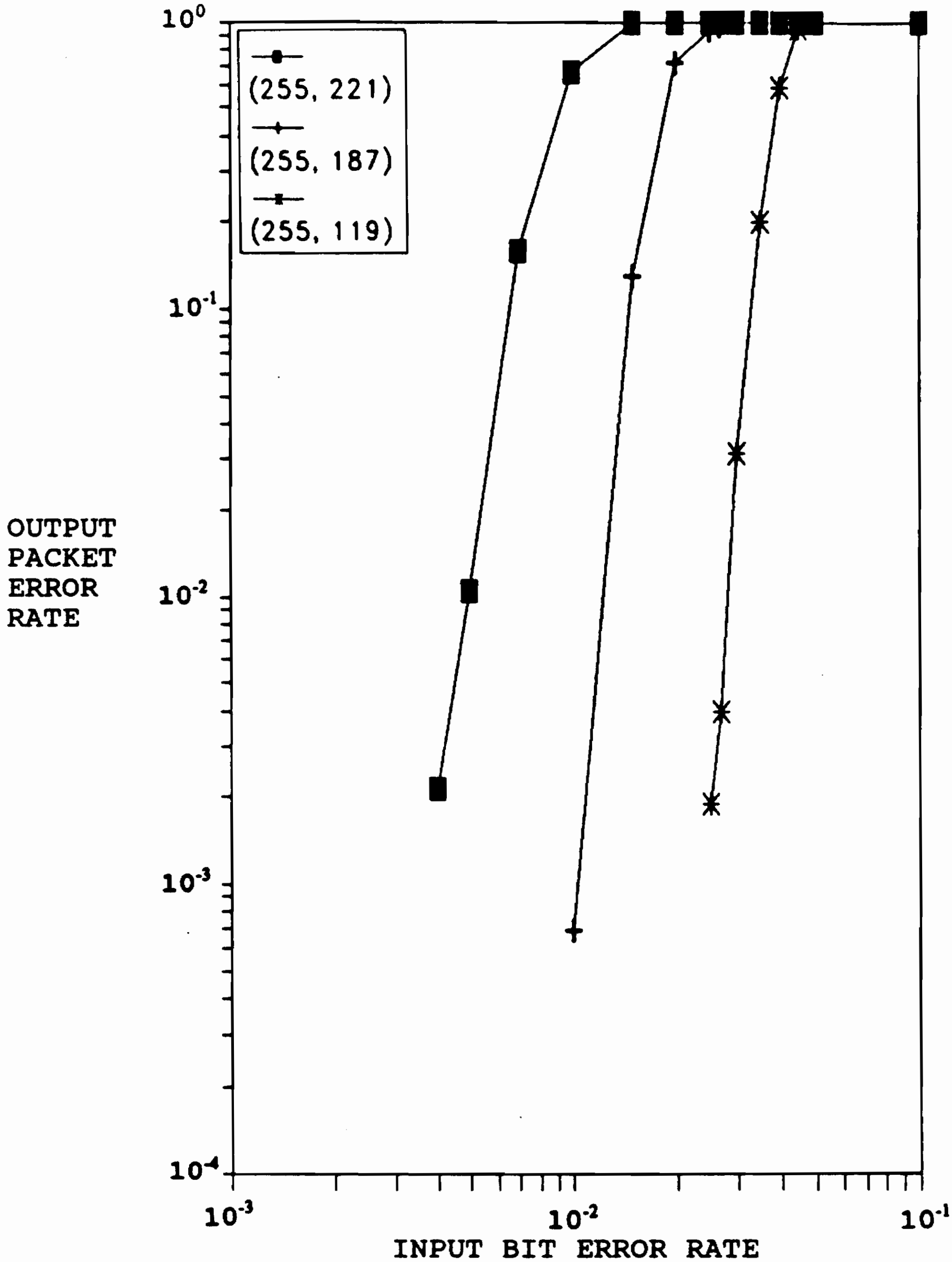


Figure 7.3.2 The output packet error rate vs. the input bit error rate for the (255,k) Reed-Solomon codes.

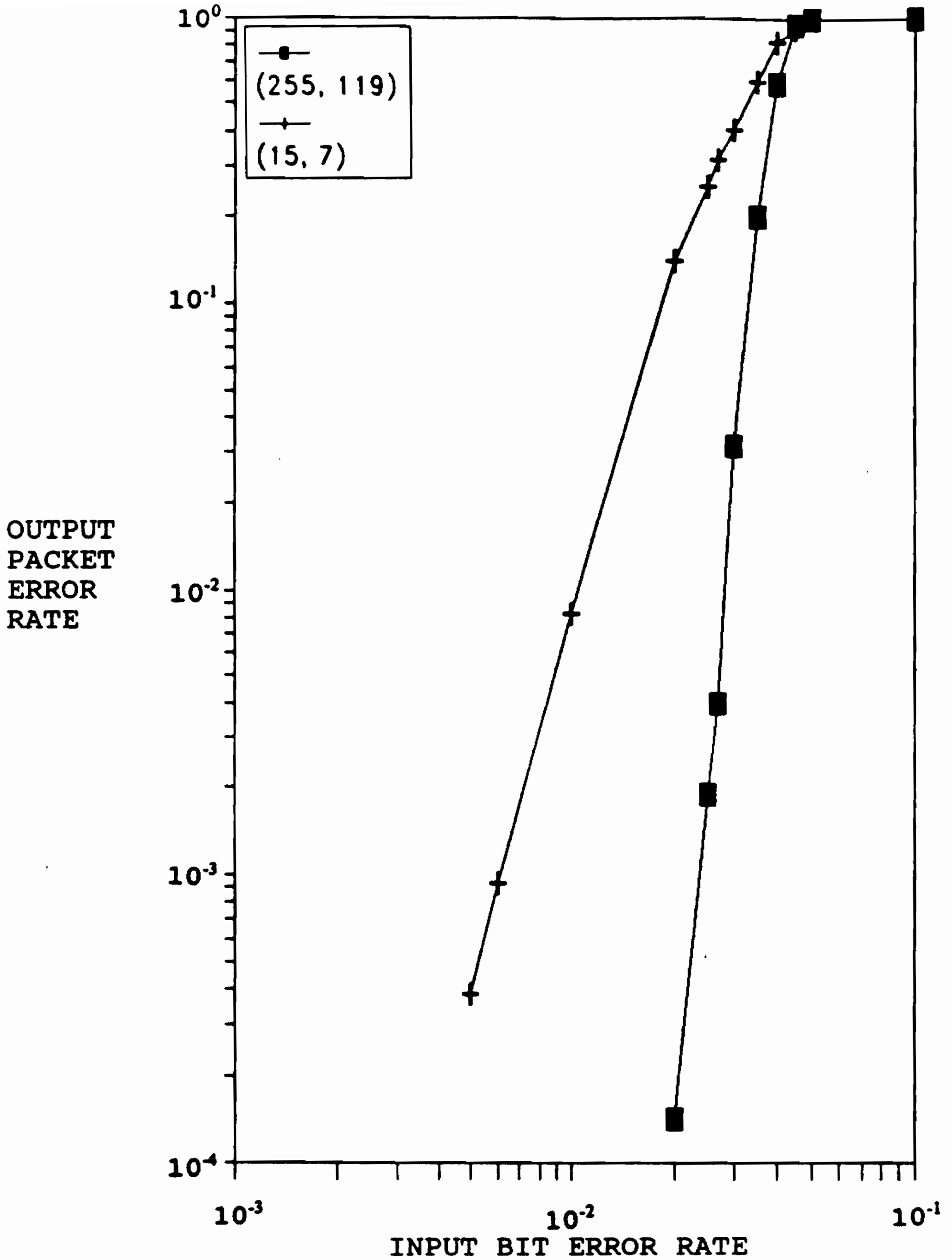


Figure 7.3.3 The output packet error rate vs. input bit error rate for the RS (255,119) and the RS RS (15,7) codes.

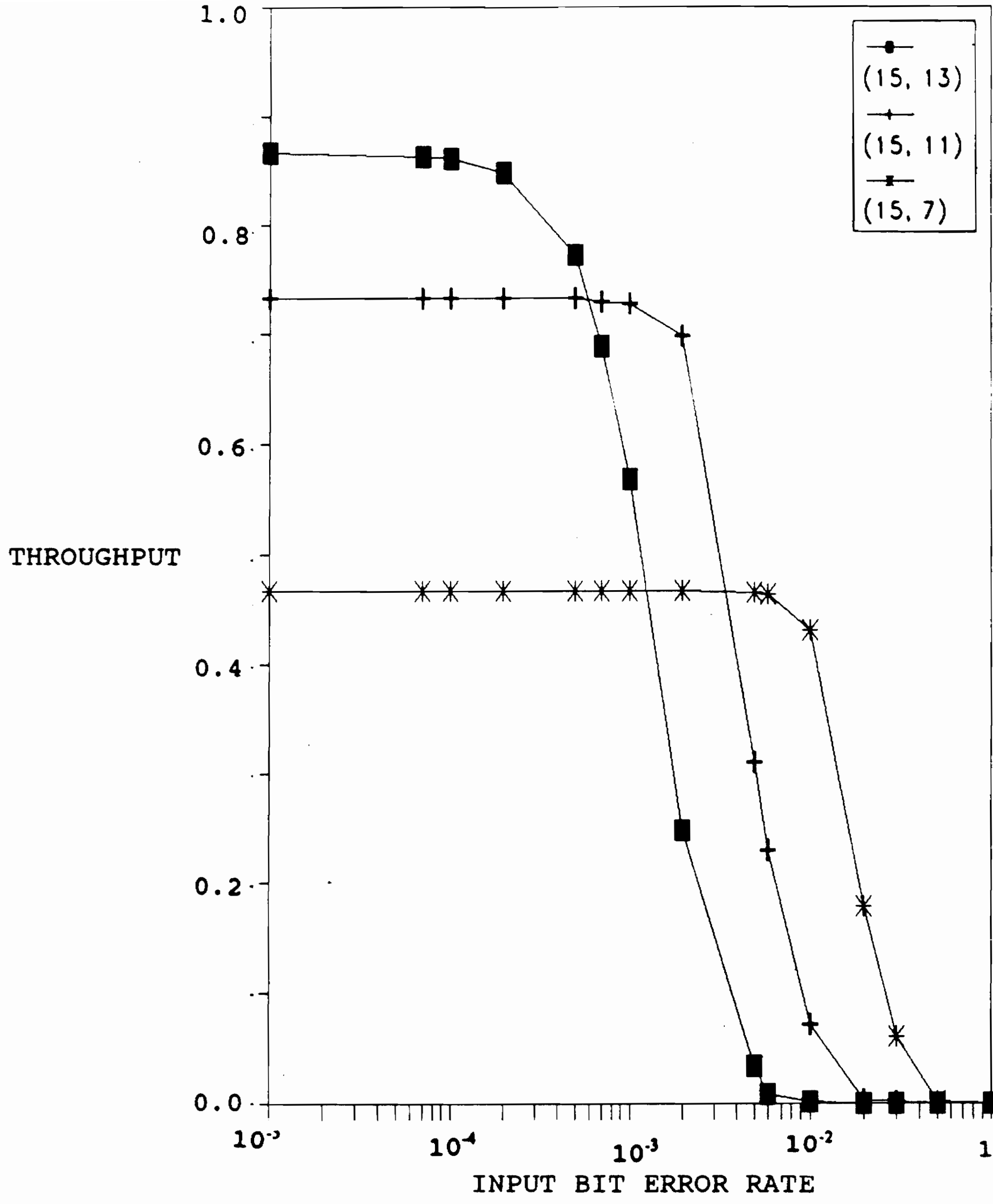


Figure 7.3.4 The throughput vs. input bit error rate for the normalized (15,k) RS codes using go-back-10 ARQ.

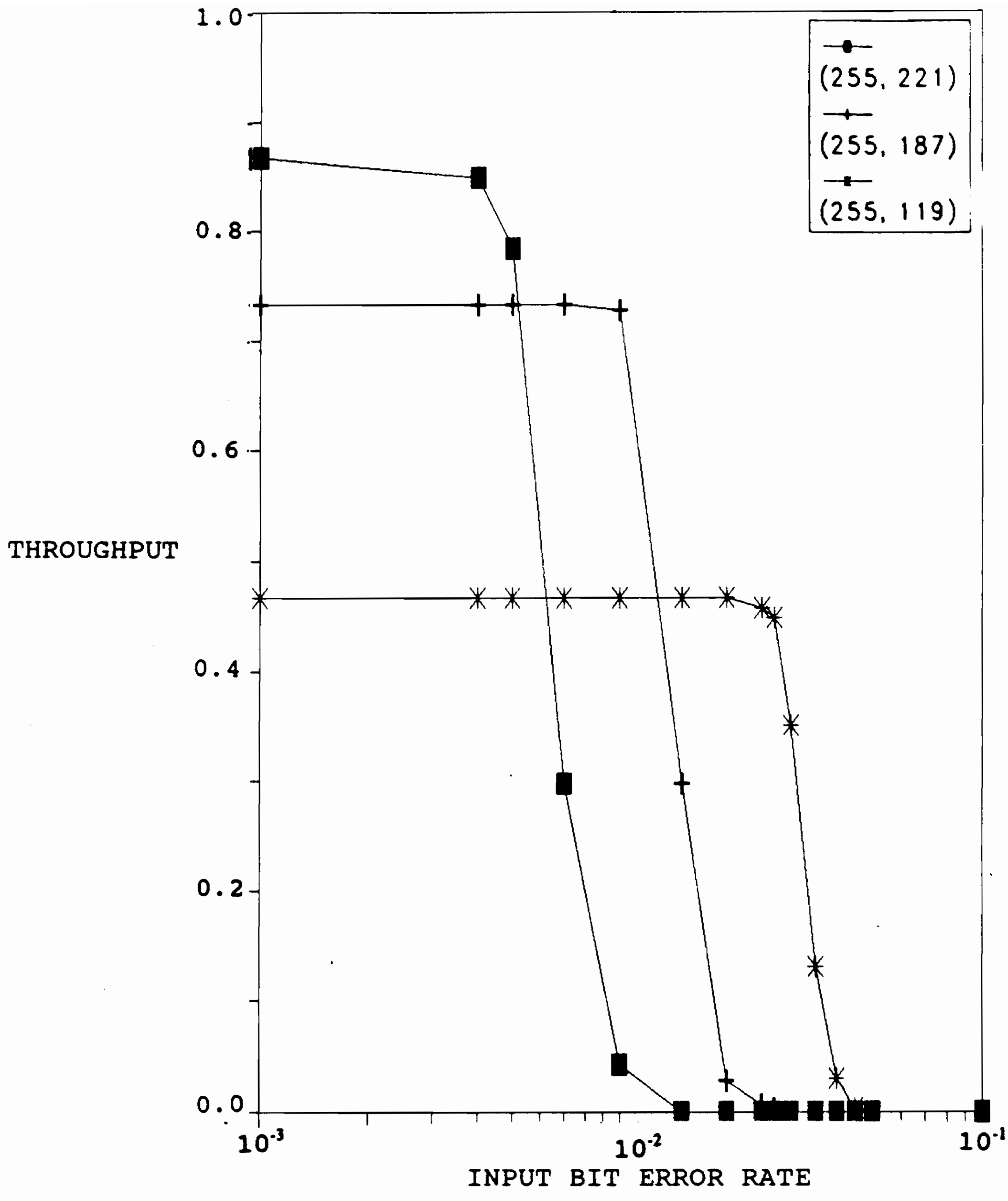


Figure 7.3.5 The throughput vs. input bit error rate for the RS (255,k) codes using go-back-10 ARQ.

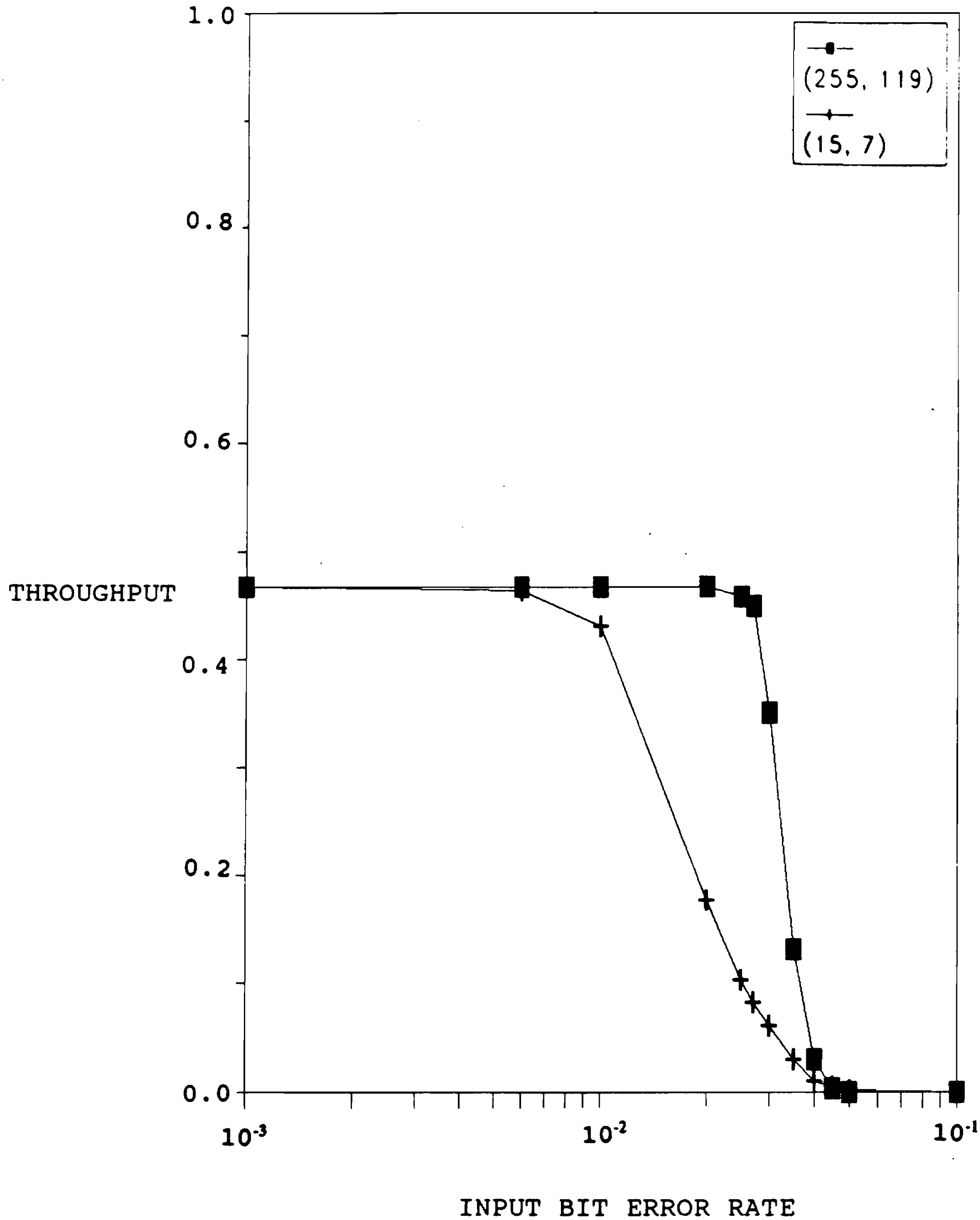


Figure 7.3.6 Throughput vs. input bit error rate for the (15,7) and (255,119) Reed-Solomon codes using go-back-10 ARQ.

7.4 Decoding

As with the encoding process, the decoding implementation is described in both the time domain and frequency domain. As noted before, these processes do not yield the same code and cannot be interchanged. However, the codes are equivalent.

An important property of Reed-Solomon codes is that the syndrome, **s**, is found by substituting the first 2t powers of alpha into the received code word, **r** = r(x). Alpha is simply a representation of a number in the Galois field. In equation form:

$$\begin{gathered} s_1=\mathbf{r}(\alpha^1) \\ s_2=\mathbf{r}(\alpha^2) \\ \vdots \\ s_{2t}=\mathbf{r}(\alpha^{2t}) \end{gathered} \qquad (7.4.1)$$

7.4.1 Hardware Implementation

For the implementation in hardware, the syndrome can be calculated through simple polynomial division. The theory behind this fact is beyond the scope of this paper. However, any coding book should have the relevant material. Lin & Costello have a readily understandable section on this material. This division can be accomplished using the simple circuit in Figure 7.4.1.1. Once this division has been performed, the synthesis of the error pattern is carried out by the Berlekamp-Massey algorithm.[24] [25] This error pattern is simply subtracted from the received code vector to yield the corrected message.

7.4.2 Software Implementation

The software implementation relies on frequency domain decoding. Once the block of data is received, a discrete Fourier transform is performed. It can be shown for Reed-Solomon codes that the Fourier transform over the Galois number field, which is used in all codes, is given by equation 7.4.1.[26] Therefore, the DFT computes the syndromes. The Berlekamp-Massey algorithm is used to compute the error pattern, and this pattern is subtracted from the received code vector.

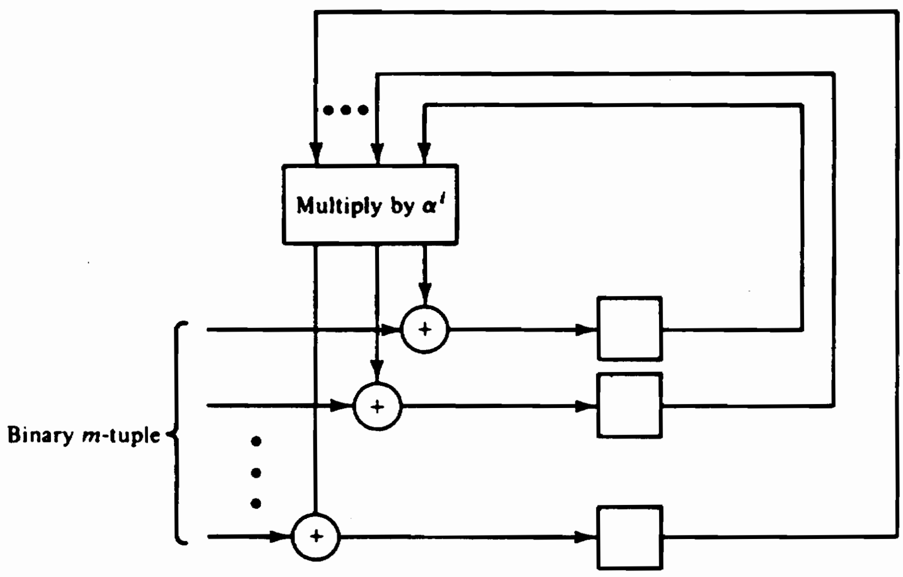


Figure 7.4.1.1 Circuit for finding the syndrome of a Reed-Solomon code.

