Title
Coding and detection for 2-dimensional channels

Permalink
https://escholarship.org/uc/item/9531n4vb

Author
Demirkan, İsmail

Publication Date
2006

Peer reviewed|Thesis/dissertation
Coding and Detection for 2-Dimensional Channels

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Communications Theory and Systems)

by

İsmail Demirkan

Committee in charge:

Professor Jack K. Wolf, Chair
Professor William S. Hodgkiss, Jr
Professor Laurence B. Milstein
Professor Alon Orlitsky
Professor Paul H. Siegel

2006
The dissertation of İsmail Demirkan is approved, and it is acceptable in quality and form for publication on microfilm:

W. D. Nejati
Paul H. Siegel
J. B. Mitani
Alan Arlitt
Jack Kil Wolf

Chair

University of California, San Diego

2006
To my family and Zeynep.
CONTENTS

Signature Page .................................................. iii
Contents .......................................................... v
List of Figures ..................................................... viii
List of Tables ...................................................... xi
Acknowledgements ................................................ xiii
Vita and Publications ............................................ xv
Abstract of the Dissertation .................................. xvi

Chapter 1 Introduction ........................................... 1
1.1 Communication Systems ....................................... 1
1.2 Two-dimensional Storage Systems ............................ 2
1.3 Channel Modeling and Detection ............................. 3
1.4 Two-dimensional Constraints and Modulation Codes ........ 3
1.5 Information Theory and Error Correcting Codes ............ 4
1.6 Outline of Dissertation ...................................... 5

Chapter 2 Characterization and Detection on 2-D Channels ....... 7
2.1 Introduction .................................................... 7
2.2 Channel Characterization ..................................... 10
   2.2.1 Linear Channel Models .................................. 10
   2.2.2 Weighted Channel Models ................................. 12
2.3 Threshold Detection .......................................... 15
   2.3.1 Threshold Detection for Data Dependent Gaussian Noise . 17
   2.3.2 Threshold Detection with Precoding ...................... 19
2.4 The IMS Algorithm with SISO Detectors ..................... 21
   2.4.1 The IMS Algorithm with the log-MAP Detector .......... 23
   2.4.2 The IMS Algorithm with the max-log-MAP Detector .... 24
   2.4.3 The IMS Algorithm with the SOVA ...................... 24
2.5 Conclusion .................................................... 26
2.6 Acknowledgements ............................................ 26

Chapter 3 Characterization and Detection on Oversampled Channels .... 28
3.1 Introduction ................................................... 28
3.2 Oversampled Gaussian Channels ............................... 29
   3.2.1 2-D Oversampled Gaussian Channels ..................... 29
   3.2.2 1-D Oversampled Gaussian Channels .................... 34
3.3 Channel Characterization on Oversampled Channels ........ 36
3.4 Minimum Mean-Square Equalizers ............................ 39
3.5 A MAP Detector for 1-D Oversampled Channels ............. 41
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7</td>
<td>Block Codes for 2-D Stripes</td>
<td>114</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Single-State Codes</td>
<td>115</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Finite-State Codes</td>
<td>116</td>
</tr>
<tr>
<td>5.8</td>
<td>Conclusion</td>
<td>117</td>
</tr>
<tr>
<td>5.9</td>
<td>Acknowledgements</td>
<td>117</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>118</td>
</tr>
<tr>
<td>6.2</td>
<td>The Depth-First Algorithm</td>
<td>120</td>
</tr>
<tr>
<td>6.3</td>
<td>Results</td>
<td>125</td>
</tr>
<tr>
<td>6.4</td>
<td>The Performance of a Coded System</td>
<td>128</td>
</tr>
<tr>
<td>6.5</td>
<td>Conclusion</td>
<td>130</td>
</tr>
<tr>
<td>6.6</td>
<td>Acknowledgements</td>
<td>131</td>
</tr>
<tr>
<td>6.7</td>
<td>Depth-First Algorithm for Designing 2-D Single-State Block Codes</td>
<td>132</td>
</tr>
<tr>
<td>6.8</td>
<td>Chapter 6</td>
<td>132</td>
</tr>
<tr>
<td>6.9</td>
<td>Combined Constraints for Perpendicular Recording Channels</td>
<td>135</td>
</tr>
<tr>
<td>6.10</td>
<td>Channel Targets</td>
<td>138</td>
</tr>
<tr>
<td>6.11</td>
<td>Recording Channels Exhibiting Low Frequency Contents</td>
<td>138</td>
</tr>
<tr>
<td>6.12</td>
<td>DC-Compensation Loops for DC-Full or DC-Attenuated Targets</td>
<td>139</td>
</tr>
<tr>
<td>6.13</td>
<td>Utilization of RDS Constraints for DC-Full or DC-Attenuated Targets</td>
<td>139</td>
</tr>
<tr>
<td>6.14</td>
<td>Utilization of RDS Constraints for DC-Free Targets</td>
<td>141</td>
</tr>
<tr>
<td>6.15</td>
<td>High Rate RDS Constraints and Long DC-Free Targets</td>
<td>142</td>
</tr>
<tr>
<td>6.16</td>
<td>Preliminaries</td>
<td>144</td>
</tr>
<tr>
<td>6.17</td>
<td>RDS Constraints</td>
<td>144</td>
</tr>
<tr>
<td>6.18</td>
<td>The RLL (0, k) Constraints</td>
<td>145</td>
</tr>
<tr>
<td>6.19</td>
<td>MTR and TMTR Constraints</td>
<td>145</td>
</tr>
<tr>
<td>6.20</td>
<td>Combined Constraints</td>
<td>145</td>
</tr>
<tr>
<td>6.21</td>
<td>Spectral Properties</td>
<td>146</td>
</tr>
<tr>
<td>6.22</td>
<td>Coding Examples</td>
<td>146</td>
</tr>
<tr>
<td>6.23</td>
<td>A Rate 20/21 Block Code</td>
<td>147</td>
</tr>
<tr>
<td>6.24</td>
<td>A Rate 100/108 Block Code</td>
<td>147</td>
</tr>
<tr>
<td>6.25</td>
<td>Conclusions</td>
<td>147</td>
</tr>
<tr>
<td>6.26</td>
<td>Acknowledgements</td>
<td>147</td>
</tr>
<tr>
<td>6.27</td>
<td>Capacity Tables</td>
<td>147</td>
</tr>
<tr>
<td>6.28</td>
<td>Power Spectral Density Plots</td>
<td>147</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>150</td>
</tr>
<tr>
<td>7.2</td>
<td>Channel Targets</td>
<td>150</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Recording Channels Exhibiting Low Frequency Contents</td>
<td>150</td>
</tr>
<tr>
<td>7.2.2</td>
<td>DC-Compensation Loops for DC-Full or DC-Attenuated Targets</td>
<td>150</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Utilization of RDS Constraints for DC-Full or DC-Attenuated Targets</td>
<td>150</td>
</tr>
<tr>
<td>7.2.4</td>
<td>Utilization of RDS Constraints for DC-Free Targets</td>
<td>150</td>
</tr>
<tr>
<td>7.2.5</td>
<td>High Rate RDS Constraints and Long DC-Free Targets</td>
<td>150</td>
</tr>
<tr>
<td>7.3</td>
<td>Preliminaries</td>
<td>150</td>
</tr>
<tr>
<td>7.3.1</td>
<td>RDS Constraints</td>
<td>150</td>
</tr>
<tr>
<td>7.3.2</td>
<td>The RLL (0, k) Constraints</td>
<td>150</td>
</tr>
<tr>
<td>7.3.3</td>
<td>MTR and TMTR Constraints</td>
<td>150</td>
</tr>
<tr>
<td>7.4</td>
<td>Combined Constraints</td>
<td>150</td>
</tr>
<tr>
<td>7.5</td>
<td>Spectral Properties</td>
<td>150</td>
</tr>
<tr>
<td>7.6</td>
<td>Coding Examples</td>
<td>150</td>
</tr>
<tr>
<td>7.6.1</td>
<td>A Rate 20/21 Block Code</td>
<td>150</td>
</tr>
<tr>
<td>7.6.2</td>
<td>A Rate 100/108 Block Code</td>
<td>150</td>
</tr>
<tr>
<td>7.7</td>
<td>Conclusions</td>
<td>150</td>
</tr>
<tr>
<td>7.8</td>
<td>Acknowledgements</td>
<td>150</td>
</tr>
<tr>
<td>7.9</td>
<td>Capacity Tables</td>
<td>150</td>
</tr>
<tr>
<td>7.10</td>
<td>Power Spectral Density Plots</td>
<td>150</td>
</tr>
<tr>
<td>8.1</td>
<td>Conclusions</td>
<td>152</td>
</tr>
<tr>
<td>8.2</td>
<td>Bibliography</td>
<td>152</td>
</tr>
</tbody>
</table>

vii
LIST OF FIGURES

1.1 A typical storage system. Acronym MC stands for “modulation code”.

2.1 (a) Recording data and (b) reading the stored data back (taken from [1]).

2.2 Recording and reading information in four steps (taken from [1]).

2.3 A communication channel model for holographic storage systems.

2.4 Discrete-time LTI channel model.

2.5 Noise histograms and their Gaussian p.d.f. fits for the full-response and 2-D PR1 targets are shown on the left and right figures, respectively.

2.6 Weighting functions for the linear and non-linear channel models where \( \alpha = \alpha_1 = \alpha_2 = 1 \).

2.7 Noise histograms along with Gaussian p.d.f. fits for the unequalized target. The plots from left to right correspond to the linear, square-root and elliptic weighting cases.

2.8 Channel model for threshold detection.

2.9 Channel model for threshold detection with precoding.

2.10 Channel model for IMS detection.

2.11 The performance of the IMS algorithm and the threshold detector for the 2-D PR1 channel.

2.12 The performance of the IMS algorithm with the log-MAP detector. The strip size for the IMS algorithm is 2.

2.13 Performance of the IMS algorithm with the log-MAP detector. The strip size for the IMS algorithm is 2.

2.14 Performance of the IMS algorithm with the max-log-MAP detector. The strip size for the IMS algorithm is 2.

2.15 Performance of the IMS algorithm using the SOVA. The strip size for IMS algorithm is 2.

2.16 Performance of the IMS algorithm with the SOVA for various window sizes.

3.1 Oversampled Gaussian channel model.

3.2 SNR plot for non-oversampled Gaussian channels for different values of densities and misalignments where \( \Delta = 1 \) and \( \sigma = 0.1 \).

3.3 SNR values for oversampled Gaussian channels for different densities and oversampling ratios in the pixel-matched case where \( \Delta = 1, \sigma = 0.1 \).

3.4 SNR values for oversampled Gaussian channels for different values of misalignments and oversampling ratios where \( \Delta = 1, \sigma = 0.1, \text{ and } \epsilon = 0.5 \).

3.5 1-D oversampled Gaussian channel model.

3.6 True and characterized SNR values for oversampled Gaussian channel where \( \Delta = 1, \sigma = 0.1, \epsilon = 0.5, M = N = 50 \).

3.7 The input-output diagram for an oversampled channel for \( u/w = 4/3 \) and \( L_x = 4 \).
3.8 Performance of the detectors for the non-oversampled and oversampled 1-D Gaussian channels in the pixel-matched case. ........................................... 45
3.9 The performance of the log-MAP detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channels for known misalignments. .......... 46
3.10 The performance of the log-MAP detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channels for unknown misalignments. ...... 47
3.11 The performance of the full-response equalizer with the threshold detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channel for known misalignments. ........................................... 48
3.12 The performance of the full-response equalizer with the threshold detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channel for unknown misalignments. ........................................... 49

4.1 A 2-D ISI channel model. ................................................................. 52
4.2 Precoded 2-D ISI channel model. ...................................................... 59
4.3 An example of connection map. ......................................................... 71
4.4 The number of error events of size \( m \times n \) for the 2-D PR1 channel where \( d^2 \leq 8 \). ........................................................................................................... 72
4.5 The complexity of the algorithm for the 2-D PR1 channel when \( d_{\text{max}}^2 = 6 \). The check IV refers to the modified squared distance with \( d_{\min}^2 = 2 \) for all \( m \) and \( n \). ................................................................. 77

5.1 The hard-square model. ................................................................. 85
5.2 Rectangular and parallelogram codewords and different types of tiling. ... 85
5.3 Violations of the hard-square constraint at hard-square array boundaries. .... 86
5.4 The orderings of the codewords along (a) diagonals and (b) rows. ............. 87
5.5 The state of the encoder, \((a, b)\). ......................................................... 87
5.6 A generating template, \( T(\bar{a}, \bar{b}, c, d) \), following the state \((a, b)\). .......... 96
5.7 The complexities of algorithm \( A \) and the brute-force algorithm. ............. 100
5.8 The complexities of algorithms \( B \) and \( C \) with their upper bounds. .......... 102
5.9 (a) Encoder and (b) decoder block diagrams. ........................................ 109
5.10 The simple template encoder for Example 5.10. .................................... 110
5.11 The construction of a \( 2 \times 2 - 6 \times 4 \) single-state block code. \( X = B, C, D, E \). 113
5.12 Tiling hard-square arrays horizontally. ............................................... 115
5.13 A generating template, \( T(\bar{a}, b) \), following the state, \( a \in V \). .......... 117

6.1 (a)-(c) Different situations of \( p \) across the \( 3 \times 3 \) codeword window, (d) the number forbidden pieces for each position in the codeword window. .......... 121
6.2 Decision tree of \( 3 \times 3 \) codewords for the n.i.1 constraint. ...................... 122
6.3 Complexity plots for the depth-first (DF) and the brute-force (BF) algorithms for the first group of constraints. ................................. 126
6.4 Complexity values for the depth-first (DF) and brute-force (BF) algorithms for the second group of constraints. ................................. 127
6.5 The simulation diagram for the rate \( 13/16 \) \( 4 \times 4 \) single-state block code. .. 128
6.6 Performance of the uncoded channel. ............................................... 129
6.7 Performance of the rate \( 13/16 \) modulation code. .................................. 130
7.1 Amplitude response of the longitudinal and perpendicular recording channels, along with some channel target responses. ......................... 136
7.2 (a) Mealy-type and (b) Moore-type graphs of the RDS constraint for $N = 4$. 139
7.3 The graph of the RLL(0,4) constraint. ................................. 139
7.4 The graphs of the (a) MTR (4) and (b) TMTR (2,3) constraints. ........... 140
7.5 Block diagram of the rate 100/108 block code. ........................... 151
7.6 Power spectral densities of the RDS constraints. ......................... 157
7.7 Power spectral densities of TMTR $(r,k)$ constraints for $r = (3,4)$. .... 158
7.8 Power spectral densities of RDS-TMTR $(N,r)$ constraints for $r = (3,4)$. . . 158
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Characterized channel parameters for the normalized full-response (A) and 2-D PR1 (B) targets. The numbers are rounded to two decimal points.</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Characterized channel parameters for the normalized target and different weighting functions. Training page size has size $100 \times 100$. The numbers are rounded to two decimal points.</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Backward impulse response estimates $(g')_{p,q}^{(i,j)}$ for data sets taken from the center of the pages.</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Characterized channel parameters and channel parameters for the full response target.</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>The percentage of bit errors caused by some error events for the 2-D PR1 channel. The bit error rates are $5.45 \times 10^{-2}$, $3.11 \times 10^{-2}$, and $1.46 \times 10^{-2}$ for 10 dB, 11 dB, and 12 dB, respectively.</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Some error events for the 2-D PR1 channel.</td>
<td>73</td>
</tr>
<tr>
<td>4.3</td>
<td>The number of error events for the 2-D PR1 channel.</td>
<td>73</td>
</tr>
<tr>
<td>4.4</td>
<td>Error events with squared distance $d^2 \leq 2.25$ for the channel given in Example 4.11.</td>
<td>74</td>
</tr>
<tr>
<td>4.5</td>
<td>Some error events with squared distance $d^2 \leq 9$ for the channel given in Example 4.12.</td>
<td>75</td>
</tr>
<tr>
<td>4.6</td>
<td>Some error events for the channel given in Example 4.13.</td>
<td>76</td>
</tr>
<tr>
<td>4.7</td>
<td>Lower bounds to squared distances of error events.</td>
<td>82</td>
</tr>
<tr>
<td>4.8</td>
<td>The minimum squared distance for a fixed error event size.</td>
<td>83</td>
</tr>
<tr>
<td>5.1</td>
<td>The rates of single-state and finite-state codes.</td>
<td>88</td>
</tr>
<tr>
<td>5.2</td>
<td>The hard-square table of $(V_0, H_0)$.</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>The hard-square table of $(V_1, H_1)$.</td>
<td>97</td>
</tr>
<tr>
<td>5.4</td>
<td>Generating polynomials of optimal generating templates.</td>
<td>105</td>
</tr>
<tr>
<td>5.5</td>
<td>Generating polynomials of some single-state block codes.</td>
<td>107</td>
</tr>
<tr>
<td>5.6</td>
<td>The hard-square table of $(V_1, H_1)$.</td>
<td>110</td>
</tr>
<tr>
<td>5.7</td>
<td>The number of states for finite-state codes by algorithm A.</td>
<td>111</td>
</tr>
<tr>
<td>5.8</td>
<td>The rates of single-state and finite-state codes for 2-D stripes.</td>
<td>115</td>
</tr>
<tr>
<td>6.1</td>
<td>Some 2-D constraints and their capacity values.</td>
<td>119</td>
</tr>
<tr>
<td>6.2</td>
<td>The rates of the optimal single-state codes for various constraints.</td>
<td>120</td>
</tr>
<tr>
<td>7.1</td>
<td>Cut-off frequencies of the RDS and RDS-TMTR constraints for $r = (3, 4)$ and various values of $N$. Values are shown as percentages of $2\pi$.</td>
<td>144</td>
</tr>
<tr>
<td>7.2</td>
<td>Prefixes and suffixes at codeword boundaries.</td>
<td>146</td>
</tr>
<tr>
<td>7.3</td>
<td>Histogram of codewords according to their $\Delta z$ values.</td>
<td>148</td>
</tr>
<tr>
<td>7.4</td>
<td>Optimized parameters for the RDS, RLL $(0, k)$, and twin constraints.</td>
<td>150</td>
</tr>
<tr>
<td>7.5</td>
<td>Capacity values for MTR $(j, k)$ constraints.</td>
<td>154</td>
</tr>
<tr>
<td>7.6</td>
<td>Capacity values for TMTR $(r, k)$ constraints for $r = (j, j + 1)$.</td>
<td>154</td>
</tr>
<tr>
<td>7.7</td>
<td>Capacity values for RDS $(N, k)$ constraints.</td>
<td>155</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>7.8</td>
<td>Capacity values for RDS-MTR $(N,j)$ constraints</td>
<td>155</td>
</tr>
<tr>
<td>7.9</td>
<td>Capacity values for RDS-TMTR $(N,r)$ constraints for $r = (j, j + 1)$</td>
<td>156</td>
</tr>
<tr>
<td>7.10</td>
<td>Capacity values for RDS-MTR $(N,j,k)$ constraints for $j = 3$</td>
<td>156</td>
</tr>
<tr>
<td>7.11</td>
<td>Capacity values for RDS-MTR $(N,j,k)$ constraints for $j = 4$</td>
<td>157</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

First of all, I am grateful to my advisor, Prof. Jack Wolf for his patience and excellent guidance that he had given me. His knowledge about his field and his ability to simplify complex problems is outstanding. I could not have done much of this dissertation research without his continual encouragement and support.

