

***SIMULATION OF MULTI-TRACK, DROP-OUT
INFECTED DIGITAL MAGNETIC RECORDING
CHANNELS***

A Thesis Submitted to the University of Manchester
for the Degree of Doctor of Philosophy
in the Faculty of Science and Engineering

Volume I of II

Information Storage Research Group and Communications Research Group.
Division of Electrical Engineering, The Manchester School of Engineering,
Faculty of Science and Engineering



THE UNIVERSITY
of MANCHESTER

BY

AJAY TANDON

1997

ProQuest Number: 10835012

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10835012

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

1C156051X

✓ 7h 20459
(DAHX2)

JOHN RYLANDS
UNIVERSITY
LIBRARY OF
MANCHESTER

TABLE OF CONTENTS

Volume I

Declaration	16
Preface	17
Acknowledgements	17
Abstract	18

1 INTRODUCTION	20
-----------------------	-----------

1.1 History of magnetic recording	20
1.2 The theory of magnetic recording	21
1.2.1 The recording process	22
1.2.2 The replay process	23
1.2.3 Modes of magnetic recording	24
1.3 Error correcting codes applied to magnetic recording channels	25
1.4 Outline of the thesis	27

2 THE MAGNETIC RECORDING CHANNEL	31
---	-----------

2.1 Introduction	31
2.2 Matching data to the characteristics of the recording channel using channel codes	33
2.2.1 Considerations of the characteristics of channel codes	33

2.2.2 The channel codes	34
2.2.2.1 The non-return to zero channel code	35
2.2.2.2 The non-return to zero one channel code	36
2.2.2.3 The phase encoding channel code	37
2.2.2.4 The frequency modulation code	38
2.2.2.5 The delay modulation code	39
2.2.2.6 The Miller squared code	40
2.3 The longitudinal model	41
2.4 Waveform generation using linear superposition	47
2.5 Recovery of the data stream using timing windows	52
2.6 The effects of errors on timing windows	54
2.7 Summary	55
 3 ADVANCED CHANNEL SIMULATIONS	 57
3.1 Introduction	57
3.2 The characteristics of the recording channel	58
3.2.1 Maximum output voltage analysis	58
3.2.2 Cross-over shift analysis	59
3.2.2.1 Cross-over shift effects on a symmetrical data pattern	61
3.2.2.2 Cross-over shift effects on a worst case data pattern	61
3.2.2.3 Cross-over shift effects on a pseudo-random data pattern	63
3.2.2.4 Average cross-over shift over a large packing density range	64
3.3 A generalised study of the predictions of the performance of recording	65

channels by the application of scaling	
3.3.1 Introduction	65
3.3.2 Calculations	65
3.3.3 Predictions	71
3.3.3.1 Analysis of average cross-over shift	72
3.3.3.2 Analysis of bit error rate	75
3.3.3.3 Analysis of packing density	77
3.3.4 Conclusions of this study	78
3.4 Incorporating large data trains	79
3.5 Incorporating multi-tracks into the simulation	80
3.6 Summary	81
4 INCORPORATING DEFECTS INTO THE RECORDING CHANNEL	82
4.1 Introduction	82
4.2 Analysis of the effects of noise by the use of the classical analytical technique	84
4.3 Pulse equalization	86
4.4 Drop-out infected channels	90
4.4.1 Modelling the drop-out characteristics	91
4.4.2 Statistical analysis of drop-outs	92
4.5 Addition of gaussian white noise samples to the recording channel	98
4.6 A comparison of the two techniques	102
4.6.1 The accuracy of the linear superposition of pulses	104
4.6.2 A study investigating the performance limitations due to ISI and	105

noise	
4.6.2.1 Introduction	105
4.6.2.2 Theoretical studies and experimental results	106
4.6.2.3 Conclusions of this study	109
4.7 Bit error rate comparison for advanced media	109
4.8 Analysis of burst error events produced by the new sampled AWGN technique	111
4.8.1 Internal cross-over shift characteristics	111
4.8.2 Comparison of data files before and after the modelling process	114
4.9 Summary	119
5 THE EFFECTS OF INTERLEAVING ON ERROR PERFORMANCE	121
ESTIMATION SCHEMES	
5.1 Introduction	121
5.2 Reed Solomon codes	122
5.3 Interleaving techniques	124
5.3.1 Horizontal symbol stack arrangement	124
5.3.2 Vertical symbol stack arrangement	125
5.3.3 Symbol interleaving	126
5.3.4 Block interleaving	127
5.3.5 Block and symbol combined interleaving	129
5.4 Error performance estimation information produced by the different interleaving techniques	130
5.4.1 Block, symbol and bit errors and error rate analysis	130

5.4.2 Analysis of the number of symbols to be corrected	131
5.4.3 Symbol error rate analysis	134
5.4.4 Symbol error rate trends	135
5.4.4.1 Trends that occur at different interleaving depths	136
5.4.4.1.1 Trends that occur using only symbol interleaving	136
5.4.4.1.2 Trends that occur using both block and symbol interleaving	139
5.4.4.2 Trends that occur with a range of block sizes for specific interleaving depths	141
5.4.5 Analysis of block size trends	150
5.4.6 Summary of the trends found	152
5.5 Summary	152
6 DISCUSSIONS AND CONCLUSIONS	154
6.1 Review of the thesis	154
6.2 Achievements of the novel simulation technique	158
6.3 Further work	160
6.3.1 Improvements to the model	160
6.3.2 Further studies of particular aspects of the model	161
6.3.3 Error correction and detection studies	163
REFERENCES	164

Volume II

APPENDIX A	174
A.1 Top/down structure analysis	174
A.2 PC requirements and program details	175
A.3 UNIX requirements and program details	181
A.3.1 The programs	181
A.3.2 The input parameter files of interest	182
A.3.2.1 The '.log' input parameter file	182
A.3.2.2 The '.mmi' input parameter file	185
A.3.3 Requirements	189
A.3.4 Structure of the model	190
A.3.5 Compiling the programs	191
A.3.6 Program examples	191
A.3.6.1 Running FAGEN	193
A.3.6.2 Running FACODE	193
A.3.6.3 Running FADROP	194
A.3.6.4 Running FAMAGMOD	194
A.3.6.5 Running FADECODE	196
A.3.6.6 Running FADISP	196
A.3.7 Alternative methods of file analysis	197
A.3.7.1 Using an error coder/decoder instead of a line coder/decoder	198
A.3.7.2 Using a combined error and line decoder	199

A.3.7.3 Replacing programs	200
A.4 Selected program modules	201
A.4.1 Linear superposition module	201
A.4.2 Longitudinal model definition pulse	202
A.4.3 Mixed model definition pulse	205
A.4.4 Pulse equalization module	207
A.4.5 Analytical bit error rate calculation module	209
A.4.6 AWGN generation module	212
A.4.7 New technique bit error rate calculation module	215
A.4.8 Drop-out generation module	218
A.4.9 Block, symbol and bit errors (and error rates) module	219
A.4.10 Vertical stacking arrangement module	224
A.4.11 Symbol interleaving module	225
A.4.12 Block interleaving module	226
A.4.13 Block and symbol interleaving module	228
APPENDIX B	231
B.1 Published papers	231
APPENDIX C	260
C.1 A detailed investigation of block, symbol and bit errors and error rates for a wide range of interleaving depths and block sizes	260

C.2 A detailed investigation of the number of symbols to be corrected for a wide range of interleaving depths and block sizes	297
C.3 A detailed investigation of the symbol error rates for a wide range of interleaving depths and block sizes	330

TABLE OF FIGURES

1.1: The magnetic record and replay process	21
1.2: The recording process	22
1.3: The magnetic recording modes	24
2.1: Block diagram of the magnetic recording channel	31
2.2: Waveforms on various parts of the channel (labelled in figure 2.1)	32
2.3: Coding a data pattern using the non-return to zero code	35
2.4: Coding a data pattern using the non-return to zero one code	36
2.5: Coding a data pattern using phase encoding	37
2.6: Coding a data pattern using the frequency modulation code	38
2.7: Coding a data pattern using the delay modulation code	39
2.8: Coding a data pattern using the Miller squared code	40
2.9: The longitudinal model replay pulse (shifted by 5 μm)	45
2.10: The longitudinal model differentiated pulse (shifted by 5 μm)	45
2.11: An example mixed model replay pulse (shifted by 50 μm)	46
2.12: An example mixed model differentiated pulse (shifted by 50 μm)	47
2.13: The longitudinal replay waveform at 20,000 bits per inch	49
2.14: The longitudinal differentiated waveform at 20,000 bits per inch	49
2.15: The longitudinal replay waveform at 100,000 bits per inch	50
2.16: The longitudinal differentiated waveform at 100,000 bits per inch	50
2.17: The longitudinal replay waveform at 160,000 bits per inch	51
2.18: The longitudinal differentiated waveform at 160,000 bits per inch	51

2.19: A sampled representation of a zero crossing point	52
2.20: A cross-over shifted point within a timing window	53
2.21: A differentiated waveform including several timing windows	54
2.22: The timing window of a bit in error	54
3.1: Maximum output voltage over a wide range of packing densities	58
3.2: Maximum output voltage (log scale) over a wide range of packing densities	59
3.3: The no cross-over shift data pattern at 100,000 bits per inch	61
3.4: Cross-over shift that occurs in the worst case data pattern as a percentage at 100,000 bits per inch	62
3.5: Cross-over shift that occurs in the worst case data pattern in the time domain at 100,000 bits per inch	62
3.6: The pseudo-random cross-over shift data pattern at 100,000 bits per inch	63
3.7: The average cross-over shift over a wide range of packing densities	64
3.8: Normalised output amplitude as a function of packing density	70
3.9: Normalised output amplitude as a function of $(a_t + d)$ / bit spacing	71
3.10: Average cross-over shift / bit spacing as a function of packing density	72
3.11: Average cross-over shift / bit spacing as a function of $(a_t + d)$	73
3.12: Average cross-over shift as a function of $D / (a_t + d)$	74
3.13: Bit error rate as a function of packing density	75
3.14: Bit error rate as a function of $(a_t + d)$	75
3.15: Bit error rate as a function of $D / (a_t + d)$	76
3.16: Packing density as a function of $(a_t + d)$	77

3.17: Packing density * ($a_t + d$) as a function of ($a_t + d$)	78
3.18: An illustration of the block structure method	79
3.19: Modelling a multi-track recording system	80
4.1: The interaction of the original pulse and time-shifted pulses	86
4.2: An example equalized and original pulse	88
4.3: An example of a distorted pulse	88
4.4: Cross-over shift analysis of the two pulses	89
4.5: The maximum output voltage analysis of the two pulses	90
4.6: A drop-out infected waveform generated at 20,000 bits per inch	91
4.7: The drop-out statistics produced for use in an eight track tape	97
4.8: A drop-out covering five tracks of an eight track tape channel	97
4.9: An example noisy differentiated waveform	99
4.10: 'C' code for the random number generator	100
4.11: 'C' code for the AWGN generator	101
4.12: The output amplitude voltage as a function of record current	105
4.13: Bit error rates as a function of record current	106
4.14: Optimum record current as a function of signal to noise ratio for different timing windows	108
4.15: Bit error rate as a function of packing density	110
4.16: The internal cross-over shift effects due to random noise	112
4.17: The internal cross-over shift effects due to a small drop-out burst	113
4.18: The internal cross-over shift effects due to a large drop-out burst	113
4.19: The simulation data file analysis	114

4.20: An XOR truth table	115
4.21: File comparison of a random error across an eight track tape	115
4.22: File comparison of a small multi-track error burst across an eight track tape	116
4.23: File comparison of a large multi-track error burst across an eight track tape	116
4.24: A burst with 5 empty guard space columns across an eight track tape	117
4.25: Burst statistics	118
5.1: The horizontal stacking arrangement of symbols	124
5.2: The vertical stacking arrangement of symbols	125
5.3: Symbol interleaving for the horizontal stack arrangement	126
5.4: Symbol interleaving for the vertical stack arrangement	126
5.5: Block interleaving for the horizontal stack arrangement	127
5.6: Block interleaving for the vertical stack arrangement	128
5.7: Combined interleaving for the horizontal stack arrangement	129
5.8: Combined interleaving for the vertical stack arrangement	129
5.9: A single track that can occur in tape systems	132
5.10: Interleaving depths for symbol interleaving only with horizontal stacking arrangement for block size 128	136
5.11: Interleaving depth trends for symbol interleaving only with vertical stacking arrangement for block size 128	137
5.12: Interleaving depth trends for both block and symbol interleaving with horizontal stacking arrangement for block size 128	139

5.13: Interleaving depth trends for both block and symbol interleaving with vertical stacking arrangement for block size 128	140
5.14: Trends of the different interleaving strategies at block size 256 and interleaving depth 5000	142
5.15: Trends of the different interleaving strategies at block size 64 and interleaving depth 5000	143
5.16: Trends of the different interleaving strategies at block size 256 and interleaving depth 500	144
5.17: Trends of the different interleaving strategies at block size 64 and interleaving depth 500	145
5.18: Trends of the different interleaving strategies at block size 256 and interleaving depth 50	146
5.19: Trends of the different interleaving strategies at block size 64 and interleaving depth 50	147
5.20: Trends of the different interleaving strategies at block size 256 and interleaving depth 10	148
5.21: Trends of the different interleaving strategies at block size 64 and interleaving depth 10	149

TABLE OF TABLES

1: The set of parameters defining the recording channel	44
2: Values of a_t and d for study about scaling	69
3: A previously used longitudinal pulse	87
4: Drop-out values required for simulating a set of packing densities	92
5: Analysis of block, symbol and bit errors and error rates at interleaving depth 100 and block size 128 using horizontal stack symbol interleaving	130
6: Analysis of number of symbols to be corrected at interleaving depth 100 and block size 128 for different stack and interleaving scenarios	132
7: Analysis of symbol error rates at interleaving depth 100 and block size 128 for different stack and interleaving scenarios	134
8: Analysis of block size trends for symbol interleaving only strategy	150
9: Analysis of block size trends for both block and symbol interleaving strategy	151

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or institution of learning.

Copyright in text of this thesis rests with the Author. Copies (by any given process) either in full, or of extracts, may be made only in accordance with instructions given by the author and logged in John Rylands University Library of Manchester. Details may be obtained from the Librarian. This page must form part of any copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the Author.

The ownership of any intellectual rights which may be described in this thesis is vested in the University of Manchester, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.

PREFACE

Ajay Tandon received his BEng in 1990 at the University of Teesside in Computer Engineering and, in 1993, an MSc in Electronic Engineering at the University of Nottingham. In October 1994, he received an EPSRC Studentship award to pursue a PhD with the Information Storage Research Group and the Communications Group at the University of Manchester.

ACKNOWLEDGEMENTS

I would like to thank all the members of both the Information Storage and Communications Research groups including Paul, Mike, Steve, Jason, Leon, Ashar, Apirat, Anis, Mustafa and Michael for advice and support and Colin, Andrew and Simon from other departments of the university. I would also like to thank Professor B.K. Middleton, Professor P.G. Farrell, Dr. Jim Miles, Dr. David Tait and Tony Arnold for specialist insights into the generation of different aspects of the magnetic model, gaussian noise, error correction codes and the network queuing system. I would also like to thank my flatmates Peter, Simon, Pauline, Darren, David, Tracy, Rachael, my family and neighbours for their great support and care in the attainment of the various goals that I set myself.

ABSTRACT

In digital magnetic recording channels, a common type of error is the burst error that occurs due the presence of drop-outs within the recording channel whether it be a single track or a multi-track channel.

This thesis examines a novel technique for the modelling of a multi-track longitudinal recording channel which uses peak detection. This technique involves sampling pulses and waveforms and combining sampled waveforms with additive white gaussian noise samples. A structure has been defined that allows the simulation to handle large data trains of 10 million bits that are required for meaningful simulations. Analysis of drop-outs has also been performed by allowing drop-outs to be incorporated into the recording channel and the burst error events that occur in multi-track drop-outs have been investigated.

Additional flexibility in the channel is described which allows the channel to function at high packing densities (exceeding one million bits per inch) and with pulses whose parameters are defined either in terms of tape and head characteristics or in terms of samples. A full analysis of the cross-over shift characteristics of the recording channel and the use of timing windows to recover the data stream from the tape medium have been produced.

To verify the validity of the new modelling technique, a previously published technique is also presented. A comparison of the bit error rates produced by both these techniques demonstrates the validity of the new technique. Also the new technique examines internal

characteristics and drop-out burst effects of the recording channel that are difficult to produce using the classical technique.

Studies have been performed that show, by the use of scaling, that the lowest practical value of $(a_1 + d)$ of 0.025 microns, can be examined. Also investigations were performed at 160,000 bit per inch (which represents leading edge tape media), and a full analysis was produced of the effects of a noisy and drop-out infected multi-track environment.

Finally, an examination of error performance estimation schemes has been conducted. Three main interleaving strategies have been investigated, these being symbol interleaving, block interleaving and a combination of both. All three strategies are analysed using horizontal and vertical stacks. A full analysis has been produced of the blocks, symbols and bits in error and the block, symbol and bit error rates. Different interleaving depths and block sizes are investigated and the improvements in error performance for a typical eight track tape have been shown.

Specifically, it can be shown that at high interleave depths, very low power error detection and correction (EDC) schemes (e.g. $t = 1 - 3$) are required to remove all the error symbols from the system. This reduces the cost and simplifies the design of the EDC coding scheme. At an interleaving depth of 5000, using a strategy of mixing both symbol and block interleaving, a 3 symbol EDC code is required to remove all the error symbols from the system.

1 INTRODUCTION

1.1 HISTORY OF MAGNETIC RECORDING

The oldest work known about the magnetic recording process was proposed by Oberlin Smith [Smith, O., (1888)]. He described the concept of recording and playing back magnetically, either on iron dust particles held by a non magnetic carrier or on a shield steel thread. This concept was published in 1888. The first working demonstration of magnetic recording was produced in 1900 by Voldermar Poulsen [Poulsen, V., (1900)] with the use of steel wire and an electromagnet. He called this device the telegraphone (a combination of *telegraph* and *telephone*) and by 1907 it was perfected. The definitive version of the telegraphone was patented in 1909. Despite the promise of this new technique, it was not widely used as the recorded signals could only be heard with the use of earphones, and the sound quality was very poor. The breakthrough in the quality of magnetic recording came in the 1930's with the development of magnetic oxide coating and the ring-type magnetic head.

The first recorder to use particle tape was called the Magnetophone and was produced in 1935. Initially, however, it had mediocre sound quality. In 1942, with the use of iron-oxide tape and A. C. bias techniques, the sound quality was improved tremendously and all other steel tape machines became obsolete. So work then continued on improving tape and head characteristics. More recently, the Philips compact cassette was introduced in 1963 and has been accepted all over the world as an audio tape standard. In 1976, JVC introduced the Video Home System (VHS) standard. Currently, digital formats such as digital audio tape

(DAT), digital compact cassette (DCC), mini-disc and digital versatile disc (DVD) have been introduced to the marketplace but a dominant format has not as yet been established.

1.2 THE THEORY OF MAGNETIC RECORDING

The magnetic recording process can be divided into two separate processes, the recording process and the replay process. The recording process records the input signals onto the magnetic tape medium using a recording head, and the replay process recovers the recorded signal using a replay head.

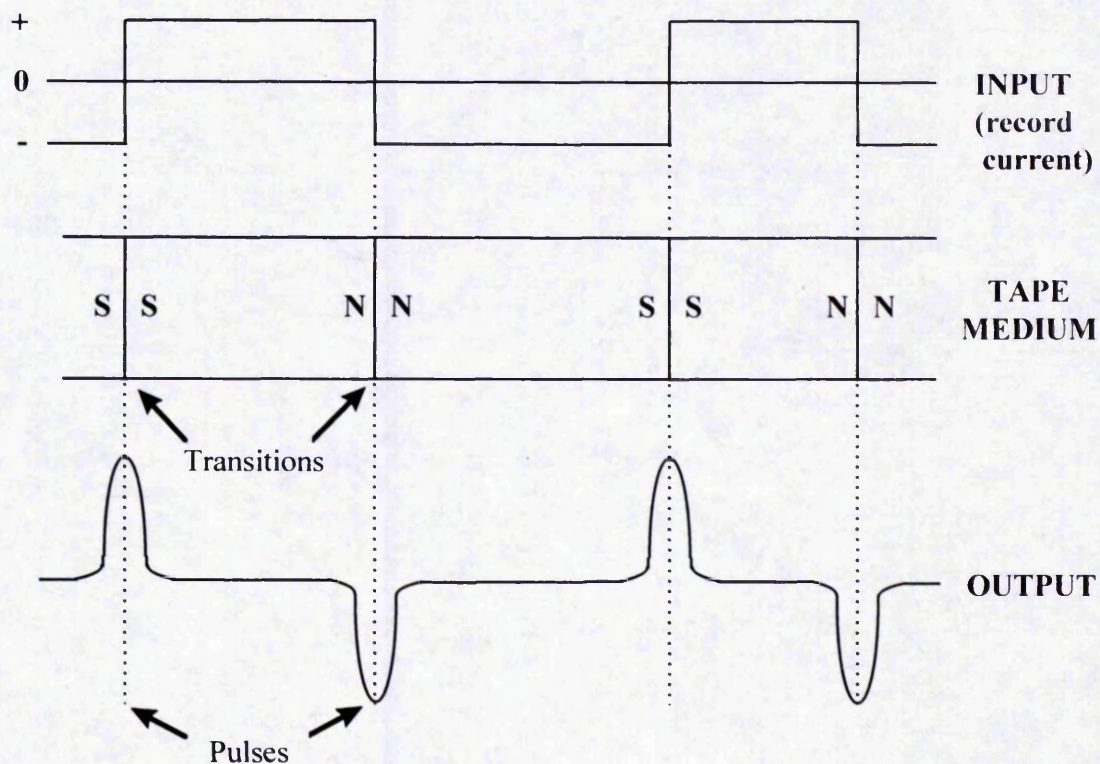


Figure 1.1: The magnetic record and replay process

Figure 1.1 shows this basic process [Watkinson, J., (1990)], with a square wave input being used to magnetise the magnetic tape medium alternately in the two possible states of record current. The replay head operates by producing in its coil a flux related to the magnetic flux of the medium. The output voltage produced is a differentiated version of the record waveform due to the head only responding to the rate of change of flux produced.

1.2.1 The Recording Process

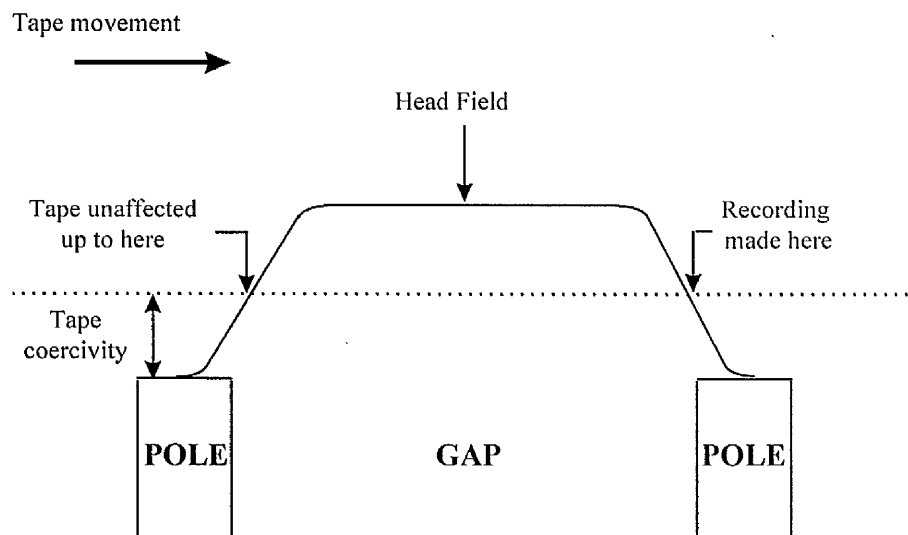


Figure 1.2: The recording process

The recording field is shown in figure 1.2 and the recording is made near the trailing pole of the head where the flux density from the head falls below the coercive force needed to change the magnetic state of the particles. The steeper the flux gradient on the trailing pole, the shorter the transition and therefore the wavelength which can be recorded. Detailed information about the recording process is provided in Chapter 2 of "*Magnetic Recording Volume I: Technology*" [Mee, C. D., and Daniel, E. D., (1996)].

To provide the best signal to noise replay characteristics the current required is a little less than the saturation level of the magnetic tape medium [Mee, C. D., and Daniel, E. D., (1996)]. This avoids the saturation effects of fringing fields around the head and crosstalk in adjacent tracks.

1.2.2 The Replay Process

The task of the replay circuits is to reconstruct the recorded waveform. When considering the replay process, the time at which the write current (and hence the flux) reverses is more important than the amplitude of the signal. This recovery process can be achieved by locating the peaks of the replay pulses. At high data rates this can be done by differentiating the signal and looking for zero-crossings. One difficulty arising is noise between the peaks. This is overcome by the gated peak detector, where only zero-crossings from a pulse which exceeds the threshold will be counted. An alternative method [Deeley, E. M., (1986)] to differentiation is to integrate the replay waveform. This method is not used due to poor characteristics and a very poor signal to noise (SNR) ratio.

Another important consideration in the replay process is the interactions between independent and opposite pulses positioned close together. Interaction between the two pulses reduces the amplitude of the signal and moves the peaks apart. This is known as peakshift (and is also known as Intersymbol Interference (ISI)).

1.2.3 Modes of Magnetic Recording

There are three modes of orientation in a magnetic medium; these are longitudinal, perpendicular and transverse. Figure 1.3 shows the geometry of these modes in relation with the motion of the magnetic medium.

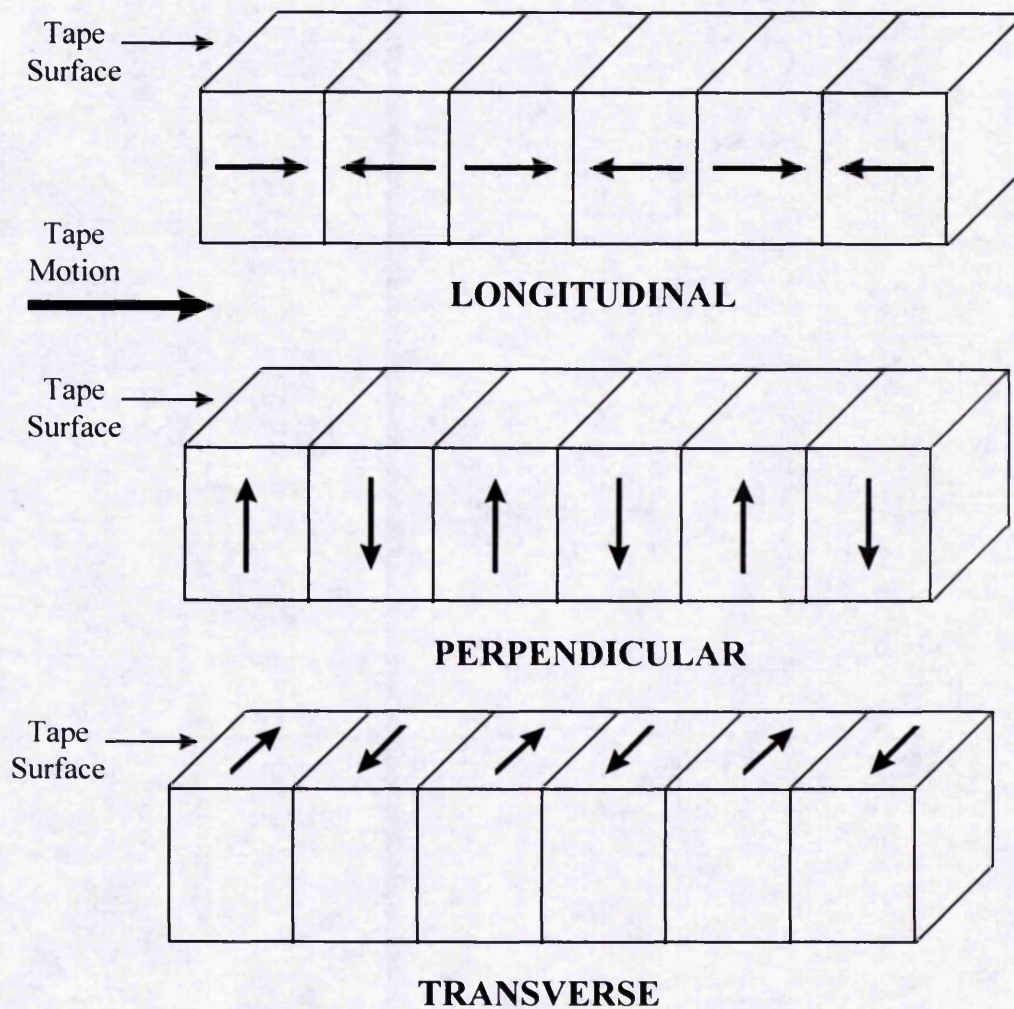


Figure 1.3: The magnetic recording modes

Longitudinal recording [Mallinson, J. C., (1985)] is the most common recording mode in use. The magnetic medium is made so that the easy axis of the medium's particles are aligned horizontally to the medium plane. It has extremely good characteristics, but at high recording densities, the replayed signal decreases rapidly. This thesis will concentrate on simulating longitudinal models.

With perpendicular recording, the medium's particles are aligned perpendicularly to the medium plane. Perpendicular recording [Iwasaki, S. and Ouchi, K., (1978)] has, in theory, some advantages over longitudinal recording, and has been predicted to produce better performance [Wallace, R. L., (1951)]. In practice, however, only a limited advantage has been found. Thus not many digital magnetic recording systems use this recording mode.

With transverse recording, the medium's particles are aligned in a transverse manner to the medium plane. Transverse recording is not in use as it has noise problems and a low signal to noise ratio. There is one paper [Miles, J. J. and Middleton, B. K., (1991)], which suggests that this recording mode has potential for extremely high density data storage.

1.3 ERROR CORRECTING CODES APPLIED TO MAGNETIC RECORDING CHANNELS

Error correction works by adding redundancy to a data stream. This redundancy has been calculated from the data stream, and when combined with the data stream creates an entity known as a code word which is transmitted in the place of the original data stream. At the

receiver this redundancy is used to detect and correct any errors in the data that occur due to the channel.

There are two main classes of error control codes, known as block codes and convolutional codes. In block codes, a block of k data digits is encoded into a code word of n digits ($n > k$). The difference between n and k is the r -symbol set of parity checks. This n symbol code word is then transmitted through the magnetic recording channel. At the receiver the r -symbol parity check is used to determine and correct any errors that may have occurred.

In convolutional codes, the coded sequence of n digits depends not only on the k data digits but also on the previous $N - 1$ data digits ($N > 1$). Hence, the coded sequence for a certain k data digits is not unique but depends on $N - 1$ earlier data digits. This thesis will not consider convolutional codes in any more detail and detailed information can be found in many books [e.g. Lathi, B. P., (1989); Wade, G., (1994)].

Different types of error control codes have been implemented for the analysis of magnetic recording channels. Such codes include the Hamming code [Lin, S., and Costello A.J., (1983)], array codes [Farrell, P.G., (1992)], the Fire code [Watkinson, J., (1990)], and Reed Solomon codes [Lin, S., and Costello A.J., (1983)].

The Hamming (n, k) code is a single error correcting code and is produced by combining the k data bits with $n - k$ additional check bits. These check bits are calculated from the data bits by the use of a generator matrix or polynomial.

Array codes are produced by restructuring data bits into a data array which is used to compute the check bits of the code. If the check bits are computed in both the horizontal and vertical directions of the data array, a single bit in error can be detected and corrected. To increase the number of bits that can be corrected by an array code, check bits are calculated in more than two dimensions.

The Fire code is a burst correcting code that is capable of correcting a single error burst in a code word. It operates by treating every bit separately and shifts the bits of the code word through a register that effectively multiplies them by a burst locating polynomial. Unfortunately, this means that the code is not very efficient at correcting the long and multiple bursts which can be present in magnetic recording channels.

Reed Solomon codes overcome the disadvantages of the Fire code as groups of bits are treated as information symbols. They are capable of correcting several error bursts in a code word. Interleaving is a technique associated with Reed Solomon codes, which allows for a particular code word to be transmitted at different times. Therefore any long error bursts occurring on the magnetic channel are spread over several code words.

1.4 OUTLINE OF THE THESIS

Several models have been developed that simulate the behaviour of a magnetic channel using the error function complement to produce bit error rates [Katz, E. R., and Campbell, T. G., (1979)]. This thesis examines a novel new method for describing pulses and waveforms [Tandon, A., Middleton, B.K., Farrell, P.G., and Miles, J.J. (1997)] which

allows for a bit by bit simulation analysis. In addition the effects of drop-outs and the addition of actual noise (which produces bit error rates) into the magnetic channel are analysed in detail. The flexibility of this new method allows the simulated channel to be used for different purposes. These include examining the theoretical limits of the channel, examining individual characteristics of the channel and comparing the data streams that have been modified by the behaviour of the channel. This data stream comparison is particularly useful for the analysis of error coding and decoding schemes.

This thesis concentrates on the peak detection method which has been the main stay of magnetic recording systems for many years and is a good vehicle for investigation of the various aspects of system performance. The model has the flexibility to be adapted to partial response and other advanced signal processing strategies.

Chapter two examines the characteristics of the basic magnetic recording channel in detail. Six channel codes, which are used to match the characteristics of the data with that of the recording channel, will be analysed. Longitudinal pulses, that cover all the different types of longitudinal output pulses will be fully described in terms of parameters which define the head and tape characteristics. The process of modelling the linear superposition of longitudinal pulses and a full description of data stream recovery by the use of timing windows will also be produced.

Chapter three examines how the predicted characteristics of the recording channel are compared with experiment and how the model is extended to a multi-track channel which can work with large data trains. To verify the characteristics of the recording channel, cross-over shift analysis will be produced using the no cross-over shift data pattern, the

pseudo-random data pattern and the worst case data pattern. A study into predicting performance of channels by using scaling will be performed, to demonstrate the power of the modelling and its ability to function at the theoretical limits of the peak detection channel.

Chapter four examines how defects are added into the channel. The defects examined include drop-outs and noise. Two techniques of describing noise, and their effects on bit errors, will be analysed. The first technique is a new technique that involves applying additive white gaussian noise samples onto the replay and differentiated waveforms, while the second technique is a previously published technique that involves determination of the effect of noise using statistical analysis. Confirmation of the validity of the former technique will be produced by comparing the bit error rates obtained by both the techniques. A comparison will be performed to validate the accuracy of the new technique compared with previous simulations and previous practical work. The simulation of pulse equalization and a mixed model (which contains both longitudinal and perpendicular components of magnetisation) will be described but used for comparison purposes only. The error events produced by the multi-track, drop-out infected recording channel will also be examined. A study into the performance limitations due to ISI and noise will also be produced to determine the effects of peakshift limited performance and noise limited performance in the recording channel.

Chapter five examines the effect of error performance using three interleaving strategies, these being symbol interleaving, block interleaving and a combination of both. All three strategies are analysed using horizontal and vertical stacks. A full analysis is conducted of the blocks, symbols and bits in error and the block, symbol and bit error rates. Different

interleaving depths and block sizes are investigated and the improvements in error performance for different scenarios are highlighted.

Finally, chapter six summarises the achievements of the new modelling method, and makes recommendations for further work that can be done in both the fields of system modelling and the error correction.

2 THE MAGNETIC RECORDING CHANNEL

2.1 INTRODUCTION

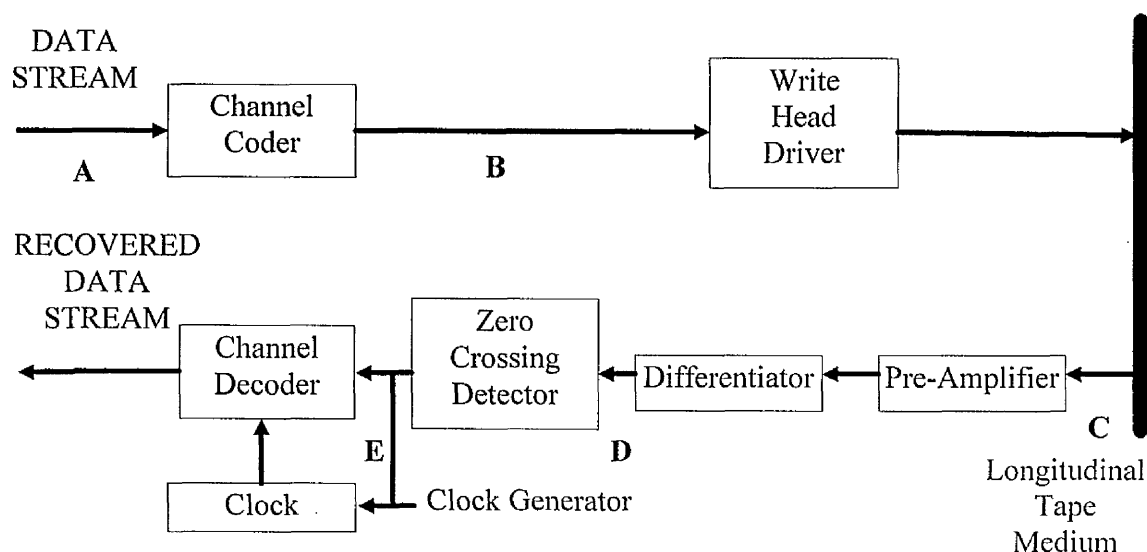


Figure 2.1: Block diagram of the magnetic recording channel

The magnetic recording channel can be represented as a series of separate modules and is shown in figure 2.1. The channel coder is used to match the characteristics of the data stream with that of the recording channel. This coded data stream is converted into a current waveform by the write-head driver, which is applied to the tape medium. The waveform is read back, at a further point in time, from the tape medium into a pre-amplifier and amplified. This process of writing pulses onto the tape medium and reading back these pulses from it is represented by a set of equations. For longitudinal recording, the equations are called the longitudinal model, and cover all types of longitudinal pulses, as long as they contain no perpendicular components.

The longitudinal pulses can then be combined at a range of packing densities, as shown in figures 2.13-2.18, using linear superposition, to produce replay and differentiated waveforms. The differentiated waveforms are used to locate the peaks of replayed pulses as zero crossing points, as these points contain bit transition information, that allow the simulation to re-create the data stream. Finally, the channel decoder is used to remove the channel code characteristics from the re-created data stream and so recover the original data stream.

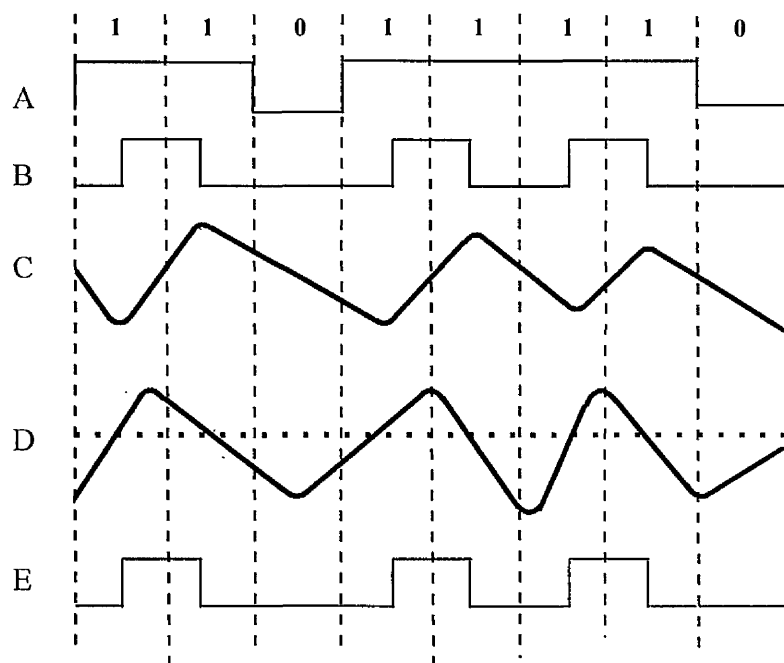


Figure 2.2: Waveforms on various parts of the channel (labelled in figure 2.1)

Figure 2.2 illustrates waveforms that occur at various parts of the channel and are labelled in figure 2.1. Waveform A is the data stream entering the channel encoder, which is then modified by the use of the delay modulation channel code (described in section 2.2.2.5), into waveform B. This waveform, in the form of a current, is applied to the tape medium. The waveform which is read back is shown in waveform C. This voltage is differentiated to

produce D and passed into a zero crossing detector to be re-created as waveform E, which as expected is a replica of B.

2.2 MATCHING DATA TO THE CHARACTERISTICS OF THE RECORDING CHANNEL USING CHANNEL CODES

Before writing data of interest onto the magnetic tape surface, this data is normally coded using a channel code. The purpose of the channel code is to match the characteristics of the data with that of the recording channel. This simulation has been designed to be flexible enough to allow a wide range of channel codes to be analysed.

2.2.1 Considerations of the Characteristics of Channel Codes

There are a number of considerations that are taken into account when determining which channel code is used in a magnetic recording channel.

(1) *Efficiency* - The storage efficiency is defined as the number of stored bits per flux reversal, and is expressed as a percentage. A value of 100% corresponds to a one bit per flux reversal.

(2) *Symbol Separation* - The symbols representing the data to be stored should be as unlike each other as possible. By doing this, it is easy to distinguish between symbols even if they are badly distorted due to defects in the recording / replay process. In a two-valued system, the symbols should be kept as far apart as possible.

(3) *Bandwidth* - The bandwidth occupied by a signal is a measure of its rate of change, and is measured in Hertz (Hz). Codes should have a spectral bandwidth matched to the recording channel. In particular, very low frequencies approaching DC should be avoided since a magnetic channel has zero gain at DC.

(4) *Self clocking* - The encoded data must eventually be decoded and separated into individual bits. A code which provides a method of splitting the bits off from one another is called self clocking and is highly desirable. For high recording densities, in particular, plenty of transitions are required for self clocking.

(5) *Read Resolution* - When decoding the coded data, a full cell length should ideally be made available for the decision making process.

(6) *Complexity* - The simpler the encoding and decoding processes are, the less they cost and the more reliable they are.

(7) *Noise Immunity* - An ideal code should have the largest immunity to extraneous signals. In magnetic recording systems noise is caused by imperfections in the magnetic coating leading to drop-outs. A drop-out is a loss of signal caused by missing magnetic material. Another source of noise is cross-talk which is the signal picked up from adjacent tracks.

2.2.2 The Channel Codes

Six common channel codes [Watkinson, J., (1990)] will be described. These codes have various characteristic advantages and disadvantages and these will be examined. The

channel codes used are: non return to zero, non return to zero one, phase encoding, frequency modulation, delay modulation and Miller squared.

2.2.2.1 The Non-Return to Zero Channel Code

In the non-return to zero (NRZ) channel code, the magnetic coating is magnetised to one or the other of the two states, depending on whether a 1 or a 0 is to be recorded. The flux changes only occur when the digital input data change from 0 to 1 or 1 to 0; thus consecutive 1's and 0's will remain at the same level. Figure 2.3 shows how NRZ codes a data pattern 1101111001110010.

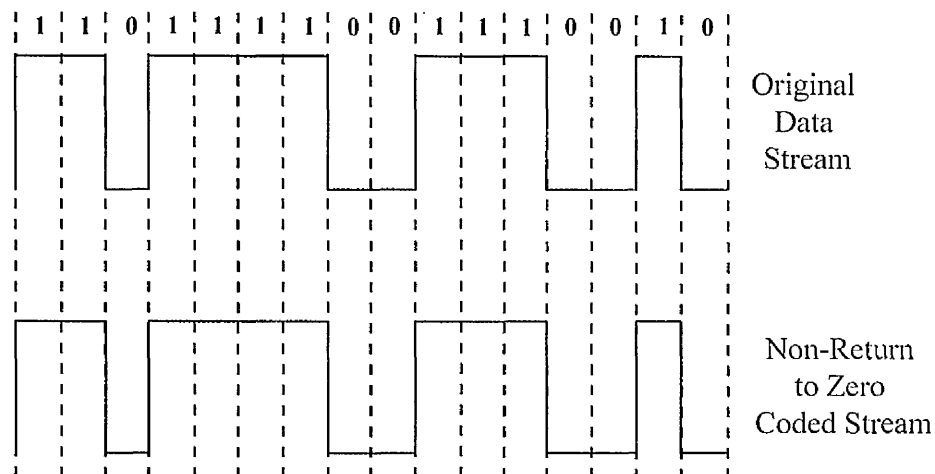


Figure 2.3: Coding a data pattern using the non-return to zero code

The NRZ method is a very efficient way (100%) of encoding binary data, and requires at most one saturation reversal (or output pulse) per bit, the maximum occurring for an alternating sequence of 1's and 0's. The NRZ code has a good read resolution, a poor

separation, requires a low frequency content, and has a fair immunity to noise. The greatest drawback with NRZ is that it is not self clocking.

2.2.2.2 The Non-Return to Zero One Channel Code

In the non-return to zero one (NRZ1 or NRZI) channel code, the current is reversed producing a flux change every time a 1 is recorded, and no flux change is generated for a 0; that is, the system changes polarity every time a 1 is recorded. Figure 2.4 shows how NRZ1 codes a data pattern *1101111001110010*.

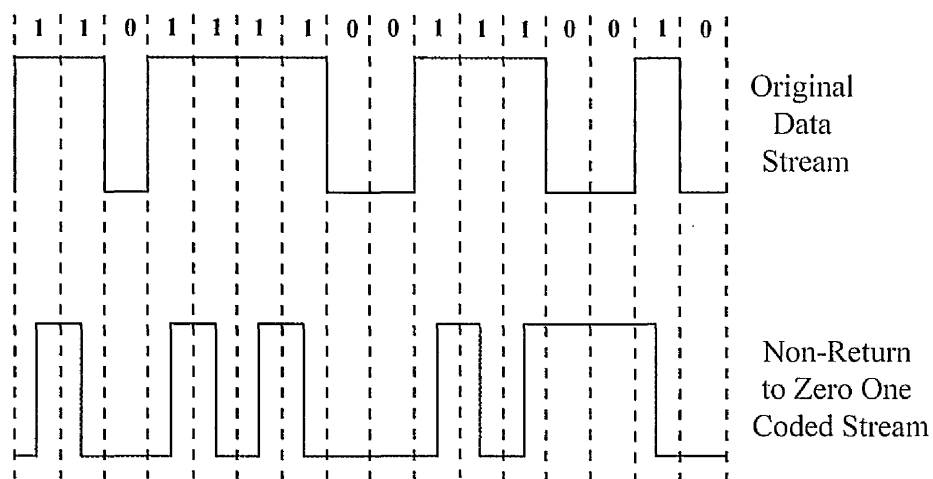


Figure 2.4: Coding a data pattern using the non-return to zero one code

The NRZ1 method is a modified version of the NRZ code and is very efficient in encoding binary data, has a good read resolution, a poor separation, requires a low frequency content, has a fair immunity to noise and is not self clocking. The advantage of the NRZ1 code over NRZ is that if a bit is misread only that bit is in error; whilst with NRZ, if a bit is in error, all succeeding bits will be corrupted until the next signal pulse is read.

2.2.2.3 The Phase Encoding Channel Code

In the phase encoding (otherwise known as Manchester) channel code, half of the bit cell is magnetised in opposite directions. A 0 is written by a change from a low to a high and a 1 is written by a change from a high to a low, in the middle of the bit cell. Figure 2.5 shows how phase encoding codes a data pattern 1101111001110010.

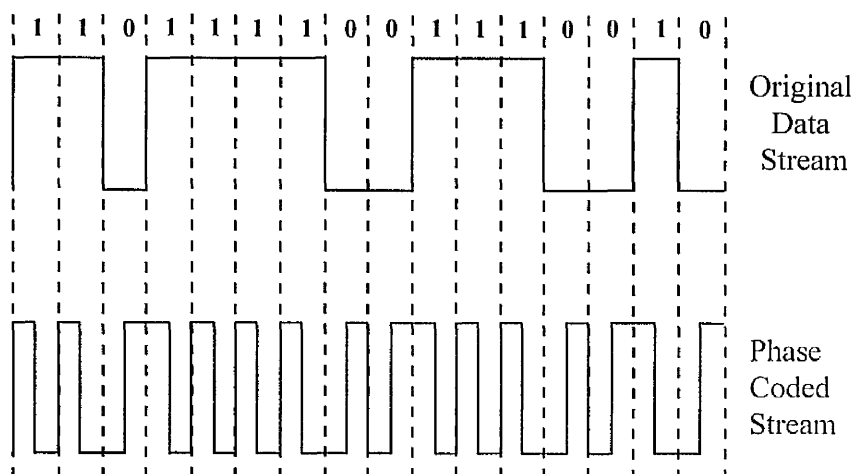


Figure 2.5: Coding a data pattern using phase encoding

Since both binary 1 and 0 provide output pulses, there is at least one output pulse per bit period, thereby allowing a clocking signal to be generated from the output and so the system is self clocking. Phase encoding has a low efficiency (50%) because a maximum of two transitions per bit are required. The read resolution is also poor. The separation is 100% because there is a maximum difference between 1's and 0's. The bandwidth requirements are good because there is no low frequency content in the recorded signal. However, as there are two flux transitions per bit, the maximum recorded frequency is

twice that of NRZ1 code at an equivalent bit density. The complexity is greater than that of NRZ1, but has a good immunity to noise.

2.2.2.4 The Frequency Modulation Channel Code

In the frequency modulation (FM) channel code, a flux change always occurs at the boundary of the magnetic cells, with a *1* causing a flux change in the mid-cell position and no change is produced for a *0*. Figure 2.6 shows how FM codes a data pattern *1101111001110010*.

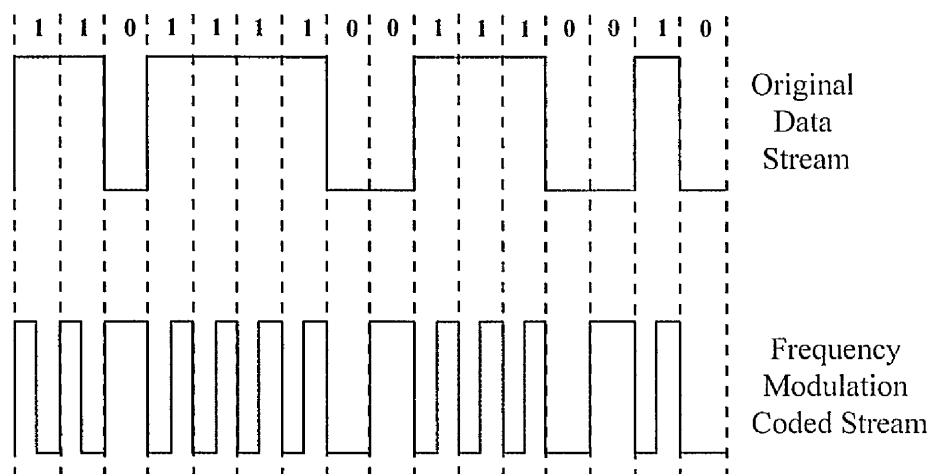


Figure 2.6: Coding a data pattern using the frequency modulation code

The frequency modulation method has identical properties to phase encoding and therefore it is self clocking, has 100% separation, a poor read resolution, good bandwidth requirements, no low frequency content, and has a good noise immunity. It is also twice the recorded frequency and is of higher complexity than NRZ1.

2.2.2.5 The Delay Modulation Channel Code

In the delay modulation (also known as modified frequency modulation (MFM)) channel code, a 1 and 0 correspond to the presence or absence, respectively, of a transition in the centre of the corresponding bit cell. However, in this case, transients at the cell boundaries only occur between bit cells that contain consecutive zeros. Figure 2.7 shows how delay modulation codes a data pattern *1101111001110010*.

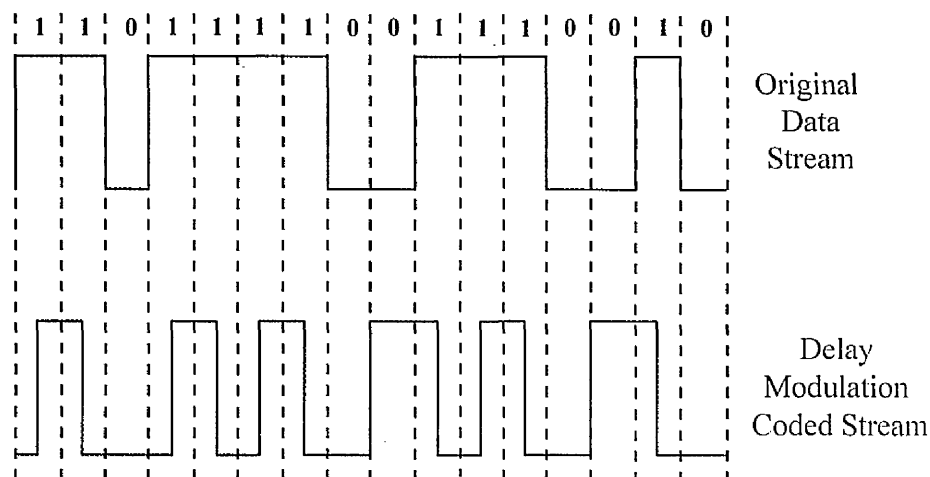


Figure 2.7: Coding a data pattern using the delay modulation code

The delay modulation code has identical properties to frequency modulation except that it is 100% efficient, but is more complex than FM and has a low frequency content. Thus, it is self clocking, has a poor read resolution, a good noise immunity and good bandwidth requirements.

2.2.2.6 The Miller Squared Channel Code

In the Miller squared channel code, a 1 and 0 correspond to the presence or absence, respectively, of a transition in the centre of the corresponding bit cell. However, in this case, transients at the cell boundaries only occur between bit cells that contain consecutive zeros. Additionally, if an even number of 1's occur then the last transition within them is omitted. Figure 2.8 shows how Miller squared codes a data pattern 1101111001110010.

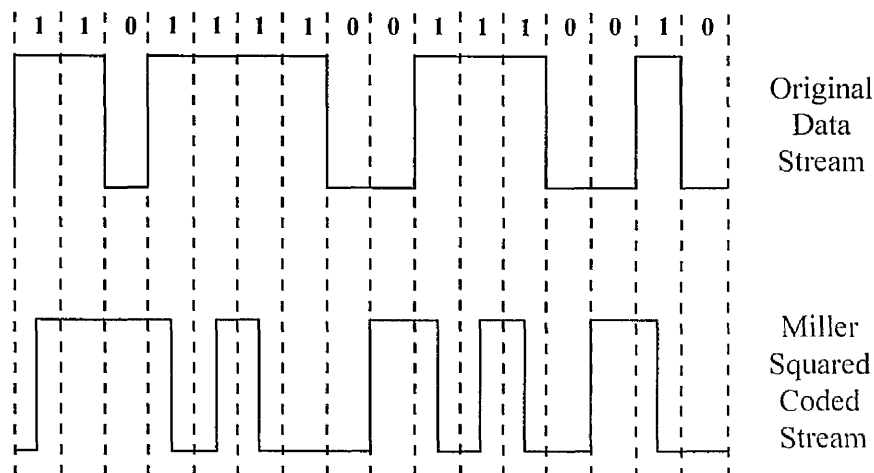


Figure 2.8: Coding a data pattern using the Miller squared code

The Miller squared channel code has the same properties as delay modulation code except that it has no low frequency content and is more complex than the delay modulation code. Thus, the Miller squared code is self clocking, has a low efficiency, a poor read resolution, good bandwidth requirements, is 100% efficient and has a good noise immunity.

There is an ambiguity, however, with the Miller squared code. If the data pattern ends in 10 or 11, it is decoded as 10. To illustrate this, the data patterns 1011 and 1010 coded using

Miller squared decode as *1010*. The method used to overcome this difficulty is for the simulation to add an extra zero at the end of the data pattern. Thus *1011* and *1010* become *10110* and *10100* respectively. After decoding, *10110* and *10100* are produced and once the extra zero is removed, become *1011* and *1010* respectively. Therefore the pattern has been recorded and replayed accurately.

2.3 THE LONGITUDINAL MODEL

The longitudinal model is used as it is an accurate representation of the longitudinal components of the digital recording channel. Major work on the longitudinal model was done in "*Output Waveforms in the Replay Process in Digital Magnetic Recording*" [Middleton, B. K., Wright, C. D., Cumpson, S. R., and Miles, J. J., (1995)]. Equation 14 of this paper provides a good description of the longitudinal recording channel.

By substituting D from the paper with d , δ with $D(1 + \alpha)$ and a_0 with a_l a more standardised version of the model description is produced. Also the effect of $(1 + \alpha)$ is fully

considered. Finally, $\left(\frac{H_g}{i}\right)$ is replaced with $\left(\frac{n\eta}{2g}\right)$ and since $\theta = 0$, $M_0 \cos \theta$ is replaced with

M_x . This produces equation (2.1) which describes the longitudinal model replay pulses.

$$e(x) = \frac{\mu_0 v \omega M_k}{\pi(1+\alpha)} \left(\frac{m}{2g} \right) \left[\begin{aligned} & 2(a+d+D(1+\alpha)) \left[\arctan \left(\frac{g+x}{a+d+D(1+\alpha)} \right) + \arctan \left(\frac{g-x}{a+d+D(1+\alpha)} \right) \right] \\ & - 2(a+d) \left[\arctan \left(\frac{g+x}{a+d} \right) + \arctan \left(\frac{g-x}{a+d} \right) \right] \\ & + (g+x) \ln \left[\frac{(a+d+D(1+\alpha))^2 + (g+x)^2}{(a+d)^2 + (g+x)^2} \right] \\ & + (g-x) \ln \left[\frac{(a+d+D(1+\alpha))^2 + (g-x)^2}{(a+d)^2 + (g-x)^2} \right] \end{aligned} \right] \quad (2.1)$$

With the parameters defined as:

- x = (head / medium) speed * time
- a_t = the recorded transition width at the top surface of the medium
- d = (head / medium) separation on replay
- D = the lesser of medium thickness and depth of recording
- α = the rate of change of transition width with depth into the medium
- μ_0 = the permeability of free space
- ω = the track width
- v = the tape velocity
- M_x = the peak magnetisation of the medium (longitudinal component)
- η = the replay head efficiency
- n = the number of turns of the replay head
- $2g$ = the gap length

Equation (2.1) is differentiated to produce the output from the differentiator and is described in equation (2.2).

$$\frac{d(e(x))}{dx} = K \left[\begin{aligned} & 2 \left[\left(\frac{(a+d+D(1+\alpha))^2}{(a+d+D(1+\alpha))^2 + (g+x)^2} \right) - \left(\frac{(a+d+D(1+\alpha))^2}{(a+d+D(1+\alpha))^2 + (g-x)^2} \right) \right] \\ & - 2 \left[\left(\frac{(a+d)^2}{(a+d)^2 + (g+x)^2} \right) + \left(\frac{(a+d)^2}{(a+d)^2 + (g-x)^2} \right) \right] \\ & + 2 \left[\left(\frac{(g+x)^2}{(g+x)^2 + (a+d+D(1+\alpha))^2} \right) - \left(\frac{(g+x)^2}{(g+x)^2 + (a+d)^2} \right) \right] \\ & + \ln((g+x)^2 + (a+d+D(1+\alpha))^2) - \ln((g+x)^2 + (a+d)^2) \\ & - 2 \left[\left(\frac{(g-x)^2}{(g-x)^2 + (a+d+D(1+\alpha))^2} \right) + \left(\frac{(g-x)^2}{(g-x)^2 + (a+d)^2} \right) \right] \\ & - \ln((g-x)^2 + (a+d+D(1+\alpha))^2) + \ln((g-x)^2 + (a+d)^2) \end{aligned} \right] \quad (2.2)$$

where $K = \frac{\mu_0 v \omega M_x}{500 \pi (1+\alpha)} \left(\frac{n\eta}{2g} \right)$

A gain factor of 1/500 has been included in equation (2.2). The reason for this is when considering a hardware differentiator circuit a gain is normally built into the circuitry. Equation (2.3) is used to model the gain factor in a simulation, with V_o being the output voltage of the differentiated waveform and V_i being the output voltage of the replay waveform.

$$V_o = \frac{1}{RC} \frac{dV_i}{dt} \quad (2.3)$$

Equation (2.3) can be re-written, in terms of distance in equation (2.4)

$$V_o = \frac{v}{RC} \frac{dV_i}{dx} \quad (2.4)$$

Putting arbitrary values of $R = 1.9 \text{ M}\Omega$ and $C = 100 \text{ }\mu\text{F}$ into equation (2.4) produces equation (2.5).

$$V_o = \frac{0.381}{1.9 * 10^6 * 100 * 10^{-6}} \frac{dV_i}{dx} = \frac{1}{500} \frac{dV_i}{dx} \quad (2.5)$$

Thus the gain factor for a tape running at 15 inches per second is 1/500. The example channel parameters shown in table 1 are comparable with advanced disk or very modern thin film tape and represent the leading edge of what is currently available.

PARAMETER	VALUE
$(a_t + d)$	$8.4 * 10^{-8}$ metres
D	$2.5 * 10^{-8}$ metres
μ_o	$4 \pi * 10^{-7}$ henries / metre
ω	$1.81 * 10^{-3}$ metres
v	0.381 metres / sec
M_x	$3.86 * 10^5$ amps / metre
$\eta * n$	18
α	0 degrees
2g	$2 * 10^{-10}$ metres

Table 1: The set of parameters defining the recording channel

Note: The small value of 2g is used to represent the narrow gap approximation.

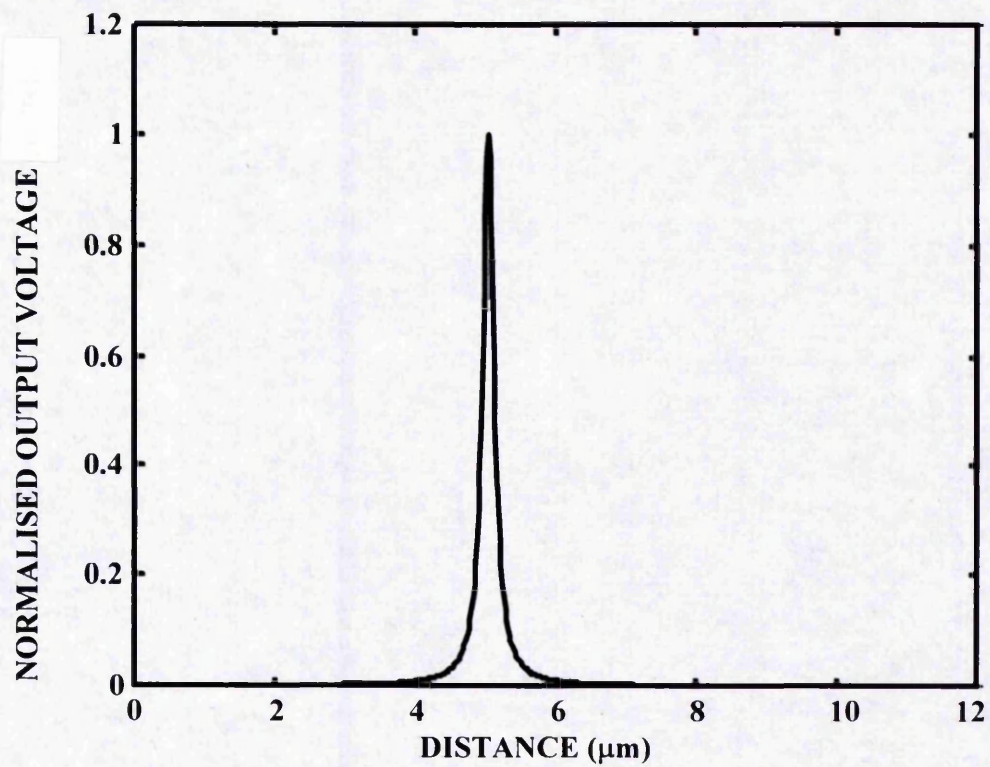


Figure 2.9: The longitudinal model replay pulse (shifted by 5 μm)

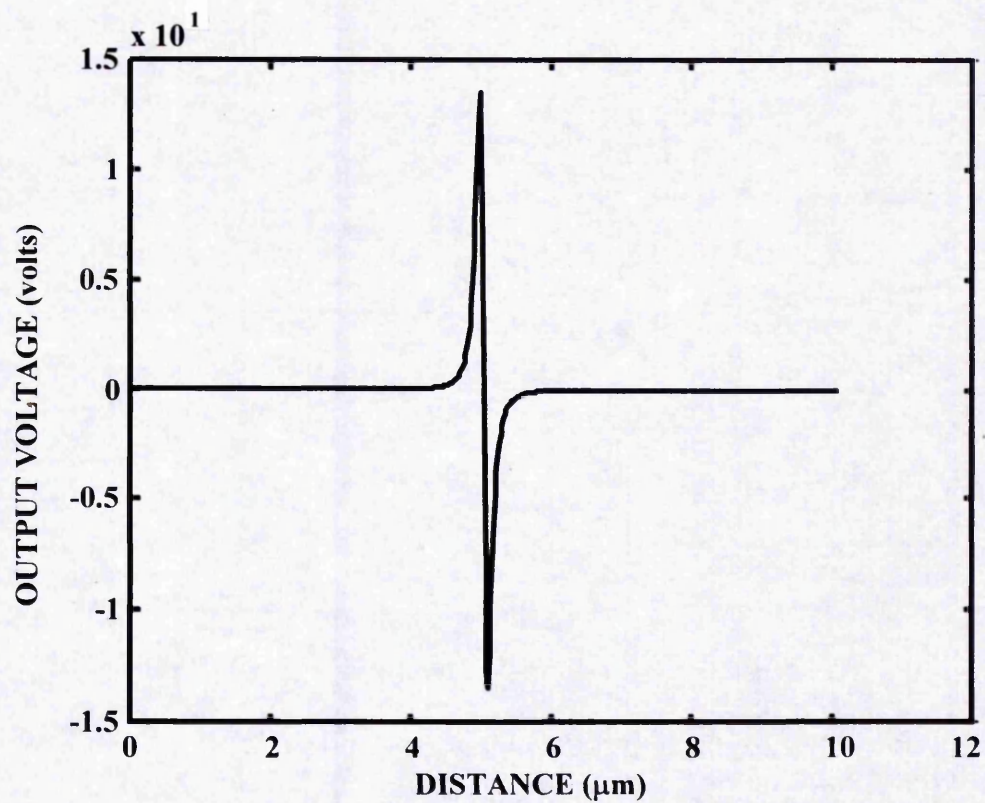


Figure 2.10: The longitudinal model differentiated pulse (shifted by 5 μm)

Figures 2.9 and 2.10 show the isolated pulses defined by equation (2.1) and equation (2.2) respectively using the set of parameters defined in table 1.