CHAPTER 8

Convolutional Codes

Convolutional codes are quite different from linear block codes.[27] The encoder output depends on the past input, and the codes are typically encoded with a sequential circuit with memory. These codes are often decoded by the Viterbi algorithm, which is asymptotically equivalent to maximum likelihood decoding. Most coding texts have good references for convolutional codes. There are also many excellent papers on convolutional codes.[28-31]

8.1 Encoding

The encoding process for a convolutional code is relatively simple. The parameter k is the number of information bits entered into the encoder. These k bits will produce an output of n bits. The output depends on the memory of the encoder. For this paper, the constraint length, L, is defined as one more than the length of all memory registers.[32] Some authors define L as the maximum number of output bits a single input bit can affect.[33] However, the first definition is the more common. Most communications systems operate with constraint lengths between 6 and 9.

Figure 8.1.1 shows an encoder for a (3,2,2) code. The register is initialized with zeros, and the k=2 information bits are entered into the encoder. Each of these bits produces n=3 output bits. A switch alternates between the 3 sequences of outputs to keep the n bits in order.

8.2 Performance

The performance of convolutional codes is dependant on the code's transfer function and a property d_{free}. The value of d_{free} is the minimum number of positions that any code word differs from any other. Higher values of n and L will yield higher values of d_{free}. The transfer function can be found by tracing paths through a state diagram and counting non-zero bits. An upper bound for the probability of bit error is given by:[34]

$$P_b < T(D)/k \tag{8.2.1}$$

where $D=2\,p(1-p)$.

p = the bit transition probability.

k = the number of input bits.

$T(D)$ = a polynomial representation of the transfer function. (i.e. $T(D)=w_{dfree}D^{dfree}+w_{dfree+1}D^{dfree+1}+\ldots$)

The transfer function representation was truncated to 5 terms. Values for d_{free} and the transfer function terms can be found in Table 8.2.1.[35] Van de Meeberg and Post have done some work to tighten the upper bound of equation 8.2.1.[36] [37] However, at small bit error rates the bounds converge to the true values. Figure 8.2.1 shows actual values vs. calculated values from equation 8.2.1. Note that the actual performance is somewhat better than the calculated performance, and one should keep this fact in mind when comparing the packet error rate to other codes. For comparison to the other codes, the packet error rate was found for a length of 2040 bits. This calculation was accomplished by using equation 5.2.2 with $P_c=P_b$ and q=2040.

$$P_p=(1-P_c)^q \qquad (5.2.2)$$

Plots of output packet error rate vs. input bit error rate for R=1/3, 1/2, 2/3, and 3/4 with L=6-9 can be found in Figures 8.2.2, 8.2.3, 8.2.4, and 8.2.5, respectively. Plots of throughput vs. input bit error rate can be found in Figures 8.2.6-8.2.9 for the same codes. Throughput was found using equation 2.2.2.2.

$$T_{GBN}=(1-P_p)k/n((1-P_p)+P_pN) \qquad (2.2.2.2)$$

The burst performance is also of interest. A convolutional code is guaranteed to correct a burst of length $(d_{free}-1)/2$ errors without interleaving. Because convolutional codes require calculating approximately 5*L metrics to choose the appropriate path, convolutional codes are interleaved to this degree. Table 8.2.2 shows the burst performance of the convolutional codes discussed in this section.

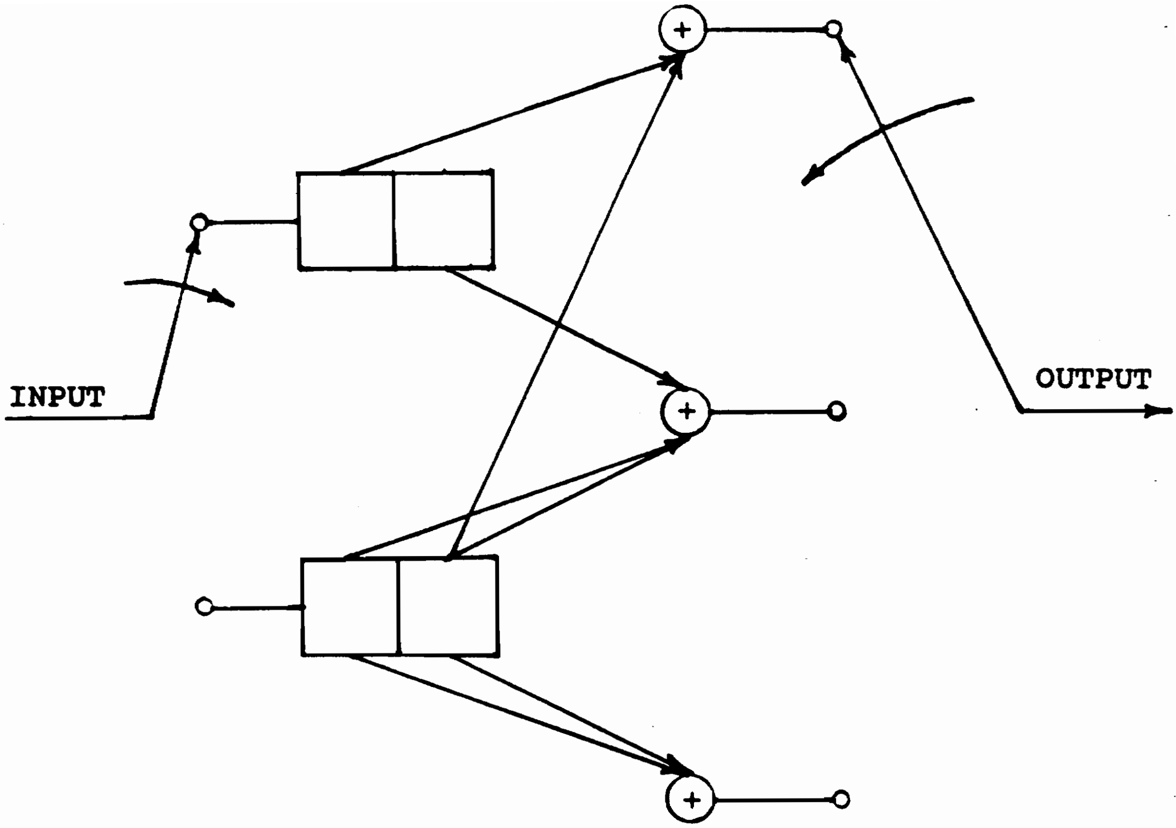


Figure 8.1.1 Encoding circuit for a (3,2,2) convolutional code.

Table 8.2.1 Values of d_{free} and transfer function coefficients for various rates of the best convolutional codes. Values from Clark and Cain, *Error-Correction Coding for Digital Communications*, 1981.

R=1/3

L	d_{free}	w_{dfree}	$w_{dfree+1}$	$w_{dfree+2}$	$w_{dfree+3}$	$w_{dfree+4}$
5	12	12	0	12	0	56
6	13	1	8	26	20	19
7	14	1	0	20	0	53
8	16	1	0	24	0	113

R=1/2

L	d_{free}	w_{dfree}	$w_{dfree+1}$	$w_{dfree+2}$	$w_{dfree+3}$	$w_{dfree+4}$
6	8	2	36	32	62	332
7	10	36	0	211	0	1404
8	10	2	22	60	148	340
9	12	33	0	281	0	2179

R=2/3

L	d_{free}	w_{dfree}	$w_{dfree+1}$	$w_{dfree+2}$	$w_{dfree+3}$	$w_{dfree+4}$
6	6	75	0	1571	0	31474
7	6	1	81	402	1487	6793
8	8	395	0	6695	0	235288
9	8	97	0	2863	0	56633

R=3/4

L	d_{free}	w_{dfree}	$w_{dfree+1}$	$w_{dfree+2}$	$w_{dfree+3}$	$w_{dfree+4}$
6	5	78	572	3831	24790	152108
7	6	919	0	31137	0	1142571
8	6	117	0	8365	0	319782
9	6	12	342	1996	12296	78145

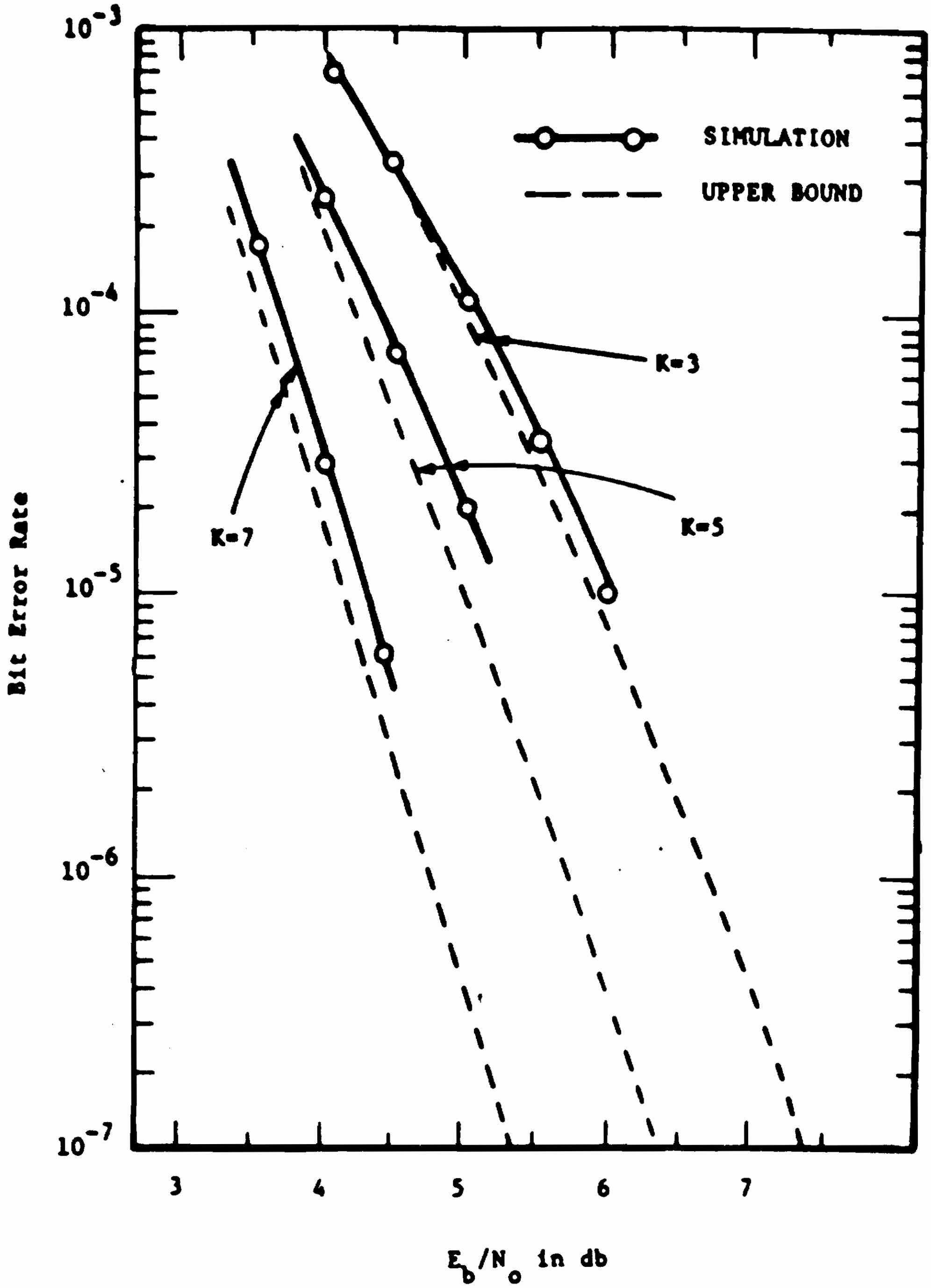


Figure 8.2.1 Plot of measured values vs. calculated values from equation 8.2.1. From Heller and Jacobs "Viterbi Decoding for Satellite and Space Communication," *IEEE Transactions on Communication Technology*, 1971.

Table 8.2.2 The number of consecutive bits which are guaranteed to be corrected with and without interleaving for various convolutional codes.

R=1/3

L	d_{free}	# OF BITS CORRECTABLE W/O INTERLEAVING	INTERLEAVING DEGREE	# OF BITS CORRECTABLE W/INTERLEAVING
5	12	5	81	405
6	13	6	68	408
7	14	6	58	348
8	16	7	51	357

R=1/2

L	d_{free}	# OF BITS CORRECTABLE W/O INTERLEAVING	INTERLEAVING DEGREE	# OF BITS CORRECTABLE W/INTERLEAVING
6	8	3	68	204
7	10	4	58	232
8	10	4	51	204
9	12	5	45	225

R=2/3

L	d_{free}	# OF BITS CORRECTABLE W/O INTERLEAVING	INTERLEAVING DEGREE	# OF BITS CORRECTABLE W/INTERLEAVING
6	6	2	68	136
7	6	2	58	116
8	8	3	51	153
9	8	3	45	135

R=3/4

L	d_{free}	# OF BITS CORRECTABLE W/O INTERLEAVING	INTERLEAVING DEGREE	# OF BITS CORRECTABLE W/INTERLEAVING
6	5	2	68	136
7	6	2	58	116
8	6	2	51	102
9	6	2	45	90

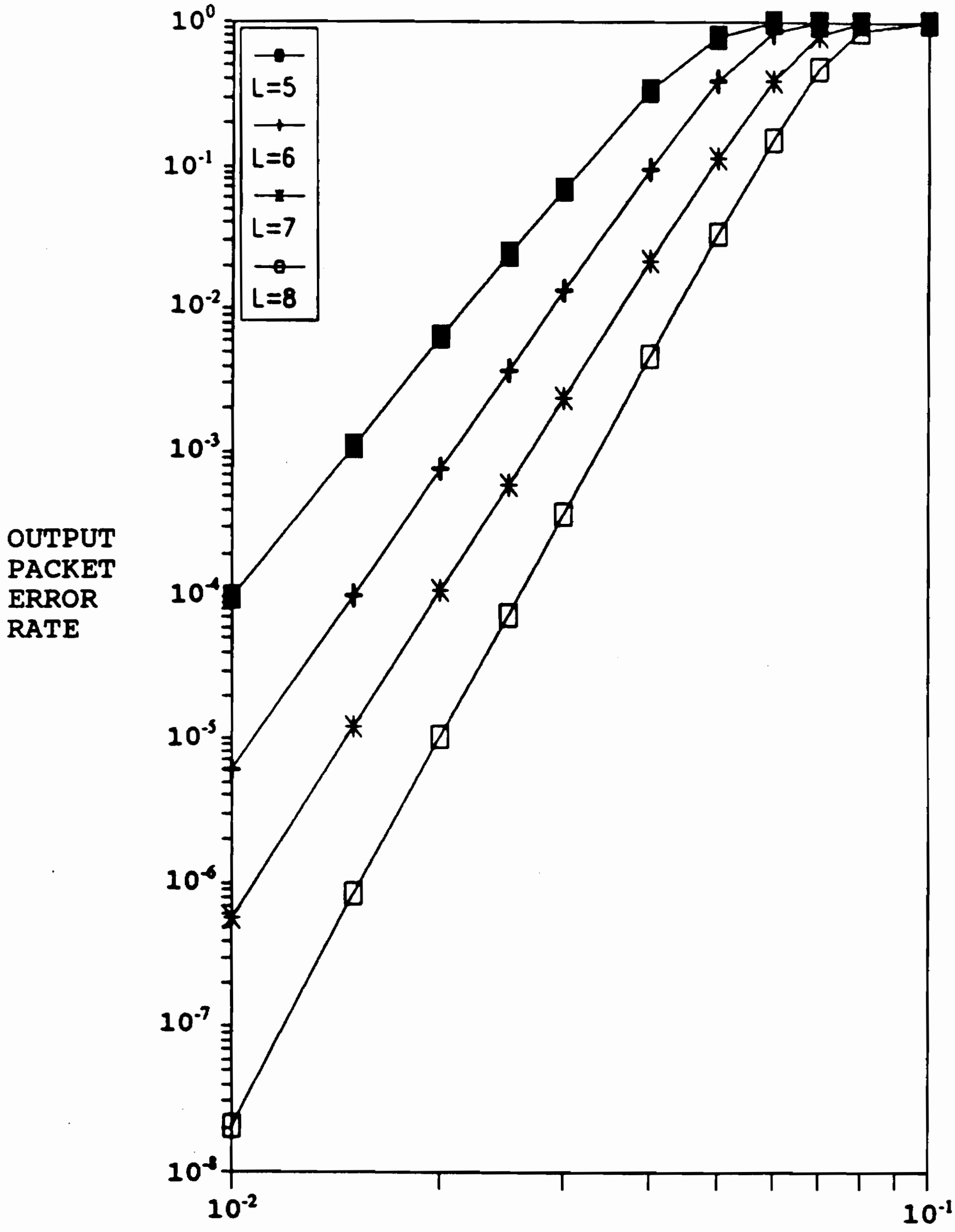


Figure 8.2.2 Output packet error rate (2040 bit packet) vs. input bit error rate for R=1/3 convolutional codes, with constraint length between 5 and 8.

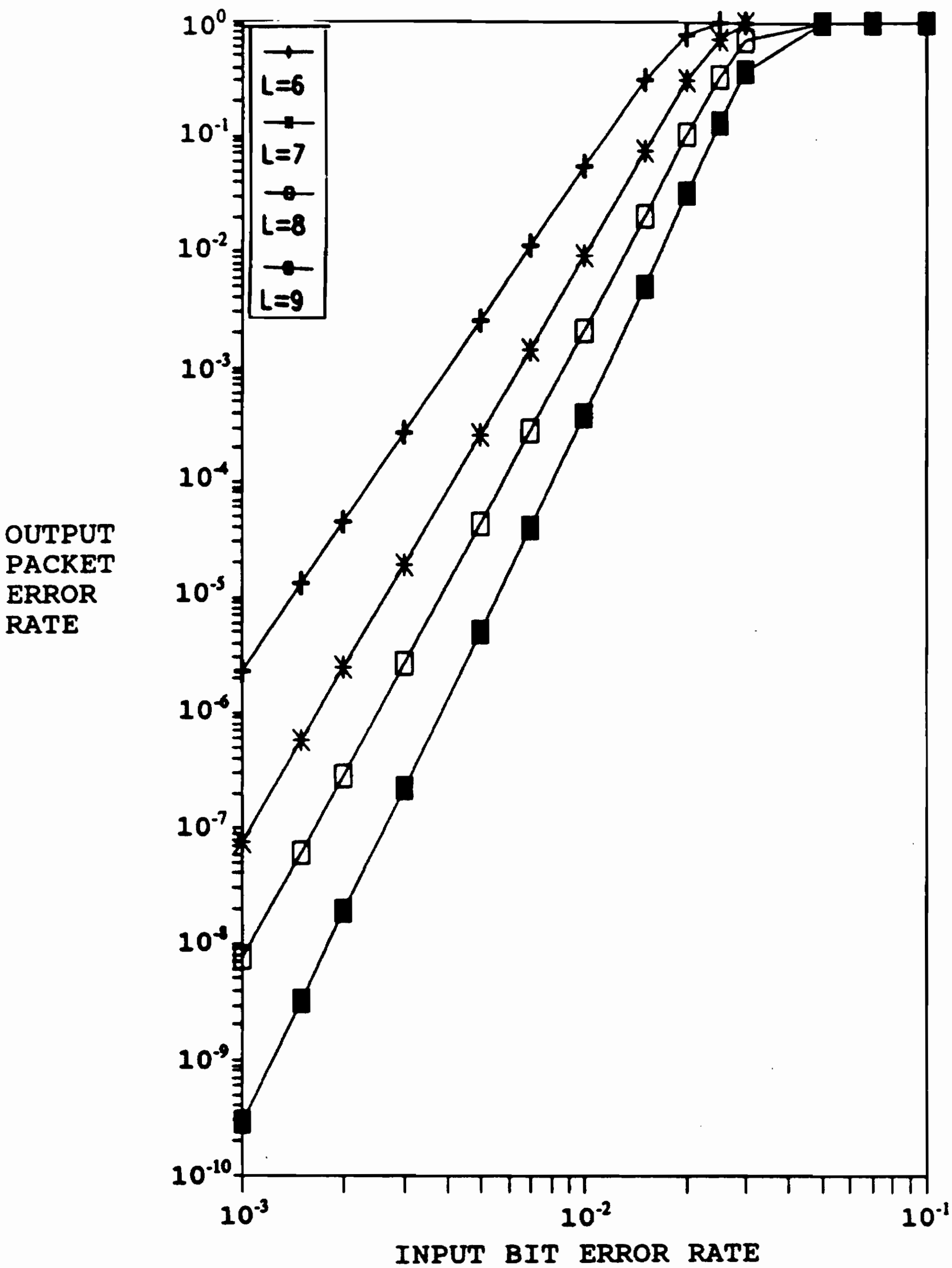


Figure 8.2.3 Output packet error rate (2040 bit packet) vs. input bit error rate for R=1/2 convolutional codes, with L=6-9.

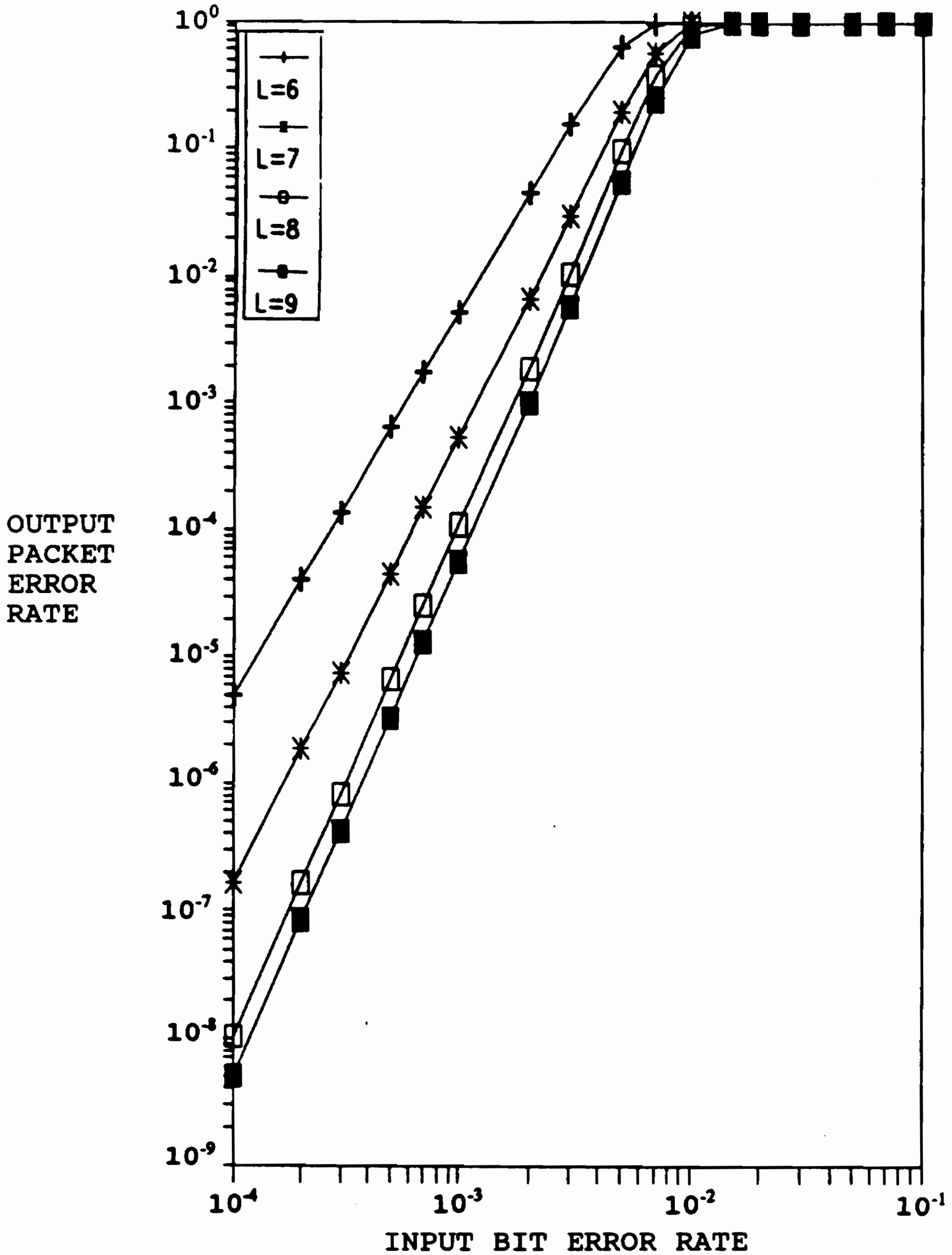


Figure 8.2.4 Output packet error rate (2040 bit packet) vs. input bit error rate for R=2/3 convolutional codes, with L=6-9.