I am grateful to Prof. Paul Siegel for his fruitful collaboration and discussions. His deep knowledge and his analytic way of thinking have influenced my approach to solve problems.

I am thankful to my other committee members, Larry Milstein, Alon Orlitsky, and Bill Hodgkiss for their time and their classes which have been a foundation for my research.

It has been a great pleasure and collaboration working with the Wolf group and STAR group members. I would like to thank the present and recent members, Sharon Aviran, Brain Kurkoski, Mike Cheng, Panu Chaichanavong, Vladimir Dorfman, Henry Pfisher, John Miller, Seyhan Karakulak, Junsheng Han, Zheng Wu. I am indebted to Marcus Marrow, Joseph Soriaga and Zeinab Taghavi for their enlightening discussions and collaborations on two-dimensional storage. I would like thank former CMRR visiting scholar Pal Ellingsen for his excellent friendship and encouragement.

I am indebted to Yuan Xing Lee from Hitachi Global Storage Technologies for his excellent guidance and collaboration that led to Chapter 7.

I would like to thank Mark Ayres, Adrian Hill and Kevin Curtis from InPhase Technologies for their helpful discussions about holographic data storage and providing the simulation data for Chapters 2 and 3.

I would like to thank present and recent CMRR staff, Betty Manoulian, Iris Villanueva, and Cheryl Hacker, for helping with administrative issues.

I am grateful to my family and my wife, Zeynep, for their love and support during last five years. I am specially grateful to my elder brother Levent Demirkan for his excellent guidance and continual encouragement about my career.

Funding for this research was provided in part by a California Institute for Telecommunications and Information Technology (CALIT2) fellowship, the National Science Foundation (NSF), the Center for Magnetic Recording Research (CMRR) and its sponsors, and InPhase Technologies in cooperation with the National Institute of Standards and Technology (NIST). My internship in the summer of 2003 was supported by Hitachi Global Storage Technologies.
VITA

1979 Born, Kastamonu, Turkey

2001 B.S., Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey

2003 M.S., Electrical and Computer Engineering (Communications Theory and Systems), University of California, San Diego

2006 Ph.D., Electrical and Computer Engineering (Communications Theory and Systems), University of California, San Diego

PUBLICATIONS


Coding and Detection for 2-Dimensional Channels

by

İsmail Demirkan

Doctor of Philosophy in Electrical Engineering
(Communications Theory and Systems)

University of California San Diego, 2006

Professor Jack K. Wolf, Chair

Coding and detection techniques for one-dimensional (1-D) intersymbol interference (ISI) channels, particularly magnetic and optical recording channels, have been studied extensively for almost three decades. On the modulation coding side, the state-splitting algorithm has been developed to design efficient systematic modulation codes. On the detection side, Viterbi detector and decision-feedback equalization (DFE) have been well-understood.

Two-dimensional (2-D) holographic data storage, has been developed to store the information pagewise instead of on 1-D tracks. This will significantly increase the storage density and read/write access of the information.

However, most of the modulation coding and detection techniques for 1-D recording systems are unusable for 2-D holographic data storage. In this work, we present various methods for modelling and equalizing 2-D ISI channels. Some low complexity detectors, such as a threshold detector, have been proposed for certain 2-D ISI channels. One of the main problems of the holographic data storage is the misalignment between written and sampled data pages. This problem is addressed by using more detector pixels than data points, which is called oversampling. We also attempted to characterize the distance properties of certain 2-D ISI channels. An algorithm for finding error events is developed for any 2-D ISI channel.

Unlike the 1-D constrained systems, the capacity of most 2-D constrained systems is not analytically known due to the lack of graph-based descriptions of such channels. This also complicates the design of efficient modulation codes. We propose algorithms for finding single-state and finite-state block codes for the hard-square constraint. The
encoding and decoding of the modulation codes can be performed easily using codeword generating templates. We also propose an algorithm for finding single-state block codes for any 2-D constrained system represented by a set of forbidden patterns.

For 1-D recording, we designed block codes satisfying the running digital sum (RDS) and time-varying maximum transition run (TMTR) constraints for perpendicular recording channels. The graphs of these constraints are combined to understand the design trade-off between the achievable coding rate and constraint parameters. The spectra of the combined constraints shows the properties of the constituent constraints. The modulations codes are designed by searching for all codewords satisfying certain constraint properties.
Chapter 1

Introduction

1.1 Communication Systems

A model of a communication system is a mathematical model which reflects our knowledge about a physical system, which is designed for transmitting information reliably between two points or storing and retrieving information reliably in a storage medium. One part of this knowledge can be deterministic quantities such as the channel impulse response or the frequency of a carrier signal. The other part of this knowledge can be random such as additive white Gaussian noise (AWGN). The parts of the physical system that are not contained in the model cause the performance of the model to deviate from the performance of the true system. We will assume that the model is an accurate enough description of the true system so that we no longer have to differentiate between the two.

Communication systems have two types of elements: signals carrying the information and systems modifying the signals. Signal and systems are divided into two categories in terms of their control: the physical channel which cannot be controlled, and other signal and systems that can be controlled and modified. All signals and systems in the second part are designed to mitigate the deleterious effects of the physical channel on the information carrying signal. In this way, the overall communication system can transmit or store the information signal reliably.

This thesis focuses on data storage systems, whose commonly used components are depicted in Fig. 1.1. The information signal is converted to a write signal by a modulator and this signal is written onto the storage medium using a write-head. The stored information is retrieved by using a read-head. The read-back signal is demodulated to obtain the information signal. Inside the channel, the write signal is modified in many ways including
interface from neighboring information bits and noise. In real storage systems, the noise depends on the information signal or correlates with other realizations of the noise. It is common to compensate these non-linear imperfections by an equalizer. The detector gives an estimate of the information signal stored in the medium. When errors occur in the detector, they can be corrected by an error-correcting code (ECC). Modulation codes can also be used to improve the detector performance by eliminating common error patterns from the detector output.

1.2 Two-dimensional Storage Systems

In the majority of communication systems, the information signal can be represented as a one-dimensional (1-D) sequence where the sequence variable is often the time. However, in two-dimensional (2-D) storage systems, the information signal can be written page by page. Therefore it is more convenient to describe it as a 2-D array. An important example of 2-D storage systems is 2-D holographic data storage, which is based on storing the information signal as an interference pattern on a photo-sensitive material. Using this technique, one can store several pages of information signals into the same volume (see Chapter 2).

The information signal is modulated using a spatial light modulator (SLM), which is an array of pixels. An information bit is modulated as ON or OFF depending on the modulation scheme. When a pixel is ON, the light is fed through the pixel and diffracted according to the point-spread function associated with the aperture of the pixel. This results in neighboring pixels interfering with each other. This interference causes 2-D intersymbol interference (ISI) in the readback signal, which is often detected by a camera. High information densities require stacking of channel symbols much closer to each other.
and therefore resulting in severe ISI in the channel. Much of this thesis is concerned with
the detection and modulation coding for 2-D ISI channels.

1.3 Channel Modeling and Detection

Recovering written data from a holographic medium is not an easy task due to the
imperfections of the optical system, such as misalignments between the SLM and the
camera devices, and lens defects. Residual imperfections can be considered as noise sources
in the optical system. There are two main noise sources for holographic data storage
systems: electrical noise and optical coherent noise depending on the information signal.
The latter noise source is more dominant and requires more complicated channel models
and detection systems.

Channel modeling and detection for a 2-D holographic data storage system is funda-
mentally more complex than that for a 1-D storage system. The stored information is read
back by a camera, which measures the intensity of the received optical signal. Therefore,
the camera output is always positive and the sign information of the received signal is lost.
This complicates the channel modeling and detection schemes since the data and noise
become dependent in this way.

A maximum likelihood sequence estimator or Viterbi algorithm [2, 3] is a commonly
used detector in communication and storage systems. Since 2-D ISI channels lack trellis-
based descriptions, it is not possible to generalize the Viterbi algorithm for these channels.
This also holds for the maximum a-posteriori algorithm (MAP) or BCJR algorithm, which
is an optimal detector [4]. Marrow proposed a suboptimal detection algorithm for 2-D ISI
channels, which is called the iterative multi-strip algorithm (IMS) [5]. When the number
of iterations is large, the IMS algorithm performs approximately the same as the optimal
detector (see Section 2.4).

1.4 Two-dimensional Constraints and Modulation Codes

A 2-D constrained system consists of binary arrays satisfying certain properties. Some
of the studies done on 2-D constraints were not motivated by 2-D storage systems, but
rather they are proposed to solve some problems in statistical physics [6]. The hard-square
and hard-hexagons models are two examples of this kind. The hard-square model is a
2-D constrained system consisting of all binary arrays on a rectangular grid in which 1's
are isolated both horizontally and vertically. In the hard-hexagons model, 1’s should be isolated in a hexagonal lattice. Although these constraints do not imply an application in storage systems, they have been studied by many information theorists [7, 8, 9, 10]. Some 2-D constraints are proposed as an extension of 1-D constraints, such as 2-D run-length-limited (RLL) constraints.

Systematic approaches for designing 1-D modulation codes and finding the capacity of constraints are mainly based on labeled graph representations of the constraints. However, these methods cannot be generalized to 2-D constraints since 2-D constraints lack descriptions in terms of labeled graphs. There are two main consequences of this. One of them is that code design techniques for a 2-D constraint become specific to that constraint and cannot be easily generalized to other 2-D constraints. Some chapters of this thesis introduce various methods for designing block codes for 2-D constraints. The other consequence is that the capacity values for most 2-D constraints remain unknown but they are approximated by various upper and lower bounding techniques.

1.5 Information Theory and Error Correcting Codes

Shannon introduced a concept of measuring the information content of a signal by its entropy [11]. This theory is called information theory. Shannon solved the basic dilemma of a communication system: reliable transmission of information at the largest coding rate, which is called the channel capacity. Shannon’s theory is based on the properties of stochastic sequences, therefore his proofs do not lead to the design of specific error correction codes.

There have been three notable breakthroughs in coding theory. The first one was developed by Reed and Solomon in early 1960’s [12] and uses the mathematics of finite fields. The simplicity of encoder and decoder made the RS codes a de facto standard in practical applications. The second breakthrough was turbo codes, which were invented by Berrou, Glavieux and Thitimajshima [13]. The performance of a good turbo code is very close to the Shannon capacity in some communication channels. The third breakthrough was the invention of low-density parity-check codes by Gallager in early 1960’s [14]. These codes were constructed by using sparse random parity check matrices and showed promising performances. However, they went largely unnoticed until the invention of turbo codes, when they were rediscovered by MacKay, who showed that they perform almost as close to capacity as turbo codes [15].
The application of a random error correcting code for 2-D channels can be performed easily by serializing 2-D codewords to 1-D codewords. This thesis will not elaborate on ECCs for 2-D storage systems.

1.6 Outline of Dissertation

Chapter 2 covers modeling, equalization and detection of 2-D ISI channels. Linear and non-linear channel models are investigated where the unknown parameters of a channel model are estimated by using the least-squares estimation method. A low complexity detector, namely the threshold detector, has been proposed for the 2-D PR1 channel, which is an extension of the 1-D PR1 channel. The IMS algorithm is based on a message-passing algorithm operating on a MAP detector. Low complexity alternatives to the MAP detector are considered in order to reduce the complexity of the IMS algorithm.

Due to mechanical imperfections and media defects, the centers of SLM and the camera pixels can shift with respect to each other. In Chapter 3, this problem is addressed and solved by using cameras with smaller pixel size as compared with the SLM pixels. This method is called oversampling. Oversampled holographic data storage systems are fundamentally difficult to analyze since the channel impulse response becomes no longer time-invariant. In Chapter 3, we propose modeling, equalization and detection methods for oversampled 2-D ISI channels. The main advantage of this system is that channel variations for the different misalignments are reduced.

For 1-D recording channels, the performance of a maximum likelihood detector depends upon the distance properties of the channels, which are defined on a trellis. Error events with minimum and near minimum distances play an important role in the performance of a recording system particularly at moderate to high signal-to-noise ratios (SNR). In Chapter 4, we analyze the distance properties of 2-D ISI channels, in particular the 2-D PR1 channel. The minimum distance of this channel is defined and proved to be equal to 2 and a complete characterization of the distance-2 error events is provided. Also, the error events with squared-Euclidean distance 6 are partially characterized. Analogous to 1-D channels, error state diagrams for 2-D channels can be constructed to characterize the error events. We propose an efficient error event search algorithm operating on error state diagrams.

Chapter 5 proposes various algorithms for finding good single-state and finite-state codes with rectangular codewords that satisfy the hard-square model. Although the code-
word size is small, single-state codes with coding rates close to the capacity can be designed. Letting the 2-D constrained sequences have finite memory increases the achievable coding rate. A method for designing low complexity encoders and decoders is also presented. When the codeword size increases, the coding rate asymptotically approaches the capacity but with a rapidly increasing complexity of the encoder and decoder.

Most 2-D constrained systems can be represented as a set of forbidden patterns, which should be avoided in the input sequences to the channel in order to improve the detector performance. Chapter 6 proposes a depth-first algorithm for finding single-state codes with rectangular codewords for such constraints. This algorithm provides optimal single-state codes, meaning that no other single-state code of the same dimensions can have larger coding rate. For certain 2-D constraints, this algorithm has low complexity compared with the brute-force algorithm. The coding rate asymptotically approaches the capacity of 2-D constraints when the codeword size increases.

Chapter 7 covers a 1-D modulation code design technique for maximum transition run (MTR) and running digital sum (RDS) constraints, which have a wide variety of applications in magnetic recording systems. The labeled graph representations of RDS and MTR (or the time-varying MTR (TMTR)) constraints are combined to obtain new classes of constraints: RDS-MTR and RDS-TMTR constraints. The capacity of the combined constraint is observed to be slightly lower than that of its constituent constraints. A method for designing block codes is also proposed to implement the properties of the combined constraints. The code design methodology is demonstrated via a rate 100/108 RDS-TMTR block code based on a rate 20/21 RDS-TMTR mother code.

Chapter 8 contains the conclusions.
Chapter 2

Characterization and Detection on 2-D Channels

2.1 Introduction

Contemporary magnetic and optical recording systems write the information on isolated lines on the disk, which are called tracks. Recent advances in holographic data storage systems seem to abandon this tradition by allowing the writing of information as a page on the recording medium. Furthermore, several pages can be written into the same volume, which promises considerably higher recording densities as compared to current systems. This new optical recording technology is often referred to by different names in literature: two-dimensional (2-D) optical storage systems, volume-oriented storage technology, or holographic data storage [16, 17, 18]. Holographic data storage systems promise faster read and write data rates than conventional 1-D storage systems since it is possible to read and write the information page by page.