An important consideration in the simulation of both the replay and differentiated pulses is the number of samples used to define a pulse. For a wide range of packing densities to be analysed, 3,500 samples per pulse are required typically over a distance of 40 ($a_t + d$). There is no restriction on the shape of pulse used, be it longitudinal, perpendicular, a combination of both orientations of magnetisation (mixed model), or alternatively user defined pulse shapes. To illustrate this, figures 2.11 and 2.12 show an example mixed model replay and differentiated pulse respectively and the mixed model is described in detail in section 4.6. The pulse asymmetry seen in these two figures is caused by the perpendicular components introduced by the mixed model.

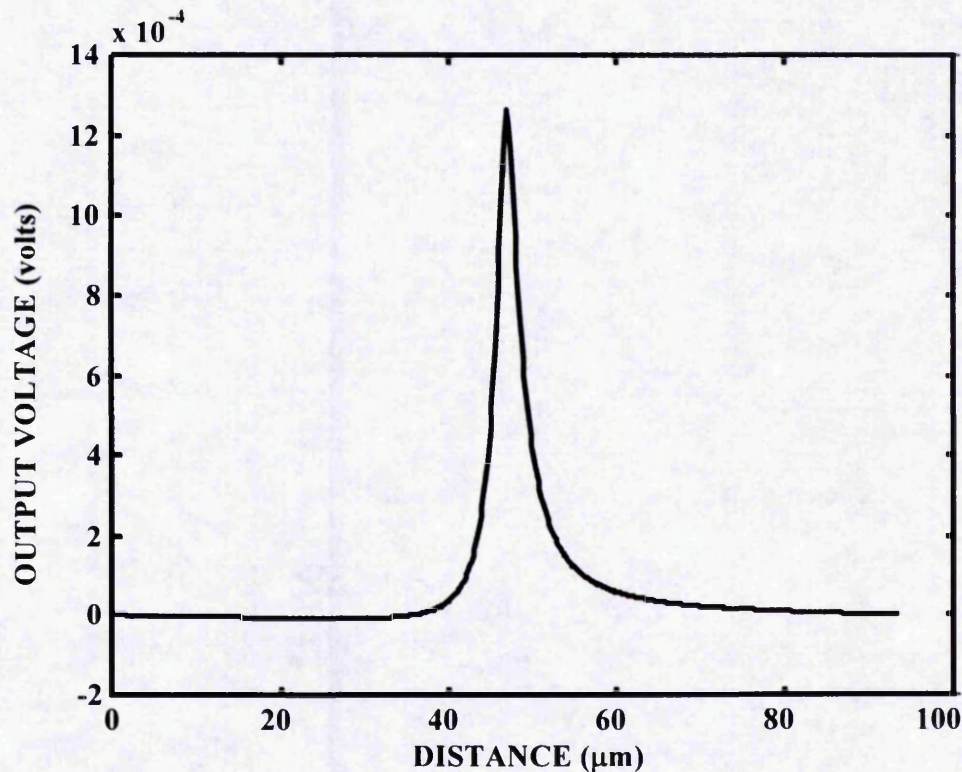


Figure 2.11: An example mixed model replay pulse (shifted by 50 μm)

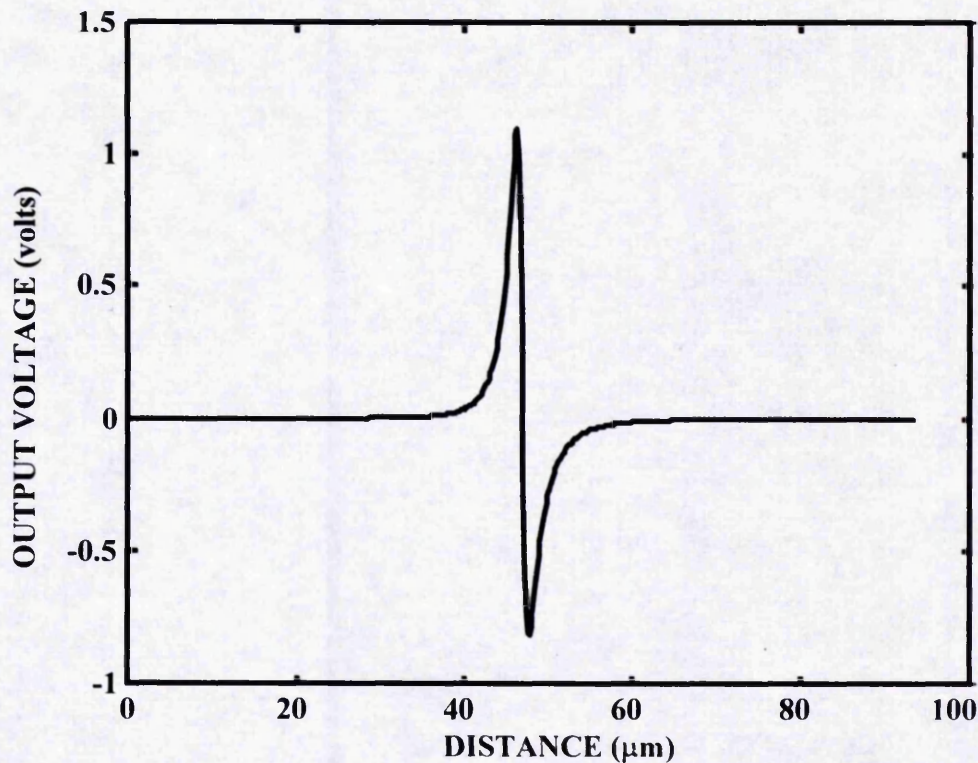


Figure 2.12: An example mixed model differentiated pulse (shifted by 50 μm)

2.4 WAVEFORM GENERATION USING LINEAR SUPERPOSITION

To simulate the behaviour of a digital tape system, the isolated pulses have to be superimposed accurately to produce the output waveform. The method used to model this behaviour is linear superposition [Hoagload, A. S., (1963); Mallinson, J. C., and Steele, C. W., (1969); Tjaden, D. L. A., (1973)]. This is stated in *"Superposition Based Analysis of Pulse Slimming Technique for Digital Recording"* [Mackintosh, N. D., (1980)] as:

At all packing densities for which the read-back process is linear, the net flux in the read-coil from any pattern of surface flux reversal is the algebraic sum of the individual flux contributions from each reversal acting on its own.

To determine the positions that pulses are placed at equation (2.6) is used.

$$\text{number of samples between two bits} = \frac{\text{spacing between two bits}}{\text{step size}} \quad (2.6)$$

with

$$\text{spacing between two bits} = \frac{1}{\text{packing density}} * 0.0254 \quad (2.7)$$

and

$$\text{step size} = \frac{2((a_t + d)) * \text{pulse width multiplier}}{\text{number of samples per pulse}} \quad (2.8)$$

where the spacing between two bits and the step size are in metres and the packing density is in bits per inch. The step size is defined as the distance between two samples. The pulse width multiplier is a multiplication factor that is applied to produce a long pulse tail length. This long tail ensures that an accurate linear superposition process occurs. For the most flexible model, the pulse width multiplier has a value of 20. Typically, this is equivalent to approximately 150 samples per $(a_t + d)$. By defining the waveforms in terms of samples, extra flexibility has been introduced, with a wide range of packing densities able to be simulated without a significant time penalty. Packing density is measured in the industrial standard unit of bits per inch. Figures 2.13 and 2.14 illustrate the replay and differentiated waveforms of the longitudinal model for pulses that are generated at 20,000 bits per inch, using a delay modulation coded pseudo-random data stream. The individual pulses can clearly be seen as a low packing density is being used.

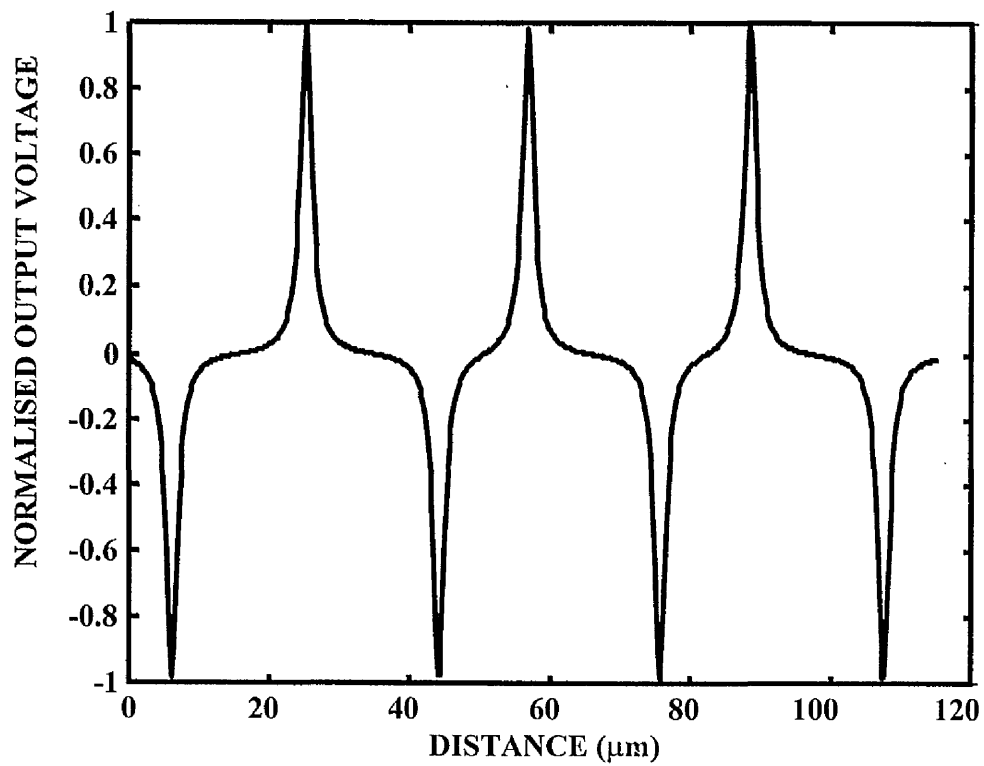


Figure 2.13: The longitudinal replay waveform at 20,000 bits per inch

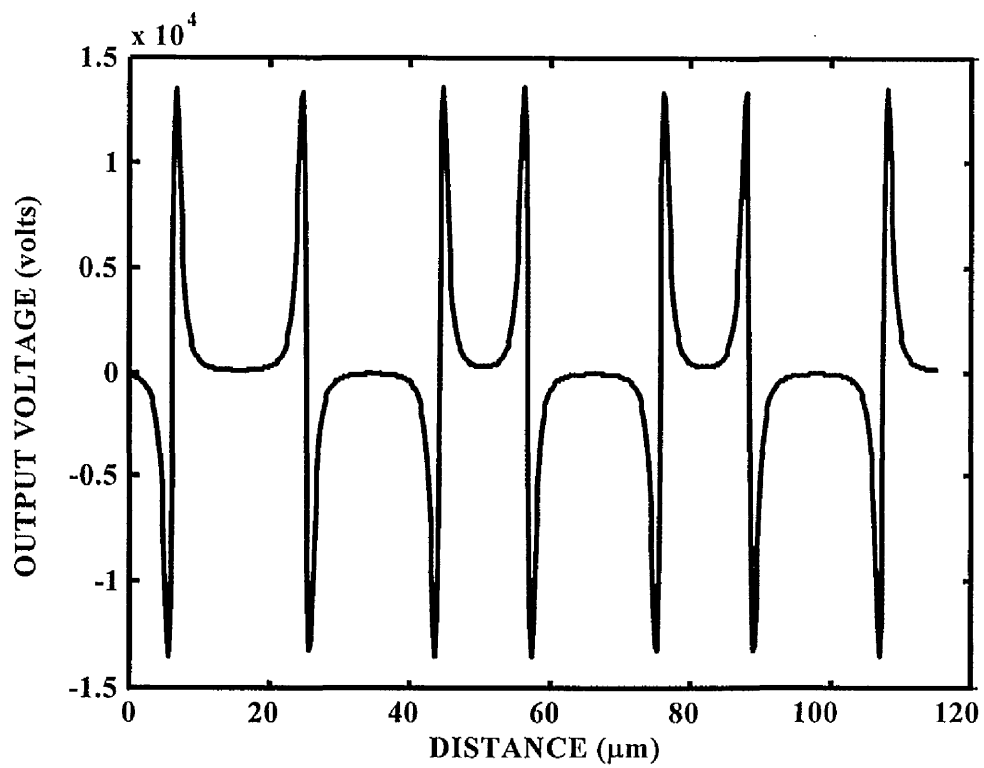


Figure 2.14: The longitudinal differentiated waveform at 20,000 bits per inch

Figures 2.15 and 2.16 illustrate the replay and differentiated waveforms of the longitudinal model for pulses that are generated at 100,000 bits per inch, using a delay modulation coded pseudo-random data stream. The superposition of pulses can clearly be seen.

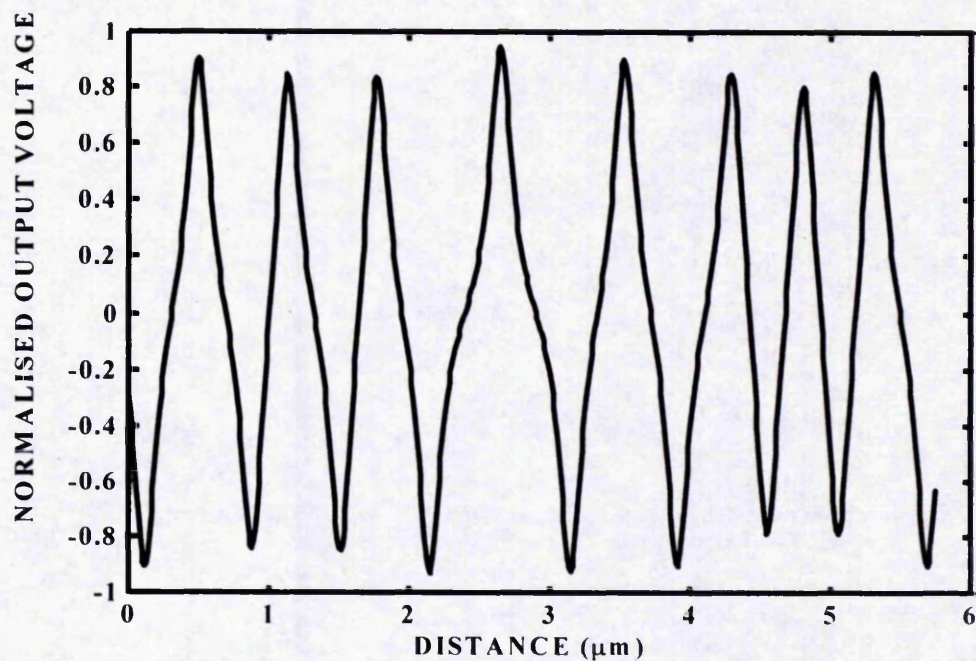


Figure 2.15: The longitudinal replay waveform at 100,000 bits per inch

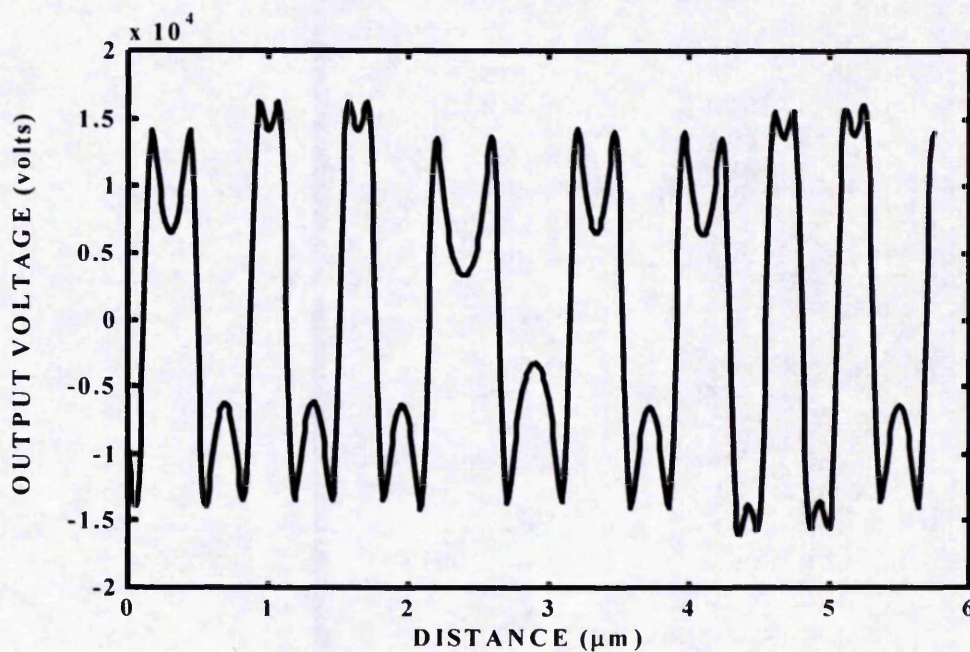


Figure 2.16: The differentiated longitudinal waveform at 100,000 bits per inch

Figures 2.17 and 2.18 illustrate the replay and differentiated waveforms of the longitudinal model for pulses that are generated at 160,000 bits per inch, using a delay modulation coded pseudo-random data stream with no cross-over shift effects. These waveforms are sinusoidal in nature and so demonstrate that the simulation functions at high packing densities and produces expected waveforms [Li Hui Hong, J.K. et al. (1993)].

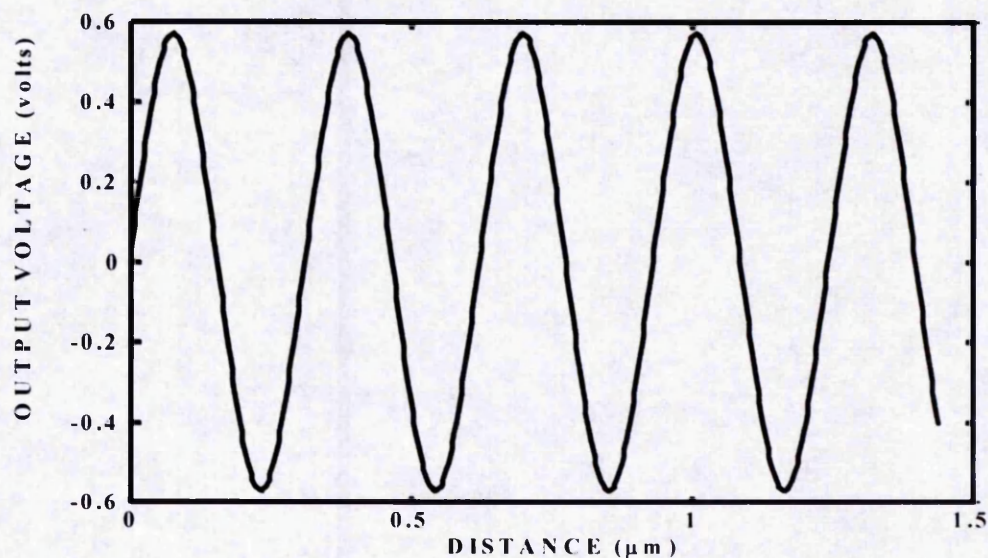


Figure 2.17: The replay longitudinal waveform at 160,000 bits per inch

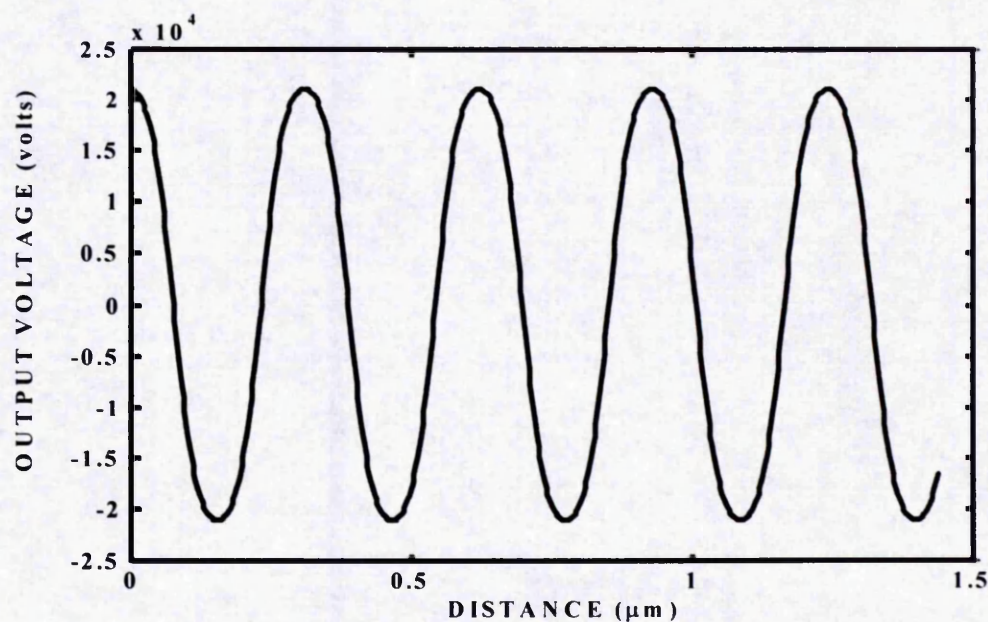


Figure 2.18: The differentiated longitudinal waveform at 160,000 bits per inch

2.5 RECOVERY OF THE DATA STREAM USING TIMING WINDOWS

WINDOWS

To recover the data stream information from the waveform produced at the output of the differentiator, the positions of the zero crossing points are required. The gradients of the waveform at the zero crossing points provide the information required to re-generate the data stream influenced by the magnetic recording channel. The effect of cross-over shift, however, can cause movement of a zero crossing point away from the position it would otherwise have been. This movement is investigated by analysing cross-over shift within timing windows .

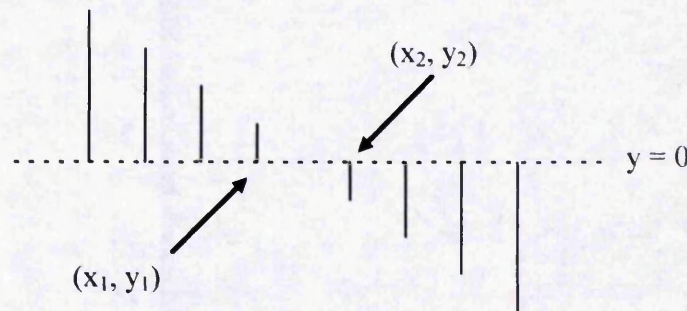


Figure 2.19: A sampled representation of a zero crossing point

In figure 2.19, a representation of a zero crossing point is shown. By defining (x_1, y_1) as the sample to the left of the zero crossing point, (x_2, y_2) as the sample to the right of the zero crossing point and $(x_{\text{zero}}, 0)$ as the zero crossing point sample, equation (2.9) is produced to calculate the value of x_{zero} and therefore accurately find the zero crossing point.

$$x_{\text{zero}} = x_1 + \left(\frac{|y_1|}{|y_1| + |y_2|} \right) \quad (2.9)$$

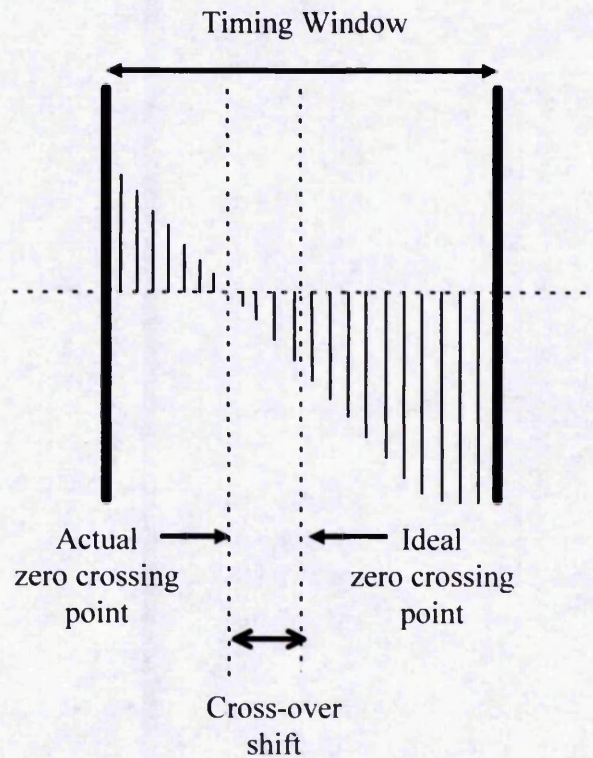


Figure 2.20: A cross-over shifted point within a timing window

Timing windows determine the limits of cross-over shift that can occur. In particular, if the cross-over shift causes pulses to be shifted outside of a timing window then the erroneous situation, that is of interest, occurs. Detailed examination of cross-over shift behaviour is therefore required. Figure 2.20 shows a cross-over shift occurring inside a timing window.

The actual zero crossing point is calculated in terms of its position within the timing window, with reference to the position of the ideal zero crossing point. If the actual zero crossing point is within the timing window, the gradient of the slope provides the bit transition information required for decoding the bit pattern. If the gradient of the slope is in the downward direction then the decoded transition is from a 0 to a 1. If the gradient of the slope is in the upward direction then the decoded transition is from a 1 to a 0. Otherwise the recreated bit is assigned the same state as the previous decoded bit.

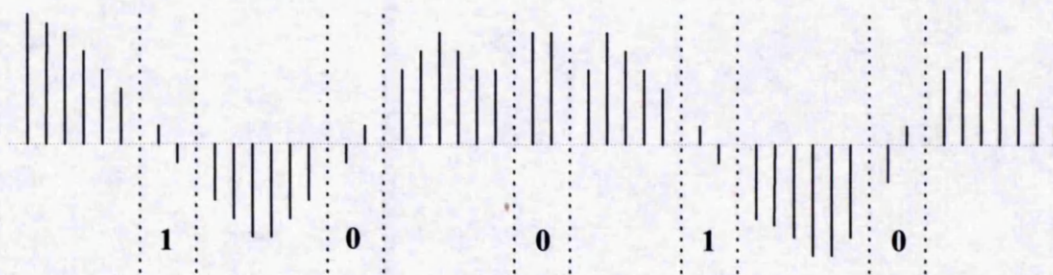


Figure 2.21: A differentiated waveform including several timing windows

Figure 2.21 illustrates a differentiated waveform that contains several timing windows, and it is clear that the data stream is recovered accurately. This recovered data stream can now be compared with the data stream going into the write head driver to determine the errors that have occurred due to the magnetic channel.

2.6 THE EFFECTS OF ERRORS ON TIMING WINDOWS

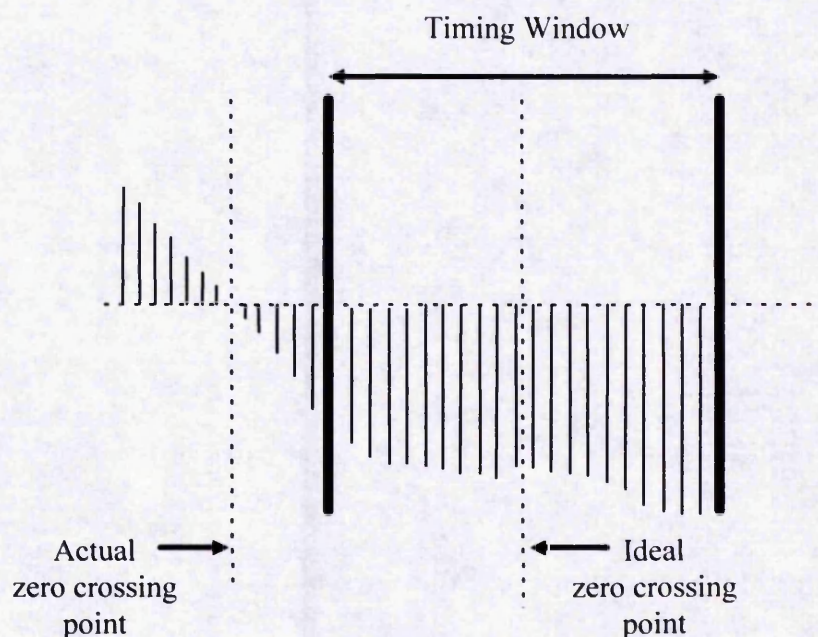


Figure 2.22: The timing window of a bit in error

Figure 2.22 shows a timing window where the zero-crossing point has been pushed outside of its timing window and therefore causes a bit in error. The two parameters that directly affect the creation of erroneous bits are signal to noise ratio (SNR) and cross-over shift (T_{ps}). The relationship between these two parameters and error rate (P_e) is [Middleton, B. K., and Jack-Kee, T., (1983)]:

$$P_e = \frac{1}{\sqrt{\pi}} \left[\frac{1}{(\text{SNR}) \sin[\omega_s(T_w - T_{ps})]} \right] \exp\left(-(\text{SNR})^2 \sin^2[\omega_s(T_w - T_{ps})]\right) \quad (2.10)$$

where ω_s is the angular frequency corresponding to the highest possible flux transition density and $2T_w$ is the timing window. Equation (2.10) indicates that the error rate increases as the signal to noise ratio decreases or the cross-over shift increases.

2.7. SUMMARY

An alternative method of describing pulses and waveforms, in peak detection systems, has been described in this chapter and compared with previous work [Li Hui Hong, J. K., (1993)]. This method describes pulses and waveforms, defined using tape and head characteristics, in terms of samples. Any pulse shape can be analysed, be it longitudinal, perpendicular, a combination of both orientations of magnetisation or alternatively user defined pulses. These pulses are combined, using linear superposition, to simulate the behaviour of a tape system. It has been shown that a wide range of packing densities are modelled accurately and so this pulse sampling method is a valid technique to use.

A new modelling technique, described in section 4.5, involves applying additive white gaussian noise samples to the sampled differentiated waveform. This technique requires data stream recovery to occur as a fundamental part of its functionality. So, the data stream recovery process has been examined in detail to illustrate how data bits can be accurately regenerated from the sampled differentiated waveform, using timing windows.

Additionally, flexibility was introduced into the simulation with the characteristics of six common channel codes examined in detail. For the rest of this thesis the only channel code which will be considered is the delay modulation channel code. This code has been chosen as comparison with previous studies is possible, and so the accuracy of the new modelling technique can be verified.

3 ADVANCED CHANNEL SIMULATIONS

3.1 INTRODUCTION

To assess the performance of the magnetic recording system, magnetic recording channel characteristics are required. These characteristics are analysed either in terms of each bit in a data pattern which has been recorded, or in terms of a particular aspect of the channel over a large packing density range. The two aspects of the channel that are of particular interest are the output amplitude behaviour and the cross-over shift behaviour. Various methods of examining these two aspects will now be presented. For this section, the simulation is carried out using the set of parameters defined in table 1.

The model can also be extended by increasing the number of tracks that can be simulated, and by allowing the simulation to handle large data trains. Large data trains are required for modelling the channel using the new technique where sampled additive white gaussian noise is added to the distortion free channel.

When examining a finite number of bits in a data pattern, an extra phenomenon has to be allowed for. Two large pulses occur at the extremes of the data pattern due to the data pattern finishing abruptly and are known as end effects. This is a limitation of the theory of linear superposition and end effect buffers are used to mask the invalid end effect information. These buffers contain a delay modulation coded string of 1's, and have a size of 32 bits. End effects are also produced in practical recording systems with preambles and post ambles used to overcome end effects.

3.2 THE CHARACTERISTICS OF THE RECORDING CHANNEL

3.2.1 Maximum Output Voltage Analysis

With an increase in the packing density, the maximum output voltage decreases. So it is important to assess the performance of the channel in the higher packing density range. Figures 3.1 and 3.2 show the output voltage and log (output voltage) for the general case model, at a range of packing densities for a leading edge tape. This output is generated using a data stream that contains a string of 1's coded using the delay modulation code, and so contains no cross-over shift effects.

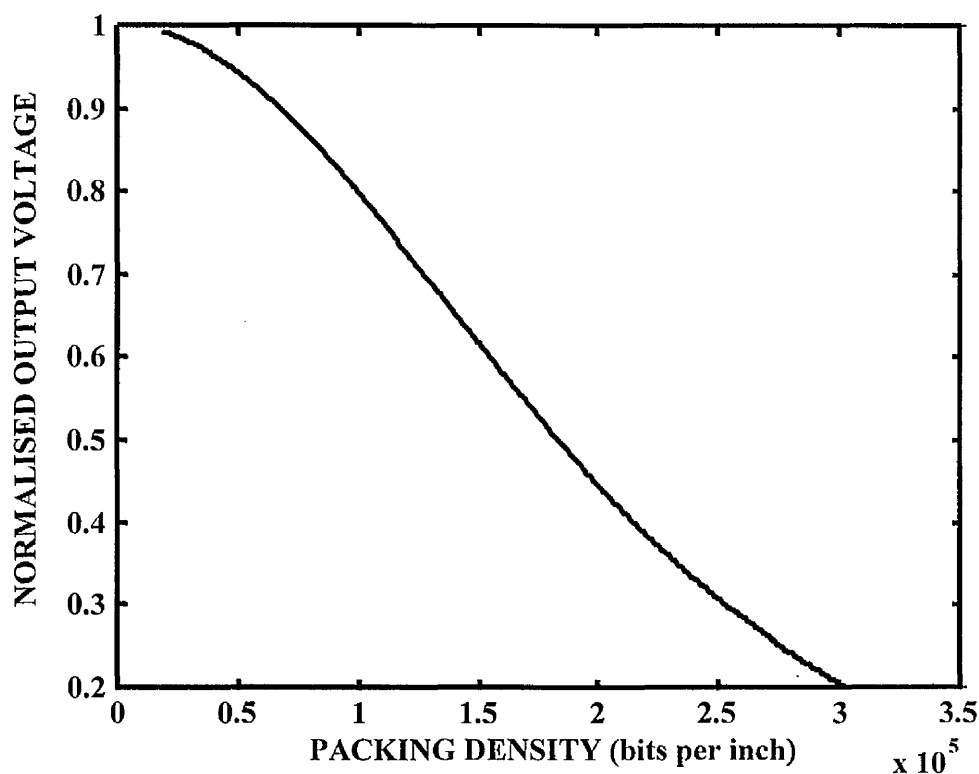


Figure 3.1: Maximum output voltage over a wide range of packing densities

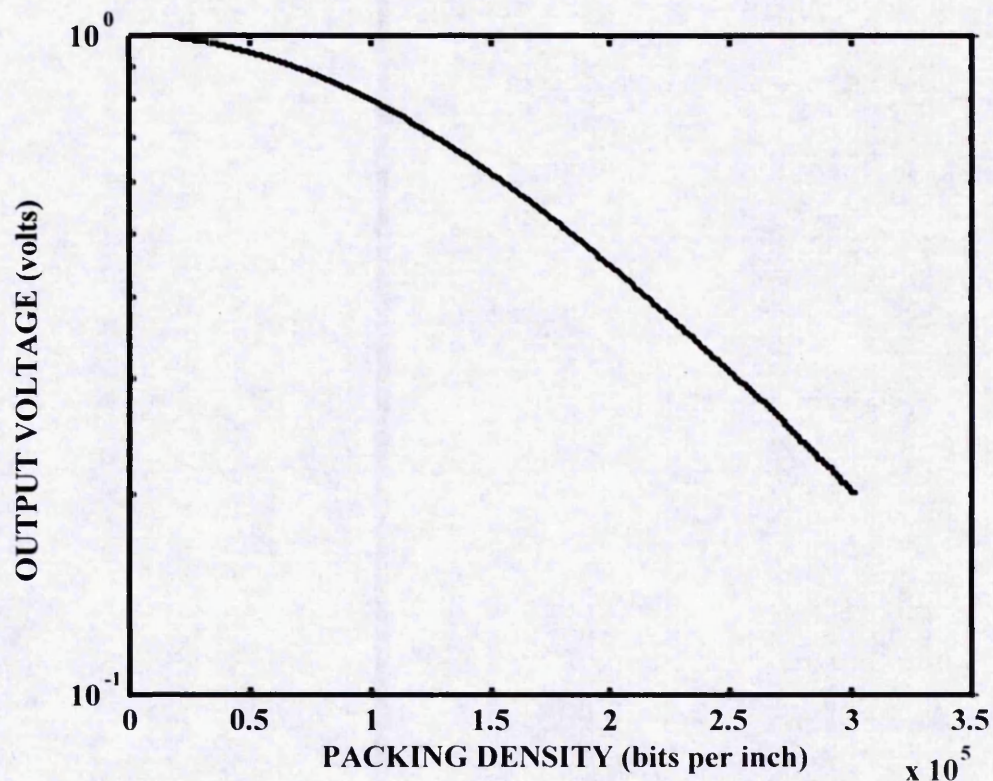


Figure 3.2: Maximum output voltage (log scale) over a wide range of packing densities

3.2.2 Cross-over Shift Analysis

The other important aspect, when considering channel performance, is cross-over shift effects. Three data patterns are used to examine the magnetic channel's cross-over shift performance. These three patterns are the no cross-over shift data pattern, the worst case data pattern and the pseudo-random data pattern.

The no cross-over data pattern is used as a check that no invalid cross-over shift occurs. This is achieved by the modelling of a square wave pattern of 0's and 1's, e.g. 0110011001100110... . A number of well spaced positive and negative pulses occur at the

transition points of the square wave pattern, and when modelled, negate each other. Therefore, no cross-over shift occurs in the model.

The worst case data pattern is used to produce the maximum amount of cross-over shift that can be produced in the channel. This is achieved by modelling the following data pattern 0110110110110110110... . A number of closely positioned positive and negative pulses occur at the transition points of the data pattern, and when modelled interact with each other. This interaction causes the maximum cross-over shift that is possible in the recording channel

The pseudo random data pattern, which has been delay modulation coded, is used to mimic the behaviour of a recording channel. Pseudo random data patterns are produced using random number generators that have good statistical properties and therefore produce statistically independent random numbers for long periods, in a feasible time scale. The method used to produce random numbers is called the Box, Muller and Marsaglia technique [Press, W. H., et al, (1994)] and is investigated in detail in section 4.3.

For these patterns, we examine the effect of the cross-over shift on each data bit in the data pattern. To examine the overall effect of cross-over shift, the average cross-over shift is calculated over a large packing density range for the worst case and pseudo-random data patterns. For the measurements in figures 3.3 - 3.7, the cross-over shift from the center of the timing window is analysed.

3.2.2.1 Cross-over Shift Effects on a Symmetrical Data Pattern

A symmetrical (or no cross-over shift) data pattern is produced using a string of *I*'s coded using the delay modulation code. Figure 3.3 shows the general case model's channel response to this data pattern, at a packing density of 100,000 bits per inch.

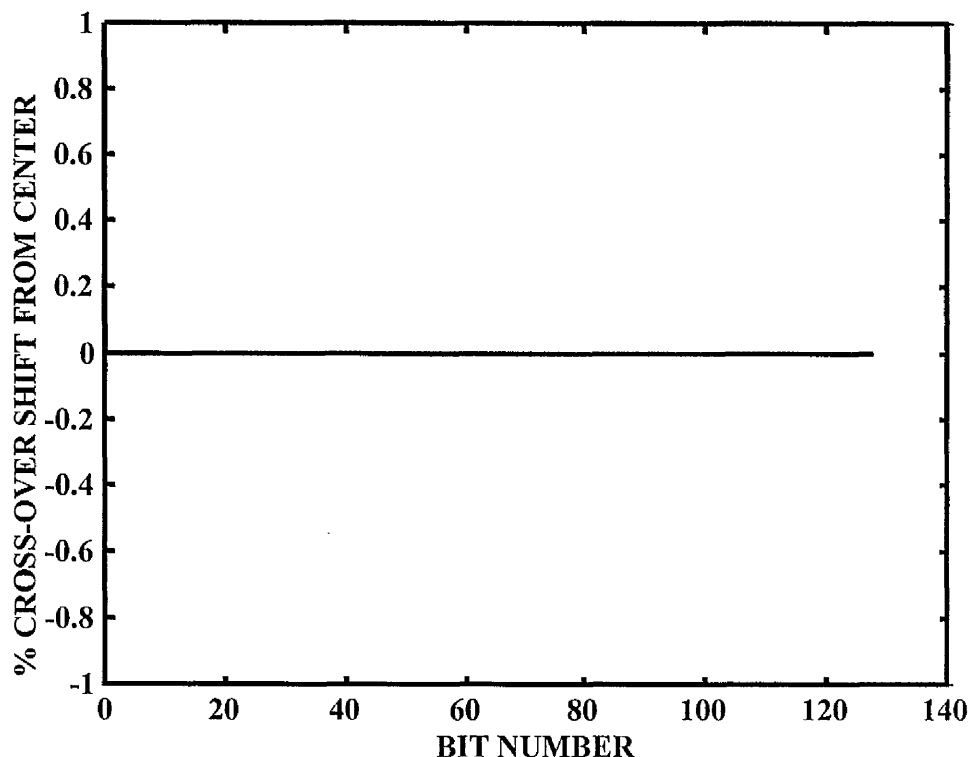


Figure 3.3: The no cross-over shift data pattern at 100,000 bits per inch

3.2.2.2 Cross-over Shift Effects on a Worst Case Data Pattern

A worst case data pattern is produced by using a string of *I*'s over the end effect bits and the data pattern *011* repeated over the valid bit range. This data is then coded using the delay modulation code. Figure 3.4 shows the general case model's channel response to this data pattern, at a packing density of 100,000 bits per inch.

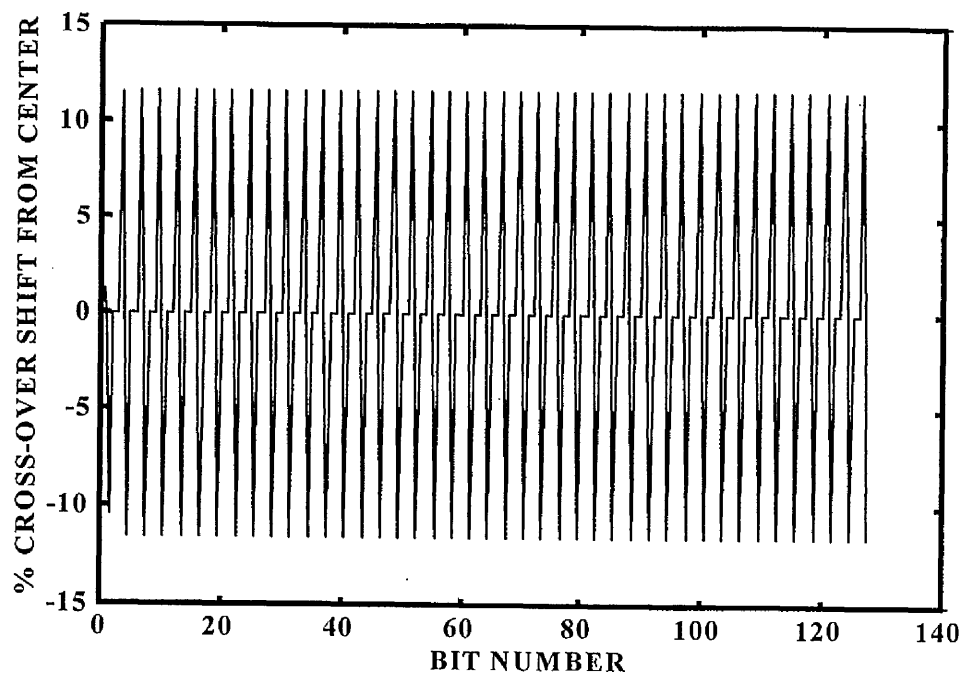


Figure 3.4: Cross-over shift that occurs in the worst case data pattern as a percentage at
100,000 bits per inch

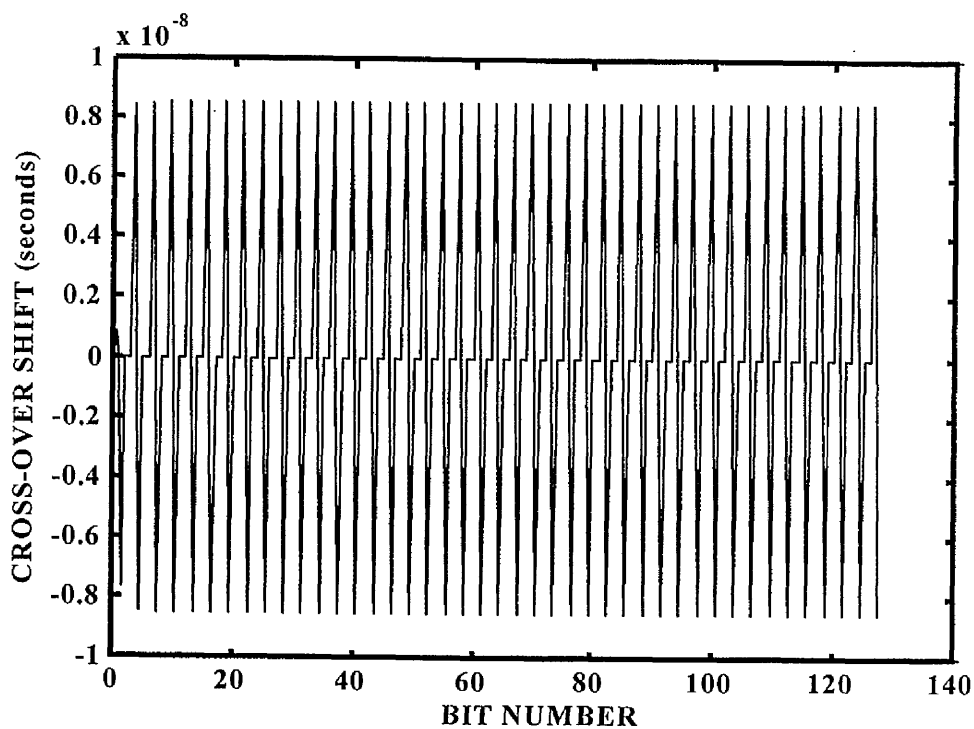


Figure 3.5: Cross-over shift that occurs in the worst case data pattern in the time domain at
100,000 bits per inch

An alternative way of looking at this data is to examine the cross-over shift in the time domain over the valid bit range. Figure 3.5 shows the general case model at a packing density of 100,000 bits per inch.

3.2.2.3 Cross-over Shift Effects on a Pseudo-Random Data Pattern

A pseudo-random data pattern is produced by using a string of 1's to mask the end-effect bits and a pseudo-random data pattern over the valid bit range. This data is then coded using the delay modulation code. Figures 3.6 shows the general case model's channel response to this data pattern, at a packing density of 100,000 bits per inch.

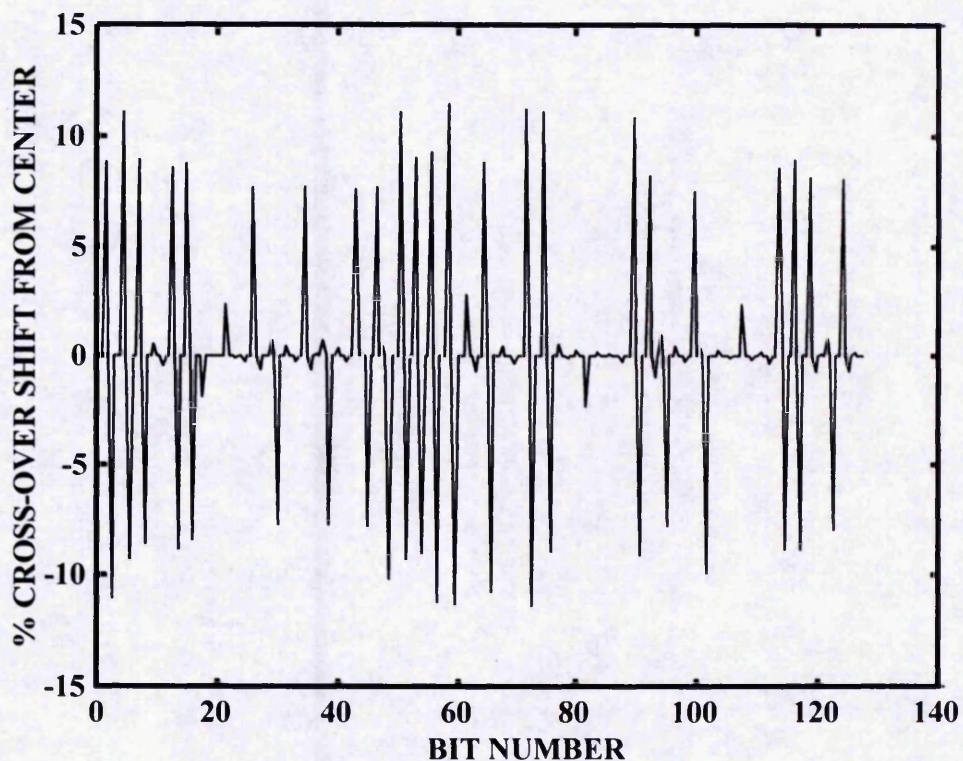


Figure 3.6: The pseudo-random cross-over shift data pattern at 100,000 bits per inch

3.2.2.4 Average cross-over shift over a large packing density range

To examine the effect of cross-over shift over a large packing density range, we can only obtain valid data while the cross-over shift of any bit is less than 45% from the center of the bit. So, once a bit is found to exceed 45% cross-over shift, no more cross-over shift values are calculated. The value of 45 % was found in practical work to be the upper limit where the channel functioned accurately. Figure 3.7 shows the average cross-over shift of the general case model over a valid packing density range, for the data bits in the worse case data and pseudo random data patterns.

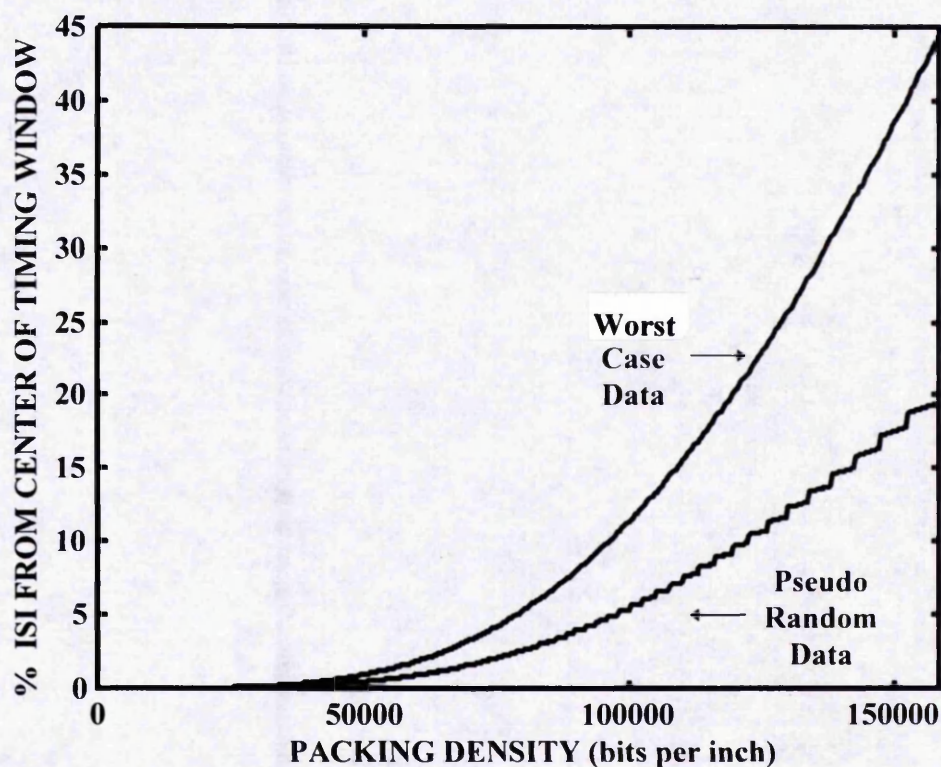


Figure 3.7: The average cross-over shift over a wide range of packing densities

It is clear from figure 3.7 that cross-over shift performance will be a significant limiting factor of the channel.

3.3 A GENERALISED STUDY OF THE PREDICTIONS OF THE PERFORMANCE OF RECORDING CHANNELS BY THE APPLICATION OF SCALING

3.3.1 Introduction

As this simulation is able to be used accurately at packing densities that exceed one million bits per inch, this study can be performed [Tandon, A., Middleton, B.K., Miles, J.J., and Cumpson, S.R., (1997)]. The aim of this study is to demonstrate that it is possible to scale the values of various parameters and variables involved without altering system performance and also produce predictions for the new parameters without repeating the calculations. This is particularly useful when extrapolating from what is understood and practicable to what is hopefully to be achieved using the peak detection strategy. These computations do not address scaling during the recording process [Mallinson, J.C., (1996)] or the problem of changes of equalization and detection strategy.

3.3.2 Calculations

The expressions for output voltages $e(\bar{x})$ from the replay head of a digital recorder can be expressed as ratios of various parameters as indicated in equation (3.1) for the general case isolated pulse shape [Mee, C. D., and Daniel, E. D., (1996); Middleton, B.K., Wright, C.D., Cumpson S. R., and Miles J. J., (1995)]

$$e(\bar{x}) = f \left[\frac{\bar{x}}{a_t + d}, \frac{D}{a_t + d}, \frac{g}{a_t + d} \right] \quad (3.1)$$

In the case where the output waveform is produced by an isolated transition and a very narrow gap, the output voltage is expressed in equation (3.2). A narrow gap head is chosen simply to focus attention on $(a_t + d)$ where intense efforts are being made to reduce their values. Inclusion of the finite g does not alter any principle but makes algebra more cumbersome.

$$e(\bar{x}) = \frac{\mu_o v \omega M_o n \eta}{\pi} \ln \left[\frac{\left[\frac{\bar{x}}{a_t + d} \right]^2 + \left[1 + \frac{D}{a_t + d} \right]^2}{\left[\frac{\bar{x}}{a_t + d} \right]^2 + 1} \right] \quad (3.2)$$

At high densities the output according to linear superposition is given in equation (3.3).

$$e(\bar{x}) = \sum_n a_n f \left[\frac{\bar{x} + nb}{a_t + d}, \frac{D}{a_t + d}, \frac{g}{a_t + d} \right] \quad (3.3)$$

where $a_n = 0, 1, -1$, and b is bit cell length. It can be seen that reducing all the parameters by the same factor leaves the output voltage at the same magnitude but increases packing density b^{-1} as a result of the reduction of b . The output waveforms remain the same shape for any amount of scaling except that the distance and time scales are altered.

The probability of error for any given bit in a recording system will be analysed. It is required for this study, and so relevant equations will be used, but a detailed examination will be provided later in chapter 4.4. Equation (3.4) describes this probability of error [Katz, E. R., and Campbell, T. G., (1979)].

$$P_e = 0.5 \left[\operatorname{erfc} \left[\frac{T_w + T_{ps}}{\sqrt{2} T_n} \right] + \operatorname{erfc} \left[\frac{T_w - T_{ps}}{\sqrt{2} T_n} \right] \right] \quad (3.4)$$

where

- $2T_w$ = Timing window of detector and decoder
- T_{ps} = Cross-over shift of the differentiated waveform
- T_n = Cross-over shift due to noise

The above expression is written in terms of time but by multiplying the top and bottom of the fractions by v the expression is written in terms of distance and is shown in equation (3.5) .

$$P_e = 0.5 \left[\operatorname{erfc} \left[\frac{x_w + x_{ps}}{x_n} \right] + \operatorname{erfc} \left[\frac{x_w - x_{ps}}{x_n} \right] \right] \quad (3.5)$$

where

- x_w = vT_w
- x_{ps} = vT_{ps}
- x_n = vT_n

Equations (3.4) and (3.5) are written as equations (3.6) and (3.7) respectively, where all the variable quantities appear as ratios. T_b is the spacing between two bits defined in the time domain. Again provided $(T_w/T_b) = (x_w/b)$ is kept constant the probability of error will remain the same upon scaling the output waveform.

$$P_e = 0.5 \left[\operatorname{erfc} \left[\frac{1 + \frac{T_{ps} T_b}{T_b T_w}}{\frac{T_n T_b}{T_b T_w}} \right] + \operatorname{erfc} \left[\frac{1 - \frac{T_{ps} T_b}{T_b T_w}}{\frac{T_n T_b}{T_b T_w}} \right] \right] \quad (3.6)$$

$$P_e = 0.5 \left[\operatorname{erfc} \left[\frac{1 + \frac{x_{ps} b}{b x_w}}{\frac{x_n b}{b x_w}} \right] + \operatorname{erfc} \left[\frac{1 - \frac{x_{ps} b}{b x_w}}{\frac{x_n b}{b x_w}} \right] \right] \quad (3.7)$$

Equation (3.8) contains a standard definition for cross-over shift due to noise [Katz, E. R., and Campbell, T. G., (1979)].

$$T_n = \frac{n'}{s''} \quad (3.8)$$

Where n' is the differential of noise at the replay head and s'' is the corresponding double differential of the signal. Using $s'' = (\pi/T_b)s'$, in a pulse crowded situation, where s' is the differential of the output signal equation (3.8) can be expressed in the reduced form shown in equation (3.9).

$$\frac{T_n}{T_b} = \frac{x_n}{b} = \frac{n'}{\pi s'} \quad (3.9)$$

where (s'/n') now represents the signal to noise ratio out of the differentiator. This is proportional to the signal to noise ratio of the head (s/n) [Hughes, G. F., and Schmidt, R. H., (1976)] and if the bandwidth of the differentiator is $\sqrt{3}(\pi/T_b)$, which is reasonable, $s/n = s'/n'$. Equation (3.10) shows the new expression for (T_n/T_b) with s being determined from figure 3.8.

$$\frac{T_n}{T_b} = \frac{x_n}{b} = \frac{n}{\pi s} \quad (3.10)$$

All the equations have now been defined in terms involving ratios that can be affected by scaling.

Curve	a_t (μm)	d (μm)	$a_t + d$ (μm)	D (μm)
1	0.1	0.076	0.176	0.025
2	0.008	0.076	0.084	0.025
3	0.006	0.040	0.046	0.025
4	0.003	0.025	0.028	0.025
5	0	0.025	0.025	0.025
6	0	0.025	0.025	0.0025

Table 2: Values of a_t and d for study about scaling

Six different media are modelled which possess a wide range of values of a_t and d varying from values currently realised in practice through to values which arguably represent what might be limiting values. These six values of a_t and d are shown in table 2. The roll-off curves are shown in figure 3.8 and result from the superposition of a data stream that contains a string of 1's coded using the delay modulation code. These roll-off curves match very closely to those produced using closed form mathematical equations [Mee, C. D., and Daniel, E. D., (1996); Middleton, B.K., Wright, C.D., Cumpson S. R., and Miles J. J., (1995)] and so demonstrates that this simulation technique functions accurately at packing densities that exceed 1 million bits per inch.

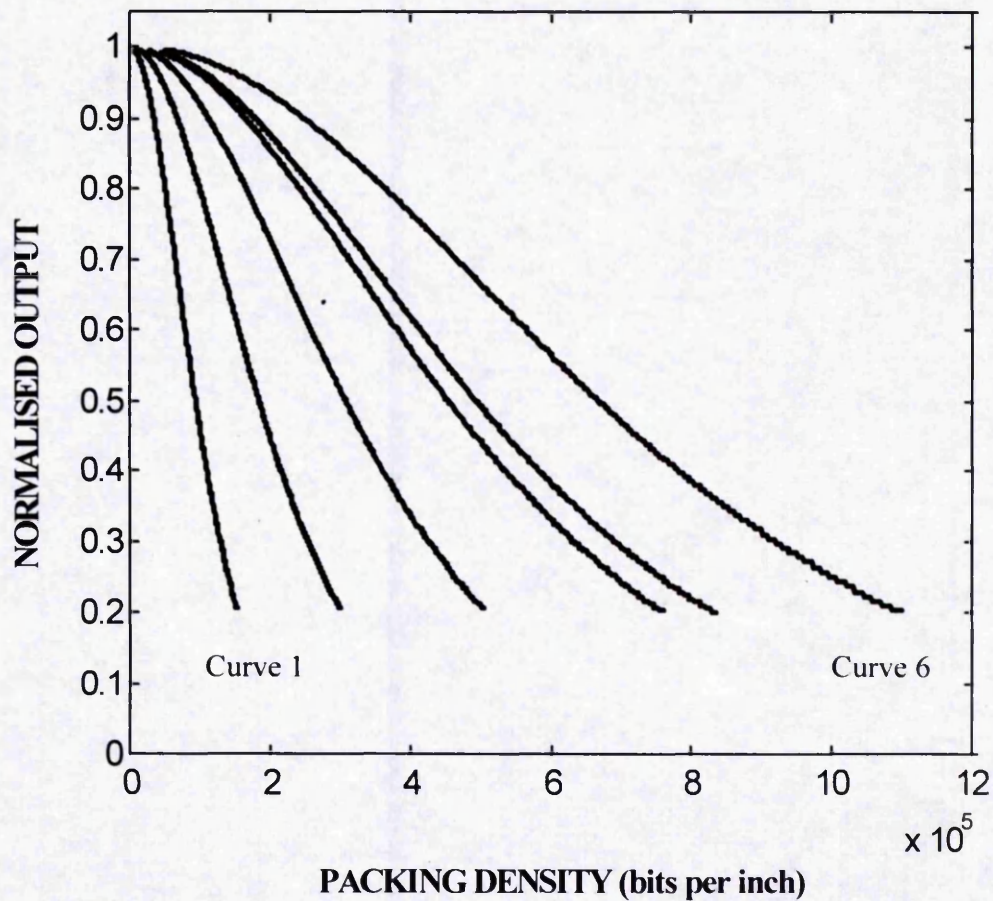


Figure 3.8: Normalised output amplitude as a function of packing density

The larger values of d correspond to flying heights of heads which are currently used in rigid disk machines while the value of 0.025 microns is the value which may represent the lowest possible value in view of surface roughness of both heads and media. The transition width of 0 represents the ultimate possibility and the roll-off curve reaching out to highest densities appears to be a limiting achievement. Figure 3.9 shows the first five data sets in table 2, and illustrate that the this limited number of curves span the range of variables likely to be used in the future.