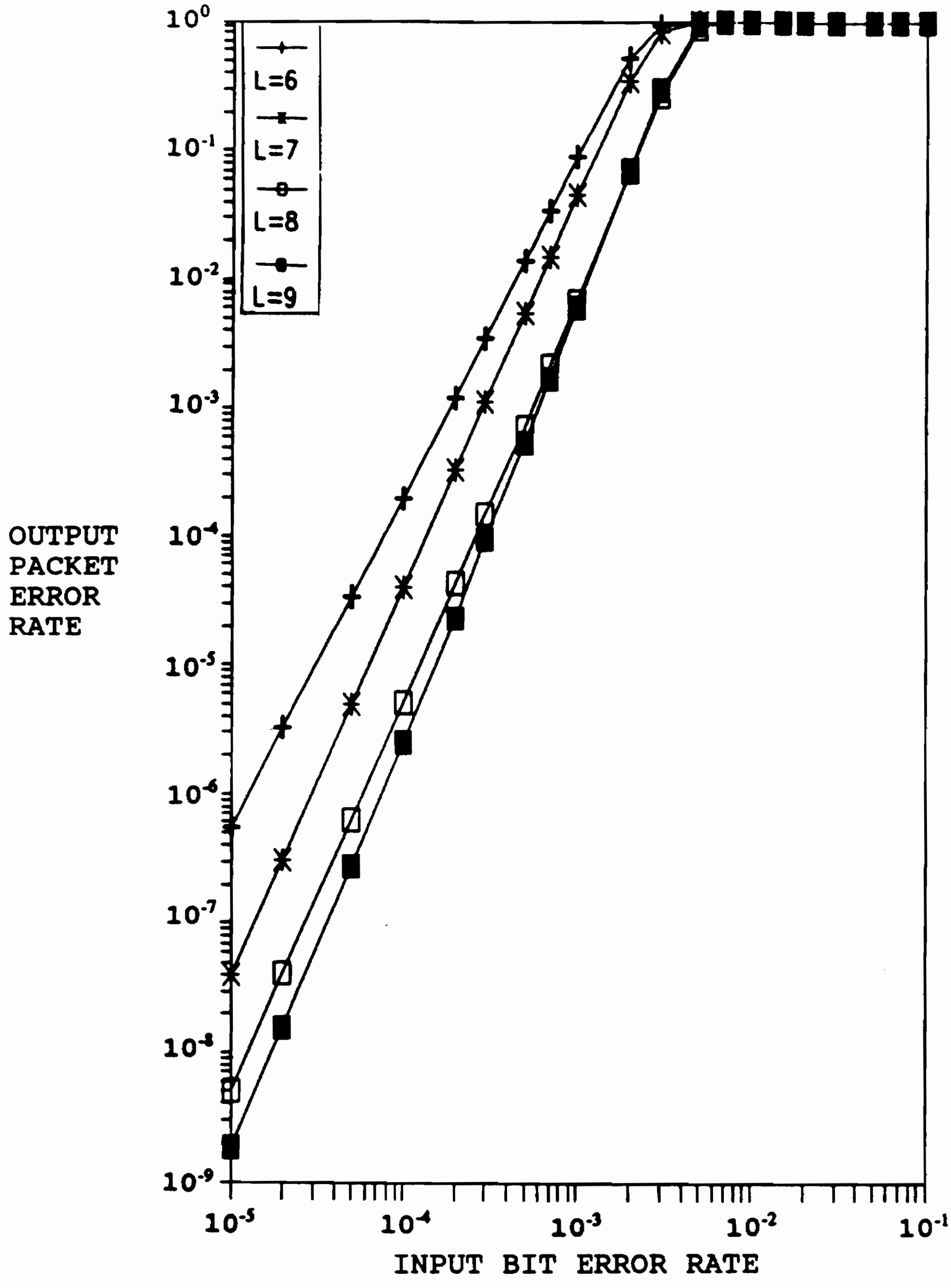


Figure 8.2.5 Output packet error rate (2040 bit packet) vs. input bit error rate for R=3/4 convolutional codes, with L=6-9.

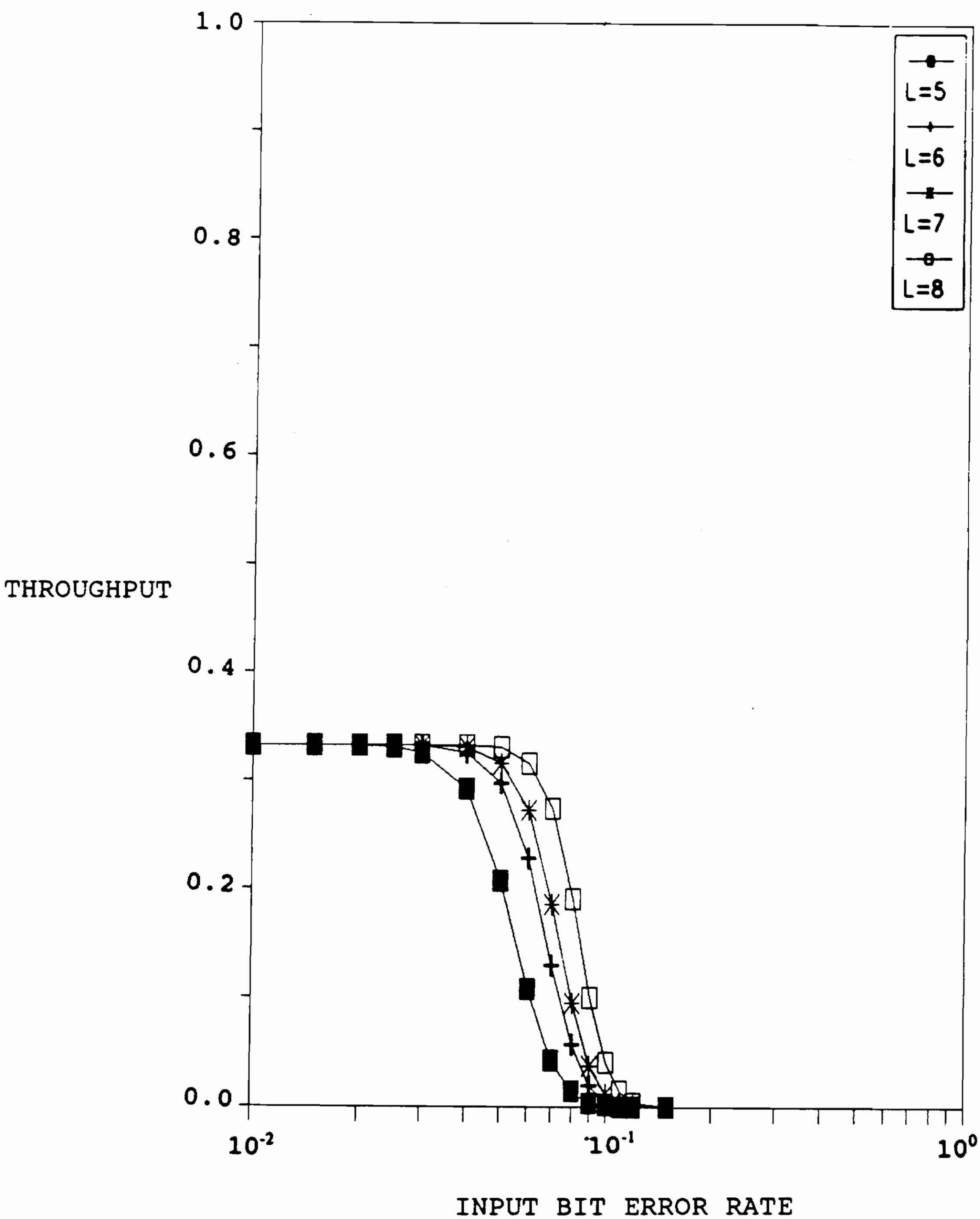


Figure 8.2.6 Throughput vs. input bit error rate for R=1/3 convolutional codes, using go-back-10 ARQ.

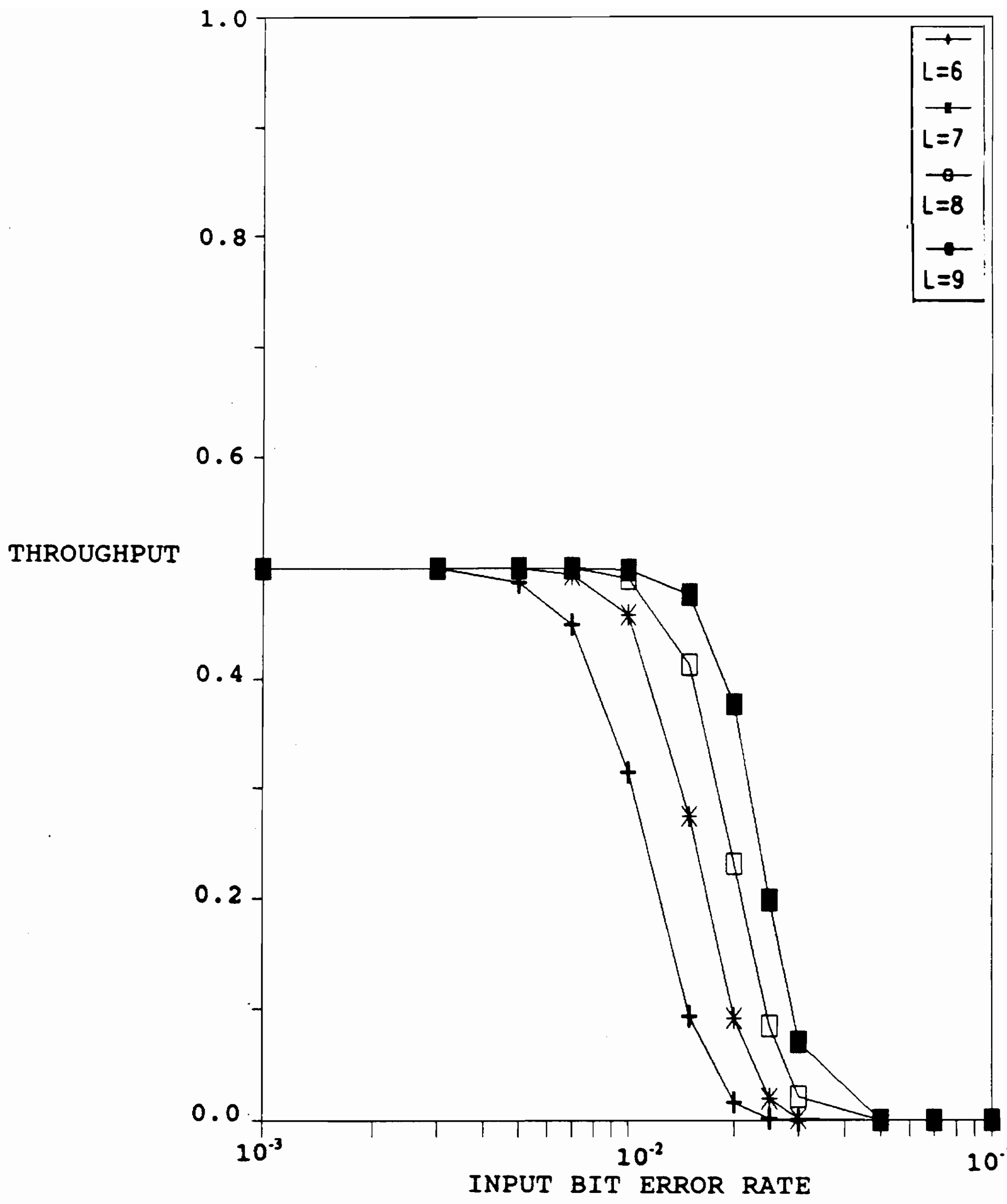


Figure 8.2.7 Throughput vs. input bit error rate for R=1/2 convolutional codes, using go-back-10 ARQ.

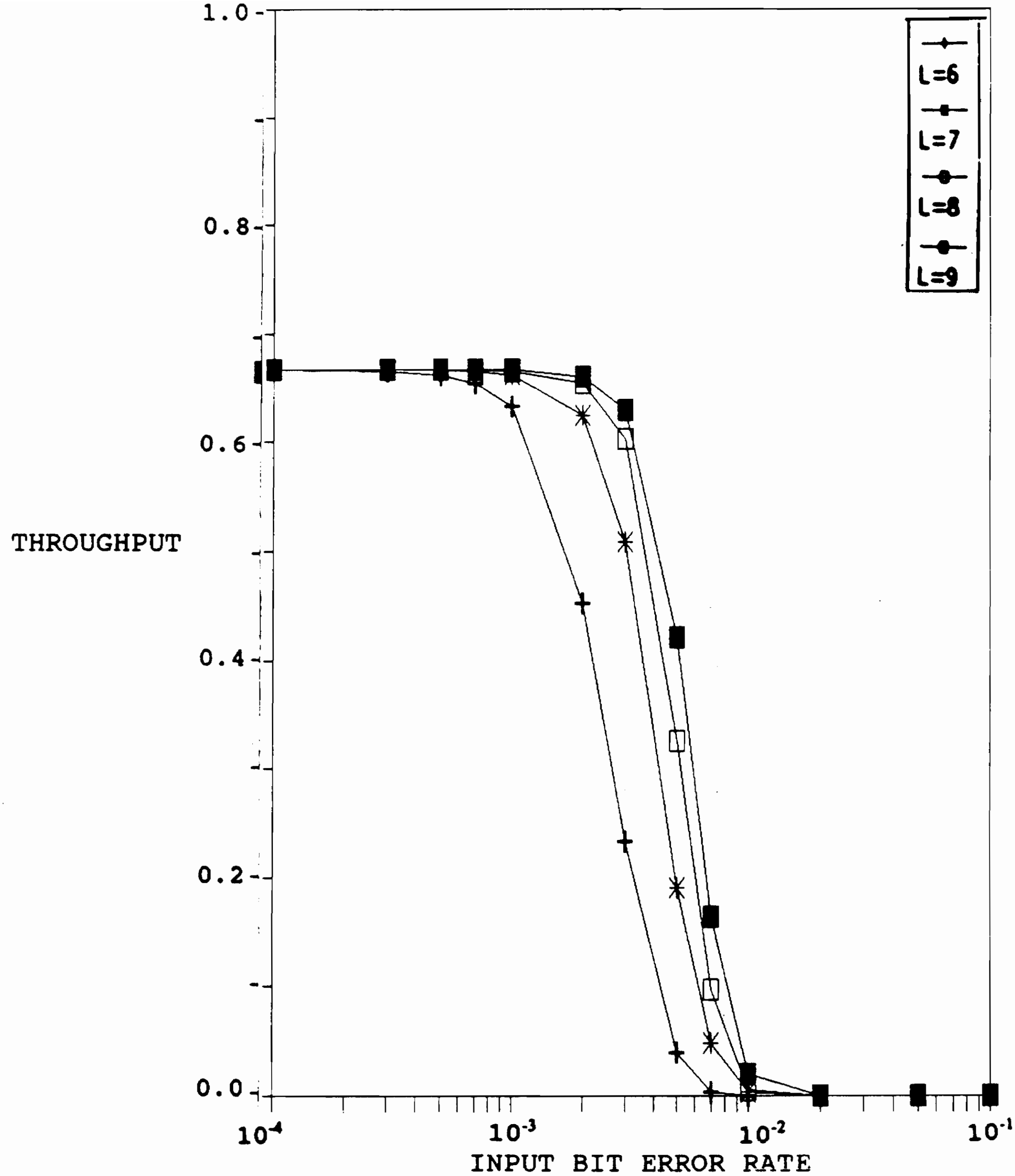


Figure 8.2.8 Throughput vs. input bit error rate for R=2/3 convolutional codes, using go-back-10 ARQ.

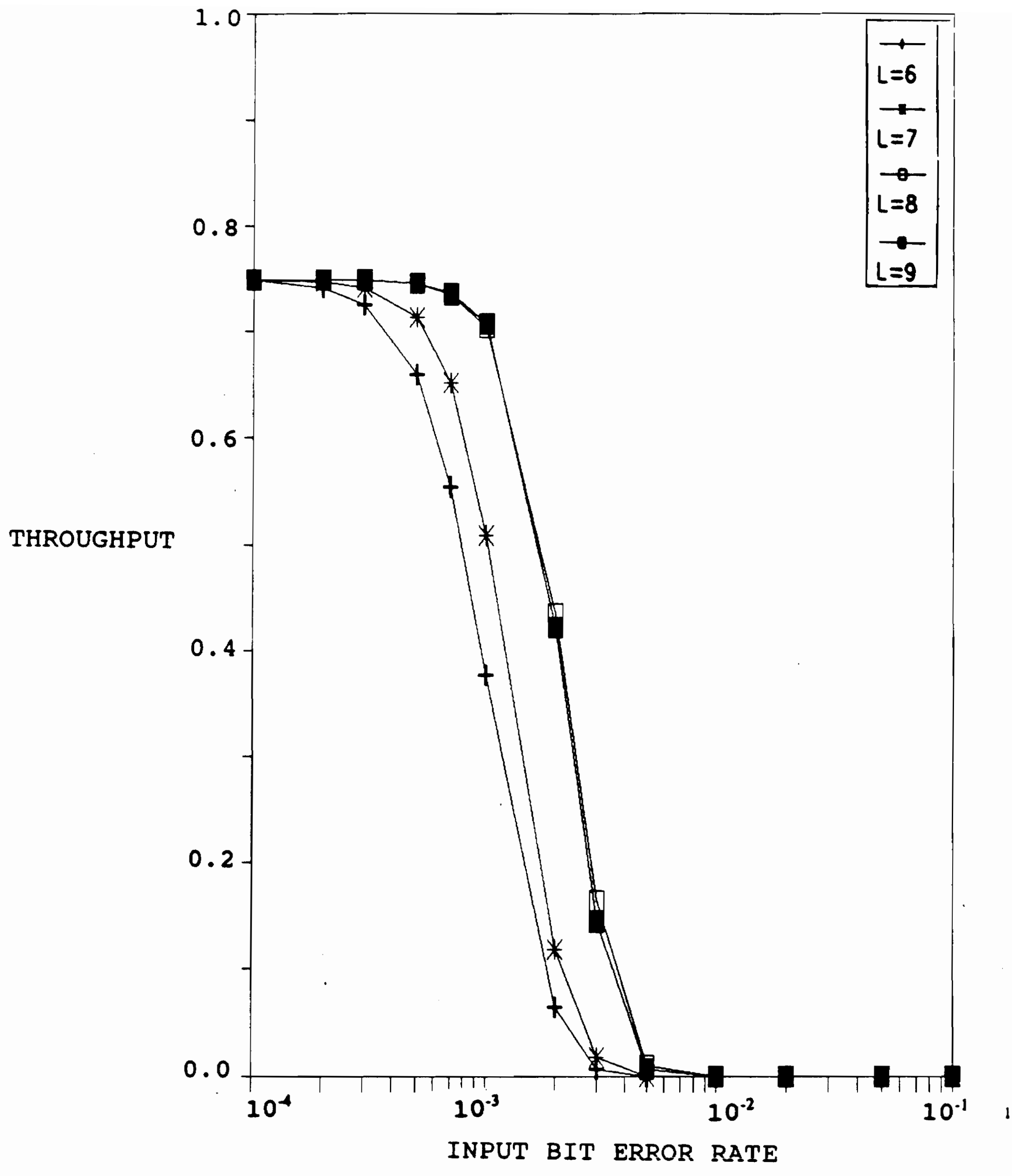


Figure 8.2.9 Throughput vs. input bit error rate for R=3/4 convolutional codes, using go-back-10 ARQ.

8.3 Decoding

The most widely used decoding scheme for convolutional codes is the Viterbi algorithm. This algorithm has been shown to be equivalent to maximum likelihood decoding. [38] [39] The process is described using a tree-like structure called a trellis. Each node is labeled with a partial metric, which is found using a predetermined formula. The path with the largest total metric is chosen to be the proper path through the trellis. Each branch corresponds to a possible k bit sequence. Once the proper path is chosen, the decoded information is recovered. The decoding process requires approximately 2^L memory units to store the metrics.[40] A delay is encountered because each step requires the calculation of the partial metrics for approximately 5*L previous nodes.[41] For a more thorough treatment of decoding see Lin and Costello's *Error Control Coding: Fundamentals and Applications*, 1983 and Forney's "The Viterbi Algorithm," *Proceedings of the IEEE*, 1973.

CHAPTER 9

Analysis of Data

The goal of this project is to determine which, if any, type of error control coding should be utilized on packetized, LEOSAT channels, using ARQ. The most important factor is the throughput of such a system. In figure 1.2.1, it has been demonstrated that coding can have a profound effect on these types of channels. The error correcting code which yields the best performance, as evaluated by certain criteria, must be selected.

As mentioned in Chapter 1, the codes must be evaluated according to specific criteria. The first of these criteria is the throughput vs. input bit error rate, BER, which will allow one to determine exactly how much information can be passed per unit time at a given bit error rate. Secondly, the simplicity of the coding process will affect the choice. A code that is complex and difficult to implement will be a poor option. The third factor that is of concern is the burst error performance of the codes. Some locations may have bursty channels, especially in the lower frequency region of the spectrum. By investigating all of these criteria, a superior error control code should be found.