Physical aspects of holographic storage have been studied by many researchers [16, 19]. A good tutorial on the basic principles of holographic data storage systems can be found in [18]. Several authors have investigated signal processing and communication theory aspects of holographic data storage [17, 20, 21, 22, 5].

The basic components of a holographic storage system are depicted Fig 2.1. The incoming optical signal is split into two parts: a reference beam and an object beam. An information page is formed through the interference pattern of the object beam passing through a pixelated input device called spatial light modulator (SLM). The reference beam
Figure 2.1 (a) Recording data and (b) reading the stored data back (taken from [1]).

intersects the object beam in the storage material, allowing the storage and later retrieval of the holograms. Several pages of data can be stored in the same volume on the recording medium by using reference beams projected from different angles. Recorded data are read by illuminating the medium with a reference beam from the angle at which the page was recorded. The hologram is read through an array of detectors, such as a charge-coupled device (CCD), with a limited spatial resolution and with inherent noise. The recording and reading processes of holographic storage systems are shown in Fig. 2.2 in four steps. In step 1, the intersection of reference and object beams creates an interference pattern. In step 2, a photo-sensitive medium is exposed to the interference pattern. In step 3, the interference pattern is stored in the medium as changes in the refractive properties of the medium. In step 4, the data is reconstructed using the reference beam illuminated from the angle at which the page was recorded.

2-D holographic data storage systems can be modeled as a communication channel as shown in Fig. 2.3. A binary data array \( d_{i,j} \) is mapped to a 2-D optical write waveform \( w(x, y) \) by a light *modulator*, such as an SLM. The optical channel alters the optical write waveform by introducing intersymbol interference (ISI), linear and non-linear noises. The channel output waveform \( r(x, y) \) is demodulated and sampled by a camera to obtain \( r_{i,j} \). Channel detectors and equalizers are employed to estimate the channel input array \( d_{i,j} \) from the camera output \( r_{i,j} \).

The generalization of one-dimensional (1-D) detection and coding methods to 2-D channels is not trivial due to the lack of a convenient graph-based description of such channels. In particular, there is no simple trellis-based maximum likelihood detection
What is a Hologram?

The intersection of two beams creates an interference pattern of bright and dark regions. A photosensitive medium records the interference pattern. The hologram is the image of the interference pattern stored within the medium. Light from one beam shining on the hologram reconstructs the data pattern.

Figure 2.2 Recording and reading information in four steps (taken from [1]).

\[ d_{i,j} \xrightarrow{\text{Modulator (SLM)}} w(x, y) \xrightarrow{\text{Channel}} r(x, y) \xrightarrow{\text{Demodulator (Camera)}} r_{i,j} \xrightarrow{\text{Channel Detector}} d'_{i,j} \]

Figure 2.3 A communication channel model for holographic storage systems.

algorithm analogous to the Viterbi algorithm. However, there are suboptimal detection techniques such as the iterative multi-strip (IMS) algorithm [5] for 2-D partial-response channels that demonstrate very good error-rate performance and appear to approximate the performance of MAP detector.

This chapter discusses some diverse topics about channel characterization and detection for 2-D ISI channels. Channel modelling with linear and non-linear models are discussed in Section 2.2. Another aspect of the detection problem for 2-D ISI channels is to design low complexity detectors. Section 2.3 discusses an optimal threshold detector for a particular
channel target, namely the 2-D PR1 channel whose impulse response is

\[
    h = \begin{bmatrix}
        1 & 1 \\
        1 & 1 
    \end{bmatrix}.
\]  

The IMS algorithm is based on a message-passing algorithm operating along with the BCJR algorithm. In principle, the IMS algorithm can use any soft-input-soft-output (SISO) detector as an inner detector. Two sub-optimal algorithms has been proposed in Section 2.4 to reduce the complexity of the IMS algorithm: the max-log-MAP detector and the soft-output Viterbi algorithm (SOVA). It is observed that the performance of the IMS algorithm with the max-log-MAP detector or SOVA is not much different than the IMS algorithm with the BCJR detector for certain 2-D ISI channels.

2.2 Channel Characterization

2-D ISI channels can be modeled as linear or non-linear systems. Unknown parameters of a channel model can be estimated by using the least-squares estimation method [23, Ch. 8]. An attempt at modelling 2-D ISI channels by using least-squares estimation has been discussed in [5, Ch. 2]. In this section, we consider improving the linear channel model by introducing a weighting function operating on the camera output. This weighted non-linear channel model increases the characterized signal-to-noise ratio (SNR) of the channel.

2.2.1 Linear Channel Models

A discrete-time linear time-invariant (LTI) channel model for the 2-D ISI channel is depicted in Fig. 2.4. The bipolar channel input array \(d_{i,j}\) is convolved with the channel impulse response \(h_{i,j}\) to obtain the channel output array \(s_{i,j}\). In optical systems, the camera measures the intensity of the optical signal, not the amplitude of the electric or magnetic fields. Therefore the channel output sequence is always positive, i.e., \(r_{i,j} \geq 0\) for all \(i,j\). The noise component is assumed to be additive white Gaussian noise \(n_{i,j}\) with zero-mean and unknown variance \(\sigma^2\). In order to ensure that \(r_{i,j} \geq 0\) for almost all \((i,j)\) pairs, an offset value \(k\) is added to the channel output. Assuming that the input array has size \(M \times N\), the unknown parameters, \(h_{i,j}\) and \(k\), can be estimated by using the channel
input-output relationship

\[ r_{i,j} = \sum_{p=1}^{M} \sum_{q=1}^{N} h_{p,q} d_{i-p,j-q} + k + n_{i,j} \]  

(2.2)

where \( d_{i,j} \) is a known training array and \( r_{i,j} \) is the array which is read.

Suppose that the impulse response of the channel to be characterized has the size \( u \times v \).

The convolution (2.2) can be written as a matrix equation

\[ \mathbf{r} = \mathbf{H}\theta \]  

(2.3)

where \( \mathbf{H} \) is a known \( MN \times (uv + 1) \) matrix with full column rank, \( \theta \) is a column vector of length \( uv + 1 \) containing unknown parameters, and \( \mathbf{r} \) is a column vector of length \( MN \) containing the camera output. The least-square estimate of the unknown parameters \( \theta \) is given by

\[ \theta' = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{r}. \]

The noise component is estimated by the estimation error, i.e., \( \mathbf{n}' = \mathbf{r} - \mathbf{H}\theta' \). If this model is in fact the true model for the channel, then the characterized noise should be zero-mean white Gaussian noise.

The minimum least-squares estimation error is given by

\[ J_{\text{min}} = (\mathbf{n}')^T \mathbf{n}' = \mathbf{r}^T (\mathbf{r} - \mathbf{H}\theta') \]

which is equal to the characterized noise power. The estimate of the channel output sequence is given as \( s'_{i,j} = r_{i,j} - k' - n'_{i,j} \), where \( k' \) is the offset estimate and \( n'_{i,j} \) is the noise estimate at the \((i,j)\)th position. Therefore, the SNR estimate can be estimated as the ratio of the estimated total signal power and total noise power:

\[ \zeta' = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} |s'_{i,j}|^2}{J_{\text{min}}}. \]

The standard deviation of the noise can be estimated by

\[ \sigma' = \sqrt{\frac{J_{\text{min}}}{MN}}. \]
The channel output array can be scaled by a constant to normalize the channel impulse response. In this case, the channel input-output relationship becomes

$$\alpha r_{i,j} = \sum_{p=1}^{M} \sum_{q=1}^{N} (\alpha h)_{p,q} d_{i-p,j-q} + \alpha k + (\alpha n)_{i,j}$$  \hspace{1cm} (2.4)$$

where the normalization coefficient $\alpha$ is chosen such that the matched-filter bound of the channel is 1, i.e.,

$$||\alpha h'||^2 = \alpha^2 \sum_{p=1}^{u} \sum_{q=1}^{v} |h'_{p,q}|^2 = 1.$$  

The new offset and standard deviation estimates for the normalized channel are $\alpha k'$ and $\alpha \sigma'$, respectively, while the SNR estimate stays the same.

Example 2.1. Channel output arrays of size 752 $\times$ 752 are equalized to two different channel targets: full response and the 2-D PR1 targets. The impulse responses for the full response target is $h = 1$ and the impulse response of the 2-D PR1 target is given in (2.1). Equalizers for both targets have 16-taps and are discussed in detail in Section 3.4. Different data sets of size 100 $\times$ 100 taken from the center and corner of the page are used to estimate the channel parameters (see Table 2.1). The resolution of the camera is 10-bits. The output of the camera is normalized such that the largest value of $r$ is 1. The data set taken from the corner of the page is worse than that taken from the center of the page since, in general, non-linear effects increase from the center to corner of the page. The SNR value for the full response target is better than that of the 2-D PR1 target, which indicates that the full response target is a better fit for this channel.

Noise histograms for some channel characterizations are shown in Fig. 2.5, where the data sets are taken from the center of the page. The Gaussian probability density function (p.d.f.) fits are also shown to indicate the similarity between the characterized and ideal noise distributions. When the differences on the points near the peaks of the histograms are compared, the equalizer for the full-response target linearizes the channel better than that for the 2-D PR1 target.

2.2.2 Weighted Channel Models

The channel input-output relationship (2.4) can be expressed as follows

$$\alpha r_{i,j} - \alpha k = \sum_{p=1}^{M} \sum_{q=1}^{N} (\alpha h)_{p,q} d_{i-p,j-q} + (\alpha n)_{i,j}. \hspace{1cm} (2.5)$$

$$\sum_{p=1}^{u} \sum_{q=1}^{v} |h'_{p,q}|^2 = 1.$$  

where the normalization coefficient $\alpha$ is chosen such that the matched-filter bound of the channel is 1, i.e.,

The new offset and standard deviation estimates for the normalized channel are $\alpha k'$ and $\alpha \sigma'$, respectively, while the SNR estimate stays the same.

Example 2.1. Channel output arrays of size 752 $\times$ 752 are equalized to two different channel targets: full response and the 2-D PR1 targets. The impulse responses for the full response target is $h = 1$ and the impulse response of the 2-D PR1 target is given in (2.1). Equalizers for both targets have 16-taps and are discussed in detail in Section 3.4. Different data sets of size 100 $\times$ 100 taken from the center and corner of the page are used to estimate the channel parameters (see Table 2.1). The resolution of the camera is 10-bits. The output of the camera is normalized such that the largest value of $r$ is 1. The data set taken from the corner of the page is worse than that taken from the center of the page since, in general, non-linear effects increase from the center to corner of the page. The SNR value for the full response target is better than that of the 2-D PR1 target, which indicates that the full response target is a better fit for this channel.

Noise histograms for some channel characterizations are shown in Fig. 2.5, where the data sets are taken from the center of the page. The Gaussian probability density function (p.d.f.) fits are also shown to indicate the similarity between the characterized and ideal noise distributions. When the differences on the points near the peaks of the histograms are compared, the equalizer for the full-response target linearizes the channel better than that for the 2-D PR1 target.

2.2.2 Weighted Channel Models

The channel input-output relationship (2.4) can be expressed as follows

$$\alpha r_{i,j} - \alpha k = \sum_{p=1}^{M} \sum_{q=1}^{N} (\alpha h)_{p,q} d_{i-p,j-q} + (\alpha n)_{i,j}. \hspace{1cm} (2.5)$$

where the normalization coefficient $\alpha$ is chosen such that the matched-filter bound of the channel is 1, i.e.,

$$||\alpha h'||^2 = \alpha^2 \sum_{p=1}^{u} \sum_{q=1}^{v} |h'_{p,q}|^2 = 1.$$  

The new offset and standard deviation estimates for the normalized channel are $\alpha k'$ and $\alpha \sigma'$, respectively, while the SNR estimate stays the same.

Example 2.1. Channel output arrays of size 752 $\times$ 752 are equalized to two different channel targets: full response and the 2-D PR1 targets. The impulse responses for the full response target is $h = 1$ and the impulse response of the 2-D PR1 target is given in (2.1). Equalizers for both targets have 16-taps and are discussed in detail in Section 3.4. Different data sets of size 100 $\times$ 100 taken from the center and corner of the page are used to estimate the channel parameters (see Table 2.1). The resolution of the camera is 10-bits. The output of the camera is normalized such that the largest value of $r$ is 1. The data set taken from the corner of the page is worse than that taken from the center of the page since, in general, non-linear effects increase from the center to corner of the page. The SNR value for the full response target is better than that of the 2-D PR1 target, which indicates that the full response target is a better fit for this channel.

Noise histograms for some channel characterizations are shown in Fig. 2.5, where the data sets are taken from the center of the page. The Gaussian probability density function (p.d.f.) fits are also shown to indicate the similarity between the characterized and ideal noise distributions. When the differences on the points near the peaks of the histograms are compared, the equalizer for the full-response target linearizes the channel better than that for the 2-D PR1 target.

2.2.2 Weighted Channel Models

The channel input-output relationship (2.4) can be expressed as follows

$$\alpha r_{i,j} - \alpha k = \sum_{p=1}^{M} \sum_{q=1}^{N} (\alpha h)_{p,q} d_{i-p,j-q} + (\alpha n)_{i,j}. \hspace{1cm} (2.5)$$
Table 2.1 Characterized channel parameters for the normalized full-response \((A)\) and 2-D PR1 \((B)\) targets. The numbers are rounded to two decimal points.

<table>
<thead>
<tr>
<th>Channel Target</th>
<th>(\alpha)</th>
<th>(\alpha h')</th>
<th>(\alpha k')</th>
<th>(\zeta') (dB)</th>
<th>(\alpha \sigma')</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) (1 \times 1) center</td>
<td>6.25</td>
<td>2.66</td>
<td>7.46</td>
<td>0.42</td>
<td></td>
</tr>
<tr>
<td>(A) (1 \times 1) corner</td>
<td>6.34</td>
<td>2.71</td>
<td>7.21</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>(A) (3 \times 3) center</td>
<td>6.23</td>
<td>(-0.02 -0.05 -0.01)</td>
<td>0.99</td>
<td>2.65</td>
<td>6.66</td>
</tr>
<tr>
<td>(A) (3 \times 3) corner</td>
<td>6.31</td>
<td>(-0.01 -0.05 -0.02)</td>
<td>0.99</td>
<td>2.69</td>
<td>7.39</td>
</tr>
<tr>
<td>(B) (2 \times 2) center</td>
<td>6.97</td>
<td>0.49 0.46</td>
<td>0.54 0.51</td>
<td>2.71</td>
<td>6.93</td>
</tr>
<tr>
<td>(B) (2 \times 2) corner</td>
<td>7.10</td>
<td>0.46 0.50</td>
<td>0.50 0.54</td>
<td>2.77</td>
<td>6.67</td>
</tr>
<tr>
<td>(B) (4 \times 4) center</td>
<td>6.93</td>
<td>(-0.03 0 0 -0.02)</td>
<td>0.49 0.47</td>
<td>2.70</td>
<td>7.06</td>
</tr>
<tr>
<td>(B) (4 \times 4) corner</td>
<td>7.08</td>
<td>(-0.02 0.01 0.01 -0.02)</td>
<td>0.46 0.50</td>
<td>2.76</td>
<td>6.67</td>
</tr>
</tbody>
</table>

The left-hand side of (2.5) can be viewed as a linear function of \(r_{i,j}\). Therefore, adding an offset and the normalization are just linear weighting functions on the camera output. In this section, we consider non-linear weighting functions operating on the camera output. Consider the following equation

\[
w(r_{i,j}) - k = \sum_{p=1}^{M} \sum_{q=1}^{N} h_{p,q} d_{i-p,j-q} + n_{i,j}. \tag{2.6}
\]

where \(w\) is a non-linear weighting function such that the matched-filter bound of the channel is normalized to 1. Therefore, we can follow the same estimation procedure as in Subsection 2.2.1 regardless of the weighting function.

There are two weighting function considered in this chapter: square-root and elliptic weighting functions. Since the camera measures the intensity of the optical wave, it may be a reasonable approximation that the camera output is the square of the underlying channel response. Taking the square-root of the camera output gives an approximate value of the
Figure 2.5 Noise histograms and their Gaussian p.d.f. fits for the full-response and 2-D PR1 targets are shown on the left and right figures, respectively.

underlying channel response. Therefore, the square-root weighting function is given

\[ w_1(r) = \alpha_1 \sqrt{r} \]

where \( \alpha_1 \) is a normalization constant such that \( ||\alpha_1 h'||^2 = 1 \). Since the largest value of \( r \) is 1, \( w_1(1) = \alpha_1 \) as in the linear case. Another weighting function can be obtained by considering one quarter of an ellipse:

\[ w_2 = \alpha_2 \sqrt{1 - (r - 1)^2}. \]

These weighting functions along with the linear weighting are shown in Fig. 2.6.