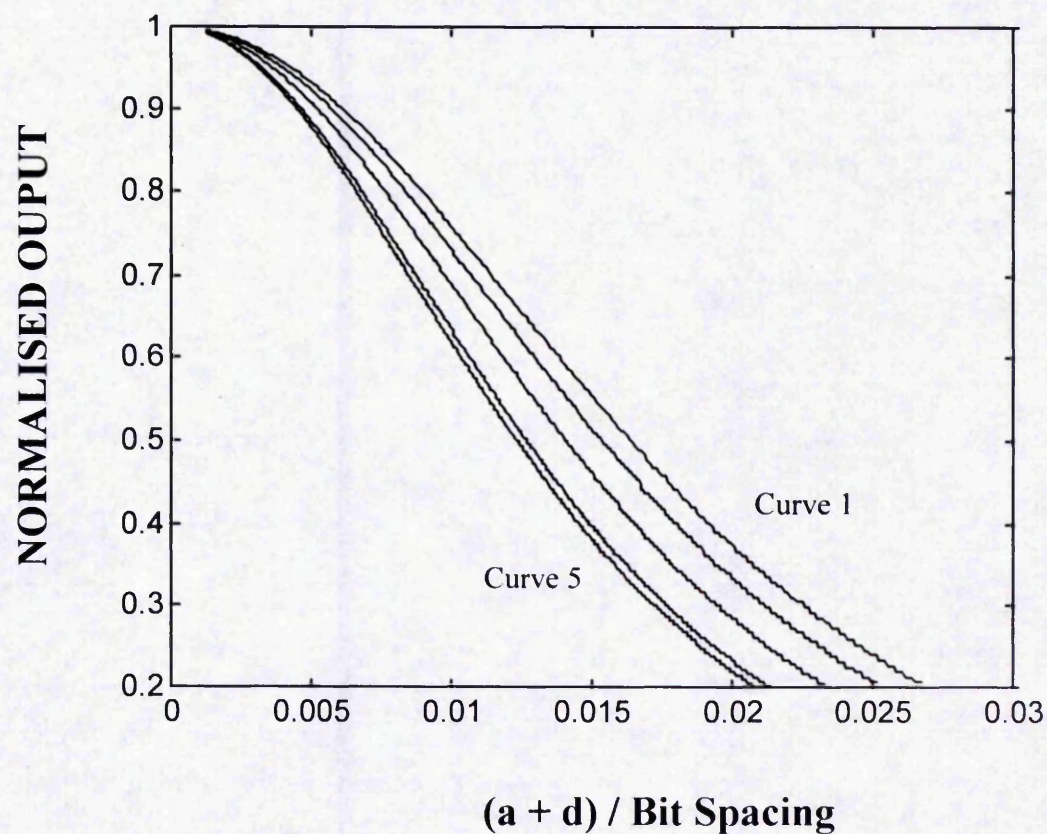


Figure 3.9: Normalised output amplitude as a function of $(a_i + d)$ / bit spacing

3.3.3 Predictions

Detailed analysis of three main characteristics is performed. These characteristics are the average cross-over shift, the packing density and the bit error rate.

3.3.3.1 Analysis of average cross-over shift

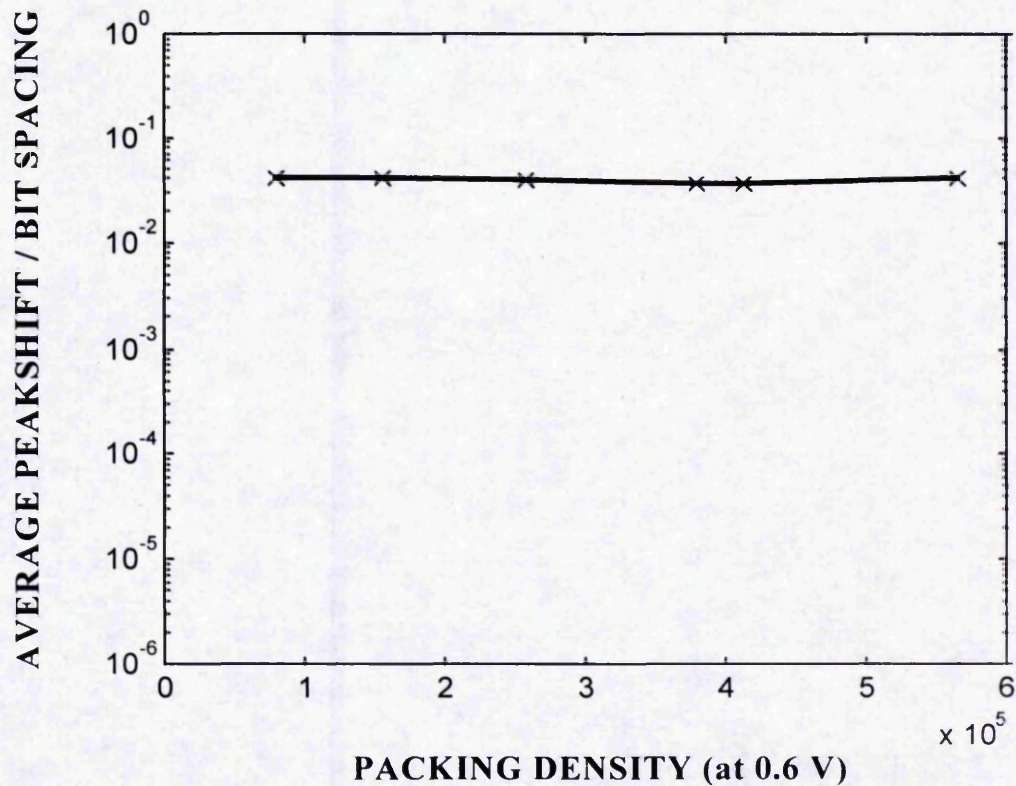


Figure 3.10: Average cross-over shift / bit spacing as a function of packing density

In figures 3.10 and 3.11 it is clear to observe that the average cross-over shift as a fraction of bit spacing is a constant and is independent of $(a_t + d)$, D and the packing densities at a normalised output voltage of 0.6. This voltage value is used as it represents a typical point on a roll-off curve where, in practice, digital storage systems operate.

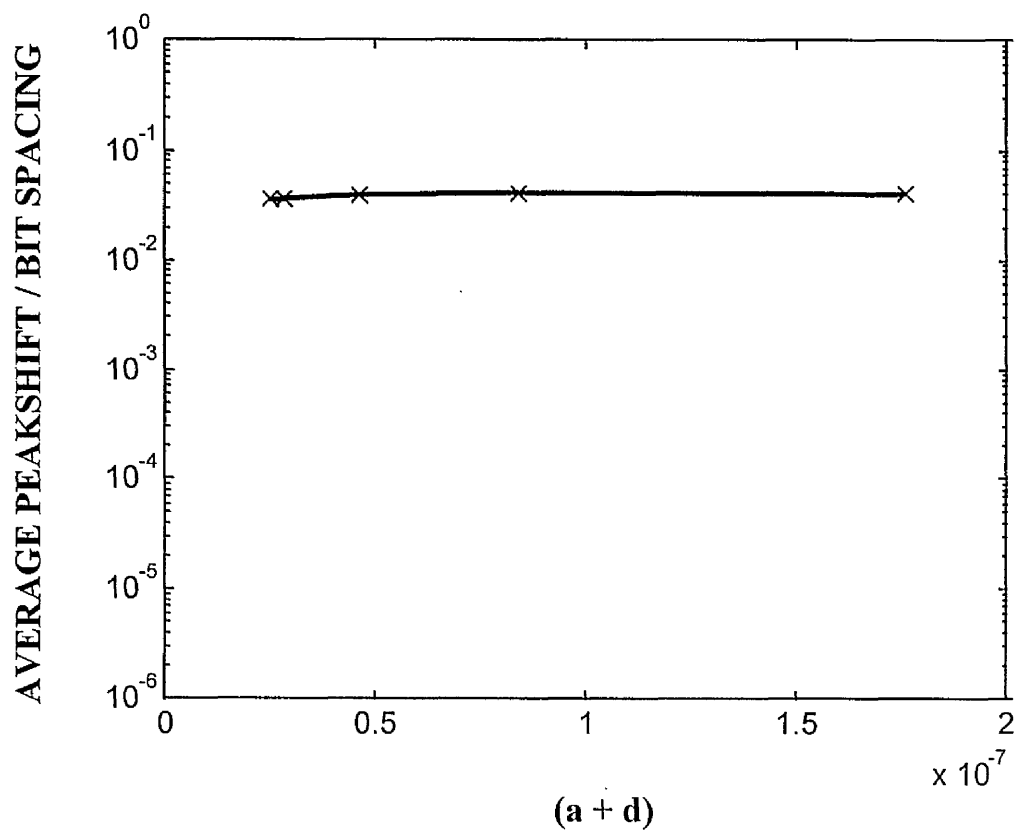


Figure 3.11: Average cross-over shift / bit spacing as a function of $(a_1 + d)$

Figure 3.12 shows the average cross-over shift plotted against $D / (a_1 + d)$ at packing densities which are given by the normalised output voltage of 0.6 according to figure 3.8. The curves in figures 3.10-3.12, occur as expected, and confirms that the theory of scaling of parameters is valid, as described in equation (3.3).

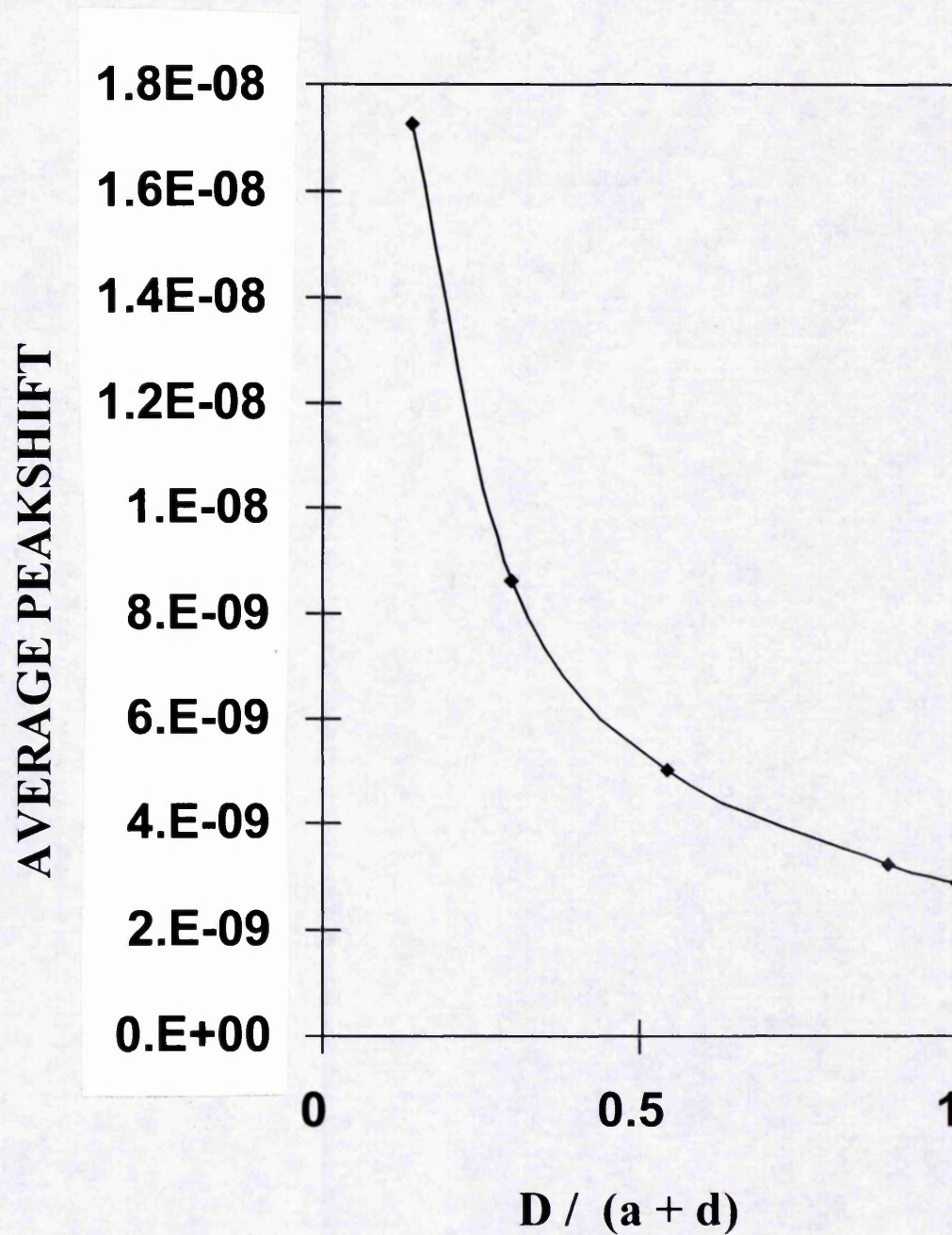


Figure 3.12: Average cross-over shift as a function of $D / (a_1 + d)$

3.3.3.2 Analysis of bit error rate

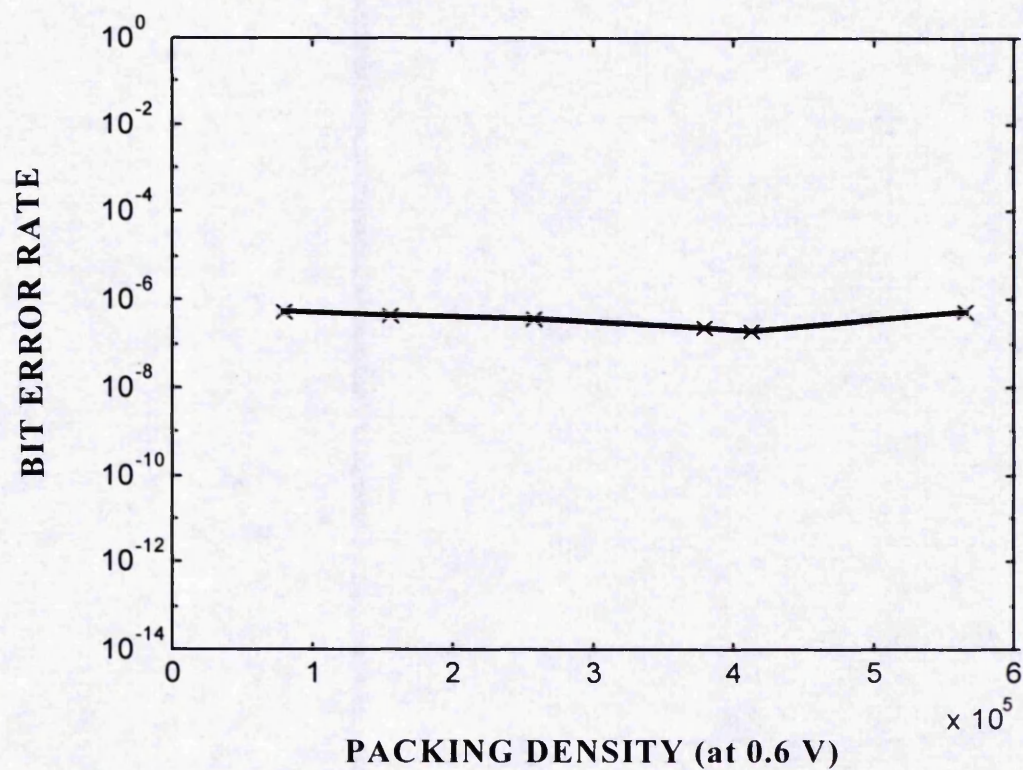


Figure 3.13: Bit error rate as a function of packing density

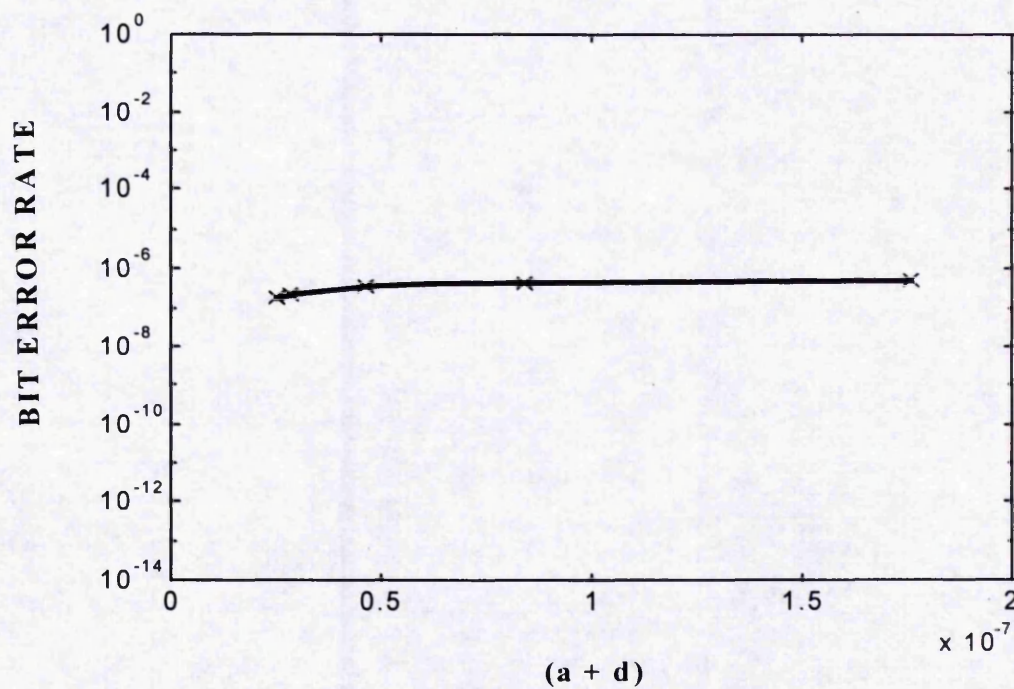


Figure 3.14: Bit error rate as a function of $(a_t + d)$

In figures 3.13 and 3.14 it is clear to observe that the bit error rate is a constant and is independent of $(a_t + d)$, D and the packing densities at a normalised output voltage of 0.6. Figure 3.15 shows the bit error rate plotted against $D / (a_t + d)$ at packing densities which are given by the normalised output voltage of 0.6 according to figure 3.8. Again figures 3.13-3.15, occur as expected, and confirms that the theory of scaling of parameters is valid as described in equations (3.3) and (3.7).

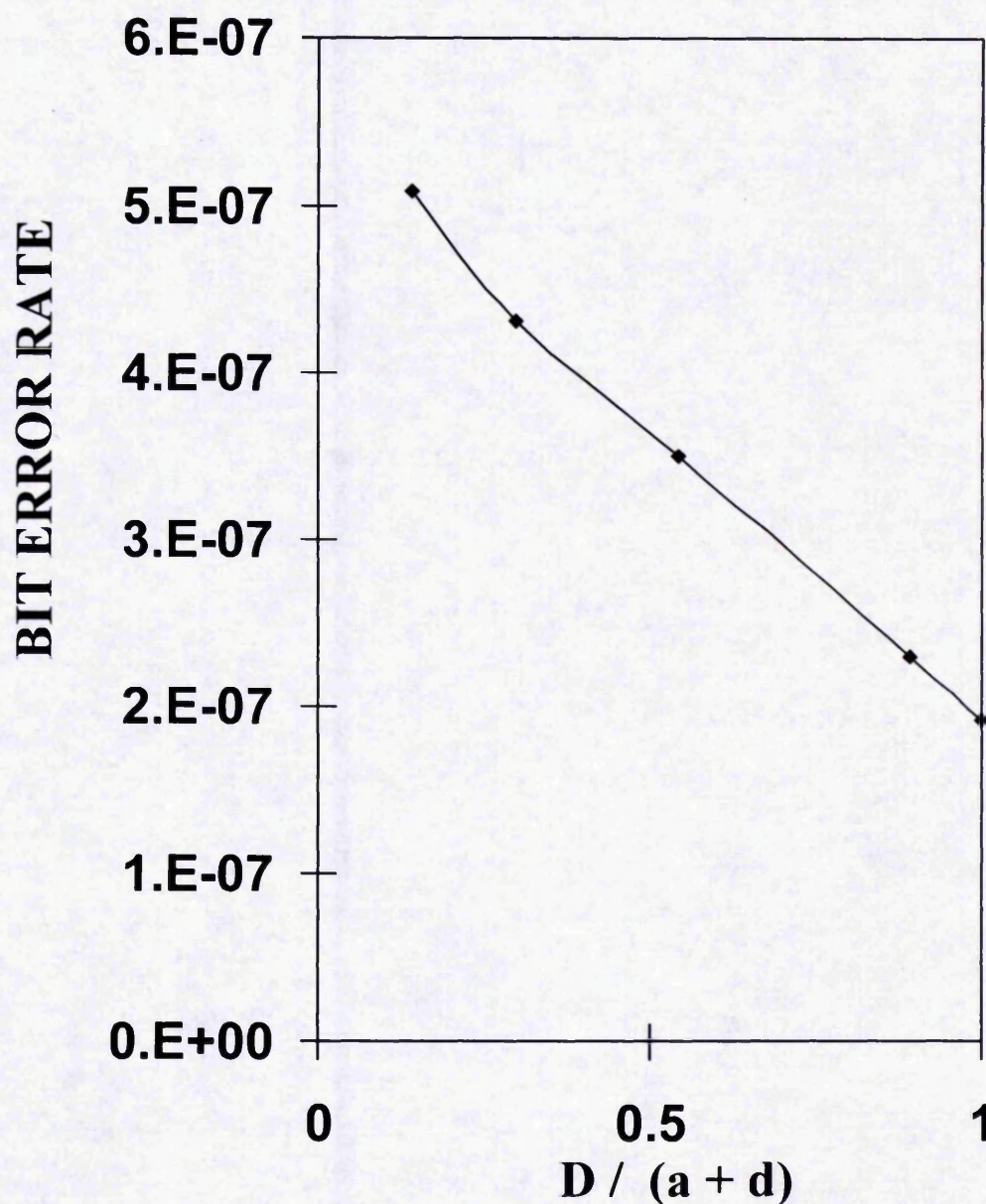


Figure 3.15: Bit error rate as a function of $D / (a_t + d)$

3.3.3.3 Analysis of packing density

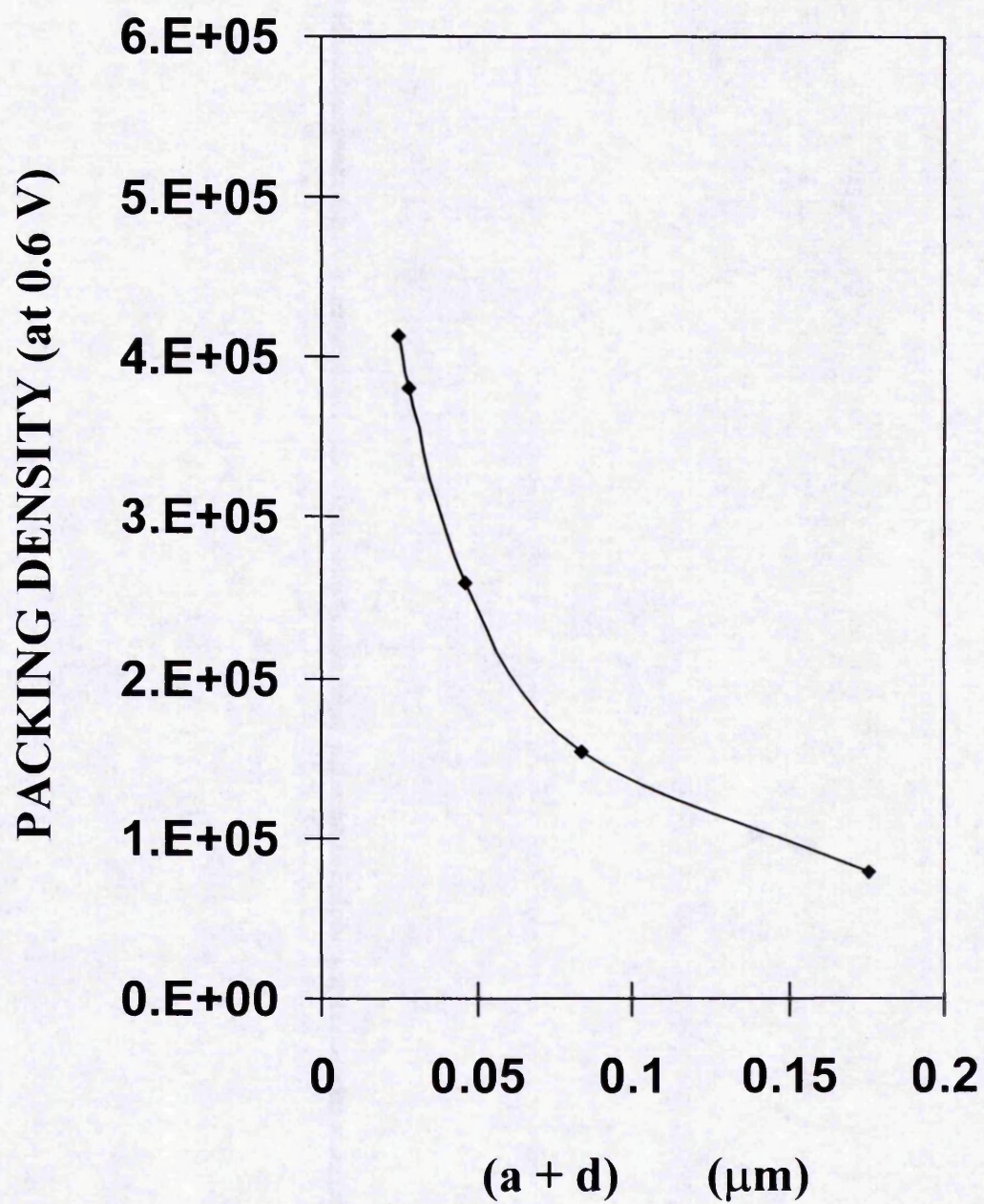


Figure 3.16: Packing density as a function of $(a_t + d)$

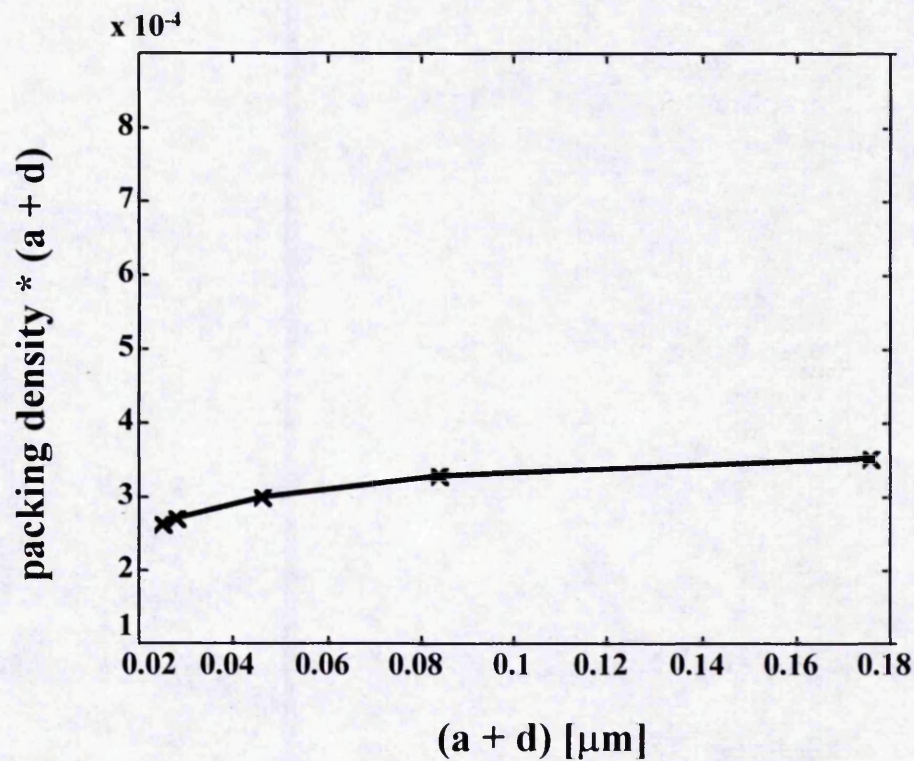


Figure 3.17: Packing density * $(a_1 + d)$ as a function of $(a_1 + d)$

Figure 3.16 shows the packing density plotted against $(a_1 + d)$ and D . Figure 3.17 shows the packing density * $(a_1 + d)$ plotted against $(a_1 + d)$. If the technology is able to reduce $(a_1 + d)$ by a certain amount, figure 3.17 will give the packing density that is possible while maintaining the same error rate.

3.3.4 Conclusions Of This Study

This study has performed a detailed analysis of the average cross-over shift, the packing density and the bit error rate characteristics of the recording channel. The study has clearly demonstrated that it is possible to scale the values of various parameters and variables involved without altering system performance and produce predictions for the new

parameters without repeating the calculations. It has also been demonstrated that the simulation is usable at packing densities that exceed 1 million bits per inch.

3.4 INCORPORATING LARGE DATA TRAINS

Large data trains are required for the new modelling technique, described in section 4.5, and are efficiently modelled using a block structure method. This block structure method allows the simulation to handle any input data size length specified. This is achieved by organising the input data file into blocks, of at most 512 bits, and analysing each bit individually. Once a particular block has been analysed, the next block is processed until the whole input data file has been analysed. Extra buffers, however, are also required to enable interactions between blocks to be accurately modelled and end effects to be considered.

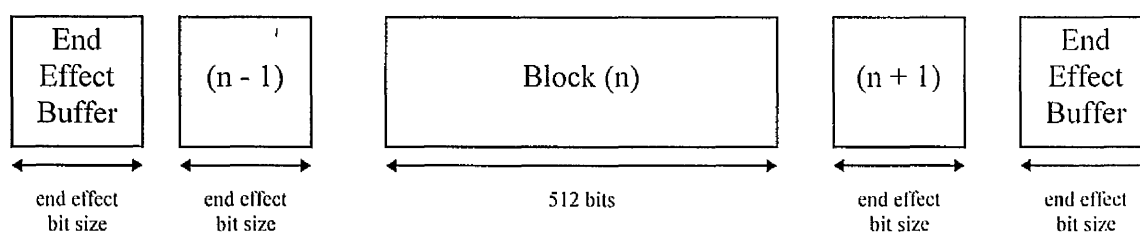


Figure 3.18: An illustration of the block structure method

Figure 3.18 illustrates this concept. For a block (n) to be analysed accurately, information from the previous block (n - 1) and the next block (n + 1) are required to model the effects at the edges of block (n). The delay modulation buffers are used to hold the end effect information. The end effect buffer is defined as a string of 1's coded using the delay modulation code, as this produces no cross-over shift effects on the valid data pattern itself.

Once block (n) has been analysed then block (n + 1) is processed and so on until all the data has been processed and analysed.

3.5 INCORPORATING MULTI-TRACKS INTO THE SIMULATION

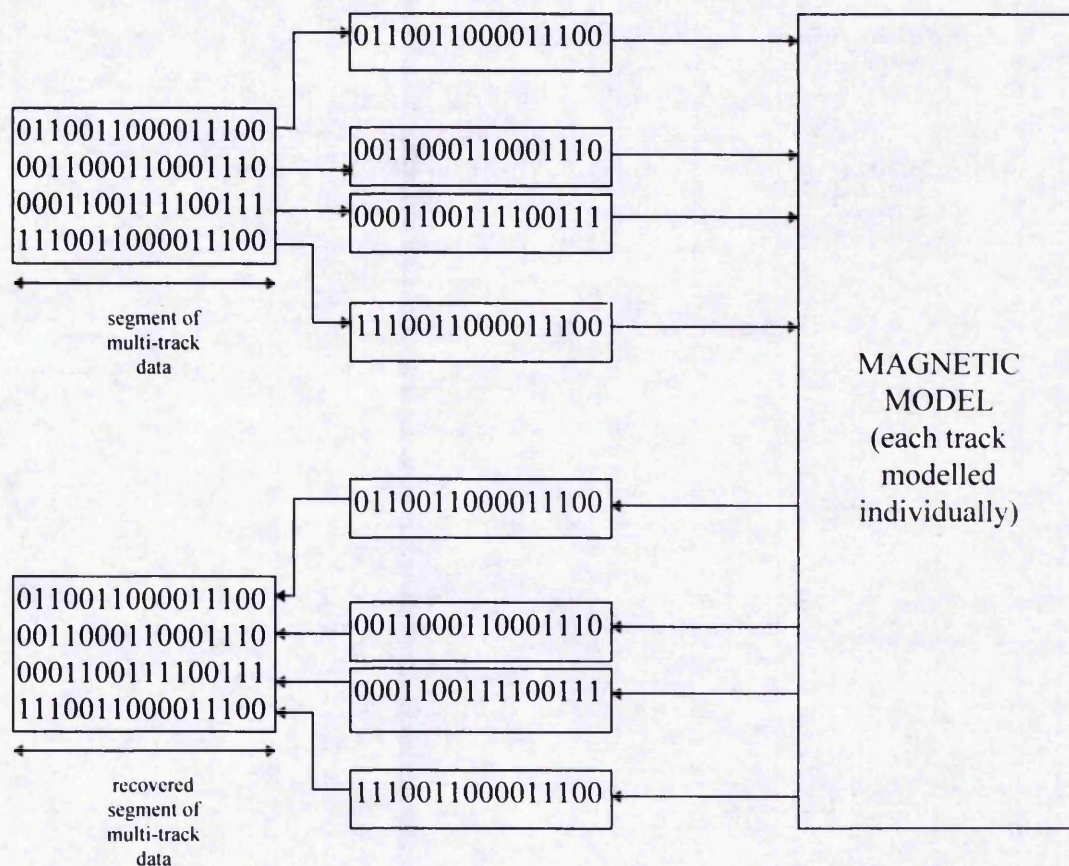


Figure 3.19: Modelling a multi-track recording system

Tape systems often contain more than one track. By assuming that cross-talk between tracks is negligible, an efficient method for multi-track simulation has been produced that improves the flexibility of the simulation. This method is to separate each track into a file and model each file individually as shown in figure 3.19. Thus each track has been

modelled accurately and by reconstructing each track, the effects of recovered multi-tracks can be analysed. This method allows the user of the simulation to specify the number of tracks that are required.

3.6 SUMMARY

A full implementation of cross-over shift analysis has been examined in different cross-over situations, and the limits of the channels performance have been illustrated using different data patterns to perform the analysis. When new pulse shapes are defined we can determine the usefulness of the pulse under examination, by investigating its cross-over shift performance over a wide range of packing densities.

A study was conducted which demonstrated the power of the modelling tool for performance analysis of the recording channel as individual parameters of interest can be investigated in detail, and that the simulation can function accurately at the theoretical upper limits of what can be achieved using peak detection channels. It was shown in this study that values of parameters can be scaled to produce predictions of new parameters. This is very useful for system design as characteristics of new tapes and disks can be easily investigated as system performance is unaltered.

Also the channel has been extended to handle both multi-tracks and large data trains. Both of these extensions are required when it comes to designing error performance schemes using the new modelling technique described in chapter 5.

4 INCORPORATING DEFECTS INTO THE RECORDING CHANNEL

4.1 INTRODUCTION

To realistically simulate a magnetic channel, defects must be considered. In this simulation, two types of defect are to be modelled. These are drop-outs and additive white gaussian noise (AWGN).

Drop-outs are reductions in the signal amplitude which occur as a result of local variations in tape properties, the record process or the replay process. In digital magnetic systems, drop-outs may lead to errors in the stored data. By modelling drop-outs, error schemes can be devised that mask the presence of drop-outs in both single track and multi-track systems.

AWGN is used to simulate random noise in the magnetic channel. This random noise includes media noise, broad-band noise, components from overwrite or adjacent track pickup, and from other noise sources. Two techniques will be considered to produce bit error rates. Firstly, a new technique will be described [Tandon, A., Middleton, B.K., Farrell, P.G., and Miles, J.J. (1997)], where simulated noise is generated and added to large data streams. So, for example, if the bit error rate is 1 in a hundred thousand, then a data stream of 10 million bits will be used to produce 100 errors. Following this the classical analytical technique using the error function complement to calculate bit error rates will be

produced [Katz, E. R., and Campbell, T. G., (May 1979)], and used to verify the validity of the new method.

In previous simulations [Loze, M. K. et al. (1984)], pulse equalization has been used to increase the performance of the channel. Therefore, pulse equalization will also be examined in detail, but is only implemented for the classical analytical technique. By combining pulse equalization and the classical technique the validity of the new sampled waveform method can be verified. Also since previous simulations [Li Hui Hong, J.K. et al. (1993)] contain both longitudinal and perpendicular components, a model that contains such components will be introduced for the use of comparison purposes only. A study will also be produced which examines the performance limitations due to ISI and noise to determine the effects of peakshift limited performance and noise limited performance in the recording channel.

Finally, drop-out burst error events [Tandon, A., Middleton, B.K., Farrell, P.G., and Miles, J.J. (1997)] will be investigated in detail. In particular internal cross-over shift characteristics and the analysis of burst error statistics will be described. It is these aspects which provide the new sampled AWGN technique with its modelling advantage, and advanced error correction and detection schemes can be devised using this additional information.

4.2 ANALYSIS OF THE EFFECTS OF NOISE BY THE USE OF THE CLASSICAL ANALYTICAL TECHNIQUE

The traditional technique used for simulating noise is to use the error function complement [Katz, E. R., and Campbell, T. G., (1979)] to define the bit error rate of the system.

The bit error rate (E) is defined in equation (4.1).

$$E = \frac{1}{2} \left[\operatorname{erfc} \left(\frac{T_w - \tau_1}{\sqrt{2} \tau_2} \right) + \operatorname{erfc} \left(\frac{T_w + \tau_1}{\sqrt{2} \tau_2} \right) \right] \quad (4.1)$$

where $2T_w$ is the timing window, τ_1 is the cross-over shift induced bitshift, and τ_2 is the noise induced bitshift. The error function complement [Middleton, B. K., and Jack-Kee, T., (1983)] is defined in equation (4.2).

$$\operatorname{erfc}(x) = \frac{1}{\sqrt{\pi}} \int_x^\infty e^{-x^2} dx \quad (4.2)$$

To calculate the bitshift due to noise (τ_2) equation (4.3) is used.

$$\tau_2 = \frac{n'(t_o)}{S''_i} \quad (4.3)$$

In equation (4.3), $n'(t_o)$ is the noise as measured after filtering and differentiation and S''_i is the second derivative of the signal waveform at the i^{th} pulse. The relationship of S''_i and

S'_i (the first derivative of the signal waveform) is shown in equation (4.4), with T being defined as the spacing between two bits.

$$S''_i = \frac{\pi}{T} S'_i \quad (4.4)$$

An approximation exists in equation (4.4) with the assumption that the gradient at every cross-over point is the same in every bit of the data sequence being tested. This approximation is not made in the new modelling technique and therefore allows the new technique to produce a more accurate simulation. Combining equations (4.3) and (4.4) gives the general form for the noise induced bitshift and is described in equation (4.5).

$$\tau_2 = \frac{n'(t_o) T}{S'_i} \quad (4.5)$$

The cross-over induced bitshift (τ_1) is calculated for each bit of a pseudo-random 128 bit sequence, with the error rate also being calculated for each bit. This is then averaged to produce the bit error rate used in the analysis.

4.3 PULSE EQUALIZATION

To confirm that the new sampled waveform technique shows a good agreement with previous simulations and previous practical work, pulse equalization [Loze, M. K. et al. (1984)] is required. Earlier work needed pulse equalization as a method to enhance the channel performance of the tape medium. Pulse equalization has only been applied to the classical analytical technique, and its use in the new sampled AWGN technique is not considered.

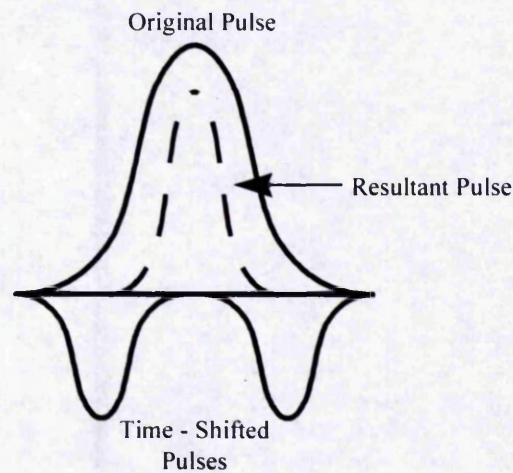


Figure 4.1: The interaction of the original pulse and time-shifted pulses

Pulse equalization improves performance by reducing cross-over shift effects and increasing packing density characteristics. To implement pulse equalization, two time-shifted pulses, of a negative sense to the original pulse and with a lower magnitude are added to the original pulse, as shown in figure 4.1.

PARAMETER	VALUE
$(a_t + d)$	$7.8 * 10^{-7}$ metres
D	$3 * 10^{-8}$ metres
μ_0	$4 \pi * 10^{-7}$ henries / metre
ω	$1.81 * 10^{-3}$ metres
v	0.381 metres / sec
M_x	$8.16 * 10^4$ amps / metre
$\eta * n$	18
α	0 degrees
2g	$2 * 10^{-10}$ metres

Table 3: A previously used longitudinal pulse.

Note: The small value of 2g is used to represent the narrow gap approximation.

To illustrate pulse slimming, an example longitudinal pulse defined in table 3, [Li Hui Hong, J.K. et al. (1993)] is used. The effect of adding the time-shifted pulses is to slim the pulse by reducing the width of the pulse. As the time-shifted pulses move away from the center of the original pulse, the width of the equalized pulse decreases to an optimum value. An example of an equalized pulse is shown in figure 4.2, along with the original pulse. The optimum equalized pulse, is produced by defining the time-shifted pulses as being 30% of the size of the original pulse and positioned 2% away from the center of the pulse (which is equivalent to being $2.456693 * 10^{-6}$ seconds away).

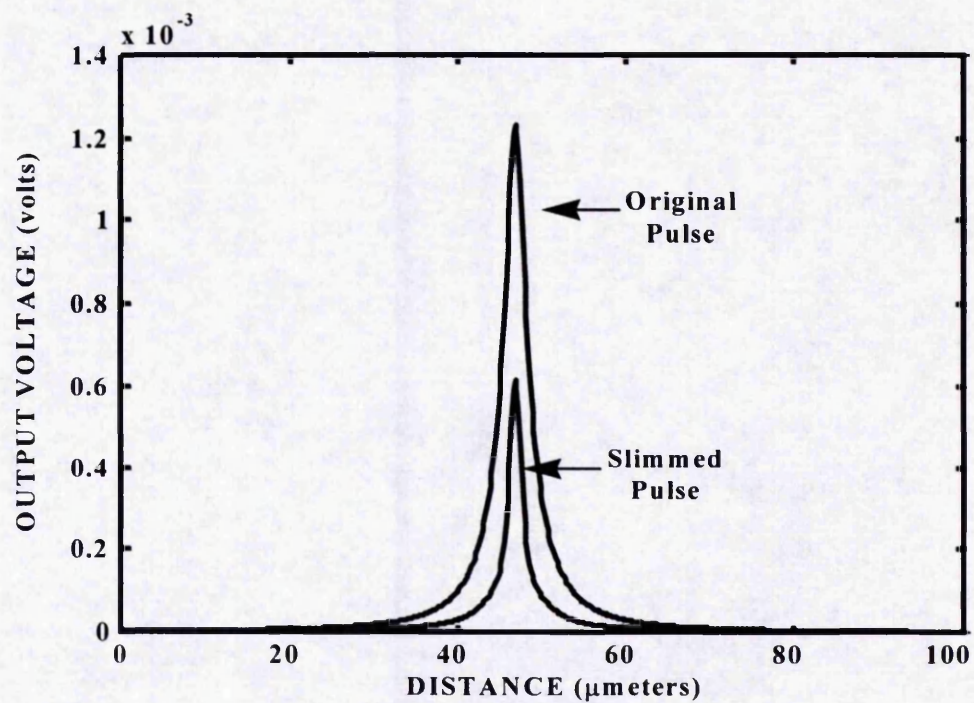


Figure 4.2: An example equalized and original pulse

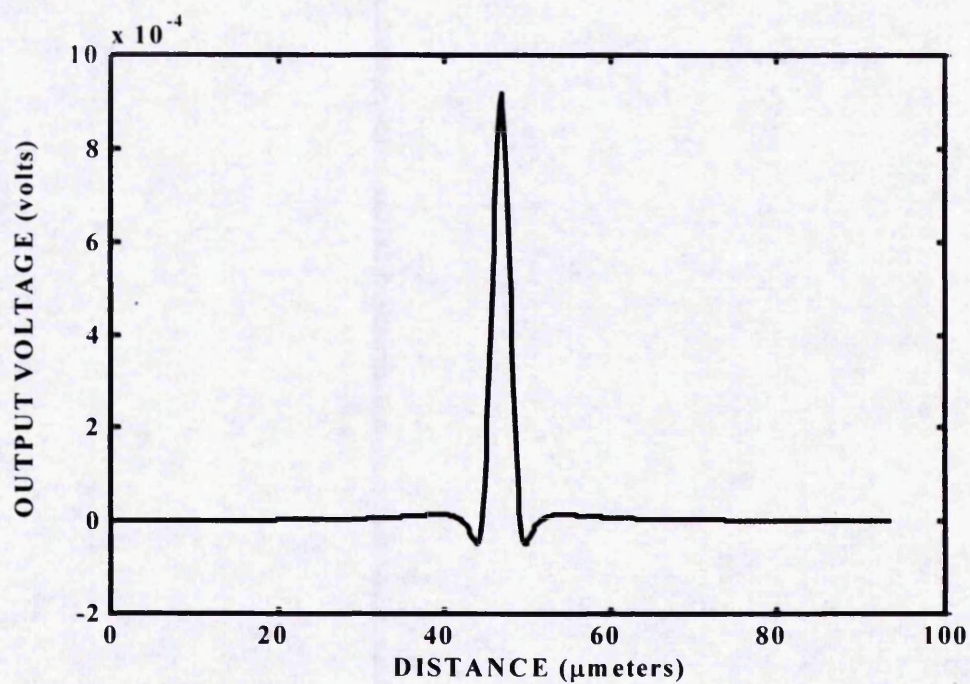


Figure 4.3: An example of a distorted pulse

If the time-shifted pulses are moved further apart than the optimum position then the time-shifted pulses become visible and the pulse is considered to be distorted. Figure 4.3 shows an example of a distorted pulse.

Figures 4.4 and 4.5 show the cross-over shift characteristics and the maximum output voltage analysis, respectively, of the original and equalised pulses.

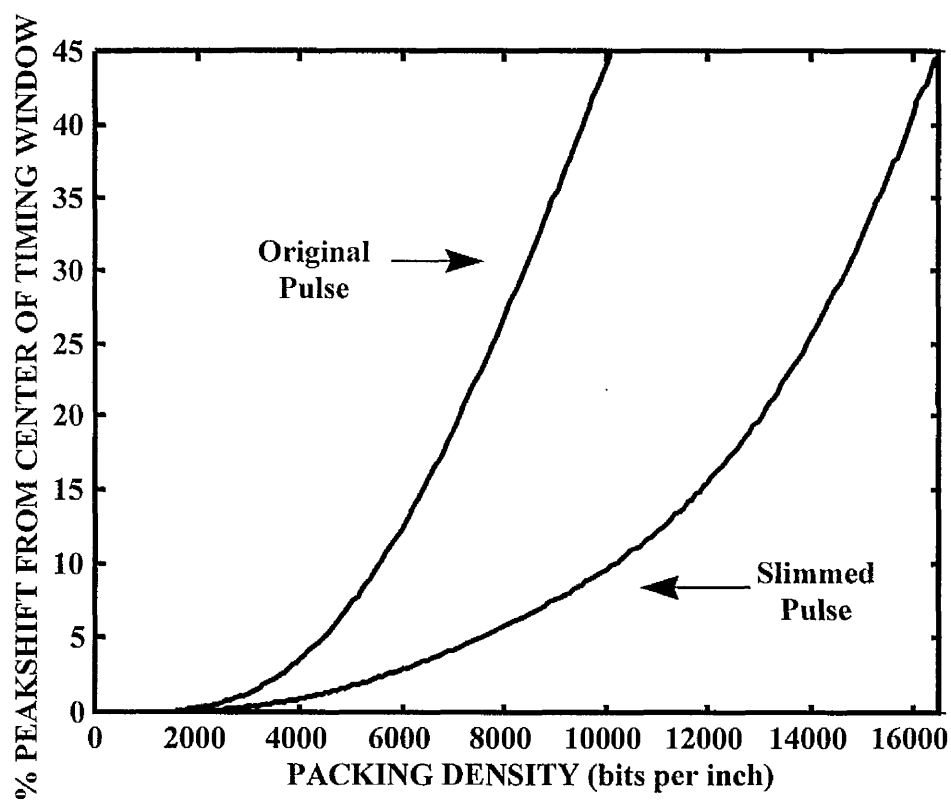


Figure 4.4: Cross-over shift analysis of the two pulses

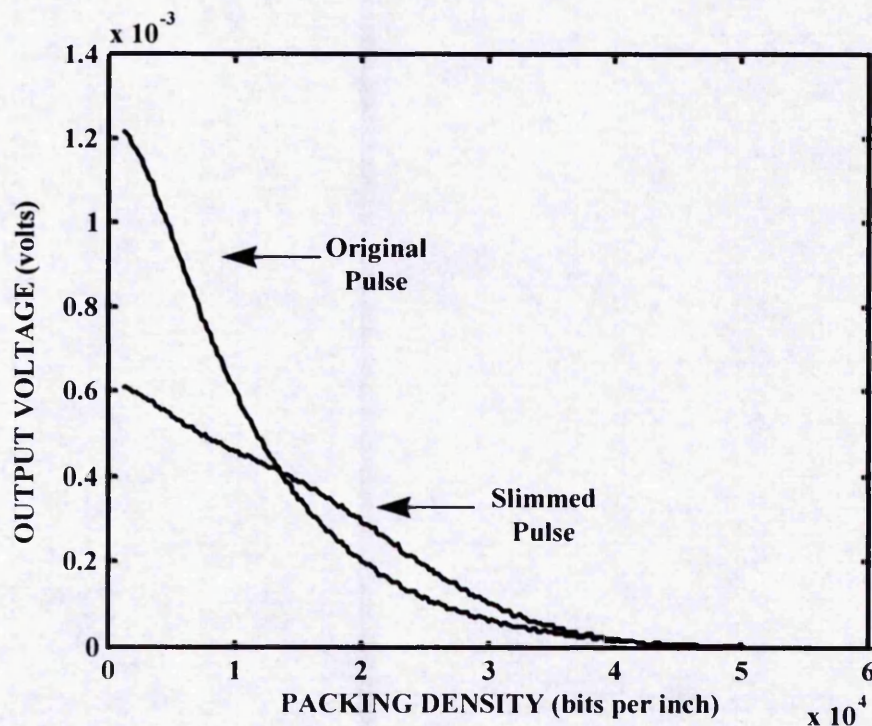


Figure 4.5: The maximum output voltage analysis of the two pulses

4.4 DROP-OUT INFECTED CHANNELS

Drop-outs occur, in general, by the incursion of dust between the recording medium and the replay head and this causes reductions in the reproduced signal amplitudes of the recorded waveform. The characteristics of the simulated drop-outs are examined in detail in this section and statistical analysis is performed to calculate the number, length, width, and position of drop-outs. Drop-outs that cause the loss of clock synchronisation have not been considered.

4.4.1 Modelling The Drop-Out Characteristics

The major effect of a drop-out is to increase the size of the parameter d (which is the (head / medium) spacing on replay).

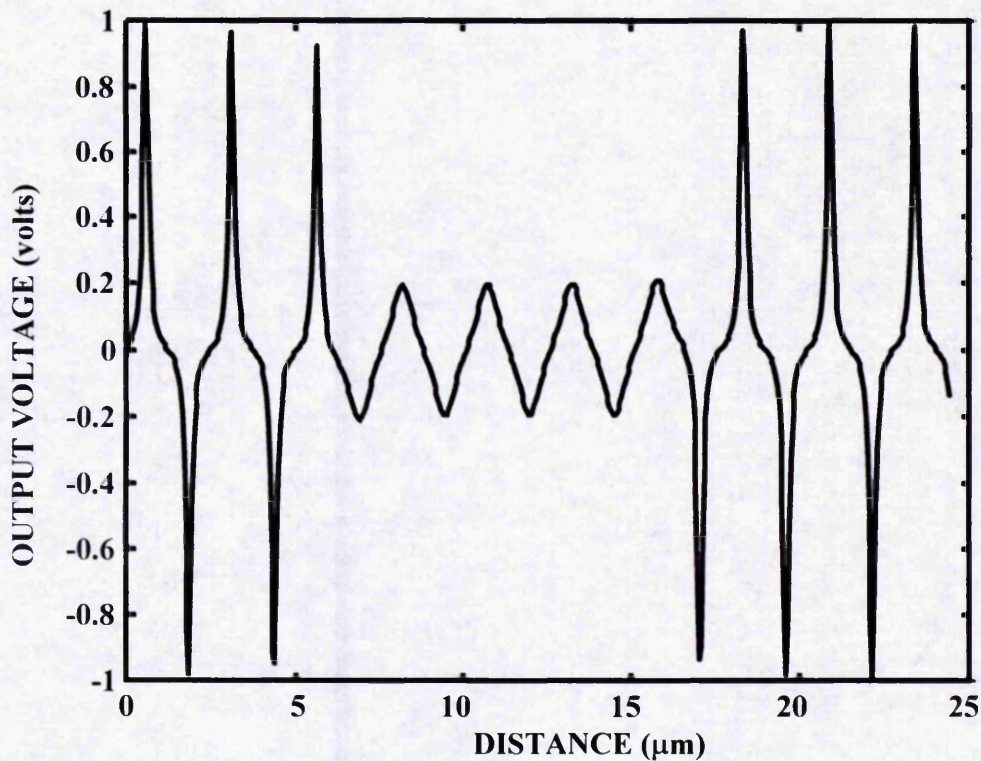


Figure 4.6: A drop-out infected waveform generated at 20,000 bits per inch

Also an increase in the pulse width of the drop-out waveform occurs when compared with the pulse width of the original waveform. The larger pulse width causes zero-crossing points of the differentiated waveform to be pushed outside of timing windows.

PACKING DENSITY (bpi)	DROP-OUT VALUE (m)
150000	$1.69 * 10^{-7}$
155000	$1.67 * 10^{-7}$
160000	$1.64 * 10^{-7}$
165000	$1.62 * 10^{-7}$
170000	$1.60 * 10^{-7}$
175000	$1.58 * 10^{-7}$
180000	$1.56 * 10^{-7}$
185000	$1.54 * 10^{-7}$

Table 4: Drop-out values required for simulating a set of packing densities

Figure 4.6 shows a simulated drop-out which suffers an 80% loss of signal amplitude compared with the original waveform signal output. The drop-out value, which is an extra head to medium spacing during a drop-out, at 20,000 bits per inch is $3.96 * 10^{-7}$ m, and this drop-out value replaces $(a_t + d)$ over the drop-out region. Table 4 shows drop-out values required for the analysis of a sample set of packing densities, with the packing densities defined in bits per inch and the drop-out values in metres.

4.4.2 Statistical Analysis Of Drop-Outs

Statistical analysis is required to calculate the number, length, width, and position of drop-outs. This calculation is required only for the new modelling technique, which involves applying AWGN samples to the differentiated waveform. The position of the drop-out is calculated using standard pseudo-random number generation techniques, and can occur anywhere on the tape. For a single track channel this size is 10 million bits, but for an eight track channel, the size of every track is $10 \text{ million} / 8$ which is 1,250,000 bits. Each track in a multi-track channel is assumed to contain the same amount of data. A greater detail of analysis is required for the length, width and number of drop-outs.

The statistics used to calculate the length of drop-out are taken from the previously published experimental results [Baker, B. R., (1977)]. From experimental data curves equation (4.6) is derived.

$$N_d = 52480 e^{\left(\frac{-d_d}{d_o}\right)} \quad (4.6)$$

where N_d is the number of drop-out events per unit diameter, d_d is the diameter of a drop-out and d_o is a characteristic length associated with a drop-out ($= 0.143 * 10^{-3}$ m). Using the standard Monte Carlo method [Press, W. H., et al, (1994)], random drop-outs with the correct length distribution are produced.

To determine how many tracks, in the absence of statistics, a drop-out covers a simple rule is used. In a multi-track scenario, the probability of the drop-out covering two tracks is taken to be half the probability of the drop-out covering a single track. The probability of the drop-out covering three tracks is taken to be half the probability of the drop-out covering two tracks, and so on.

The statistics used to calculate the total number of drop-out events are taken from previously published experimental results [Baker, B. R., (1977)]. These show an exponential relationship between drop-out frequency and drop-out length. This leads to the total number of drop-out events (N) being defined in equation (4.7).

$$N = \int_{d_{min}}^{d_{max}} \left(\frac{52480}{0.1 * 10^{-3}} e^{\left(\frac{-d_d}{d_o}\right)} \right) dd_d \times \text{Area of tape} \quad (4.7)$$

with d_{\max} and d_{\min} being the maximum and minimum lengths of the drop-out respectively. We have found that these formulae fit experimental results when considering small and medium sized drop-out sizes, provided d_{\max} is carefully chosen. We assign d_{\min} with the size of a single bit ($1.64 * 10^{-7}$ m), and d_{\max} with the size of 152 bits ($0.025 * 10^{-3}$ m). By putting these values into equation (4.7), equation (4.8) is produced.

$$\frac{N}{\text{Area of Tape}} = 11951.27 \quad (4.8)$$

For this analysis, using an advanced tape, we use a linear density of 154,520 bits per inch, and the area of the tape used is 0.54 m^2 and stores 10 million bits of data. Applying this in equation (4.8) gives the number of drop-out events to be 6489 but this is for 1.4 Gbits of data. For 10 million bits of data, the pro rata number of drop-out events are scaled down by 140, and therefore the total of number of drop-out events used is 47.

Three large drop-outs are also added to simulate exceptional drop-outs that occur. The first drop-out occurs on a single track but has a size of 3000 bits, the second drop-out occurs over half the total number of tracks with a size of 1750 bits on each track, and the last drop-out covers all the tracks with a size of 1000 bits on each track. Figure 4.7 shows the drop-out statistics produced for use in an eight track tape, with the unit for length being bits.

0: length = 70 width = 1 track_start = 3, position = 241640
1: length = 26 width = 2 track_start = 3, position = 808377
2: length = 54 width = 4 track_start = 2, position = 394481
3: length = 38 width = 1 track_start = 0, position = 214503
4: length = 98 width = 1 track_start = 3, position = 301255
5: length = 103 width = 1 track_start = 0, position = 860087
6: length = 48 width = 3 track_start = 1, position = 1095176
7: length = 46 width = 3 track_start = 4, position = 1002736
8: length = 41 width = 2 track_start = 3, position = 739678
9: length = 16 width = 4 track_start = 0, position = 463158
10: length = 38 width = 2 track_start = 4, position = 923544
11: length = 9 width = 1 track_start = 2, position = 188809
12: length = 139 width = 2 track_start = 0, position = 311256
13: length = 65 width = 3 track_start = 5, position = 1097080
14: length = 11 width = 2 track_start = 4, position = 371208
15: length = 13 width = 5 track_start = 2, position = 246763
16: length = 79 width = 3 track_start = 4, position = 1015471
17: length = 8 width = 1 track_start = 1, position = 914947
18: length = 82 width = 6 track_start = 0, position = 586728
19: length = 63 width = 2 track_start = 0, position = 1087528
20: length = 116 width = 4 track_start = 1, position = 1023631
21: length = 115 width = 2 track_start = 0, position = 507755
22: length = 98 width = 1 track_start = 0, position = 211025
23: length = 64 width = 2 track_start = 6, position = 887851

24: length = 7 width = 1 track_start = 3, position = 353210
25: length = 6 width = 3 track_start = 2, position = 456236
26: length = 13 width = 6 track_start = 1, position = 508797
27: length = 38 width = 2 track_start = 3, position = 712994
28: length = 21 width = 3 track_start = 1, position = 339420
29: length = 25 width = 1 track_start = 4, position = 993126
30: length = 47 width = 2 track_start = 0, position = 703292
31: length = 91 width = 1 track_start = 2, position = 1248292
32: length = 10 width = 1 track_start = 5, position = 581752
33: length = 95 width = 1 track_start = 1, position = 436092
34: length = 132 width = 3 track_start = 0, position = 814453
35: length = 6 width = 2 track_start = 2, position = 187145
36: length = 40 width = 1 track_start = 2, position = 227437
37: length = 11 width = 1 track_start = 4, position = 503213
38: length = 90 width = 1 track_start = 6, position = 99007
39: length = 100 width = 6 track_start = 0, position = 850120
40: length = 74 width = 4 track_start = 2, position = 528445
41: length = 73 width = 2 track_start = 5, position = 703081
42: length = 39 width = 6 track_start = 0, position = 766829
43: length = 1 width = 1 track_start = 2, position = 661871
44: length = 25 width = 4 track_start = 0, position = 1216790
45: length = 50 width = 1 track_start = 0, position = 172776
46: length = 25 width = 1 track_start = 1, position = 950052
47: length = 3000 width = 1 track_start = 3, position = 228564

48: length = 1750 width = 4 track_start = 0, position = 576939

49: length = 1000 width = 8 track_start = 0, position = 1087757

Figure 4.7: The drop-out statistics produced for use in an eight track tape

In figure 4.7, length corresponds to the length of the drop-out, width corresponds to the number of tracks that the drop-out covers, track_start corresponds to the track at which the drop-out starts (a value from 0 to 7) and position corresponds to the position on an individual track (or set of tracks) where the drop-out begins.

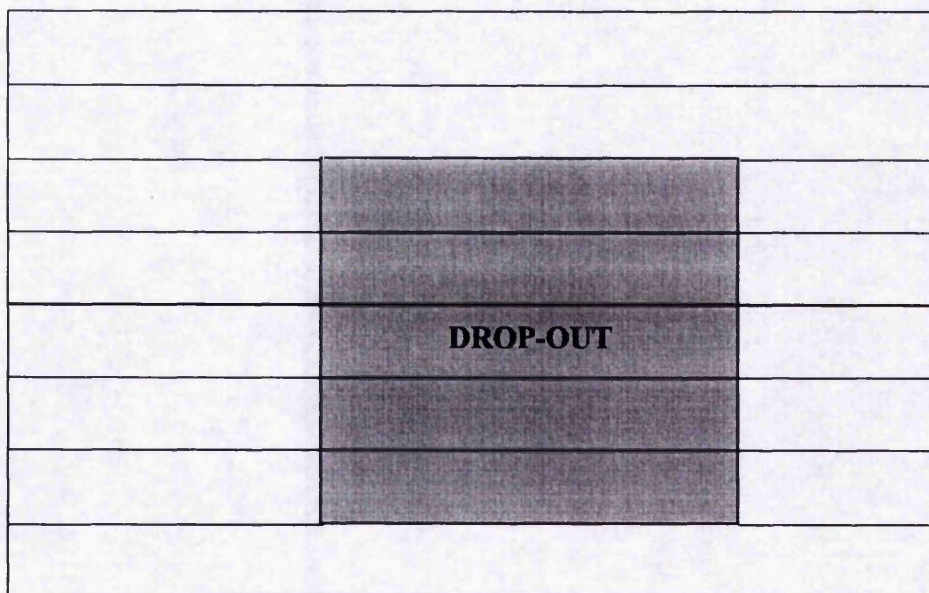


Figure 4.8: A drop-out covering five tracks of an eight track tape channel

An example of a drop-out covering more than one track of an eight track system is shown in figure 4.8. It is important to realise that for this work the assumption that the shape of

the drop-out is rectangular has been made. Further research into the shape of drop-outs is required before a simulation can take this factor into consideration.

4.5 ADDITION OF GAUSSIAN WHITE NOISE SAMPLES TO THE RECORDING CHANNEL

The fundamental idea behind the novel new modelling technique explored in this thesis is straightforward. By applying AWGN samples to the sampled output of the differentiator (figure 2.20), bit error rates can be produced and analysis of internal channel characteristics can be obtained. These internal characteristics are difficult to obtain using classical analytical analysis methods. For the new technique, bit error rates are found by examining the cross-over shift of every bit in a data train and averaging that cross-over shift over the size of the data train. It has already been shown, in figures 3.3 - 3.6 that the cross-over shift of every bit in a data train can be examined accurately.

In typical recording channels, bit error rates of 1 error in 10^5 bits are common, so this new technique requires that data trains of at least 10 million bits are used, and therefore around 100 errors due to random channel noise are produced. The only aspect of this new technique that is yet to be examined in detail is the generation of AWGN samples. A classical method for producing AWGN samples is to use the Box, Muller and Marsaglia technique [Press, W. H., et al, (1994)]. This technique produces a fast simulation algorithm that allows for a feasible time scale when simulating a 10 million bit data train. The expression used to produce the random deviates with a gaussian distribution is given in equation (4.9).

$$p(y) dy = \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} dy \quad (4.9)$$

where $p(y)$ is the probability distribution of the random deviate y . An example of a noisy waveform produced using this method is shown in figure 4.9.