9.1 Throughput Criterion

The first of the parameters to be examined is the throughput vs. input bit error rate. A code which has a high throughput for a given input bit error rate will be regarded as being superior to one with a low throughput.

The (7,4) Hamming code was included as a benchmark because it is one of the most readily understood codes. Its performance is orders of magnitudes poorer than the best Reed-Solomon or convolutional codes. For this reason, it is not chosen as a viable code.

The (23,12) Golay code was included because it is one of the few perfect codes. By comparison of Figure 6.2.2 with plots from the stronger codes, one can conclude that the performance of the Golay code is indeed inferior. Therefore, the Golay code was rejected.

Form Figure 7.3.6, one can draw the conclusion that the (255,119) Reed-Solomon code has a much higher throughput, for a given input BER, than the (15,7) code. Figures 7.3.4 and 7.3.5 also support this conclusion. The (255,k) class is comprised of 8-bit symbols and the (15,k) class of 4-bit symbols. Whether 1 bit or more than 1 bit in a symbol is in error makes no difference, because the entire symbol will be in error in either case. Since the (255,k) class has more bits per symbol, it can support a higher bit error rate. Therefore, one can conclude that the (255,k) class of codes yields a higher throughput than the (15,k).

From Figures 8.2.6 - 8.2.9 one can deduce that convolutional codes with the higher constraint length have better performance, for a given rate, R. In every case, the code with the higher constraint length yields a higher throughput at an equivalent bit error rate. The total number of possible bit combinations is dependant on the number of memory units, one less than the constraint length. As the number of combinations increases so does the distance between each correct codeword. Therefore, as the constraint length is increased, the performance of the code is better. However, one should also note that the complexity of the code increases as 2^{L-1}, and hence the superior codes are also far more complex.

In order to determine the best code with respect to the throughput criterion, the Reed-Solomon and convolutional codes must be compared. Figure 9.1.1 and 9.1.2 show the throughput vs. input bit error rate for codes of rate approximately 1/2 and 3/4, respectively. Figure 9.1.1 shows the (255,119) and (15,7) RS codes and convolutional codes with L=6 and 9. The (255,119) Reed-Solomon code has the higher throughput at a given BER where retransmissions dominate. At lower bit error rates, the fact that the convolutional code is a rate 0.500 code and, the RS is a rate 0.467 code comes into play. A rate 0.500 RS code would perform more closely to the L=9 convolutional code. However, Figure 9.1.3 demonstrates that the RS code is a superior error correcting code. If the bit rate and the round-trip delay were increased, the (255,119) code would have a definite advantage over the convolutional codes. Figure 9.1.2 shows that the RS (255,187) code has an even greater advantage than the rate 3/4, L=9 convolutional code. The difference in performance shows that almost an order of magnitude increase in input BER is required to yield the same throughput. Even the RS (15,11) code has a decided edge over the convolutional code. In this case, the RS codes have a rate of 0.733 compared to 0.750 for the convolutional codes. This difference of less than 3% is almost negligible compared to the difference in performance. Figure 9.1.4 confirms the power of the RS codes. One should also note that Figure 8.2.1 shows that performance for convolutional codes is slightly worse than the bound in the figures. Because the Reed-Solomon code is a superior error correcting code and has equal or greater performance in the throughput plots, the (255,119) Reed-Solomon code is chosen to be the best for throughput.

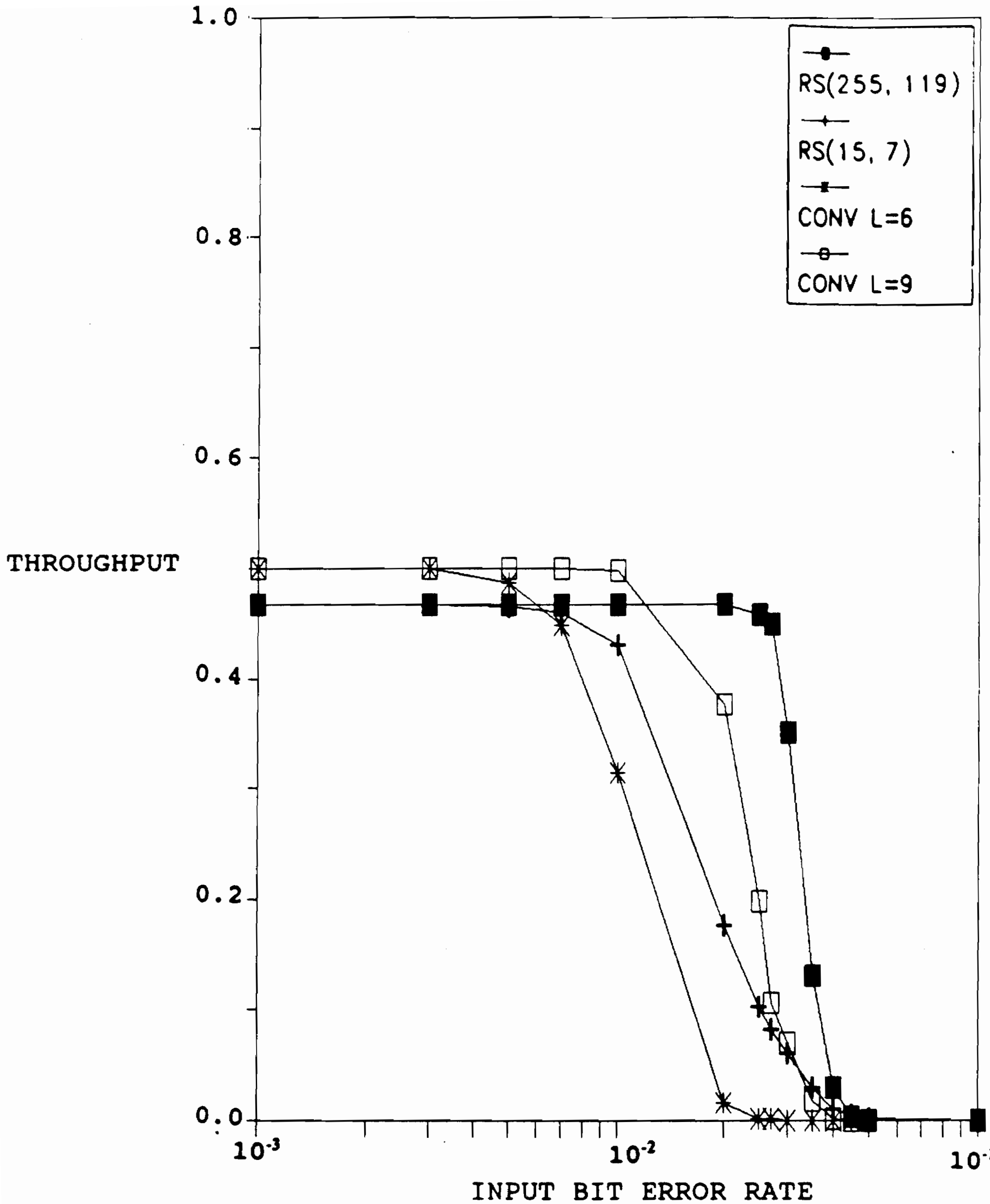


Figure 9.1.1 Throughput vs. input bit error rate for approximately rate 1/2 codes, including the (255,119) and (15,7) Reed-Solomon and the R=1/2, L=6 and 9 convolutional codes.

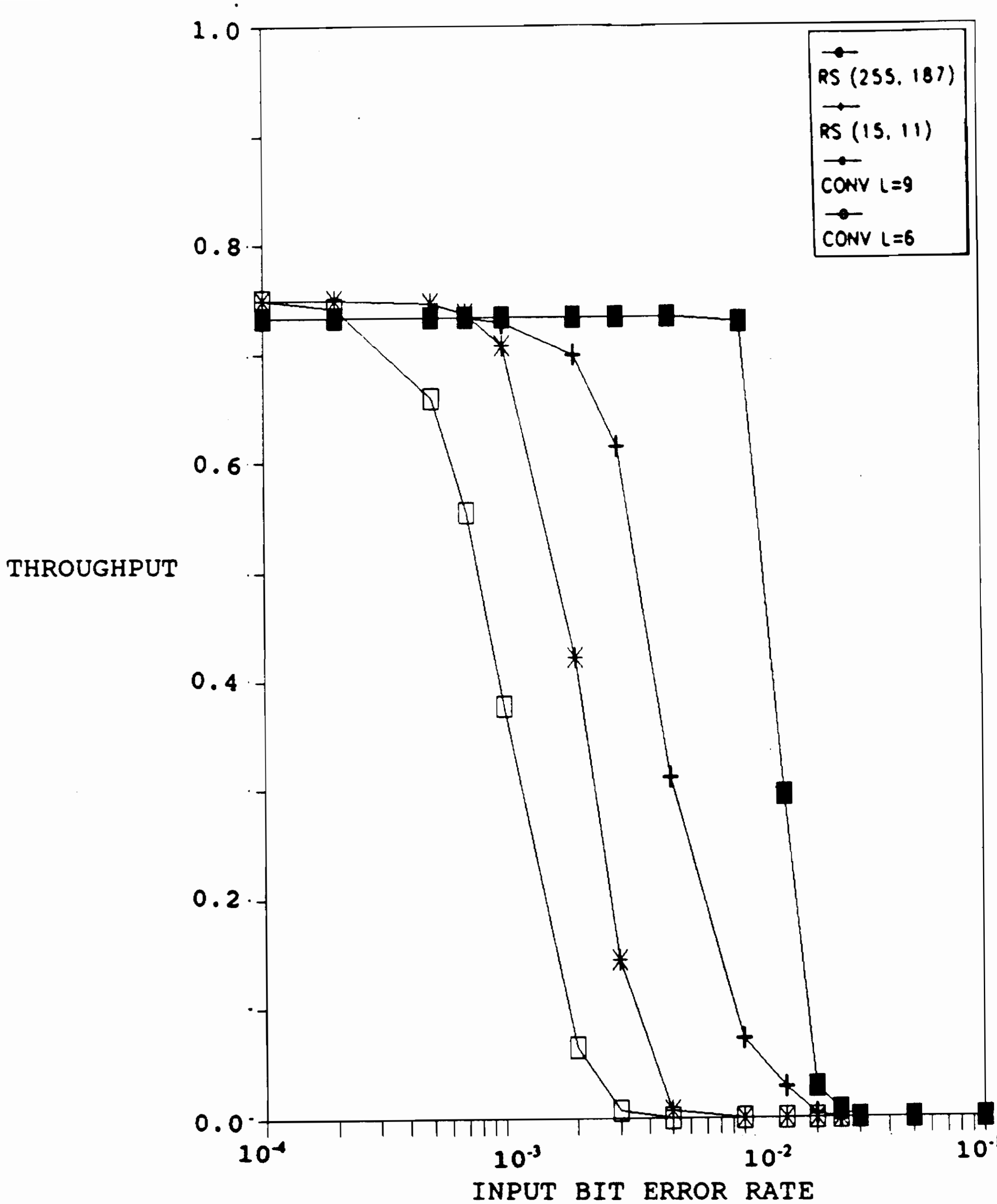


Figure 9.1.2 Throughput vs. input bit error rate for approximately rate 3/4 codes, including the (255,187) and (15,11) Reed-Solomon and the R=3/4, L=6 and 9 convolutional codes.

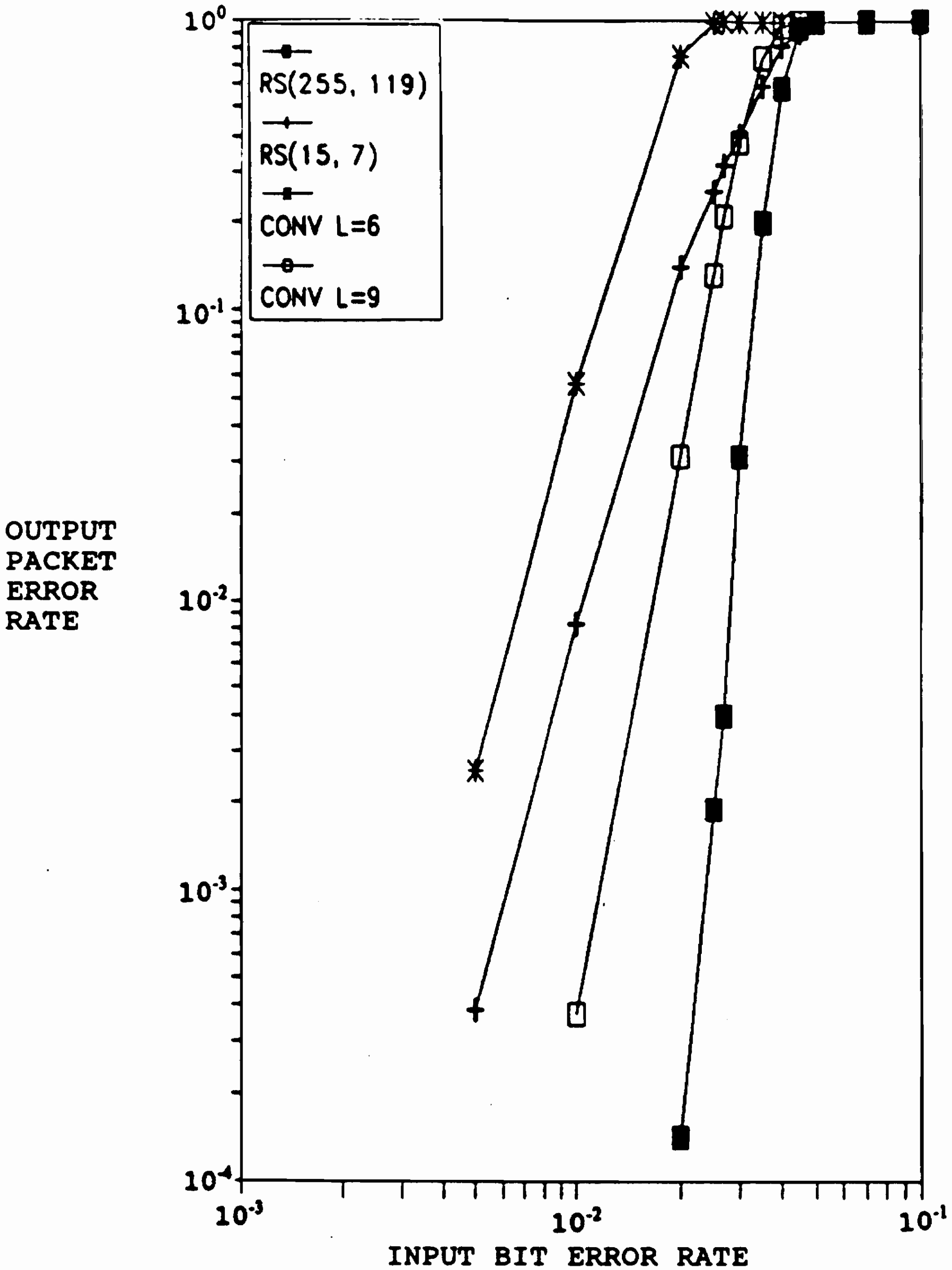


Figure 9.1.3 Output packet error rate vs. input bit error rate for approximately R=1/2 codes.

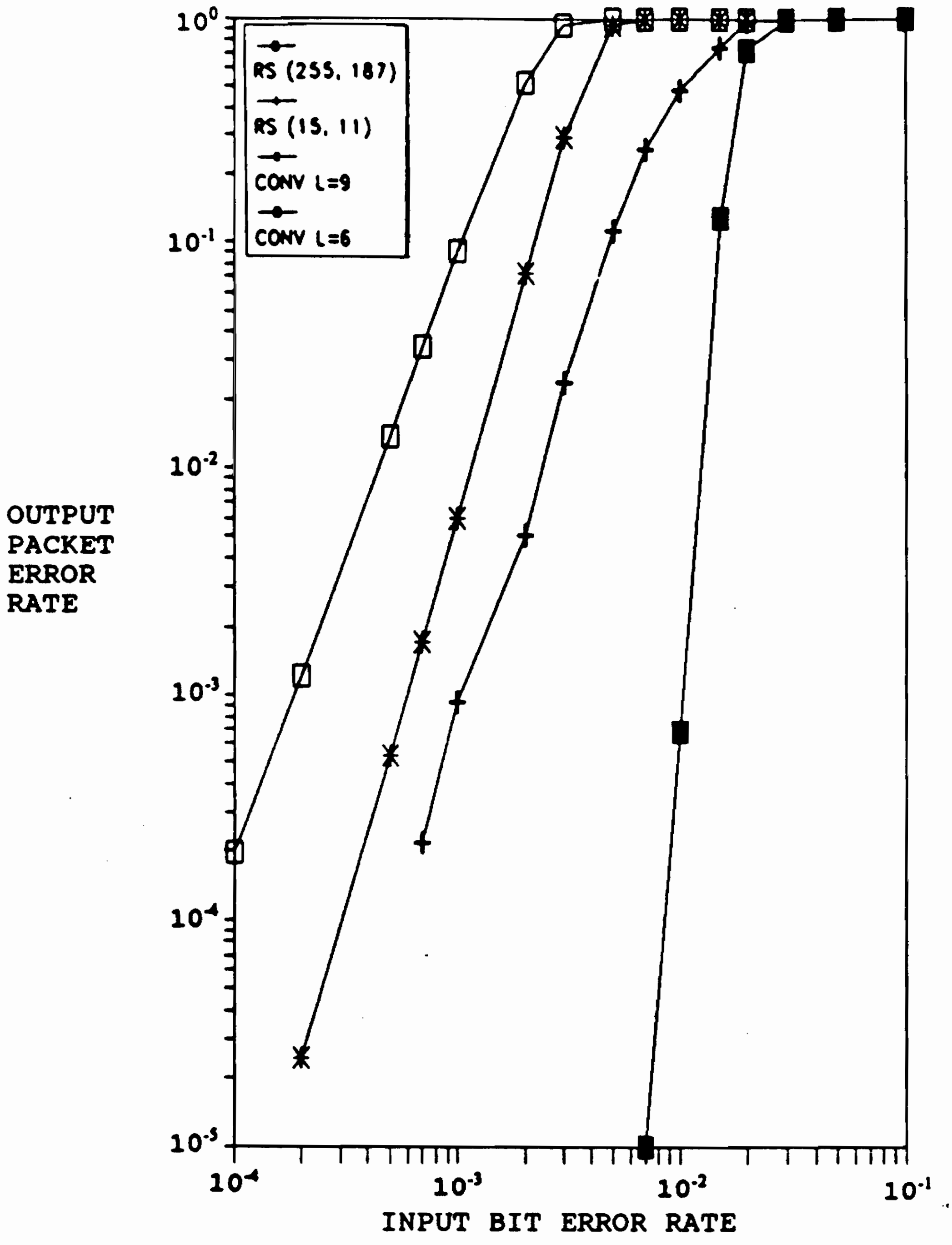


Figure 9.1.4 Output packet error rate vs. input bit error rate for approximately R=3/4 codes.

9.2 Coding Simplicity Criterion

Another important evaluation criterion is the simplicity of the encoding and decoding processes. If a code is difficult to implement, it is of little practical use. The complexity of the encoding process for all the codes is negligible when compared to the decoding process. Therefore, the decoding procedures will be examined in more detail. Table 9.2.1 contains the approximate number of operations for decoding the following codes. These values assume that t errors have occurred during transmission.

The (7,4) Hamming code is the easiest to code. It is also the simplest of the codes examined. The encoding is performed using a simple table look-up system in encoder memory. For the systematic code, three bits are simply appended to each set of four bits entered into the encoder. The decoding process is accomplished by adding certain bits of the received codeword and using table look-up again to determine the word which was most likely received. This process requires 20 addition operations and 1 memory access per codeword.

The coding procedure for the (23,12) Golay code is somewhat more complicated. The encoding is performed by a linear feedback shift register. The bits are simply shifted into the register and the codeword is shifted out. The decoding process is somewhat more complex. However, it still only requires the use of an LFSR and a complex series of tests at various taps. In order to decode 1 codeword, 46 shifts of 50 logic operations per shift are required.

The coding of the Reed-Solomon codes is even more complex. The encoding is performed in a similar fashion to the Golay code. An LFSR is used to form the proper codewords, but this LFSR is more complex since the operation is performed over multi-bit symbols and not individual bits. In order to calculate each of the 2t syndromes, 8 additions and 8 multiplications are required. Finding the error locator polynomial requires $2t^2$ additions and $2t^2$ multiplications. Solving for the roots of the error locator polynomial requires nt additions and nt multiplications. Calculating the values of the t errors requires $3t^3/2$ multiplications and $t^2 + t^3/2$ additions.[42] Note that all of these operations are over the Galois field of 256 elements. Decoders of this type have been made which have bit rates of over 1 Mb/s.[43]

Convolutional codes also have complex coding systems. The encoder is one of the more simple to implement. It is simply a switched circuit with several exclusive-or gates and memory elements. However, the complexity increases greatly with the constraint length, L, as more connections and memory units are required. The decoding process is much more complex. For each state, the metrics for the last 5L bits must be accessed from memory, a new metric must be calculated, and 5L metrics must be stored in memory. The number of states is 2^{L-1}, and this process must be performed for each information bit. The path with the largest metric, the maximum likelihood sequence, must also be kept in storage until the packet is decoded.[44] The hardware to access memory locations also adds to the complexity of the decoder. The decoding speed can be improved by using 2^{L-1} parallel processors, but the complexity remains the same. Hardware for Viterbi decoders is offered by companies such as Qualcomm and Stanford Telecom and can operate at over 50 Mb/s.

In summary, the Hamming coding is the simplest to implement followed by the Golay, Reed-Solomon and the convolutional codes. The Reed-Solomon decoding is computationally more complex than the convolutional decoding, but due to the immense number of memory accesses, convolutional decoders are the most difficult to implement. Note that Table 9.2.1 is based on the R=1/2, L=9 convolutional code, which is comparable to the RS (255,119), and the number of decoding operations could be greatly reduced if L were decreased. Convolutional decoders can perform decoding more quickly than RS decoders, if 2^{L-1}

processors are employed. Both RS and convolutional coders have been successfully implemented in hardware.

Table 9.2.1 The number of decoding operations can be used as a measure of decoder complexity. The RS code is computationally complex, and the convolutional code requires large amounts of memory which requires additional complexity.