The effects of these weighting functions are simulated on the training pages where the channel output is not equalized. In fact, these weighting functions do not improve the SNR performance of data sets used in Subsection 2.2.1 since the equalizers for the full-response and 2-D PR1 targets have already implemented the square-root weighting function. We propose the following channel target for the unequalized channel

\[
h = \begin{bmatrix}
h_{1,1} & h_{1,2} & h_{1,3} \\
h_{2,1} & h_{2,2} & h_{2,3} \\
h_{3,1} & h_{3,2} & h_{3,3}
\end{bmatrix}.
\]
Characterized channel parameters for this target are shown in Table 2.2. The benefit of the square-root and elliptic weightings are clear since there is about 0.7-1.1 dB performance gain compared with the linear weighting. The performance of the elliptic weighting is better than that of the square-root weighting in both cases, indicating that the elliptic weighting linearizes the channel better than the square-root weighting.

Noise histograms for some weighted channel characterizations are shown in Fig. 2.7, where the training data are taken from the center of the page. Clearly, the square-root weighting linearizes the channel better than the linear weighting, but worse than the elliptic weighting. When these plots are compared with Fig. 2.5, the equalizers for both the full response and the 2-D PR1 targets linearize the channel better than the square-root and elliptic weighting methods.

2.3 Threshold Detection

Threshold detectors are simple channel detectors often implemented for channels whose eye-diagrams are open. When the eye-diagram of the channel is closed, more sophisticated detectors are needed such as the ML or MAP detectors to eliminate the ISI in the channel.
Table 2.2 Characterized channel parameters for the normalized target and different weighting functions. Training page size has size 100 × 100. The numbers are rounded to two decimal points.

<table>
<thead>
<tr>
<th>Characterization</th>
<th>$\alpha$</th>
<th>$\alpha h'$</th>
<th>$\alpha k'$</th>
<th>$\zeta$ (dB)</th>
<th>$\sigma'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (center)</td>
<td>5.38</td>
<td>[-0.01 0.05 -0.02]</td>
<td>[-0.12 0.95 0.26]</td>
<td>0 0.14 0.01</td>
<td>1.02</td>
</tr>
<tr>
<td>Linear (corner)</td>
<td>7.22</td>
<td>[0.19 0.26 0.02]</td>
<td>[0.43 0.81 0.22]</td>
<td>-0.02 -0.05 -0.01</td>
<td>1.11</td>
</tr>
<tr>
<td>Square-root (center)</td>
<td>6.31</td>
<td>[-0.01 0.05 -0.01]</td>
<td>[0.10 0.95 0.24]</td>
<td>0 0.13 0.01</td>
<td>2.83</td>
</tr>
<tr>
<td>Square-root (corner)</td>
<td>8.07</td>
<td>[0.18 0.25 0.03]</td>
<td>[0.42 0.83 0.21]</td>
<td>-0.02 -0.05 -0.01</td>
<td>3.32</td>
</tr>
<tr>
<td>Elliptic (center)</td>
<td>6.25</td>
<td>[0 0.05 -0.01]</td>
<td>[0.09 0.96 0.22]</td>
<td>0 0.12 0.01</td>
<td>3.81</td>
</tr>
<tr>
<td>Elliptic (corner)</td>
<td>7.69</td>
<td>[0.18 0.24 0.03]</td>
<td>[0.41 0.84 0.20]</td>
<td>-0.01 -0.05 -0.01</td>
<td>4.37</td>
</tr>
</tbody>
</table>

An example of 2-D channels for which threshold detection can be used is

$$h = \begin{bmatrix} \beta \\ \beta & 1 & \beta \\ \beta \end{bmatrix}$$

when $\beta < 0.25$. The optimal threshold point for this channel is 0 when the channel input is binary.

Consider a 2-D ISI channel with a known impulse response and additive Gaussian noise depending on the channel output (see Fig. 2.8). The correlation of the noise and the data is a common situation in magnetic and holographic recording. In this section, we derive the optimal threshold points for such channels including the possibility that the bipolar input data is not equally likely. This method is also useful for the unknown channels when the threshold points are determined by using the histogram of the channel output symbols. These histograms can be approximated by Gaussian density functions with different variances.
2.3.1 Threshold Detection for Data Dependent Gaussian Noise

Consider a channel with two symbols $\mu_0$ and $\mu_1$ where $\mu_0 < \mu_1$. If the channel symbol values are unknown, they can be estimated roughly as the mean values of the corresponding histogram. The noise in the channel is zero-mean AWG with variances $\sigma_0$ and $\sigma_1$ for symbols $\mu_0$ and $\mu_1$, respectively. Let $p_0$ and $p_1$ be the prior probabilities of the channel symbols $\mu_0$ and $\mu_1$, respectively. Assume a decision rule where if $r > t$, $\mu_1$ is chosen and if $r < t$, $\mu_0$ is chosen. Then the probability of of error, $P_e(t)$, is given by

$$P_e(t) = p_0 P(r > t | d = \mu_0) + p_1 P(r < t | d = \mu_1)$$

$$= p_0 \Phi \left( \frac{\mu_0 - t}{\sigma_0} \right) + p_1 \Phi \left( \frac{t - \mu_1}{\sigma_1} \right)$$

where $\Phi(\cdot)$ is the cumulative distribution function of the unit Gaussian random variable. The probability of error can be minimized subject to the threshold point $t_0$. The probability of error has a minimum at $t_0$ if $P'_e(t_0) = 0$ and $P''_e(t_0) > 0$. The first derivative condition
gives
\[ \frac{p_1}{\sigma_1} f \left( \frac{t - \mu_1}{\sigma_1} \right) = \frac{p_0}{\sigma_0} f \left( \frac{t - \mu_0}{\sigma_0} \right) \] (2.7)
where \( f(\cdot) \) is the density function of the unit Gaussian random variable. Substituting the formula of \( f(\cdot) \) into (2.7) and taking the logarithm of both sides gives
\[ \frac{1}{2} \left[ \left( \frac{t - \mu_1}{\sigma_1} \right)^2 - \left( \frac{t - \mu_0}{\sigma_0} \right)^2 \right] + \ln \frac{p_0 \sigma_1}{p_1 \sigma_0} = 0. \]

Rearranging the terms, we obtain the following quadratic equation
\[ (\sigma_0^2 - \sigma_1^2) t^2 - 2(\sigma_0^2 \mu_1 - \sigma_1^2 \mu_0) t + \left[ \sigma_0^2 \mu_1^2 - \sigma_1^2 \mu_0^2 + 2\sigma_0^2 \sigma_1^2 \ln \frac{p_0 \sigma_1}{p_1 \sigma_0} \right] = 0. \] (2.8)

Let \( a, b, \) and \( c \) be the coefficients of (2.8) such that \( at^2 + bt + c = 0 \). When \( a = 0 \), i.e., \( \sigma_0 = \sigma_1 \triangleq \sigma \), this equation reduces to a linear equation and the solution is given by
\[ t = \frac{\mu_1^2 - \mu_0^2 + 2\sigma^2 \ln \frac{p_0}{p_1}}{2(\mu_1 - \mu_0)}. \] (2.9)

**Example 2.2.** When \( p_0 = p_1 \) and \( \sigma_0 = \sigma_1 \), \( t = (\mu_0 + \mu_1)/2 \) as expected. For unequal probabilities and large variances, the term with the \( \ln \) function becomes significant. For example, when \( \mu_0 = 0 \), \( \mu_1 = 1 \), \( \sigma = 1 \) and \( p_0 = 4p_1 \), \( t = 1.8863 \) that is interestingly larger than \( \mu_1 \).

Equation (2.8) is quadratic when \( a \neq 0 \), i.e., \( \sigma_0 \neq \sigma_1 \) and \( \sigma_0 = \sigma_1 \neq 0 \). The discriminant is given by
\[ \Delta = (\mu_0 - \mu_1)^2 - 2(\sigma_0^2 - \sigma_1^2) \ln \frac{p_0 \sigma_1}{p_1 \sigma_0}. \]

When \( \Delta \geq 0 \), there are two real solutions to the equation given by
\[ t_{1,2} = \frac{(\sigma_0^2 \mu_1 - \sigma_1^2 \mu_0) \pm \sigma_0 \sigma_1 \sqrt{(\mu_0 - \mu_1)^2 - 2(\sigma_0^2 - \sigma_1^2) \ln \frac{p_0 \sigma_1}{p_1 \sigma_0}}}{\sigma_0^2 - \sigma_1^2}. \] (2.10)

The solution minimizing \( P_b(t) \) has to satisfy the second-order derivative condition \( P_b''(t) \geq 0 \). The second derivative of \( P_b(t) \) is given by
\[ P_b''(t) = \frac{p_0}{\sigma_0} f' \left( \frac{\mu_0 - t}{\sigma_0} \right) + \frac{p_1}{\sigma_1} f' \left( \frac{t - \mu_1}{\sigma_1} \right) \]
where \( f'(t) = -(t/\sqrt{2\pi})e^{-t^2/2} \). If both solutions given by (2.10) minimize \( P_b(t) \), then the one giving the global minimum is the best threshold. If \( (p_0 \sigma_1)/(p_1 \sigma_0) = 1 \), the term with the function \( \ln \) can be dropped and the optimal points simplify as
\[ t_1 = \frac{\sigma_0 \mu_1 - \sigma_1 \mu_0}{\sigma_0 - \sigma_1}, \]
and
\[ t_2 = \frac{\sigma_0 \mu_1 + \sigma_1 \mu_0}{\sigma_0 + \sigma_1}. \]

**Example 2.3.** For \( \mu_0 = 0, \mu_1 = 1, \sigma_0 = 1 + \epsilon \) and \( \sigma_1 = 1 - \epsilon \) and \( p_0 = p_1 = 1/2 \), we expect that the optimal threshold is 1/2 for a small value of \( \epsilon \). The first solution becomes \( t_1 = \frac{1+\epsilon}{2\epsilon} \) which is very large, hence \( t_1 \) cannot be a threshold point. The second solution is \( t_2 = \frac{1+\epsilon}{2} \), which is very close to 1/2 as expected.

When \( \Delta < 0 \), there is no minimum or maximum of \( P_b(t) \), i.e., \( P_b(t) \) is a monotone increasing or decreasing function of \( t \). The possible situations are given by
\[
P_b(t) = \begin{cases} 
\text{increasing}, & \text{if } a > 0 \text{ and } c > 0 \\
\text{decreasing}, & \text{if } a < 0 \text{ and } c < 0.
\end{cases}
\]
If \( P_b(t) \) is a monotone increasing function of \( t \), then the threshold becomes \( t = -\infty \). That is, \( P_b(t) \) is minimized when the first symbol is ignored and not detected at all.

### 2.3.2 Threshold Detection with Precoding

A method for designing threshold detectors for 2-D channels with high ISI is to use precoding schemes to open the eye of the channel. In this section, we present a precoding scheme and an optimal threshold detection for the 2-D PR1 channel. The block diagram of the threshold detection with precoding can be seen in Fig. 2.9. A precoding scheme for the 2-D PR1 channel is given by
\[
d_{i,j} = u_{i,j} \oplus d_{i-1,j} \oplus d_{i,j-1} \oplus d_{i-1,j-1}
\]
where \( u_{i,j} \) is a binary input array and \( \oplus \) denotes the binary XOR operator. The channel output \( s_{i,j} \) can be directly related to the channel input array as follows
\[
s_{i,j} = d_{i,j} + d_{i-1,j} + d_{i,j-1} + d_{i-1,j-1}
= u_{i,j} \oplus d_{i-1,j} \oplus d_{i,j-1} \oplus d_{i-1,j-1} + d_{i-1,j} + d_{i,j-1} + d_{i-1,j-1}
\equiv u_{i,j} \pmod{2}.
\]
When there is no noise, the detection rule is given by
\[
u'_{i,j} = \begin{cases} 
0 & \text{if } s_{i,j} = 0, 2, 4 \\
1 & \text{if } s_{i,j} = 1, 3.
\end{cases}
\]
Suppose that AWGN with zero-mean and variance $\sigma^2$ is added to the channel output. In this case, the half-way threshold points, $1/2$, $3/2$, $5/2$ and $7/2$, are not optimal since the probability of the channel symbols are not equal. The optimal threshold points minimizing the bit error probability can be found in a manner similar to the two-symbol case, yet the derivation becomes cumbersome. Instead, we can derive the probability of channel symbol errors $P_s$ which is an upper bound to $P_e$, i.e., $P_s \leq P_e$. When the noise variance for all symbols are small compared to signal separations, this approximation will accurately provide the threshold points.

Consider a channel with five symbols $\mu_i$ such that $\mu_i < \mu_{i+1}$ for $0 \leq i \leq 4$. If the channel symbol values are unknown, they can be estimated roughly as the means of the corresponding histograms. The noise is zero-mean AWGN with variances $\sigma_i^2$ for each symbol $\mu_i$. Let $p_i$ be the probability of the channel symbol $\mu_i$. Let $t_i$ be the threshold point between $\mu_{i-1}$ and $\mu_i$ for $1 \leq i \leq 4$. The probability of channel symbol error is given by

$$P_s = p_0 P(r > t_1|s = \mu_0) + p_1 P(r < t_1|s = \mu_1)$$
$$+ p_1 P(r > t_2|s = \mu_1) + p_2 P(r < t_2|s = \mu_2)$$
$$+ p_2 P(r > t_3|s = \mu_2) + p_3 P(r < t_3|s = \mu_3)$$
$$+ p_3 P(r > t_4|s = \mu_3) + p_4 P(r < t_4|s = \mu_4)$$
$$= p_0 \Phi \left( \frac{\mu_0 - t_1}{\sigma_0} \right) + p_1 \Phi \left( \frac{t_1 - \mu_1}{\sigma_1} \right) + p_2 \Phi \left( \frac{\mu_1 - t_2}{\sigma_1} \right) + p_2 \Phi \left( \frac{t_2 - \mu_2}{\sigma_2} \right)$$
$$+ p_2 \Phi \left( \frac{\mu_2 - t_3}{\sigma_2} \right) + p_3 \Phi \left( \frac{t_3 - \mu_3}{\sigma_3} \right) + p_3 \Phi \left( \frac{\mu_3 - t_4}{\sigma_3} \right) + p_4 \Phi \left( \frac{t_4 - \mu_4}{\sigma_4} \right).$$

The threshold points can be computed as described in the previous subsection. For example, $t_1$ can be computed by considering only the first two terms of the above expression, since the derivative operation eliminates the other terms, which do not depend on $t_1$.

**Example 2.4.** For the 2-D PR1 channel with precoding, the channel symbols are $\mu_i = i$, $0 \leq i \leq 4$, and the probability of these symbols are $p_0 = p_4 = 1/16$, $p_1 = p_3 = 4/16$ and
\( p_2 = 6/16 \) assuming that the input bits are equally likely. The optimal threshold points are computed by using (2.9) for the uniform noise with variance \( \sigma_i^2 = 0.5 \) for all \( i \)'s: \( t_1 = 0.36, t_2 = 1.46, t_3 = 2.54 \) and \( t_4 = 3.64 \).

While determining the threshold points, if any of the discriminants is less than zero, then one of the symbols cannot be detected. The optimal threshold point will be either \( \infty \) or \( -\infty \) suggesting that one of the symbols can be ignored in the detection.

In practice, when the channels are characterized, equalizers do not yield the exact target, but give channel impulse responses close to the target. Consider the following characterized channel impulse response for the 2-D PR1 target

\[
\tilde{h} = \begin{bmatrix} h_0 & h_1 \\ h_2 & h_3 \end{bmatrix}
\]

where \( h_0, h_1, h_2, h_3 \approx 1 \). The threshold detection discussed in this section can still be applied to this channel with a slight loss in the performance. The channel symbol values can be found by averaging for all possible combinations:

\[
\mu_i = \begin{cases} 
0, & \text{if } i = 0 \\
\frac{1}{4} \sum_{j_1} h_{j_1}, & \text{if } i = 1 \\
\frac{1}{6} \sum_{j_1, j_2} h_{j_1} + h_{j_2}, & \text{if } i = 2 \text{ and for } 0 \leq j_1 < j_2 \leq 3 \\
\frac{1}{4} \sum_{j_1, j_2, j_3} h_{j_1} + h_{j_2} + h_{j_3}, & \text{if } i = 3 \text{ and for } 0 \leq j_1 < j_2 < j_3 \leq 3 \\
h_0 + h_1 + h_2 + h_3, & \text{if } i = 4.
\end{cases}
\]

### 2.4 The IMS Algorithm with SISO Detectors

The block diagram of the channel with the IMS detector is shown in Fig. 2.10. The IMS algorithm [5] requires an estimate of the channel impulse response \( h_{i,j}' \) and the noise variance \( (\sigma^2)' \). In this case, precoding is not necessary since the IMS algorithm can eliminate the ISI from the channel output and give the estimate of the channel input \( d_{i,j} \).

In this section, we consider the 2-D PR1 channel with zero-mean AWGN. Figure 2.11 shows the performance comparison between the IMS algorithm and the threshold detector for a 100 \( \times \) 100 data array taken from the center of a page. The IMS detector performs better even after the first iteration.