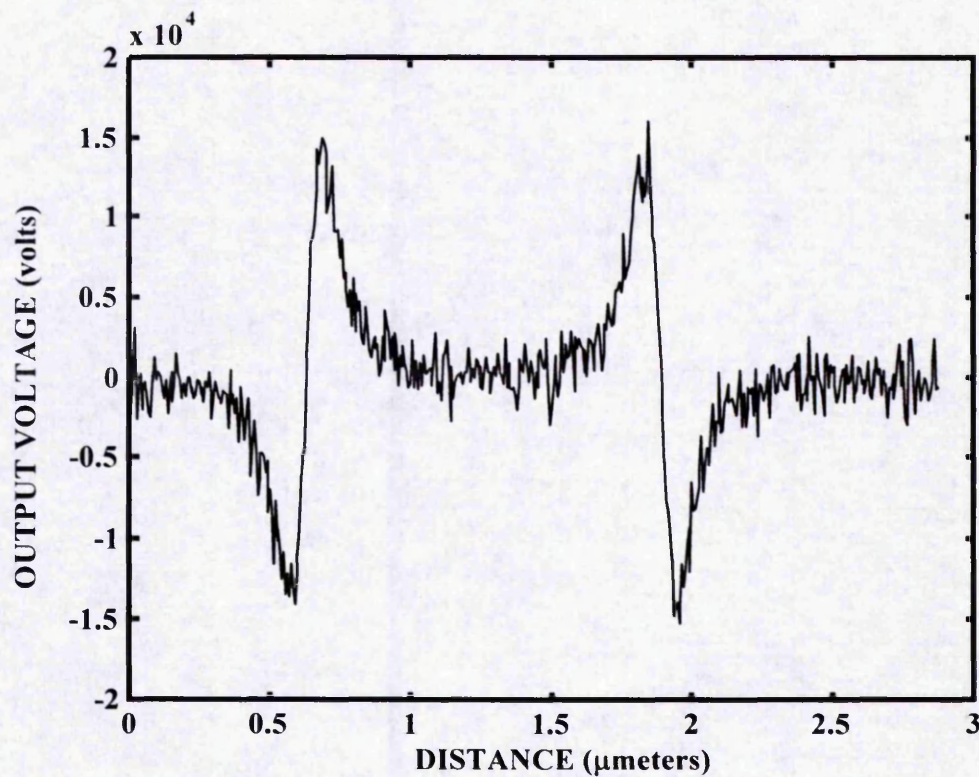


Figure 4.9: An example noisy differentiated waveform

A closer examination of the 'C' code will be performed to demonstrate some of the subtler aspects of the AWGN generator.

```

float ran2 (int *idum)
{
    int j, k;
    static int idum2 = 123456789;
    static int iy = 0;
    static int iv[NTAB];
    float temp;

    if (*idum <= 0)
    {
        if (-(*idum) < 1)
            *idum = 1;
        else
            *idum = -(*idum);
        idum2 = (*idum);
        for (j = (NTAB + 7); j >= 0; j--)
        {
            k = (*idum) / IQ1;
            *idum = IA1 * (*idum - k * IQ1) - k * IR1;
            if (*idum < 0)
                *idum += IM1;
            if (j < NTAB)
                iv[j] = *idum;
        }
        iy = iv[0];
    }
    k = (*idum) / IQ1;
    *idum = IA1 * (*idum - k * IQ1) - k * IR1;
    if (*idum < 0)
        *idum += IM1;
    k = idum2 / IQ2;
    idum2 = IA2 * (idum2 - k * IQ2) - k * IR2;
    if (idum2 < 0)
        idum2 += IM2;
    j = iy / NDIV;
    iy = iv[j] - idum2;
    iv[j] = *idum;
    if (iy < 1)
        iy += IMM1;
    if ((temp = AM * iy) > RNMX)
        return RNMX;
    else
        return temp;
}

```

Figure 4.10: 'C' code for the random number generator

The 'C' code for the pseudo-random number generator is shown in figure 4.10. The classical method used to generate random numbers is known as L'Ecuyer with Bays-Durham shuffle [Press, W. H., et al, (1994)]. The significant advantage that this number generator possesses is the desirable property of producing statistically independent random numbers for very long periods ($> 2 * 10^{18}$ numbers). In addition, it is a very speedy number generator, and so allows for feasible times when working with large data trains.

```
float add_noise (int *idum, float ex_1, float std_1)
{
    float fac, rsq, v1, v2;
    static int iset = 0;
    static float gset;

    if (iset == 0)
    {
        do
        {
            v1 = 2.0 * ran2(idum) - 1.0;
            v2 = 2.0 * ran2(idum) - 1.0;
            rsq = v1*v1+v2*v2;
        }
        while (rsq >= 1.0 || rsq == 0.0);

        fac = sqrt (-2.0 * log(rsq) / rsq);
        gset = ex_1 + (std_1 * v1 * fac);
        iset = 1;
        return (ex_1 + (std_1 * v2 * fac));
    }
    else
    {
        iset = 0;
        return (gset);
    }
}
```

Figure 4.11: 'C' code for the AWGN generator

Figure 4.11 contains the 'C' code that is used to generate AWGN samples using the Box, Muller and Marsaglia technique [Press, W. H., et al, (1994)]. This generator has the

advantage of generating two independent gaussian samples at every call to the procedure. Therefore, AWGN generation times are halved and this means that feasible simulation times occur when working with large data trains. The two input variables ex_I and std_I are the mean of the signal (which is assumed to be zero as its actual value is extremely small) and standard deviation of interest respectively. The standard deviation is defined in equation (4.10).

$$\text{Standard Deviation} = \frac{\text{noise voltage (RMS)}}{\sqrt{2}} \quad (4.10)$$

All the elements required to produce the new modelling technique have been examined in detail. This technique is implemented by applying AWGN samples to the sampled output of the differentiator and producing a recovered data stream. By comparing the data stream entering the model with the stream leaving the model, bit errors can be evaluated and bit error rates produced. Using a Hewlett Packard workstation, the time taken was 4½ hours for the model to handle a data train of 10 million bits.

4.6 A COMPARISON OF THE TWO TECHNIQUES

To determine the accuracy of the new sampled AWGN technique, two aspects of previous work are examined. By comparing the linear superposition effects and bit error rates produced by both techniques, the validity of the new technique can be established. Previous work includes previously published results [Hudson, V. N. et al. (1986); Middleton, B. K., and Jack-Kee, T. (1983)] obtained by experimental work on actual tape systems and

simulations produced using the classical analytical technique [Li Hui Hong, J.K. et al. (1993)].

The previous work, however, contained both longitudinal and perpendicular components, and is known as the mixed model [Li Hui Hong, J.K. et al. (1993)]. As this simulation can handle any pulse shape, mixed model pulses can be applied. The description of mixed model pulses is shown in equation (4.11).

$$e(x) = \frac{\mu_o v \omega n \eta M_x}{(1+\alpha)\pi} \ln \left[\frac{(a_t + d + D(1+\alpha))^2 + x^2}{(a_t + d)^2 + x^2} \right] - \frac{2\mu_o v \omega n \eta M_x}{(1+\alpha)\pi} \frac{M_y}{M_x} \left\{ \left[\arctan \left(\frac{x}{a_t + d + D(1+\alpha)} \right) \right] - \left[\arctan \left(\frac{x}{a_t + d} \right) \right] \right\} \quad (4.11)$$

with

M_y = The peak magnetisation of the medium (perpendicular component).

Equation (4.11) is differentiated to produce the output from the differentiator and is described in equation (4.12).

$$e(x) = \frac{\mu_o v \omega n \eta M_x}{500(1+\alpha)\pi} \left[\frac{2x((a_t + d)^2 - (a_t + d + D(1+\alpha))^2)}{((a_t + d + D(1+\alpha))^2 + x^2)((a_t + d)^2 + x^2)} \right] - \frac{2\mu_o v \omega n \eta M_x}{500(1+\alpha)\pi} \frac{M_y}{M_x} \left[\frac{a_t + d + D(1+\alpha)}{(a_t + d + D(1+\alpha))^2 + x^2} - \frac{a_t + d}{(a_t + d)^2 + x^2} \right] \quad (4.12)$$

Figures 2.10 and 2.11 show the isolated pulses defined by equation (4.11) and (4.12) respectively.

4.6.1 The Accuracy Of The Linear Superposition Of Pulses

By examining output voltage amplitude curves for a wide range of packing densities, the accuracy of the linear superposition theorem of the new technique is determined. Table 3 contains an output pulse measured experimentally [Hudson, V. N. et al. (1986); Middleton, B. K., and Jack-Kee, T. (1983)] and simulated [Li Hui Hong, J.K. et al. (1993)].

Figure 4.12 shows the corresponding output voltage amplitude as a function of record current for different packing densities for the pulse defined in table 3. These output amplitudes match very closely previously published experimental work [Hudson, V. N. et al. (1986)] and previous simulations [Li Hui Hong, J.K. et al. (1993)]. Thus confidence in the accuracy of the linear superposition of the new sampled waveform technique has been established.

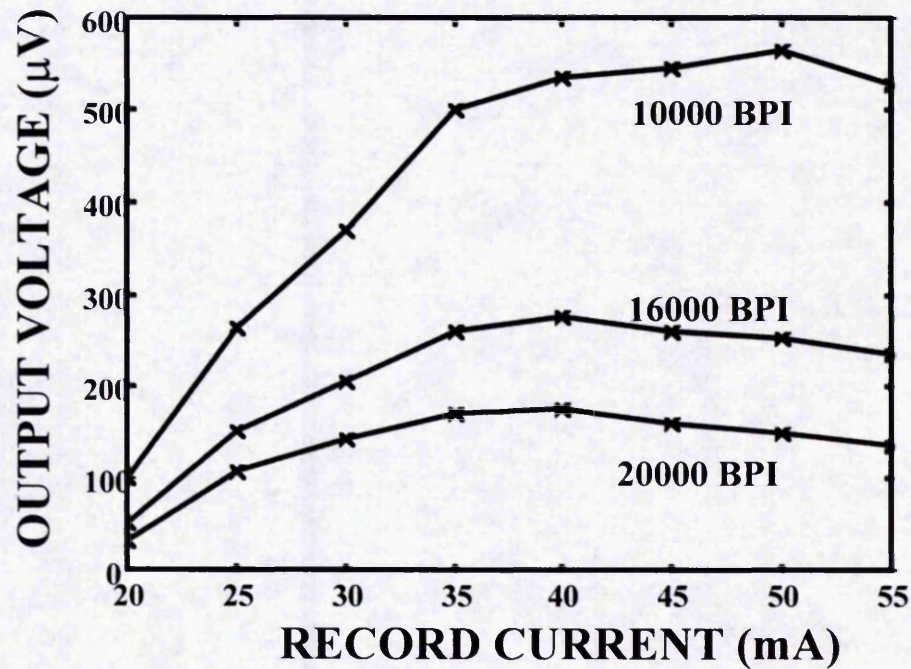


Figure 4.12: The output amplitude voltage as a function of record current

4.6.2 A Study Investigating the Performance Limitations due to ISI and Noise

4.6.2.1 Introduction

To minimise the probability of error of replayed information, the record current amplitude is optimised. The process of optimisation selects record current values which shape the replayed waveforms in such a way that the effects of variations of signal amplitude and peakshift i.e. intersymbol interference (ISI), are, in combination with noise, minimised and lead to the lowest error rate.

This study identifies a method which shows which of ISI and noise dominates the process of error formation at optimum record currents. This study operates at packing densities between 15,000 and 20,000 bits per inch for the pulse defined in table 3. As pulse equalisation is required for this study, a direct comparison with the new sampled AWGN technique is not possible. Therefore this study uses the classical analytical technique only.

4.6.2.2 Theoretical studies and experimental results

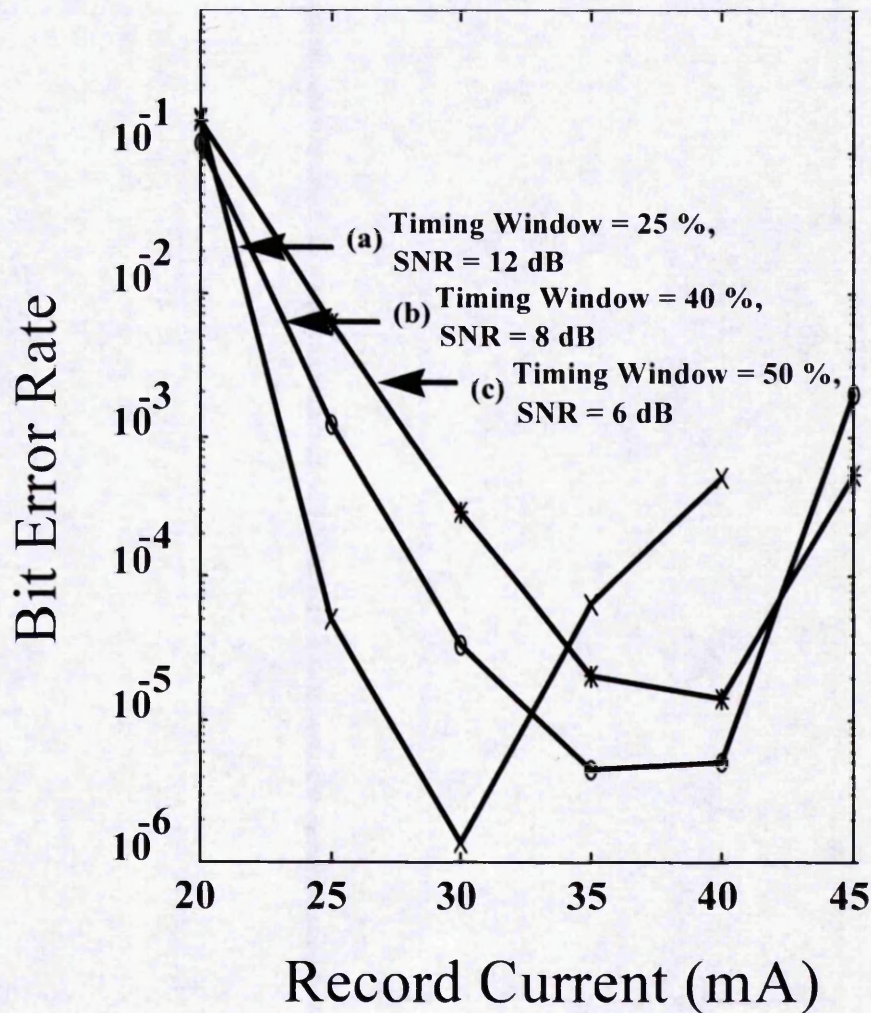


Figure 4.13: Bit error rates as a function of record current

Figure 4.13 shows some predicted error rate curves as a function of record current for different timing windows, expressed as a fraction of bit cell size, and different noise levels at a packing density of 17,000 bits per inch. Curve a shows a good agreement with the experimental curve produced in previous work [Hudson, V. N. et al. (1986)] while curves b and c for higher noise levels and wider timing windows show minima at higher record current levels. Note that the curve a in figure 4.13 has a minimum at a current amplitude of 30 mA which is close to that which minimises both $(a_t + d)$ and the pulse width. This indicates that since narrower pulses cause less overlapping at high densities, and therefore reduced peakshift, that current optimisation has minimised peakshift. Therefore the position of the minimum is an indicator that in this case performance is limited by the effects of peakshift. Curve c in figure 4.13, which corresponds to a high noise situation has its minimum at 40 mA which corresponds to the current which maximises output amplitude. Therefore it appears that in this case optimisation of current is to maximise signal to noise ratio and therefore for this setting the system was noise limited. This is a situation similar to those of Middleton and Brown [Middleton B.K., and Brown, T. (1980)] and Middleton and Jack-Kee [Middleton B.K., and Jack-Kee, T. (1983)]. Therefore the range of optimisation curves in figure 4.13 has been reflected in experimental observations. Curve b represents a situation between the two.

Figure 4.14 shows predictions of optimum current as a function of signal to noise ratio for three different timing windows. Some interpolation of results has taken place to produce more points on these curves than were originally obtained experimentally. It can be seen that higher noise and therefore lower signal to noise ratio pushes optimum current towards higher values to indicate noise limited performance as discussed above. Narrower timing

windows reduce optimum record current to indicate peakshift limited performance also as explained above.

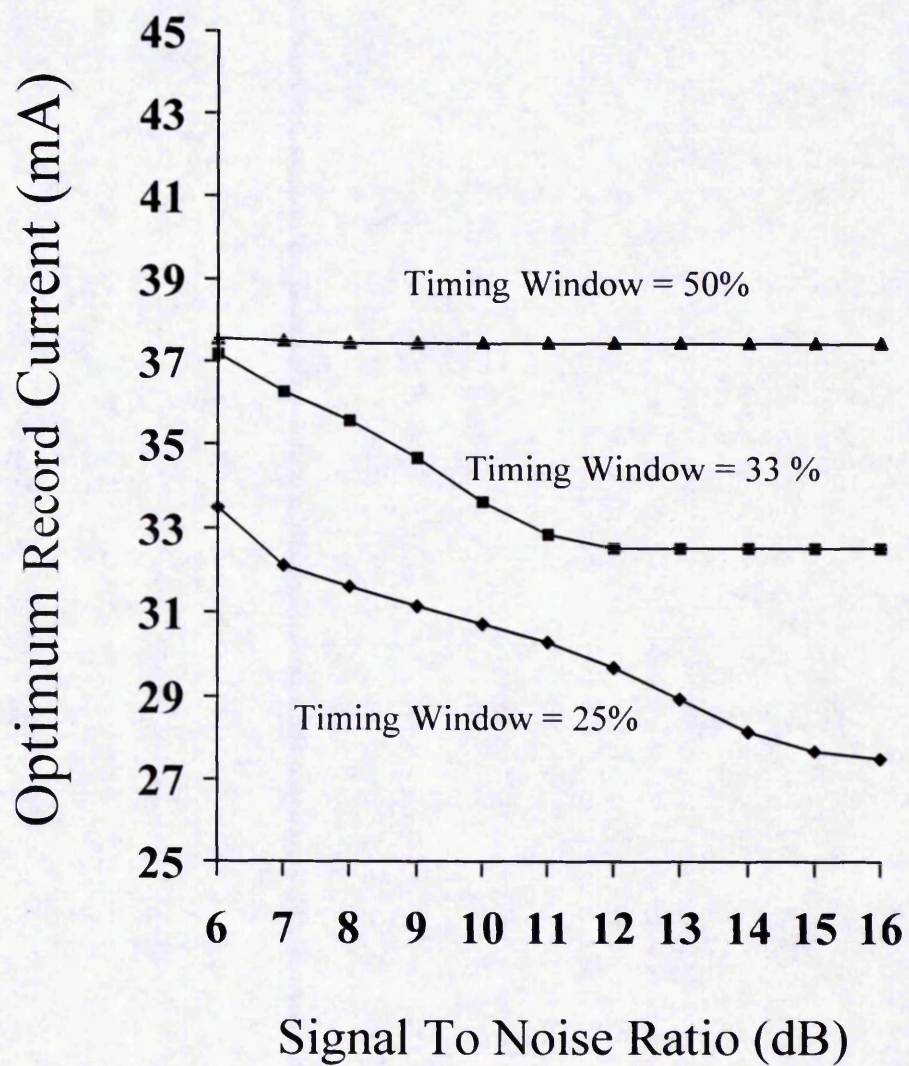


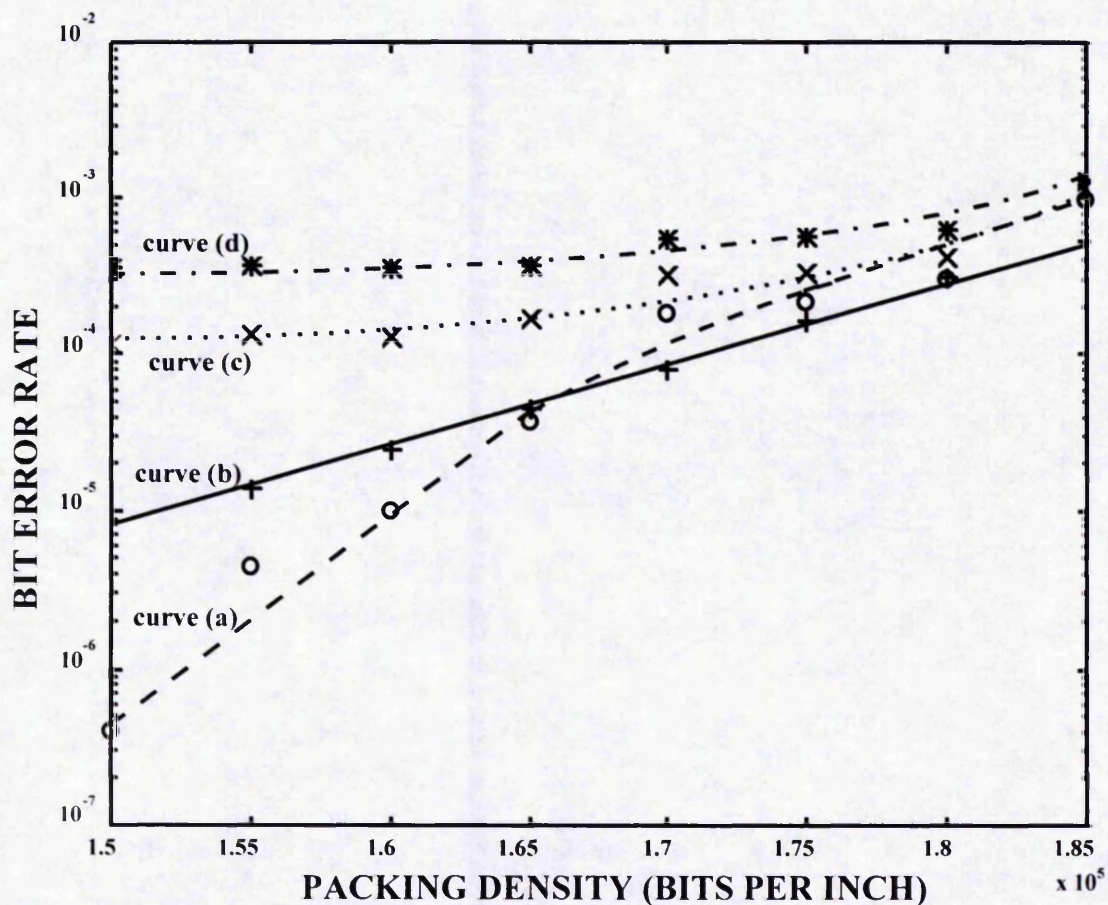
Figure 4.14: Optimum record current as a function of signal to noise ratio for different timing windows

4.6.2.3 Conclusions of this study

It has been shown by simulations that optimum record currents in digital recording systems are a reflection of the contributions of noise and peakshift to the error rates occurring in those systems. It has therefore been demonstrated that by plotting graphs of error rate as a function of record current level that it is possible to obtain information on the limiting processes which occur in those systems. Such information is important as it points to critical tape and system signal and noise characteristics in need of adjustment or alteration during system development.

4.7 BIT ERROR RATE COMPARISON FOR ADVANCED MEDIA

To achieve a comparison between the bit error rates of both techniques, advanced tape media is used as this media do not require pulse equalization. The pulses used in an advanced tape system have been defined in table 1, and are typical of advanced media such as metal evaporated tape or modern disk drives. Figure 4.15 shows the variation of bit error rate with packing density. Curves a and b are the bit error rates of the new sampled AWGN technique and the classical analytical technique respectively. There is a good level of agreement between curves a and b at the upper limits of the curves. The divergence in the lower limits of the curves occurs due to an approximation for the second differential of the signal in the classical analytical technique [Katz, E. R., and Campbell, T. G., (1979)]. This approximation does not exist in the new technique and therefore shows that the new technique produces more realistic simulations.



- Curve (a) - Bit error rates produced by the AWGN technique
- Curve (b) - Bit error rates produced by the classical analytical technique
- Curve (c) - Bit error rates produced by the AWGN technique including drop-outs across a single track tape
- Curve (d) - Bit error rates produced by the AWGN technique including drop-outs across an eight track tape

Figure 4.15: Bit error rate as a function of packing density

Curves c and d are the bit error rates produced, using the new technique with drop-outs included in a single track and multi-track situation respectively. Again, the trends at the upper limits of these curves are good, while at lower packing densities the curves tend to a limiting error rate. Thus confidence in bit error rates produced using the new technique has been verified and an examination of drop-out bit error rates have been produced.

4.8 ANALYSIS OF BURST ERROR EVENTS PRODUCED BY THE NEW SAMPLED AWGN TECHNIQUE

A significant advantage of the new technique is the ability for drop-out burst error events to be investigated in depth. In the past, such an investigation has not been possible as our previous models [Loze, M. K. et al. (1986)] had a limited data size of 1024 bits. The two aspects of burst error events that are examined are the internal cross-over shift characteristics that cause single or burst errors to occur and the comparison of data files before and after the modelling process which allow an analysis of burst error statistics.

4.8.1 Internal Cross-over Shift Characteristics

There are three cases to be considered when examining internal cross-over shift characteristics. Figures 4.16, 4.17 and 4.18 show the displacement of cross-overs from the clock position as a percentage of the timing window. The detector has been assumed to detect and operate on the first cross-over of a timing window which may contain multiple cross-overs. The poor signal to noise ratio and multiple cross-overs in timing windows cause the asymmetric behaviour in the three figures.

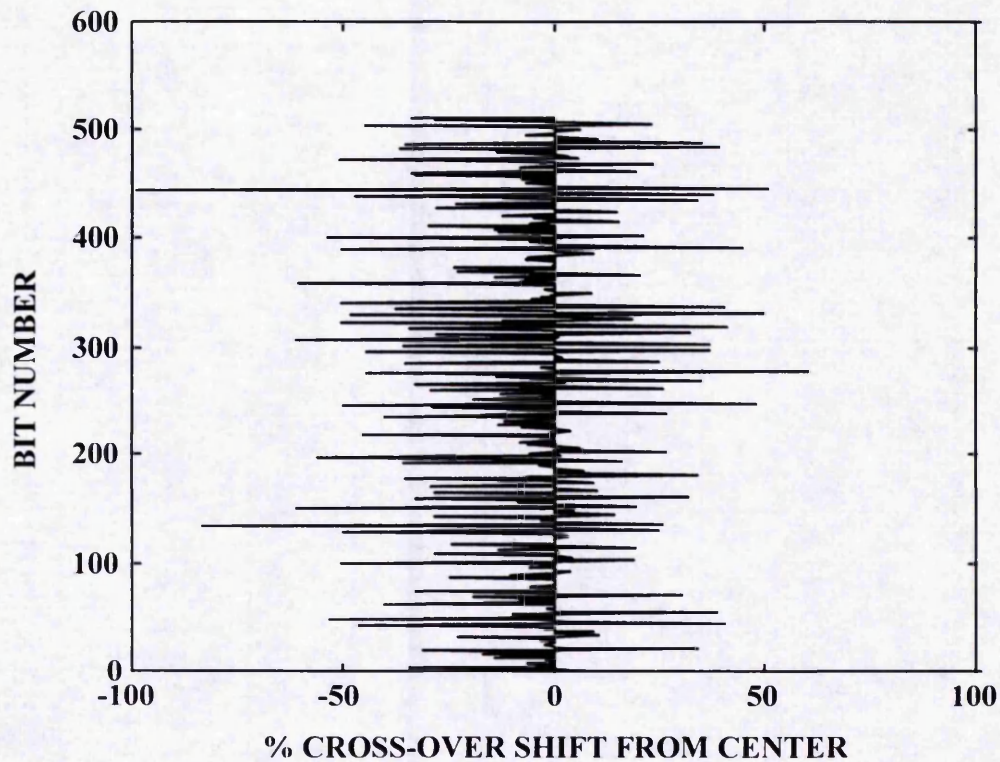


Figure 4.16: The internal cross-over shift effects due to random noise

Figure 4.16 shows one cross-over going past the timing window and is an example of a single error due to noise, figure 4.17 shows a display of errors corresponding to the passage of a small drop-out and figure 4.18 shows a display of errors corresponding to the passage of a large drop-out burst that causes intense error activity. The information in the above three figures is useful when designing error detection and correction codes.

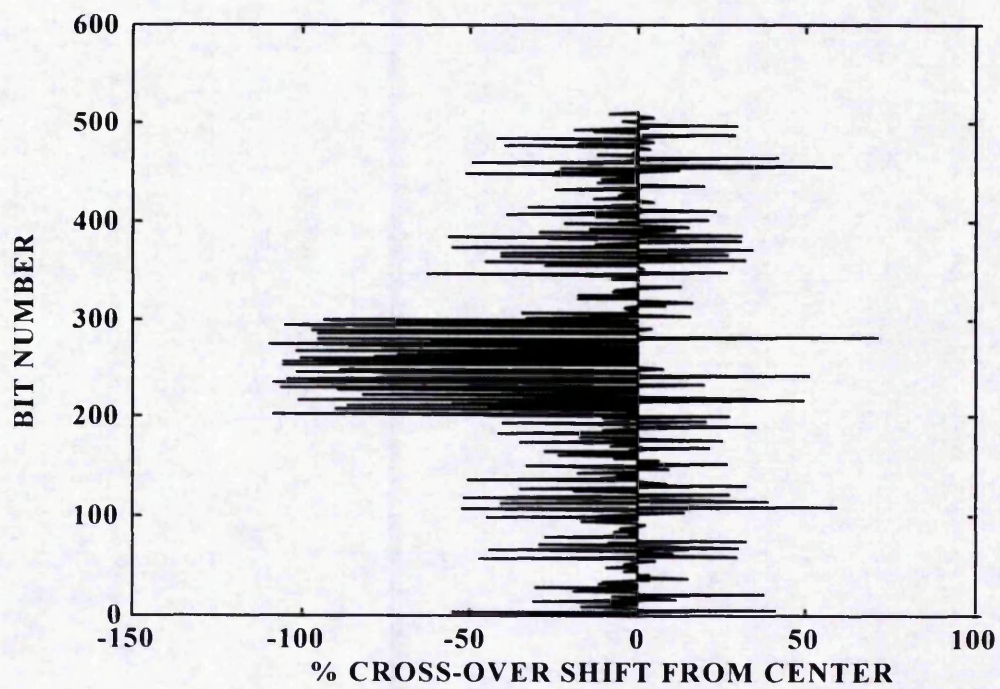


Figure 4.17: The internal cross-over shift effects due to a small drop-out burst

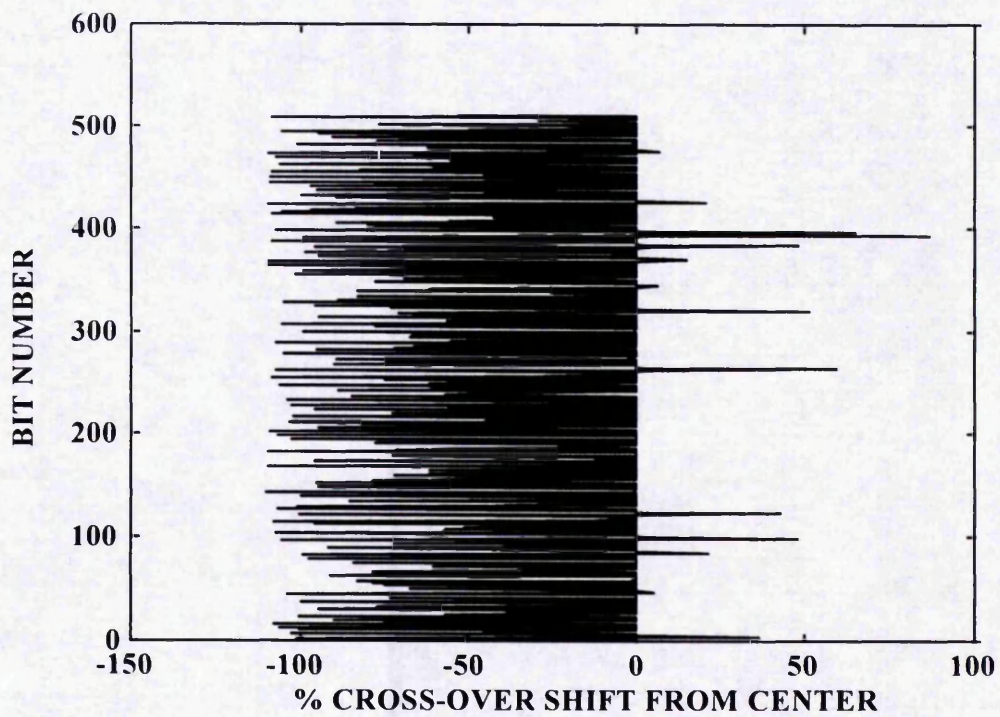


Figure 4.18: The internal cross-over shift effects due to a large drop-out burst

4.8.2 Comparison of Data Files Before And After The Modelling Process

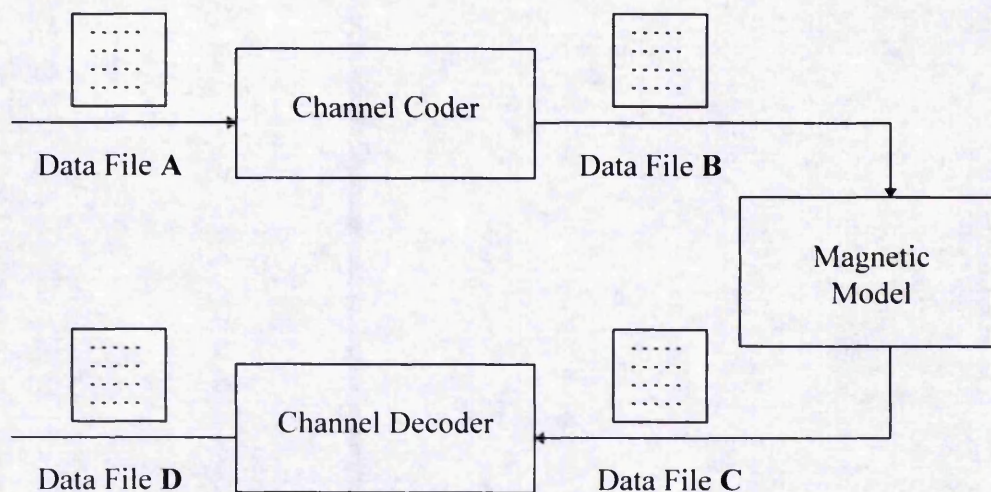


Figure 4.19: The simulation data file analysis

There are two cases where data files shown in figure 4.19 can be compared to obtain knowledge of the statistics of data that can occur due to random noise, drop-out bursts or a combination of both. The first case is gaining knowledge of statistics that occur due to the modelling process only by the comparison of data files B and C. The second case is gaining knowledge of the statistics that occur due to the channel coding and modelling process by the comparison of data files A and D. Since all the data files contain either 0's or 1's, comparison is straightforward using an XOR truth table as shown in 4.20 with '-' signifying no error has occurred and '*' signifying a bit in error.

0	0	-
0	1	*
1	0	*
1	1	-

Figure 4.20: An XOR truth table

--
--
**
--
--
--
--
--

Figure 4.21: File comparison of a random error across an eight track tape

Figures 4.21, 4.22, 4.23 show different types of errors that can occur when comparing data files A and D. In these figures, an eight track delay modulation coded tape has been assumed. These figures show the comparison of these two data files for a random error, a small multi-track error burst and a large multi-track error burst respectively.

```

-----
-----
-----*--**-----
--**--**-----**
-----
*-----**-----**
-----
-----

```

Figure 4.22: File comparison of a small multi-track error burst across an eight track tape

```

-----*--*-----**-----**-----*****--*-----**-----
--*-----*-----**-----**-----*-----
--**--*-----*-----**-----**-----**-----**--**
--*-----*-----*-----
-----**-----*-----**-----**-----*-----
-----*-----**-----**-----**-----**-----*-----
-----*-----**-----**-----**-----**-----**-----
**-----*-----**-----*-----**-----**-----**-----

```

Figure 4.23: File comparison of a large multi-track error burst across an eight track tape

For the analysis to be done in this thesis comparing data files B and C provides no useful additional information. However, when dealing with particular specific codes that combine channel coding with error coding [Obi, J. and Farrell, P. G., (1993)], the information in data files B and C is important. The information in data files A and D is useful for

examining error performance estimation of error detection and correction techniques used in Reed Solomon codes as interleaving [Wade, G., (1994)] and product codes [Sweeney, P., (1991)] are straightforward to implement.

A final consideration in analysing statistics is the definition of a burst. If we say that a burst has 1 empty guard space column, this means that at least 1 empty column on either side of the burst occurs. Thus the error-free 'guard space' on either side of the burst is at least 1 column. An empty column is defined as a column that contains no errors within it. Therefore, in the burst itself, each column must contain at least 1 error. If a burst has 5 guard space empty columns, as shown in figure 4.24, this means that there are at least 5 empty columns either side of the burst with a maximum of 4 empty columns allowable within the burst.

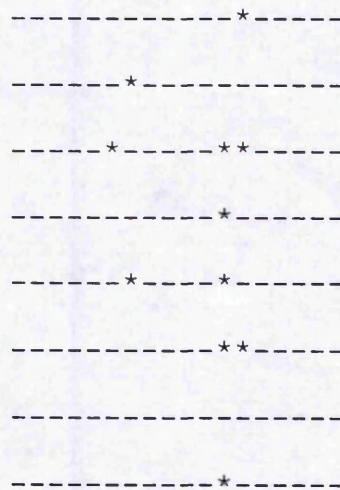


Figure 4.24: A burst with 5 empty guard space columns across an eight track tape

Statistics of bursts can also be analysed and the statistics of interest are:

- (1) *Max Length* is the maximum length of a burst in bits.
- (2) *Density* is the total number of errors in bursts / total size of bursts.
- (3) *Average Burst Rate* is the total number of bursts / size of a track.
- (4) *Average Burst Length* is the total length of bursts / total number of bursts.

with the size of a track being defined as the number of bits along a track.

EMPTY BURST COLUMNS	1	5	10	20
Max Length	25	503	583	866
Density	$1.612981 * 10^{-1}$	$1.097362 * 10^{-1}$	$8.919709 * 10^{-2}$	$7.062167 * 10^{-2}$
Average Burst Rate	$1.036548 * 10^{-3}$	$3.839066 * 10^{-4}$	$2.527385 * 10^{-4}$	$1.647599 * 10^{-4}$
Average Burst Length	2.407407	9.554167	17.85443	34.59223

Figure 4.25: Burst statistics

Figure 4.25 shows the burst statistics produced on the Hewlett Packard workstation using a number of different burst column sizes.

4.9 SUMMARY

A full implementation of the modelling of drop-outs has been conducted with the statistical analysis of drop-outs evaluated from previously published results.

AWGN has been produced using a new technique of applying sampled AWGN onto a sampled differentiated channel. AWGN was also produced using the channel statistical technique to verify the validity of the new technique. For comparison with previous work, pulse equalization (for the classical analytical technique only) and a mixed model (a model that contains both longitudinal and perpendicular components) were described.

The accuracy of the linear superposition of pulses and the bit error rates produced by both techniques were examined. It was found that the technique produced more accurate simulations as some of the assumptions made in the classical analytical technique were removed.

A study which examined the performance limitations due to ISI and noise was produced. It was shown by simulations that optimum record currents are a reflection of the contributions of noise and peakshift to the error rates occurring in recording systems. Higher noise pushes the optimum current towards higher values and so indicates noise limited performance. Narrower timing windows reduce record current and indicates peakshift limited performance. So, by plotting graphs of error rate as a function of record current level, it is possible to obtain information on the limiting processes which occur.

Finally, burst error event statistics, due to random noise and drop-outs, were investigated and can be used in the production of error detection and correction schemes. Internal cross-over shift characteristics were described and are intended for use with error correction schemes. A comparison of the data files produced before and after the modelling process was also conducted and this provides error performance estimation statistics.

5 THE EFFECTS OF INTERLEAVING ON ERROR PERFORMANCE ESTIMATION SCHEMES

5.1 INTRODUCTION

In the design of powerful error detection and correction (EDC) codes, for multi-track recording channels, Reed Solomon block symbol codes are used [Sweeney, P., (1991)]. It is particularly effective with Reed Solomon codes to use a powerful technique known as interleaving [Wade, G., (1994)] which alters the number of error symbols within a block and can therefore improve error performance. Important parameters that influence the efficiency and cost effectiveness of such codes include block size, interleave depth and the power of the symbol correcting code. Using error performance estimation techniques allows a designer to experiment with these parameters to produce efficient EDC codes.

An advantage of this simulation is the ability to analyse these parameters in depth with a minimal time penalty. As well as bit errors and bit error rates, symbol errors, symbol error rates, block errors and block error rates are produced. Therefore the designer is provided with the information required to improve error performance depending on the availability of resources such as the amount of memory that the system possesses.

Different approaches for the implementation of interleaving exist. Horizontal stack (i.e. symbols along tracks) symbol interleaving, vertical stack (i.e. symbols across tracks) symbol interleaving, horizontal stack block interleaving and vertical stack block interleaving will be analysed. The effects of combining block and symbol interleaving will

also be investigated. These different approaches will then be compared and appropriate approaches for different scenarios will be discussed.

5.2 REED SOLOMON CODES

Reed Solomon (RS) codes [Sweeney, P., (1991)] have qualities, applicable to multi-track tape systems, that make use of these codes more attractive than other coding schemes. In particular, RS codes are symbol oriented rather than bit oriented in nature. This is ideal for a multi-track tape system which is organised in terms of groups of bit streams rather than a single stream of bits. Another attractive feature of RS codes is the ability to efficiently handle burst errors that span a number of adjacent symbols, as often occurs in multi-track tape systems [Abdel-Ghaffar, K. A. S., and Hassner, M., (1991)]. The parameters of RS codes are as follows:

$$n = 2^m - 1 \text{ or } 2^m \text{ or } 2^m + 1 \quad (5.1)$$

$$n - k = 2t \quad (5.2)$$

$$d = 2t + 1 (= (n - k) + 1) \quad (5.3)$$

$$R = k/n = (n - 2t) / n \quad (5.4)$$

Equations (5.1) - (5.4) are the parameters of a t - error correcting RS code with symbols drawn from GF (2^m) [Sweeney, P., (1991)]. The equations (5.1) - (5.4) use m bit words, n

symbol codewords (i.e. a block size of n), k message symbols and a minimum distance between codewords of d . RS codes have good performance as they are maximum distance separable, since $d = n - k + 1$ and this is the highest possible value that d can have, for a given n and k . The values of n in equation (5.1) of 2^m and $2^m + 1$ refer to extended and doubly extended RS codes respectively.

The codes used below all have 8-bit symbols (i.e. $m = 8$). The maximum block length used is 256 symbols, corresponding to the extended case. The other block lengths used correspond to shortened RS codes, where the appropriate number of information symbols are set to zero to produce the block length desired. A wide range of values of t are used, with corresponding effect on the rate, R , of the code. Increasing the value of t reduces the rate R , for a constant value of n , as shown in equation (5.4). A suitable generator polynomial for a t error correcting RS code is:

$$g(x) = (x + \alpha^1) (x + \alpha^2) (x + \alpha^3) (x + \alpha^4) \dots\dots\dots (x + \alpha^{2t}) \quad (5.5)$$

RS codes, because of their cyclic behaviour, are normally decoded using algebraic techniques [Sweeney, P., (1991)] which involve the solution of syndrome equations to find the location and value of the symbol errors. However, this thesis will focus on assessing the performance of different scenarios without actually conducting the decoding process.

In practice, decoding chips for RS codes have been implemented [Green, M. B., and Drolet, G., (1992); Hasan, M. A., and Bhargava, V. K. (1992)]. It is hoped that working on error performance estimation schemes of RS codes will lead to improved algorithms which can then be implemented in the next generation of decoding chips.

5.3 INTERLEAVING TECHNIQUES

5.3.1 Horizontal Symbol Stack Arrangement

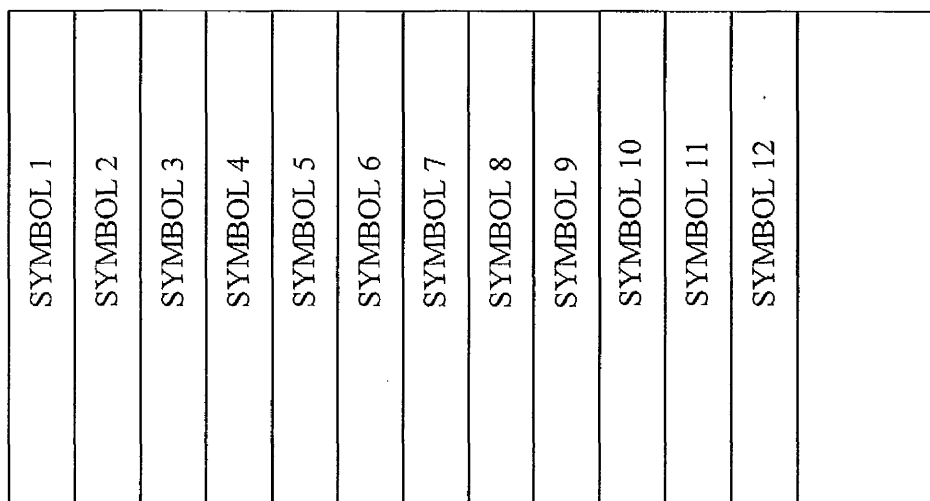


Figure 5.1: The horizontal stacking arrangement of symbols

Figure 5.1 shows the arrangement of symbols required to produce a horizontal symbol stack on a multi-track tape system. It can be seen that the symbols are positioned vertically across the tape tracks but the stack is horizontal. If a burst of errors occurs on the tape medium due to for example, a drop-out, the errors are contained within the symbols. With a powerful enough EDC code, these symbols can be corrected and therefore the effects of the burst of errors can be removed or at least diminished.

5.3.2 Vertical Symbol Stack Arrangement

SYMBOL 1	SYMBOL 9	
SYMBOL 2	SYMBOL 10	
SYMBOL 3	SYMBOL 11	
SYMBOL 4	SYMBOL 12	
SYMBOL 5	SYMBOL 13	
SYMBOL 6	SYMBOL 14	
SYMBOL 7	SYMBOL 15	
SYMBOL 8	SYMBOL 16	

Figure 5.2: The vertical stacking arrangement of symbols

Figure 5.2 shows the arrangement of symbols required to produce a vertical symbol stack on a multi-track tape system. It can be seen that the symbols are positioned horizontally down the tape (along the tape tracks) but the stack is vertical. Once the bottom of the tape is reached, the stack is continued at the top of the tape at the next available position along the tape. The information in figure 4.22 provides the reason that this arrangement of symbol stacking is being investigated. From figure 4.22, it is seen that if symbols were stacked vertically and a burst of errors occurred, there would be more bits in error in each symbol with less error symbols and therefore the possibility exists that this stacking arrangement will lead to more efficient EDC coding than can be achieved by using horizontal stacking.

5.3.3 Symbol Interleaving

SYMBOL 1
SYMBOL 5
SYMBOL 2
SYMBOL 6
SYMBOL 3
SYMBOL 7
SYMBOL 4
SYMBOL 8
SYMBOL 9
SYMBOL 13
SYMBOL 10
SYMBOL 14

Figure 5.3: Symbol interleaving for the horizontal stack arrangement

SYMBOL 1	SYMBOL 9	
SYMBOL 5	SYMBOL 13	
SYMBOL 2	SYMBOL 10	
SYMBOL 6	SYMBOL 14	
SYMBOL 3	SYMBOL 11	
SYMBOL 7	SYMBOL 15	
SYMBOL 4	SYMBOL 12	
SYMBOL 8	SYMBOL 16	

Figure 5.4: Symbol interleaving for the vertical stack arrangement

Figures 5.3 and 5.4 show symbol interleaving for the horizontal and vertical stack arrangements respectively for, as an example, a symbol interleaving depth of 2 with an EDC code block size of 4. The process of symbol interleaving involves rearranging the

symbols within each block according to a specified interleaving depth. For a depth of 2 and block size of 4, the first symbol of the original block is placed into the first slot of the symbol interleaved stack. The fifth symbol is then put in the second slot (as it is the first symbol of the second block). Following this, the second symbol is placed in the third slot and the sixth symbol is put in the fourth slot. This is repeated until all the slots of the symbol interleaved stack are filled. Thus figure 5.3 is the symbol interleaved stack of figure 5.1, and figure 5.4 is the symbol interleaved stack of figure 5.2.

5.3.4 Block Interleaving

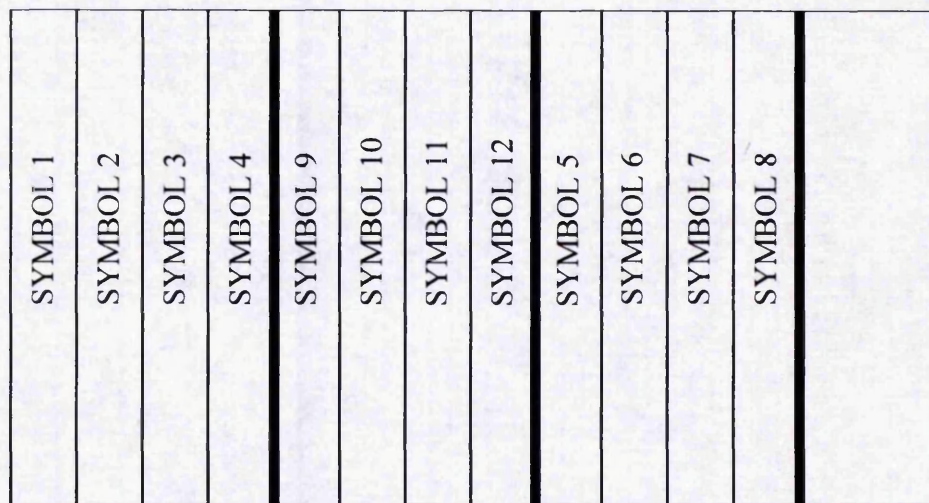


Figure 5.5: Block interleaving for the horizontal stack arrangement

SYMBOL 1	SYMBOL 5	
SYMBOL 2	SYMBOL 6	
SYMBOL 3	SYMBOL 7	
SYMBOL 4	SYMBOL 8	
SYMBOL 9	SYMBOL 13	
SYMBOL 10	SYMBOL 14	
SYMBOL 11	SYMBOL 15	
SYMBOL 12	SYMBOL 16	

Figure 5.6: Block interleaving for the vertical stack arrangement

Figures 5.5 and 5.6 show block interleaving for the horizontal and vertical stack arrangements respectively for, as an example, an EDC code block size of 4 and a block interleaving depth of 2. The process of block interleaving involves rearranging blocks of symbols according to a specified block interleaving depth. For a block interleaving depth of 2, the first four symbols (symbols 1, 2, 3 and 4) of the original block are placed into the first four slots of the block interleaved stack. The next four slots are filled with the symbols in the third block, i.e. symbols 9, 10, 11 and 12. The next four slots are then filled with the symbols in the second block, i.e. symbols 5, 6, 7 and 8. The next four slots are then filled with the symbols in the fourth block, i.e. symbols 13, 14, 15 and 16. This process is repeated until all the slots of the block interleaved stack are filled. Thus figure 5.5 is the block interleaved stack of figure 5.1, and figure 5.6 is the block interleaved stack of figure 5.2.

5.3.5 Block and Symbol Combined Interleaving

SYMBOL 1
SYMBOL 9
SYMBOL 2
SYMBOL 10
SYMBOL 3
SYMBOL 11
SYMBOL 4
SYMBOL 12
SYMBOL 5
SYMBOL 13
SYMBOL 6
SYMBOL 14

Figure 5.7: Combined interleaving for the horizontal stack arrangement

SYMBOL 1	SYMBOL 5	
SYMBOL 9	SYMBOL 13	
SYMBOL 2	SYMBOL 6	
SYMBOL 10	SYMBOL 14	
SYMBOL 3	SYMBOL 7	
SYMBOL 11	SYMBOL 15	
SYMBOL 4	SYMBOL 8	
SYMBOL 12	SYMBOL 16	

Figure 5.8: Combined interleaving for the vertical stack arrangement

Figures 5.7 and 5.8 show combined interleaving for the horizontal and vertical stack arrangements respectively for, as an example, an EDC code block size of 4 and a block interleaving depth of 2 and a symbol interleaving depth of 2. The process of block and symbol combined interleaving involves rearranging both the blocks of symbols and the symbols within the block. These individual processes have been described in sections 5.2.3 and 5.2.4 with block interleaving being performed before symbol interleaving is carried out. Thus figure 5.7 is the combined interleaved stack of figure 5.1, and figure 5.8 is the combined interleaved stack of figure 5.2.

5.4 ERROR PERFORMANCE ESTIMATION INFORMATION PRODUCED BY THE DIFFERENT INTERLEAVING TECHNIQUES

5.4.1 Block, Symbol and Bit Errors and Error Rate Analysis

Symbols corrected (<i>t</i>)	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	676	1.379873e-1	1560	2.438475e-3	2013	4.012697e-4
2	218	4.449888e-2	1145	1.789778e-3	1367	2.724966e-4
3	141	2.878138e-2	929	1.452143e-3	1071	2.134922e-4
4	79	1.612574e-2	398	6.221238e-4	710	1.415308e-4
5	45	9.185548e-3	230	3.595188e-4	483	9.628081e-5
6	27	5.511329e-3	132	2.063325e-4	304	6.059910e-5
8	9	1.837110e-3	84	1.313025e-4	116	2.312334e-5
10	2	4.082466e-4	21	3.282563e-5	31	6.179514e-6
11	1	2.041233e-4	11	1.719438e-5	17	3.388766e-6
12	0	0	0	0	0	0

Table 5: Analysis of block, symbol and bit errors and error rates at interleaving depth 100 and block size 128 using horizontal stack symbol interleaving

Table 5 shows the analysis of bit, block symbol errors and error rates at an interleaving depth of 100 and a block size of 128 symbols. The stacking arrangement used in table 5 was horizontal and only symbol interleaving was performed. The number of errors are those occurring in a sample of 10,000,384 bits (= 1,250,048 symbols). The table shows how many symbols the code must be capable of correcting in order to remove all the errors in the sample.

Table 5 produces the maximum amount of information about the effects of symbol correction on blocks, symbols and bits. Therefore the designer has the scope to experiment with altering, for example, the block size and examining the effects on blocks, symbols and bits. Thus if the EDC code is block, symbol or bit orientated, the designer can focus on the appropriate columns of table 5. In sections 5.4.2 and 5.4.3, the focus is on the analysis of symbols as RS codes are symbol orientated in nature.

Appendix C.1 contains the results of a detailed investigation of bit, block and symbol errors and error rates for a wide range of interleaving depths and block sizes. A horizontal stack with symbol interleaving was used for this investigation.

5.4.2 Analysis of the Number of Symbols to be Corrected

INTERLEAVING						
Symbols corrected (<i>t</i>)	horizontal stack - symbol	vertical stack - symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
3	745	548	1285	709	779	522
5	327	265	1040	627	259	172
6	219	151	986	573	91	112

7	128	95	944	566	56	42
8	88	47	928	558	0	18
9	43	20	901	558	0	0
10	23	0	841	548	0	0
11	12	0	841	526	0	0
12	0	0	817	514	0	0
18	0	0	676	257	0	0
25	0	0	343	26	0	0
26	0	0	291	0	0	0
34	0	0	35	0	0	0
35	0	0	0	0	0	0

Table 6: Analysis of number of symbols to be corrected at an interleaving depth 100 and a block size of 128 for different stack and interleaving scenarios

Table 6 shows the analysis of the number of symbols to be corrected at an interleaving depth of 100 and a block size of 128. The analysis was performed with both the horizontal and vertical stacking arrangements and with the three interleaving techniques.

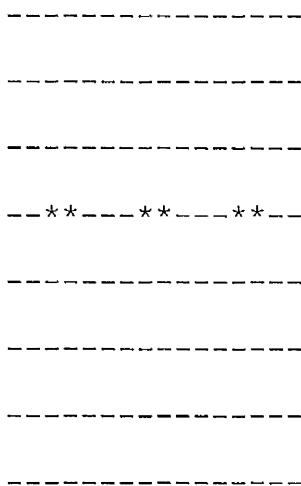


Figure 5.9: A single track burst that can occur in tape systems

It is observed that without any symbols being corrected there is a difference in the number of symbols to be corrected in the vertical stack compared with the horizontal stack. This difference is due to the nature of bursts in tape systems. Figure 5.9 shows this difference using a single track burst case that can occur in tape systems. Figure 5.9, if handled using horizontal stacking, produces six error symbols with each symbol containing one error. If, however, vertical stacking is used then two error symbols are produced with three errors contained in each of these symbols. Therefore it is not surprising that the vertical stack has fewer error symbols.

Appendix C.2 contains the results of a detailed investigation of the number of symbols to be corrected for a wide range of interleaving depths and block sizes. In table 6, the results columns show the number of symbols left to be corrected at an interleaving depth 100 and a block size of 128. So for horizontal stacking with symbol interleaving, for a three-symbol correcting EDC code, that there are 745 error symbols left to correct, but for a ten-symbol correcting EDC code, there are only 23 error symbols left to correct. A result of 0 indicates that this EDC code can correct all the error symbols produced. For horizontal stacking, with symbol interleaving, a twelve-symbol correcting EDC code is the minimum required to correct all the error symbols.

By examining all the results columns it is clear that a combination of both symbol and block interleaving produces the best results (i.e. the minimum possible value of t required to correct all the error symbols), with horizontal stacking producing better results than vertical stacking. This result however, is specific to an interleaving depth of 100 and a block size of 128. For other block sizes and interleave depths, as shown in appendix C.2, the best results can be obtained by following other interleaving and stacking strategies. For

example, for an interleave depth of 200 and a block size of 192, the best results are produced using a vertical stack and symbol only interleaving strategy, whereas for an interleave depth of 1000 and a block size of 256 a combination of both symbol and block interleaving produces the best results but it does not matter about the stack being vertical or horizontal. By examining the trends of the symbol error rates, it may be possible to obtain strategies for producing the best results for different scenarios.

For all the results shown in appendix C.2, and in table 6, it is clearly demonstrated that doing the block interleaving process on its own produces the worst results, and so this is disregarded from the analysis at an interleaving depth of 5000 and in the results produced in appendix C.3. Also no trends in section 5.4.4 have been analysed which use the block interleaving process on its own.

5.4.3 Symbol Error Rate Analysis

INTERLEAVING				
Symbols corrected (<i>t</i>)	horizontal stack - symbol	vertical stack - symbol	horizontal stack - both	vertical stack - both
0	2.487753e-3	1.618634e-3	2.487753e-3	1.618634e-3
3	1.188061e-3	8.739028e-4	1.242282e-3	8.324403e-4
5	5.214712e-4	4.225990e-4	4.130307e-4	2.742907e-4
6	3.492422e-4	2.408017e-4	1.451189e-4	1.786079e-4
7	2.041233e-4	1.514978e-4	8.930394e-5	6.697795e-5
8	1.403348e-4	7.495152e-5	0	2.870484e-5
9	6.857267e-5	3.189426e-5	0	0
10	3.667840e-5	0	0	0
11	1.913656e-5	0	0	0
12	0	0	0	0

Table 7: Analysis symbol, error rates at interleaving depth 100 and block size 128 for different stack and interleaving scenarios

Table 7 shows the analysis of symbol error rates at an interleaving depth of 100 and a block size of 128. The analysis was performed with both the horizontal and vertical stacking arrangements using the symbol interleaving only technique and using the both block and symbol interleaving technique.

Appendix C.3 contains the results of a detailed investigation of symbol error rates for a wide range of interleaving depths and block sizes. In section 5.4.2, it was shown that using block interleaving only produced the worst results in all cases as shown in appendix C.2. So no more block interleaving only analysis was done. This is reflected in appendix C.3 where the block interleaving columns have been left blank. Appendices C.2 and C.3 have the same form format so that a direct comparison of number of errors to be corrected and symbol error rates can be made for specific block sizes and interleave depths.

From the results in appendix C.3, figures 5.10 to 5.21 were produced which illustrate the trends of block sizes and interleaving depths, and will be examined in detail for the different scenarios.

5.4.4 Symbol Error Rate Trends

Two sets of trends will be discussed, the trends that occur at different interleaving depths and the trends that occur with a range of block sizes for specific interleaving depths. The aim is to determine the trends that produce the lowest correcting power EDC symbol codes required to remove all the symbol errors from the simulation. This is desirable as a low t (high rate) EDC symbol code is more efficient, cheaper and simpler to produce in a product.

For all the plots the value $8.000000e-7$ (which is lower than 10^{-6} due to the size of the sample) is used to replace the first result of 0 in the tables in appendix C.3. This is done to illustrate the trend of the curves and because 0 can not be plotted on a log scale.

5.4.4.1 Trends that occur at different interleaving depths

5.4.4.1.1 Trends that occur using only symbol interleaving

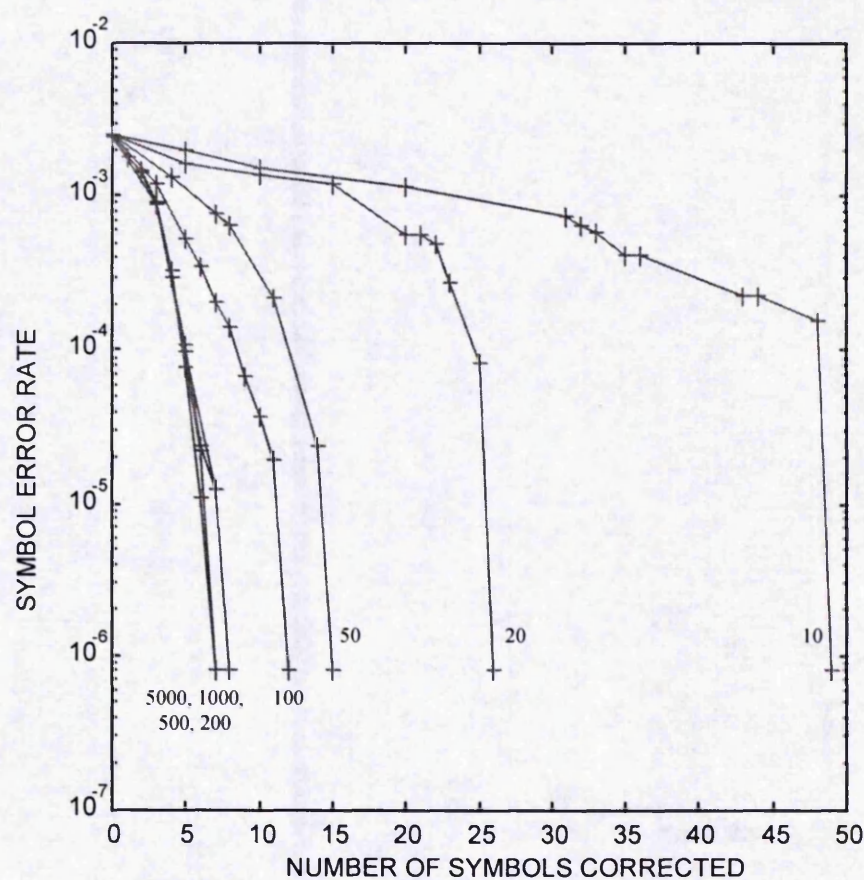


Figure 5.10: Interleaving depth trends for symbol interleaving only with horizontal stacking arrangement for block size 128

Figure 5.10 shows the trends produced using symbol interleaving only with the horizontal stacking arrangement, for a block size of 128. The block size used for figures 5.10 to 5.13 is not significant as all the block sizes have been found to have similar interleaving depth trends. Figure 5.10 shows that high interleaving depths enable the use of a low t EDC symbol code to remove all the symbol errors out of the simulation. The best trend occurs at a depth of 5000 where an EDC symbol code with $t = 7$ (i.e. a RS code that can correct 7 symbols) can remove all the symbol errors out of the system.