CODE TYPE	OPERATIONS PER CODEWORD	CODEWORDS PER PACKET	TOTAL NUMBER OF OPERATIONS
(7,4) Hamming	20 Additions	292	5840
	1 Memory Access		292
(23,12) Golay	2300 Logic	89	204,700
(255,119) RS	$t^3/2+3t^2+(16+n)t$ Adds.	1	189,516
	$3t^3/2+2t^2+(16+n)t$ Mults.	1	499,324
R=1/2 L=9 Conv.	2^{L-1} Mem. Acc. (5L Metrics)	1	11,750,400
	2^{L-1} Metric Calculations	1	261,120
	2^{L-1} Mem. Acc. (5L Metrics)	1	11,750,400

9.3 Burst Error Criterion

Burst errors cause a great deal of retransmissions to occur because of the number of consecutive bits which are corrupted. Transmissions in the range of several hundred MHz are subject to noise bursts due to man-made noise sources. In order to combat this noise, a good code should be able to correct long bursts of corrupted bits.

The evaluation of this criterion is simpler than the others because the number of consecutive bits which is guaranteed to be corrected can be calculated exactly. Table 9.3.1 gives the burst correcting capability of all the codes previously examined and whether or not interleaving is required. The Reed-Solomon codes are by far the best in this respect. The more powerful RS codes can correct more than twice the number of consecutive bit errors than the best R=1/2 convolutional codes. The (15,k) codes have slightly better performance than the corresponding (255,k) codes. However, one should note than the (255,k) class is the only one which does not require interleaving, thus reducing coding complexity greatly. For this criterion, the (255,k) Reed-Solomon codes are by far superior.

Table 9.3.1 Maximum number of consecutive bits which are guaranteed to be corrected by various codes.

CODE TYPE	CORRECTABLE BURST LENGTH	INTERLEAVING REQUIRED
(7,4) Hamming	292	YES
(23,12) Golay	267	YES
(15,13) RS	133	YES
(15,11) RS	269	YES
(15,7) RS	541	YES
(255,221) RS	129	NO
(255,187) RS	265	NO
(255,119) RS	537	NO
R=3/4 L=6 CONV	136	YES
R=3/4 L=7 CONV	116	YES
R=3/4 L=8 CONV	102	YES
R=3/4 L=9 CONV	90	YES
R=2/3 L=6 CONV	136	YES
R=2/3 L=7 CONV	116	YES
R=2/3 L=8 CONV	153	YES
R=2/3 L=9 CONV	135	YES
R=1/2 L=6 CONV	204	YES
R=1/2 L=7 CONV	232	YES
R=1/2 L=8 CONV	204	YES
R=1/2 L=9 CONV	225	YES
R=1/3 L=5 CONV	405	YES
R=1/3 L=6 CONV	408	YES
R=1/3 L=7 CONV	348	YES
R=1/3 L=8 CONV	357	YES

9.4 Evaluation

Now that all the criteria have been evaluated, an error control code must be chosen. The Hamming and Golay codes perform too poorly on the throughput criterion to be chosen as the best codes. Therefore, the Reed-Solomon and convolutional codes are the best.

The high constraint length convolutional codes and the Reed-Solomon codes have similar performance on the throughput criterion. The RS codes perform better at higher rates and are more powerful and, therefore, would perform better over a more challenging channel. The coding simplicity criterion also favors the RS codes due to the fact that convolutional codes require a large number of storage elements to store partial metrics and the hardware to access them. The burst error criterion decidedly goes to the RS codes because they can correct more than twice the number of errors without using interleaving. Therefore, the Reed-Solomon codes are the best choice if evaluated by the above criteria.

BLANK PAGE

(NO TEXT MISSING)

CHAPTER 10

IMPLEMENTATION

This chapter will discuss how error control coding could be implemented into a packet system. The order in which the encoding/decoding process is performed is of particular importance. This order also affects the packet assembly.

10.1 System Requirements

Three major areas must be addressed to insure the proper reception of data. The system must be able to receive the packet, attempt to correct errors, and detect errors and request retransmission if errors are not correctable. In order to determine when to begin reception of the packet, a flag must be transmitted. The correction of errors is handled by the Reed-Solomon error correcting code. If these are not correctable, an error detection code will be used to request packet retransmission.

10.2 System Overview

The system must be developed in order to facilitate the above requirements. The first is the requirement of the flag.

10.2.1 The Flag

Since the flag is what tells the system to begin receiving, it is one of the most important pieces of information. Also since it is the first byte received, it cannot be encoded by the same error control code as the rest of the packet. The problem which arises is that at high bit error rates the probability that the flag is received incorrectly is very large. At a BER of 10^{-1}, the probability that the flag is received correctly is only 0.43.

The probability of an 8 bit flag being in error can be found by:

$$P_{fe}=1-(1-p)^{8} \qquad (10.2.1.1)$$

where, P_{fe}= the probability of a flag error.

p = the bit error rate.

One should note another potential problem. The flag symbol can also be found in the data stream. This additional problem will be a factor in determining what type of flag configuration to use. Therefore, in order to insure that the packet is not discarded, the opening flag must not be in error, the flag must not appear in the data stream, and the remainder of the packet must not be in error.

The first solution is reducing the length of the flag symbol. If the length of the flag were reduced from 8 bits to 4 bits the probability of correct reception would be 0.66 at a BER of 10^{-1}. There are two drawbacks to this solution. The first is that there is an increased probability that this flag could be artificially reproduced by noise when in fact no signal is present. The second is an increased probability of the flag appearing in the data stream. This shortened flag is 32 times more likely to appear in the data as the 8 bit flag. One should recall that in order to prevent a flag from appearing in the data, bit stuffing is used. The decoding section of the bit stuffing routine must recognize when a bit has been stuffed. The complexity of this routine would increase greatly if a shortened flag were used. This increase would also slow the processing of packets. The expanded use of bit stuffing would also reduce the throughput of the system.

A second solution involves the use of a short error correcting code, which would include only the flag. The decoding of this code must take place before the first bit of the packet arrives, or buffers must be used to store the packet while decoding of the flag occurs. A convolutional code or short BCH code would be a good choice.

Because the flag is only 8 bits in length, a convolutional code would be a poor choice. The convolutional codes achieve maximum effectiveness only when their decoding approaches that of maximum likelihood. This requires the encoded word to be approximately five times the constraint length, L.[45] Their Hamming distance with short words is only a fraction of their free distance, d_{free}.

The (7,4) Hamming code is one of the most efficient of the short BCH codes. It can correct 1 error with the use of only 3 redundant bits. Therefore, it will be used in evaluating the suitability of this solution. Figure 10.2.1.1 is a graph of the probability of the flag being in error when encoded with a (7,4) Hamming code, the probability of the remainder of the packet being in error using a RS (255,119) code, and their probabilistic sum, which is the probability of the entire packet being in error.

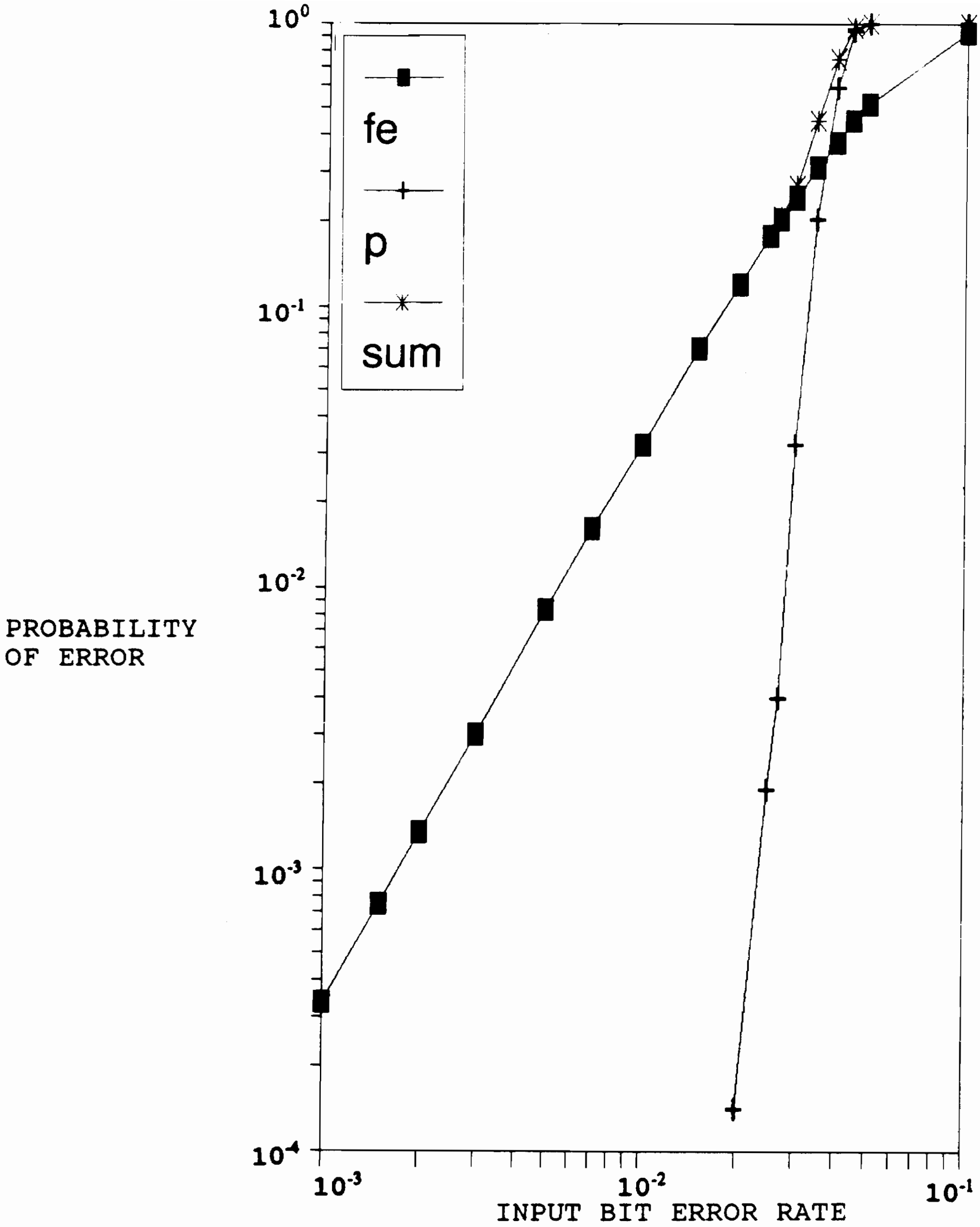


Figure 10.2.1.1 The probability of the flag being in error when encoded with a (7,4) Hamming code, (7,4); the probability that the balance of the packet is in error when encoded with an RS (255,119) code, p; and the probability of the packet being rejected, sum.

The third solution is to add multiple flags to the beginning of the packet. The probability of one of two flags being received correctly is 0.68 at a BER of 10^{-1}. This solution has some of the same drawbacks as the other but to a lesser degree. In order to determine whether the flag received correctly was the first or the second, the flags cannot be identical. Therefore, the chance of a flag being found in the bit stream is doubled. Recall that in the previous solution the flag was 32 times more likely to appear. Due to the fact that more than one flag is needed, the throughput also suffers slightly. More than two flags can be used if throughput at high bit error rates is a great priority.

The probability of a flag being received without error and j flags were sent is given by:

$$P_{fe,j} = (P_{fe})^j = (1-(1-p)^8)^j \qquad (10.2.1.2)$$

where, $P_{fe,j}$= the probability that at least one of j flags are received without error.

P_{fe} = the probability that a flag is received correctly.

j = the number of flags preceding the packet.

p = the bit error rate.

Since the receiver is looking for flags at all times, it is possible that errors during transmission have caused a flag to be found in the data stream. One should note that it is possible to stuff a single bit is such a way as to require 2 bits to be in error in order produce a flag symbol. A sequence that is changed to a flag symbol which will result in a discarded packet. The probability that bit errors will cause a flag will appear in the data stream before error correction is approximated by:

$$P_{fid} = 1-(1-0.003418*j*p^2)^{255} \qquad (10.2.1.3)$$

One can see that there is a trade-off between the number of flags preceding a packet and the probability that the flag will be found in the packet. The probabilistic sum, P_{sum}, of $P_{fe,j}$, P_{fid}, and P_p is the probability of the packet being discarded due to an error.

$$P_{sum} = 1-(1-P_{fe,j})*(1-P_{fid})*(1-P_p) \qquad (10.2.1.4)$$

Figures 10.2.1.3-5 are plots of $P_{fe,j}$, P_{fid}, P_p and P_{sum} for 1 flag and the (255,221), (255,187), and (255,119) RS codes, respectively. Figure 10.2.1.6 is a comparison of the sums from the 3 different codes and Figure 10.2.1.7 is a graph of these probabilities translated to throughput. Figures 10.2.1.8-12 are plots of the same probabilities for the use of 2 flags, and Figures 10.2.1.13-17 are for 3 flags. Figure 10.2.1.18 is a comparison of the throughput for the (255,119) code with 1,2 and 3 flags being used.

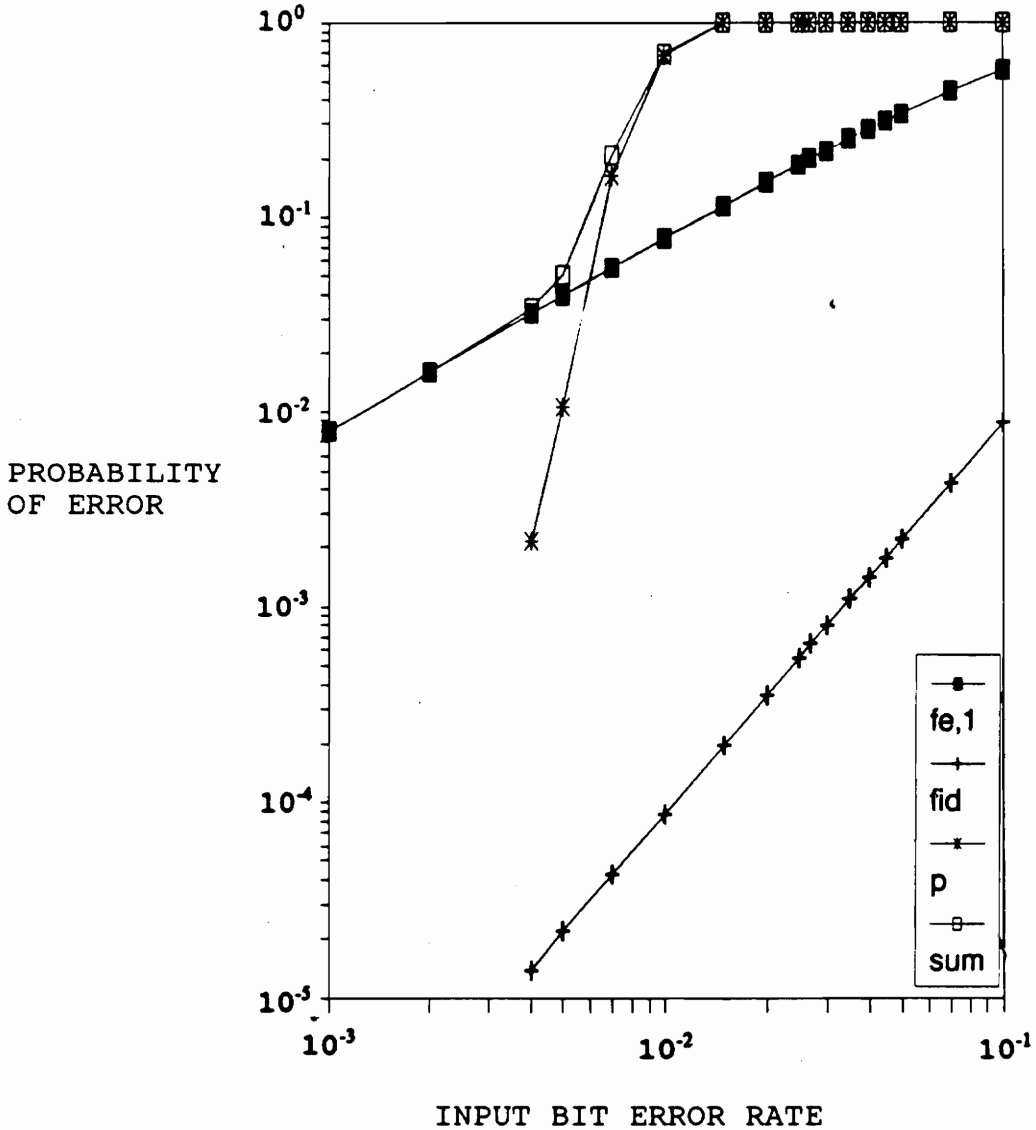


Figure 10.2.1.3 The probabilities of the opening flag being in error $P_{fe,1}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,221) RS code and a single flag being used.

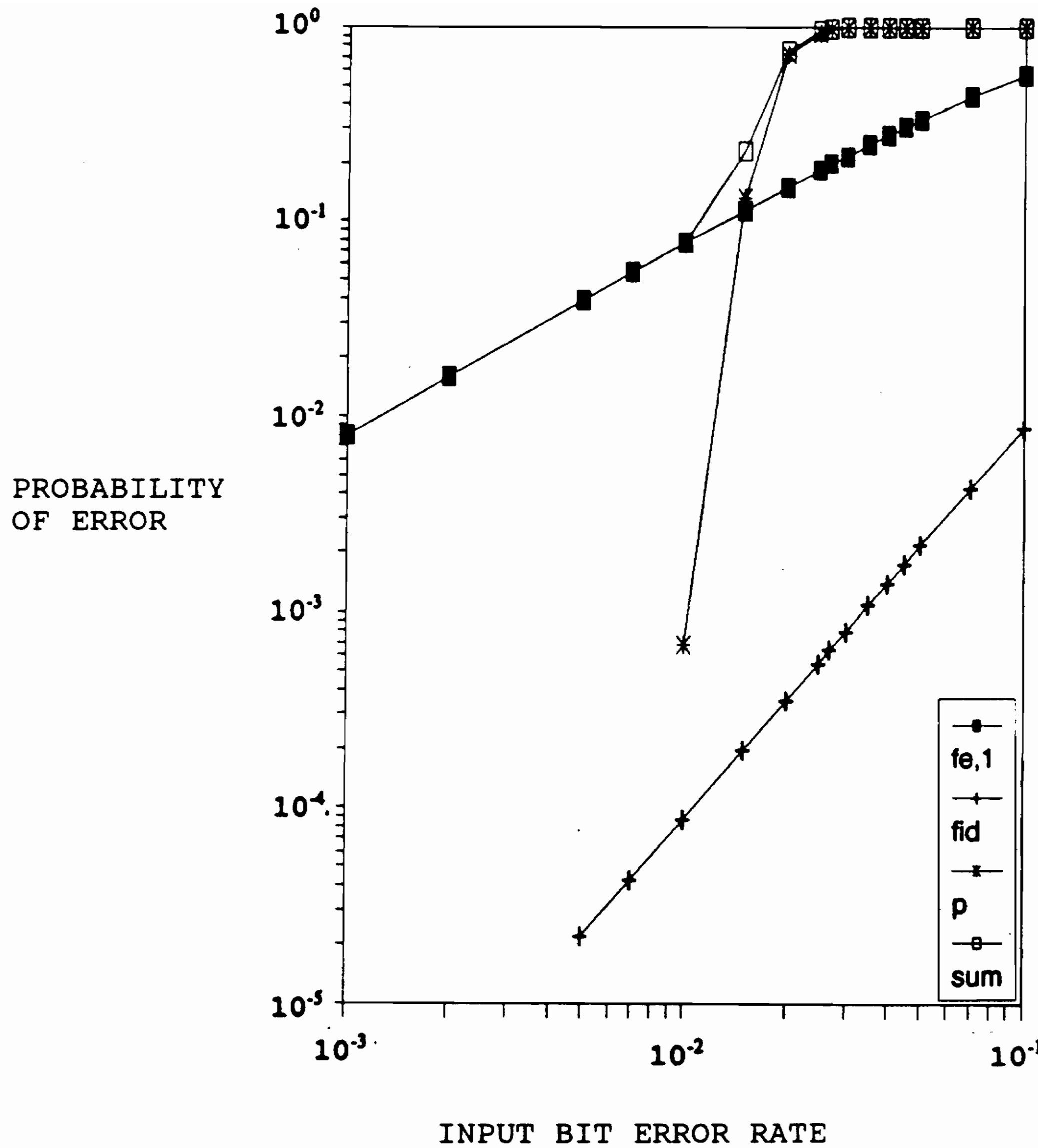


Figure 10.2.1.4 The probabilities of the opening flag being in error $P_{fe,1}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,187) RS code and a single flag being used.

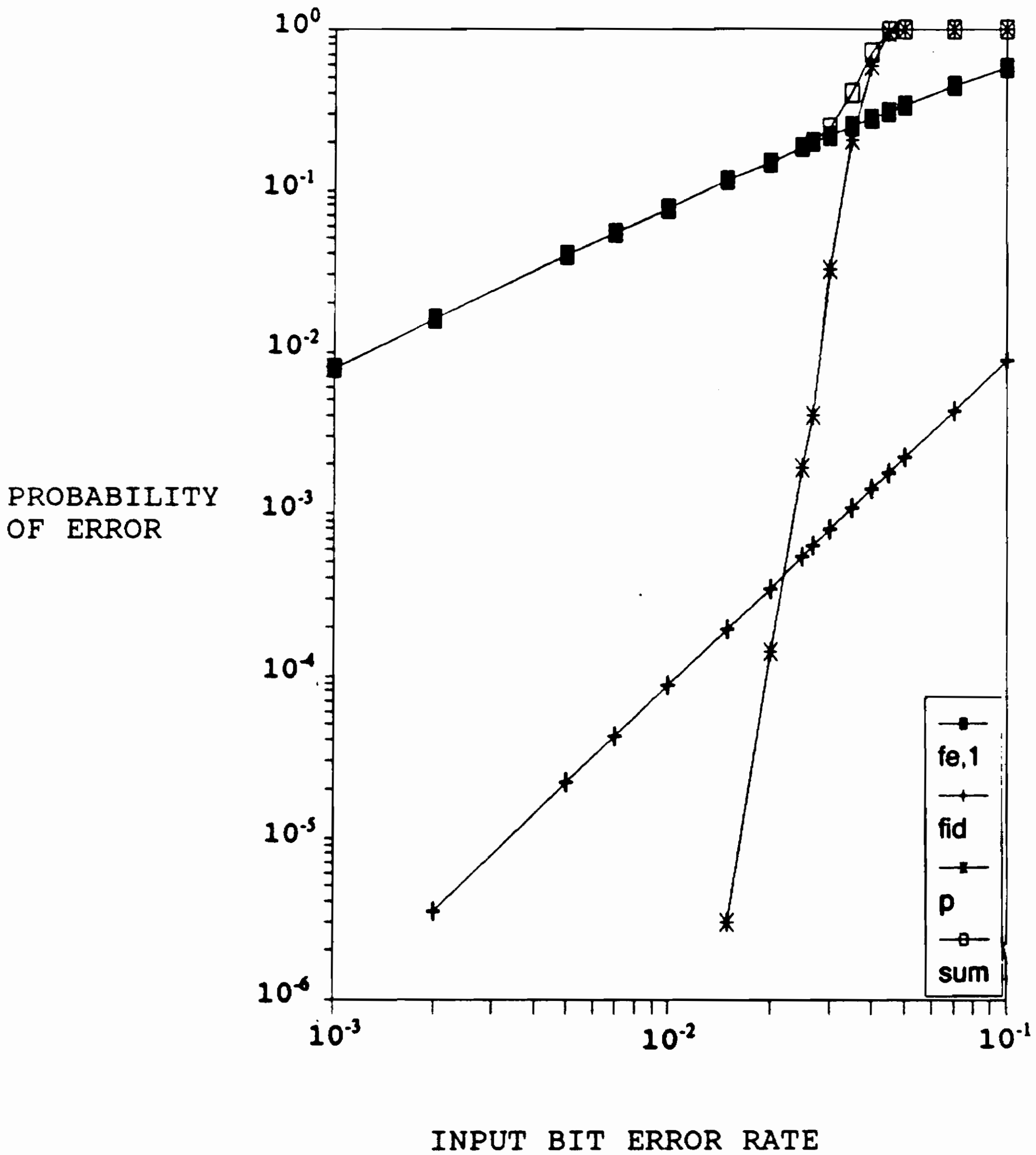


Figure 10.2.1.5 The probabilities of the opening flag being in error $P_{fe,1}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,119) RS code and a single flag being used.