The IMS algorithm is based on the message-passing algorithm working on adjacent rows of a decoding page. Check nodes can be considered as BCJR algorithms working on the corresponding strips, whereas bit-nodes are connected to the strips that contain the
Figure 2.10 Channel model for IMS detection.

Figure 2.11 The performance of the IMS algorithm and the threshold detector for the 2-D PR1 channel.

same bits. The underlying BCJR algorithm is working on an alphabet of size $2^n$, where $n$ is the strip size (see [5, Ch. 3] for details of the IMS algorithm). The check nodes of the IMS algorithm can be implemented by using any SISO algorithm. Some of the low complexity alternatives to the BCJR algorithm are considered in the following subsections.

The performance of the IMS algorithm will be compared with the 1-D log-MAP detector, which is an optimal detector. The 1-D log-MAP algorithm is implemented by converting the 2-D impulse response into a 1-D impulse response. For the codeword size of $4 \times 4$, the 1-D impulse response corresponding to $h_B$ is given by

$$h = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$
In general, for the codeword size of $M \times N$, the 1-D impulse response is given by

$$ h = \begin{bmatrix} 1 & 1 & 0^{1 \times (N-1)} & 1 & 1 \end{bmatrix} $$

where $0^{1 \times (N-1)}$ is the all-zero row vector of length $N - 1$. When $N$ is large, this algorithm becomes impractical, since the number of states in the decoding trellis grows exponentially with $N$.

### 2.4.1 The IMS Algorithm with the log-MAP Detector

The BCJR algorithm is commonly implemented in the logarithmic domain in order to achieve numerical stability. This algorithm is often referred as the log-MAP algorithm [24].

The performance of the IMS algorithm with the log-MAP detector is shown in Fig. 2.12 for codewords of size $4 \times 4$. After four iterations, the performance of the IMS algorithm becomes indistinguishable from that of the 1-D log-MAP algorithm. Figure 2.13 shows the performance of the IMS algorithm using the log-MAP algorithm for a larger decoding page size of $100 \times 100$. In this case, there are error floors for every iteration probably caused by the short-cycles in the message-passing graph of the IMS algorithm.
2.4.2 The IMS Algorithm with the max-log-MAP Detector

One of the sub-optimal variants of the log-MAP detector is the max-log-MAP detector that replaces the floating-point intensive computations with the maximum function [24]. The simulation for Fig. 2.12 is repeated for the IMS algorithm with the max-log-MAP detector (see Fig. 2.14). The performance of the IMS algorithm with the max-log-MAP detector is very similar to that of the IMS algorithm with the log-MAP detector. However, the underlying max-log-MAP detector has significantly less complexity than the log-MAP detector.

2.4.3 The IMS Algorithm with the SOVA

SOVA was invented by Hagenauer and Hoeher as a low complexity alternative to the log-MAP detector [25]. The SOVA computes cumulative path metrics in the same way that the Viterbi algorithm does. However, it also stores reliability values about input sequences corresponding to each survivor path. Reliability values are updated at each iteration according to the results of add-compare-select operations. Finally, the survivor path and its reliability values are combined to obtain the soft-output information about the input sequence. The performance of the SOVA is observed to be 2 dB worse than the
The max-log-MAP algorithm at a probability of bit error of $10^{-3}$ [26]. Later, the SOVA was extended to be equivalent to the max-log-MAP algorithm for binary codes by Fossorier et al. and for non-binary codes by Cong et al. [27, 28]. In this section, we consider the version of the SOVA which is equivalent to the max-log-MAP algorithm. The performance of the IMS algorithm with the SOVA is almost the same as that of the IMS algorithm with the log-MAP or max-log-MAP detectors as shown in Fig. 2.15.

The complexity of the SOVA increases rapidly with trellis length since at every iteration all reliability values for past input sequences have to be updated. However, in practice this can be limited to a window of size $\delta$. Figure 2.16 shows that a window size of 10 is sufficient to avoid significant performance degradation after 10 iterations.

Figure 2.17 shows the performance comparison for the IMS algorithms working with different detectors when the codeword size is $100 \times 100$. The SOVA detector has window size $\delta = 10$. There is no performance difference between these detectors, although the IMS algorithms with the max-log-MAP and the SOVA detectors have significantly less complexity than the IMS algorithm with the log-MAP detector.
Figure 2.15 Performance of the IMS algorithm using the SOVA. The strip size for IMS algorithm is 2.

2.5 Conclusion

In this chapter, the basic principles of holographic storage systems are introduced. Holographic recording channels are modeled by using the least-squares solutions technique from estimation theory. Linear and non-linear channel models are investigated in order to improve the SNR value of the channel. Optimal threshold detectors for certain channels are implemented along with precoding schemes to avoid tedious 2-D detection problems. The IMS algorithm is modified to work with any SISO detector in order to obtain low complexity detectors for 2-D ISI channels.

2.6 Acknowledgements

The author is grateful to Paul H. Siegel, Zeinab Taghavi, Joseph Soriaga, Mark Ayres and Adrian Hill regarding their helpful discussions and InPhase Technologies for providing measured data from their holographic storage system.
Figure 2.16 Performance of the IMS algorithm with the SOVA for various window sizes. The codeword size is $100 \times 100$. The strip size for the IMS algorithm is 2.

Figure 2.17 Performance comparison of the IMS algorithms with the log-MAP, the max-log-MAP and the SOVA detectors. The number of iterations is 10.
Chapter 3

Characterization and Detection on Oversampled Channels

3.1 Introduction

In Chapter 2, we have discussed channel characterization and detection techniques for 2-D ISI channels. An implicit assumption in this chapter was that all SLM and camera pixels aligned perfectly so that the SLM and camera pixel centers coincide. In practice, this is not true due to misalignments in the optical recording systems. There are reserved blocks for finding the correct alignment across the page. The auto-correlation of a reserved block gives a peak at the center of the reserved block so that the misalignment can be determined. Once the estimate of the misalignment is determined, one way to minimize the effect of different misalignments in the channel is to use oversampling. For an oversampled system, the size of a SLM pixel is larger than that of a camera pixel. The main advantage of this system is that channel variations for different misalignments are reduced.

However, designing detection and equalization schemes for oversampled channels are challenging. In order to facilitate the understanding, we first consider an artificial oversampled channel model — a continuous-time Gaussian channel whose intensity profile is a 2-D Gaussian density function (Section 3.2). It is observed that the channel impulse response for oversampled channels varies periodically across the channel input and camera output arrays. Channel characterization for oversampled channels can be performed similar to non-oversampled channels (Section 3.3). In Section 3.4, an equalization method is discussed for oversampled channels based on the minimum mean-square error (MMSE)
criterion. Detection methods, such as the IMS algorithm or the threshold detector, are difficult to apply for oversampled channels. Instead, we consider a much simpler system in Section 3.5: a MAP detector for 1-D oversampled Gaussian channels. Although the detector is implemented for 1-D Gaussian oversampled channels, it provides a cornerstone to implement the IMS detector for 2-D oversampled channels. Section 3.6 compares the results of full-response equalization with a threshold detector and a MAP detector for 1-D oversampled channels. The performance of the MAP detector for oversampled and non-oversampled channels are observed to be the same when there is no misalignment. When there is a misalignment in the channel, the MAP detector for oversampled channels performs as well as in the case of no misalignment. However, the MAP detector for the non-oversampled channel performs worse than when there is a misalignment in the channel.

3.2 Oversampled Gaussian Channels

We consider a 2-D holographic channel model assuming that the channel intensity profile is a 2-D Gaussian density function. The choice of the 2-D Gaussian function is not related to the underlying physical channel but it makes the analytical derivations more tractable. In fact, most of the derivations shown in this section can be generalized to any 2-D intensity function.

3.2.1 2-D Oversampled Gaussian Channels

The block diagram of an oversampled Gaussian channel is shown in Fig. 3.1. The camera is oversampling at a rate $u/w$, where $u$ and $w$ are assumed to be relatively prime and commonly $w \geq u$. If the SLM pixel size is $\Delta \times \Delta$, then the camera pixel size is $\Delta' \times \Delta'$, where $\Delta' = (w/u)\Delta$. If the channel input $d$ is an $m \times n$ binary array, then the camera output $c$ is an $(mu/w) \times (nu/w)$ array where we assume $w|m$ and $w|n$. An input
bit $d_{i,j} = 1$ is modulated as a shifted version of the 2-D Gaussian density function

$$f(x - i\Delta, y - j\Delta) = \frac{1}{2\pi\epsilon^2} \exp\left(-\frac{(x - i\Delta)^2 + (y - j\Delta)^2}{2\epsilon^2}\right)$$

where $\epsilon$ is called the density parameter. If $d_{i,j} = 0$, then it is modulated as void of signal. The 2-D Gaussian density function is separable into two 1-D Gaussian density functions as $f(x, y) = f(x)f(y)$ where

$$f(x) = \frac{1}{\sqrt{2\pi\epsilon}} \exp\left(-\frac{x^2}{2\epsilon^2}\right).$$

Therefore, the channel output can be represented as

$$s(x, y) = \sum_{i,j} d_{i,j} f(x - i\Delta)f(y - j\Delta).$$

The channel output is perturbed by a continuous-time Gaussian noise $n(x, y)$ with zero-mean and variance $\sigma^2$. The received waveform $r(x, y)$ may have negative values due to the noise $n(x, y)$. However, the occurrence of this problem can be minimized by adding an offset to the channel output. In this model, this problem is ignored for simplicity.

When the centers of some SLM and camera pixels coincide, the SLM and camera planes are called pixel-matched. Note that for oversampled channels, i.e., $u/w \neq 1$, it is impossible to align the SLM and camera planes such that all SLM and camera pixel centers coincide. If there is no oversampling (i.e., $u/w = 1$), every SLM pixel center coincides with one camera pixel center in the pixel-matched case. SLM and camera pixels can shift relative to each other due to the misalignments in the optical system or due to the equalization of the channel for different targets. For instance, the camera pixel center is shifted by $\delta_x = \delta_y = \Delta/2$ in order to obtain the 2-D PR1 target.

Assuming that the camera pixels are shifted by $(\delta_x, \delta_y)$ with respect to the SLM pixels, the camera measures the intensity of the received waveform as follows

$$r_{i,j} = \int_{i\Delta' - \frac{\Delta'}{2} + \delta_x}^{i\Delta' + \frac{\Delta'}{2} + \delta_x} \int_{j\Delta' - \frac{\Delta'}{2} + \delta_y}^{j\Delta' + \frac{\Delta'}{2} + \delta_y} (s(x, y) + n(x, y))dxdy \triangleq s_{i,j} + n_{i,j}.$$ 

where $s_{i,j}$ and $n_{i,j}$ are the signal and noise parts of the camera output. The signal part can be written as a convolution of the channel output with a discrete-time channel impulse response

$$s_{i,j} = \sum_{p,q} d_{p,q} h_x\left(i - \frac{u}{w}p\right) h_y\left(j - \frac{u}{w}q\right)$$

(3.1)
where the channel coefficients are given by
\[
  h_x \left( i - \frac{u}{w}p \right) = \int_{i \Delta' - \frac{\Delta'}{2} + \delta_x}^{i \Delta' + \frac{\Delta'}{2} + \delta_x} f(x - p\Delta) \\
  = \Phi \left( \frac{\Delta'}{\epsilon} \left( i - \frac{u}{w}p + \frac{\delta_x}{\Delta'} + \frac{1}{2} \right) \right) - \Phi \left( \frac{\Delta'}{\epsilon} \left( i - \frac{u}{w}p + \frac{\delta_x}{\Delta'} - \frac{1}{2} \right) \right)
\]
and \( \Phi(\cdot) \) is the cumulative distribution function of the unit Gaussian random variable.

The channel input-output relationship (3.1) is not an ordinary convolution. The channel coefficients multiplying the input bits \( d_{p,q} \) are different for different channel outputs \( s_{i,j} \). In fact, the channel varies periodically along the \( x \) and \( y \) coordinates. For \( k,l \in \mathbb{Z} \), \( s_{i-ku,j-lu} \) and \( s_{i,j} \) are obtained with the same set of channel coefficients:
\[
  s_{i-ku,j-lu} = \sum_{p,q} d_{p,q} h_x \left( i - ku - \frac{u}{w}p \right) h_y \left( j - lu - \frac{u}{w}q \right) \\
  = \sum_{p,q} d_{p,q} h_x \left( i - \frac{u}{w}(p + kw) \right) h_y \left( j - \frac{u}{w}(q + lw) \right) \\
  = \sum_{p,q} d_{p-ku,q-lw} h_x \left( i - \frac{u}{w}p \right) h_y \left( j - \frac{u}{w}q \right).
\]
In other words, there are different impulse responses for different shifts of the impulse functions at the input. Let \( \rho_{p,q} \) be a discrete-time impulse function defined as 1 if \( p = q = 0 \); 0 otherwise. The channel impulse response for the input \( d_{p,q} = \rho_{p-p_0,q-q_0} \) is defined as
\[
  h_{i,j}^{(p_0,q_0)} \triangleq h_x \left( i - \frac{u}{w}p_0 \right) h_y \left( j - \frac{u}{w}q_0 \right)
\]
where \( p_0 \) and \( q_0 \) are fixed positive integers. This function is periodic since for \( k,l \in \mathbb{Z} \),
\[
  h_{i,j}^{(p_0,q_0)} = h_{i-ku,j-lu}^{(p_0-ku_0,lw)}.
\]
Therefore, there are \( w^2 \) different channel impulse responses \( \{ h_{i,j}^{(p_0,q_0)} \} \) for the different shifts of the impulse function, namely \( \{ \rho_{p-ku_0,q-lw} \} \) for \( 0 \leq p_0,q_0 < w \). These channel impulse responses are called forward impulse responses. An arbitrary value of the channel impulse response \( h_{i,j}^{(p,q)} \) can be found by using (3.2) where \( k \) and \( l \) are chosen such that \( 0 \leq p - ku, q - lw < w \).

In practical implementations, it is necessary to limit the size of a forward impulse response. Suppose that \( h_{i,j}^{(p_0,q_0)} \) will be characterized as an \( L_x \times L_y \) array. The range of \( i \) can be chosen as
\[
  -\frac{L_x}{2} \leq i - \frac{u}{w}p_0 + \frac{\delta_x}{\Delta'} < \frac{L_x}{2}.
\]
Rewriting the terms gives the following interval for $i$:

$$\left[-\frac{L_x}{2} + \frac{u}{w}p_0 - \frac{\delta_x}{\Delta'}, \frac{L_x}{2} + \frac{u}{w}p_0 - \frac{\delta_x}{\Delta'}\right].$$

There are exactly $L_x$ integers in this interval. Similarly, the interval for $j$ can be given by

$$\left[-\frac{L_y}{2} + \frac{u}{w}q_0 - \frac{\delta_y}{\Delta'}, \frac{L_y}{2} + \frac{u}{w}q_0 - \frac{\delta_y}{\Delta'}\right].$$

The forward channel responses indicate how the camera pixels are affected by the channel input at a specific location. Alternatively, the channel impulse response can viewed as a weighting function operating on the channel input to obtain a specific channel output. The set of channel weights to obtain a fixed channel output $s_{i_0,j_0}$ is denoted as $\{g_{p,q}^{(i_0,j_0)}\}$ where $g_{p,q}^{(i_0,j_0)} \triangleq h_{i_0,j_0}^{(p,q)}$. This set of channel weights is called a backward impulse response.

Using the relationship (3.2), there are $u^2$ distinct backward impulse responses $\{g_{p,q}^{(i_0,j_0)}\}$ for the different shifts of the output, namely $\{s_{i,j}\}$ for $0 \leq i_0, j_0 < u$. An arbitrary value of the backward channel impulse response $g_{p,q}^{(i,j)}$ can be found by using (3.2) where $k$ and $l$ are chosen such that $0 \leq i - ku, j - lu < u$.

In practical implementations, it is necessary to limit the size of a backward impulse response $g_{p,q}^{(i_0,j_0)}$. Suppose that a forward impulse response has size $L_x \times L_y$. The range of $p$ in the corresponding backward impulse response is given by

$$-\frac{L_x}{2} \leq i_0 - \frac{u}{w}p + \frac{\delta_x}{\Delta'} < \frac{L_x}{2}.$$ 

Rewriting the terms gives the following interval for $p$:

$$\left(\frac{w}{u} \left(-\frac{L_x}{2} + i_0 + \frac{\delta_x}{\Delta'}\right), \frac{w}{u} \left(\frac{L_x}{2} + i_0 + \frac{\delta_x}{\Delta'}\right)\right].$$

There are exactly $(u/w)L_x$ integers in this interval. Therefore, $L_x$ has to be chosen such that $u|L_x$. Similarly, the interval for $q$ can be given by

$$\left(\frac{w}{u} \left(-\frac{L_y}{2} + j_0 + \frac{\delta_y}{\Delta'}\right), \frac{w}{u} \left(\frac{L_y}{2} + j_0 + \frac{\delta_y}{\Delta'}\right)\right].$$

In this case, $L_y$ has to be a multiple of $u$, i.e., $u|L_y$.