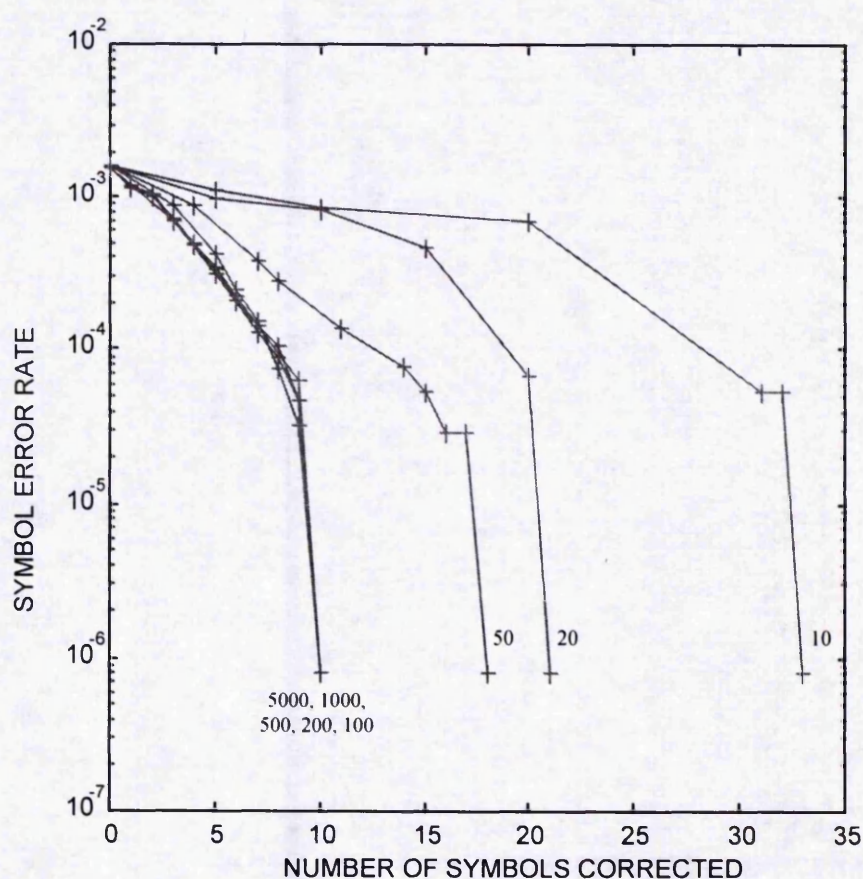


Figure 5.11: Interleaving depth trends for symbol interleaving only with vertical stacking arrangement for block size 128

Figure 5.11 shows the trends produced using symbol interleaving only with the vertical stacking arrangement, for a block size of 128. As in figure 5.10, high interleaving depths produce the lowest t EDC symbol code required to remove all the symbol errors out of the system. The best trend occurs at depths above 100 where an EDC symbol code with $t = 9$ symbols (i.e. a RS code that can correct 9 symbols) can remove all the symbol errors out of the system.

Comparing the two stacking arrangements for symbol interleaving reveals that using high interleaving depths is a significant technique for producing the low t EDC symbol codes required to remove all the symbol errors out of the simulation. It is also found that horizontal stacking produces better results than vertical stacking as a $t = 9$ EDC symbol code is required to remove all the errors from the simulation in vertical stacking whereas horizontal stacking requires only a $t = 7$ EDC symbol code, at an interleaving depth of 5000.

5.4.4.1.2 Trends that occur using both block and symbol interleaving

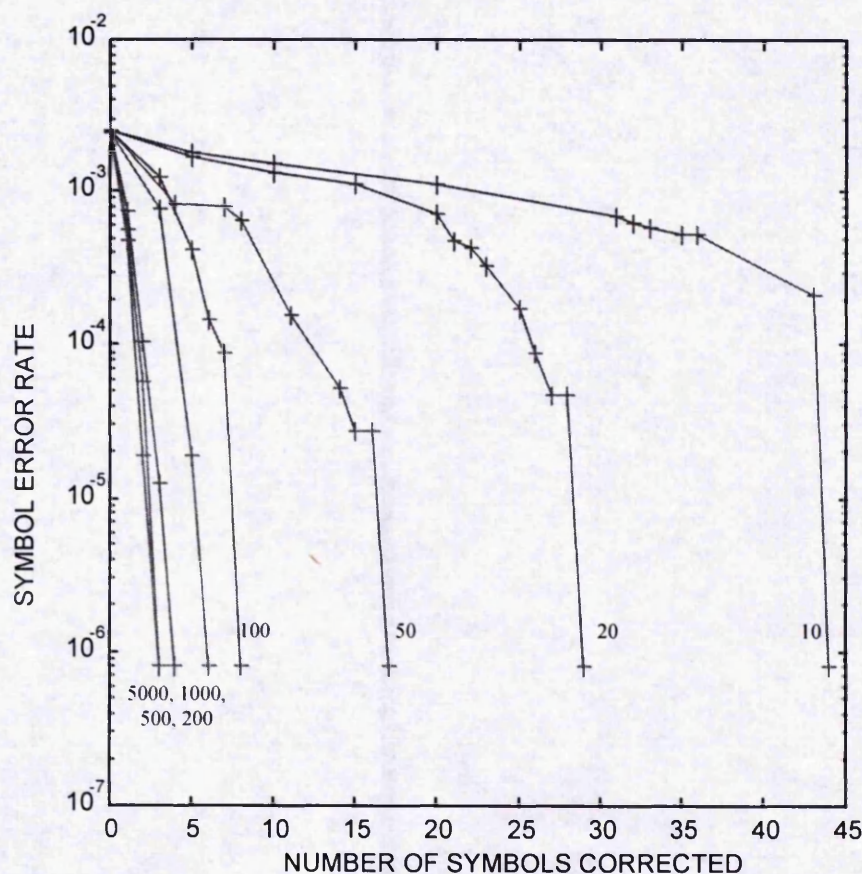


Figure 5.12: Interleaving depth trends using both block and symbol interleaving with horizontal stacking arrangement for block size 128

Figure 5.12 shows the trends produced using both block and symbol interleaving with the horizontal stacking arrangement, for a block size of 128. Again high interleaving depths produce the lowest t EDC symbol code required to remove all the symbol errors from the system. The best trend occurs at a depth of 5000 where an EDC symbol code with $t = 3$ symbols (i.e. a RS code that can correct 3 symbols) can remove all the symbol errors out of

the system. This is significantly better performance than was achievable using symbol interleaving only.

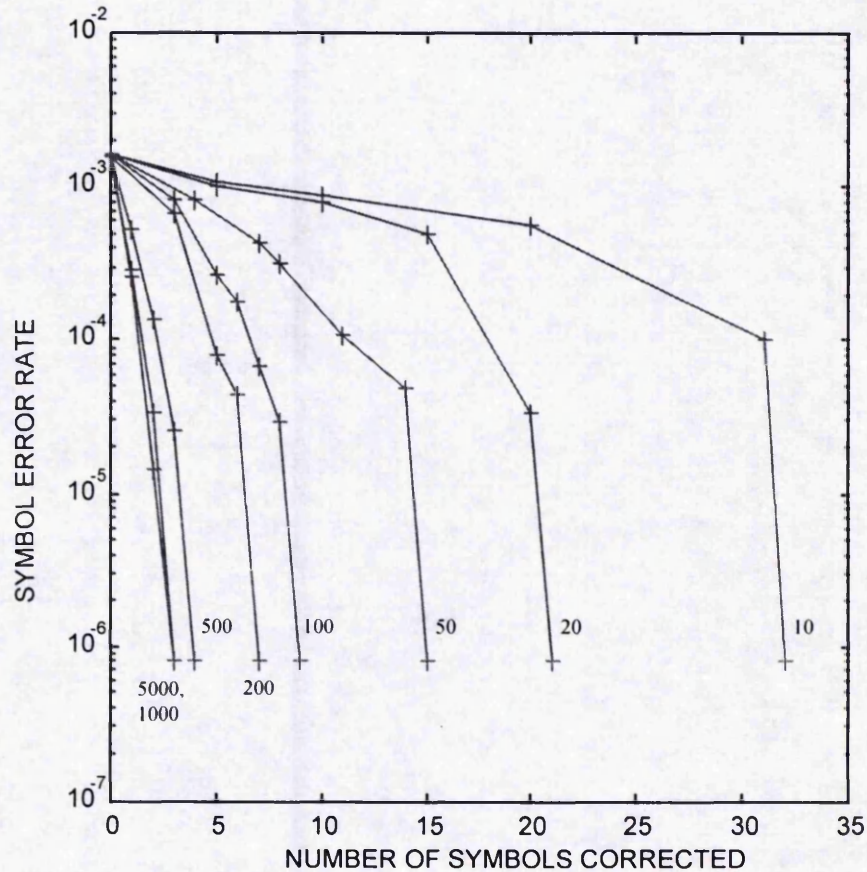


Figure 5.13: Interleaving depth trends using both block and symbol interleaving with vertical stacking arrangement for block size 128

Figure 5.13 shows the trends produced using both block and symbol interleaving with the vertical stacking arrangement, for a block size of 128. Again high interleaving depths produce the lowest t EDC symbol code required to remove all the symbol errors from the system. The best trend occurs at a depth of 5000 where an EDC symbol code with $t = 3$ symbols (i.e. a RS code that can correct 3 symbols) can remove all the symbol errors from

the system. This is significantly better performance than was achievable using symbol interleaving only, but is the same result as was achieved with horizontal interleaving.

Comparing the two stacking arrangements for both block and symbol interleaving reveals that using high interleaving depths is a significant technique for producing the low EDC symbol codes required to remove all the symbol errors out of the simulation. It is also found that horizontal stacking produces similar results to vertical stacking at a depth of 5000 where, for both stacking arrangements, a $t = 3$ EDC symbol code removes all the symbol errors out of the simulation.

5.4.4.2 Trends that occur with a range of block sizes for specific interleaving depths

In section 5.4.4.1, there has been a focus on getting the best possible results from the simulation with no consideration of limitations placed upon the designer. This section will focus on examining the effects of the stacking and interleaving strategies at a range of interleaving depths and block sizes, and therefore allow the designer to, depending on the limitations (such as memory available), make a sensible choice of what strategy at a particular blocking size and interleaving depth is appropriate. In figures 5.14 to 5.21 the curves are labelled **A**, **B**, **C** and **D**. The meaning of these labels are as follows:

A - A horizontal stacking arrangement with symbol interleaving only.

B - A vertical stacking arrangement with symbol interleaving only.

C - A horizontal stacking arrangement with both block and symbol interleaving.

D - A vertical stacking arrangement with both block and symbol interleaving.

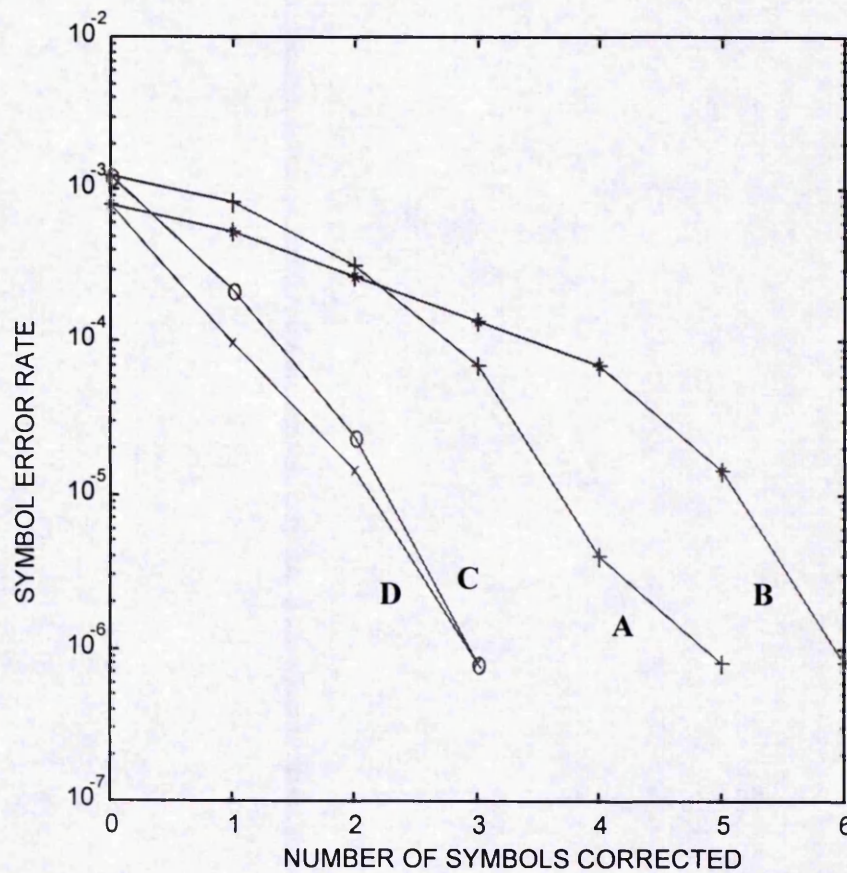


Figure 5.14: Trends of the different strategies at block size 256 and interleaving depth 5000

Figure 5.14 shows the trends of the different strategies at a block size of 256 and an interleaving depth of 5000. It is clear that the both curves **C** and **D** produce the best results i.e., both block and symbol interleaving should be used and the stacking arrangement is not important.

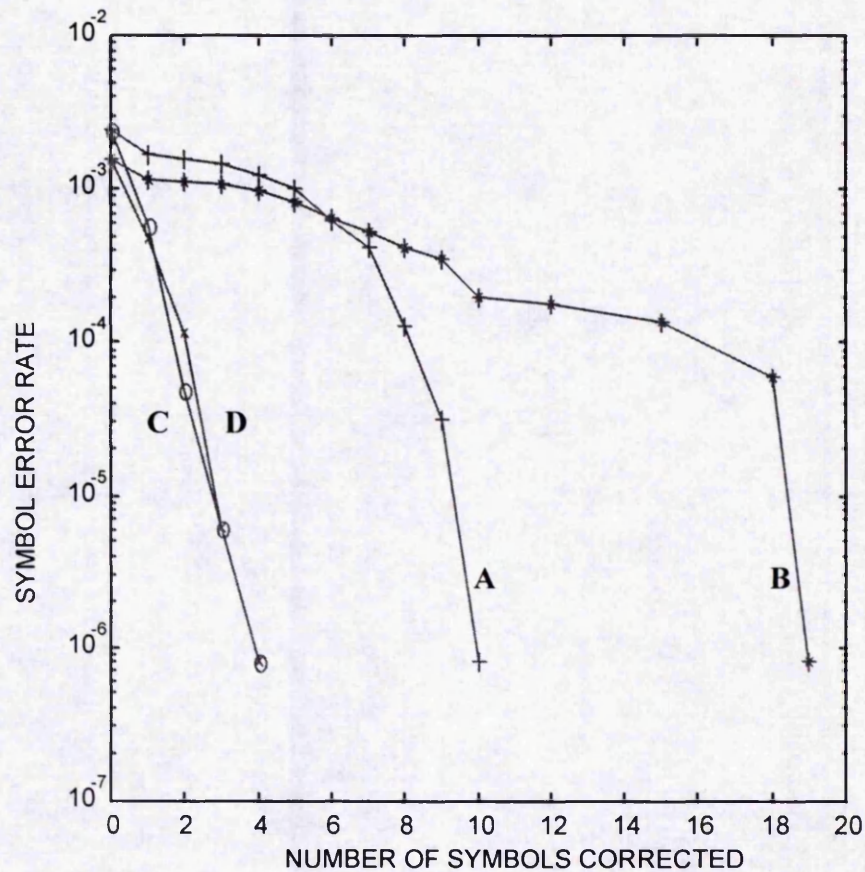


Figure 5.15: Trends of the different strategies at block size 64 and interleaving depth 5000

Figure 5.15 shows the trends of the different strategies at a block size of 64 and an interleaving depth of 5000. It is clear that the both curves **C** and **D** produce the best results i.e., both block and symbol interleaving should be used and the stacking arrangement is not important. Thus it can be said that for any block size at an interleaving depth of 5000, using both block and symbol interleaving is the sensible choice of strategy.

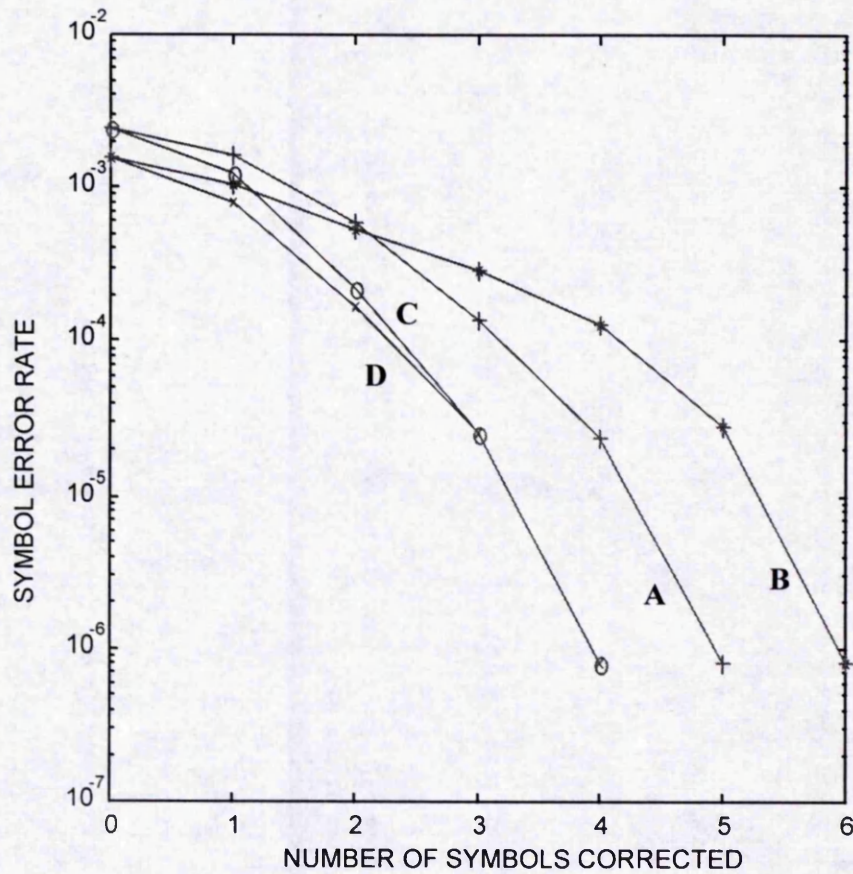


Figure 5.16: Trends of the different strategies at block size 256 and interleaving depth 500

Figure 5.16 shows the trends of the different strategies at a block size of 256 and an interleaving depth of 500. It is clear that the both curves **C** and **D** produce the best results i.e., both block and symbol interleaving should be used and the stacking arrangement is not important.

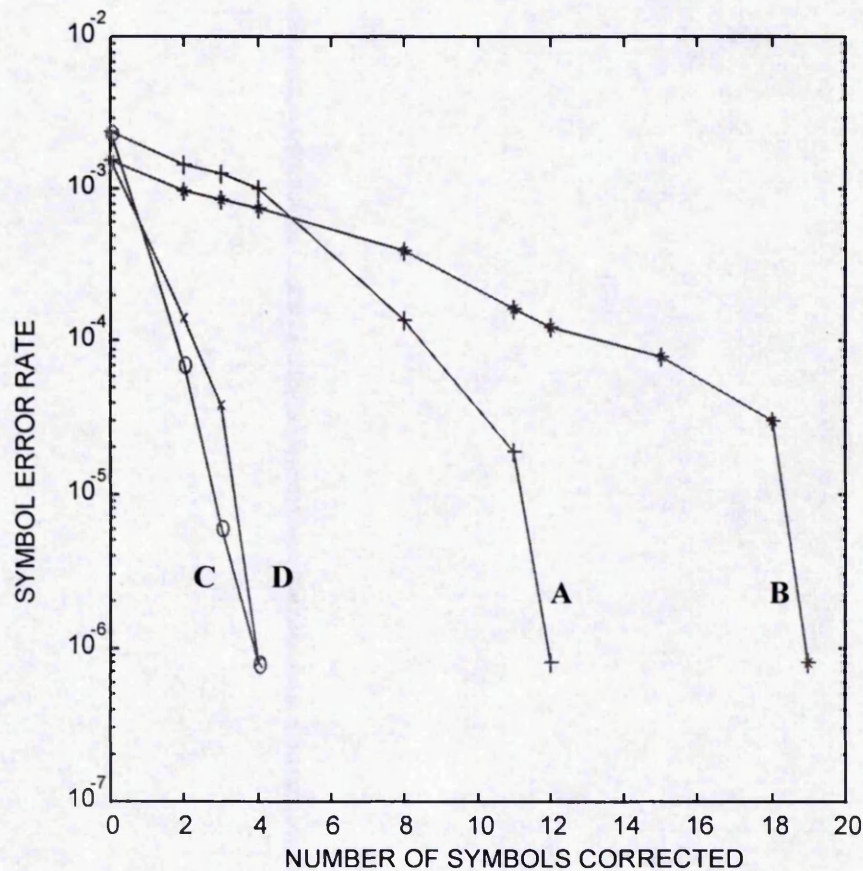


Figure 5.17: Trends of the different strategies at block size 64 and interleaving depth 500

Figure 5.17 shows the trends of the different strategies at a block size of 64 and an interleaving depth of 500. It is clear that the both curves **C** and **D** produce the best results i.e., both block and symbol interleaving should be used and the stacking arrangement is not important. Thus it can be said that for any block size above an interleaving depth of 500, using both block and symbol interleaving is the sensible choice of strategy.

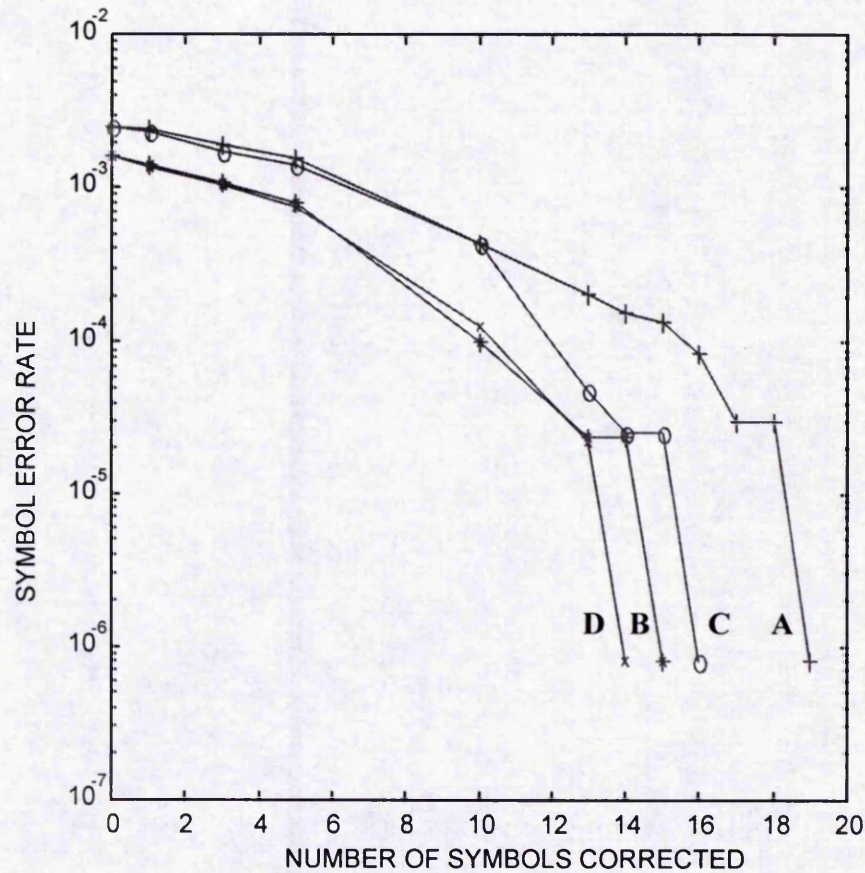


Figure 5.18: Trends of the different strategies at block size 256 and interleaving depth 50

Figure 5.18 shows the trends of the different strategies at a block size of 256 and an interleaving depth of 50. It is clear that curve **D** produces the best results i.e., both block and symbol interleaving should be used with a vertical stacking arrangement.

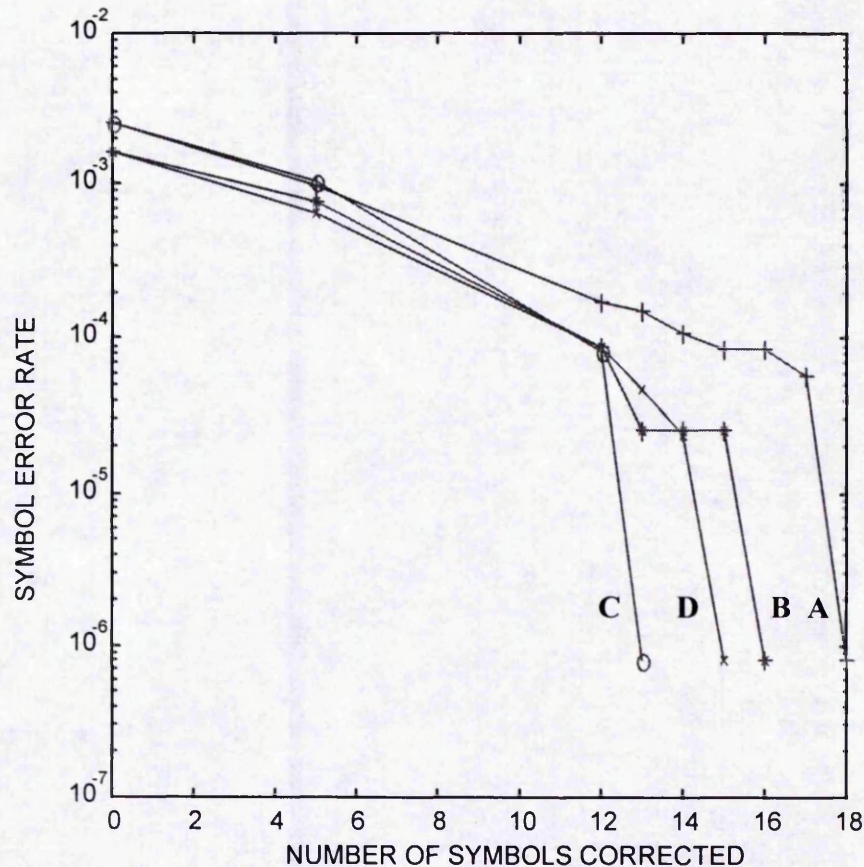


Figure 5.19: Trends of the different strategies at block size 64 and interleaving depth 50

Figure 5.19 shows the trends of the different strategies at a block size of 64 and an interleaving depth of 50. It is clear that curve **C** produces the best results i.e., both block and symbol interleaving should be used with a horizontal stacking arrangement.

In figures 5.18 and 5.19 it can be observed how close the curves are together in terms of the number of symbols corrected. This means that it is easy for different particular strategies in the interleave depth region between 50 and 500 to be the best strategy. For example, for a block size of 192 and interleave depth of 200 symbol interleaving only and a vertical stacking arrangement produces the best result. Therefore it is recommended for the region

between 50 and 500 that the designer obtain manually which strategy is best as a general trend cannot be established.

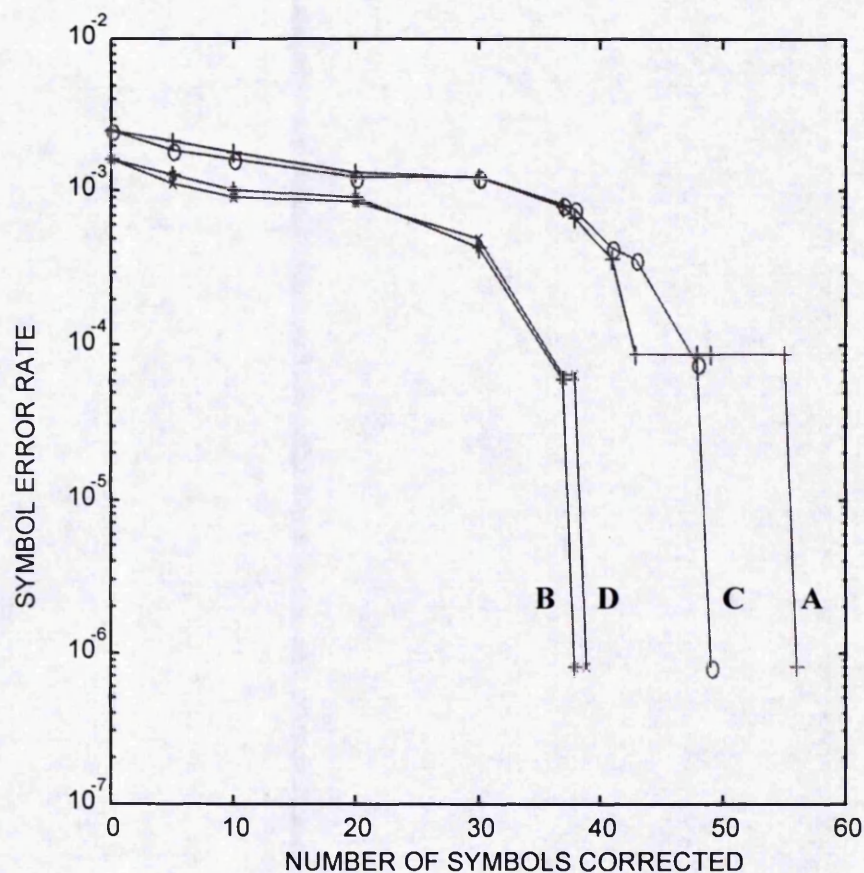


Figure 5.20: Trends of the different strategies at block size 256 and interleaving depth 10

Figure 5.20 shows the trends of the different strategies at a block size of 256 and an interleaving depth of 10. It is clear that curve **B** produces the best results i.e., symbol interleaving only should be used with a vertical stacking arrangement.

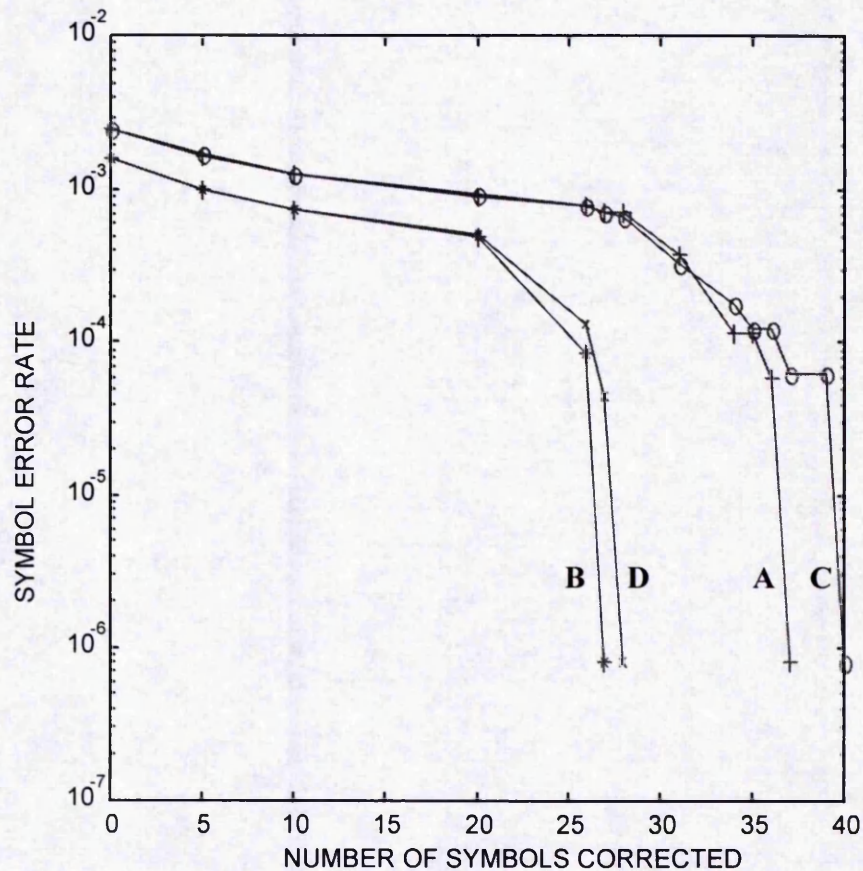


Figure 5.21: Trends of the different strategies at block size 64 and interleaving depth 10

Figure 5.21 shows the trends of the different strategies at a block size of 64 and an interleaving depth of 10. It is clear that curve **B** produces the best results i.e., symbol interleaving only should be used with a vertical stacking arrangement. Thus it can be said that for any block size at interleaving depths up to 50, using symbol interleaving only with a vertical stacking arrangement is the sensible choice of strategy.

There are cases where this strategy is not the best but for the vast majority of cases this strategy is the one to use. In the cases found that did not have this strategy as the best, the best strategy only required one less symbol to be corrected. An example of this is at an

interleave depth of 10 and block size of 128 where the best result is both block and symbol interleaving with a vertical stacking arrangement. This best result strategy requires 31 symbols to be corrected whilst the general case strategy requires 32 symbols to be corrected to remove all the symbol errors from the simulation.

5.4.5 Analysis of Block Size Trends

This section will focus on analysing what the effects of changing the block size has on the best performance required to remove all the error symbols from the system. This analysis is performed at a range of interleaving depths, for the symbol interleaving only and both block and symbol interleaving strategies.

INTERLEAVING DEPTH	BLOCK SIZE			
	256	192	128	64
5000	5	5	7	10
1000	6	6	8	12
500	5	6	7	12
200	8	5	8	12
100	11	8	10	12
50	15	14	15	16
20	25	23	21	25
10	38	43	33	27

Table 8: Analysis of block size trends for the symbol interleaving only strategy

INTERLEAVING DEPTH	BLOCK SIZE			
	256	192	128	64
5000	3	3	3	4
1000	4	4	3	4
500	4	5	4	4
200	8	8	6	6
100	9	9	8	10
50	14	15	15	13
20	29	26	21	28
10	41	43	32	28

Table 9: Analysis of block size trends for both block and symbol interleaving strategy

Block size trends for the symbol interleaving only strategy, and for both block and symbol interleaving strategy, are shown in tables 8 and 9 respectively. The results shown are the best result required to remove all the errors from the system. Thus for an interleave depth of 5000 and a block size of 256, a $t = 3$ EDC symbol code is required to remove all the symbol errors in the system for both block and symbol interleaving, and a $t = 5$ EDC symbol code is required for the symbol interleaving only strategy.

Examining tables 8 and 9, it is clear to see that high interleave depths provide the lowest EDC symbol codes required to remove all the error symbols in the system. This is true for all four block sizes. It can also be seen that at high interleaving depths the best results are produced with high block sizes. For low interleaving depths, the opposite is true with low block sizes producing the best results.

5.4.6 Summary of the trends found

It was found that, for the best possible results, a designer should use large interleaving depths, large block sizes and a strategy of using both block and symbol interleaving. The use of large interleaving depths is the significant factor in reducing the power of the EDC symbol code required to remove all the error symbols in the system. If the designer has a restriction on using large interleaving depths (for example, on the amount of memory available), then the following rules should be used:

- If the designer is able to use an interleave depth above 500 then the strategy to use is large block sizes and both block and symbol interleaving.
- If the designer is required to use an interleave depth of between 50 and 500 then the designer must manually work out which strategy to use.
- If the designer is required to use an interleave depth below 50 then the strategy to use is small block sizes and symbol interleaving only.

5.5 SUMMARY

It has been clearly shown that trends can be produced to show the effects of interleaving on error performance schemes. It was also shown that very small t EDC symbol codes can be used in high interleaving depth scenarios. For example, a strategy of using both block and symbol interleaving, with an interleave depth of 5000 enables the designer to use a $t = 3$

EDC symbol code to remove all the error symbols from the system. This EDC code is relatively efficient (e.g. has a high rate), simple to design and cheap to implement.

The difficulty is the memory required to handle high interleaving depths and the designer may find cases where the interleave depth is restricted. This difficulty, however, is becoming less significant as, in recent years, the price of memory has plummeted. Thus it is recommended that a designer use high interleave depths as this can save costs because the design of the EDC symbol code is simpler and therefore can be implemented in a single VLSI decoder chip.

6 DISCUSSIONS AND CONCLUSIONS

6.1 REVIEW OF THE THESIS

This thesis has examined a novel technique for the modelling of a multi-track longitudinal recording channel using peak detection. This technique has been designed to be flexible enough to include head and tape characteristics, multi-tracks, drop-outs and bit error rates. This technique involves applying additive white gaussian noise samples onto sampled replay and differentiated waveforms, and thus determining the bit error rates produced using 10 million bit data trains. This technique also allows any line code to be analysed and provides additional information which is required to analyse error performance estimation schemes. Three main interleaving strategies have been investigated, these being symbol interleaving, block interleaving and a combination of both. All three strategies were analysed using horizontal and vertical stacks. The results that were produced are of interest to designers of error correction and detection codes that require interleaving.

Chapter two examined the effects of describing pulses and waveforms, in terms of samples, using tape and head characteristics. This allowed any pulse shape to be analysed be it longitudinal, perpendicular, a combination of both orientations of magnetism or alternatively user defined pulses. Pulses were then combined using linear superposition to simulate the behaviour of bit sequences used in a tape system. The longitudinal waveforms produced, with channel parameters that are comparable with modern disk or very modern film tape, were tested at a range of packing densities including 20,000, 100,000 and 160,000 bits per inch. This behaviour was compared with previously published work [Li.

Hui, Hong, J.K., (1993)]. It was shown that a wide range of packing densities were modelled accurately for tape media and thus the use of sampled pulses and waveforms is a valid technique. Data stream recovery was described in detail and illustrated how data bits are accurately regenerated from the sampled differentiated waveform using timing windows. Six channel codes, which are used to match the characteristics of the data with that of the recording channel, were also examined in detail.

Chapter three examined the performance of the recording channel. The limits of the channel's performance were illustrated by the cross-over shift performance of different data patterns and by maximum output voltage performance over a wide packing density range. The data patterns were tested for the advanced longitudinal channel, at packing densities with an upper limit of 160,000 bits per inch. It was found that cross-over shift performance was the significant limiting factor of the channel. The channel was also extended to handle both multi-tracks and large data trains, with the assumption that cross-talk between tracks is negligible. A study was also conducted to demonstrate the power of the modelling tool for performance analysis as individual parameters that define the channel can be investigated in detail. The study demonstrated that it is possible to scale values of various parameters and variables involved in the recording channel without repeating calculations. As work is being done to minimise the value of $(a_t + d)$, this study was conducted in the range of $(a_t + d)$ of 0.176 microns to 0.025 microns. The lower limit of 0.025 microns is considered as the best possible value of $(a_t + d)$. Test were also performed at 1,000,000 bits per inch and it was demonstrated that the simulation is usable at packing densities that exceed 1 million bits per inch and that the simulation can function accurately at the theoretical upper limits of what can be achieved using peak detection channels.

Chapter four examined the incorporation of defects into the recording channel. The two types of defects that were modelled were drop-outs and additive white gaussian noise (AWGN). A full implementation of drop-outs that cause an 80% loss of signal amplitude when compared with the original waveform signal output was conducted, with the statistical analysis of drop-outs evaluated from previously published results. Drop-outs that cause the loss of synchronisation were not considered. AWGN was implemented using two techniques. A new technique [Tandon, A., Middleton, B.K., Farrell, P.G., and Miles, J.J. (1997)] where simulated AWGN waveforms are generated and added to large data streams, and the classical analytical technique [Katz, E. R., and Campbell, T. G., (May 1979)] which required the use of the error function complement, were used to produce bit error rates. Verification of the validity of the two techniques were performed in two stages. The classical analytical technique was implemented and compared with previously published results [Li Hui Hong, J.K. et al. (1993)] to demonstrate the accuracy of this technique. Advanced media, which represent the current state of technology, were then used to compare results from the new technique with those from the classical technique. It was found that there was a good level of agreement of the curves produced using both techniques, and the new technique was able to remove an approximation for the second differential of the signal and so produced more realistic simulations. All the tests were performed in the packing density range of 100,000 to 160,000 bits per inch, and included working in a drop-out and noise infected environment.

A study of the performance limitations due to ISI and noise was conducted and it was demonstrated that optimum record currents in digital recording systems are a reflection of the contributions of noise and peakshift to the error rates occurring in those systems. Again comparisons were made with experimental observations [Middleton B.K., and Brown, T.

(1980); Middleton B.K., and Jack-Kee, T. (1983); Hudson, V. N. et al. (1986)] and it could be seen that higher noise levels and therefore lower signal to noise ratios pushes the optimum current towards higher values and indicates noise limited performance. Narrower timing windows reduce optimum record current to indicate peak shift limited performance.

A significant advantage of the new technique is the ability for drop-out burst error events to be analysed in depth. Two aspects of burst error events that were examined are the internal cross-over shift characteristics that cause single or burst errors to occur and the comparison of data files before and after the modelling process which allow an analysis of burst error statistics. A full set of results were presented and these results were used to produce error performance estimation statistics and information required for error detection and correction work.

Chapter five examined the effect of interleaving strategies on error performance estimation schemes. Three main interleaving strategies have been investigated, these being symbol interleaving, block interleaving and a combination of both. All three strategies were analysed using horizontal and vertical stacks. A full analysis was produced of the blocks, symbols and bits in error and the block, symbol and bit error rates. Different interleaving depths and block sizes were also investigated. It was found that using block interleaving only produced the worst results in all cases and should not be considered as a strategy.

It was also found that, for the best possible results, a designer should use large interleaving depths, large block sizes and a strategy of using both block and symbol interleaving. The use of large interleaving depths is the significant factor in reducing the power of the EDC symbol code required to remove all the error symbols in the system. For example, a

strategy of using both block and symbol interleaving, with an interleave depth of 5000 enables the designer to use a $t = 3$ EDC symbol code to remove all the error symbols from the system. This EDC code is relatively efficient (e.g. has a high rate), simple to design and cheap to implement.

If the designer has a restriction on using large interleaving depths (for example, on the amount of memory available), then the following rules should be used:

- If the designer is able to use an interleave depth above 500 then the strategy to use is large block sizes and both block and symbol interleaving.
- If the designer is required to use an interleave depth of between 50 and 500 then the designer must manually work out which strategy to use.
- If the designer is required to use an interleave depth below 50 then the strategy to use is small block sizes and symbol interleaving only.

6.2 ACHIEVEMENTS OF THE NOVEL SIMULATION TECHNIQUE

The aim of this thesis was to build a comprehensive model of a recording channel that uses peak detection and timing windows. Much innovation was required to ensure the design of a comprehensive recording channel simulation with maximum flexibility and realistic simulation times. These novel techniques, as discussed in section 6.1, are summarised as:

- Use of samples to define pulses and waveforms without any restriction on the shape of the pulse.
- Use of samples in the definition of AWGN rather than statistics.
- The ability to accurately regenerate channel data from differentiated waveforms.
- The ability to handle any line code including ones not currently known about.
- The ability to work with large data streams to produce bit error rates.
- The ability to work with more than one track.
- The ability to work at the predicted upper limits of peak detection channels.
- The ability to work with and investigate the effects of drop-outs.
- The ability to examine the effects of ISI and cross-over on the performance of the channel.
- The ability to fully investigate the behaviour of error performance estimation schemes.
- The ability to be used to fully investigate the behaviour of actual error correction and detection schemes.

6.3 FURTHER WORK

This thesis has presented a comprehensive simulation model of a multi-track recording channel that uses peak detection and timing windows. Also the effects of noise and drop-outs have been considered, to provide the ability to analyse error performance. Further improvements to the simulation, however, could be made and will now be discussed.

6.3.1 Improvements to the Model

For the new AWGN technique, pulse equalization has not been implemented. This was due to lack of time in investigating the properties of pulse equalized noise. If this investigation is performed then pulse equalization can be added as another feature in the new simulation model.

For the new AWGN technique, mixed model analysis was not successfully completed. This was due to an error caused by the fact that the zero crossing point does not occur in the center of the ideal timing window position. The perpendicular components cause a slight shift to the left or right of this ideal timing window position. This slight shift can be calculated and the timing window position corrected but due to the addition of real noise, unexpected effects occur within the timing window as real noise can cause this slight shift to be increased and therefore the timing window gets placed in an incorrect position. The result of this is that the effects of the peakshift are not produced as the timing window is centred on the zero crossing point produced by both the peakshift and the slight shift. So no

error is produced in a case where perhaps peakshift would have caused an error. Thus the resultant BER of the mixed model is unreliable and requires further work.

A way of increasing the usefulness and flexibility of this model is to add the facility to perform signal processing in the model, particularly sequence detection. This was not considered in this thesis as the aim of this thesis was to produce a model capable of analysing error correction and detection codes for multi-track tape systems. PRML is a popular type of sequence detection with disk drive manufacturers and it would be useful to integrate PRML into this model.

6.3.2 Further Studies of Particular Aspects of the Model

For this simulation, the fundamental idea is the use of samples to define AWGN, replayed and differentiated waveforms. Much time was spent optimising the simulation to cut down simulation times from several weeks to 4½ hours. This was achieved by optimising every aspect of the simulation program. Further work can still be attempted to try to reduce this simulation time. Areas which can be looked at include investigating reducing the data train size to one million bits and statistically calculating the bit error rates for ten million bits from this reduced data train. An improvement of a factor of around 10 would occur and reduce simulation times to around half an hour. Another examination of the number of samples used to define a pulse could also be conducted. If, instead of defining the number of samples as a static number, a dynamic definition could be produced then time improvements would occur. If this dynamic definition is used, the effects of noise must be carefully considered.

In this thesis an assumption that the shape of the drop-out was rectangular was made and that the value of $(a_t + d)$ for the drop-out is constant. Examining real tape systems suggests that drop-outs are irregular in shape. Also it is believed that drop-outs have different values of $(a_t + d)$ depending upon the length of the drop-out. If further practical studies of these two factors could be performed then the simulation could be modified to handle these drop-out characteristics.

Further study of timing windows can also be conducted as within timing windows, multiple cross-over points can occur. The effects of multiple cross-overs can be investigated in the hope of increasing the accuracy of the simulation. Also an alternative to ideal zero-crossing points and timing windows can be investigated. This alternative is to replace ideal zero-crossing points with a modelled phase lock loop system that computes the cross-over points. Phase locked loops are commonly used in practise in the hardware design of recording channels.

Another interesting study would be to investigate if there are any particular data patterns which are susceptible to producing random noise errors. If this information is known then perhaps another study into how to avoid susceptible data patterns could be conducted. This could be a very useful way of preventing errors from occurring rather than trying to correct them once they have occurred.

New studies can also be performed. If, in the future, noise with spectra other than white is produced, this new definition can be incorporated into the model. If better tape media are produced in the future, then the parameters of this new media can be used by this simulation, and a full analysis of this media produced.

6.3.3 Error Correction and Detection Studies

Using this model to investigate the behaviour of actual Reed Solomon codes on multi-track tape systems would provide useful knowledge for the code designer about the behaviour of specific Reed Solomon codes. With this knowledge research could be done into how to increase the information that can be held on the recording channel by producing more efficient novel Reed Solomon codes for multi-track tape systems. An interesting area of investigation is the use of soft decision Reed Solomon codes [Sweeney, P., (1991)]. Also the effects of new types of codes could be investigated and compared with the Reed Solomon equivalent. Turbo codes [Berrou, C., et al (1993)] are an example of a new type of code which theoretically produces more efficient results than Reed Solomon equivalents in some scenarios, but which has not been applied in magnetic storage.

The work done using error performance could also be continued with an in depth investigation of what occurs when different depths are used for symbol and block interleaving and then these interleaving techniques are combined together. It is hoped in this way to achieve more efficient error performance in the system.

REFERENCES

Abdel-Ghaffer, K. A. S., and Hassner, M., "*Multilevel Error-Control Codes for Data Storage Channels*", IEEE Trans. Magn., MAG - 37, p735 - 741, May 1991.

Baker, B.R., "*A Dropout Model for a Digital Tape Recorder*", IEEE Trans. Magn., MAG - 13, p1196 - 1198, Sept. 1977.

Berrou, C., et al "*Near Shannon Limit Error-Correcting Coding and Decoding : Turbo Codes*", Proc. ICC, p1064 - 1070, 1993.

Deeley, E.M., "*Integrating and Differentiating Channels in Digital Tape Recording*", Radio and Electronic Engineer, Vol 56, p169, April 1986.

Farrell P.G., "*A Survey of Array Error Control Codes*", European Transactions on Telecommunications, Vol 3, No 5, p441- 454, Sept 1992.

Feibel, W., "*Using ANSI-C in UNIX*", McGraw Hill, p9, 1990.

Green M. B., and Drolet G., "*A Universal Reed-Solomon Decoder Chip*", 16th Biennial Symposium on Communications, p327 - 330, May 1992.

Hasan M. A., and Bhargava V. K., "*A VLSI Architecture for a Low Complexity Rate-Adaptive Reed-Solomon Encoder*", 16th Biennial Symposium on Communications, p331 - 334, May 1992.

Hoagland, A.S., "*Digital Magnetic Recording*", Wiley, 1963.

Hudson, V.N., Loze, M.K., and Middleton, B.K., "*Measurement of Error Rates in a Digital Magnetic Recording System*", IERE Conf. on Video and Data Recording, March 1986.

Hughes, G.F, and Schmidt, R.K., "*On Noise in Digital Recording* ", IEEE Trans. Magn., MAG - 12, p752 - 754, Nov. 1976.

Iwasaki, S. and Ouchi, K., "*Co-Cr Recording Films with Perpendicular Magnetic Anisotropy*", IEEE Trans. Magn., MAG - 14, p849, Sept. 1978.

Jacoby, G.V., "*High Density Recording with Write Current Shaping*", IEEE Trans. Magn., MAG - 15, p1124 - 1130, July 1979.

Kameyama, T. et al., "*Improvement of Recording Density by means of Cosine Equalizers*", IEEE Trans. Magn., MAG - 12, p746 - 748, Nov. 1976.

Katz, E.R., and Campbell, T.G., "*Effect of Bitshift Distribution on Error Rate in Magnetic Recording*", IEEE Trans. Magn., MAG - 15, p1050 - 1053, May 1979.

Kernighan, B.W., and Ritchie, D.M., *"The C Programming Language"*, Prentice-Hall, 1978.

Lathi, B.P., *"Modern Digital and Analog Communication Systems"*, Holt, Rinehart and Winston Inc., 1989.

Li Hui Hong, J.K., Middleton, B.K. and Miles J.J., *"Modern Digital and Analog Communication Systems"*, J. Magn. And Magn. Mat. 120, p206 - 209, 1993.

Lin, S., and Costello, A.J., *"Error Control coding Fundamentals and Applications"*, Prentice Hall, 1983.

Löze, M.K., Middleton, B.K., and Ryley, A., *"Simulation of Digital Magnetic Systems"*, IERE Conf. on Video and Data Recording, No 59, p1, April 1984.

Mackintosh, N.D., *"Superposition Based Analysis of Pulse - Slimming Technique for Digital Recording"*, Radio and Electronic Engineer, Vol 50, No 6, June 1980.

Mallinson, J.C., *"The Next Decade in Magnetic Recording"*, IEEE Trans. Magn., MAG - 21, p1217, May 1985.

Mallinson, J.C., *"Scaling in Magnetic Recording "*, IEEE Trans. Magn., MAG - 32, p599, March 1996.

Mallinson, J.C., and Steele, C. W., "*Theory of Linear Superposition in Tape Recording*", IEEE Trans. Magn., MAG - 5, p886, 1969.

Mathworks, "*The Student Edition of Matlab*", Prentice-Hall, p3, 1992.

Mee C.D., and Daniel E.D., "*Magnetic Recording. Vol I: Technology*", McGraw Hill, 1987.

Middleton, B.K., and Brown, T., "*Recording Tape Properties and Digital Recording System Performance*", Radio and Electronic Engineer, Vol 50, p 467, November 1980.

Middleton, B.K., and Jack-Kee, T., "*Performance of Digital Magnetic Recording Channels Subject to Noise and Drop-outs*", Radio and Electronic Engineer, Vol 53, No 12, p 393-402, November 1983.

Middleton, B.K., Wright, C.D., Cumpson, S.R., and Miles, J.J., "*Output Waveforms in the Replay Process in Digital Magnetic Recording*", IEEE Trans, Magn., Vol 31, No 3, May 1995.

Miles, J.J., and Middleton, B.K., "*Micromagnetic Simulations of Transverse Recording*", Fifth Joint INTERMAG M. M. Conference, Pittsburgh, A.A., June 18th - 21st, 1991.

Obi, J. and Farrell, P.G., "*Combined DC free, runlength limited, and burst error correcting code for a four track magnetic tape*", Bangor Symposium on Communications, May 1993.

Poulsen, V., "*The Telegraphone*", *Annales der Physik*, Vol. 3, p754, Nov. 1900.

Press, W. H., et al, "*Numerical Recipes in C*", Cambridge University Press, p 288 - 290, 1994.

Pritchard, C. J., and Middleton, B. K., "*An Instrument for Detecting and Recording Drop-outs from Magnetic Tape*", *Measurements of Science and Technology*, p 684-685, 1991

Schneider, R.C., "*An Improved Pulse - Slimming for Magnetic Recording*", *IEEE Trans, Magn.*, No 11, p 1240 - 1241, 1975.

Smith, O., "*Some Possible Forms of the Phonograph*", *The Electrical World*, p161, Sept. 1888.

Sweeney, P., "*Error Control Coding - An Introduction* ", Prentice Hall Ltd, 1991.

Tandon, A., Middleton, B.K., and Miles, J.J., "*Identification of Noise and ISI Limited Performance in Digital Magnetic Tape Recording Systems*", *J. Inf. Recording*, Vol. 23, pp347-351, 1996.

Tandon, A., Middleton, B.K., Miles, J.J., and Cumpson, S.R., "*The Application of Scaling to the Prediction of Digital Magnetic Recording Systems*", *J. Phys. D: Appl. Phys.*, Vol. 30, pp542-545, 1997.

Tandon, A., Middleton, B.K., Farrell, P.G., and Miles, J.J., "*A Simulation of a Noisy and Drop-out Infected Multi-track Digital Magnetic Recording Channel for 10 Million bit Data Trains*", J. Inf. Recording, Vol. 23, pp469-487, 1997.

Tjaden, D.L.A., "*Some Notes on Superposition in Digital Magnetic Recording*", IEEE Trans. Magn., MAG - 9, p331, 1973.

Qin, D., and Middleton, B.K., "*An Experimental Study of a Write Equalizer for Unbiased Digital Magnetic Recording*", I.E.E.E Trans. Magn., MAG - 27, No 6, Nov. 1991

Wade G., "*Signal Coding and Processing*", Cambridge University Press, 1994.

Wallace, R.L., "*The Reproduction of Magnetically Recorded Signals*", Bell Syst. Tech. J., Vol 30, p1145, 1951.

Wang, P.S., "*An Introduction to ANSI-C in UNIX*", Wadsworth Inc, p2, 1992.

Watkinson, J., "*Coding for Digital Recording*", Focal Press, 1990.

***SIMULATION OF MULTI-TRACK, DROP-OUT
INFECTED DIGITAL MAGNETIC RECORDING
CHANNELS***

A Thesis Submitted to the University of Manchester
for the Degree of Doctor of Philosophy
in the Faculty of Science and Engineering

Volume II of II

Information Storage Research Group and Communications Research Group,
Division of Electrical Engineering, The Manchester School of Engineering,
Faculty of Science and Engineering



THE UNIVERSITY
of MANCHESTER

BY

AJAY TANDON

1997

K1500561

7h 20459
(DRHXM)

TABLE OF CONTENTS: VOLME II

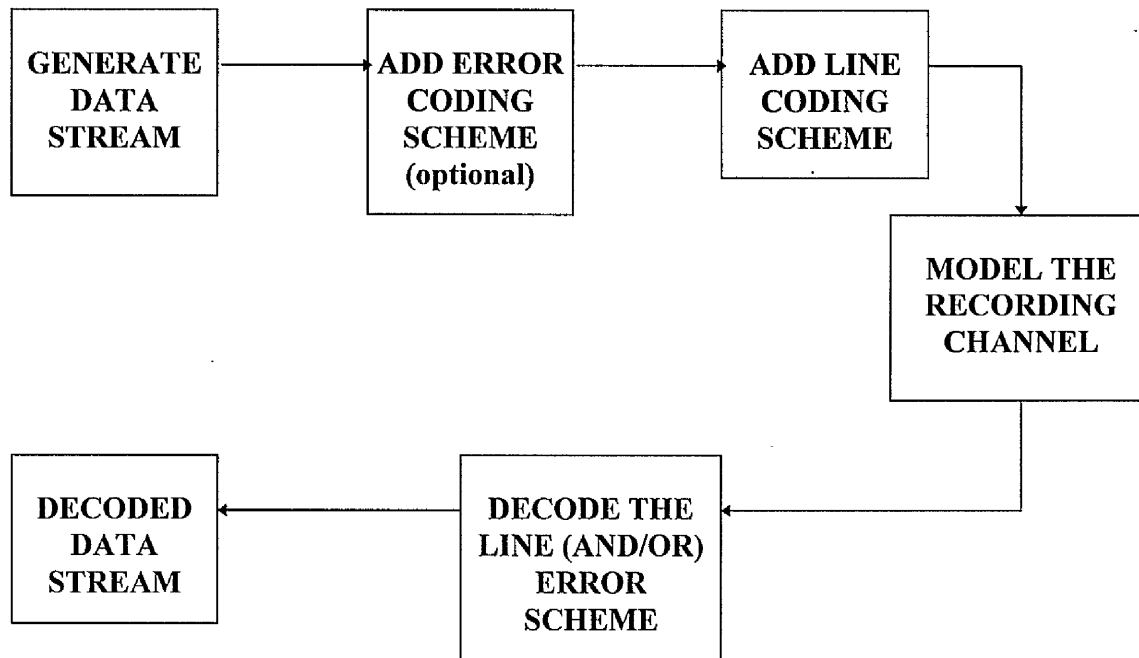
APPENDIX A	174
A.1 Top/down structure analysis	174
A.2 PC requirements and program details	175
A.3 UNIX requirements and program details	181
A.3.1 The programs	181
A.3.2 The input parameter files of interest	182
A.3.2.1 The '.log' input parameter file	182
A.3.2.2 The '.mmi' input parameter file	185
A.3.3 Requirements	189
A.3.4 Structure of the model	190
A.3.5 Compiling the programs	191
A.3.6 Program examples	192
A.3.6.1 Running FAGEN	193
A.3.6.2 Running FACODE	193
A.3.6.3 Running FADROP	194
A.3.6.4 Running FAMAGMOD	194
A.3.6.5 Running FADECODE	196
A.3.6.6 Running FADISP	196
A.3.7 Alternative methods of file analysis	197
A.3.7.1 Using an error coder/decoder instead of a line coder/decoder	198
A.3.7.2 Using a combined error and line decoder	199

A.3.7.3 Replacing programs	200
A.4 Selected program modules	201
A.4.1 Linear superposition module	201
A.4.2 Longitudinal model definition pulse	202
A.4.3 Mixed model definition pulse	205
A.4.4 Pulse equalization module	207
A.4.5 Analytical bit error rate calculation module	209
A.4.6 AWGN generation module	212
A.4.7 New technique bit error rate calculation module	215
A.4.8 Drop-out generation module	218
A.4.9 Block, symbol and bit errors (and error rates) module	219
A.4.10 Vertical stacking arrangement module	224
A.4.11 Symbol interleaving module	225
A.4.12 Block interleaving module	226
A.4.13 Block and symbol interleaving module	228
 APPENDIX B	 231
 B.1 Published papers	 231
 APPENDIX C	 260
 C.1 A detailed investigation of block, symbol and bit errors and error rates for a wide range of interleaving depths and block sizes	 260

C.2 A detailed investigation of the number of symbols to be corrected for a wide range of interleaving depths and block sizes	297
C.3 A detailed investigation of the symbol error rates for a wide range of interleaving depths and block sizes	330

APPENDIX A

A.1 Top/Down Structure Analysis



Above is the breakdown of the structure of the simulation in terms of blocks and is the top level in the top/down design method. Using this structure the initial ideas for the feasibility of the simulation were verified. Each block in the above diagram is represented by a different program (or programs) with files stored on the hard disk used to provide information for the next block. To illustrate this once a data stream is generated, it is saved with the extension .odf. This .odf file is then used as the input for the coding scheme. Thus independent programs can be written and thoroughly tested, before combining the programs to produce the full simulation. The top/down methodology provides a robust and efficient way of minimising the possibility of errors occurring in the programming cycle and was chosen as the methodology for this thesis.

A.2 PC Requirements and Program Details

The minimum requirements for the PC are as follows (realistic requirements in brackets):

CPU: 486 - DX66 (Pentium Processor or better)

RAM: 16 Megabytes (20 MB)

DOS

Windows 3.1 or later

Matlab

The following programs were designed for the PC, and work by using the classical analytical method for generating BER's.

(1) *MAGTAPE.EXE* - This program is used to control all the other programs. It does this by either running *PREPROC.EXE* OR *PROCESS.EXE*, depending if the pre-processor program, or the processor execution programs are required.

(2) *PREPROC.EXE* - This program sets up all the pre-processor functions. It allows the user to generate or modify the input parameters with the extension '.mmi'. Also, the user can create data files, which generate no peakshift, worse case and pseudo random data patterns in a variety of line codes. These codes possess different filename extensions and these are:

'*.odf*' - Original Data File - Contains the uncoded data pattern.

'.lb1' - Line Binary 1 - Contains data patterns coded using the NRZ code.
 '.lb2' - Line Binary 2 - Contains data patterns coded using the NRZI code.
 '.lb3' - Line Binary 3 - Contains data patterns coded using the PHASE code.
 '.lb4' - Line Binary 4 - Contains data patterns coded using the FM code.
 '.lb5' - Line Binary 5 - Contains data patterns coded using the MFM code.
 '.lb6' - Line Binary 6 - Contains data patterns coded using the MILLER code.
 '.ld1' - Line Dropout 1 - Contains NRZ code and drop-outs.
 '.ld2' - Line Dropout 2 - Contains NRZI code and drop-outs.
 '.ld3' - Line Dropout 3 - Contains PHASE code and drop-outs.
 '.ld4' - Line Dropout 4 - Contains FM code and drop-outs.
 '.ld5' - Line Dropout 5 - Contains MFM code and drop-outs.
 '.ld6' - Line Dropout 6 - Contains MILLER code and drop-outs.
 '.db1' - Decode Binary 1 - Contains decode NRZ data patterns.
 '.db2' - Decode Binary 2 - Contains decode NRZI data patterns.
 '.db3' - Decode Binary 3 - Contains decode PHASE data patterns.
 '.db4' - Decode Binary 4 - Contains decode FM data patterns.
 '.db5' - Decode Binary 5 - Contains decode MFM data patterns.
 '.db6' - Decode Binary 6 - Contains decode MILLER data patterns.
 '.rdf' - Received Data File - Contains the decoded uncoded data pattern.

Thus, if a user takes an uncoded data pattern (for example 'test.odf'), and line codes it using NRZ code, the coded data pattern filename retains the original name, but has a different extension. In this case it would be 'test.lb1'. So it is easy to identify what codes have been generated the decoded binary files and the received data files are not used in the pre-processor, but are included her for completeness.

- (1) *PROCESS.EXE* - This program executes a user selected program from a menu structure.
- (2) *LNCODE.EXE* - This program is incorporated into *PREPROC.EXE* but is also defined separately so that a user can line code a file without executing the pre-processor program.
- (3) *LNDECODE.EXE* - This program decodes a decoded binary file and produces the received data file.
- (4) *LNCODDEC.EXE* - This program incorporates both *LNCODE.EXE* and *LNDECODE.EXE* and is used to ensure that data patterns are being coded and decoded accurately.
- (5) *BNWVGEN.EXE* - This program takes the binary data from the line binary extensions, the decoded binary extensions, the original data file, or the received data file and converts the binary numbers into patterns. It is useful for producing displays comparing data within two files. An example of this is comparing error positions in two files.
- (6) *FILECMP.EXE* - This program compares two files for errors and outputs the positions of the errors within the two files.
- (7) *PDAVPEAK.EXE* - This program produces Average Peakshift vs Packing Density waveforms.

(8) *BTCENPK.EXE* - This program produces Peakshift from Center (seconds) vs Packing Density waveforms.

(9) *BTPERP.K.EXE* - This program produces % Peakshift from Center vs Packing Density waveforms.

(10) *PDERROR.EXE* - This program produces Number of Errors vs Packing Density waveforms.

(11) *ISOPULSE.EXE* - This program produces an analysis of isolated pulses. The valid options listed are for the individual pulse. The other options were experiments and proved to be of no use for the main simulation work.

(12) *DROPPOS.EXE* - This program analyses the 20 % value drop-out position (actually the first value under 20 %). This is achieved by altering ($a_i + d$) until the superimposed waveform reaches the 20 % point. It is advisable that the user notes the value from the screen as it is tricky to read it from the graph.

(13) *RPDIFWV.EXE* - This program produces replay and differentiated waveforms for the user to analyse.

(14) *PDMAXREP.EXE* - This program produces Maximum Replay Output Voltage vs Packing Density waveforms.

(15) *PDMAXDIF.EXE* - This program produces Maximum Differentiated Output Voltage vs Packing Density waveforms.

(16) *REPVALUE.EXE* - This program produces a signal voltage value of the replay waveform.

(17) *DIFVALUE.EXE* - This program produces a signal and noise voltage value of the differentiated waveform. From this program the signal and noise powers required for the classical analytical method of analysing noise are obtained.