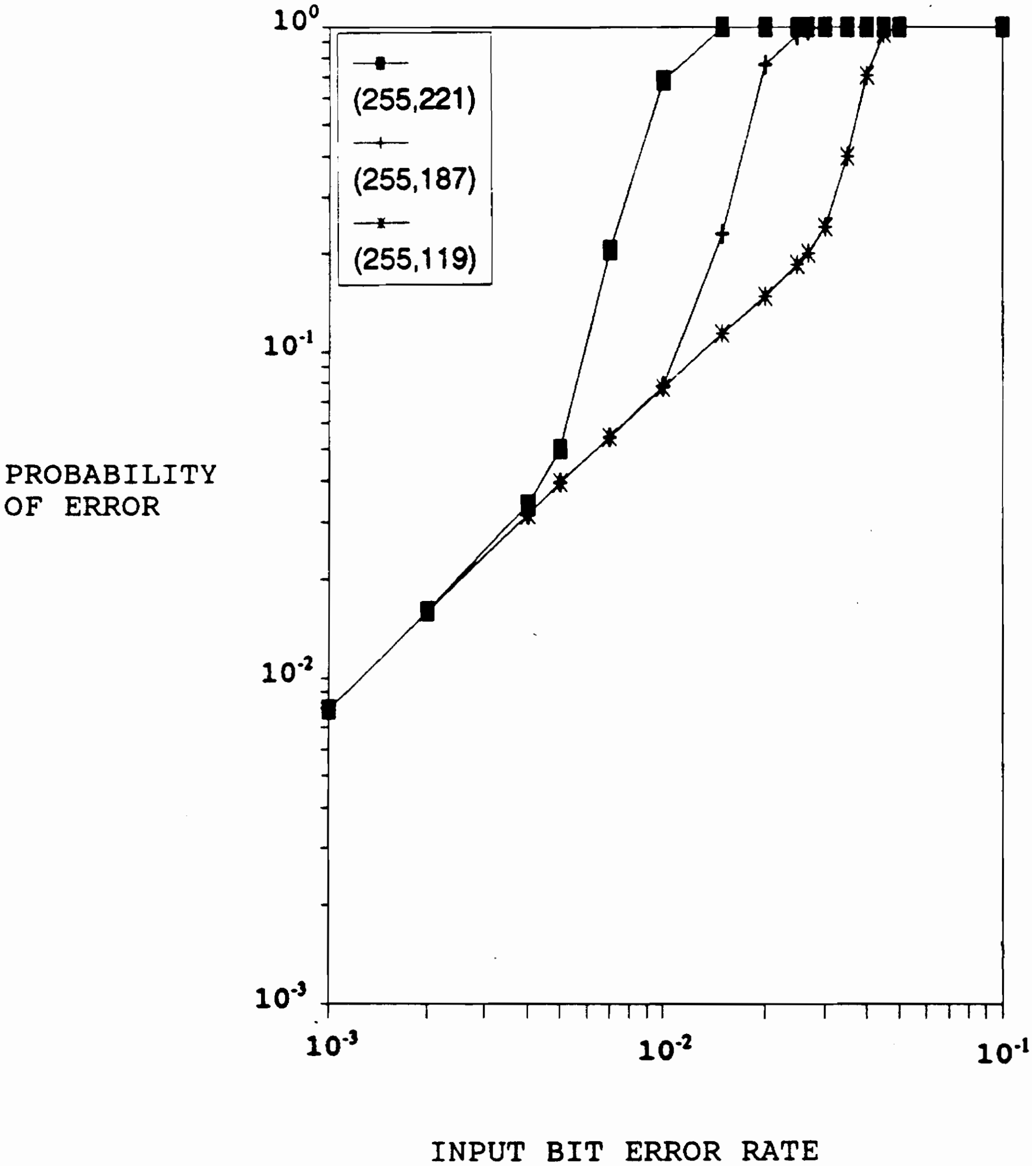


Figure 10.2.1.6 The probability P_{sum} for the (255,221), (255,187), and (255,119) RS codes, using a single flag.

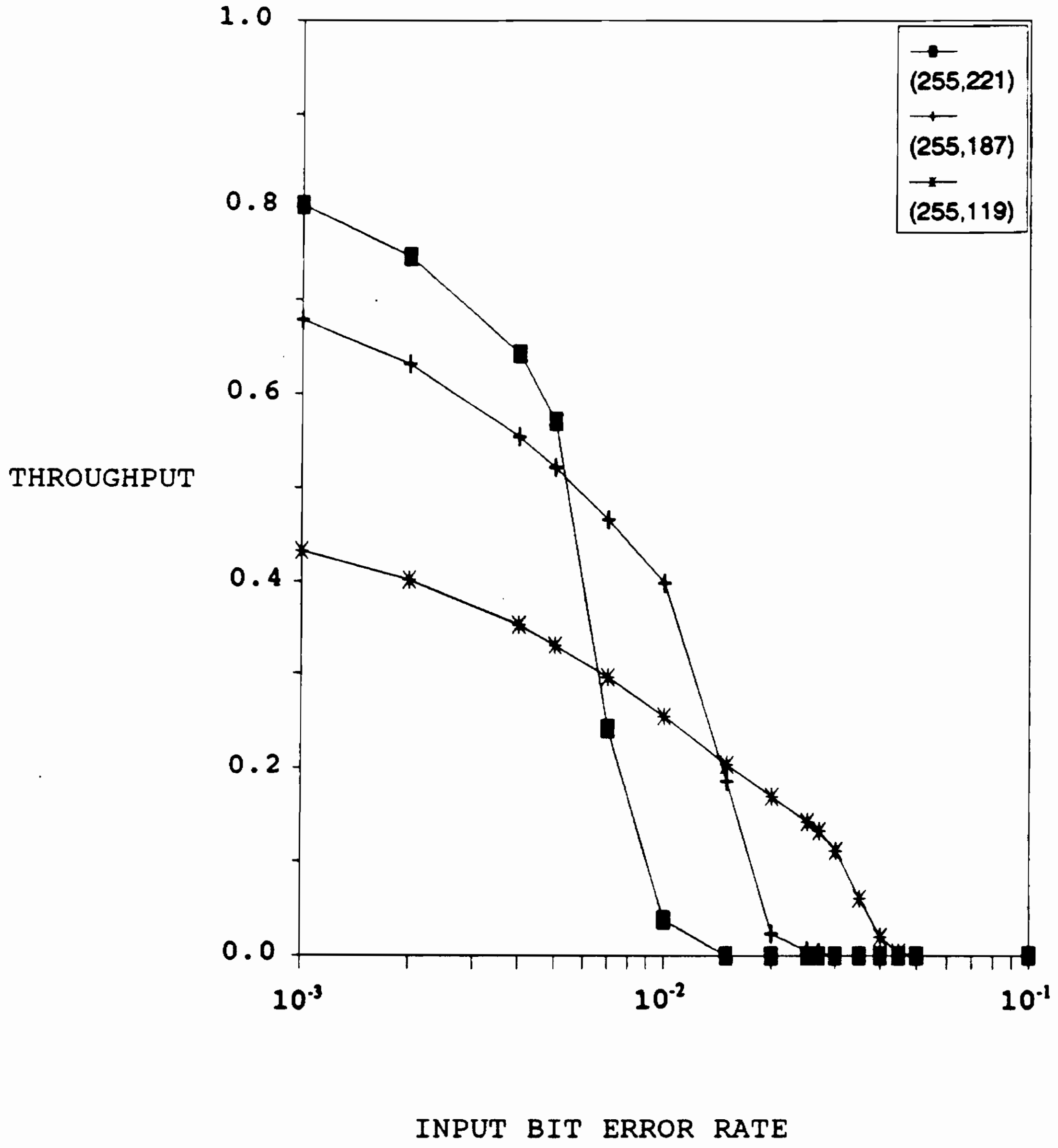


Figure 10.2.1.7 The throughput of a go-back-10 ARQ system which would be achieved using the probabilities in Figure 10.2.1.6 with a single flag.

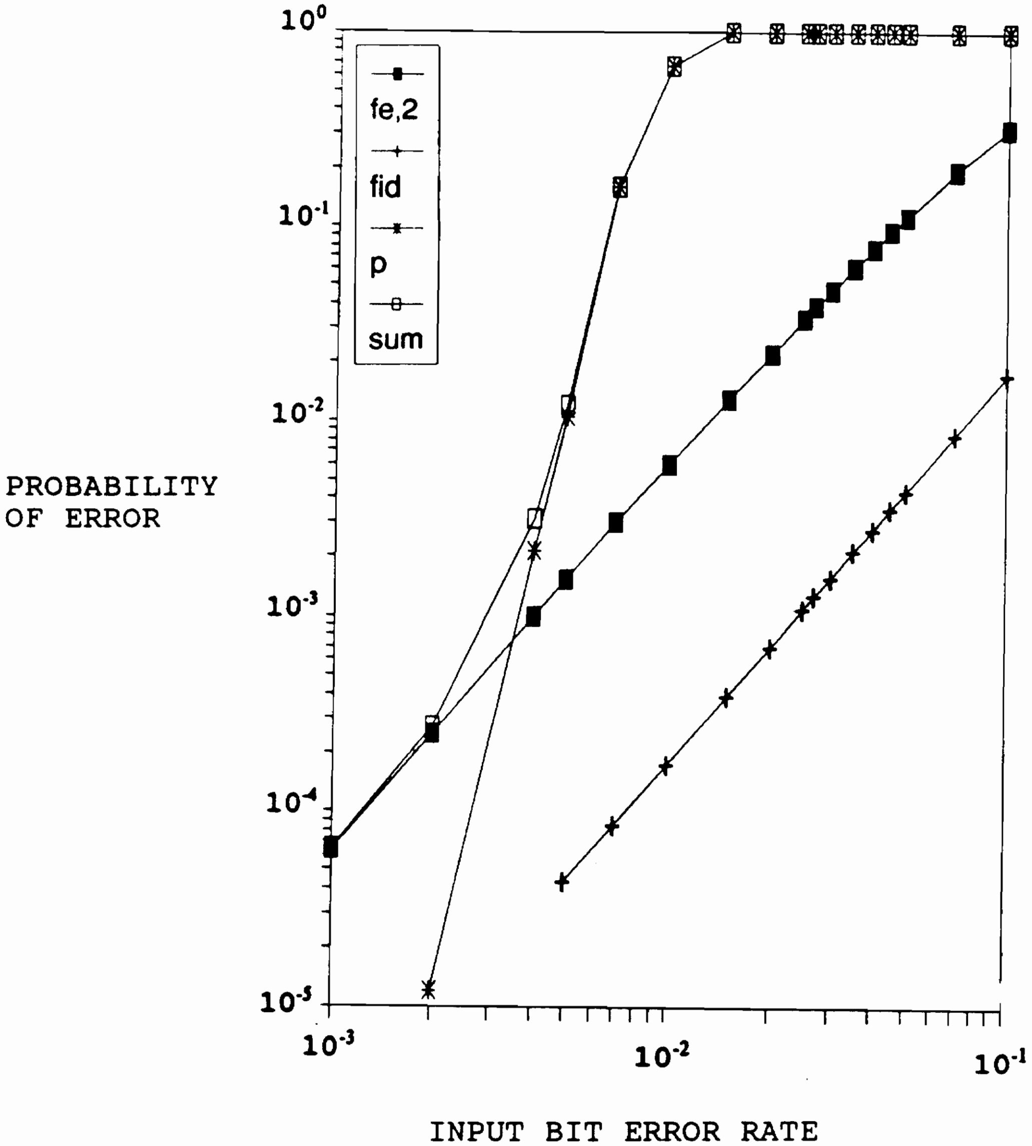


Figure 10.2.1.8 The probabilities of the opening flag being in error $P_{fe,2}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,221) RS code and two flags being used.

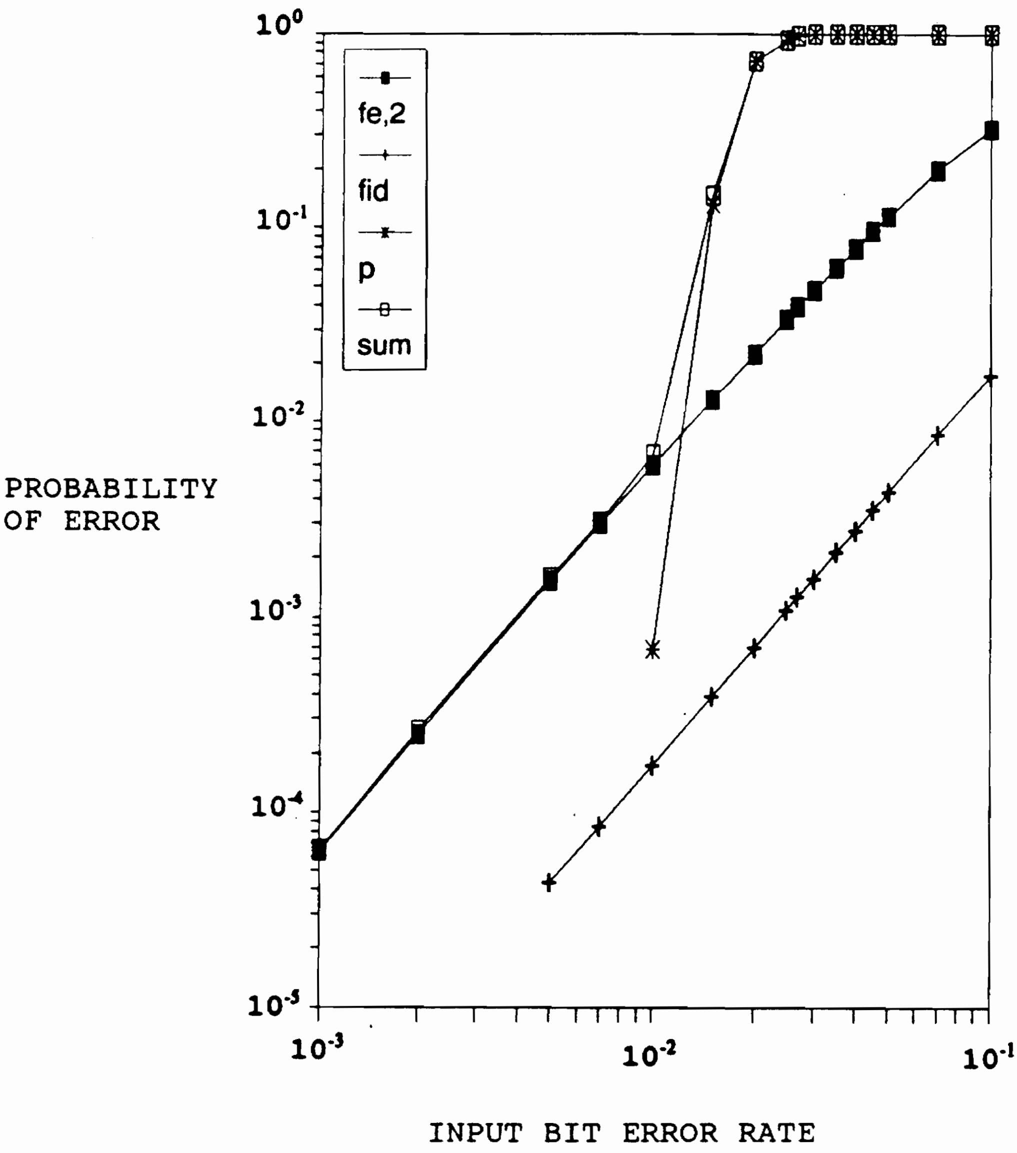


Figure 10.2.1.9 The probabilities of the opening flag being in error $P_{fe,2}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,187) RS code and two flags being used.

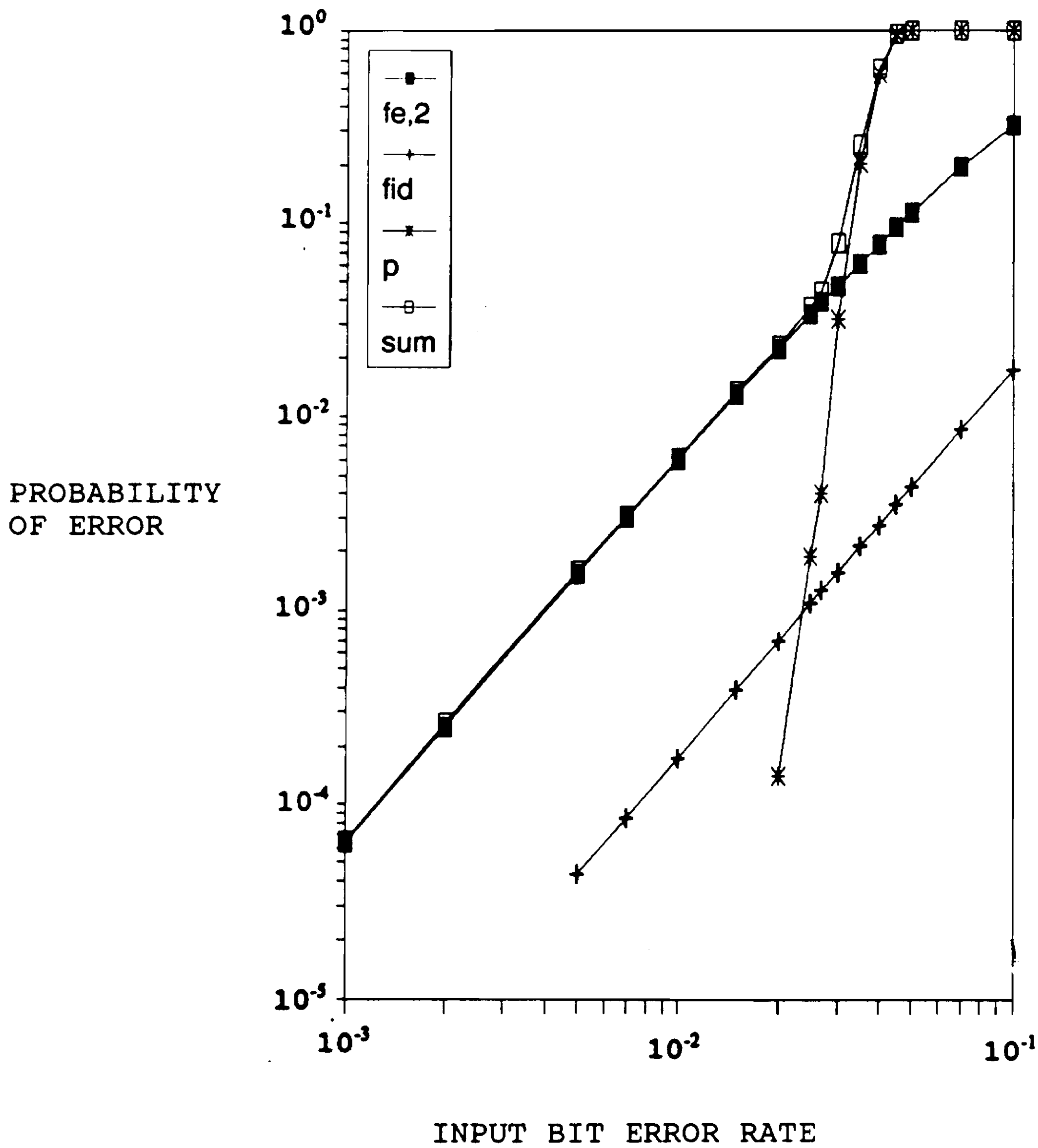


Figure 10.2.1.10 The probabilities of the opening flag being in error $P_{fe,2}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,119) RS code and two flags being used.

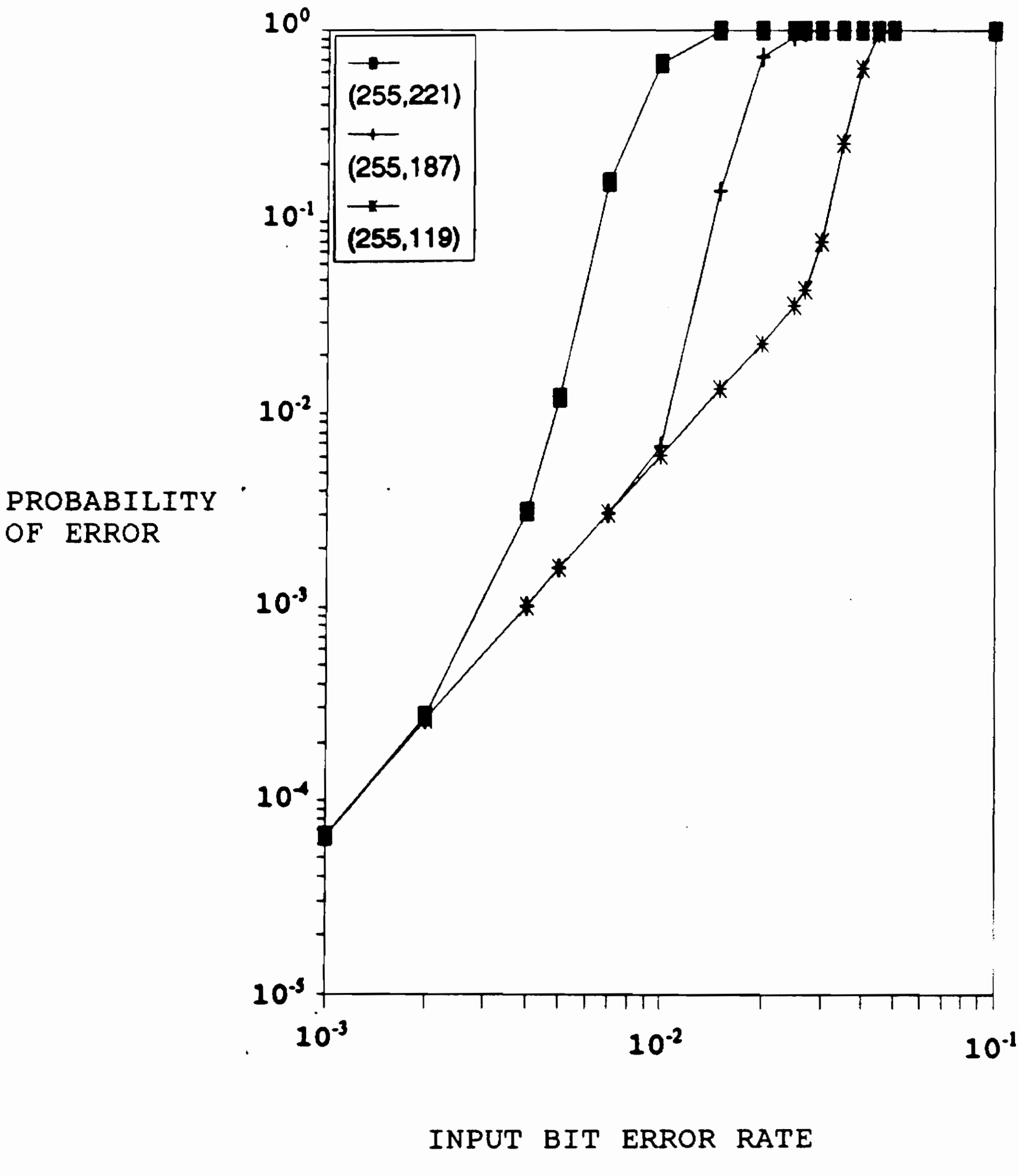


Figure 10.2.1.11 The probability P_{sum} for the (255,221), (255,187), and (255,119) RS codes, using two flags.