It is easy to show that the noise terms $n_{i,j}$ are independent and identically distributed Gaussian random variables with zero mean and variance $(\Delta')^2\sigma^2$.

The SNR value for this channel can be defined as

$$\zeta \triangleq \frac{E[s_{i,j}^2]}{E[\eta_{i,j}^2]} = \frac{1}{4uw} \sum_{i=0}^{u-1} \sum_{j=0}^{u-1} \left[\sum_{p,q} g_{p,q}^{(i,j)} \right]^2 + \sum_{p,q} \left(g_{p,q}^{(i,j)}\right)^2 \right] \right] / (\Delta')^2\sigma^2.$$
Example 3.1. The channel impulse response of a non-oversampled channel with parameters $\Delta = 1$ and $\epsilon = 0.5$ is given by

$$h = \begin{bmatrix}
0.000 & 0.000 & 0.001 & 0.000 & 0.000 \\
0.000 & 0.025 & 0.107 & 0.025 & 0.000 \\
0.001 & 0.107 & 0.466 & 0.107 & 0.001 \\
0.000 & 0.025 & 0.107 & 0.025 & 0.000 \\
0.000 & 0.000 & 0.001 & 0.000 & 0.000
\end{bmatrix}.$$  

The size of this channel impulse response is roughly $3 \times 3$. The SNR value for this channel is $15.00$ dB for $\sigma = 0.1$. For $\Delta = 1$ and $\epsilon = 1$, the channel impulse is given by

$$h = \begin{bmatrix}
0.004 & 0.015 & 0.023 & 0.015 & 0.004 \\
0.015 & 0.058 & 0.093 & 0.058 & 0.015 \\
0.023 & 0.093 & 0.147 & 0.093 & 0.023 \\
0.015 & 0.058 & 0.093 & 0.058 & 0.015 \\
0.004 & 0.015 & 0.023 & 0.015 & 0.004
\end{bmatrix}.$$  

The size of the channel impulse response is at least $(5,5)$. The SNR value for this channel is $14.08$ dB for $\sigma = 0.1$. When there is a misalignment as much as $(\delta_x, \delta_y) = (0.5, 0.5) \Delta$, the channel impulse with parameters $\Delta = 1$ and $\epsilon = 0.5$ becomes

$$h = \begin{bmatrix}
0.018 & 0.046 & 0.046 & 0.018 \\
0.046 & 0.117 & 0.117 & 0.046 \\
0.046 & 0.117 & 0.117 & 0.046 \\
0.018 & 0.046 & 0.046 & 0.018
\end{bmatrix}.$$  

The size of this channel impulse response is roughly $4 \times 4$. The center of this impulse response is similar to that of the 2-D PR1 channel.

Example 3.2. For an oversampled Gaussian channel with parameters $u/w = 1/2$, $\Delta = 1$, $\epsilon = 0.5$, $\delta_x = \delta_y = 0$ and $L_x = L_y = 3$, there are four forward impulse responses:

$$h^{(0,0)} = \begin{bmatrix}
0.001 & 0.022 & 0.001 \\
0.022 & 0.911 & 0.022 \\
0.001 & 0.022 & 0.001
\end{bmatrix}, \quad h^{(0,1)} = \begin{bmatrix}
0 & 0.011 & 0.011 \\
0 & 0.477 & 0.477 \\
0 & 0.011 & 0.011
\end{bmatrix},$$

$$h^{(1,0)} = \begin{bmatrix}
0 & 0 & 0 \\
0.011 & 0.477 & 0.011 \\
0.011 & 0.477 & 0.011
\end{bmatrix}, \quad h^{(1,1)} = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0.25 & 0.25 \\
0 & 0.25 & 0.25
\end{bmatrix}.$$
The corresponding backward impulse response has size $6 \times 6$ and given by

$$g^{(0,0)} = \begin{bmatrix}
0 & 0.011 & 0.022 & 0.011 & 0 & 0 \\
0.011 & 0.250 & 0.477 & 0.250 & 0.011 & 0 \\
0.022 & 0.477 & 0.911 & 0.477 & 0.022 & 0 \\
0.011 & 0.250 & 0.477 & 0.250 & 0.011 & 0 \\
0 & 0.011 & 0.022 & 0.011 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.$$

The SNR value of this channel is 20.51 dB for $\sigma = 0.1$.

Figure 3.2 shows the SNR values for different densities and misalignments for the non-oversampled Gaussian channel. The pixel-matched case is superior to the other cases when $\epsilon \leq 0.7$. Figure 3.3 shows the SNR values for different oversampling ratios and densities. When the density of the channel increases, the performances of oversampled channels become slightly worse. Figure 3.4 shows the same plot for different values of misalignments between camera and SLM pixels. In this case, the SNR does not vary for different values of shifts. These SNR plots are misleading when comparing the performance of a MAP detector for oversampled and non-oversampled channels. In Section 3.5, simulations using a MAP detector indicate that the oversampled Gaussian channel with ratio $4/3$ has similar performance compared with the non-oversampled channel.

### 3.2.2 1-D Oversampled Gaussian Channels

The 1-D oversampled Gaussian channel is a simplified version of the 2-D oversampled Gaussian channel and is used to investigate the basic properties of oversampled channels. The block diagram of a 1-D oversampled Gaussian channel is shown in Fig. 3.5. The camera is oversampled at the rate $u/w$, where $u$ and $w$ are assumed to be relatively prime. If the SLM pixel size is $\Delta$, then the camera pixel size is $\Delta'$, where $\Delta' = (w/u)\Delta$. If the channel input sequence $d$ has length $m$, then the camera output sequence $c$ has length $(u/w)m$ such that $w|m$. Assuming that the camera pixels are shifted by $\delta_x$ with respect to the SLM pixels, the camera output sequence is given by

$$r_i = \int_{i\Delta' - \frac{\Delta'}{2} + \delta_x}^{i\Delta' + \frac{\Delta'}{2} + \delta_x} (s(x) + n(x)) dx \triangleq s_i + n_i.$$

The signal part of the camera output can be written as

$$s_i = \sum_p d_ph \left( i - \frac{u}{w} p \right) \quad (3.4)$$
where
\[ h \left( i - \frac{u}{w} p \right) = \int_{i \Delta' - \frac{\Delta'}{2} + \delta_x}^{i \Delta' + \frac{\Delta'}{2} + \delta_x} f(x - p \Delta) \]
\[ = \Phi \left( \frac{\Delta'}{\epsilon} \left( i - \frac{u}{w} p + \frac{\delta}{\Delta'} + \frac{1}{2} \right) \right) - \Phi \left( \frac{\Delta'}{\epsilon} \left( i - \frac{u}{w} p + \frac{\delta}{\Delta'} - \frac{1}{2} \right) \right). \]

The forward and backward impulse responses for this channel can be defined in a similar manner to the 2-D oversampled Gaussian channel. This channel varies periodically such that for \( k \in \mathbb{Z} \), \( s_{i-kw} \) and \( s_i \) are obtained with the same set of channel coefficients. The forward impulse response \( h_i^{(p_0)} \) is defined as the response of the channel to the shifted impulse function \( \rho_{p-p_0} \). There are \( w \) different forward impulse responses \( \{h_i^{(p_0)}\} \) for different shifts of the impulse function. For \( 0 \leq p_0 < w \), the forward impulse response is given by
\[ h_i^{(p_0)} \triangleq h \left( i - \frac{u}{w} p_0 \right). \]

This function is periodic since for \( k \in \mathbb{Z} \),
\[ h_i^{(p_0)} = h_i^{(p_0-kw)}. \tag{3.5} \]

An arbitrary value of the channel impulse response \( h_i^{(p)} \) can be found by using (3.5) where \( k \) is chosen such that \( 0 \leq p - kw < w \).
Figure 3.3 SNR values for oversampled Gaussian channels for different densities and oversampling ratios in the pixel-matched case where $\Delta = 1$, $\sigma = 0.1$.

The backward impulse response is defined as $g_p^{(i_0)} \triangleq h_{i_0}^{(p)}$ for all values of $p$ and $0 \leq i_0 < u$. There are $u$ distinct backward impulse responses $\{g_p^{(i_0)}\}$ for the different shifts of the output, namely $\{s_{i_0}\}$ for $0 \leq i_0 < u$. Practical implementations of the forward and the backward impulse responses can be described in a similar manner as in Subsection 3.2.1.

It is easy to show that the noise terms $n_i$ are independent and identically distributed Gaussian random variables with zero mean and variance $(\Delta')\sigma^2$. The SNR value for this channel can be derived as

$$\zeta \triangleq \frac{E[s_i^2]}{E[n_i^2]} = \frac{1}{4u} \sum_{i=0}^{u-1} \left[ \left( \sum_p g_p^{(i)} \right)^2 + \sum_p \left( g_p^{(i)} \right)^2 \right] \Delta'\sigma^2.$$

### 3.3 Channel Characterization on Oversampled Channels

The channel parameters for oversampled channels can be characterized in a similar manner as for non-oversampled channels. Let $d_{p,q}$ be a channel input array and $r_{i,j}$ be the corresponding camera output array. For an oversampled channel at ratio $u/w$, the channel
Figure 3.4 SNR values for oversampled Gaussian channels for different values of misalignments and oversampling ratios where $\Delta = 1$, $\sigma = 0.1$, and $\epsilon = 0.5$.

input-output relationship is given by

$$r_{i,j} = \sum_{p,q} d_{p,q} g_{p,q}^{(i,j)} + k + n_{i,j}$$

where $k$ is the offset and the noise is assume to be AWG. In this equation, the training array $d_{p,q}$ and $r_{i,j}$ are known. Therefore, one can estimate the unknown channel coefficients using the least-squares solution similar to Section 2.2. In practice the span of a backward impulse response can be limited to $(w/u) L_x \times (w/u) L_y$ where $u|L_x$ and $u|L_y$. For such a size limitation, the ranges of $p$ and $q$ are discussed in Subsection 3.2.1. For a training input array of size $Mw \times Nw$, there are $MN$ equations involving a backward impulse response $g_{p,q}^{(i,j)}$ since the backward impulse responses are periodic with period $u$. When $M \gg (w/u)L_x$ and $N \gg (w/u)L_y$, there is sufficient information to estimate the backward

Figure 3.5 1-D oversampled Gaussian channel model.
impulse responses. After estimating the backward impulse responses, one can estimate the SNR value as

$$\zeta' = \frac{\text{Total signal power}}{\text{Total noise power}} = \frac{\sum_{i,j} \sum_{p,q} d_{p,q} (g'_{p,q}(i,j))^2}{\sum_{i,j} (n'_{i,j})^2}.$$ 

Using the SNR estimate, we can estimate the continuous time noise variance as follows

$$\sigma'^2 = \frac{1}{4\pi^2} \sum_{u=0}^{u-1} \sum_{v=0}^{v-1} \left[ \left( \sum_{p,q} (g'_{p,q}(i,j))^2 \right)^2 + \sum_{p,q} (g'_{p,q}(i,j))^2 \right].$$

**Example 3.3.** We consider characterizing an oversampled Gaussian channel with parameters given in Example 3.2. The backward impulse response is estimated as follows

$$\begin{bmatrix}
0.010 & 0.008 & 0.029 & 0.019 & -0.007 & -0.002 \\
0.008 & 0.246 & 0.482 & 0.256 & -0.001 & -0.005 \\
0.021 & 0.481 & 0.920 & 0.482 & 0.018 & -0.009 \\
0.012 & 0.251 & 0.484 & 0.254 & 0.011 & 0.001 \\
0.014 & 0.010 & 0.022 & 0.012 & 0.002 & -0.009 \\
-0.008 & -0.001 & -0.001 & -0.004 & 0.003 & 0.005
\end{bmatrix}.$$ 

The SNR and noise variance estimates are $\zeta' = 20.63$ dB and $(\sigma'^2) = 0.0107$. Note that, these results are very close to the true values of the channel. Figure 3.6 depicts the true and the characterized SNR values for different oversampling ratios. The SNR estimates are very close to the true channel parameters for both the pixel-matched and misaligned cases.

**Example 3.4.** We now consider characterizing an unknown channel with oversampling ratio of $4/3$. The channel is characterized with the parameters $L_x = L_y = 4$ and $M = N = 50$ by using the data taken from a corner, an edge and the center of the pages. The backward channel impulse response estimates for the center of the data page is tabulated in Table 3.1. Table 3.2 shows the characterized channel parameters for different data sets. The channel parameters of the equalized channel of the full response target are also shown for comparison. It can be observed that the oversampled channel model is more appropriate for the channel than the full-response equalizer in terms of the SNR values of the resulting channels. However, this characterization is weak for the data sets taken from the corners of pages.

In all estimates, the continuous-time noise level is very high, which makes the detector performance worse. Clearly, the full-response target has a low continuous-time noise level
Figure 3.6 True and characterized SNR values for oversampled Gaussian channel where $\Delta = 1$, $\sigma = 0.1$, $\epsilon = 0.5$, $M = N = 50$.

but low SNR values. For the center of the data page, the channel impulse response estimate for the full-response target is given by

\[
    h^{(0,0)} = g^{(0,0)} = \begin{bmatrix}
        -0.009 & 0.002 & -0.009 \\
        -0.014 & 0.429 & -0.008 \\
        -0.005 & -0.022 & -0.008
    \end{bmatrix}.
\]

3.4 Minimum Mean-Square Equalizers

This section discusses an equalization method based on the MMSE criterion. This equalizer matches the discrete-time channel impulse response to a predefined target such that equalizer coefficients minimize the MMSE. We attempt to analyze this method in the context of oversampled channels.

The method is based on filtering the camera output $r_{i,j}$ by a finite impulse response (FIR) filter $c_{i,j}$ to estimate the channel input sequence $d_{p,q}$. The center of the weighting filter for the input estimate $d'_{p,q}$ will be the channel output which is the most affected position from $d_{p,q}$. This position can be found by feeding the channel with an impulse function shifted by $(p, q)$ and taking the location where the channel output is maximum.
Table 3.1 Backward impulse response estimates \((g^{i,j}_{p,q})\) for data sets taken from the center of the pages.

<table>
<thead>
<tr>
<th>(i) (\backslash j)</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.01</td>
<td>0.08</td>
<td>-0.02</td>
</tr>
<tr>
<td></td>
<td>0.37</td>
<td>0.34</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>-0.02</td>
<td>-0.03</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.03</td>
<td>0.02</td>
</tr>
<tr>
<td>1</td>
<td>0.08</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>-0.01</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>-0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>0.20</td>
<td>0.05</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>-0.02</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>-0.04</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 3.2 Characterized channel parameters and channel parameters for the full response target.

<table>
<thead>
<tr>
<th></th>
<th>Characterized channel</th>
<th>Full response target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\zeta') (dB)</td>
<td>(\sigma')</td>
</tr>
<tr>
<td>Corner</td>
<td>3.83</td>
<td>0.167</td>
</tr>
<tr>
<td>Edge</td>
<td>11.20</td>
<td>0.101</td>
</tr>
<tr>
<td>Center</td>
<td>11.56</td>
<td>0.098</td>
</tr>
</tbody>
</table>

Therefore the optimal position can be expressed by

\[
(i^*, j^*) = \arg \max_{i,j} h_x \left( i - \frac{u}{w} p \right) h_y \left( j - \frac{u}{w} q \right).
\]

The function \(h_x(i - (u/w)p)\) is a maximum when \(i = (u/w)p - \delta_x/\Delta'\). However \(i\) is integer, so it is necessary to take the nearest point \([ (u/w)p - \delta_x/\Delta' ]\), where the rounding function is defined as

\[
[a] = \begin{cases} 
  [a], & \text{if } a - [a] < 0.5 \\
  [a], & \text{if } a - [a] \geq 0.5.
\end{cases}
\]

Therefore, the optimal position for the filter center is \((\theta_p, \theta_q)\) where \(\theta_p \triangleq [(u/w)p - \delta_x/\Delta']\) and \(\theta_q \triangleq [(u/w)q - \delta_y/\Delta']\).

The estimated input array can be written as

\[
d'_{p,q} = \sum_{i,j} c_{i,j} r_{i+\theta_p,j+\theta_q}
\]

\[
= \sum_{p',q'} d'_{p',q'} \sum_{i,j} c_{i,j} h_x \left( i + \theta_p - \frac{u}{w} p' \right) h_y \left( j + \theta_q - \frac{u}{w} q' \right) + \sum_{i,j} c_{i,j} \tilde{n}_{i+\theta_p,j+\theta_q}
\]

\[
\triangleq \sum_{p',q'} d'_{p',q'} \tilde{h}_{p-p',q-q'} + \tilde{n}_{p,q}.
\]
The discrete-time channel impulse response $\tilde{h}$ can be matched to a predefined target by choosing the coefficients $c_{i,j}$ such that MMSE is minimum, i.e.,

$$\|d - d'\|^2 = \sum_{p,q} (d_{p,q} - d'_{p,q})^2$$

is minimum. However, if the true noise in the system is Gaussian, then the equalized noise $\tilde{n}_{p,q}$ is colored. Therefore the detection algorithm, such as the IMS algorithm, will not be the best performing detector for this channel.