(18) *BTCODDEC.EXE* - This program decodes the coded bit information from the waveforms themselves and produces files with decoded binary extensions. Thus the behaviour of the replay head reading information from the tape has been simulated.

(19) *BERPKDEN.EXE* - This program produces the Bit Error Rate vs Packing Density waveforms.

(20) *BERTIM.EXE* - This program produces the Bit Error Rate vs Timing Window waveforms.

(21) *BERAD.EXE* - This program produces the Bit Error Rate vs the parameter $(a_t + d)$ waveforms.

(22) *BERMO.EXE* - This program produces the Bit Error Rate vs the parameter M_o waveforms.

(23) *BERD.EXE* - This program produces the Bit Error Rate vs the parameter D waveforms.

(24) *BERMYMX.EXE* - This program produces the Bit Error Rate vs the parameter (M_Y/M_X) waveforms.

(25) *BERV.EXE* - This program produces the Bit Error Rate vs the parameter V waveforms.

(26) *BERG.EXE* - This program produces the Bit Error Rate vs the parameter 2G waveforms.

(27) *BERI.EXE* - This program produces the Bit Error Rate vs the parameter I waveforms.

(28) *BERW.EXE* - This program produces the Bit Error Rate vs the parameter W waveforms.

(29) *BERALPHA.EXE* - This program produces the Bit Error Rate vs the parameter α waveforms.

(30) *BERTHETA.EXE* - This program produces the Bit Error Rate vs the parameter θ waveforms.

A.3 UNIX Requirements and Program Details

The following programs are designed for use with the UNIX HP Workstation. They are designed to be used for the generation and analysis of a drop-out infected, noisy magnetic recording channel, which includes multi-tracks in 10 million bit data trains.

A sub-directory structure is also required with 4 sub-directories: tmp1, tmp2, tmp3, tmp4. Each of these sub-directories should have enough storage space for the files. That is at least 25 MB in each sub-directory, and 10 MB in the home directory (with the home directory above the sub-directories). This requirement for home and sub-directory space will increase if you wish to run the model several times simultaneously, which is the advantage of using the UNIX HP workstation over the PC.

(A.3.1) The Programs

FAGEN.EXE - Used to generate pseudo-random input files that are stored in tmp1.

FACODE.EXE - Used to MFM line code the input files generated by FAGEN.
The input files are in tmp1, and the coded files stored in tmp2.

FADROP.EXE - Used to insert drop-outs into the coded files. The coded files are in tmp2, and the drop-out files are stored in tmp2.

FAMAGMOD.EXE - Used to model the drop-out files and generate '.m' files and

modelled files. The drop-out files are in tmp2, the '.m' files are stored in the root directory. The modelled files are stored in tmp3. This program needs to be run using the network queuing system (NQS). Talk to Tony Arnold at the MCC for information about NQS if you do not know anything about it.

FADECODE.EXE - Used to decode the modelled files and strip out the MFM code. The modelled files are stored in tmp3, and the stripped files are stored in tmp4.

FADISP.EXE - Used to analyse the stripped files (stored in tmp4) and produce useful analysis of the error event bursts that occur.

(A.3.2) The Input Parameter Files of Interest

There are two types of files that are used for holding input parameters, and are recognised by their separate extensions. These extensions are '.log' and '.mmi'.

(A.3.2.1) The '.log' Input Parameter File

The '.log' file is used for processing the following programs: FAPRE, FAGEN, FACODE, FADROP, FADECODE and FADISP. The '.log' file contains all the information required by these programs to run successfully.

An example '.log' file (called initial.log) is shown below.

```
TRACKS = 8  
DATA_SIZE = 5000000  
DATA_NAME = zzzz  
UNIX = y  
DECODE_NAME = dmfm  
DISP_NAME = rdf  
EMPTY_COLS = 5  
SAVE_BURST_INFO = burst  
SAVE_MORE_BURST_INFO = burst2  
SAVE_DROPOUT_INFO = dropout
```

Each parameter will now be described.

TRACKS - Used to specify the number of tracks being simulated

DATA_SIZE - Used to specify the total data size of all the tracks before being coded.
Therefore for 10 million coded bits to be produced specify 5000000. Note the program will generate the value above 5000000 which is exactly divisible by 512.

DATA_NAME - Used to specify the name of the data files used in the model. This must be 4 characters long.

UNIX - Used to specify if you are using UNIX or not. Always use ' y '. This is because I designed the modelling programs in ANSI-C and so they can be used on a PC or a UNIX Workstation. Professor Farrell prefers the Workstation as you can process several models simultaneously.

DECODE_NAME - Used to specify the extension of the decoder. There are two choices: mfm - decode MFM files (which do not contain drop-outs), or dmfm - which decodes MFM files which contain drop-outs.

DISP_NAME - Used to specify the extension of interest for the program FADISP. There are three choices: mfm - analyse MFM files (which do not contain drop-outs), dmfm - which analyses mfm files that contain drop-outs, or rdf - which analyses files that have the mfm code stripped out of them by FADECODE.

EMPTY_COLS - Used to maximum specify the number of empty columns that can be contained within a burst error event.

SAVE_BURST_INFO - Used to specify the filename (without extension) that saves burst information. The extension '.txt' is added automatically.

SAVE_MORE_BURST_INFO - Used to specify the filename (without extension) that saves additional burst information. The extension '.txt' is added automatically.

SAVE_DROPOUT_INFO - Used to specify the filename (without extension) that saves drop-out information. The extension '.txt' is added automatically. The drop-out information

tells you the positions where the dropouts were originally put in by FADROP. Remember when considering files that have the MFM code stripped out of them the positions are halved, as drop-outs are currently only added to line coded files.

(A.3.2.2) The '.mmi' Input Parameter File

The '.mmi' file is used for processing the FAMAGMOD program. The '.mmi' file contains all the information required by this program to run successfully.

An example '.mmi' file (called curve7.mmi) is shown below.

Rem

Rem created: 24 Mar 1996

Rem

Model_Code = GENERAL CASE MODEL

A = 8.400000e-08

D = 2.500000e-08

M = 3.857827e+08

G = 1.000000e-10

V = 3.810000e-01

I = 1.800000e+01

O = 4.000000e-07

W = 1.814286e-03

ALPHA = 0.000000e+00

THETA = 0.000000e+00
GAIN = 2.000000e-03
TRACKS = 8
ADD_NOISE = Y
NOISE_VALUE = 1.716798e+03
SLIM_PULSE = N
DROPOUTS = Y
DROPOUT_VAL = 1.640000e-07
CODE_NAME = MFM
DATA_NAME = ZZZZ
PACKING_DENSITY = 320000
STEP_ACCU = 3500
TAIL_LENGTH = 20
UNIX = N
SEED = 13
END_EFFECT_BITS = 32
TIMING_WINDOW_SIZE = 44

Each parameter will now be described.

Rem - Allows you to add remarks, such as the date the file was created. This does not work perfectly, so please do not put any of the parameter names below after the Rem statement.

Model_Code - This specifies the model code name. Always use GENERAL CASE MODEL. Other models were devised but had clocking problems and so are not used. Talk to Professor B. K. Middleton for more information.

A - This is $(a_t + d)$, with a_t being the recorded transition width at the top of the medium, and d being the (head / medium) separation on replay.

D - This parameter is the lesser of medium thickness and depth of recording.

M - This is parameter M_o , with M_o being the peak magnetisation of the medium.

G - This is (gap length / 2).

V - This is the field velocity.

I - This is $(n\eta)$, with the parameter n being the number of turns and η being the head efficiency.

O - This is $(\mu_0 / \pi I)$, with μ_0 being the permeability of free space.

W - This is the track width.

ALPHA - This is the rate of change of transition width with depth into the medium.

THETA - This is the angle of the easy axis.

GAIN - This is gain of the differentiator in the magnetic model.

TRACKS - This is the number of tracks being analysed

ADD_NOISE - This informs the model that we want to add noise.

NOISE_VALUE - This is the actual noise value used to generate additive white gaussian noise.

SLIM_PULSE - This informs the model that we want to use pulse equalization.

DROPOUTS - This informs the model that we want to add drop-outs.

DROPOUT_VAL - This is the new $(a_i + d)$ value used to create drop-outs.

CODE_NAME - This tells the model the name of the line code used. For the MFM line code the model then looks for the input file with the extension '.lb5' (if drop-outs are not being analysed), or '.ld5' if the input file contains drop-outs.

DATA_NAME - This tells the model the 4 character name of the input files.

PACKING_DENSITY - This tells the model the packing density which is to be used for analysis. Remember that this is the line coded packing density. For example if a value of

320000 is used, the line coded packing density is 320000, but the real packing density is 160000.

STEP_ACCU - This tells the model the number of samples that are used to define an isolated pulse, before superposition.

TAIL_LENGTH - This tells the model the length of the tail of the pulse.

UNIX - This tells the model if the UNIX environment is being used. Always use 'y'.

SEED - This is the seed used for the random number generators.

END_EFFECT_BITS - This tells the model the number of end-effect bits that are to be used.

TIMING_WINDOW_SIZE - This tells the model the size of the timing window.

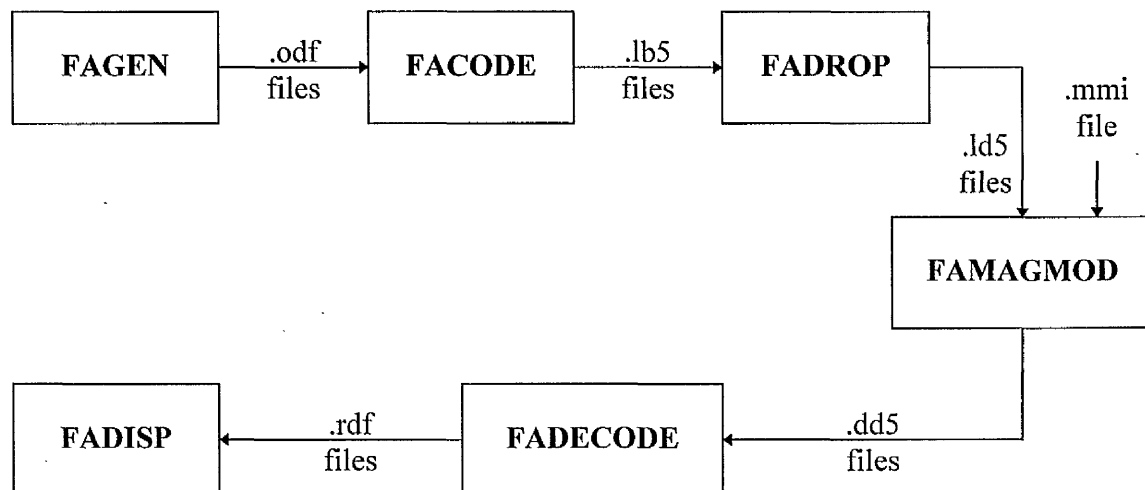
(A.3.3) Requirements

You must be working on a Unix Workstation.

You must have a home directory and four sub-directories tmp1, tmp2, tmp3 and tmp4.

You must have MATLAB available to display the '.m' files that are generated by FAMAGMOD.

(A.3.4) Structure of the Model



Above is a flow-diagram representation of the interaction of the programs. The extensions of the files are shown in the diagram and are used for inputs and outputs for the various programs. These files share common filenames, with the extensions used to identify what the file contains.

The common filename is in the following format, four characters followed by four numbers, for example, zzzz0000. The four characters are contained in the parameter DATA_NAME which is in both the .log file and the .mmi file. The four digits represent the track number of the file. Therefore zzzz0000 is the first track, zzzz0001 is the second track, zzzz0002 is the third track and so on. Thus the maximum number of tracks that can theoretically be simulated is 10000. To tell the model how many tracks we require we use the TRACKS parameter contained in both the .log file and the .mmi file.

The meaning of the extensions are as follows:

.odf - The original data file generated by FAGEN, and contains pseudo-random data.

.lb5 - The line binary (coded) 5 file generated by FACODE, and contains MFM coded files.

.ld5 - The line drop-out 5 file generated by FADROP, and contains MFM coded files that include drop-outs.

.dd5 - The decoded drop-out 5 file generated by FAMAGMOD, and contains the modelled files generated using .ld5 input files.

.db5 - The decoded binary 5 file generated by FAMAGMOD, and contains the modelled files generated using .lb5 input files.

.rdf - the received data file generated by FADECODE, and contains the decoded files with the MFM code stripped out.

(A.3.5) Compiling the Programs

For all the programs, in each of the source code files contains a line similar to this:

```
cc fadisp.c -o fadisp +Oall -lmalloc -lm -Aa
```

Use the equivalent line in each of the source code files to compile the source code. This is compiled using the cc compiler. The meaning of the extensions is as follows.

-o This is the output file name option, with the name of the executable file placed directly after this option (but separated by a space).

+Oall This is the extensive compilation option, so that the program runs at maximum optimisation but it takes some time to compile.

-lmalloc Link the malloc library in the compilation process.

-lm Link the maths library in the compilation process.

-Aa Compile using strict ANSI-C.

(A.3.6) PROGRAM EXAMPLES

The following section contains an example for all the programs and you can repeat it to see how the programs work. I assume you have compiled all the source code into executables. Execute FAGEN, then FACODE, FADROP, FAMAGMOD, FADECODE and finally FADISP.

The names of the input files are initial.log and curve7.mmi.

(A.3.6.1) Running FAGEN

Type the following in:

```
fagen initial
```

A display will appear which is used to confirm that the process is working.

In tmp1 there should be 8 files zzzz0000.odf - zzzz0007.odf, when the program has finished running.

(A.3.6.2) Running FACODE

Type the following in:

```
facode initial
```

A display will appear which is used to confirm that the process is working.

In tmp2 there should be 8 files zzzz0000.lb5 - zzzz0007.lb5, when the program has finished running.

(A.3.6.3) Running FADROP

Type the following in:

```
fadrop initial
```

A display will appear which is used to confirm that the process is working.

In tmp2 there should be 8 files zzzz0000.db5 - zzzz0007.db5, when the program has finished running. In the home directory, a file called dropout.txt is also created which tells you at what positions the dropouts were added into by FADISP.

(A.3.6.4) Running FAMAGMOD

FAMAGMOD is more complex to execute than any of the other programs as it needs to use the network queuing system (NQS). Talk to Tony Arnold at the MCC for information about NQS if you do not know anything about it.

If your workstation is not NQS, you have to remotely login into a workstation that is NQS.

Do this by typing:

```
rlogin meehpf -l mbhptaa
```

but replace meehpf with the workstation name you want to access, and mbhptaa with your user name. Then login to the remote station.

Set up a text file (called magtape for example) which contains the following line only:

```
famagmod curve7
```

Then to run the NQS job type:

```
qsub -q meehpl magtape
```

The NQS job will now be automatically set up by NQS. You should be running in the plong option. If not, talk to Tony Arnold at the MCC.

To check the NQS job, on a NQS workstation, type:

```
qjob - a
```

The process takes about 4½ hours to run. In tmp3 there should be 8 files zzzz0000.ld5 - zzzz0007.ld5, when the program has finished running. Two files with the filename magtape, but with different extensions will also be created by NQS system itself. These files contain the normal output and error output of the famagmod program.

For example:

magtape.o207 - This contains the normal output of magtape, with NQS
 assigning 207 as the process number automatically.

magtape.e207 - This contains the error output of magtape, with NQS
assigning 207 as the process number automatically.

Note: 207 is an example process number, NQS will assign the process number as a value of
its choice, you have no control over this number.

(A.3.6.5) Running FADECODE

Type the following in:

```
fadecode initial
```

A display will appear which is used to confirm that the process is working.

In tmp4 there should be 8 files zzzz0000.rdf - zzzz0007.rdf, when the program has finished
running.

(A.3.6.6) Running FADISP

Type the following in:

```
fadisp initial
```

A display will appear which is used to confirm that the process is working. A set of
statistical analysis of bursts will also be produced and displayed on screen. In the home

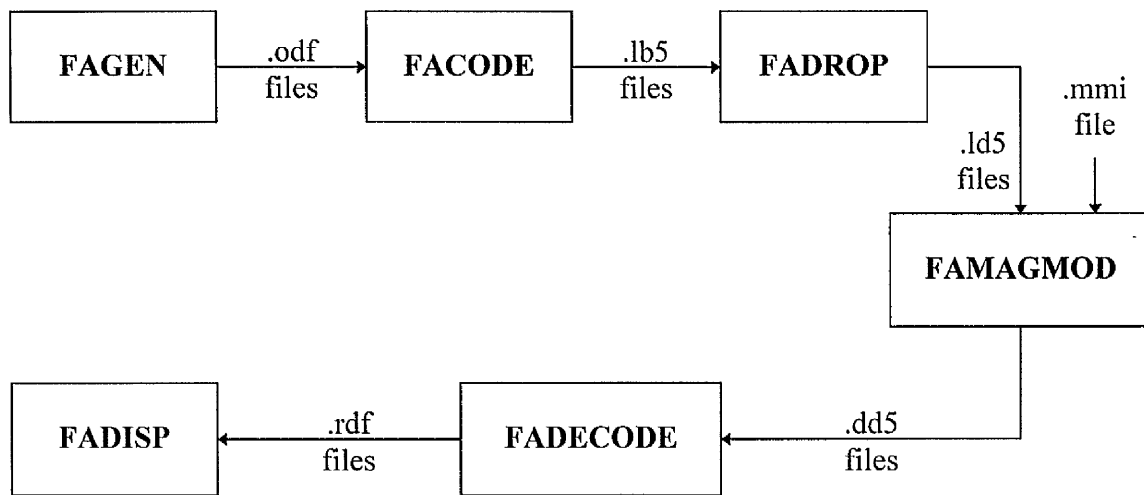
directory, two files called burst.txt and burst2.txt are also created which provides displays and detailed information about the bursts. Full details of these output files are described in section 4.

FADISP has the ability to compare either .rdf and .odf files, .ld5 and .dd5 files, or else .lb5 and .db5 files by specifying the DISP_NAME parameter (in a .log file) with rdf, dmfm, or mfm respectively.

(A.3.7) ALTERNATIVE METHODS OF FILE ANALYSIS

There are possible alternative ways of using these programs if other situations are to be examined. This section will discuss these alternative situations. I know that examination of structures where an error coder is used to replace the line coder can occur in the future, and the situation that Julie Obi did in her PhD, where the line decoder and error decoder are combined. By discussing these situations I hope useful information is provided to you in your work.

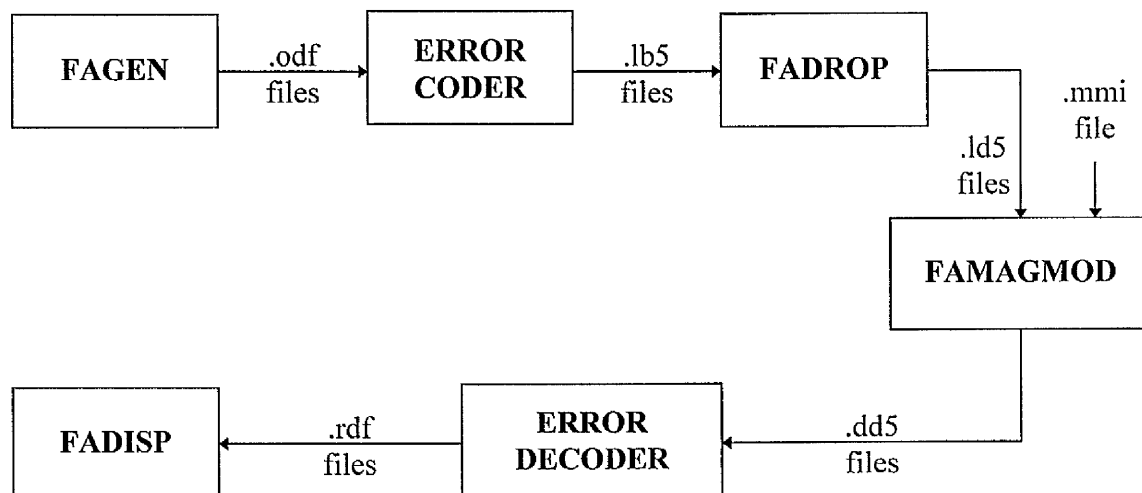
You should realise that the program FAMAGMOD, which models the tape system should not be touched unless you have had discussions with both Professor P. G. Farrell and Professor B. K. Middleton. It is an incredibly sensitive program with situations where things can go drastically wrong. If you want to fiddle with this program read my PhD thesis and end of first year report (available through Professor B. K. Middleton) for important information about my work. We will review the current modelling situation again:



(A.3.7.1) Using an Error Coder/Decoder instead of the Line Coder/Decoder

Remember that to each program the input and output are just 1's and 0's in a file with the extension identifying what type of data is in the file. The exception to this is FADROP as drop-out bits are represented by an 8 (= drop-out 0) or a 9 (= drop-out 1).

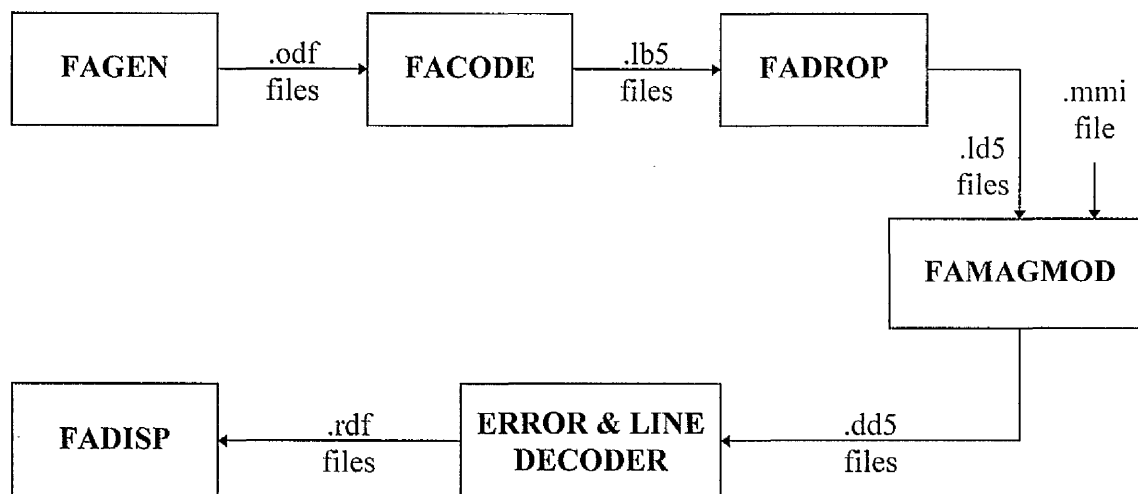
So all you have to do is to use the extensions to process the binary data files. So to add an error coder use the following structure:



Yes it looks the same, but you write an error coder and error decoder (to replace FACODE and FADECODE respectively). As long as you read in the input files (.odf for the coder and .dd5 for the decoder) and save the data as the output files (.lb5 for the coder and .rdf for the decoder), the modelling process will work. Remember the filename structure (e.g. read in zzzz0000.odf - zzzz0007.odf for the coder and save the files generated by the coder as zzzz0000.lb5 - zzzz0007.lb5). Follow a similar idea for the decoder you write.

(A.3.7.2) Using a Combined Error and Line Decoder

This is a simpler idea than section (A.3.7.1), see the diagram below:



This is less complex than section (A.3.7.1) as we only write a new decoder (to replace FADECODE). You read in the input files .dd5 and save the data as the output files .rdf. Remember the filename structure (e.g. read in the files zzzz0000.dd5 - zzzz0007.dd5 for the decoder and save the files generated as zzzz0000.rdf - zzzz0007.rdf).

(A.3.7.3) Replacing Programs

You can replace all the programs you want but apart from the above two situations, I see you only replacing FADISP, by a display analysis program for you error decoder.

It is important to realise that you need pseudo-random files to go into your coder as it is only pseudo-random files that produce 1 error in 10^5 statistics that you require. **Beware** of being asked to use files with all 0's as the input files to the line coder as this produces a no peak-shift pattern when MFM coded and you get no bit error statistics due to noise.

A.4 Selected Program Modules

(A.4.1) Linear Superposition Module

```

/*****

The function 'superimpose_1' does the superposition of data to produce the
differentiated waveform.

*****/

void superimpose_1 (int start_pos_1, int begin_pos_1, int size_of_disp_1, float *Pos_diff_1,
                    float *Neg_diff_1, char *test_1, float *Diff_array_1)
{
    int counter, addr, temp;

    counter = begin_pos_1 + 1;
    while (*(test_1 + counter) != '2')
    {
        addr = start_pos_1 + (size_of_disp_1 * (counter - 1));
        temp = 0;
        if ((test_1[counter - 1] == '0') && (test_1[counter] == '1'))
            while (*(Pos_diff_1 + temp) != 1000.0)
                *(Diff_array_1 + addr++) += *(Pos_diff_1 + temp++);
        else
            if ((test_1[counter - 1] == '1') && (test_1[counter] == '0'))
                while (*(Neg_diff_1 + temp) != 1000.0)
                    *(Diff_array_1 + addr++) += *(Neg_diff_1 + temp++);
    }
}

```



```

        counter++;
    }
}

```

The above module is the code used to produce linear superposition waveforms from isolated replay or differentiated waveforms. This is the code that is central to the whole new technique that has been described in this thesis. It has been optimised for speed and in many of the programs devised it is integrated into the main code instead of being a separate module to optimise the speed of the programs.

(A.4.2) Longitudinal Model Definition Module

```

/*****

The function 'gen_case_model' calculates the sample values used to model the
General Case waveform.

*****/

void gen_case_model (float *Pos_diff_1, float *Neg_diff_1, struct Gen_case_type *gen_1,
                    double begin_point_1, double step_size_1, int step_accu_1)
{
    double x, temp1, temp2, temp3, temp4, temp5, temp6, temp7, temp8,
           temp9, temp10, temp11, temp12, temp13, temp14, temp15,
           temp16, tempU, tempV, tempW, tempX, tempY, tempZ, C1, C2,
           C3, C4, C5, C6, C7, C8, C9, C10;

```

```

int    count;

C1 = (gen_1->O * gen_1->V * gen_1->W * gen_1->M *
      (cos (gen_1->THETA)) * ((gen_1->I) / (2 * (gen_1->G)))) /
      (1 + gen_1->ALPHA);

C2 = 2 * (gen_1->A + (gen_1->D * (1 + gen_1->ALPHA)));

C3 = gen_1->G / (gen_1->A + (gen_1->D * (1 + gen_1->ALPHA)));

C4 = 1 / (gen_1->A + (gen_1->D * (1 + gen_1->ALPHA)));

C5 = 2 * gen_1->A;

C6 = gen_1->G / gen_1->A;

C7 = 1 / gen_1->A;

C8 = gen_1->G;

C9 = (gen_1->A + (gen_1->D * (1 + gen_1->ALPHA))) *
      (gen_1->A + (gen_1->D * (1 + gen_1->ALPHA)));

C10 = gen_1->A * gen_1->A;

```

```

for (count = 0; count < step_accu_1; count++)
{
    x = (count * step_size_1) + (begin_point_1);

    temp1 = C3 + (C4 * x);
    temp2 = temp1 * temp1;
    temp3 = C3 - (C4 * x);
    temp4 = temp3 * temp3;
    temp5 = C6 + (C7 * x);
    temp6 = temp5 * temp5;
    temp7 = C6 - (C7 * x);
    temp8 = temp7 * temp7;
    temp9 = C8 + x;

```

```

temp10 = temp9 * temp9;

temp11 = C9 + temp10;

temp12 = C10 + temp10;

temp13 = C8 - x;

temp14 = temp13 * temp13;

temp15 = C9 + temp14;

temp16 = C10 + temp14;


tempU = C2 * C4 * ((1 / (1 + temp2)) - (1 / (1 + temp4)));

tempV = C5 * C7 * ((1 / (1 + temp6)) - (1 / (1 + temp8)));

tempW = 2 * ((temp10 / temp11) - (temp10 / temp12));

tempX = log (temp11) - log (temp12);

tempY = 2 * ((temp14 / temp15) - (temp14 / temp16));

tempZ = log (temp15) - log (temp16);


*(Pos_diff_1 + count) = (float) ((gen_1->GAIN * C1 * (tempU -
                                tempV + tempW + tempX - tempY - tempZ));

*(Neg_diff_1 + count) = (float) (-1 * *(Pos_diff_1 + count));

}

*(Pos_diff_1 + step_accu_1) = (float) 1000.0;

*(Neg_diff_1 + step_accu_1) = (float) 1000.0;

}

```

The above module is the code used to define the longitudinal model and is used to generate the isolated longitudinal replay and differentiated pulses. This code converts the mathematical definition of the model into a sampled representation of the model which can then be used for the linear superposition process and analysis using the new technique defined in this thesis.

(A.4.3) Mixed Model Definition Module

```

/*****

The function 'mixed_model' calculates the sample values used to model the
Mixed Model waveform.

*****/

void mixed_model (float *Pos_rep_1, float *Neg_rep_1, float *Pos_diff_1,
                  float *Neg_diff_1, struct Mixed_model_type *mix_1,
                  double begin_point_1, double step_size_1, int step_accu_1)
{
    double x, temp1, temp2, temp3, tempP, tempQ, tempR, tempU, tempV,
           tempW, C1, C2, C3, C4, C5, C6, m, c;

    int    count;

    C1 = (mix_1->O * mix_1->V * mix_1->W * mix_1->M *
          (cos (mix_1->THETA)) * mix_1->I) / (1 + mix_1->ALPHA);
    C2 = (mix_1->A + (mix_1->D * (1 + mix_1->ALPHA))) *
          (mix_1->A + (mix_1->D * (1 + mix_1->ALPHA)));
    C3 = mix_1->A * mix_1->A;
    C4 = (2 * mix_1->O * mix_1->V * mix_1->W * mix_1->M *
          (cos (mix_1->THETA)) * mix_1->I *
          ((sin (mix_1->THETA)) / (cos (mix_1->THETA)))) / (1 + mix_1->ALPHA);
    C5 = mix_1->A + (mix_1->D * (1 + mix_1->ALPHA));
    C6 = mix_1->A;

```

```

for (count = 0; count < step_accu_1; count++)
{
    x = (count * step_size_1) + (begin_point_1);

    temp1 = x * x;
    temp2 = C2 + temp1;
    temp3 = C3 + temp1;

    tempP = C1 * log (temp2 / temp3);
    tempQ = atan (x / C5);
    tempR = atan (x / C6);

    *(Pos_rep_1 + count) = (float) (tempP - (C4 * (tempQ - tempR)));
    *(Neg_rep_1 + count) = (float) (-1 * *(Pos_rep_1 + count));

    tempU = C1 * ((2 * x * (C3 - C2)) / (temp2 * temp3));
    tempV = C5 / ((C5 * C5) + temp1);
    tempW = C6 / ((C6 * C6) + temp1);

    *(Pos_diff_1 + count) = (float) (mix_1->GAIN * (tempU - (C4 * (tempV - tempW))));
    *(Neg_diff_1 + count) = (float) (-1 * *(Pos_diff_1 + count));
}

m = (*(Pos_rep_1 + step_accu_1 - 1) - *(Pos_rep_1)) / (step_accu_1 - 1);
c = *(Pos_rep_1);

for (count = 0; count < step_accu_1; count++)
{
    *(Pos_rep_1 + count) -= (float) ((m * count) + c);
    *(Neg_rep_1 + count) += (float) ((m * count) + c);
}

```

```

        *(Pos_diff_1 + count) -= (float) m;

        *(Neg_diff_1 + count) += (float) m;

    }

    *(Pos_rep_1 + step_accu_1) = (float) 1000.0;

    *(Neg_rep_1 + step_accu_1) = (float) 1000.0;

    *(Pos_diff_1 + step_accu_1) = (float) 1000.0;

    *(Neg_diff_1 + step_accu_1) = (float) 1000.0;

}

```

The above module is the code used to define the mixed model and is used to generate the isolated mixed model replay and differentiated pulses. This code converts the mathematical definition of the model into a sampled representation of the model which can then be used for the linear superposition process and analysis using the new technique defined in this thesis.

(A.4.4) Pulse Equalisation Module

```

/*****

The function 'pulse_slim' handles the pulse slimming effects of the model.

*****/

void pulse_slim (float *Pos_diff_1, float *Neg_diff_1, int scale_1, int sample_1, int step_accu_1)
{

    float *Diff_end_pos, *Diff_end_neg;

```



```

int    count, count2;

Diff_end_pos = (float *) malloc ((step_accu_1 + 1) * sizeof (float));
if (! Diff_end_pos)
{
    printf ("\nDIFF_END_POS memory allocation has failed\n");
    exit (1);
}

Diff_end_neg = (float *) malloc ((step_accu_1 + 1) * sizeof (float));
if (! Diff_end_neg)
{
    printf ("\nDIFF_END_NEG memory allocation has failed\n");
    exit (1);
}

for (count = 0; count < step_accu_1; count++)
{
    *(Diff_end_pos + count) = *(Pos_diff_1 + count) / scale_1;
    *(Diff_end_neg + count) = *(Neg_diff_1 + count) / scale_1;
}

*(Diff_end_pos + step_accu_1) = '\0';
*(Diff_end_neg + step_accu_1) = '\0';

count2 = 0;
for (count = sample_1; count < step_accu_1; count++)
{

    *(Pos_diff_1 + step_accu_1 - count) += *(Diff_end_neg + step_accu_1 - count2);

```

```

        *(Pos_diff_1 + count) += *(Diff_end_neg + count2);

        *(Neg_diff_1 + step_accu_1 - count) += *(Diff_end_pos + step_accu_1 - count2);
        *(Neg_diff_1 + count) += *(Diff_end_pos + count2++);

    }

    free (Diff_end_pos);
    free (Diff_end_neg);
}

```

The above module is the code used to equalise the pulse. The slimming process occurs by defining new pulses which are of a negative sense and smaller amplitude to the original pulse and adding them to the original pulse.

(A.4.5) Analytical Bit Error Rate Calculation Module

```

loop3 = 0;

for (loop = temp_count; loop < range; loop++)
{
    test = 0;

    start_pos2 = start_pos + mix_disp + (size_of_disp * loop);

    temp_tim = ((double) size_of_disp / timing_factor);
    timing_beg = (int) temp_tim;

    window_beg = (start_pos2 + (step_accu / 2)) - timing_beg;
    timing_window = window_beg + (2 * timing_beg);
}

```

```

for (loop2 = window_beg; loop2 < timing_window; loop2++)
{
    if (((*(Out_array + loop2) > 0) && (*(Out_array + loop2 + 1) < 0)) ||
        (*(Out_array + loop2) < 0) && (*(Out_array + loop2 + 1) > 0)))
    {
        value1 = *(Out_array + loop2);
        pos1 = loop2;
        value2 = *(Out_array + loop2 + 1);
        test = 1;
    }
}

if (!test)
    *(Result_array + loop3++) = 0.0;
else
{
    diff = (fabs (value1) / (fabs (value1) + fabs (value2)));

    *(Result_array + loop3++) = (((((double) (start_pos2 + (step_accu / 2)
        - pos1)) - diff) * step_size) / velocity);
}

}

*(Result_array + ++loop3) = 1000;

timing_window2 = (bit_spacing / (timing_factor * velocity));
noise_timing = ((noise_value * bit_spacing) / (PI * signal_value * velocity));
if ((slim == 'Y') || (slim == 'y'))
{

```

```

        noise_timing *= ((2 / (scale * scale)) + 1);

        printf ("\nWorking on a slimmed pulse\n");

    }

    *(Error_rate + err_cnt) = 0.0;

    loop = 0;

    do

    {

        if (*(Result_array + loop) != 0.0)

        {

            x_minus = ((timing_window2 - Result_array [loop]) / (sqrt (2) * noise_timing));

            x_plus = ((timing_window2 + Result_array [loop]) / (sqrt (2) * noise_timing));

            temp_rate = (0.5 * (ERFC (x_minus) + ERFC (x_plus)));

            *(Error_rate + err_cnt) += temp_rate;

        }

        loop++;

    }

    while (*(Result_array + loop) != 1000.0);

    loop--;

    *(Error_rate + err_cnt) /= loop;

    printf ("\nPacking density: %d   D: %e", (packing_density / 2), para_loop);

    printf (" Error Rate: %e\n", *(Error_rate + err_cnt));

    err_cnt++;

}

```

The above module is the code used to calculate bit error rates using the statistics rather than by generating real noise. The paper by Katz and Campbell [Katz, E. R., and Campbell, T. G., (May 1979)] was used to build this code and is dependant on error function complement to produce bit error rates. This bit error rate analysis has been done previously

and compared with experimental results to demonstrate the accuracy of the analysis technique. I used this as a comparison for my technique which involves generating real noise and ten million bit data trains.

(A.4.6) AWGN Generation Module

```

/*****

Thanks to the book 'NUMERICAL RECEIPES IN C' for the function 'ran2'.
See pages 282 - 283.

*****/

float ran2 (int *idum)
{
    int  j, k;
    static int idum2 = 123456789;
    static int iy = 0;
    static int iv[NTAB];
    float temp;

    if (*idum <= 0)
    {
        if (-(*idum) < 1)
            *idum = 1;
        else
            *idum = -(*idum);
        idum2 = (*idum);
    }
}

```

```

for (j = (NTAB + 7); j >= 0; j--)
{
    k = (*idum) / IQ1;
    *idum = IA1 * (*idum - k * IQ1) - k * IR1;
    if (*idum < 0)
        *idum += IM1;
    if (j < NTAB)
        iv[j] = *idum;
}
iy = iv[0];
}
k = (*idum) / IQ1;
*idum = IA1 * (*idum - k * IQ1) - k * IR1;
if (*idum < 0)
    *idum += IM1;
k = idum2 / IQ2;
idum2 = IA2 * (idum2 - k * IQ2) - k * IR2;
if (idum2 < 0)
    idum2 += IM2;
j = iy / NDIV;
iy = iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
    iy += IMM1;
if ((temp = AM * iy) > RNMX)
    return RNMX;
else
    return temp;
}

```



```
/******
```

Thanks to the book 'NUMERICAL RECEIPES IN C' for the function 'gasdev'.

See pages 289 - 290. This is the same as the function 'add_noise'

```
*****/
```

```
float add_noise (int *idum, float ex_1, float std_1)
```

```
{
```

```
float fac, rsq, v1, v2;
```

```
static int iset = 0;
```

```
static float gset;
```

```
if (iset == 0)
```

```
{
```

```
do
```

```
{
```

```
    v1 = 2.0 * ran2(idum) - 1.0;
```

```
    v2 = 2.0 * ran2(idum) - 1.0;
```

```
    rsq = v1*v1+v2*v2;
```

```
}
```

```
while (rsq >= 1.0 || rsq == 0.0);
```

```
fac = sqrt (-2.0 * log(rsq) / rsq);
```

```
gset = ex_1 + (std_1 * v1 * fac);
```

```
iset = 1;
```

```
return (ex_1 + (std_1 * v2 * fac));
```

```
}
```

```
else
```

```

{
    iset = 0;
    return (gset);
}
}

```

The above module is the code used to generate additive white gaussian noise values. The procedure 'add_noise' adds the AGWN into the main array, and the procedure 'ran2' generates a long run of statistically independent random values which allows for the accurate simulation of 10 million bit data trains worth of gaussian noise. I would like to thank the book 'NUMERICAL RECIPIES IN C' for helping me devise the above routines without which a successful outcome to my research would have been more difficult.

(A.4.7) New Technique Bit Error Rate Calculation Module

```

/*****

The function 'calc_errors' calculates the number of errors in the current
block.

*****/

void calc_errors (char unix_1, int *status_1, int *data_size_1,
                  int * error_numb_1, char *end_routine_1, char *decode_array,
                  char *output_name, char *test_array, int block_no_1,
                  char *store1, int *buffer_temp1,
                  int end_effect_bits_1, char *test_char_1)

```

```

{
int  temp_count, temp_count2, loop, size, loop3;

    /* Calculate number of errors in valid data */

if ((*status_1 == 0) || (*status_1 == 1))
    temp_count = (2 * end_effect_bits_1) + 3;
else
    temp_count = (2 * end_effect_bits_1) + 1;

if (*end_routine_1 != 'T')
{
    if ((*status_1 == 0) || (*status_1 == 1))
        temp_count2 = (2 * end_effect_bits_1) + 1;
    else
        temp_count2 = (2 * end_effect_bits_1) + 3;
}
else
{
    if ((*status_1 == 0) || (*status_1 == 1))
        temp_count2 = end_effect_bits_1 + 1;
    else
        temp_count2 = end_effect_bits_1 + 3;
}

loop3 = 1;
if (unix_1 == 'n')
    printf("\n");
for (loop = temp_count; (loop < (*data_size_1 - temp_count2)); loop++)
{

```

```

if (unix_1 == 'n')
{
    printf ("%c", *(decode_array + loop));

    if ((loop3 % 64) == 0)
        printf ("\n");

    loop3++;
}

if (*(test_array + 1 + loop) == '8')
    (*(test_array + 1 + loop)) = '0';

if (*(test_array + 1 + loop) == '9')
    (*(test_array + 1 + loop)) = '1';

if (*(decode_array + loop) != *(test_array + 1 + loop))
    (*error_numb_1)++;

}

size = *data_size_1 - temp_count2 - temp_count;

if (size == VALID_BLOCK)
    write_to_buffer (0, store1, output_name, decode_array,
                    temp_count, buffer_temp1);

*test_char_1 = *(decode_array + (*data_size_1 - temp_count2 - 1));

if (unix_1 == 'n')
    printf ("\nThe block number is %d\n", block_no_1);

    /* Number of errors in valid data has been calculated */
}

```

The above module is the code used to calculate the errors that occur in the new technique of calculating bit error rates described in this thesis. The above produce only counts the errors in a single block so it is called by the main program in a looping structure until the whole data train has been analysed.

(A.4.8) Dropout Generation Module

```
drop_out = 'F';  
if ((strcmp (c, "Y") == 0) || (strcmp (c, "y") == 0))  
{  
    drop_out = 'T';  
    if (model_type == 'G')  
    {  
        gen.A = value;  
        gen_case_model (Drp_pos_diff, Drp_neg_diff, &gen, begin_point, step_size, step_accu);  
        if ((strcmp (slim, "Y") == 0) || (strcmp (slim, "y") == 0))  
            pulse_slim (Drp_pos_diff, Drp_neg_diff, scale, sample, step_accu);  
    }  
    else  
    {  
        mix.A = value;  
        mixed_model (Drp_pos_rep, Drp_neg_rep, Drp_pos_diff, Drp_neg_diff,  
                     &mix, begin_point, step_size, step_accu);  
        if ((strcmp (slim, "Y") == 0) || (strcmp (slim, "y") == 0))  
            pulse_slim (Drp_pos_diff, Drp_neg_diff, scale, sample, step_accu);  
    }  
}
```

The above module is the code used to generate drop-out pulses which are added in the place of ordinary pulses if a drop-out occurs. These drop-out pulses differ from normal pulses as they have a different (a+d) value.

(A.4.9) Block, Symbol and Bit Errors (and Error Rates) Module

```

/*****

The function 'analyse_array' analyses the array.

*****/

void analyse_array (char **array_1, int no_tracks_1)
{
    int block_size, interleave_depth, row, temp_array_size, loop,
        bit_counter, symbol_counter, block_counter, temp_loop, error_cols,
        correctable_symbols, error_symbols, error_blocks, error_bits,
        disp_val, loop2, tmp, tmp2, pess_symbols, pess_bits,
        pess_symbol_count, pess_bit_count, error_bits_total;

    double bit_error_rate, symbol_error_rate, block_error_rate, pess_symbol_error_rate, pess_bit_error_rate;

    char **temp_array, valid, exit_proc, temp;

    do
    {
        error_blocks = 0;

        error_symbols = 0;

        error_bits_total = 0;

        pess_symbol_count = 0;

        pess_bit_count = 0;

        temp_loop = 0;

        bit_counter = 0;
    }
}

```



```

block_counter = 0;

symbol_counter = 0;

disp_val = 0;


printf("\nWhat is the interleave depth?: ");

scanf("%d", &interleave_depth);

getchar ();


printf("\nWhat is the block size?: ");

scanf("%d", &block_size);

getchar ();

temp_array_size = block_size * interleave_depth;


temp_array = (char **) malloc(no_tracks_1 * sizeof(char *));

for (row = 0; row < no_tracks_1; row++)
{
    temp_array[row] = (char *) malloc((temp_array_size + 1) * sizeof(char));

    if (! temp_array[row])
    {
        printf("\nCannot allocate all tracks ... not enough memory\n");

        exit (1);
    }
}

printf("\nHow many symbols to correct?: ");

scanf("%d", &correctable_symbols);

getchar ();

valid = 'F';

do
{

```

```

for (loop = 0; loop < temp_array_size; loop++)

for (row = 0; row < no_tracks_1; row++)

{

if (temp_array [row] [loop] != 'X')

    temp_array [row] [loop] = array_1 [row] [loop + disp_val];

else

    valid = 'T';

}

if ((valid == 'F') && (interleave_depth > 1))

    interleave_array (array_1, temp_array, disp_val, block_size,

                      no_tracks_1, interleave_depth);

for (loop2 = 0; loop2 < interleave_depth; loop2++)

{

    tmp = loop2 * block_size;

    tmp2 = tmp + block_size;

    if (valid == 'F')

    {

        error_cols = 0;

        for (loop = tmp; loop < tmp2; loop++)

            for (row = 0; row < no_tracks_1; row++)

            {

                if (temp_array [row] [loop] == '*')

                {

                    if (((loop - tmp) + disp_val) != temp_loop)

                    {

```

```

        error_cols++;

        temp_loop = ((loop - tmp) + disp_val);
    }

}

if (error_cols > correctable_symbols)
{
    error_blocks++;

    error_symbols += error_cols;

    pess_symbols = error_cols + correctable_symbols;

    if (pess_symbols > block_size)

        pess_symbols = block_size;

    pess_symbol_count += pess_symbols;

    error_bits = 0;

    for (loop = tmp; loop < tmp2; loop++)

        for (row = 0; row < no_tracks_1; row++)

            if (temp_array [row] [loop] == '*')

                error_bits ++;

    error_bits_total += error_bits;

    pess_bits = error_bits + (correctable_symbols * no_tracks_1);

    if (pess_bits > (block_size * no_tracks_1))

        pess_bits = block_size * no_tracks_1;

    pess_bit_count += pess_bits;
}

block_counter++;

symbol_counter += block_size;

disp_val += block_size;

bit_counter += (block_size * no_tracks_1);
}

```

```

    }

}

while (valid == 'F');

free (temp_array);

block_counter--;

symbol_counter -= block_size;

bit_counter -= (block_size * no_tracks_1);

exit_proc = 'F';

bit_error_rate = ((double) error_bits_total / (double) bit_counter);

symbol_error_rate = ((double) error_symbols / (double) symbol_counter);

block_error_rate = ((double) error_blocks / (double) block_counter);

pess_symbol_error_rate = ((double) pess_symbol_count /

(double) symbol_counter);

pess_bit_error_rate = ((double) pess_bit_count / (double) bit_counter);

printf("\n\ninterleave depth = %4d ", interleave_depth);

printf("block size = %3d ", block_size);

printf("symbols corrected = %3d", correctable_symbols);

printf("\n\nblocks in error = %8d ", error_blocks);

printf("block error rate = %e", block_error_rate);

printf("\n\n symbols in error = %8d ", error_symbols);

printf("symbol error rate = %e ", symbol_error_rate);

printf("\n\nbits in error = %8d ", error_bits_total);

printf("bit error rate = %e ", bit_error_rate);

printf("\n\nDo you want to do another ?: ");

scanf ("%c", &temp);

getchar ();

if ((temp == 'N') || (temp == 'n'))

    exit_proc = 'T';

```

```

}

while (exit_proc == 'F');

}

```

The above module is the code used to analyse block, symbol and bit errors (and error rates). It is used in the analysis of error performance schemes described in chapter 5 of this thesis.

(A.4.10) Vertical Stacking Arrangement Module

```

void vertical_transform (char **new_array_1, char **orig_array_1, int max_col, int max_row)
{
    int row, col, loop = 0;

    do
    {
        for (col = 0; col < max_col; col++)
        {
            for (row = 0; row < 8; row++)
            {
                new_array_1 [col] [(row + loop)] = orig_array_1 [row] [(col + loop)];
            }
        }

        loop += 8;
    }

    while (loop < max_row);
}

```

```

for (col = 0; col < 8; col++)
{
    new_array_1 [col] [max_row] = 'X';
}
}

```

The above module is the code used to transform blocks into a vertical stacking arrangement so that vertical stacking interleaving can be achieved.

(A.4.11) Symbol Interleaving Module

```

/*****

The function 'interleave_array' interleaves the array.

*****/

void interleave_array (char **array_2, char **temp_array_1,
                      int disp_val_1, int block_size_1, int no_tracks_2,
                      int interleave_depth_1)
{
    int loop, depth, pos, row;

    loop = 0;
    do
    {
        depth = 0;
        do

```



```

    {
        pos = ((depth * block_size_1) + (loop / interleave_depth_1));
        for (row = 0; row < no_tracks_2; row++)
            temp_array_1[row][loop] = array_2 [row] [(pos + disp_val_1)];
        depth++;
        loop++;
    }
    while (depth < interleave_depth_1);
}
while (loop < (block_size_1 * interleave_depth_1));
}

```

The above module is the code used to interleave symbols either horizontally or vertically as described in chapter 5.

(A.4.12) Block Interleaving Module

```

/*****

The function 'interleave_array_2' interleaves the array.

*****/

void interleave_array_2 (char **array_2, char **temp_array_1,
                        int disp_val_1, int block_size_1, int no_tracks_2, int interleave_depth_1)
{
    int loop, block_depth, pos, row, col;
    loop = 0;
    do

```

```

{
    block_depth = 0;
    do
    {
        pos = ((block_depth * block_size_1) + ((loop / (interleave_depth_1 * 8) * 8)));
        for (col = 0; col < 8; col++)
        {
            for (row = 0; row < no_tracks_2; row++)
            {
                temp_array_1 [row][loop] = array_2 [row] [(pos + disp_val_1)];
            }
            loop++;
            pos++;
        }
        block_depth++;
    }
    while (block_depth < interleave_depth_1);
}
while (loop < (block_size_1 * (interleave_depth_1)));
}

```

The above module is the code used to interleave blocks as described in chapter 5.

(A.4.13) Block and Symbol Interleaving Module

```
/******  
  
The function 'interleave_array_3' interleaves the array.  
  
*****/  
void interleave_array_3 (char **array_2, char **temp_array_1,  
                        int disp_val_1, int block_size_1, int no_tracks_2, int interleave_depth_1)  
{  
    int loop, depth, block_depth, pos, row, col, temp_array_size_1;  
    char **temp_array_2;  
  
    temp_array_size_1 = block_size_1 * interleave_depth_1;  
    temp_array_2 = (char **) malloc(8 * sizeof(char *));  
    for (row = 0; row < 8; row++)  
    {  
        temp_array_2[row] = (char *) malloc((temp_array_size_1 + 1) * sizeof(char));  
        if (!temp_array_2[row])  
        {  
            printf("\nCannot allocate all tracks ... not enough memory\n");  
            exit (1);  
        }  
    }  
    loop = 0;  
    do  
    {  
        block_depth = 0;  
        do
```

```

{
    pos = ((block_depth * block_size_1) + ((loop / (interleave_depth_1 * 8) * 8)));
    for (col = 0; col < 8; col++)
    {
        for (row = 0; row < no_tracks_2; row++)
        {
            temp_array_2 [row][loop] = array_2 [row] [(pos + disp_val_1)];
        }
        loop++;
        pos++;
    }
    block_depth++;
}

while (block_depth < interleave_depth_1);
}

while (loop < (block_size_1 * (interleave_depth_1)));

loop = 0;
do
{
    depth = 0;
    do
    {
        pos = ((depth * block_size_1) + (loop / interleave_depth_1));
        for (row = 0; row < no_tracks_2; row++)
            temp_array_1 [row][loop] = temp_array_2 [row] [pos];
        depth++;
        loop++;
    }
}

```

```
        while (depth < interleave_depth_1);  
    }  
    while (loop < (block_size_1 * interleave_depth_1));  
    free (temp_array_2);  
}
```

The above module is the code used to interleave both blocks and symbols as described in chapter 5.

APPENDIX B

B.1 Published Papers

I have published the following papers:

Tandon, A., Middleton, B.K., and Miles, J.J., "*Identification of Noise and ISI Limited Performance in Digital Magnetic Tape Recording Systems*", J. Inf. Recording, Vol. 23, pp347-351, 1996.

Tandon, A., Middleton, B.K., Miles, J.J., and Cumpson, S.R., "*The Application of Scaling to the Prediction of Digital Magnetic Recording Systems*", J. Phys. D: Appl. Phys., Vol. 30, pp542-545, 1997.

Tandon, A., Middleton, B.K., Farrell, P.G., and Miles, J.J., "*A Simulation of a Noisy and Drop-out Infected Multi-track Digital Magnetic Recording Channel for 10 Million bit Data Trains*", J. Inf. Recording, Vol. 23, pp469-487, 1997.

IDENTIFICATION OF NOISE AND ISI LIMITED PERFORMANCE IN DIGITAL MAGNETIC TAPE RECORDING SYSTEMS

A. TANDON, B. K. MIDDLETON and J. J. MILES

*Division of Electrical Engineering The Manchester School of Engineering
University of Manchester Oxford Road Manchester, M13 9PL, England*

Simulations have been carried out to show how by observing the variation of error rate with record current amplitude in digital tape recording systems the relative contributions of noise and ISI to limiting system performance can be identified.

KEY WORDS: Magnetic recording, digital recording.

1. INTRODUCTION

It is well known that record current amplitude in digital magnetic tape recording systems has to be optimised to minimise the probability of error of replayed information. The process of optimisation selects record current values which shape the replayed waveforms in such a way that the effects of variations of signal amplitude and peak shift, *i.e.* intersymbol interference (ISI), are, in combination with noise, minimised and lead to the lowest error rate. This study identifies a method which shows which of ISI and noise dominates the process of error formation at optimum record currents.

This work reports on some new simulations based on earlier work by Li [1] which described some simulations of a digital recording system which had earlier been built and tested by Hudson *et al.* [2]. That recording system used a magnetic tape transport operating at densities between 15,000 and 20,000 bits per inch and gave a number of results relevant to the process of current optimisation.

2. THEORETICAL STUDIES AND EXPERIMENTAL RESULTS

A computer model of a digital recording system employing peak detection [1, 2] has been developed which employs delay modulation encoded pseudo random data patterns for which the output waveforms are constructed by the linear superposition of isolated pulse shapes. The replayed waveforms are amplified, filtered

Correspondence to be addressed to Professor B. K. Middleton.

(differentiated), and cross over detected to (hopefully) reproduce the digital waveforms previously recorded. The effects of peak shift and noise on the positions of the crossovers is to move them from their intended positions and if the new positions are outside the timing window of operation of the decoder errors occur. This process can be accounted for in the manner described by Katz and Campbell [3] to determine the probability with which crossovers are moved outside the decoder timing window to produce errors. Therefore the probability of error is calculated.

The output pulse shapes were expressed in terms of parameters which were measured experimentally by Hudson *et al.* [2] and which were tabulated by Li [1]. These parameters were used for the calculation of the output pulse shapes which were then used in the superposition process. Figure 1 shows the corresponding output voltage amplitude versus record current for different packing densities, and also pulse width P_{50} and the parameter $(a_t + d)$, transition width plus head to medium spacing on replay, as functions of record current. The outputs computed for Figure 1 matched very closely the experimental results of Hudson *et al.* [1, 2] as would be expected.

Figure 2 shows some predicted error rate curves as a fraction of record current for different timing windows, expressed as a fraction of bit cell size, and different noise

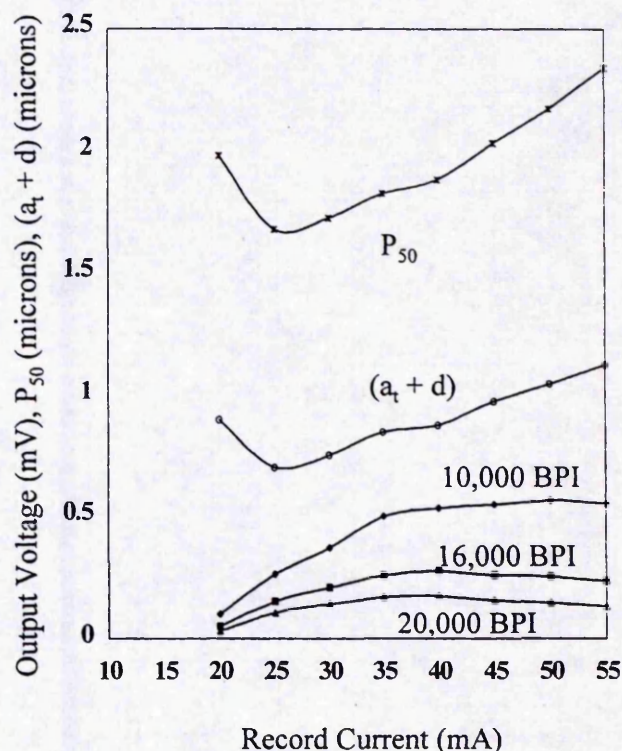


Fig. 1 Output voltage amplitude, at various bit packing densities, pulse width p_{50} and the parameter $(a_t + d)$ as functions of record current amplitude.

levels at a packing density of 17000 bits per inch (670 bits per mm). Curve *a* shows good agreement with the experimental curve produced by Hudson *et al.* [2] while curves *b* and *c* for higher noise levels and wider timing windows show minima at higher record current levels. Note that the curve marked *a* in Figure 2 has a minimum at a current amplitude of 30 mA which is close to that which minimises both $(a_i + d)$ and the pulse width in Figure 1. This indicates that since narrower pulses cause less overlapping at high densities, and therefore reduced peak shift, that current optimisation has minimised peak shift. Therefore the position of the minimum is an indicator that in this case performance is limited by the effects of peak shift. The curve marked *c* in Figure 2, which corresponds to a high noise situation has its minimum at 40 mA which corresponds to the current which maximises output amplitude. Therefore it appears that in this case optimisation of current is to maximise signal to noise ratio and therefore for this setting the system was noise limited. This is a situation similar to those of Brown and Middleton [4] and Middleton and Jack-Kee [5]. Therefore the range of optimisation curves in Figure 2 has been reflected in experimental observations. Curve *b* represents a situation intermediate between these two.

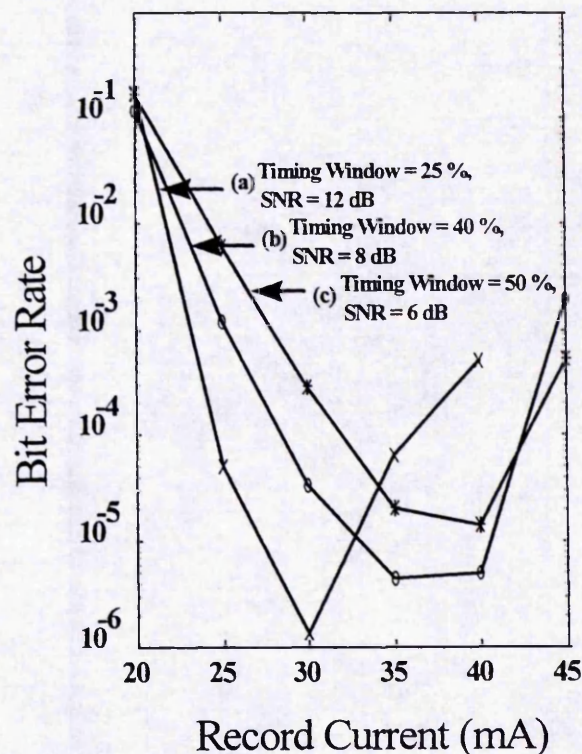


Fig. 2 Predictions of digital recording system error rate as a function of record current amplitude for different timing windows and signal to noise ratios.

Figure 3 shows predictions of optimum current as a function of signal to noise ratio for three different timing windows. Some interpolation of results has taken place to produce more points on these curves than were originally obtained experimentally. It can be seen that higher noise and therefore lower signal to noise ratio pushes the optimum current towards higher values to indicate noise limited performance as discussed above. Narrower timing windows reduce optimum record current to indicate peak shift limited performance also as explained above. Both of these results follow from the discussion in Figure 2.

3. CONCLUSIONS

It has been shown by simulations that optimum record currents in digital recording systems are a reflection of the contributions of noise and peak shift to the error rates occurring in those systems. It has therefore been demonstrated that by plotting graphs of error rate as a function of record current level that it is possible to obtain information on the limiting processes which occur in those systems. Such information is important as it points to critical tape and system signal and noise characteristics is need of adjustment or alteration during system development.

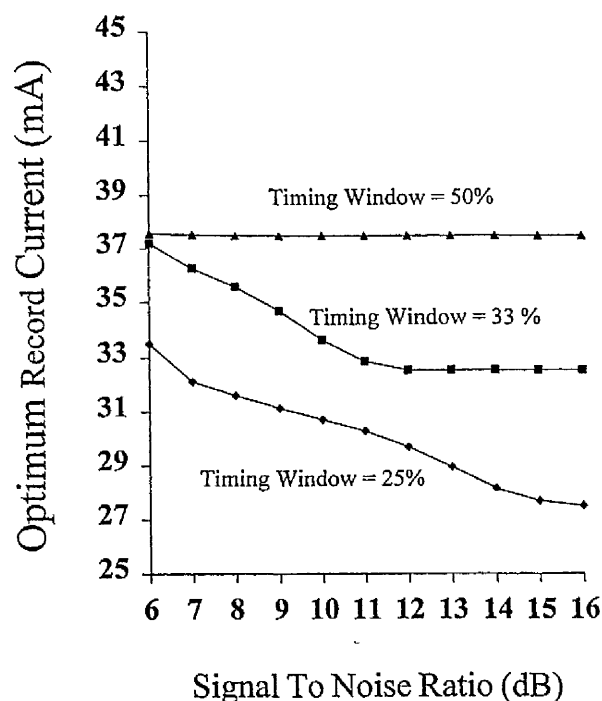


Fig. 3 Optimum record current as a function of signal to noise ratio for different timing windows.

ACKNOWLEDGEMENT

Thanks are due to Professor P. G. Farrell for helpful discussions, EPSRC for a studentship for A. Tandon, and British Gas for support of B. K. Middleton.

References

- [1] J.K. Li Hui Hong, B.K. Middleton and J.J. Miles: *J. Magn. and Magn. Mat.*, **120** 206 (1993).
- [2] V.N. Hudson, M.K. Loze and B.K. Middleton: *IERE Conf. Proc.* **67** 177 (1986).
- [3] E.R. Katz and T.E. Campbell: *IEEE Trans. Magn.* MAG-15, 1050 (1979).
- [4] B.K. Middleton and T. Brown: *Radio and Elec. Eng.*, **50** 467 (1980).
- [5] B.K. Middleton and T. Jack-Kee: *Radio and Elec. Eng.*, **53** 393 (1983).

The application of scaling to the prediction of the performance of digital magnetic recording systems

A Tandon, B K Middleton, J J Miles and S R Cumpson

Division of Electrical Engineering, The Manchester School of Engineering,
University of Manchester, Oxford Road, Manchester M13 9PL, UK

Received 17 July 1996

Abstract. The performances of digital magnetic recording systems using peak detection are shown to be described by equations that involve parameters and variables which occur in ratios. Provided that these quantities vary in proportion there is no change in system performance. Thus scaling of one parameter shows how scaling of another leads to a system of unchanged performance. It is further shown that the influence of a wide range of values of parameters on system performance can be accounted for by a limited range of computations. Results are given for media and systems of high performance which arguably represent the current state of the art and possible future aspirations.

1. Introduction

The performance of a digital magnetic recording system using peak detection has previously been predicted, for a given set of tape and system parameters, at particular packing densities and compared with experimental results on a working system [1]. Thus the foundations for successful computer modelling of digital recording systems have been laid. The aim of this work has been to demonstrate that it is possible to scale the values of the various parameters and variables involved without altering system performance and also to produce predictions for the new parameters without repeating the calculations. This is particularly useful when extrapolating from what is understood and practicable to what is hoped to be achieved. The reasons why this is possible is explained with reference to relevant recording and system performance expressions, and computations are carried out to give some numerical results and show how they may be used. The computations are carried out for a range of values of variables which correspond to currently achievable technology and extend through to values which could be considered limiting values for digital recording.

Computations are restricted to peak detection systems employing inductive replay heads and are intended to be particularly useful for showing how changes in recorded transition width, caused by changes of media and record head properties, and replay head-to-medium spacing influence system performance. They do not address scaling during the recording process [2] or the problem of changes of equalization and detection strategy, which is an additional consideration for attention elsewhere.