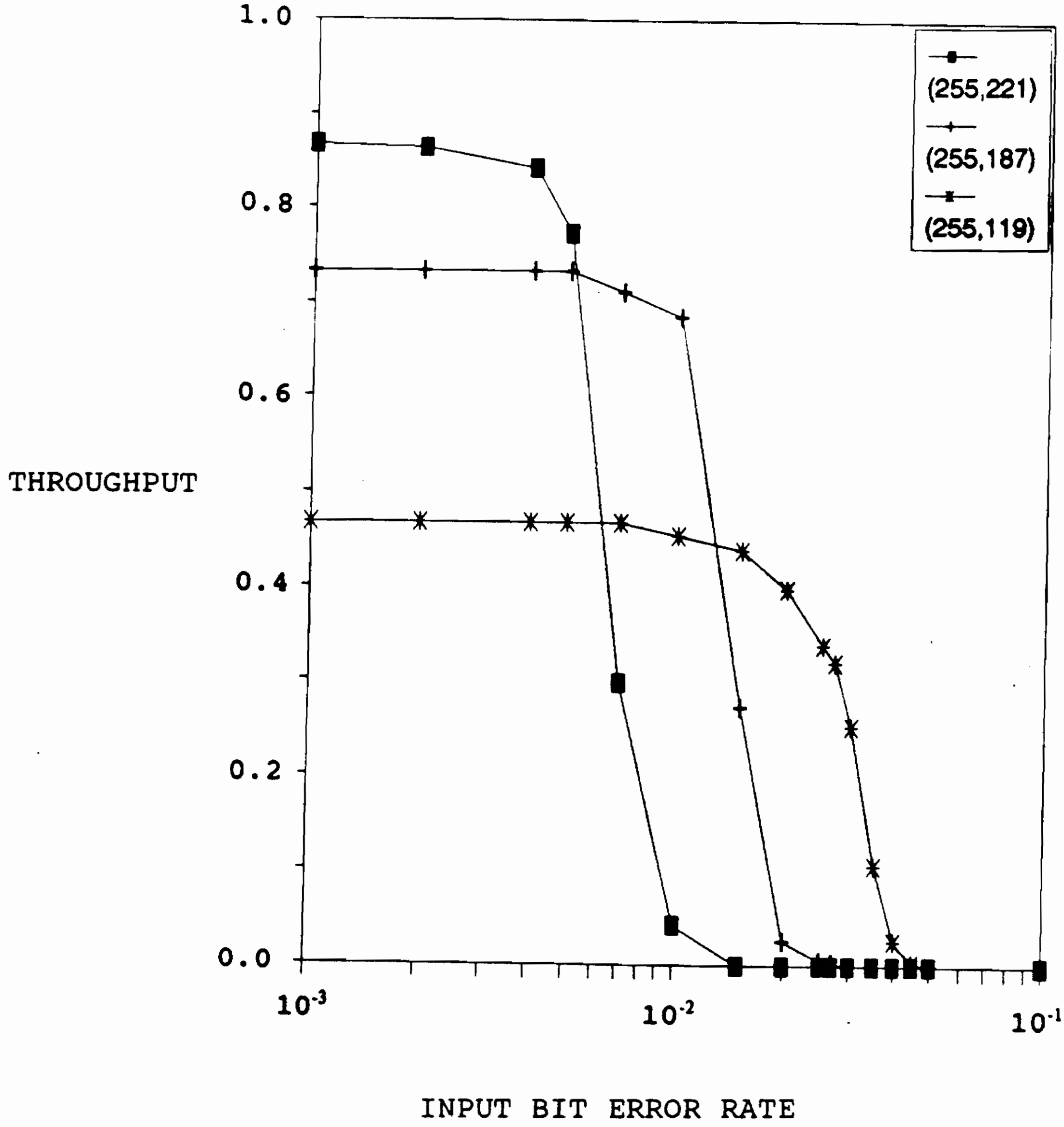


Figure 10.2.1.12 The throughput of a go-back-10 ARQ system which would be achieved using the probabilities in Figure 10.2.1.11 with two flags.

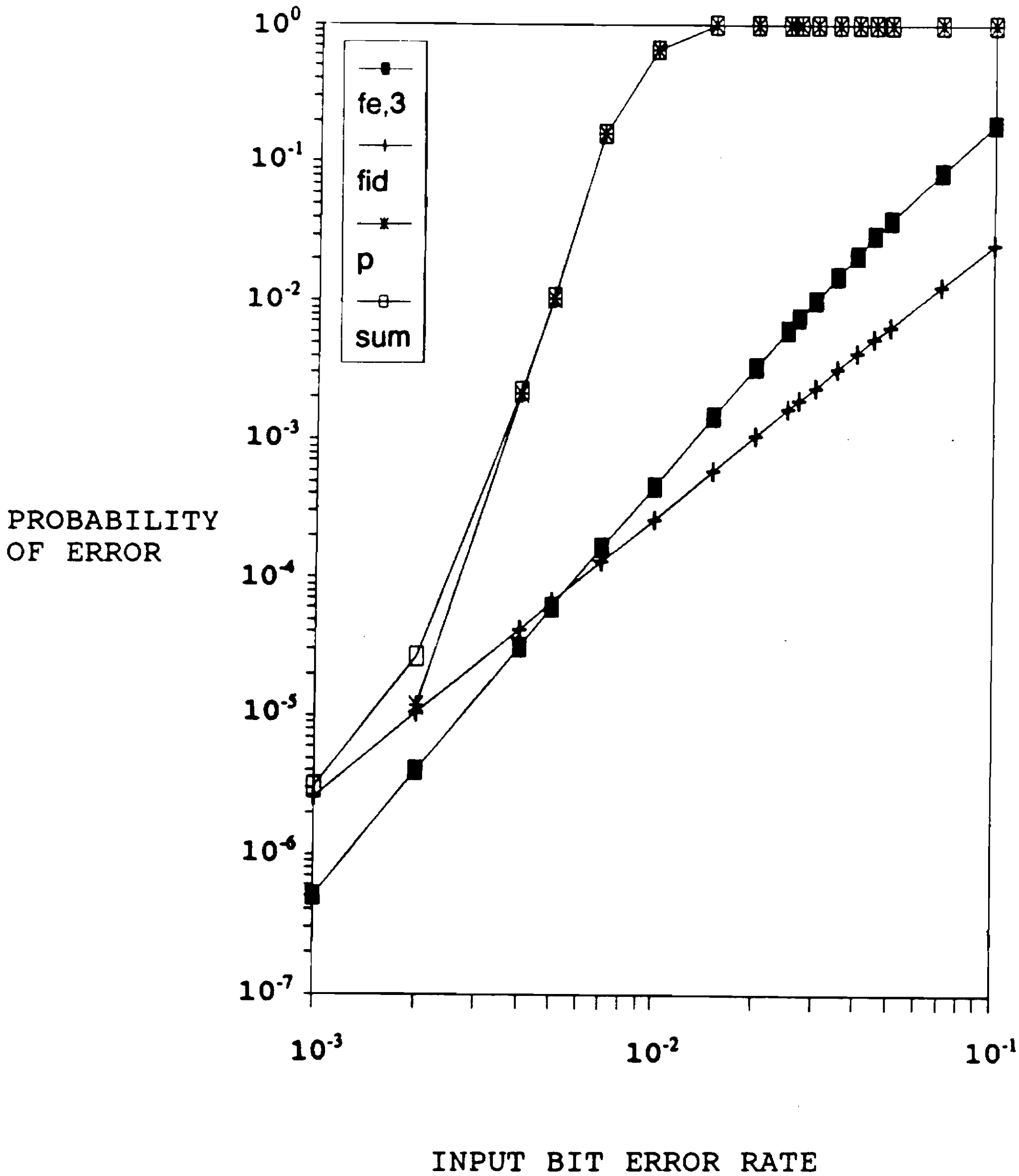


Figure 10.2.1.13 The probabilities of the opening flag being in error $P_{fe,3}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,221) RS code and three flags being used.

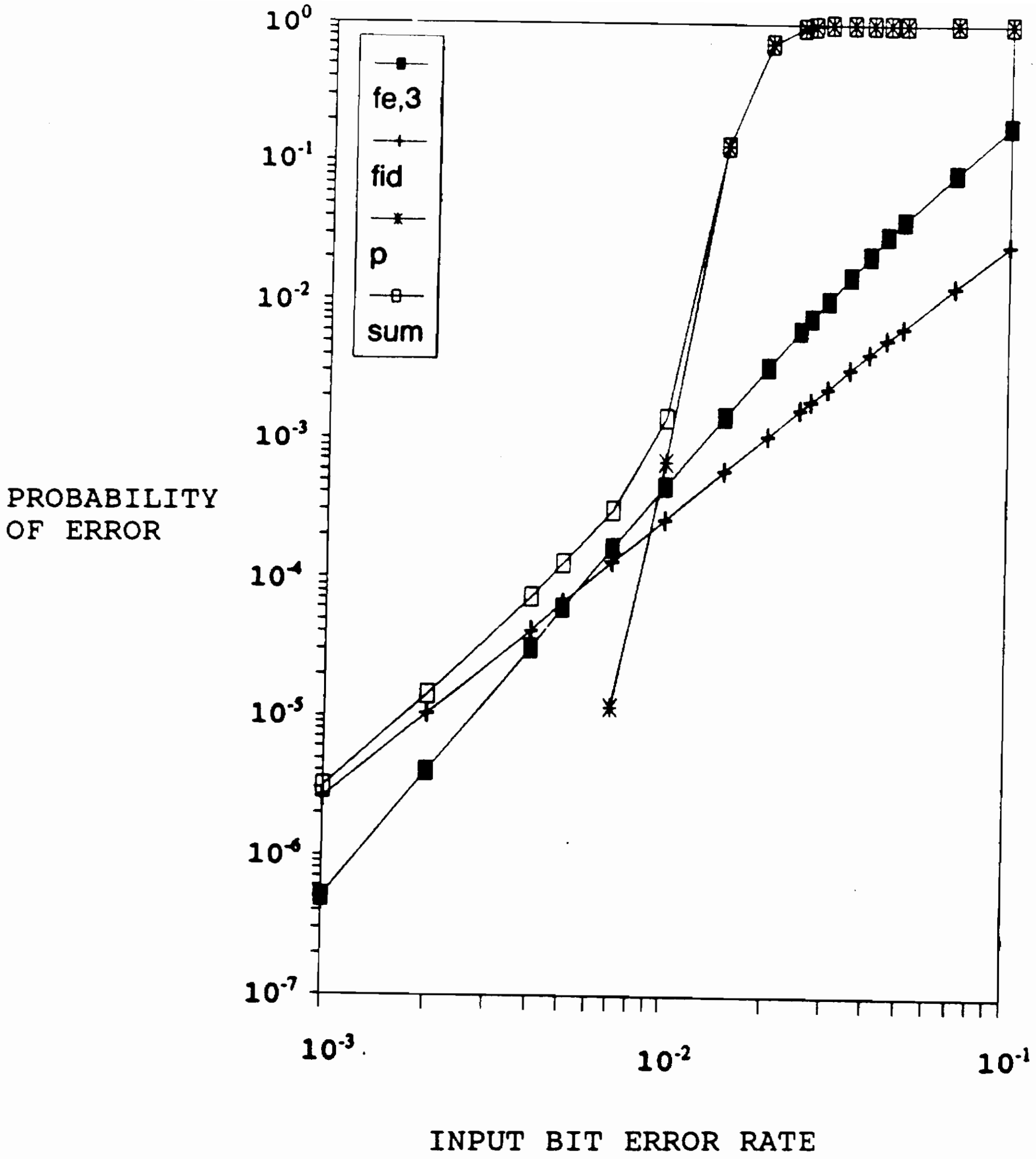


Figure 10.2.1.14 The probabilities of the opening flag being in error $P_{fe,3}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,187) RS code and three flags being used.

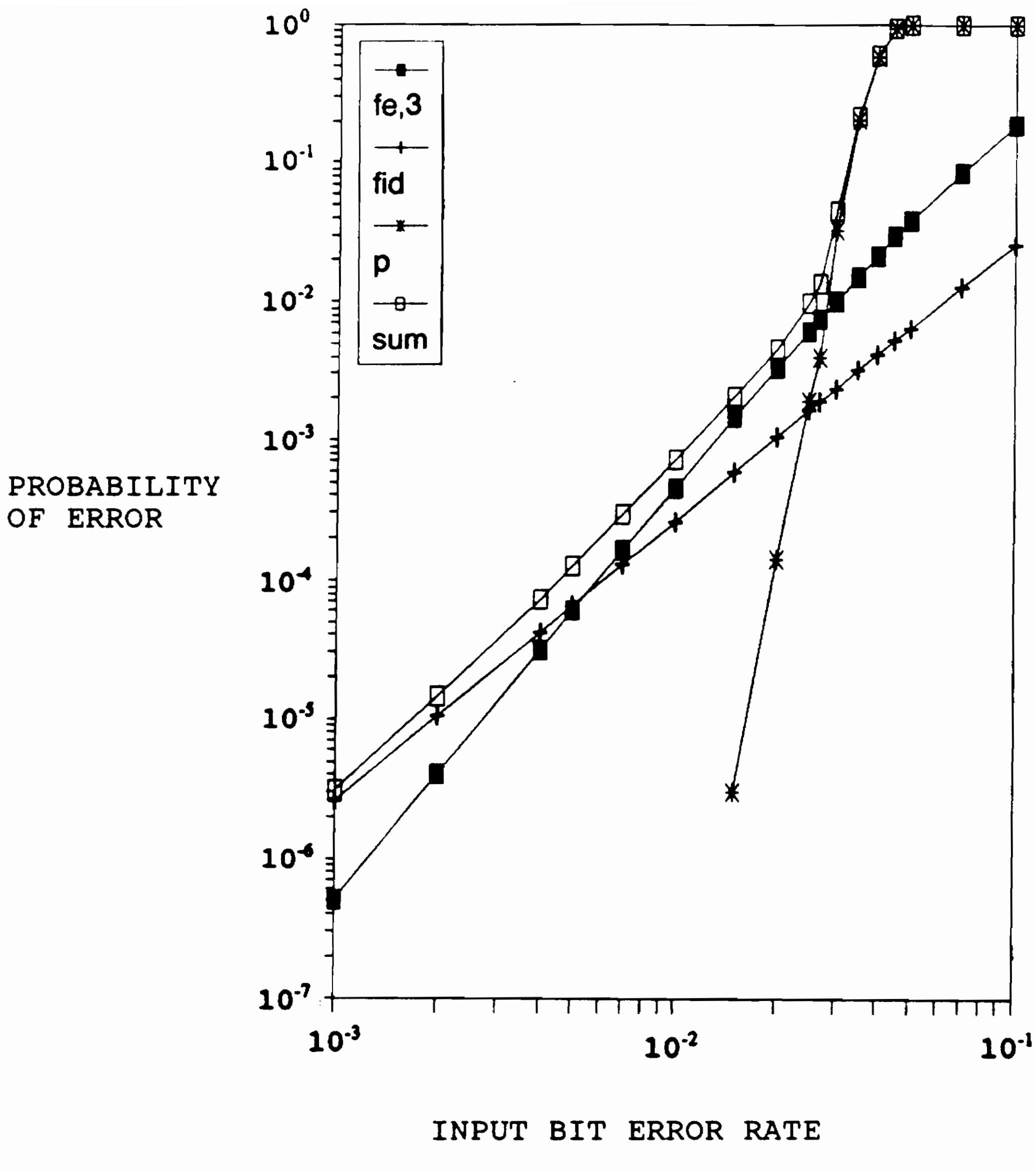


Figure 10.2.1.15 The probabilities of the opening flag being in error $P_{fe,3}$, the flag appearing in the data stream P_{fid}, the packet being in error P_p, and their sum P_{sum}, for the (255,119) RS code and three flags being used.

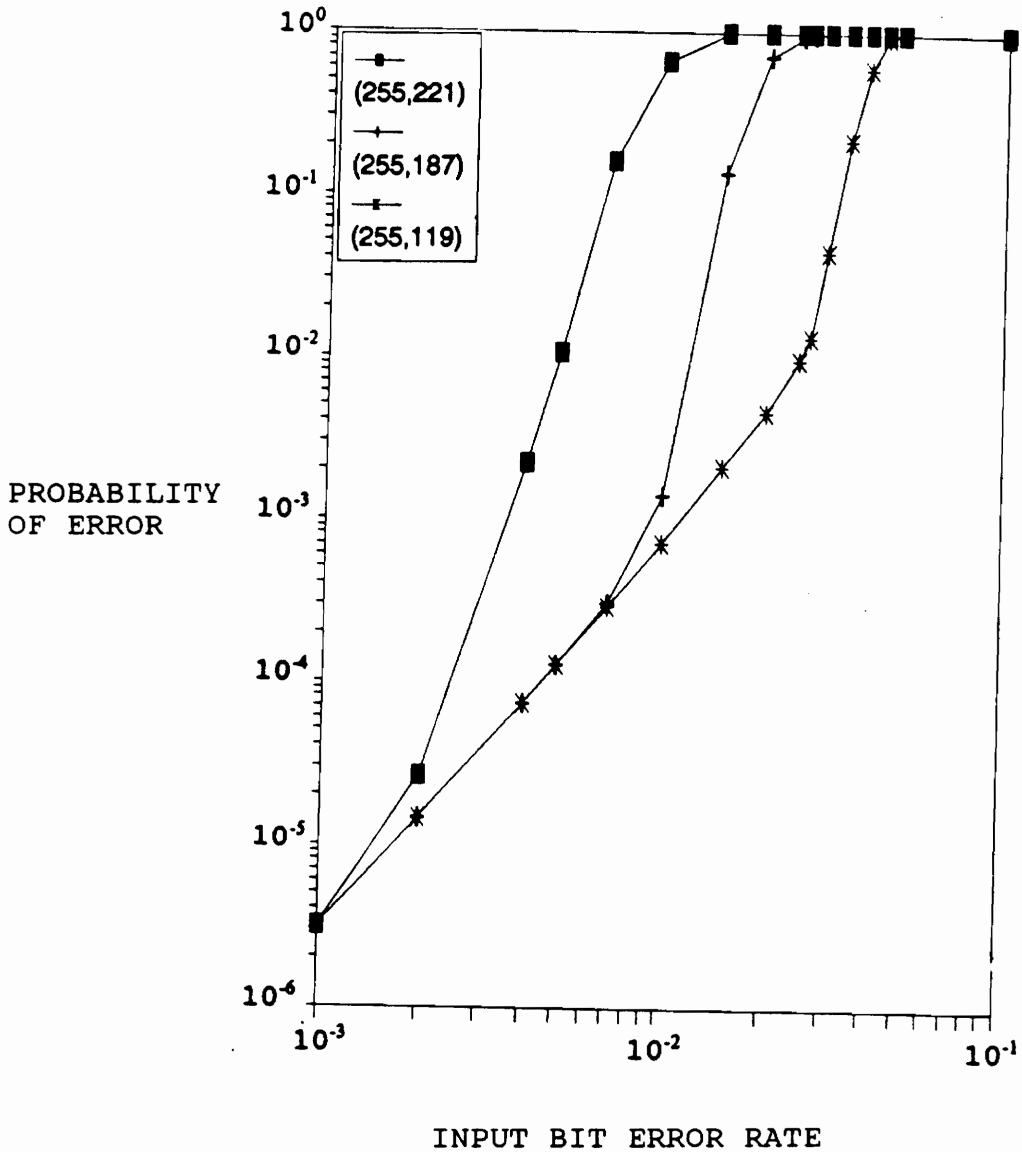


Figure 10.2.1.16 The probability P_{sum} for the (255,221), (255,187), and (255,119) RS codes, using three flags.