The design of minimum mean-square equalizer for the 1-D oversampled Gaussian channels can be accomplished in a manner similar to the 2-D case. After the channel output is equalized, threshold detection can be applied to obtain the estimated input sequence as discussed in Section 2.3.

**Example 3.5.** Consider an unknown channel with oversampling ratio $4/3$ equalized to the 2-D PR1 and full-response targets. Equalizer filters have 16-taps, since $L_x = L_y = 4$. The data set sizes are $150 \times 150$ and $200 \times 200$ for the channel input and camera output, respectively. The normalized channel parameters are shown in Table 2.1.

### 3.5 A MAP Detector for 1-D Oversampled Channels

In this section, we propose a MAP detector implementation for 1-D oversampled channels. This case is much simpler than the implementation of the IMS algorithm for 2-D oversampled channels, yet the complete implementation of the IMS algorithm can be performed based on the discussion here.

Since the underlying channel impulse response is periodic, the trellis for the MAP algorithm is also periodic. For an oversampling ratio of $u/w$, there are $u$ outputs corresponding to $w$ input bits. Therefore, one can design a trellis with $2^w$ states where each state has $2^w$ edges with input-output symbols of size $w$ and $u$, respectively. Obviously, this trellis eliminates the time-varying nature of the channel, but it is impractical when $w$ is large. It is therefore of interest to design low complexity time-varying alternatives to this trellis. Let $d_p$ be a binary channel input sequence of length $wM$ and let $r_i$ be the camera output with length $uM$.

An example of the channel input-output diagram is shown in Fig. 3.7. The periodic stages of the diagram are shown with dashed lines. This diagram has the following properties:
Figure 3.7 The input-output diagram for an oversampled channel for $u/w = 4/3$ and $L_x = 4$.

- The first input bit of the first periodic stage is assumed to be 0. There should be 2 zero input bits before the first input data $d_0$.

- It is required to have 2 tailing zeros at the end of input sequence to terminate the trellis at the zero state. Therefore, there are 3 input bits while there are 8 outputs.

Using the input-output diagram, we can define the state of the MAP algorithm as a 2-tuple $\sigma^{(i)} = \{d_{i-1}, d_i\}$, where the memory of the channel is $\mu = 1$. There exist a edge from state $\sigma^{(i)} = \{d_{i-1}, d_i\}$ to $\sigma^{(i+1)} = \{d'_i, d'_{i+1}\}$ if $d'_i = d_i$. Note that there may be more than one output for some input bits. For example, the edge from $\sigma^{(1)} = \{d_0, d_1\}$ to $\sigma^{(2)} = \{d_1, d_2\}$ has two outputs $r_2$ and $r_3$ as shown in Fig. 3.7.

The trellis model given in this example can be generalized as follows:

- The channel memory is $\mu = (w/u)L_x - 1$ where $u|L_x$.

- The first input data bit is the second bit in the first stage of the trellis. There should be $\mu$ zeros before the data sequence including the first bit of the first stage. Likewise, there should be $\mu$ trailing zeros at the end of the channel input. In this way, we guarantee that the trellis starts and end with the all-zero state. The number of channel input and output bits are $wM - \mu$ and $uM$, respectively.

The MAP detector needs to be modified for this periodic trellis. The only modification required is the computation $\gamma_k(p, q)$ metric, where $k$ is the index of the input, $p$ and $q$ are previous and next states, respectively. Let $\{s_i\}$ and $\{r_i\}$ be the set of channel outputs and camera outputs corresponding to the edge between $p$ and $q$ at time index $k$, respectively.
The metric for each output is multiplied to obtain the cumulative metric as follows

\[
\gamma_k(p, q) = \prod_i \exp\left\{ -\frac{1}{2\sigma^2} |r_i - s_i|^2 \right\} P(d_k)
\]

\[
= P(d_k) \exp\left\{ -\sum_i \frac{1}{2\sigma^2} |r_i - s_i|^2 \right\}
\]

where \(P(d_k)\) is the priori probability of the input bit \(d_k\). The backward and forward metrics of the MAP detector stay the same. In practice, this modified MAP detector can be implemented in the log domain similar to the log-MAP detector.

### 3.6 Simulation Results

We consider two detection schemes for the 1-D oversampled Gaussian channels: 3-tap full-response equalization with threshold detector as discussed in Sections 2.3 and 3.4, and the log-MAP detector discussed in Section 3.5.

The performance of both detectors are compared for two channels: the 1-D non-oversampled Gaussian channel and the 1-D oversampled Gaussian channel at ratio \(u/w = 4/3\). Both channels are simulated with the same continuous-time noise variance and some parameters of both channels are \(\Delta = 1\), \(\epsilon = 0.5\). For all simulations, the channel inputs are equally likely binary sequences of length 150 and each simulation is repeated 10,000 times.

The performances of the detectors for the pixel-matched case are shown in Fig. 3.8. The performances of the oversampled and the non-oversampled Gaussian channels are interestingly the same for the log-MAP detector. However, the performance of the full-response equalizer with the threshold detector is poor compared with the log-MAP algorithm in both the oversampled and the non-oversampled case.

Figure 3.9 shows the performance variations of the log-MAP detector for known misalignments between the SLM and camera pixels. The log-MAP detector is adapted to the misaligned version of the channel. Obviously, the oversampled channel does not suffer from these known misalignments, while the non-oversampled channel has a performance loss of about 1 dB. Fig. 3.10 shows the results of the same simulation for unknown misalignments. The log-MAP detector is not adapted to the misaligned version of the channel, i.e., the detector uses the trellis for the pixel-matched case. In this case, the log-MAP detector performs significantly worse when \(\delta_x\) is close 0.5\(\Delta\) for both channels.
Figure 3.11 shows the performance of the full-response equalizer with a threshold detector for known misalignments between SLM and camera pixels. When $\delta_x$ is small, the performance for the non-oversampled channel is better. However, when $\delta_x$ is close to $0.5\Delta$, the performance for the oversampled channel is better. The reason is that the non-oversampled channel requires longer equalizer filters when $\delta_x$ is large. Similar to the log-MAP detector, this detector is not significantly affected by the variations of the known shifts for the oversampled channel. Similar to the log-MAP detector, when there are unknown misalignments in the channels, this detector performs poorly for both non-oversampled and oversampled channels. (see Fig. 3.12).

3.7 Conclusion

We have discussed the effect of oversampling for 2-D ISI channels. The oversampled channel model is discussed by considering a toy channel with a Gaussian intensity profile. Well-known channel characterization and equalization methods are generalized to oversampled channels. A MAP detector tuned for the 1-D Gaussian oversampled channel is also presented. Although oversampling does not improve the detector performance in terms of the bit-error-rate, it provides immunity to the detector for different misalignments in the channel.

3.8 Acknowledgements

The author is grateful to Paul H. Siegel, Zeinab Taghavi, Mark Ayres and Adrian Hill regarding their helpful discussions and InPhase Technologies for the simulation data.
Figure 3.8 Performance of the detectors for the non-oversampled and oversampled 1-D Gaussian channels in the pixel-matched case.
Figure 3.9 The performance of the log-MAP detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channels for known misalignments.
Figure 3.10 The performance of the log-MAP detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channels for unknown misalignments.
Figure 3.11 The performance of the full-response equalizer with the threshold detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channel for known misalignments.
Figure 3.12 The performance of the full-response equalizer with the threshold detector for the (a) non-oversampled and (b) oversampled 1-D Gaussian channel for unknown misalignments.
Chapter 4

Error Event Characterization on 2-D ISI Channels

4.1 Introduction

Detection and coding for 2-D ISI channels have been the subject of much research recently because of advances in holographic storage technology. A generalization of 1-D detection and coding methods to 2-D channels is not trivial due to the lack of convenient graph-based descriptions of such channels. In particular, there is no simple trellis-based maximum likelihood (ML) detection algorithm analogous to the 1-D Viterbi algorithm.

However, there are suboptimal detection techniques such as the IMS algorithm for 2-D ISI channels that demonstrate very good error-rate performance and appear to approximate the performance of an ML detector [5]. It is therefore of interest to identify the dominant 2-D error events, where we define an error event as the difference between the recorded and the decoded data arrays. In the 2-D setting, error events can be classified as closed or open depending on whether the area of the smallest square region containing nonzero differences is bounded or unbounded. Empirical evidence has shown that data arrays forming dominant error events for the IMS algorithm generate channel outputs with small squared-Euclidean distance. Therefore, it is important to characterize the 2-D error events with small squared-Euclidean distance, so that 2-D distance-enhancing constrained codes can be designed to improve system performance [29].

Karabed et al. introduced an analytic method to characterize the distance properties of some 1-D partial-response channels [30]. In this chapter, we extend this method to
characterize the closed error events of some 2-D ISI channels. In particular, we study the
2-D PR1 channel whose impulse response is given by

\[ h = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \] (4.1)

The analytic method used for characterizing error events for the 2-D PR1 channel
is tedious to apply for most 2-D channels, particularly for the channels whose impulse
responses span \( u \times v \) arrays where \( u, v > 2 \). For 1-D ISI channels, efficient search algorithms
working on error-state diagrams have been developed to characterize error events for high
order partial response channels \([31, 32]\). Error-state diagrams for 2-D ISI channels can be
generated by fixing the size of the error event in the horizontal or vertical direction. In this
chapter, we propose a bounded depth-search algorithm for finding closed error events for
any 2-D ISI channel. The complexity of the algorithm solely depends on the underlying
2-D ISI channel.

This chapter is organized as follows. In Section 4.2, we present a 2-D ISI channel model
and define its distance properties. In Section 4.3, the characterization of minimum and
near-minimum distance error events of the 2-D PR1 channel is investigated by studying
the channel impulse response in the spectral domain. By generalizing this concept, some
distance properties of the channels other than the 2-D PR1 channel can be found (see
Section 4.4).

The method of precoding is commonly used in 1-D recording systems to invert the ISI
effect of the channel. In Subsection 4.3.4, we discuss the effect of a precoding scheme on
error events for the 2-D PR1 channel. For 1-D channels, the probability of error event and
bit error can be bounded from above by the union bound. In Section 4.5, we generalize this
concept to 2-D channels. Error state diagrams and a bounded depth-search algorithm are
introduced in Section 4.6. Analytical results are compared with the simulated results for
the 2-D PR1 channel. In Section 4.7, we propose a computational lower bound on distance
of error events for any 2-D ISI channel. This method generalizes the concepts presented
in Sections 4.3 and 4.4 to any 2-D ISI channel.

4.2 The 2-D ISI Channel

Consider a 2-D ISI channel with bipolar input array \( x = \{x_{i,j}\} \), channel impulse
response \( h = \{h_{i,j}\}_{i=0,j=0}^{u-1,v-1} \), and output \( y = x * h \) (see Fig. 4.1). Additive white Gaussian
noise $\eta = \{\eta_{i,j}\}$ with zero mean and variance $\sigma^2$ is added to the channel output array to obtain the received array $r = \{r_{i,j}\}$.

For a channel output array $y$ and its estimated array $y'$, the normalized squared-Euclidean distance is defined as

$$d^2(y, y') \triangleq \sum_{i,j} [(y_{i,j} - y'_{i,j})/2]^2.$$

which is taken to be $\infty$ if the sum is unbounded. Normalized squared-Euclidean distances will be referred as *squared distances*. The quantity $d^2(y, y')$ can be expressed in terms of the corresponding input arrays $x$ and $x'$, respectively,

$$d^2(y, y') = d^2(\epsilon * h, 0)$$

where $\epsilon = (x - x')/2 \in \{0, +1, -1\}^*$ is the normalized channel input error array. The normalized channel input error arrays are called *error events*, whose elements are commonly represented by the symbols $\{0, +, -\}$. The *distance* of an error event $\epsilon$ is defined as $d(\epsilon) \triangleq \sqrt{d^2(\epsilon * h, 0)}$.

Analogous to the 1-D channels, the error events for 2-D channels are classified as either open or closed. An error event is *closed* if the area of the smallest square region containing nonzero differences is bounded. Error events which are not closed are called *open*. Let $E_{\text{closed}}$ be the set of closed error events, $E_{\text{open}}$ be the set of open error events, and $E = E_{\text{closed}} \cup E_{\text{open}}$ be their union. We define the *minimum closed event distance* $d_{<\rangle} = \min_{\epsilon \in E_{\text{closed}}} d(\epsilon * h, 0)$ and the *minimum event distance* $d_{<} = \min_{\epsilon \in E} d(\epsilon * h, 0)$.

1-D sequences are often represented in the $D$-transform domain, which is equivalent to the $z$-transform where $D = z^{-1}$. Likewise, 2-D arrays can be represented in the $(D, E)$-
transform domain. For an array $x$, the $(D, E)$-transform of $x$ is defined as
\[ x(D, E) \triangleq \sum_{i,j} x_{i,j} D^i E^j. \]

In this representation, the impulse response of the 2-D PR1 channel is given by
\[ h(D, E) = 1 = D + E + DE \]
and the channel input-output relationship becomes
\[ y(D, E) = x(D, E)h(D, E). \] (4.2)

The minimum closed event distance of a 2-D ISI channel can be expressed as
\[ d_{<} = \min_{\epsilon(D,E) \neq 0, \epsilon \in \mathcal{E}_{\text{closed}}} ||h(D,E)\epsilon(D,E)|| \]
where
\[ \epsilon(D,E) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \epsilon_{i,j} D^i E^j, \quad \epsilon_{i,j} \in \{-1,0,+1\} \] (4.3)
is the polynomial which corresponds to a normalized input error array $\epsilon = \{\epsilon_{i,j}\}_{i=0,j=0}^{m-1,n-1}$ such that the edges of $\epsilon$, $\epsilon_{0,s}$, $\epsilon_{s,0}$, $\epsilon_{m-1,s}$, $\epsilon_{s,n-1}$ contain at least one non-zero entry.

Here, we have defined error events for 2-D channels in a way analogous to 1-D channels. However, error events for 2-D channels cannot be interpreted as differences of paths in the decoding trellis, since such trellis representations do not exist for 2-D channels.

### 4.3 Error Event Characterization on the 2-D PR1 Channel

The minimum and near minimum distance error events play an important role in determining the performance of an ML detector. Table 4.1 shows some error events along with the percentage of bit errors caused by these error events for the 2-D PR1 channel. The detector used in this simulation is the IMS algorithm working on $10 \times 10$ codewords. It is clear that the error events with squared distance 4 dominate the performance when the SNR increases. Therefore, the minimum squared distance of this channel is at most 4. In fact, we will prove in the following subsection that this is indeed the case.

The minimum and near-minimum distance error events can be characterized by studying spectral properties of the channel transfer function and the corresponding limitations on error coefficients, $\{\epsilon_{i,j}\}$ [30, Sec. III.A]. Using this method, the minimum distance error events (distance-2) are now completely characterized for the 2-D PR1 channel. In addition, the error events with squared distance 0 and 6 are partially characterized.
Table 4.1 The percentage of bit errors caused by some error events for the 2-D PR1 channel. The bit error rates are $5.45 \times 10^{-2}$, $3.11 \times 10^{-2}$, and $1.46 \times 10^{-2}$ for 10 dB, 11 dB, and 12 dB, respectively.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$d^2(\epsilon)$</th>
<th>$\zeta = 10$ dB</th>
<th>$\zeta = 11$ dB</th>
<th>$\zeta = 12$ dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[+]$</td>
<td>4</td>
<td>16.69</td>
<td>15.78</td>
<td>14.95</td>
</tr>
<tr>
<td>$[+]$</td>
<td>4</td>
<td>8.66</td>
<td>9.46</td>
<td>11.14</td>
</tr>
<tr>
<td>$[+ \ - \ ]$</td>
<td>4</td>
<td>10.25</td>
<td>10.60</td>
<td>11.34</td>
</tr>
<tr>
<td>$[+ \ - \ + \ ]$</td>
<td>4</td>
<td>4.53</td>
<td>5.32</td>
<td>6.02</td>
</tr>
<tr>
<td>$[+ \ - \ + \ ]$</td>
<td>4</td>
<td>4.68</td>
<td>5.45</td>
<td>5.77</td>
</tr>
<tr>
<td>$[+ \ - \ + \ - \ ]$</td>
<td>4</td>
<td>2.02</td>
<td>2.46</td>
<td>3.46</td>
</tr>
<tr>
<td>$[+ \ - \ ]$</td>
<td>4</td>
<td>2.02</td>
<td>2.74</td>
<td>3.35</td>
</tr>
<tr>
<td>$[+ \ - \ ]$</td>
<td>4</td>
<td>2.39</td>
<td>3.00</td>
<td>3.4</td>
</tr>
<tr>
<td>$[+ \ 0 \ - \ ]$</td>
<td>6</td>
<td>1.02</td>
<td>0.85</td>
<td>0.60</td>
</tr>
<tr>
<td>$[0 \ + \ - \ ]$</td>
<td>6</td>
<td>0.95</td>
<td>0.78</td>
<td>0.44</td>
</tr>
</tbody>
</table>

4.3.1 Minimum Distance Error Events

**Proposition 4.1.** The minimum closed event distance of the 2-D PR1 channel is 2. All distance-2 closed error events are of the form

\[
\epsilon = \begin{bmatrix}
+ & - & \cdots & \epsilon_{0,n-1} \\
- & + \\
& & \vdots & \\
\epsilon_{m-1,0} & \cdots & \epsilon_{m-1,n-1}
\end{bmatrix}
\]

and their negatives. Here $\epsilon_{m-1,0} = + (\epsilon_{0,n-1} = +)$ if $m$ (n) is odd; otherwise $\epsilon_{m-1,0} = - (\epsilon_{0,n-1} = -)$. The bottom right entry is determined as $\epsilon_{m-1,n-1} = \epsilon_{m-1,0}\epsilon_{0,n-1}$.