2. Theory

The expressions for output voltages $e(\bar{x})$ from the replay head of a digital recorder can be expressed as ratios of various parameters, as indicated in equation (1) for an isolated pulse shape [3,4]:

$$e(\bar{x}) = f\left(\frac{\bar{x}}{a+d}, \frac{D}{a+d}, \frac{g}{a+d}\right) \quad (1)$$

where $\bar{x} = vt$, the medium velocity-time product, equal to the distance along the medium; a is the transition width parameter; d is the head-to-medium spacing on replay; g is the replay head gap length; and D is the recording medium thickness. In the case in which the output waveform is produced by an isolated transition and a very narrow gap head [1,3-5]

$$e(\bar{x}) = \frac{\mu_0 v w M_0 n \eta}{\pi} \ln \left(\frac{\left(\frac{\bar{x}}{a+d}\right)^2 + \left(1 + \frac{D}{a+d}\right)^2}{\left(\frac{\bar{x}}{a+d}\right)^2 + 1} \right) \quad (2)$$

where μ_0 is the permeability of free space, w is the track width, M_0 is the magnetization either side of a recorded transition, n is the number of turns on the head and η is the head efficiency.

A narrow gap head is chosen for these computations simply to focus attention onto the term $a+d$, because intense efforts are being made to reduce their values. Inclusion of finite g does not alter any principle but makes the algebra more cumbersome.

At high densities the output according to linear superposition is given by

$$e(\bar{x}) = \sum_n a_n f\left(\frac{\bar{x} + nb}{a+d}, \frac{D}{a+d}, \frac{g}{a+d}\right) \quad (3)$$

where $a_n = 0, \pm 1$ and b is the bit size. It can be seen that reducing all the parameters by the same factor leaves the output voltage at the same magnitude but increases the packing density b^{-1} as a result of the reduction in b . The output waveforms remain the same shape for any amount of scaling except that the distance and time scales are altered.

The probability of error for the recovery of any bit in a digital recording system employing peak detection, consisting of amplification, differentiation and cross over detection, is given by [6]

$$P_e = 0.5 \left[\operatorname{erfc} \left(\frac{T_w + T_{ps}}{T_n} \right) + \operatorname{erfc} \left(\frac{T_w - T_{ps}}{T_n} \right) \right] \quad (4a)$$

where $2T_w$ is the timing window of detector and decoder; T_{ps} is the peak shift in output waveform, equal to the cross over shift in differentiated waveform; and T_n is the cross over shift due to noise. The above expression is written in terms of time but by multiplying the top and bottom of the fractions by v they could be expressed in terms of distance, as are the earlier equations:

$$P_e = 0.5 \left[\operatorname{erfc} \left(\frac{x_w + x_{ps}}{x_n} \right) + \operatorname{erfc} \left(\frac{x_w - x_{ps}}{x_n} \right) \right] \quad (4b)$$

where

$$x_w = vT_w$$

$$x_{ps} = vT_{ps}$$

$$x_n = vT_n$$

Then (4a) and (4b) may be written in the modified format

$$P_e = 0.5 \left[\operatorname{erfc} \left(\frac{1 + \frac{T_{ps}}{T_b} \frac{T_n}{T_w}}{\frac{T_n}{T_b} \frac{T_n}{T_w}} \right) + \operatorname{erfc} \left(\frac{1 - \frac{T_{ps}}{T_b} \frac{T_n}{T_w}}{\frac{T_n}{T_b} \frac{T_n}{T_w}} \right) \right] \quad (5a)$$

$$= 0.5 \left[\operatorname{erfc} \left(\frac{1 + \frac{x_{ps}}{b} \frac{b}{x_w}}{\frac{x_n}{b} \frac{b}{x_w}} \right) + \operatorname{erfc} \left(\frac{1 - \frac{x_{ps}}{b} \frac{b}{x_w}}{\frac{x_n}{b} \frac{b}{x_w}} \right) \right] \quad (5b)$$

wherein all the variable quantities appear as ratios. Again, provided that $(T_w/T_b) = (x_w/b)$ is kept constant, the probability of error will remain the same upon scaling the output waveform.

The cross over shift due to noise has been shown to be given by [6]

$$T_n = n'/s'' \quad (6a)$$

where n' is the differential of noise at the replay head and s'' is the corresponding double differential of the signal. By using $s'' = (\pi/T_b)s'$, where s' is the differential of the output signal, and expressing (6a) in reduced form, we obtain

$$\frac{T_n}{T_b} = \frac{x_n}{b} = \frac{n'}{\pi s'} \quad (6b)$$

where s'/n' now represents the signal-to-noise ratio out of the differentiator (RMS:RMS). This is proportional to the signal-to-noise ratio of the head s/n [7] and, if the bandwidth of the differentiator is $\sqrt{3\pi}/T_b$, which is reasonable, $s/n = s'/n'$ [7]. Thus for these calculations

$$\frac{T_n}{T_b} = \frac{n}{\pi s} \quad (6c)$$

Table 1. Values for the various parameters used in the computations.

	a (μm)	d (μm)	$a + d$ (μm)	D (μm)
1	0.1	0.076	0.176	0.025
2	0.008	0.076	0.084	0.025
3	0.006	0.040	0.046	0.025
4	0.003	0.025	0.028	0.025
5	0	0.025	0.025	0.025
6	0	0.025	0.025	0.0025

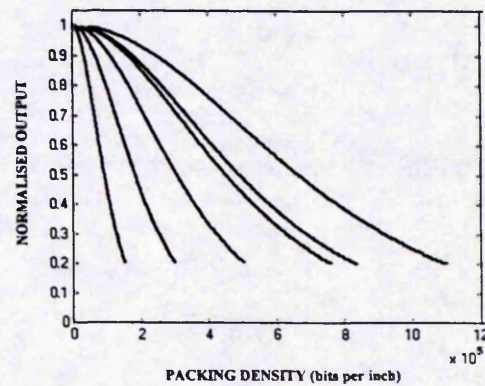


Figure 1. The normalized output amplitude as a function of the packing density b^{-1} for the various media and system parameters indicated in table 1.

where s is determined from the output versus packing curve as described later.

All the equations are now forms involving ratios and hence scaling can take place. It will be shown that, from a limited number of roll-off curves, calculations of error probability are possible for a wide range of media and circumstances.

3. Calculations

The roll-off curves resulting from the superposition of a regular sequence of the pulse shapes given by (2) of alternating polarity as calculated using (2) and (3) are shown in figure 1. Six different media are modelled, which possess a wide range of values of a and d , varying from values currently realized in practice to values which arguably represent what might be limiting values (see table 1). The larger values of d correspond to flying heights of heads which are currently used in rigid disk machines whereas the value of $0.025 \mu\text{m}$ represents a value which may arguably represent the lowest possible value in view of surface roughnesses both of heads and of media. The transition width of 0 represents the ultimate possibility and the roll-off curve reaching out to highest densities therefore appears to be a limiting achievement. Once this has been reached, increases in packing density can only be achieved by working further down the roll-off curve

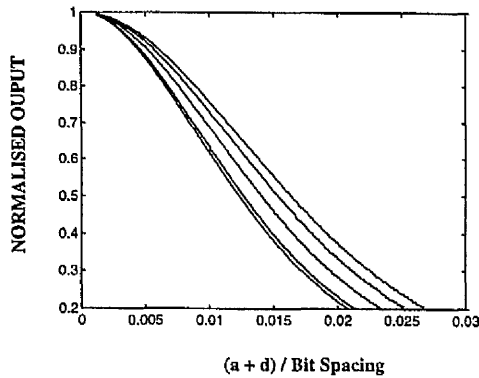


Figure 2. The normalized output amplitude as a function of the reduced packing density for media 1-5 in table 1.

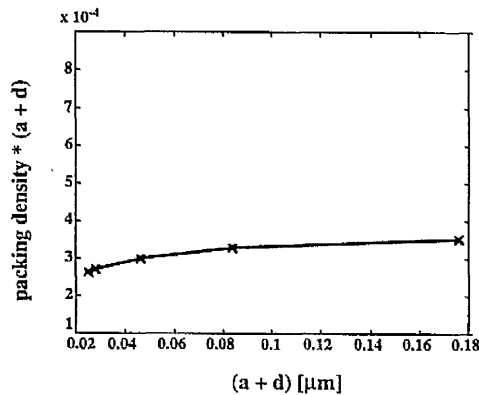


Figure 3. The packing density $x(a + d)$ as a function of $(a + d)$ for media 1-5 in table 1.

by employing superior signal processing techniques in the replay channel. Figure 2 gives five of these same curves (1-5 in table 1) but plotted in normalized form for five different values of $D/(a + d)$. These are universal curves with all quantities scaled. Although figure 1 spans a wide range of packing densities, in absolute terms, figure 2 shows that, in normalized form, the range of packing densities of interest is much smaller. Thus a limited number of curves is needed to span the range of variables likely to be used in the future. Figure 3 shows the product of packing density (at 60% of low-density output) and $a + d$ plotted as a function of $a + d$ for media 1-5 in table 1. It shows that the former varies in a limited way with the latter except at small values of $a + d$.

The following computations on system performance were carried out for a peak detection system employing amplification, differentiation, cross over detection and delay modulation. Pseudo-random data sequences of length 127 bits were used and the error rates quoted are the averages for the sequence [1].

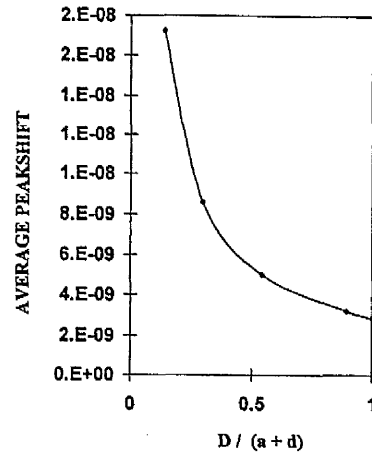


Figure 4. The average peak shift in a 127 bit pseudo-random sequence as a function of $D/(a + d)$ for media 1-5 in table 1.

Figure 4 shows the average peak shift in 127 bit pseudo-random sequences plotted against $D/(a + d)$ at packing densities which are given by the normalized output of 0.6 according to figure 2. This same average peak shift as a fraction of bit spacing is a constant independent from the value of $a + d$ and D , as would be expected from simple considerations of the geometrical aspects of the waveforms involved. This result is represented by

$$\frac{\text{Average peak shift}}{\text{Bit spacing}} = 4 \times 10^{-2} \quad (7)$$

Average bit error rates were computed for the 127 bit pseudo-random sequence using (5a) and (5b) for the different media listed for figure 1 at the packing densities corresponding to a reduced output of 0.6 on the roll-off curves of figures 1 and 2. The noise level for each medium was set at 10 dB below the signal level and the reduced timing window was set at $2x_w/b = 0.5$. With the signal and peak shift having been shown to scale, with a fixed signal-to-noise ratio and a fixed reduced timing window it turns out that the bit error rate is constant at

$$\text{Bit error rate} \approx 3 \times 10^{-7} \quad (8)$$

and is independent from packing density and $a + d$.

4. Discussion and conclusion

Figure 1 has shown the wide range of digital recording performances that can be obtained by reduction of the parameter $a + d$ which is the major factor limiting digital recording system performance. The same results when plotted in normalized form in figure 2 display a rather limited range of roll-off curves. For these curves the average peak shifts divided by the bit spacings and the bit error rates have all been shown to be constant.

A simple way to use this information is to note a particular value of $a + d$ and then use figure 3 to obtain the

corresponding packing density at 60% of maximum output. At this packing density there is a bit error rate of 3×10^{-7} using a peak detection system. If technology is able to reduce $a + d$ by a certain amount, figure 3 will give the new packing density that is possible while maintaining the same error rate.

The values for $a + d$ used in these computations represent numbers which vary from what is currently attainable in thin film disk media through to what might or might not be ultimately attainable. They show for the best media that a packing density of around 570 000 bits per inch at an output reduced to 60% of the low-density value is possible for peak detection. Packing densities up to and beyond 1 Mbit in⁻¹ do not appear easily attainable using improvements in media as indicated by $a + d$ and D since the values of $0.025 \mu\text{m}$ for both parameters might be difficult to reduce further. Improved system performances will need lower noise media and will need new strategies for signal handling and noise reduction to make operation further down the roll-off curves possible.

Acknowledgments

Thanks are due to the EPSRC for a Research Studentship for A Tandon, to British Gas for support for B K Middleton and to the CAMST for general background support for information storage research at Manchester University.

References

- [1] Li Hui Hong J K, Middleton B K and Miles J J 1993 *J. Magn. Magn. Mater.* **120** 206
- [2] Mallinson J C 1996 *IEEE Trans. Magn.* **32** 599
- [3] Middleton B K 1996 Recording and reproducing processes *Magnetic Recording Technology* ed C D Mee and E D Daniel (New York: McGraw-Hill) ch II
- [4] Middleton B K, Wright C D, Cumpson S R and Miles J J 1995 *IEEE Trans. Magn.* **31** 2365
- [5] Middleton B K and Jack-Kee T 1983 *Radio Electron. Eng.* **53** 393
- [6] Katz E R and Campbell T G 1979 *IEEE Trans. Magn.* **15** 1050
- [7] Hughes G F and Schmidt R H 1976 *IEEE Trans. Magn.* **12** 752

A SIMULATION OF A NOISY AND DROP-OUT INFECTED MULTI-TRACK DIGITAL MAGNETIC RECORDING CHANNEL FOR 10 MILLION BIT DATA TRAINS

A. TANDON, B. K. MIDDLETON, P. G. FARRELL and J. J. MILES

*Division of Electrical Engineering, The Manchester School of Engineering,
University of Manchester, Oxford Road, Manchester M13 9PL, England*

A new method for simulating a digital magnetic recording channel using peak detection and incorporating 10 million bit data trains is presented. The main features of this model include tape and head characteristics, multi-tracks, drop-outs, and bit error rates. To allow for the maximum flexibility in the recording channel, the characteristics of the channel are defined as variables. Two methods of simulating noise, and their effects on bit errors, are examined. The new method used involves applying additive white gaussian noise samples onto the replay and differentiated waveforms, while the classical method involves determination of the effect of noise using statistical analysis. By combining the former method with a 10 million bit data train, and using typical channel characteristics, around 100 errors are produced. The additional channel information that the former method includes means that very efficient error correction and detection schemes can be produced. Examples are given of the use of this simulation technique in system design and analysis.

KEY WORDS: Digital magnetic recording, Recording channel, Drop-out, Multi-track recording

1. INTRODUCTION

A variety of models that demonstrate the individual characteristics of digital magnetic recording channels utilising peak detection, and based on statistical analysis have been accomplished [1, 2, 3, 4, 5, 6, 7]. The aim of this work is to produce a new model that can incorporate a wide range of tape or disk properties, multi-tracks, drop-outs, additive white gaussian noise (AWGN), and simulate the passage of large data trains. These features, combined with the built-in flexibility of the model, enable the model to be used in many system design applications.

Analysis of bit error rates (BER's) will also be presented. To produce bit errors, noise will be produced by both a new AWGN sampling method and by statistical analysis [1]. The error rates generated by both noise methods are compared with previous simulations [2] and are used to demonstrate the validity of the new modelling technique. To produce meaningful results, using the new method, 10 million bit data trains are also required, as a typical tape has a bit error rate of 1 error in 10^5 bits.

A meaningful examination of the effects of drop-outs can also be produced using 10 million bit data trains. Of particular interest is the behaviour of drop-outs in a multitrack scenario, and so the effects of drop-outs and the statistical analysis of

both the length and width of drop-outs from previously published experimental data [3] are presented. Advanced drop-out burst error events can also be investigated using this new model. In the past this investigation has not been possible, because our previous models [4] had a limited data size of 1024 bits.

The paper is organised as follows. The theory of the recording channel will be described in section 2, with the flexibility of the model considered. The modelling of drop-outs is described in section 3 with an analysis of the statistics obtained from published experimental work [3]. To implement multi-tracks, section 4 examines the track handling issues involved. Bit error rate analysis is produced in section 5, by applying AWGN to the recording channel and by using statistical analysis. In section 6, examples of the use of this simulation model are discussed.

2. DIGITAL MAGNETIC RECORDING THEORY

The block diagram in Figure 1 shows the standard representation of a peak detection recording channel. The process of peak detection is carried out by the combination of differentiation and cross-over detection, using timing windows. To model the output waveform of the recording channel, before differentiation, a particular shape for an isolated pulse is examined [8]. This isolated pulse corresponds to a single step change in the record current. Linear superposition [9, 10] is then applied to a sequence of these pulses to produce a recording channel waveform. To demonstrate the behaviour of a typical recording channel, pseudo-random sequences, encoded using delay modulation, are simulated. Also, the processes required to regenerate the data stream from the recording channel waveform are considered.

Two important characteristics define the intersymbol interference (ISI) performance of the channel. The first characteristic is the maximum replay output voltage. This voltage is produced by using data patterns that, when encoded using delay modulation, have a square wave shape. At any individual packing density this maximum voltage can be obtained, but to produce performance analysis of the channel, maximum voltages are required over a wide packing density range. The second cha-

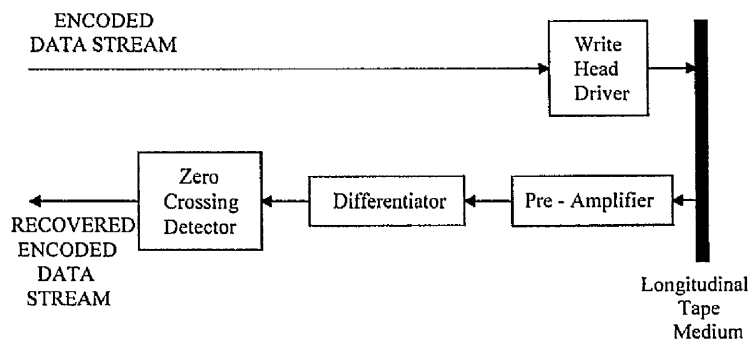


Fig. 1 Block diagram of a peak detection magnetic recording channel.

racteristic is the shifting of cross-overs influenced by ISI. This occurs due to the superposition of overlapping pulses and causes shifting of peaks as well as a reduction of amplitude. Thus zero crossing points are able to be pushed outside of their timing windows by ISI and so a source of errors is produced. To show the limits of the channel, average ISI over a wide packing density range will be produced.

When modelling a 10 million bit data size, a block structure method is used. This method will be described, ensuring that the effects of ISI are modelled accurately, with a minimal time penalty.

2.1. Definition and Simulation of the Recording Channel

Recent work defining output waveforms in longitudinal recording [8], has produced an accurate representation of an isolated replay pulse (whose parameters are in table 1), and is expressed as:

$$e(x) = \frac{\mu_0 v \omega M_x (\pi \eta)}{\pi(1+\alpha) (2g)} \left[\begin{aligned} & 2(a+d+D(1+\alpha)) \left[\arctan \left(\frac{g+x}{a+d+D(1+\alpha)} \right) + \arctan \left(\frac{g-x}{a+d+D(1+\alpha)} \right) \right] \\ & - 2(a+d) \left[\arctan \left(\frac{g+x}{a+d} \right) + \arctan \left(\frac{g-x}{a+d} \right) \right] \\ & + (g+x) \ln \left[\frac{(a+d+D(1+\alpha))^2 + (g+x)^2}{(a+d)^2 + (g+x)^2} \right] \\ & + (g-x) \ln \left[\frac{(a+d+D(1+\alpha))^2 + (g-x)^2}{(a+d)^2 + (g-x)^2} \right] \end{aligned} \right] \quad (1)$$

With the variables defined as:

- x = (head / medium) speed* time
- a_t = the recorded transition width at the top surface of the medium
- d = (head / medium) separation on replay
- D = the lesser of medium thickness and depth of recording
- α = the rate of change of transition width with depth into the medium
- μ_0 = the permeability of free space
- ω = the track width
- v = the tape velocity
- M_x = the peak magnetisation of the medium (longitudinal component)
- η = the replay head efficiency
- n = the number of turns of the replay head
- $2g$ = the gap length

The above expression is differentiated to produce the output from the differentiator which is:

$$\frac{d(e(x))}{dx} = \frac{\mu_0 v \omega M_x}{\pi(1+\alpha)} \left(\frac{n\eta}{2g} \right) \left[\begin{aligned} & 2 \left[\left(\frac{(a+d+D(1+\alpha))^2}{(a+d+D(1+\alpha))^2 + (g+x)^2} \right) - \left(\frac{(a+d+D(1+\alpha))^2}{(a+d+D(1+\alpha))^2 + (g-x)^2} \right) \right] \\ & - 2 \left[\left(\frac{(a+d)^2}{(a+d)^2 + (g+x)^2} \right) - \left(\frac{(a+d)^2}{(a+d)^2 + (g-x)^2} \right) \right] \\ & + 2 \left[\left(\frac{(g+x)^2}{(g+x)^2 + (a+d+D(1+\alpha))^2} \right) - \left(\frac{(g+x)^2}{(g+x)^2 + (a+d)^2} \right) \right] \\ & + \ln((g+x)^2 + (a+d+D(1+\alpha))^2) - \ln((g+x)^2 + (a+d)^2) \\ & - 2 \left[\left(\frac{(g-x)^2}{(g-x)^2 + (a+d+D(1+\alpha))^2} \right) + \left(\frac{(g-x)^2}{(g-x)^2 + (a+d)^2} \right) \right] \\ & - \ln((g-x)^2 + (a+d+D(1+\alpha))^2) + \ln((g-x)^2 + (a+d)^2) \end{aligned} \right] \quad (2)$$

Figures 2a and 2b show an example replay pulse and differentiated pulse respectively. An important consideration in the simulation of both the replay and differentiated pulses is the number of samples used to define a pulse. For a wide range of packing densities to be analysed, 3,500 samples per pulse are required typically over a distance of 40 ($a_i + d$).

To produce the recorded waveforms, linear superposition is carried out by adding the replay pulses and the differentiated pulses into two large single dimensional arrays at specific positions. Figures 3a and 3b show example replay and differentiated waveforms respectively, at a packing density of 100,000 bits per inch, coded

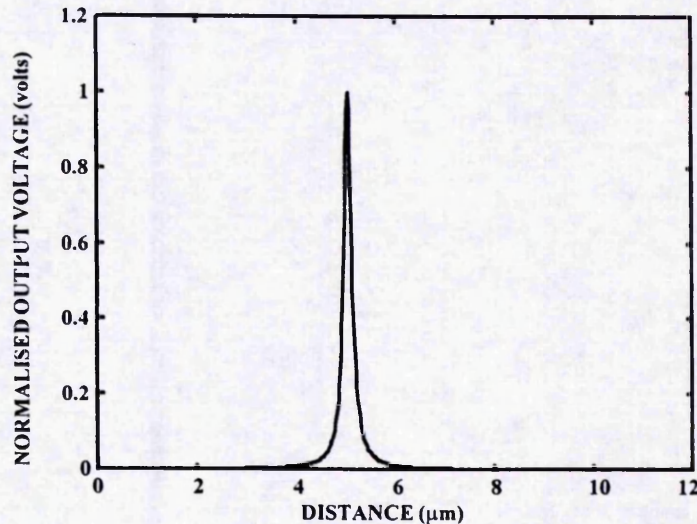


Fig. 2a A replay pulse generated using the parameters in Table 1.

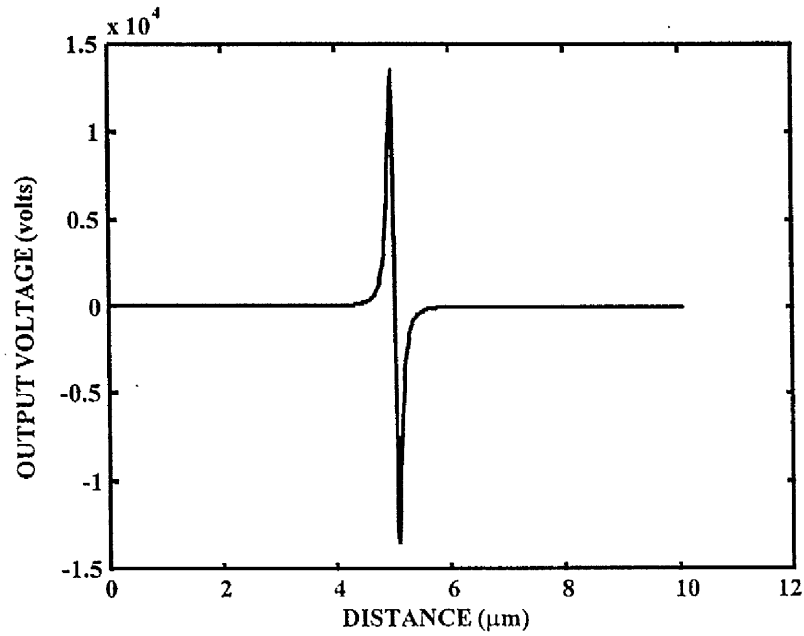


Fig. 2b A differentiated pulse generated using the parameters in Table 1.

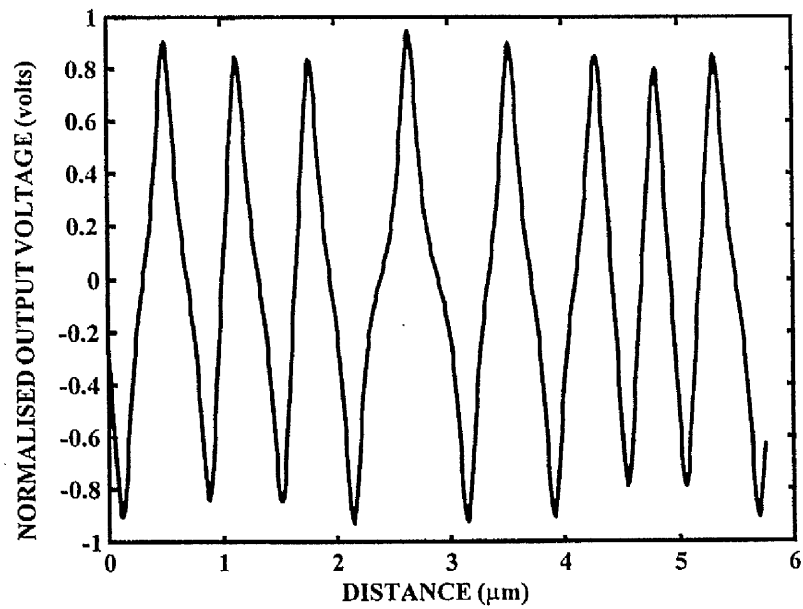


Fig. 3a A pseudo-random replay waveform, at a packing density of 100,000 bits per inch, generated using the parameters in Table 1.

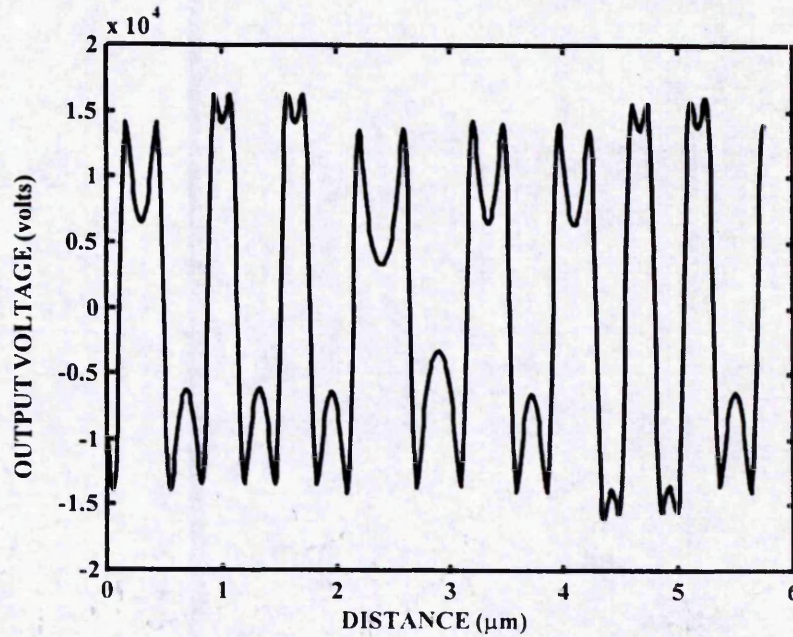


Fig. 3b A pseudo-random differentiated waveform, at a packing density of 100,000 bits per inch, generated using the parameters in Table 1.

using a pseudo-random data pattern. To determine the positions that pulses are placed at, the following expression is used:

$$\text{number of samples between two bits} = \frac{\text{spacing between two bits}}{\text{stepsize}} \quad (3)$$

with

$$\text{spacing between two bits} = \frac{1}{\text{packing density}} * 0.0254 \quad (4)$$

and

$$\text{step size} = \frac{2|(a_t + d)| * \text{pulse width multiplier}}{\text{number of samples per pulse}} \quad (5)$$

Where the spacing between two bits and the step size are in metres and the packing density is in bits per inch.

The step size is defined as the distance between two samples. The pulse width multiplier is a multiplication factor that is applied to produce a long pulse tail length. This long tail ensures that an accurate linear superposition process occurs. For the most flexible model, the pulse width multiplier has a value of 20. Typically, this is equivalent to approximately 150 samples per $(a_t + d)$.

This new sampling simulation method allows extra flexibility in channel analysis, with a wide range of packing densities able to be simulated without a significant time penalty. There is no restriction on the shape of pulse used, be it longitudinal, perpendicular, a combination of both, or alternatively user defined pulse shapes.

It is also possible to reconstruct the binary data stream from the differentiated waveform that has been previously produced. This reconstruction is achieved by assuming an ideal soft clock and superimposing timing windows onto every ideal bit position that is used. Figure 4 shows one such timing window, with the center of the timing window being at the zero crossing point of the bit if no ISI occurs. However, ISI causes the zero crossing point to be shifted from the central position. Errors can thus be produced, if ISI causes the zero crossing point to fall outside of its timing window. To illustrate bit reconstruction, Figure 5 shows a differentiated waveform with several timing windows. The slope within the timing window determines the state of the bit. If this slope is downwards, a 0 to 1 transition occurs. If the slope is upwards, a 1 to 0 transition occurs. Otherwise the recreated bit is assigned the same state as the previous decoded bit. Thus it is now possible to compare the binary data stream entering the channel with the binary data stream that has been recreated and so determine the errors that occur due to the magnetic channel.

2.2. Characteristics of the Recording Channel

To assess the performance of the recording channel, the magnetic channel characteristics are required. These characteristics are examined in terms of curves of the maximum replay output voltage and the average ISI over a large packing density range.

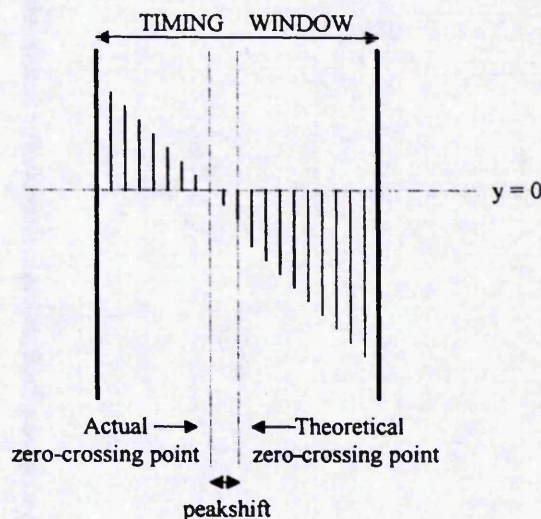


Fig. 4 A peakshift induced cross-over point within a timing window.

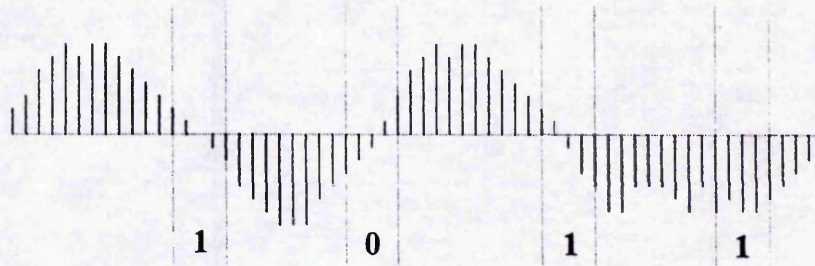


Fig. 5 Bit recovery within a sampled differentiated waveform.

Figure 6 shows the roll-off curve produced when examining the maximum output voltage over a wide packing density. It is clear to see from Figure 6 that the maximum output voltage deteriorates rapidly at high packing densities.

To analyse the average ISI over a large packing density range, a worse case data pattern is required. The worse case data pattern is the '011' repeated sequence, encoded using the delay modulation line code. Figure 7 shows the percentage ISI from the center of the timing window versus the packing density. It can be clearly seen that ISI performance is a significant limiting factor of the channel.

2.3. Extending the Model by Using a Block Structure Method

To conduct useful analysis of the effects of drop-outs and AWGN, the simulation is required to handle data sizes of 10 million bits. By the use of a block structure

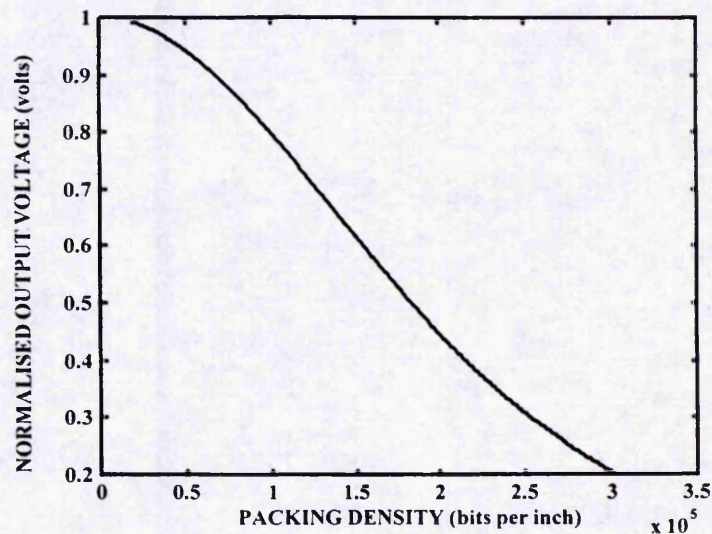


Fig. 6 Normalised output voltage as a function of packing density.

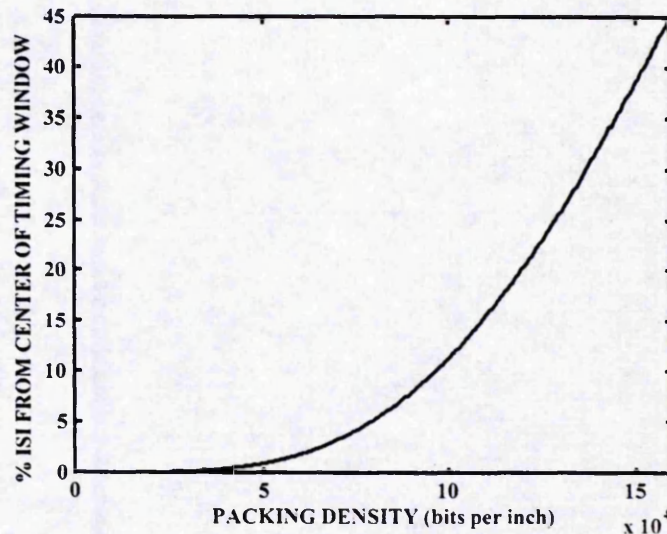


Fig. 7 % ISI from the center of timing window as a function of packing density.

method, as shown in Figure 8, the simulation can handle any input data size length specified. The data in the input data train is organised in blocks of 512 bits, with each bit in the block analysed individually to achieve an accurate simulation. For a particular block, (n), to be analysed successfully, information from the previous block (n-1) and the next block (n+1) are required to model the effects at the edges of the block (n).

Additional buffers are also required which are used to avoid end effects that occur at the extremes of the data pattern. To demonstrate this effect clearly, Figure 9 shows a data stream waveform where no end effect buffers exist. The two large pulses at either end of the waveform are the end effects, with usable data occurring between the end effect pulses. The end effect buffer is defined as a string of 1's coded using delay modulation. This creates a square wave which produces no ISI effects on the valid data range itself.

Therefore all the bits in block (n) can be superimposed accurately and the modelled data train produced. After that the next block (n+1) can be analysed, and so on until all the input data bits are simulated. No restriction now exists on the size of the input data train, and by using this method, the blocks of data are simulated efficiently.

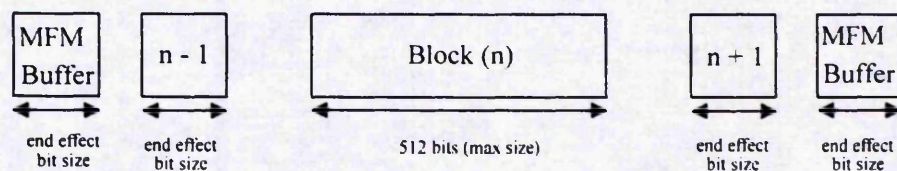


Fig. 8 The structure of data within a block.

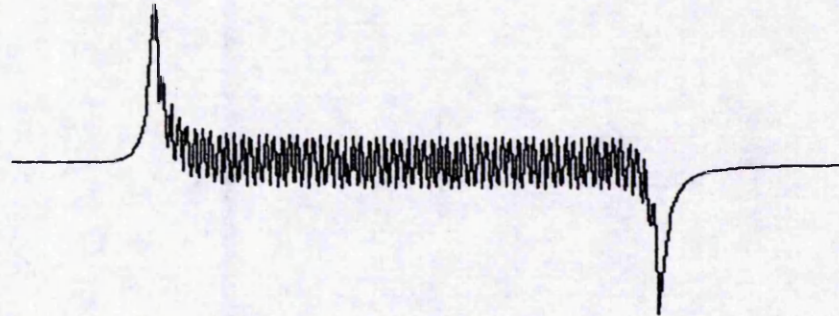


Fig. 9 A differentiated waveform demonstrating end-effects.

3. DROP-OUTS

Drop-outs occur, in general, by the incursion of dust between the recording medium and the replay head and this causes reductions in the reproduced signal amplitudes of the recorded waveform. The characteristics of the simulated drop-outs are examined in detail in this section and statistical analysis is performed to calculate the number, length, width and position of drop-outs. Drop-outs that cause the loss of clock synchronisation have not been considered.

3.1. Modelling the Drop-Out Characteristics

The major effect of a drop-out is to increase the size of parameter d (which is the (head / medium) spacing on replay). Also an increase in the pulse width of the drop-out waveform occurs when compared with the pulse width of the original waveform. The larger pulse width causes zero-crossing points of the differentiated waveform to be pushed outside of timing windows. Figure 10 shows a simulated drop-out which suffers an 80 % loss of signal amplitude compared with the original waveform signal output.

3.2. Statistical Analysis of Drop-Outs

3.2.1. The Dimensions of the Drop-Out

The statistics used to calculate the length of drop-out are taken from the previously published experimental results [3]. From experimental data curves the following equation is derived:

$$N_d = 52480 e^{\left(\frac{-d_d}{d_o}\right)} \quad (6)$$

Where N_d is the number of drop-out events per unit diameter, d_d is the diameter of a drop-out and d_o is a characteristic length associated with a drop-out ($= 0.143 \times 10^{-3} \text{m}$). Using the standard Monte Carlo method [11], random drop-outs with the correct length distribution are produced.

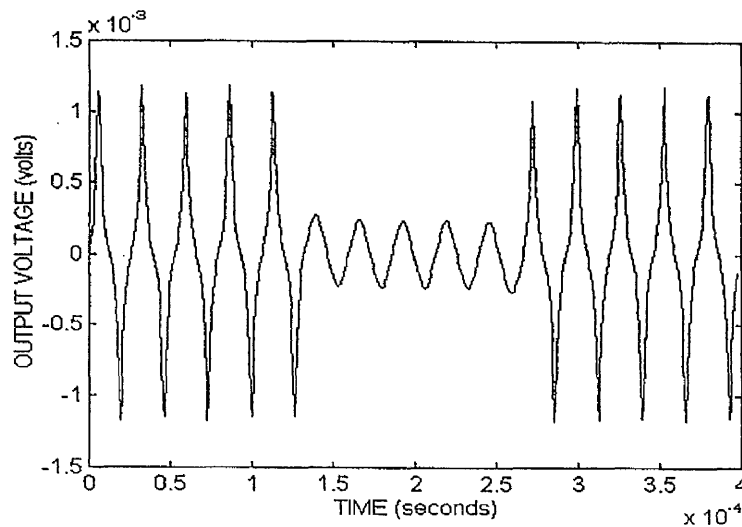


Fig. 10 An example waveform containing an 80 % loss of signal amplitude within a drop-out.

To determine how many tracks, in the absence of statistics, a drop-out covers a simple rule is used. In a multi-track scenario, the probability of the drop-out covering two tracks is taken to be half the probability of the drop-out covering a single track. The probability of the drop-out covering three tracks is taken to be half the probability of the drop-out covering two tracks, and so on.

3.2.2 The Total Number of the Drop-Out Events

The statistics used to calculate the total number of drop-out events are taken from previously published experimental results [3]. These show an exponential relationship between drop-out frequency and drop-out length. This leads to the total number of drop-out events (N) being defined as:

$$N = \int_{d_{\min}}^{d_{\max}} \frac{52480}{0.1 * 10^{-3}} e^{\left(\frac{-d_d}{d_o}\right)} \times \text{Area of tape} \quad (7)$$

with d_{\max} and d_{\min} being the maximum and minimum lengths of the drop-out respectively, d_d being the diameter of the drop-out and d_o being a characteristic length associated with a drop-out. We have found that these formulae fit experimental results when considering small and medium sized drop-out sizes, provided d_{\max} is carefully chosen. We assign d_{\min} with the size of a single bit ($1.643800155 * 10^{-7}$ m), and d_{\max} with the size of 152 bits ($0.025 * 10^{-3}$ m). By putting these values into (7):

$$\frac{N}{\text{Area of Tape}} = 11951.27677 \quad (8)$$

In this study, we use a linear density of 154,520 bits per inch, and the area of the tape used is 0,542925 m². Using this in (8) gives the number of drop-out events to be 6489 but this is for 1.4 Gbits of data. For 10 million bits of data, the pro rata number of drop-out events are scaled down by 140, and therefore the total of number of drop-out events used is 47.

Three large drop-outs are also added to simulate exceptional drop-outs that occur. The first drop-out occurs on a single track but has a size of 3000 bits, the second drop-out occurs over half the total number of tracks with a size of 1750 bits on each track, and the last drop-out covers all the tracks with a size of 1000 bits on each track.

4. IMPLEMENTATION OF MULTI-TRACKS

In a typical tape channel, where several tracks are recorded onto the tape, a drop-out may occur over many tracks or a single track. To implement multi-tracks, without incurring a restrictive time penalty, each track is simulated as a separate file. This strategy allows any number of tracks to be implemented without difficulty, and ensures that the end effect buffers are modelled accurately.

A four track tape is shown in Figure 11, as an example, with a drop-out covering all four tracks. To simulate this situation, the original information from each track is combined sequentially into an array. The drop-out is then added into this array, and each track is saved as individual files again. With the assumption that cross-talk between tracks is negligible, each track is simulated individually and so modelled accurately. To decode the simulated tracks, these tracks are again combined sequentially into an array. The resultant effects of the drop-out over several tracks can therefore be examined using this array.

5. INCORPORATING NOISE INTO THE SIMULATION

Simulating error events involves applying AWGN samples to a simulated magnetic channel [12], modelled with a large sequence of bits. The flexibility of this method

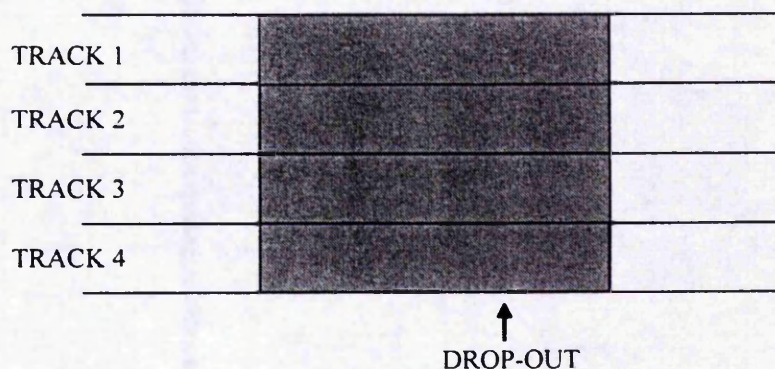


Fig. 11 A drop-out covering all four tracks in a four track tape.

allows for the AWGN samples to be replaced by noise samples with a different spectral content. This method produces meaningful BER's and also additional information about the cross-over shifts of each bit in the data train. These results are compared with those of a classical statistical method that involves using the probability of error upon small sequences of bits to produce meaningful BER's.

5.1. Calculating BER's by adding noise samples

To examine the internal characteristics of the magnetic channel, particularly the cross-over shift within timing windows, AWGN is applied to a noiseless differentiated channel. By defining the noiseless channel in terms of samples, it is straightforward to add AWGN samples to produce a noisy channel. The number of cross-overs occurring outside the timing windows can be examined using data trains of 10 million bits with a BER of around 1 error in 10^5 bits.

A classical method for producing gaussian samples is to use the Box, Muller and Marsaglia technique [11,13]. This technique produces a fast simulation algorithm that allows for a feasible time scale when simulating a 10 million bit data train. The expression used to produce the random deviates with a gaussian distribution is:

$$p(y)dy = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy \quad (9)$$

where $p(y)$ is the probability distribution of the random deviate y . An example of a noisy waveform produced using this method is shown in Figure 12.

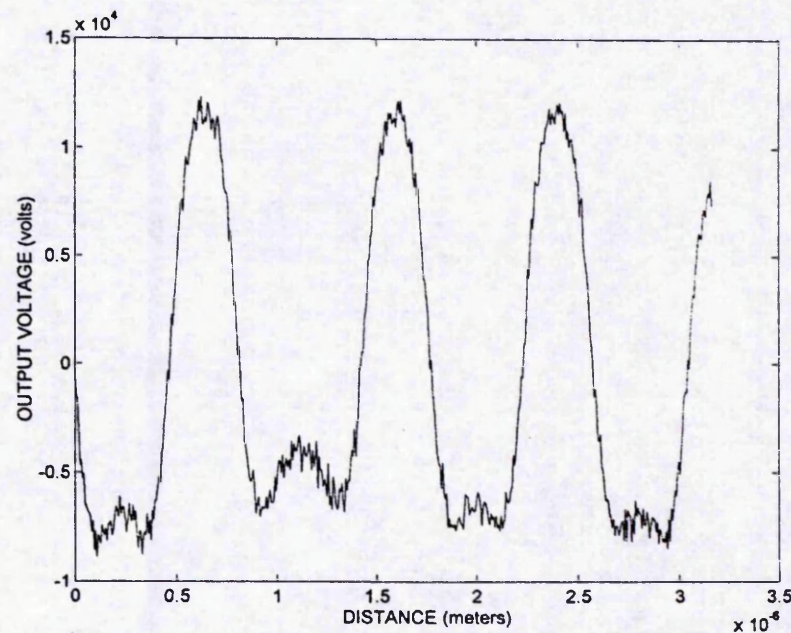


Fig. 12 A noisy waveform produced using the new sampled AWGN method.

5.2. Calculating BER's Using a Statistical Method

The classical statistical method is well documented [1], and determines the probability that the combination of ISI and noise will shift zero crossing points of the differentiated waveforms out of the timing window $2T_w$ that is used. The expression derived [1] is:

$$\text{BER} = \frac{1}{2M} \sum_{i=1}^M \left\{ \text{erfc} \left(\frac{T_w + t_{\text{isi}}(i)}{\sqrt{2} t_n(i)} \right) + \text{erfc} \left(\frac{T_w - t_{\text{isi}}(i)}{\sqrt{2} t_n(i)} \right) \right\} \quad (10)$$

Where

$t_{\text{isi}}(i)$ = the peakshift induced cross-over shift of the i^{th} bit

$t_n(i)$ = the noise induced cross-over shift of the i^{th} bit

M = the number of bits in the sequence under test

$$\text{erfc}(x) = \frac{1}{\sqrt{\pi}x} e^{-x^2}$$

This produces a mathematical (or numerical) prediction of average error rates, but is unable to provide error pattern information, or allow for detailed investigation of internal characteristics of the recording channel.

6. SIMULATION RESULTS

To illustrate the flexibility of the model, several examples are demonstrated for various uses of the model in system design and analysis.

In Figures 13 and 14, that are used to verify the characteristics of the simulation, an earlier tape system is used as listed in Table 2. Figure 13 shows the output voltage as a function of record current predicted by the model, and this agrees with previously published experimental work [2]. The analysis of error rates as a function of record current, is shown in Figure 14 and the curve has been verified by previously published models [2] and fits in with previously published experimental results [6]. On this basis this theory has been shown to be in accord with practical experience.

For the rest of this section, an advanced tape system is assumed, whose parameters are listed in Table 1, which are typical of advanced media such as metal evaporated tape or modern disk drives. Figure 15 shows the variation of error rate with packing density using the sampled AWGN (curve a) methods. The classical method (curve b) shows a good level of agreement with curve a, at the upper limits of the curves. The divergence in the lower limits of the curves is due to an approximation for the second differential of the signal in the statistical method [1]. Drop-outs are also added to the simulation in both the single track (curve c) and the multi-track (curve d) situations. Again, the trends at the upper limits of these curves are good, while at lower packing densities the curves tend to a limiting error rate, as is well known to be the case.

A significant advantage of the AWGN method is straightforward analysis of different internal characteristics of the channel, and in particular peakshift induced

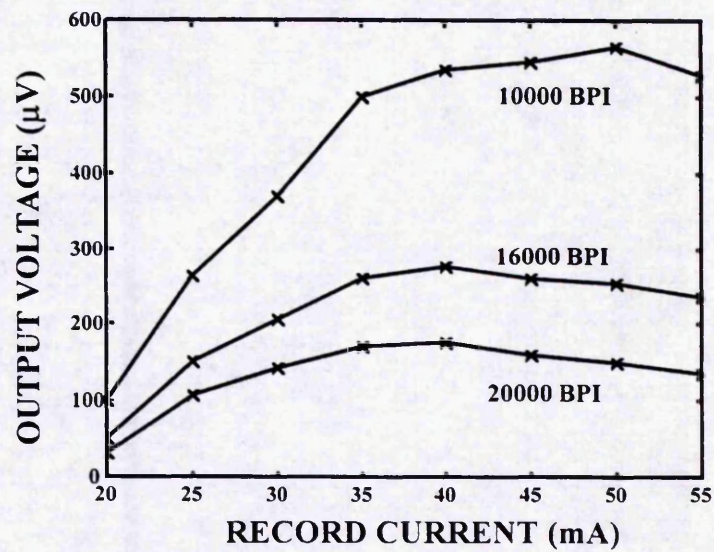


Fig. 13 Output voltage as a function of record current.

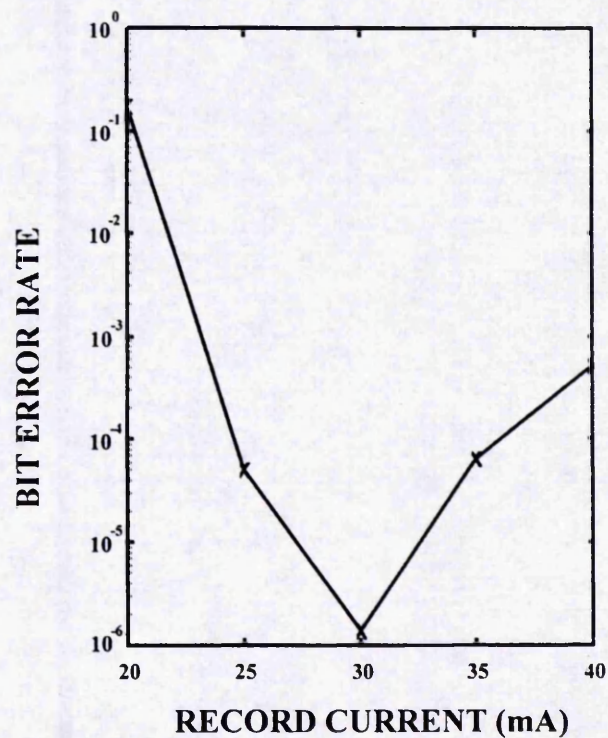


Fig. 14 Bit error rate as a function of record current.

Table 1 Parameters assumed for advanced disk and tape systems.

<i>Parameter</i>	<i>Value</i>
$(a_i + d)$	$8.4 * 10^{-8}$ metres
D	$2.5 * 10^{-8}$ metres
μ_o	$4 * 10^{-7}$
ω	$1.814286 * 10^{-3}$ metres
v	0.381 metres / sec
M_x	$3.857827 * 10^8$ Amps / metre
$\eta * n$	18
α	0
$2g$	$2 * 10^{-10}$ metres

For this advanced tape system, we have assumed that there is an extremely narrow gap in our computations.

Table 2 Parameters for a tape system used for earlier simulations.

<i>Parameter</i>	<i>Value</i>
$(a_i + d)$	$7.5 * 10^{-7}$ metres
D	$3.2 * 10^{-6}$ metres
μ_o	$4 * 10^{-7}$
ω	$1.814286 * 10^{-3}$ metres
v	0.381 metres / sec
M_x	$4.720641 * 10^4$ Amps / metre
(M_y / M_x)	0.3
$\eta * n$	18
α	0

The previous study used a narrow gap approximation (which includes perpendicular components) as presented in [2]. The (M_y / M_x) parameter controls the effect of the perpendicular components.

cross-over analysis. Figures 16, 17 and 18 show the displacement of cross-overs from the clock position as a percentage of the timing window. Figure 16 shows one cross-over going past the timing window and is an example of a single error due to noise. The detector has been assumed to detect and operate on the first cross-over of a timing window which may contain multiple cross-overs. Figure 17 shows a burst of error events corresponding to the passage of a small drop-out, and Figure 18 shows a display of errors corresponding to the passage of a drop-out burst that causes intense error activity. The poor signal to noise ratio and multiple cross-overs in timing windows cause the asymmetric behaviour shown in Figures 16, 17 and 18.

7. CONCLUSIONS

A comprehensive model of a multi-track digital magnetic recording channel has been presented. The new modelling technique is advantageous over previous models due to its flexibility and its ability to work with large data trains. This allows for a more realistic analysis as the internal mechanisms of the channel can be examined in

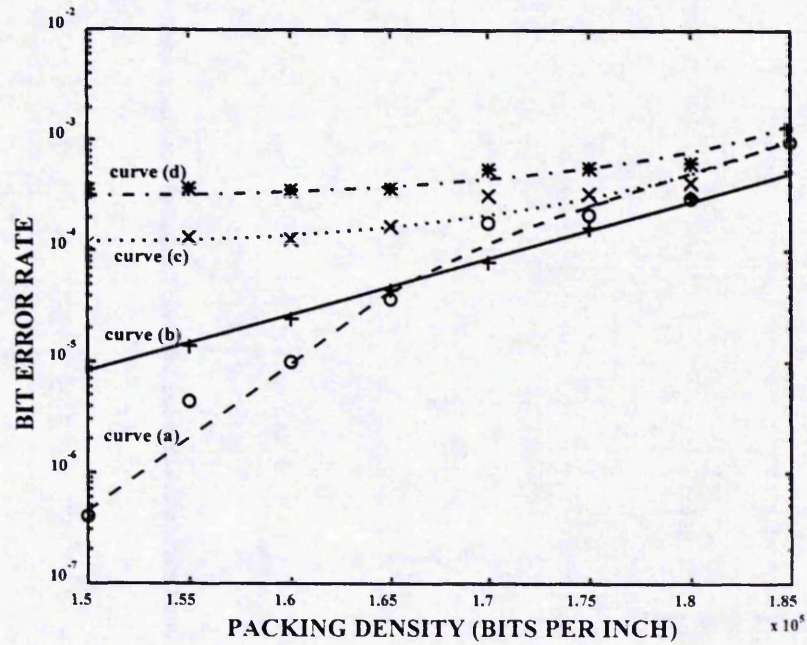


Fig. 15 Bit error rate as a function of packing density.

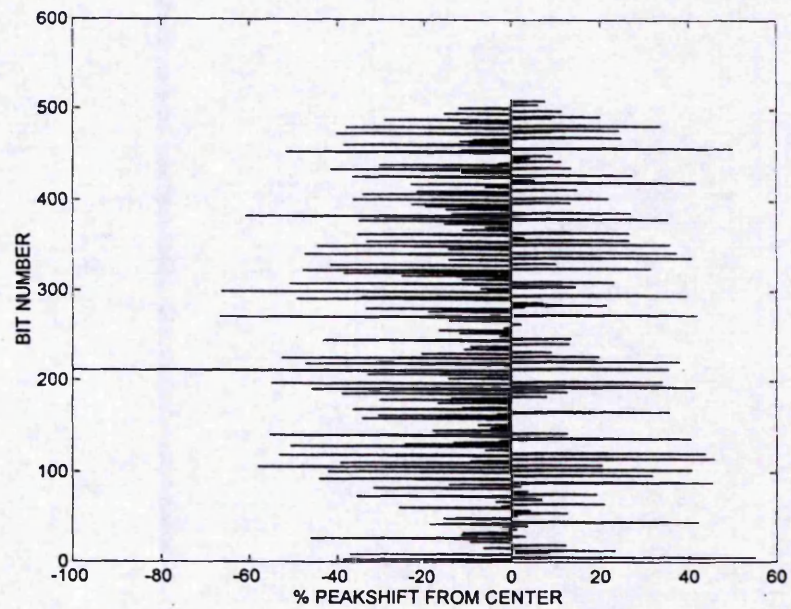


Fig. 16 % peakshift as a function of bit number from the center of a timing window for a noise induced error.

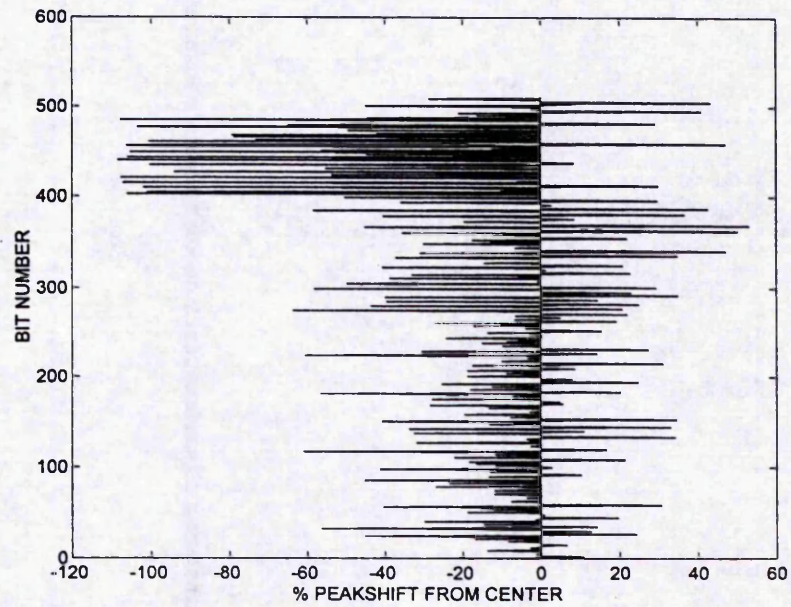


Fig. 17 % peakshift as a function of bit number from the center of a timing window for a small drop-out induced error.

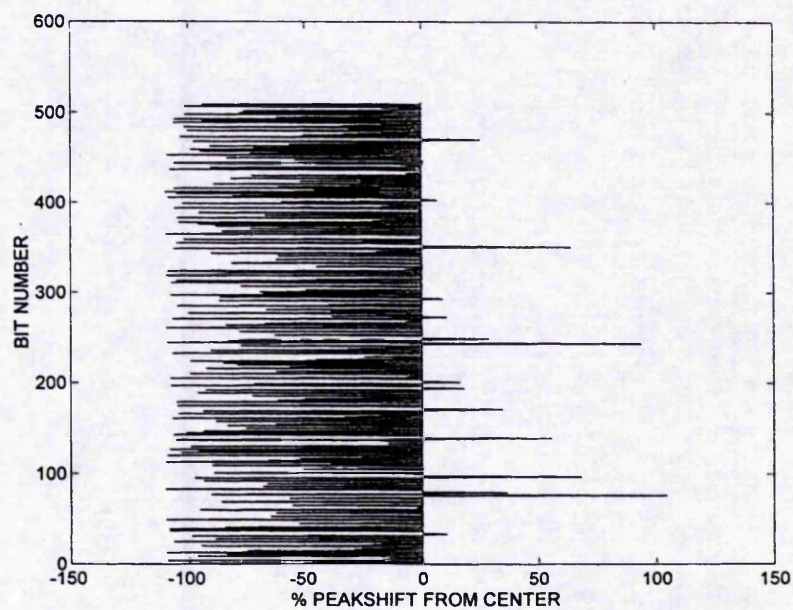


Fig. 18 % peakshift as a function of bit number from the center of a timing window for a large drop-out induced error.

greater detail than is currently possible by using traditional statistical peak detection models. This new model can also be used for many new system design applications, including, for example, error correction and detection where the masking of drop-outs in a noisy multi-track system is an important area of research.

ACKNOWLEDGEMENT

Thanks are due to EPSRC for a studentship for A. Tandon, and British Gas for support of B.K. Middleton.

References

- [1] K.R. Katz and T.E. Campbell: *IEEE Trans. Magn.*, MAG-15, 1050-1053 (1979).
- [2] J.K. Li Hui Hong, B.K. Middleton and J.J. Miles: *J. Magn. and Magn. Mat.*, 120, 206 (1993).
- [3] B.R. Baker: *IEEE Trans. Magn.*, MAG-13, 1196-1198 (1977).
- [4] M.K. Loze, B.K. Middleton and A. Ryley: *IERE Conf. Proc.* No. 67, 185 (1986).
- [5] B.K. Middleton and T. Jack-Kee: *Radio and Elec. Eng.*, 53, 393 (1983).
- [6] V.N. Hudson, M.K. Loze and B.K. Middleton: *IERE Conf. Proc.* No. 67, 177 (1986).
- [7] B.K. Middleton and T. Brown: *Radio and Elec. Eng.*, 50, 467 (1980).
- [8] B.K. Middleton, C.D. Wright, S.R. Cumpson and J.J. Miles: *IEEE Trans. Magn.*, MAG-31, 2365-2379 (1995).
- [9] J.C. Mallinson and C.W. Steele: *IEEE Trans. Magn.*, MAG-5, 886-890 (1969).
- [10] D.L.A. Tjaden: *IEEE Trans. Magn.*, MAG-9, 331-335 (1973).
- [11] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery: "Numerical Recipes in C". Cambridge University Press, (1992).
- [12] R.H. Georgi: "A Multi-Feature Detection System for Digital Magnetic Recording", PhD Thesis, Cranfield Institute of Technology, College of Aeronautics, (1992).
- [13] D.E. Knuth: "Art of Computer Programming", Addison-Wesley. (1981).