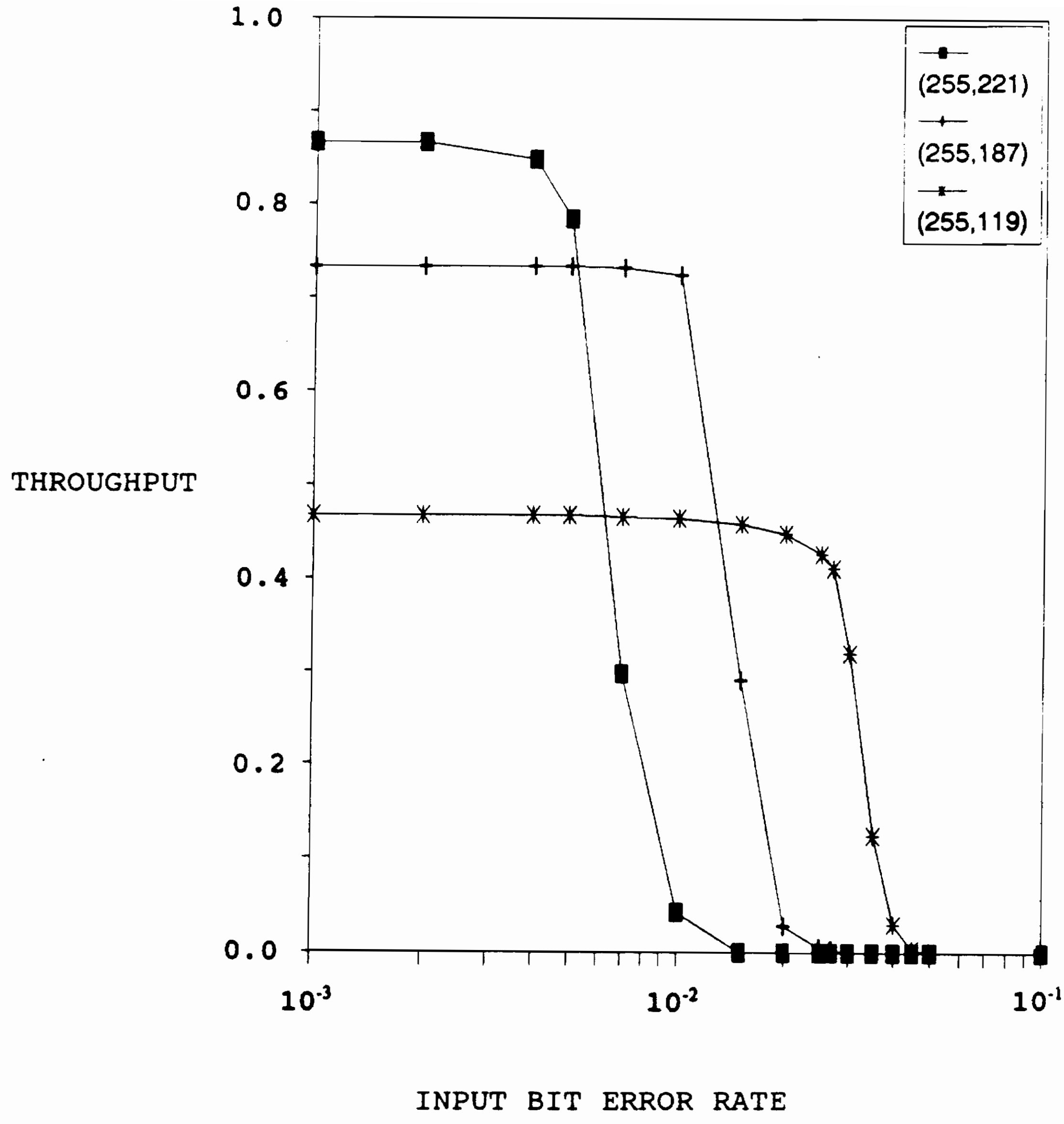


Figure 10.2.1.17 The throughput of a go-back-10 ARQ system which would be achieved using the probabilities in Figure 10.2.1.16 with three flags.

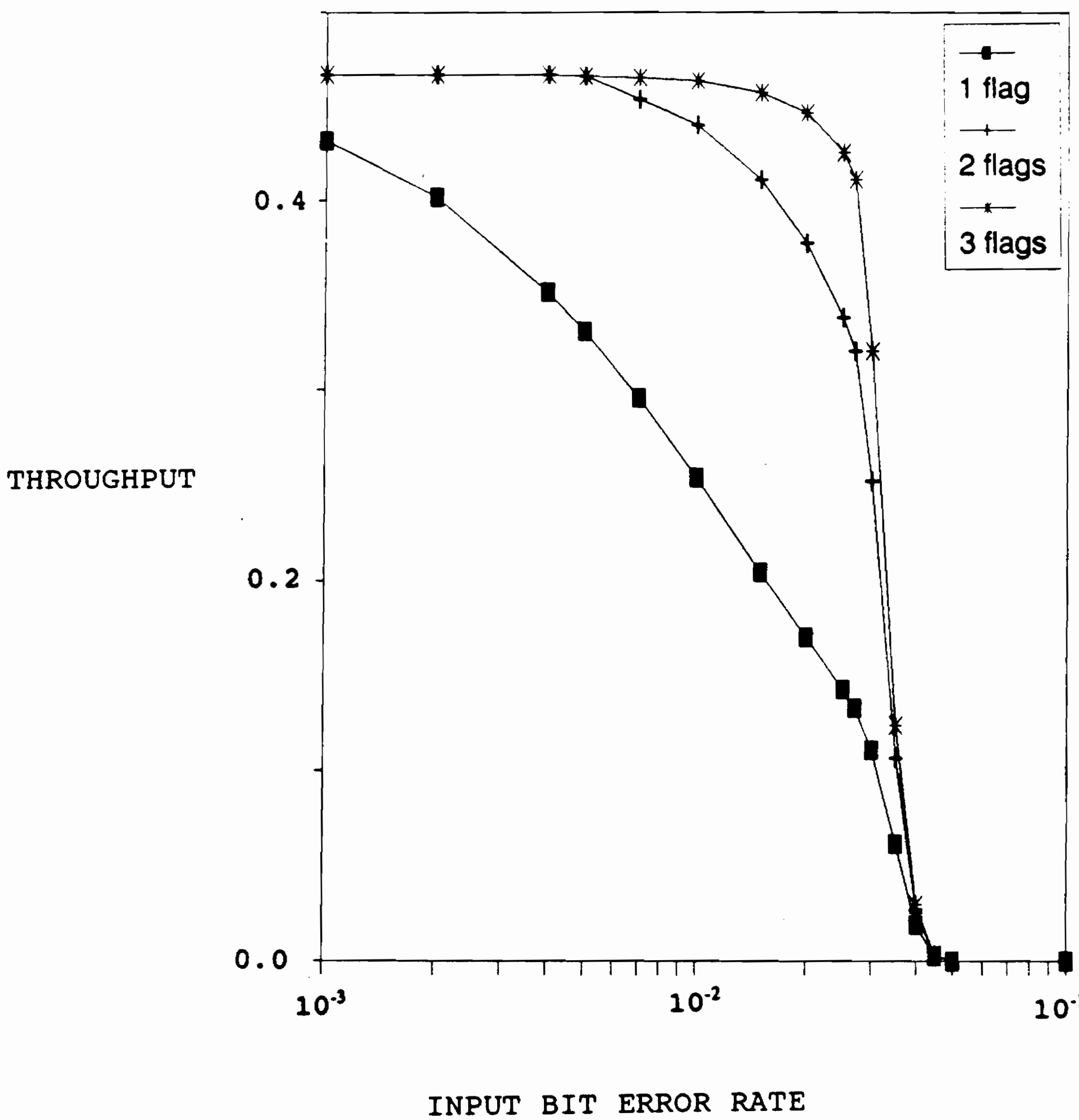


Figure 10.2.1.18 The throughput of a go-back-10 ARQ system for a (255,119) RS code, using 1, 2 and 3 flags.

By observing the previous figures, one can determine which probabilities dominate the maximum throughput. If one flag is used, as in Figure 10.2.1.5, the probability of the opening flag being in error dominates the other probabilities, and a powerful code such as the (255,119) RS would prove wasteful. With the use of three flags, as in Figure 10.2.1.15, the (255,119) code proves quite advantageous. The graph in Figure 10.2.1.18 shows that the addition of more flags is subject to the law of diminishing returns, as the throughput is increased only slightly at intermediate bit error rates.

In order to choose which of the three solutions is best to implement, they must be compared with respect to throughput and ease of implementation. The first solution is discounted because a short flag would appear in the data stream far too often. Figure 10.2.1.19 is a plot of the packet throughput if the (7,4) Hamming code is used, if 2 flags are used, and if 3 flags are used, when the balance of the packet is encoded using the (255,119) Reed-Solomon code. One can see that in Figure 10.2.1.19, the throughput using the encoded flag is much lower than that for 2 or 3 flags. Since the performance for the encoded flag is dominated by the flag being in error and not the packet being in error, one can conclude that this solution yield poor performance for the other RS codes.

In order to implement the encoded flag, a codec must be used. Although the (7,4) Hamming codex is simpler than most, it would still increase the system complexity and require memory accesses in order to complete the decoding. The multiple flag method simply requires discarding redundant flags and a slight modification to the bit stuffing routine. Therefore, the multiple flag solution is the simplest and provides higher throughput.

Through the use of Figures 10.2.1.3-18, one should be able to approximate the number of flags that would be required for a particular channel. If a high degree of error correction is required, a greater number of flags would also be required.

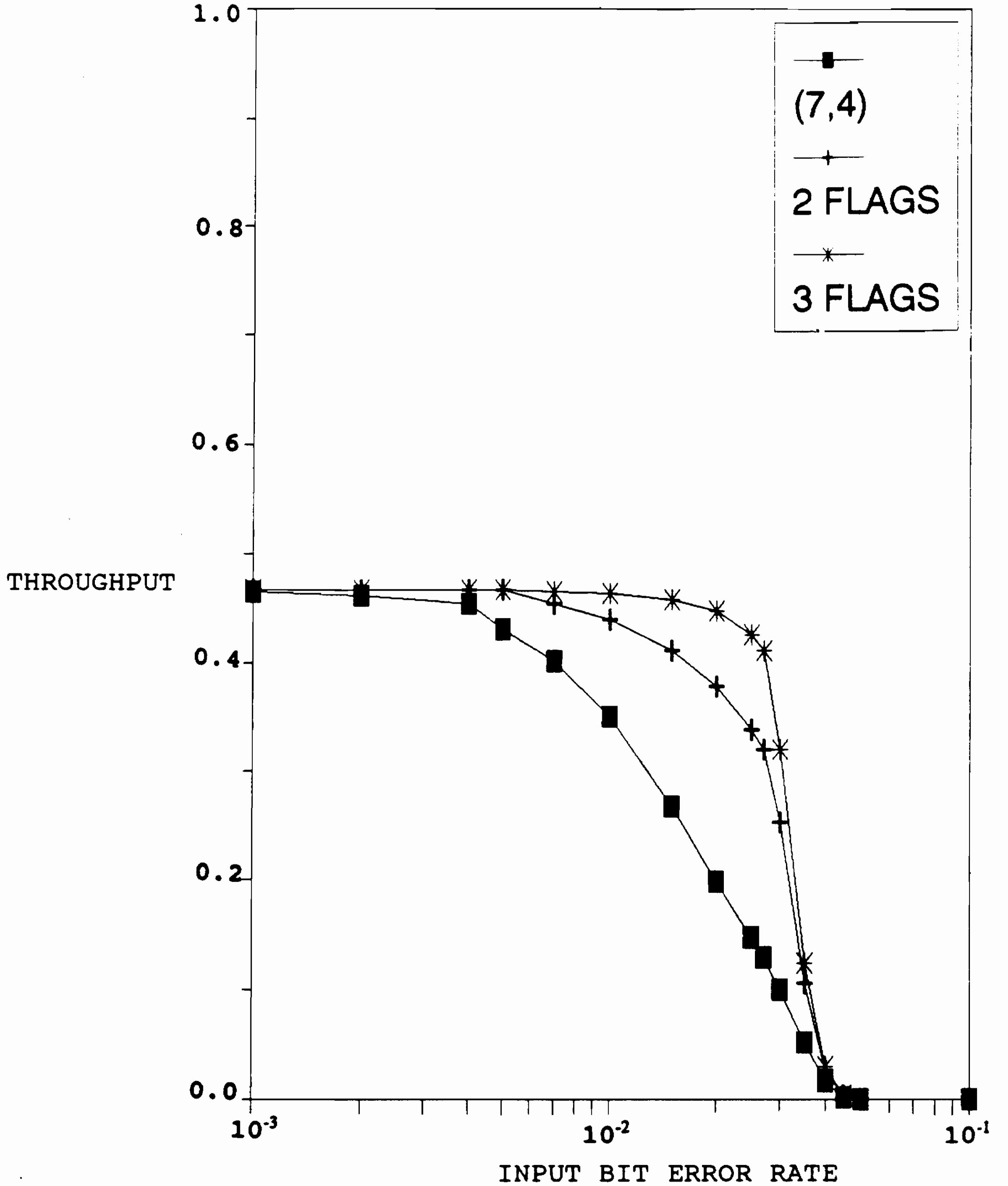


Figure 10.2.1.19 The throughput for the Go-Back-10 ARQ scheme with the flag encoded with a (7,4) Hamming code, the use of 2 flags, and the use of 3 flags, with the balance of the packet encoded with an RS (255,119) code.

10.2.2 Error Correction

The Reed-Solomon code will be used to provide the error correction. The entire packet, with exception of the flags, will be encoded by the RS codex. Recall that this includes the address, control, protocol identifier, information and frame check sequence fields. The overhead for all but the information field is 144 bits, which leaves 1896 bits, or 237 8-bit bytes, for the data, bit stuffing and redundancy of the error control code. If the amount of redundancy is preset, the packet assembler could determine the exact number of bits which are required to be stuffed. The number of information bits to be transmitted could be found by subtracting the redundant and stuffed bits from the 1896 total bits. The entire message to be transmitted could be divided accordingly.

10.2.3 Error Detection

The process of error detection will be provided by the frame check sequence. This detection only code will be the same one that is used in the current AX.25 system. The fields which will be encoded with this code are the address, control, protocol identifier, and information. Since the fields are encoded with this code prior to the RS encoding, this code will act as a back-up. There is a finite, although very small, possibility that the RS code can incorrectly decode the received packet. The frame check sequence would detect such an occurrence and prevent faulty data from being further processed.

A block diagram of the entire system can be found in Figure 10.2.3.1.

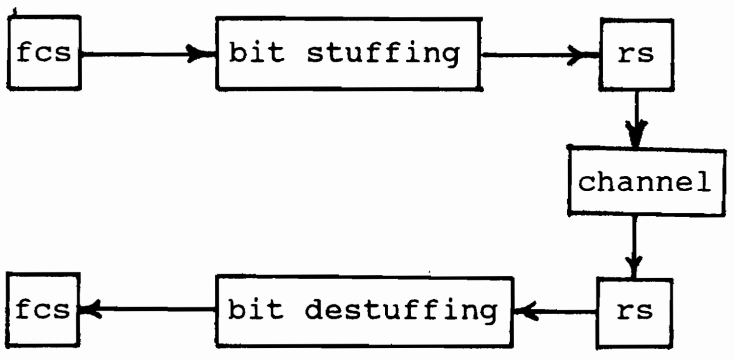


Figure 10.2.3.1 Block diagram of the proposed system. The order is of particular importance to optimize the system.

10.3 Packet Assembly

The packet assembly will be governed by the above system requirements and compatibility with the current AX.25 system. Obviously, the flag must begin the packet, in order to signal the receiver to begin reception. In hardware based systems, the redundant information is derived as the data is shifted through the encoder and out to the channel. Therefore, these bits will be found at the end of the packet. Because the frame check sequence is calculated before the RS encoding, it will precede the RS check bits. Therefore, the RS check bits will be the final bits sent, except for a closing flag. The remainder of the packet will be left as specified in the original AX.25 protocol. Note that, with the exception of the RS redundant bits, the original packet remains intact. This implies that if a Reed-Solomon codec was used after the original packet assembly, the RS coding would be transparent to the current system. The proposed packet order can be found in Figure 10.3.1.

FLAG(S)	ADDRESS	CONTROL	PID	INFO	FCS	RS
01111110	112 BITS	8 BITS	8 BITS	N*8 BITS	16 BITS	2t*8 BITS

Figure 10.3.1 The order of the packet is dictated by the system requirements and the original packet.

CHAPTER 11

CONCLUSIONS

11.1 Summary

The purpose of this project is to determine which error correcting code would be the most effective if implemented in a system which employs the AX.25 protocol. Many codes were discounted from consideration before evaluation due to poor performance. Those which were considered are the (7,4) Hamming code, (23,12) Golay code, various rate and constraint length convolutional codes, and Reed-Solomon codes over the Galois fields of 16 and 256 elements.

The (255,k) class Reed-Solomon codes over GF (256) have been chosen due to the evaluation of three criteria. These were: the throughput for a given input bit error rate; the simplicity of coding; and the performance over bursty channels. The RS codes have superior performance at high rates and are powerful and, therefore, would perform well over a challenging channel. The lack of required memory accesses gives the RS codes an edge over the others for the coding simplicity criterion. The burst error criterion decidedly goes to the RS codes because they can correct a large number of errors without using interleaving. Therefore, the Reed-Solomon codes are the best choice.

The rate of the best RS code has been unspecified. It is up to the user to evaluate the channel and decide how much error correcting power is needed. The graphs in this paper should provide an adequate benchmark for this determination.

It has also been determined that using multiple opening flags will reduce the possibility of the system not recognizing the flag, since the flag can not be encoded with the rest of the packet. With the use of an increasing number of flags, better throughput can be achieved. However, the advantages diminish rapidly if more than two or three flags are used.

11.2 Further Research

There are several other factors which must be considered. Convolutional codes can be easily decoded using soft decisions. Soft decision decoding usually yields about a 3 dB gain over hard decision decoding. However, soft decision decoding requires a much more complex receiving station and a more complex computation of partial metrics. Some research has been done in soft decision decoding for BCH codes[46] [47], although it is seldom used in practice.

Different evaluation criteria may modify the results found in this paper. One may find another set of criteria by which the codes should be measured. A different channel may cause a different set of results. One may have a different type of ARQ system in operation, such as selective-repeat. The round-trip delay time or the bit rate may also cause a different code to be chosen as superior.

Another factor which was not investigated in this study is that of code combining. Reed-Solomon codes are often combined with shorter BCH codes or convolutional codes. These codes can be quite formidable on noisy channels.

Lin and Yu have developed a type of ARQ which sends a high rate code used for error detection only.[48] When errors are detected, the erroneous codeword is stored at the receiver. The transmitter then encodes k parity bits, from an invertible code, which are sent over the channel. The k parity bits are inverted to recover the original message. If the message is still in error, these k bits are used in conjunction with the original received codeword to recover the information. If the data is still irrecoverable, the transmitter alternates sending the information vector and the parity vector until the codeword is decoded correctly. Using this scheme, a high rate code is used until errors are present and then a low rate code is used. This type of error control scheme would be interesting to investigate further.

Some of the above ideas, or a combination of them, could provide challenging research in the future.

REFERENCES

[1] S. Lin, D.J. Costello Jr. and M.J. Miller, "Automatic-Repeat-Request Error-Control Schemes," *IEEE Communications Magazine*, Vol. 22, Dec. 1984, pp. 5-16.

[2] S. Lin and D.J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983, p. 102.

[3] Ibid [2] pp. 102-103.

[4] Ibid [2] p. 67.

[5] A.J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, Vol. 13, Apr. 1967, pp. 260-269.

[6] T.L. Fox, *AX.25 Amateur Packet-Radio Link-Layer Protocol*, American Radio Relay League, Inc., Newington, CT, 1984.

[7] S. Horzepa, *Your Gateway to Packet Radio*, American Radio Relay League, Inc., Newington, CT, 1989, p. 12-6

[8] R.S. Hamming, "Error Detecting and Error Correcting Codes," *Bell Sys. Tech. Journal,* 1950, pp. 147-160.

[9] Ibid [2].

[10] R.E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Co., Reading, MA, 1983.

[11] G.C. Clark, Jr. and J.B. Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, New York, NY, 1981.

[12] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North Holland, Amsterdam, 1977.

[13] W.W. Peterson and E.J. Weldon, *Error Correcting Codes*, MIT Press, Cambridge, MA, 1972.

[14] J.G. Proakis, *Digital Communications*, McGraw-Hill, Inc., New York, NY, 1989. p. 381.

[15] M.J.E. Golay, "Notes on Digital Coding," *Proc. IRE,* 1949, p. 657.

[16] Ibid [2].

[17] Ibid [12].

[18] Ibid [14] p. 396.

[19] T. Kasami, "A decoding Procedure for Multiple-Error -Correcting Cyclic Codes," *IEEE Trans. Info. Theory,* IT-10, 1964, pp. 134-139.

[20] I.S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *J. Soc. Indust. Appl. Math,* 1960, pp. 300-304.

[21] Ibid [10].

[22] Ibid [10].

[23] R.L. Miller, T.K. Truong, I.S. Reed, "Efficient Program for Decoding the(255,223) Reed-Solomon Codes over GF(256) with both Errors and Erasures, Using Transform Decoding," *IEE Proc.* Vol. 127, Pt. E, No. 4, July 1980, pp. 136-141.

[24] E.R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, New York, NY, 1968.

[25] J.L. Massey, "Shift Register Synthesis and BCH Decoding," *IEEE Trans. Info. Theory,* IT-17, 1969, pp. 122-127.

[26] Ibid [10] pp. 207-211, 248-256.

[27] A.J. Viterbi, "Convolutional Codes and Their Performance in Communications Systems," *IEEE Trans. Comm. Tech.*, Vol. 19, Oct. 1971, pp. 751-772.

[28] Ibid [27].

[29] G.D. Forney Jr., "The Viterbi Algorithm," *Proc. IEEE*, March 1973, pp.268-278.

[30] G.D. Forney Jr., "Convolutional Codes II: Maximum Likelihood Decoding," *Info. Control*, July, 1979, pp. 222-226.

[31] G.D. Forney Jr., "Convolutional Codes III: Sequential Decoding," *Info. Control*, July, 1979, pp. 267-297.

[32] Ibid [14] p. 443.

[33] Ibid [2] p. 291.

[34] Ibid [19] p. 759.

[35] Ibid [11] pp. 402-403.

[36] L. Van De Meeberg, "A Tightened Bound on the Error Probability of Binary Convolutional Codes with Viterbi Decoding," *IEEE Trans. Info. Theory*, May 1974, pp. 389-391.

[37] K.A. Post, J.P.M. Schalkwijk and J.P.J.C. Aarts, Explicit Evaluation of Viterbi's Union Bounds on Convolutional Code Performance for the Binary Symmetric Channel," *IEEE Trans. Info. Theory*, Nov. 1979 pp. 737-743.

[38] Ibid [29].

[39] Ibid [30].

[40] Ibid [2] p. 337.

[41] Ibid [14] p. 459.

[42] Ibid [2] pp. 167-177.

[43] Ampex V5125/5126 ECC Encoder/Decoder Data Sheet.

[44] Ibid [2] pp. 337-342.

[45] Ibid [2] pp. 308-312.

[46] J.K. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis, *IEEE Trans. Info. Theory*, Jan. 1978, pp. 76-80.

[47] D.P. Chou and S.G. Wilson, *Soft Decision Decoding of CRC Codes*, Master Thesis, Department of Electrical Engineering, University of Virginia, 1991.

[48] S. Lin and P.S. Yu, "A Hybrid ARQ Scheme with Parity Retransmission for Error Control of Satellite Channels," *IEEE Trans. Comm.*, July 1982, pp. 1701-1719.

VITA

Bill Jesser was born in Waynesboro, Virginia on February 19, 1968. He graduated from the University of Virginia in 1990 with a Bachelor of Science in Electrical Engineering. While at the University of Virginia, he worked with Reed-solomon error correcting codes. In 1990, he began Master's work at Virginia Polytechnic Institute and State University. Presently, his interests include research in LEOSATs and mobile communication.