APPENDIX C

C.1 A Detailed Investigation of Block, Symbol and Bit Errors and Error Rates for a Wide Range of Interleaving Depths and Block Sizes

BLOCK SIZE: 64 INTERLEAVE DEPTH: NONE

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	155	1.586814e-2	1560	2.495393e-3	2013	4.025021e-4
3	80	8.190008e-3	1405	2.247453e-3	1854	3.707098e-4
5	68	6.961507e-3	1352	2.162674e-3	1794	3.587128e-4
7	59	6.040131e-3	1294	2.069897e-3	1727	3.453160e-4
9	49	5.016380e-3	1208	1.932330e-3	1638	3.275203e-4
10	43	4.402129e-3	1148	1.836353e-3	1575	3.149234e-4
15	32	3.276003e-3	1004	1.606009e-3	1423	2.845308e-4
20	25	2.559378e-3	881	1.409257e-3	1276	2.551380e-4
25	20	2.047502e-3	765	1.223702e-3	1134	2.267449e-4
30	14	1.433251e-3	593	9.485693e-4	916	1.831555e-4

BLOCK SIZE: 64 INTERLEAVE DEPTH: 5

BLOCK		SYMBOL		BIT	
Symbols corrected	number of errors	error rate	number of errors	error rate	error rate
0	211	2.159894e-2	1560	2.495138e-3	4.024609e-4
3	108	1.105538e-2	1353	2.164052e-3	3.600756e-4
5	76	7.779711e-3	1212	1.938530e-3	3.266871e-4
7	62	6.346607e-3	1124	1.797779e-3	3.042948e-4
9	47	4.811137e-3	999	1.597848e-3	2.737054e-4
10	41	4.196850e-3	939	1.501881e-3	2.609098e-4
15	23	2.354386e-3	705	1.127610e-3	2.067286e-4
20	19	1.944928e-3	632	1.010851e-3	1.885348e-4
25	12	1.228375e-3	474	7.581380e-4	1.421509e-4
30	12	1.228375e-3	474	7.581380e-4	1.421509e-4

BLOCK SIZE: 64 INTERLEAVE DEPTH: 10

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	286	2.927628e-2	1560	2.495138e-3	2013	4.024609e-4
3	107	1.095301e-2	1213	1.940129e-3	1622	3.242879e-4
5	64	6.551336e-3	1031	1.649030e-3	1412	2.823024e-4
7	51	5.220596e-3	948	1.516276e-3	1311	2.621094e-4
9	46	4.708773e-3	905	1.447500e-3	1264	2.527127e-4
10	36	3.685126e-3	805	1.287555e-3	1129	2.257220e-4
15	22	2.252022e-3	629	1.006052e-3	915	1.829368e-4
20	19	1.944928e-3	577	9.228811e-4	857	1.713408e-4
25	17	1.740199e-3	527	8.429087e-4	779	1.557462e-4
30	8	8.189170e-4	270	4.318508e-4	390	7.797305e-5

BLOCK SIZE: 64 INTERLEAVE DEPTH: 20

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	399	4.080172e-2	1560	2.492586e-3	2013	4.020494e-4
3	97	9.919215e-3	1045	1.669713e-3	1420	2.836116e-4
5	68	6.953676e-3	914	1.460400e-3	1258	2.512559e-4
7	56	5.726557e-3	837	1.337368e-3	1151	2.298851e-4
9	47	4.806217e-3	763	1.219130e-3	1069	2.135076e-4
10	44	4.499438e-3	733	1.171196e-3	1039	2.075158e-4
15	26	2.658759e-3	500	7.989058e-4	689	1.376115e-4
20	9	9.203395e-4	207	3.307470e-4	264	5.272778e-5
26	1	1.022599e-4	27	4.314091e-5	36	7.190152e-6
27	0	0	0	0	0	0

BLOCK SIZE: 64 INTERLEAVE DEPTH: 50

BLOCK			SYMBOL			BIT		
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors	error rate		
0	649	6.623125e-2	1560	2.487499e-3	2013	4.012288e-4		
2	166	1.694050e-2	977	1.577876e-3	1322	2.634995e-4		
3	130	1.326666e-2	869	1.385664e-3	1193	2.377873e-4		
5	71	7.245637e-3	613	9.774594e-4	856	1.706169e-4		
7	44	4.490254e-3	445	7.095750e-4	615	1.225811e-4		
10	11	1.122564e-3	154	2.455608e-4	207	4.125899e-5		
12	7	7.143586e-4	109	1.738060e-4	137	2.730668e-5		
15	3	3.061537e-4	53	8.451117e-5	70	1.395232e-5		
17	2	2.041025e-4	36	5.740382e-5	50	9.965940e-6		
18	0	0	0	0	0	0		

BLOCK SIZE: 64 INTERLEAVE DEPTH: 100

BLOCK		SYMBOL		BIT	
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors
0	759	7.745688e-2	1560	2.487499e-3	2013
2	175	1.785897e-2	882	1.406394e-3	1203
3	135	1.377692e-2	762	1.215047e-3	1054
4	93	9.490764e-3	594	9.471630e-4	819
5	64	6.531279e-3	449	7.159532e-4	605
6	32	3.265639e-3	257	4.097995e-4	327
8	8	8.164098e-4	79	1.259695e-4	105
10	3	3.061537e-4	34	5.421472e-5	43
11	1	1.020512e-4	12	1.913461e-5	15
12	0	0	0	0	0

BLOCK SIZE: 64 INTERLEAVE DEPTH: 200

BLOCK		SYMBOL		BIT	
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors
0	730	7.449740e-2	1560	2.487499e-3	2013
2	166	1.694050e-2	931	1.484526e-3	1300
3	134	1.367486e-2	835	1.331450e-3	1194
4	110	1.122564e-2	739	1.178373e-3	1071
5	85	8.674355e-3	614	9.790540e-4	900
6	52	5.306664e-3	416	6.633330e-4	607
8	10	1.020512e-3	98	1.562659e-4	124
10	2	2.041025e-4	23	3.667466e-5	27
11	1	1.020512e-4	12	1.913461e-5	15
12	0	0	0	0	0

BLOCK SIZE: 64 INTERLEAVE DEPTH: 500

BLOCK		SYMBOL		BIT	
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors
0	744	7.440744e-2	1560	2.437744e-3	2013
1	254	2.540254e-2	1070	1.672042e-3	1443
3	146	1.460146e-2	818	1.278253e-3	1094
4	100	1.000100e-2	634	9.907241e-4	849
5	67	6.700670e-3	469	7.328858e-4	614
6	28	2.800280e-3	235	3.672242e-4	289
8	9	9.000900e-4	88	1.375138e-4	101
10	2	2.000200e-4	23	3.594109e-5	27
11	1	1.000100e-4	12	1.875188e-5	15
12	0	0	0	0	0

BLOCK SIZE: 64 INTERLEAVE DEPTH: 1000

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	693	6.930693e-2	1560	2.437744e-3	2013	3.932034e-4
1	213	2.130213e-2	1080	1.687669e-3	1460	2.851848e-4
3	158	1.580158e-2	950	1.484523e-3	1311	2.560803e-4
4	123	1.230123e-2	810	1.265752e-3	1141	2.228738e-4
5	93	9.300930e-3	660	1.031353e-3	950	1.855654e-4
6	52	5.200520e-3	414	6.469397e-4	593	1.158319e-4
8	11	1.100100e-3	105	1.640789e-4	131	2.558850e-5
10	1	1.000100e-4	12	1.875188e-5	15	2.929980e-6
11	1	1.000100e-4	12	1.875188e-5	15	2.929980e-6
12	0	0	0	0	0	0

BLOCK SIZE: 128 INTERLEAVE DEPTH: NONE

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	122	2.495952e-2	1560	2.495393e-3	2013	4.025021e-4
3	61	1.248976e-2	1434	2.293842e-3	1887	3.773082e-4
5	48	9.828010e-3	1378	2.204264e-3	1823	3.645114e-4
7	42	8.599509e-3	1338	2.140279e-3	1773	3.545138e-4
10	37	7.575758e-3	1292	2.066697e-3	1723	3.445162e-4
25	14	2.866503e-3	878	1.404458e-3	1253	2.505391e-4
30	12	2.457002e-3	822	1.314880e-3	1197	2.393418e-4
40	10	2.047502e-3	759	1.214105e-3	1127	2.253452e-4
50	10	2.047502e-3	759	1.214105e-3	1127	2.253452e-4
60	7	1.433251e-3	591	9.453701e-4	915	1.829555e-4

BLOCK SIZE: 128 INTERLEAVE DEPTH: 5

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	157	3.214578e-2	1560	2.495393e-3	2013	4.025021e-4
3	80	1.638002e-2	1405	2.247453e-3	1854	3.707098e-4
5	65	1.330876e-2	1339	2.141879e-3	1776	3.551136e-4
7	56	1.146601e-2	1278	2.044303e-3	1704	3.407171e-4
10	41	8.394758e-3	1143	1.828355e-3	1550	3.099246e-4
25	13	2.661753e-3	677	1.082937e-3	970	1.939528e-4
30	10	2.047502e-3	593	9.485693e-4	879	1.757573e-4
40	10	2.047502e-3	593	9.485693e-4	879	1.757573e-4
50	10	2.047502e-3	593	9.485693e-4	879	1.757573e-4
60	4	8.190008e-4	250	3.999027e-4	349	6.978303e-5

BLOCK SIZE: 128 INTERLEAVE DEPTH: 10

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	206	4.213541e-2	1560	2.492841e-3	2013	4.020905e-4
3	105	2.147678e-2	1358	2.170050e-3	1804	3.603434e-4
5	74	1.513602e-2	1221	1.951128e-3	1645	3.285836e-4
7	55	1.124974e-2	1100	1.757773e-3	1482	2.960249e-4
10	36	7.363469e-3	930	1.486117e-3	1273	2.542778e-4
25	19	3.886275e-3	662	1.057859e-3	944	1.885611e-4
30	13	2.659030e-3	497	7.941936e-4	692	1.382248e-4
40	5	1.022704e-3	227	3.627403e-4	293	5.852584e-5
48	2	4.090816e-4	98	1.556016e-4	128	2.556760e-5
49	0	0	0	0	0	0

BLOCK SIZE: 128 INTERLEAVE DEPTH: 20

BLOCK			SYMBOL			BIT		
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors	error rate		
0	294	6.001225e-2	1560	2.487753e-3	2013	4.012697e-4		
3	115	2.347418e-2	1219	1.943955e-3	1644	3.277136e-4		
5	68	1.388038e-2	1022	1.629797e-3	1405	2.800715e-4		
7	53	1.081853e-2	925	1.475110e-3	1300	2.591409e-4		
9	47	9.593795e-3	873	1.392185e-3	1248	2.487753e-4		
10	44	8.981425e-3	843	1.344343e-3	1211	2.413997e-4		
15	36	7.348438e-3	739	1.178493e-3	1093	2.178777e-4		
20	15	3.061849e-3	356	5.677179e-4	537	1.070451e-4		
25	2	4.082466e-4	52	8.292509e-5	69	1.375440e-5		
26	0	0	0	0	0	0		

BLOCK SIZE: 128 INTERLEAVE DEPTH: 50

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	498	1.016534e-1	1560	2.487753e-3	2013	4.012697e-4
2	189	3.857930e-2	1102	1.757374e-3	1486	2.962180e-4
3	130	2.653603e-2	925	1.475110e-3	1257	2.505693e-4
5	82	1.673811e-2	711	1.133841e-3	985	1.963491e-4
7	48	9.797918e-3	486	7.750306e-4	667	1.329592e-4
9	28	5.715452e-3	316	5.039294e-4	413	8.232707e-5
10	18	3.674219e-3	216	3.444581e-4	289	5.760901e-5
12	4	8.164932e-4	55	8.770923e-5	72	1.435242e-5
14	1	2.041233e-4	15	2.392070e-5	19	3.787444e-6
15	0	0	0	0	0	0

BLOCK SIZE: 128 INTERLEAVE DEPTH: 100

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	676	1.379873e-1	1560	2.487753e-3	2013	4.012697e-4
2	218	4.449888e-2	976	1.556440e-3	1367	2.724966e-4
3	141	2.878138e-2	745	1.188061e-3	1071	2.134922e-4
4	79	1.612574e-2	497	7.925724e-4	710	1.415308e-4
5	45	9.185548e-3	327	5.214712e-4	483	9.628081e-5
6	27	5.511329e-3	219	3.492422e-4	304	6.059910e-5
8	9	1.837110e-3	88	1.403348e-4	116	2.312334e-5
10	2	4.082466e-4	23	3.667840e-5	31	6.179514e-6
11	1	2.041233e-4	12	1.913656e-5	17	3.388766e-6
12	0	0	0	0	0	0

BLOCK SIZE: 128 INTERLEAVE DEPTH: 200

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	784	1.568314e-1	1560	2.437988e-3	2013	3.932427e-4
1	346	6.921384e-2	1122	1.753476e-3	1521	2.971297e-4
2	248	4.960992e-2	926	1.447164e-3	1279	2.498547e-4
3	129	2.580516e-2	569	8.892403e-4	789	1.541324e-4
4	39	7.801560e-3	209	3.266278e-4	266	5.196352e-5
5	11	2.200440e-3	69	1.078341e-4	84	1.640953e-5
6	2	4.000800e-4	15	2.344219e-5	22	4.297735e-6
7	1	2.000400e-4	8	1.250250e-5	11	2.148867e-6
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 128 INTERLEAVE DEPTH: 500

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	781	1.562312e-1	1560	2.437988e-3	2013	3.932427e-4
1	355	7.101420e-2	1134	1.772229e-3	1530	2.988879e-4
2	247	4.940988e-2	918	1.434662e-3	1267	2.475104e-4
3	128	2.560512e-2	561	8.767378e-4	782	1.527649e-4
4	36	7.201440e-3	193	3.016228e-4	249	4.864254e-5
5	11	2.200440e-3	68	1.062713e-4	83	1.621418e-5
6	2	4.000800e-4	14	2.187938e-5	21	4.102383e-6
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 128 INTERLEAVE DEPTH: 1000

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	782	1.564313e-1	1560	2.437988e-3	2013	3.932427e-4
1	348	6.961392e-2	1126	1.759727e-3	1520	2.969344e-4
2	246	4.920984e-2	922	1.440913e-3	1267	2.475104e-4
3	133	2.660532e-2	583	9.111197e-4	800	1.562813e-4
4	38	7.601520e-3	203	3.172510e-4	256	5.001000e-5
5	10	2.000400e-3	63	9.845719e-5	78	1.523742e-5
6	2	4.000800e-4	15	2.344219e-5	22	4.297735e-6
7	1	2.000400e-4	8	1.250250e-5	11	2.148867e-6
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: NONE

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	111	3.409091e-2	1560	2.495393e-3	2013	4.025021e-4
3	51	1.566339e-2	1434	2.293842e-3	1885	3.769083e-4
5	41	1.259214e-2	1389	2.221860e-3	1836	3.671107e-4
7	37	1.136364e-2	1363	2.180270e-3	1802	3.603124e-4
10	32	9.828010e-3	1317	2.106688e-3	1753	3.505148e-4
25	15	4.606880e-3	1021	1.633203e-3	1403	2.805318e-4
50	9	2.764128e-3	817	1.306882e-3	1199	2.397417e-4
75	6	1.842752e-3	651	1.041347e-3	950	1.899538e-4
80	6	1.842752e-3	651	1.041347e-3	950	1.899538e-4
90	4	1.228501e-3	478	7.646140e-4	742	1.483639e-4

BLOCK SIZE: 192 INTERLEAVE DEPTH: 5

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	138	4.234428e-2	1560	2.493096e-3	2013	4.021316e-4
3	69	2.117214e-2	1419	2.267759e-3	1870	3.735649e-4
5	56	1.718319e-2	1362	2.176665e-3	1807	3.609795e-4
7	46	1.411476e-2	1298	2.074384e-3	1731	3.457972e-4
10	38	1.166002e-2	1225	1.957720e-3	1653	3.302154e-4
25	19	5.830009e-3	908	1.451110e-3	1271	2.539043e-4
50	7	2.147898e-3	448	7.159660e-4	714	1.426339e-4
80	1	3.068426e-4	90	1.438325e-4	150	2.996510e-5
89	1	3.068426e-4	90	1.438325e-4	150	2.996510e-5
90	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: 10

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	171	5.247008e-2	1560	2.493096e-3	2013	4.021316e-4
3	89	2.730899e-2	1399	2.235796e-3	1848	3.691700e-4
5	64	1.963793e-2	1289	2.060001e-3	1722	3.439993e-4
10	40	1.227370e-2	1104	1.764345e-3	1509	3.014489e-4
20	22	6.750537e-3	865	1.382390e-3	1247	2.491098e-4
25	19	5.830009e-3	795	1.270520e-3	1159	2.315303e-4
40	13	3.988954e-3	592	9.460980e-4	871	1.739973e-4
50	1	4.083229e-4	56	8.932217e-5	74	1.475411e-5
51	1	4.083229e-4	56	8.932217e-5	74	1.475411e-5
52	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: 20

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	249	7.640380e-2	1560	2.493096e-3	2013	4.021316e-4
3	118	3.620743e-2	1299	2.075982e-3	1740	3.475951e-4
5	78	2.393372e-2	1121	1.791513e-3	1531	3.058438e-4
7	57	1.749003e-2	985	1.574166e-3	1374	2.744803e-4
10	45	1.380792e-2	883	1.411156e-3	1265	2.527056e-4
15	34	1.043265e-2	745	1.190613e-3	1079	2.155489e-4
20	20	6.136852e-3	488	7.798916e-4	724	1.446315e-4
25	8	2.454741e-3	215	3.435998e-4	315	6.292670e-5
28	1	3.068426e-4	29	4.634602e-5	43	8.589994e-6
29	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: 50

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	427	1.294332e-1	1560	2.462868e-3	2013	3.972558e-4
2	195	5.910882e-2	1203	1.899250e-3	1632	3.220673e-4
3	141	4.274022e-2	1041	1.643490e-3	1446	2.853611e-4
5	96	2.909973e-2	845	1.334053e-3	1215	2.397744e-4
7	60	1.818733e-2	612	9.662019e-4	890	1.756372e-4
9	37	1.121552e-2	417	6.583434e-4	626	1.235381e-4
10	25	7.578054e-3	297	4.688921e-4	439	8.663452e-5
12	7	2.121855e-3	95	1.499823e-4	131	2.585222e-5
14	1	3.031222e-4	15	2.368142e-5	17	3.354868e-6
15	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: 100

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	615	1.864201e-1	1560	2.462868e-3	2013	3.972558e-4
1	391	1.185208e-1	1336	2.109225e-3	1782	3.516691e-4
2	217	6.577751e-2	988	1.559816e-3	1367	2.697708e-4
3	142	4.304335e-2	763	1.204595e-3	1079	2.129354e-4
4	99	3.000909e-2	591	9.330479e-4	852	1.681381e-4
5	57	1.727796e-2	381	6.015080e-4	548	1.081451e-4
6	23	6.971810e-3	177	2.794407e-4	250	4.933629e-5
7	12	3.637466e-3	100	1.578761e-4	136	2.683894e-5
8	4	1.212489e-3	36	5.683540e-5	45	8.880532e-6
9	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: 200

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	938	2.759635e-1	1560	2.390409e-3	2013	3.855684e-4
1	400	1.176817e-1	1022	1.566024e-3	1392	2.666225e-4
2	161	4.736687e-2	544	8.335785e-4	764	1.463359e-4
3	46	1.353339e-2	199	3.049304e-4	264	5.056634e-5
4	14	4.118858e-3	71	1.087943e-4	96	1.838776e-5
5	1	2.942042e-4	6	9.193881e-6	9	1.723853e-6
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: 500

	BLOCK		SYMBOL		BIT	
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	886	2.532152e-1	1560	2.322092e-3	2013	3.745490e-4
1	408	1.166047e-1	1082	1.610579e-3	1451	2.699804e-4
2	184	5.258645e-2	634	9.437220e-4	865	1.609463e-4
3	61	1.743355e-2	265	3.944579e-4	332	6.177360e-5
4	17	4.858531e-3	89	1.324783e-4	111	2.065322e-5
5	4	1.143184e-3	24	3.572449e-5	34	6.326212e-6
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 192 INTERLEAVE DEPTH: 1000

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	889	2.223056e-1	1560	2.031758e-3	2013	3.277186e-4
1	421	1.052763e-1	1092	1.422231e-3	1469	2.391548e-4
2	180	4.501125e-2	610	7.944695e-4	846	1.377297e-4
3	54	1.350338e-2	232	3.021589e-4	302	4.916594e-5
4	15	3.750938e-3	76	9.898308e-5	99	1.611731e-5
5	1	2.500625e-4	6	7.814454e-6	9	1.465210e-6
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: NONE

BLOCK		SYMBOL		BIT	
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors
0	102	4.176904e-2	1560	2.495393e-3	2013
3	46	1.883702e-2	1444	2.309838e-3	1897
5	41	1.678952e-2	1421	2.273047e-3	1870
7	35	1.433251e-2	1383	2.212262e-3	1822
10	30	1.228501e-2	1339	2.141879e-3	1777
25	14	5.733006e-3	1065	1.703586e-3	1453
50	6	2.457002e-3	762	1.218904e-3	1130
75	6	2.457002e-3	762	1.218904e-3	1130
100	5	2.047502e-3	662	1.058942e-3	961
125	2	8.190008e-4	324	5.182740e-4	506

BLOCK SIZE: 256 INTERLEAVE DEPTH: 5

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	124	5.073650e-2	1560	2.493351e-3	2013	4.021727e-4
3	60	2.454992e-2	1427	2.280777e-3	1880	3.756010e-4
5	54	2.209493e-2	1398	2.234426e-3	1847	3.690080e-4
7	47	1.923077e-2	1352	2.160904e-3	1790	3.576201e-4
10	35	1.432079e-2	1246	1.991484e-3	1669	3.334457e-4
25	20	8.183306e-3	1016	1.623875e-3	1397	2.791035e-4
50	8	3.273322e-3	610	9.749642e-4	856	1.710183e-4
75	4	1.636661e-3	333	5.322346e-4	454	9.070364e-5
94	1	4.091653e-4	95	1.518387e-4	120	2.397453e-5
95	0	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: 10

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	157	6.410780e-2	1560	2.488261e-3	2013	4.013516e-4
3	85	3.470804e-2	1411	2.250600e-3	1864	3.716440e-4
5	69	2.817477e-2	1340	2.137352e-3	1780	3.548961e-4
10	40	1.633320e-2	1106	1.764113e-3	1515	3.020605e-4
20	22	8.983258e-3	833	1.328667e-3	1208	2.408509e-4
25	20	8.166599e-3	786	1.253700e-3	1161	2.314800e-4
40	6	2.449980e-3	286	4.274704e-4	399	7.955256e-5
50	1	4.083299e-4	56	8.932217e-5	74	1.475411e-5
55	1	4.083299e-4	56	8.932217e-5	74	1.475411e-5
56	0	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: 20

BLOCK		SYMBOL		BIT	
Symbols corrected	number of errors	error rate	number of errors	error rate	number of errors
0	202	8.214721e-2	1560	2.478142e-3	2013
3	104	4.229362e-2	1364	2.166785e-3	1809
5	83	3.375356e-2	1272	2.020638e-3	1704
10	58	2.358682e-2	1083	1.720402e-3	1472
15	41	1.667344e-2	872	1.385218e-3	1247
20	22	8.946726e-3	530	8.419327e-4	754
25	8	3.253355e-3	226	3.590128e-4	279
30	1	4.066694e-4	34	5.401078e-5	42
33	1	4.066694e-4	34	5.401078e-5	42
34	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: 50

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	359	1.465904e-1	1560	2.488261e-3	2013	4.013516e-4
3	159	6.492446e-2	1180	1.882146e-3	1611	3.212009e-4
5	111	4.532462e-2	973	1.551973e-3	1376	2.743467e-4
7	74	3.021641e-2	735	1.172354e-3	1063	2.119408e-4
9	30	1.224990e-2	371	5.917594e-4	517	1.030794e-4
10	20	8.166599e-3	271	4.322555e-4	365	7.277364e-5
12	12	4.899959e-3	180	2.871070e-4	239	4.765178e-5
15	5	2.041650e-3	85	1.355783e-4	124	2.472310e-5
18	1	4.083299e-4	19	3.030574e-5	25	4.984496e-6
19	0	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: 100

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	528	2.112845e-1	1560	2.438475e-3	2013	3.933214e-4
2	249	9.963986e-2	1145	1.789778e-3	1551	3.030509e-4
3	177	7.082833e-2	929	1.452143e-3	1273	2.487323e-4
5	57	2.280912e-2	398	6.221238e-4	536	1.047294e-4
6	29	1.160464e-2	230	3.595188e-4	296	5.783563e-5
7	15	6.002401e-3	132	2.063325e-4	178	3.477954e-5
8	9	3.601441e-3	84	1.313025e-4	121	2.364227e-5
9	2	8.003201e-4	21	3.282563e-5	32	6.252501e-6
10	1	4.001601e-4	11	1.719438e-5	20	3.907813e-6
11	0	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: 200

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	822	3.162755e-1	1560	2.344652e-3	2013	3.781878e-4
1	429	1.650635e-1	1167	1.753980e-3	1549	2.910149e-4
2	186	7.156599e-2	681	1.023531e-3	915	1.719036e-4
3	78	3.001154e-2	357	5.365645e-4	489	9.186977e-5
4	26	1.000385e-2	149	2.239443e-4	191	3.588369e-5
5	12	4.617160e-3	79	1.187356e-4	107	2.010238e-5
6	5	1.923817e-3	37	5.561033e-5	53	9.957255e-6
7	2	7.695267e-4	16	2.404771e-5	25	4.696818e-6
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: 500

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	973	3.893557e-1	1560	2.438475e-3	2013	3.933214e-4
1	448	1.792717e-1	1035	1.617835e-3	1371	2.678806e-4
2	115	4.601841e-2	369	5.767932e-4	472	9.222439e-5
3	21	8.403361e-3	87	1.359919e-4	116	2.266532e-5
4	3	1.200480e-3	15	2.344688e-5	17	3.321641e-6
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

BLOCK SIZE: 256 INTERLEAVE DEPTH: 1000

Symbols corrected	BLOCK		SYMBOL		BIT	
	number of errors	error rate	number of errors	error rate	number of errors	error rate
0	957	3.191064e-1	1560	2.031927e-3	2013	3.277460e-4
1	442	1.473825e-1	1045	1.361131e-3	1384	2.253355e-4
2	124	4.134712e-2	409	5.327297e-4	510	8.303549e-5
3	29	9.669890e-3	124	1.615122e-4	161	2.621316e-5
4	6	2.000667e-3	32	4.168056e-5	38	6.186958e-6
5	2	6.668890e-4	12	1.563021e-5	16	2.605035e-6
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0

C.2 A Detailed Investigation of the Number of Symbols to be Corrected for a Wide Range of Interleaving Depths and Block Sizes

INTERLEAVE DEPTH: 10

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	1031	620	1083	637	1070	640
10	805	470	865	489	796	469
20	577	314	573	214	593	323
26	501	54	453	127	498	82
27	447	0	345	127	444	28
28	447	0	317	127	416	0
31	239	0	286	67	208	0
34	73	0	187	35	112	0
35	73	0	117	0	77	0
36	37	0	117	0	77	0
37	0	0	80	0	40	0
39	0	0	41	0	40	0
40	0	0	41	0	0	0
41	0	0	0	0	0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	914	622	1048	618	1065	694
10	733	452	861	511	689	440
20	207	116	510	246	255	120
24	52	25	441	88	31	53
25	27	0	416	88	31	28
26	27	0	390	88	31	28
27	0	0	336	88	31	28
28	0	0	308	60	31	0
29	0	0	308	60	31	0
30	0	0	308	0	31	0
31	0	0	277	0	0	0
34	0	0	147	0	0	0
38	0	0	39	0	0	0
39	0	0	0	0	0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	613	476	1091	623	643	407
12	109	55	759	453	52	55
13	96	16	720	440	0	29
14	68	16	664	412	0	15
15	53	16	634	367	0	0
16	53	0	602	351	0	0
17	36	0	602	317	0	0
18	0	0	584	299	0	0
23	0	0	547	151	0	0
28	0	0	356	127	0	0
34	0	0	262	35	0	0
35	0	0	192	0	0	0
40	0	0	41	0	0	0
41	0	0	0	0	0	0

INTERLEAVE DEPTH: 100

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
3	762	573	1291	708	602	455
6	257	346	1009	603	136	175
9	34	170	927	550	10	10
10	34	80	877	540	0	0
11	12	36	833	518	0	0
12	0	0	773	458	0	0
15	0	0	658	334	0	0
20	0	0	552	246	0	0
25	0	0	416	88	0	0
29	0	0	308	60	0	0
30	0	0	308	0	0	0
35	0	0	112	0	0	0
38	0	0	39	0	0	0
39	0	0	0	0	0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
2	931	700	1353	801	567	420
4	739	588	1166	674	89	112
5	614	483	1071	639	24	42
6	416	345	1029	609	0	0
8	98	178	977	550	0	0
11	12	36	877	499	0	0
12	0	0	817	463	0	0
20	0	0	586	338	0	0
30	0	0	405	168	0	0
34	0	0	373	38	0	0
37	0	0	160	38	0	0
38	0	0	122	0	0	0
42	0	0	43	0	0	0
43	0	0	0	0	0	0

INTERLEAVE DEPTH: 500

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
2	926	620	1354	807	46	90
3	818	551	1291	711	4	24
4	634	471	1171	667	0	0
8	88	251	960	559	0	0
11	12	103	867	518	0	0
12	0	79	807	458	0	0
15	0	51	691	334	0	0
18	0	19	590	284	0	0
19	0	0	552	246	0	0
24	0	0	308	88	0	0
29	0	0	308	60	0	0
30	0	0	308	0	0	0
38	0	0	39	0	0	0
39	0	0	0	0	0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 64

INTERLEAVING							
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both	
0	1560	1015	1560	1015	1560	1015	
1	1080	727	1523	891	388	310	
2	1010	701	1353	801	34	64	
3	950	674	1290	726	4	4	
4	810	606	1166	674	0	0	
8	105	232	977	550	0	0	
11	12	115	877	499	0	0	
12	0	115	817	463	0	0	
18	0	38	625	377	0	0	
19	0	0	606	358	0	0	
30	0	0	405	168	0	0	
37	0	0	160	38	0	0	
38	0	0	122	0	0	0	
42	0	0	43	0	0	0	
43	0	0	0	0	0	0	

INTERLEAVE DEPTH: 5000

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015			1560	1015
1	1079	729			375	309
2	1007	709			31	73
3	938	688			4	4
4	794	632			0	0
5	634	522			0	0
6	400	420			0	0
7	267	343			0	0
8	83	271			0	0
9	20	226			0	0
10	0	126			0	0
12	0	115			0	0
15	0	87			0	0
18	0	38			0	0
19	0	0			0	0

INTERLEAVE DEPTH: 10

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	1221	694	1196	702	1105	674
10	930	538	1013	543	950	553
20	711	434	677	385	709	354
31	466	33	520	101	430	64
32	402	33	520	69	398	0
33	369	0	520	36	365	0
35	266	0	380	36	330	0
36	266	0	344	0	330	0
43	144	0	156	0	132	0
44	144	0	156	0	0	0
48	98	0	156	0	0	0
49	0	0	107	0	0	0
55	0	0	56	0	0	0
56	0	0	0	0	0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	1022	616	1032	598	1050	638
10	843	518	794	513	824	503
15	739	287	729	383	692	307
20	356	42	604	110	444	21
21	356	0	499	68	297	0
22	312	0	411	46	275	0
23	174	0	296	0	206	0
25	52	0	174	0	108	0
26	0	0	148	0	56	0
27	0	0	148	0	29	0
28	0	0	64	0	29	0
29	0	0	35	0	0	0
34	0	0	35	0	0	0
35	0	0	0	0	0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
4	821	557	1160	652	814	516
7	486	238	920	501	491	273
8	406	174	824	461	403	201
11	139	86	700	440	99	67
14	15	49	647	307	32	30
15	0	34	602	232	17	0
16	0	18	570	168	17	0
17	0	18	553	134	0	0
18	0	0	499	62	0	0
21	0	0	320	22	0	0
22	0	0	298	0	0	0
27	0	0	178	0	0	0
34	0	0	35	0	0	0
35	0	0	0	0	0	0

INTERLEAVE DEPTH: 100

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
3	745	548	1285	709	779	522
5	327	265	1040	627	259	172
6	219	151	986	573	91	112
7	128	95	944	566	56	42
8	88	47	928	558	0	18
9	43	20	901	558	0	0
10	23	0	841	548	0	0
11	12	0	841	526	0	0
12	0	0	817	514	0	0
18	0	0	676	257	0	0
25	0	0	343	26	0	0
26	0	0	291	0	0	0
34	0	0	35	0	0	0
35	0	0	0	0	0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
3	569	443	1305	744	486	432
5	69	208	1097	683	12	52
6	15	130	1043	641	0	28
7	8	81	1008	613	0	0
8	0	57	992	565	0	0
9	0	30	974	538	0	0
10	0	0	934	528	0	0
15	0	0	770	438	0	0
20	0	0	664	203	0	0
25	0	0	292	26	0	0
26	0	0	240	0	0	0
30	0	0	70	0	0	0
34	0	0	70	0	0	0
35	0	0	0	0	0	0

INTERLEAVE DEPTH: 500

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1134	749	1523	891	482	338
2	918	639	1359	805	68	88
3	561	450	1296	742	8	16
4	193	310	1188	706	0	0
6	14	134	1040	647	0	0
7	0	92	1005	619	0	0
9	0	40	946	560	0	0
10	0	0	936	540	0	0
15	0	0	803	403	0	0
22	0	0	542	46	0	0
23	0	0	450	0	0	0
30	0	0	35	0	0	0
34	0	0	35	0	0	0
35	0	0	0	0	0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack - symbol	vertical stack - symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1126	740	1522	891	302	169
2	922	640	1362	811	12	9
3	583	445	1302	736	0	0
5	63	199	1089	676	0	0
7	8	81	993	631	0	0
8	0	57	985	591	0	0
9	0	30	967	573	0	0
10	0	0	947	553	0	0
15	0	0	824	439	0	0
20	0	0	665	204	0	0
25	0	0	362	26	0	0
26	0	0	310	0	0	0
30	0	0	31	0	0	0
31	0	0	0	0	0	0

INTERLEAVE DEPTH: 5000

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015			1560	1015
1	1129	754			358	187
2	939	660			36	21
3	585	465			0	0
4	209	317			0	0
5	49	217			0	0
6	7	139			0	0
7	0	90			0	0
8	0	66			0	0
9	0	30			0	0
10	0	0			0	0

INTERLEAVE DEPTH: 10

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	1289	752	1283	740	1119	709
10	1104	627	1034	637	1024	627
20	865	601	891	563	840	601
30	708	325	765	330	819	316
40	592	84	480	88	421	43
42	426	43	396	46	379	43
43	426	0	396	46	336	0
45	292	0	307	46	247	0
46	292	0	307	0	247	0
51	52	0	162	0	105	0
52	0	0	162	0	53	0
53	0	0	109	0	0	0
54	0	0	55	0	0	0
55	0	0	0	0	0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	1121	631	1130	650	1078	631
10	883	573	882	518	859	539
15	745	286	711	310	704	341
22	382	23	349	66	361	122
23	313	0	280	66	338	76
25	215	0	256	66	240	26
26	111	0	230	66	240	0
28	29	0	203	66	157	0
29	0	0	174	66	128	0
32	0	0	112	66	66	0
33	0	0	79	0	0	0
35	0	0	79	0	0	0
39	0	0	40	0	0	0
40	0	0	0	0	0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1453	860	1522	890	1425	845
3	1041	659	1276	684	1039	660
5	845	202	1026	605	861	499
10	297	109	783	435	224	145
13	43	28	623	244	50	15
14	15	0	609	188	36	15
15	0	0	579	143	36	0
17	0	0	445	77	19	0
18	0	0	335	41	19	0
19	0	0	258	41	0	0
20	0	0	199	21	0	0
21	0	0	157	0	0	0
22	0	0	69	0	0	0
23	0	0	0	0	0	0

INTERLEAVE DEPTH: 100

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1336	813	1522	891	1314	799
2	988	681	1358	785	958	647
3	763	510	1283	680	760	512
5	381	202	1029	611	426	294
7	100	32	904	566	162	46
8	36	0	880	542	18	22
9	0	0	844	506	0	22
10	0	0	804	446	0	22
11	0	0	771	391	0	0
15	0	0	559	143	0	0
20	0	0	178	21	0	0
21	0	0	136	0	0	0
24	0	0	25	0	0	0
25	0	0	0	0	0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1022	692	1522	892	1168	795
2	544	434	1358	790	864	641
3	199	170	1283	682	654	545
4	71	50	1167	638	490	421
5	6	0	1052	608	315	241
6	0	0	986	572	165	109
7	0	0	916	544	74	32
8	0	0	876	528	18	0
9	0	0	840	510	0	0
15	0	0	518	145	0	0
20	0	0	157	21	0	0
21	0	0	115	0	0	0
25	0	0	26	0	0	0
26	0	0	0	0	0	0

INTERLEAVE DEPTH: 500

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1082	711	1525	895	742	487
2	634	467	1361	809	168	141
3	265	260	1298	734	21	21
4	89	140	1194	690	5	5
5	24	70	1084	670	0	0
6	0	46	1048	622	0	0
7	0	25	999	587	0	0
8	0	9	975	571	0	0
9	0	0	939	535	0	0
15	0	0	584	159	0	0
20	0	0	223	21	0	0
21	0	0	181	0	0	0
23	0	0	24	0	0	0
24	0	0	0	0	0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1092	717	1523	892	703	467
2	610	491	1359	804	165	97
3	232	278	1293	717	12	4
4	76	138	1189	689	0	0
5	6	78	1079	649	0	0
6	0	54	1025	601	0	0
7	0	33	983	538	0	0
8	0	9	951	514	0	0
9	0	0	924	496	0	0
15	0	0	527	145	0	0
20	0	0	181	21	0	0
21	0	0	139	0	0	0
25	0	0	26	0	0	0
26	0	0	0	0	0	0

INTERLEAVE DEPTH: 5000

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015			1560	1015
1	1084	709			382	171
2	618	481			48	15
3	222	277			0	0
4	66	141			0	0
5	6	71			0	0
6	0	53			0	0
7	0	25			0	0
8	0	9			0	0
9	0	0			0	0

INTERLEAVE DEPTH: 10

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	1340	793	1332	808	1170	706
10	1106	626	1157	625	1011	578
20	833	578	834	558	786	525
30	786	270	757	277	761	307
37	503	38	520	38	514	41
38	427	0	482	0	476	41
41	227	0	281	0	275	0
43	56	0	154	0	233	0
48	56	0	110	0	49	0
49	56	0	61	0	0	0
55	56	0	61	0	0	0
56	0	0	61	0	0	0
60	0	0	61	0	0	0
61	0	0	0	0	0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
5	1272	739	1241	736	1138	685
10	1083	569	1040	539	988	554
20	530	46	630	181	664	286
24	226	25	474	25	402	81
25	226	0	349	0	377	56
26	148	0	297	0	273	56
28	93	0	159	0	218	29
29	64	0	101	0	131	0
31	34	0	40	0	131	0
33	34	0	40	0	34	0
34	0	0	40	0	0	0
36	0	0	40	0	0	0
39	0	0	40	0	0	0
40	0	0	0	0	0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1500	885	1526	891	1457	863
3	1180	679	1286	695	1070	649
5	973	498	1048	612	888	471
10	271	63	708	248	272	81
13	128	15	395	44	30	14
14	100	15	297	30	16	0
15	85	0	222	0	16	0
16	53	0	206	0	0	0
17	19	0	138	0	0	0
18	19	0	84	0	0	0
19	0	0	65	0	0	0
20	0	0	65	0	0	0
21	0	0	44	0	0	0
22	0	0	0	0	0	0

INTERLEAVE DEPTH: 100

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1417	843	1527	902	1396	826
3	929	547	1312	741	907	589
5	398	176	1115	605	491	305
7	132	28	1000	534	186	59
8	84	12	952	470	114	27
9	21	12	853	371	33	0
10	11	12	773	271	13	0
11	0	12	652	205	13	0
12	0	0	616	97	13	0
13	0	0	434	45	0	0
16	0	0	230	17	0	0
17	0	0	179	0	0	0
21	0	0	66	0	0	0
22	0	0	0	0	0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1167	734	1528	898	1247	833
3	357	302	1297	732	819	602
5	79	76	1074	591	517	379
7	16	8	975	476	188	141
8	0	0	927	436	84	45
9	0	0	837	328	21	0
10	0	0	717	248	11	0
11	0	0	585	171	0	0
14	0	0	295	45	0	0
15	0	0	235	0	0	0
17	0	0	119	0	0	0
20	0	0	65	0	0	0
22	0	0	23	0	0	0
23	0	0	0	0	0	0

INTERLEAVE DEPTH: 500

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1035	660	1529	901	784	510
2	369	340	1375	823	142	106
3	87	187	1306	739	16	16
4	15	83	1206	679	0	0
5	0	18	1096	624	0	0
6	0	0	1048	564	0	0
10	0	0	764	270	0	0
12	0	0	584	121	0	0
14	0	0	316	15	0	0
15	0	0	241	0	0	0
17	0	0	141	0	0	0
19	0	0	86	0	0	0
21	0	0	44	0	0	0
22	0	0	0	0	0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1560	1015	1560	1015	1560	1015
1	1045	666	1526	897	818	545
2	409	348	1366	811	190	81
3	124	174	1291	739	28	12
4	32	82	1171	667	0	0
5	12	12	1081	597	0	0
6	0	0	1045	525	0	0
8	0	0	941	436	0	0
11	0	0	602	171	0	0
14	0	0	312	45	0	0
15	0	0	252	0	0	0
17	0	0	119	0	0	0
20	0	0	65	0	0	0
22	0	0	23	0	0	0
23	0	0	0	0	0	0

C.3 A Detailed Investigation of Symbol Error Rates for a Wide Range of Interleaving Depths and Block Sizes

INTERLEAVE DEPTH: 10

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.495138e-3	1.623439e-3			2.495138e-3	1.623439e-3
5	1.649030e-3	9.916573e-4			1.711409e-3	1.023646e-3
10	1.287555e-3	7.517402e-4			1.273160e-3	7.501408e-4
20	9.228811e-4	5.022264e-4			9.484722e-4	5.166215e-4
26	8.013231e-4	8.637015e-5			7.965247e-4	1.311547e-4
27	7.149529e-4	0			7.101546e-4	4.478452e-5
28	7.149529e-4	0			6.653700e-4	0
31	3.822679e-4	0			3.326850e-4	0
34	1.167596e-4	0			1.791381e-4	0
35	1.167596e-4	0			1.231574e-4	0
36	5.917955e-5	0			1.231574e-4	0
37	0	0			6.397789e-5	0
39	0	0			6.397789e-5	0
40	0	0			0	0
41	0	0			0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.492586e-3	1.621779e-3			2.492586e-3	1.621779e-3
5	1.460400e-3	9.938388e-4			1.701669e-3	1.108881e-3
10	1.171196e-3	7.222109e-4			1.100892e-3	7.030371e-4
20	3.307470e-4	1.853461e-4			4.074420e-4	1.917374e-4
24	8.308621e-5	3.994529e-5			4.953216e-5	8.468402e-5
25	4.314091e-5	0			4.953216e-5	4.473873e-5
26	4.314091e-5	0			4.953216e-5	4.473873e-5
27	0	0			4.953216e-5	4.473873e-5
28	0	0			4.953216e-5	0
29	0	0			4.953216e-5	0
30	0	0			4.953216e-5	0
31	0	0			0	0
34	0	0			0	0
38	0	0			0	0
39	0	0			0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.487499e-3	1.618469e-3			2.487499e-3	1.618469e-3
5	9.774594e-4	7.590060e-4			1.025296e-3	6.489820e-4
12	1.738060e-4	8.770028e-5			8.291662e-5	8.770028e-5
13	1.530768e-4	2.551281e-5			0	4.624196e-5
14	1.084294e-4	2.551281e-5			0	2.391826e-5
15	8.451117e-5	2.551281e-5			0	0
16	8.451117e-5	0			0	0
17	5.740382e-5	0			0	0
18	0	0			0	0
23	0	0			0	0
28	0	0			0	0
34	0	0			0	0
35	0	0			0	0
40	0	0			0	0
41	0	0			0	0

INTERLEAVE DEPTH: 100

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.487499e-3	1.618469e-3			2.487499e-3	1.618469e-3
3	1.215047e-3	9.136774e-4			9.599194e-4	7.255205e-4
6	4.097995e-4	5.517145e-4			2.168589e-4	2.790463e-4
9	5.421472e-5	2.710736e-4			1.594550e-5	1.594550e-5
10	5.421472e-5	1.275640e-4			0	0
11	1.913461e-5	5.740382e-5			0	0
12	0	0			0	0
15	0	0			0	0
20	0	0			0	0
25	0	0			0	0
29	0	0			0	0
30	0	0			0	0
35	0	0			0	0
38	0	0			0	0
39	0	0			0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.487499e-3	1.618469e-3			2.487499e-3	1.618469e-3
2	1.484526e-3	1.116185e-3			9.041101e-4	6.697112e-4
4	1.178373e-3	9.375957e-4			1.419150e-4	1.785897e-4
5	9.790540e-4	7.701679e-4			3.826921e-5	6.697112e-5
6	6.633330e-4	5.501199e-4			0	0
8	1.562659e-4	2.838300e-4			0	0
11	1.913461e-5	5.740382e-5			0	0
12	0	0			0	0
20	0	0			0	0
30	0	0			0	0
34	0	0			0	0
37	0	0			0	0
38	0	0			0	0
42	0	0			0	0
43	0	0			0	0

INTERLEAVE DEPTH: 500

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.437744e-3	1.586096e-3			2.437744e-3	1.586096e-3
2	1.447020e-3	9.688469e-4			7.18219e-5	1.406391e-4
3	1.278253e-3	8.610236e-4			6.250625e-6	3.750375e-5
4	9.907241e-4	7.360111e-4			0	0
8	1.375138e-4	3.922267e-4			0	0
11	1.875188e-5	1.609536e-4			0	0
12	0	1.234498e-4			0	0
15	0	7.969547e-5			0	0
18	0	2.969047e-5			0	0
19	0	0			0	0
24	0	0			0	0
29	0	0			0	0
30	0	0			0	0
38	0	0			0	0
39	0	0			0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 64

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.437744e-3	1.586096e-3			2.437744e-3	1.586096e-3
1	1.687669e-3	1.136051e-3			6.063106e-4	4.844234e-4
2	1.578283e-3	1.095422e-3			5.313031e-5	1.000100e-4
3	1.484523e-3	1.053230e-3			6.250625e-6	6.250625e-6
4	1.265752e-3	9.469697e-4			0	0
8	1.640789e-4	3.625363e-4			0	0
11	1.875188e-5	1.797055e-4			0	0
12	0	1.797055e-4			0	0
18	0	5.938094e-5			0	0
19	0	0			0	0
30	0	0			0	0
37	0	0			0	0
38	0	0			0	0
42	0	0			0	0
43	0	0			0	0

INTERLEAVE DEPTH: 5000

BLOCK SIZE: 64

INTERLEAVING							
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both	
0	2.437744e-3	1.586096e-3			2.437744e-3	1.586096e-3	
1	1.686106e-3	1.139176e-3			5.859961e-4	4.828608e-4	
2	1.573595e-3	1.107923e-3			4.844234e-5	1.140739e-4	
3	1.465772e-3	1.075108e-3			6.250250e-6	6.250250e-6	
4	1.240749e-3	9.875988e-4			0	0	
5	9.907241e-4	8.157066e-4			0	0	
6	6.250250e-4	6.563156e-4			0	0	
7	4.172292e-4	5.359911e-4			0	0	
8	1.297005e-4	4.234798e-4			0	0	
9	3.125313e-5	3.531603e-4			0	0	
10	0	1.968947e-4			0	0	
12	0	1.797055e-4			0	0	
15	0	1.359511e-4			0	0	
18	0	5.938094e-5			0	0	
19	0	0			0	0	

INTERLEAVE DEPTH: 10

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.492841e-3	1.621945e-3			2.492841e-3	1.621945e-3
5	1.951128e-3	1.108995e-3			1.765762e-3	1.077035e-3
10	1.486117e-3	8.597106e-4			1.518076e-3	8.836802e-4
20	1.136160e-3	6.935212e-4			1.132964e-3	5.656832e-4
31	7.446564e-4	5.273318e-5			6.871293e-4	1.022704e-4
32	6.423860e-4	5.273318e-5			6.359941e-4	0
33	5.896528e-4	0			5.832609e-4	0
35	4.250614e-4	0			5.273318e-4	0
36	4.250614e-4	0			5.273318e-4	0
43	2.301084e-4	0			2.109327e-4	0
44	2.301084e-4	0			0	0
48	1.566016e-4	0			0	0
49	0	0			0	0
55	0	0			0	0
56	0	0			0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.487753e-3	1.618634e-3			2.487753e-3	1.618634e-3
5	1.629797e-3	9.823433e-4			1.674449e-3	1.017427e-3
10	1.344343e-3	8.260614e-4			1.314044e-3	8.021407e-4
15	1.178493e-3	4.576827e-4			1.103542e-3	4.895770e-4
20	5.677179e-4	6.697795e-5			7.080527e-4	3.348898e-5
21	5.677179e-4	0			4.736298e-4	0
22	4.975505e-4	0			4.385461e-4	0
23	2.774801e-4	0			3.285109e-4	0
25	8.292509e-5	0			1.722290e-4	0
26	0	0			8.930394e-5	0
27	0	0			4.624668e-5	0
28	0	0			4.624668e-5	0
29	0	0			0	0
34	0	0			0	0
35	0	0			0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 128

INTERLEAVING							
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both	
0	2.487753e-3	1.618634e-3			2.487753e-3	1.618634e-3	
4	1.309260e-3	8.882553e-4			8.276562e-4	8.228720e-4	
7	7.750306e-4	3.795417e-4			7.830042e-4	4.353567e-4	
8	6.474536e-4	2.774801e-4			6.426694e-4	3.205374e-4	
11	2.216651e-4	1.371453e-4			1.578766e-4	1.068458e-4	
14	2.392070e-5	7.814095e-5			5.103082e-5	4.784140e-5	
15	0	5.422025e-5			2.711012e-5	0	
16	0	2.870484e-5			2.711012e-5	0	
17	0	2.870484e-5			0	0	
18	0	0			0	0	
21	0	0			0	0	
22	0	0			0	0	
27	0	0			0	0	
34	0	0			0	0	
35	0	0			0	0	

INTERLEAVE DEPTH: 100

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.487753e-3	1.618634e-3			2.487753e-3	1.618634e-3
3	1.188061e-3	8.739028e-4			1.242282e-3	8.324403e-4
5	5.214712e-4	4.225990e-4			4.130307e-4	2.742907e-4
6	3.492422e-4	2.408017e-4			1.451189e-4	1.786079e-4
7	2.041233e-4	1.514978e-4			8.930394e-5	6.697795e-5
8	1.403348e-4	7.495152e-5			0	2.870484e-5
9	6.857267e-5	3.189426e-5			0	0
10	3.667840e-5	0			0	0
11	1.913656e-5	0			0	0
12	0	0			0	0
18	0	0			0	0
25	0	0			0	0
26	0	0			0	0
34	0	0			0	0
35	0	0			0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.437988e-3	1.586255e-3			2.437988e-3	1.586255e-3
3	8.892403e-4	6.923260e-4			7.595269e-4	6.751350e-4
5	1.078341e-4	3.250650e-4			1.875375e-5	8.126625e-5
6	2.344219e-5	2.031656e-4			0	4.375875e-5
7	1.250250e-5	1.265878e-4			0	0
8	0	8.908032e-5			0	0
9	0	4.688438e-5			0	0
10	0	0			0	0
15	0	0			0	0
20	0	0			0	0
25	0	0			0	0
26	0	0			0	0
30	0	0			0	0
34	0	0			0	0
35	0	0			0	0

INTERLEAVE DEPTH: 500

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.437988e-3	1.586255e-3			2.437988e-3	1.586255e-3
1	1.772229e-3	1.170547e-3			7.532757e-4	5.282306e-4
2	1.434662e-3	9.986372e-4			1.062713e-4	1.375275e-4
3	8.767378e-4	7.032657e-4			1.250250e-5	2.500500e-5
4	3.016228e-4	4.844719e-4			0	0
6	2.187938e-5	2.094169e-4			0	0
7	0	1.437788e-4			0	0
9	0	6.251250e-5			0	0
10	0	0			0	0
15	0	0			0	0
22	0	0			0	0
23	0	0			0	0
30	0	0			0	0
34	0	0			0	0
35	0	0			0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.437988e-3	1.568255e-3			2.437988e-3	1.568255e-3
1	1.759727e-3	1.156481e-3			4.719694e-4	2.641153e-4
2	1.440913e-3	1.000200e-3			1.875375e-5	1.406531e-5
3	9.111197e-4	6.954516e-4			0	0
5	9.845719e-5	3.109997e-4			0	0
7	1.250250e-5	1.265878e-4			0	0
8	0	8.908032e-5			0	0
9	0	4.688438e-5			0	0
10	0	0			0	0
15	0	0			0	0
20	0	0			0	0
25	0	0			0	0
26	0	0			0	0
30	0	0			0	0
31	0	0			0	0

INTERLEAVE DEPTH: 5000

BLOCK SIZE: 128

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.437988e-3	1.586225e-3			2.437988e-3	1.586225e-3
1	1.764415e-3	1.178361e-3			5.594869e-4	2.922459e-4
2	1.467481e-3	1.031456e-3			5.626125e-5	3.281906e-5
3	9.142453e-4	7.267078e-4			0	0
4	3.266378e-4	4.954116e-4			0	0
5	7.657782e-5	3.391303e-4			0	0
6	1.093969e-5	2.172309e-4			0	0
7	0	1.406531e-4			0	0
8	0	1.031456e-4			0	0
9	0	4.688438e-5			0	0
10	0	0			0	0

INTERLEAVE DEPTH: 10

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.493096e-3	1.622111e-3			2.493096e-3	1.622111e-3
5	2.060001e-3	1.201800e-3			1.788317e-3	1.133080e-3
10	1.764345e-3	1.002033e-3			1.636494e-3	1.002033e-3
20	1.382390e-3	9.604812e-4			1.342436e-3	9.604812e-4
30	1.131482e-3	5.193950e-4			1.308875e-3	5.050118e-4
40	9.460980e-4	1.342436e-4			6.728163e-4	6.871995e-5
42	6.808070e-4	6.871995e-5			6.056945e-4	6.871995e-5
43	6.808070e-4	0			5.369745e-4	0
45	4.666564e-4	0			3.947402e-4	0
46	4.666564e-4	0			3.947402e-4	0
51	8.310320e-5	0			1.678045e-4	0
52	0	0			8.470134e-5	0
53	0	0			0	0
54	0	0			0	0
55	0	0			0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.493096e-3	1.622111e-3			2.493096e-3	1.622111e-3
5	1.791513e-3	1.008425e-3			1.722793e-3	1.008425e-3
10	1.411156e-3	9.157334e-4			1.372801e-3	8.613966e-4
15	1.190613e-3	4.570676e-4			1.125089e-3	5.449652e-4
22	6.104889e-4	3.675719e-5			5.769280e-4	1.949729e-4
23	5.002173e-4	0			5.401708e-4	1.214585e-4
25	3.435998e-4	0			3.835532e-4	4.155160e-5
26	1.773934e-4	0			3.835532e-4	0
28	4.634602e-5	0			2.509077e-4	0
29	0	0			2.045617e-4	0
32	0	0			1.054771e-4	0
33	0	0			0	0
35	0	0			0	0
39	0	0			0	0
40	0	0			0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.462868e-3	1.602443e-3			2.462868e-3	1.602443e-3
1	2.293940e-3	1.357735e-3			2.249735e-3	1.334053e-3
3	1.643490e-3	1.040404e-3			1.640333e-3	1.041982e-3
5	1.334053e-3	7.751718e-4			1.359313e-3	7.878019e-4
10	4.688921e-4	1.720850e-4			3.536425e-4	2.289204e-4
13	6.788673e-5	4.420531e-5			7.893806e-5	2.368142e-5
14	2.368142e-5	0			5.683540e-5	2.368142e-5
15	0	0			5.683540e-5	0
17	0	0			5.683540e-5	0
18	0	0			5.683540e-5	0
19	0	0			0	0
20	0	0			0	0
21	0	0			0	0
22	0	0			0	0
23	0	0			0	0

INTERLEAVE DEPTH: 100

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.462868e-3	1.602443e-3			2.462868e-3	1.602443e-3
1	2.109225e-3	1.283533e-3			2.074492e-3	1.261430e-3
2	1.559816e-3	1.075136e-3			1.512453e-3	1.021459e-3
3	1.204595e-3	8.051682e-4			1.199859e-3	8.083258e-4
5	6.015080e-4	3.189098e-4			6.725523e-4	4.641558e-4
7	1.578761e-4	5.052036e-5			2.557593e-4	7.262302e-5
8	5.683540e-5	0			2.841770e-5	3.473275e-5
9	0	0			0	3.473275e-5
10	0	0			0	3.473275e-5
11	0	0			0	0
15	0	0			0	0
20	0	0			0	0
21	0	0			0	0
24	0	0			0	0
25	0	0			0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 192

INTERLEAVING							
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both	
0	2.390409e-3	1.555298e-3			2.390409e-3	1.555298e-3	
1	1.566024e-3	1.060361e-3			1.789742e-3	1.218189e-3	
2	8.335785e-4	6.650240e-4			1.323919e-3	9.822129e-4	
3	3.049304e-4	2.604933e-4			1.002133e-3	8.351108e-4	
4	1.087943e-4	7.661567e-5			7.508336e-4	6.451040e-4	
5	9.193881e-6	0			4.826787e-4	3.692875e-4	
6	0	0			2.528317e-4	1.670222e-4	
7	0	0			1.133912e-4	4.903403e-5	
8	0	0			2.758164e-5	0	
9	0	0			0	0	
15	0	0			0	0	
20	0	0			0	0	
21	0	0			0	0	
25	0	0			0	0	
26	0	0			0	0	

INTERLEAVE DEPTH: 500

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.322092e-3	1.510848e-3			2.322092e-3	1.510848e-3
1	1.610579e-3	1.058338e-3			1.104482e-3	7.249095e-4
2	9.437220e-4	6.591391e-4			2.500714e-4	2.098814e-4
3	3.944579e-4	3.870153e-4			3.125893e-5	3.125893e-5
4	1.324783e-4	2.083929e-4			7.442603e-6	7.442603e-6
5	3.572449e-5	1.041964e-4			0	0
6	0	6.847945e-5			0	0
7	0	3.721301e-5			0	0
8	0	1.339668e-5			0	0
9	0	0			0	0
15	0	0			0	0
20	0	0			0	0
21	0	0			0	0
23	0	0			0	0
24	0	0			0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.031758e-3	1.321945e-3			2.031758e-3	1.321945e-3
1	1.422231e-3	9.338272e-4			9.155935e-4	6.082250e-4
2	7.944695e-4	6.394828e-4			2.148975e-4	1.263337e-4
3	3.021589e-4	3.620697e-4			1.562891e-5	5.209636e-5
4	9.898308e-5	1.797324e-4			0	0
5	7.814454e-6	1.015879e-4			0	0
6	0	7.033008e-5			0	0
7	0	4.297949e-5			0	0
8	0	1.172168e-5			0	0
9	0	0			0	0
15	0	0			0	0
20	0	0			0	0
21	0	0			0	0
25	0	0			0	0
26	0	0			0	0

INTERLEAVE DEPTH: 5000

BLOCK SIZE: 192

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	1.625325e-3	1.057503e-3			1.625325e-3	1.057503e-3
1	1.129393e-3	7.386894e-4			3.979963e-4	1.781606e-4
2	6.438788e-4	5.011419e-4			5.001000e-5	1.562813e-5
3	2.312963e-4	2.885994e-4			0	0
4	6.876375e-5	1.469044e-4			0	0
5	6.251250e-6	7.397313e-5			0	0
6	0	5.521938e-5			0	0
7	0	2.604688e-5			0	0
8	0	9.376875e-6			0	0
9	0	0			0	0

INTERLEAVE DEPTH: 10

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.488261e-3	1.618964e-3			2.488261e-3	1.618964e-3
5	2.137352e-3	1.264866e-3			1.866195e-3	1.126097e-3
10	1.764113e-3	9.984943e-4			1.612584e-3	9.219324e-4
20	1.328667e-3	9.219324e-4			1.253700e-3	8.373954e-4
30	1.253700e-3	4.306605e-4			1.213825e-3	4.896769e-4
37	8.023045e-4	6.061147e-5			8.198499e-4	6.539659e-5
38	6.810816e-4	0			7.592385e-4	6.539659e-5
41	3.620738e-4	0			4.386357e-4	0
43	8.932217e-5	0			3.716440e-4	0
48	8.932217e-5	0			7.815690e-5	0
49	8.932217e-5	0			0	0
55	8.932217e-5	0			0	0
56	0	0			0	0
60	0	0			0	0
61	0	0			0	0

INTERLEAVE DEPTH: 20

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.478142e-3	1.612381e-3			2.478142e-3	1.612381e-3
5	2.020638e-3	1.173940e-3			1.807772e-3	1.088158e-3
10	1.720402e-3	9.038862e-4			1.569490e-3	8.800580e-4
20	8.419327e-4	7.307340e-5			1.054799e-3	4.543259e-4
24	3.590128e-4	3.971381e-5			6.385980e-4	1.286727e-4
25	3.590128e-4	0			5.988842e-4	8.895893e-5
26	2.351057e-4	0			4.336748e-4	8.895893e-5
28	1.477354e-4	0			3.463044e-4	4.606802e-5
29	1.016673e-4	0			2.081003e-4	0
31	5.401078e-5	0			2.081003e-4	0
33	5.401078e-5	0			5.401078e-5	0
34	0	0			0	0
36	0	0			0	0
39	0	0			0	0
40	0	0			0	0

INTERLEAVE DEPTH: 50

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.488261e-3	1.618964e-3			2.488261e-3	1.618964e-3
1	2.392558e-3	1.411609e-3			2.323972e-3	1.376518e-3
3	1.882146e-3	1.083031e-3			1.706692e-3	1.035180e-3
5	1.551973e-3	7.943293e-4			1.416394e-3	7.512633e-4
10	4.322555e-4	1.004874e-4			4.338506e-4	1.291981e-4
13	2.041650e-4	2.392558e-5			4.785116e-5	2.233054e-5
14	1.595039e-4	2.392558e-5			2.552062e-5	0
15	1.355783e-4	0			2.552062e-5	0
16	8.453706e-5	0			0	0
17	3.030574e-5	0			0	0
18	3.030574e-5	0			0	0
19	0	0			0	0
20	0	0			0	0
21	0	0			0	0
22	0	0			0	0

INTERLEAVE DEPTH: 100

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.438475e-3	1.586572e-3			2.438475e-3	1.586572e-3
1	2.214948e-3	1.317715e-3			2.182123e-3	1.291141e-3
3	1.452143e-3	8.550295e-4			1.417755e-3	9.206808e-4
5	6.221238e-4	2.751100e-4			7.674945e-4	4.767532e-4
7	2.063325e-4	4.376751e-5			2.907413e-4	9.222439e-5
8	1.313025e-4	1.875750e-5			1.781963e-4	4.220438e-5
9	3.282563e-5	1.875750e-5			5.158313e-5	0
10	1.719438e-5	1.875750e-5			2.032063e-5	0
11	0	1.875750e-5			2.032063e-5	0
12	0	0			2.032063e-5	0
13	0	0			0	0
16	0	0			0	0
17	0	0			0	0
21	0	0			0	0
22	0	0			0	0

INTERLEAVE DEPTH: 200

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.344652e-3	1.525527e-3			2.344652e-3	1.525527e-3
1	1.753980e-3	1.103189e-3			1.874218e-3	1.251984e-3
3	5.365645e-4	4.539005e-4			1.230942e-3	9.047951e-4
5	1.187356e-4	1.142266e-4			7.770417e-4	5.696301e-4
7	2.404771e-5	1.202386e-5			2.825606e-4	2.119205e-4
8	0	0			1.262505e-4	6.763419e-5
9	0	0			3.156262e-5	0
10	0	0			1.653280e-5	0
11	0	0			0	0
14	0	0			0	0
15	0	0			0	0
17	0	0			0	0
20	0	0			0	0
22	0	0			0	0
23	0	0			0	0

INTERLEAVE DEPTH: 500

BLOCK SIZE: 256

INTERLEAVING						
Symbols corrected	horizontal stack-symbol	vertical stack-symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both
0	2.438475e-3	1.586572e-3			2.438475e-3	1.586572e-3
1	1.617835e-3	1.031663e-3			1.225490e-3	7.971939e-4
2	5.767932e-4	5.314626e-4			2.219638e-4	1.656913e-4
3	1.359919e-4	2.923044e-4			2.501000e-5	2.501000e-5
4	2.344688e-5	1.297394e-4			0	0
5	0	2.813625e-5			0	0
6	0	0			0	0
10	0	0			0	0
12	0	0			0	0
14	0	0			0	0
15	0	0			0	0
17	0	0			0	0
19	0	0			0	0
21	0	0			0	0
22	0	0			0	0

INTERLEAVE DEPTH: 1000

BLOCK SIZE: 256

INTERLEAVING							
Symbols corrected	horizontal stack -symbol	vertical stack -symbol	horizontal stack - block	vertical stack - block	horizontal stack - both	vertical stack - both	
0	2.031927e-3	1.322055e-3			2.031927e-3	1.322055e-3	
1	1.361131e-3	8.674767e-4			1.065459e-3	7.098720e-4	
2	5.327297e-4	4.532761e-4			2.474783e-4	1.055039e-4	
3	1.615122e-4	2.266380e-4			3.647049e-5	1.563021e-5	
4	4.168056e-5	1.068064e-4			0	0	
5	1.563021e-5	1.563021e-5			0	0	
6	0	0			0	0	
8	0	0			0	0	
11	0	0			0	0	
14	0	0			0	0	
15	0	0			0	0	
17	0	0			0	0	
20	0	0			0	0	
22	0	0			0	0	
23	0	0			0	0	