The proof of the proposition is based on an expansion of (4.2) using the definition of closed error events (4.3). Let $y(D,E)$ be the channel output to a closed error array $\epsilon(D,E)$. For $m, n \geq 2$, $y(D,E)$ can be expanded as follows

\[
y(D,E) = (1 + D + E + DE) \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \epsilon_{i,j}D^iE^j
\]

\[
y(D,E) = y_1(D,E) + y_2(D,E) + y_3(D,E)
\]

where $y_1(D,E)$ contains the terms with a single error coefficient, which correspond to the corners of the error event:

\[
y_1(D,E) = \epsilon_{0,0} + \epsilon_{m-1,0}D^m + \epsilon_{0,n-1}E^n + \epsilon_{m-1,n-1}D^mE^n.
\]
Each of the terms in the second group, \( y_2(D, E) \), has two error coefficients corresponding to the edges of the error event:

\[
y_2(D, E) = \sum_{i=1}^{m-1} (\epsilon_{i,0} + \epsilon_{i-1,0})D^i + \sum_{j=1}^{n-1} (\epsilon_{0,j} + \epsilon_{0,j-1})E^j + \sum_{i=1}^{m-1} (\epsilon_{i,n-1} + \epsilon_{i-1,n-1})D^iE^n + \sum_{j=1}^{n-1} (\epsilon_{m-1,j} + \epsilon_{m-1,j-1})D^mE^j.
\]

Each of the terms in the third group, \( y_3(D, E) \), has four error coefficients corresponding to the middle of the error event:

\[
y_3(D, E) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} (\epsilon_{i,j} + \epsilon_{i-1,j} + \epsilon_{i,j-1} + \epsilon_{i-1,j-1})D^iE^j.
\]

Considering the terms in \( y_1(D, E) \) and \( y_2(D, E) \), one can prove the following results.

**Lemma 4.1.** Depending on the size of the error event, the squared distance of \( y_1(D, E) + y_2(D, E) \) is bounded from below as follows:

1. If \( m, n \geq 2 \), \( \|y_1(D, E) + y_2(D, E)\|^2 \geq 8 - (|\epsilon_{0,0}| + |\epsilon_{m-1,0}| + |\epsilon_{0,n-1}| + |\epsilon_{m-1,n-1}|) \).

2. For other values of \( m \) and \( n \), \( \|y_1(D, E) + y_2(D, E)\|^2 \geq 4 \).

**Proof.** Part 1. If all corners of an error event are non-zero, then \( \|y_1(D, E)\|^2 = 4 \). If any of the corners is zero, then \( \|y_1(D, E)\|^2 \) decreases by 1 while \( \|y_2(D, E)\|^2 \) increases by 2. To understand this, let \( \epsilon_{0,0} = 0 \). Let \( j' \leq n - 1 \) be the index of the first non-zero entry of the first row, i.e., \( \epsilon_{0,j'} \neq 0 \) and \( \epsilon_{0,j} = 0 \) for \( 0 \leq j < j' \). Similarly, we can define \( i' \leq m - 1 \) as the index of the first non-zero entry on the first column. The first term in \( y_1(D, E) \) disappears, whereas \( y_2(D, E) \) introduces two new terms with single coefficients which are not zero: \( \epsilon_{i',0}D^{i'} \) and \( \epsilon_{0,j'}E^{j'} \). A similar proof holds for the other corners. Therefore, if all corners are zero, then \( \|y_1(D, E)\|^2 = 0 \) but \( \|y_2(D, E)\|^2 \geq 8 \).

Part 2. If \( m = 1 \) and \( n \geq 1 \), then \( \epsilon_{0,0} \) and \( \epsilon_{0,n-1} \) have to be non-zero from the definition of the closed error events (4.3). The expression for \( y_1(D, E), (4.4) \), is still valid for this case, where \( \epsilon_{m-1,0} = \epsilon_{0,0} \) and \( \epsilon_{m-1,n-1} = \epsilon_{0,n-1} \). Therefore \( \|y_1(D, E)\|^2 = 4 \). The proof of the case \( m \geq 1 \) and \( n = 1 \) is similar to this proof.

**Proof of Proposition 4.1.** Lemma 4.1 states that the distance of a closed error event is at least 2. This lower bound can be attained when all the corners of the error event are nonzero, and all other terms in \( y_2(D, E) \) and \( y_3(D, E) \) are zero. Assuming that \( \epsilon_{0,0} = +, \)
the condition $y_2(D, E) = 0$ implies that all edges of the error event have to be in alternating form. The other corner coefficients of the error event are not free and are determined as stated in the proposition. The condition $y_3(D, E) = 0$ implies that all internal coefficients have to be in alternating form. For example, if $m, n \geq 2$ and $\epsilon_{0,0} = +$, then we obtain $\epsilon_{1,0} = \epsilon_{0,1} = -$ and $\epsilon_{1,1} = +$ by the conditions $y_2(D, E) = 0$ and $y_3(D, E) = 0$. All other entries can be uniquely determined in this way. This concludes the proof the proposition. \qed

**Example 4.1.** Flipping one bit in the input array changes the four adjacent entries at the channel output by 1. Therefore the smallest minimum distance error event is $[+]$ (or its negative $[-]$). Some other distance-2 error events are

$$[+-], [+ -], [+ +], [- -]$$

and their negatives.

The number of input arrays supporting each distance-2 error event is just 2. Hence, the larger the error event area, the less probable it is to encounter that error event. More precisely, the probability of having a minimum distance error event of size $m \times n$ is $2^{-mn+1}$.

### 4.3.2 Open Error Events

Proposition 4.1 implies that all error events with $d < 2$ are open error events. This class of open error events is particularly important for an ML detector using a finite window size which is smaller than the decoding page size. In this case, the distance between some ambiguous arrays can be less than 2, which makes the detector performance poor. Here we give some examples of open error events with squared distance 0, 1, 2 and 3.

**Example 4.2.** The channel output error array corresponding to an open error event is given by

$$y(D, E) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (\epsilon_{i,j} + \epsilon_{i-1,j} + \epsilon_{i,j-1} + \epsilon_{i-1,j-1}) D^i E^j.$$ 

In order to obtain a distance-0 error event, each term in this sum has to be zero. Therefore the error coefficients for each term have to be one of the following combinations

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} + & 0 \\ 0 & - \end{bmatrix}, \begin{bmatrix} + & 0 \\ - & 0 \end{bmatrix}, \begin{bmatrix} + & + \\ - & - \end{bmatrix}, \begin{bmatrix} + & - \\ - & + \end{bmatrix}$$
and their rotations and reflections.

Some distance-0 error events can be obtained by tiling the plane with the same pattern vertically and horizontally. Examples of such patterns include

\[
\begin{bmatrix}
+ & 0 \\
0 & -
\end{bmatrix}, \begin{bmatrix}
+ & - \\
- & +
\end{bmatrix}, \begin{bmatrix}
+ & + & - & + \\
- & - & + & -
\end{bmatrix}.
\]

(4.5)

One of the supporting arrays of the left-most error event in (4.5) can be obtained by tiling the plane with the following pattern

\[
\begin{bmatrix}
+1 & +1 \\
+1 & -1
\end{bmatrix}.
\]

Example 4.3. A distance-1 open error event can be obtained by tiling one quadrant of the plane with the following pattern

\[
\begin{bmatrix}
+ & - \\
- & +
\end{bmatrix}.
\]

(4.6)

Similarly, an error event with squared distance 2 can be obtained by tiling the region \([a, \infty) \times [b_1, b_2]\) with the same pattern where \(a, b_1, b_2 \in \mathbb{Z}\), \(b_2 > b_1\) and \(b_2 - b_1\) is even. Note that tiling a bounded rectangle with this pattern gives a minimum distance closed error event. Also note that tiling a half plane with this pattern gives a distance-0 open error event.

Example 4.4. An error event with squared distance 3 is given by the following form:

\[
\epsilon = \begin{bmatrix}
+ & - & + & - & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
+ & 0 \\
- & 0 & \epsilon_1 \\
\vdots & \vdots & \vdots
\end{bmatrix}
\]

where \(\epsilon_1\) is a distance-1 error event. This event is the union of two open events: \(\epsilon_1\) and the event along two boundaries (the leftmost column and topmost row). The latter has squared distance 2.
4.3.3 Error Events with Squared Distance 6

We now describe two classes of error events with squared distance 6. The first class is obtained by combining two distance-2 error events $\epsilon_1$ and $\epsilon_2$ in the following way

$$
\begin{bmatrix}
\epsilon_1 & 0 \\
0 & \epsilon_2
\end{bmatrix}
$$

(4.7)

where the bottom right corner of $\epsilon_1$ is the negative of the top left corner of $\epsilon_2$. The second error event class has the following form

$$
\begin{bmatrix}
\epsilon_1 & \epsilon_2 \\
0 & \epsilon_3
\end{bmatrix}
$$

(4.8)

where $\epsilon_1$, $\epsilon_2$, and $\epsilon_3$ are distance-2 error events such that there are no adjacent $+'$s and no adjacent $'-$'s at the error event boundaries, horizontally and vertically.

The following patterns are the smallest error events of the first and the second classes, respectively:

$$
\begin{bmatrix}
+ & 0 \\
0 & -
\end{bmatrix},
\begin{bmatrix}
+ & - \\
0 & +
\end{bmatrix}.
$$

4.3.4 Effect of Precoding

The precoded 2-D ISI channel model is shown in Fig. 4.2. Let $u \in \{0, 1\}^*$ be an unconstrained user array at the input to the precoder. A precoder complementing the effect of the 2-D PR1 channel is given by

$$a_{i,j} = u_{i,j} \oplus a_{i-1,j} \oplus a_{i,j-1} \oplus a_{i-1,j-1}$$

where $\oplus$ is the modulo-2 sum operation. The channel input array $x \in \{-1, +1\}^*$ is obtained from $a$ using the binary-to-bipolar conversion $x_{i,j} = 2a_{i,j} - 1$. The channel output $y$ is obtained by convolving $x$ with $h$. Adding AWGN $n$ with zero mean and variance $\sigma^2$ to $y$ gives the noisy channel output $r$. We now discuss two detection scenarios for the precoded 2-D PR1 channel.
Threshold Detection

The user array $u$ is directly related to the channel output array as follows

$$
\frac{y_{i,j}}{2} = \frac{1}{2}(x_{i,j} + x_{i-1,j} + x_{i,j-1} + x_{i-1,j-1})
= a_{i,j} + a_{i-1,j} + a_{i,j-1} + a_{i-1,j-1} - 2 \\
= u_{i,j} \oplus a_{i-1,j} \oplus a_{i,j-1} \oplus a_{i-1,j-1} + a_{i,j-1} + a_{i-1,j-1} - 2
$$

We can treat the result of modulo-2 summing operations as a real number when it is added to the rest of the expression. Therefore, this expression can be simplified as

$$
\frac{y_{i,j}}{2} \equiv u_{i,j} \pmod{2}.
$$

(4.9)

Threshold detection provides the estimate of the channel output array $y'$ using the received array $r$. The estimate of the user data array can be obtained by using (4.9) as follows

$$
\hat{u}_{i,j} = \begin{cases} 
0, & y'_{i,j} = -4, 0, 4 \\
1, & y'_{i,j} = -2, 2.
\end{cases}
$$

If the threshold detector makes an error, i.e., $y' \neq y$, then it is directly reflected in $u'$.

The IMS Detection

The IMS algorithm provides soft information for the estimated channel input array, $x'$. After hard decisions are made for $x'$, the estimated user array, $\hat{u}'$, is given by

$$
\hat{u}'_{i,j} = a'_{i,j} \oplus a'_{i-1,j} \oplus a'_{i,j-1} \oplus a'_{i-1,j-1}
$$

(4.10)

where $a'_{p,q} = (1 + x'_{p,q})/2$.

Error Events for the Precoded Channel

For a precoded system, a user error event $\epsilon_u$ is defined as the difference between a given and estimated user array; i.e., $\epsilon_u = u - \hat{u}'$. In fact, $\epsilon_u$ and $\epsilon = (x - x')/2$ are related to each other according to the following relationship.
Proposition 4.2. $\epsilon_u = \text{sgn}(\epsilon * h)[\epsilon * h \pmod{2}]$, where the multiplication between $\text{sgn}(\epsilon * h)$ and $[\epsilon * h \pmod{2}]$ is element-wise.

Proof. The estimated user array given in (4.10) can be expressed as follows
$$u' = \frac{1 + x'}{2} * h \pmod{2}.$$ The same formula applies to the user array at the input to the precoder
$$u = \frac{1 + x}{2} * h \pmod{2}.$$ Therefore the user error event is given by
$$\epsilon_u = \frac{1 + x}{2} * h \pmod{2} - \frac{1 + x'}{2} * h \pmod{2} = \text{sgn}(\epsilon_x * h)[\epsilon_x * h \pmod{2}].$$

A $k \times l$ distance-2 error event $\epsilon$ in the channel input arrays corresponds to the following $(m + 1) \times (n + 1)$ error event in the user data arrays
$$\epsilon_u = \begin{bmatrix} \epsilon_{0,0} & 0 & \cdots & \epsilon_{0,n-1} \\ 0 & 0 & & \\ \vdots & \vdots & & \\ \epsilon_{m-1,0} & \cdots & \epsilon_{k-1,n-1} \end{bmatrix}.$$ The Hamming weight of $\epsilon_u$ always satisfies $w(\epsilon_u) = 4$. This is a result of the one-to-one relationship between $y$ and $u$ given in (4.9). For error events with squared distance 6, the user data error events $\epsilon_u$ have constant Hamming weight 6 and non-zero entries are located at the corners of the error events.

Example 4.5. The following are two examples of user data error events with squared distance 4 and 6, respectively.
$$\epsilon = \begin{bmatrix} + & - \\ - & + \end{bmatrix} \leftrightarrow \epsilon_u = \begin{bmatrix} + & 0 & - \\ 0 & 0 & 0 \\ - & 0 & + \end{bmatrix},$$
$$\epsilon = \begin{bmatrix} + & 0 \\ 0 & - \end{bmatrix} \leftrightarrow \epsilon_u = \begin{bmatrix} + & + & 0 \\ + & 0 & - \\ 0 & - & - \end{bmatrix}.$$
4.4 Extension to Other 2-D Channels

In the previous section, the distance properties of the 2-D PR1 channel are investigated by using the spectral representations of the signals. This method can be extended to prove several distance properties of some classes of 2-D ISI channels.

Proposition 4.3. A general 2-D channel with impulse response of size $2 \times 2$ is given by

$$h = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

for arbitrary values of $\alpha$, $\beta$, $\gamma$, and $\delta$. This channel achieves the matched-filter bound; i.e., the minimum closed event distance of this channel is given by

$$d_{\text{<}} = \sqrt{\alpha^2 + \beta^2 + \gamma^2 + \delta^2}.$$

Similar to the proof of Proposition 4.1, the proof of this proposition is based on an expansion of (4.2) using the definition of closed error events (4.3). Let $y(D, E)$ be the channel output corresponding to a closed error array $\epsilon(D, E)$. For $m, n \geq 2$, $y(D, E)$ can be expanded as

$$y(D, E) = (\alpha + \beta D + \gamma E + \delta DE) \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \epsilon_{i,j} D^i E^j \triangleq y_1(D, E) + y_2(D, E) + y_3(D, E)$$

where $y_1(D, E)$ contains the terms with a single error coefficient, which correspond to the corners of the error event

$$y_1(D, E) = \alpha \epsilon_{0,0} + \beta \epsilon_{k-1,0} D^m + \gamma \epsilon_{0,l-1} E^n + \delta \epsilon_{k-1,l-1} D^m E^n.$$

Each of the terms on the second expression, $y_2(D, E)$, has two error coefficients corresponding to the edges of the error event

$$y_2(D, E) = \sum_{i=1}^{m-1} (\alpha \epsilon_{i,0} + \beta \epsilon_{i-1,0}) D^i + \sum_{j=1}^{n-1} (\alpha \epsilon_{0,j} + \gamma \epsilon_{0,j-1}) E^j$$

$$+ \sum_{i=1}^{m-1} (\gamma \epsilon_{i,n-1} + \delta \epsilon_{i-1,n-1}) D^i E^n + \sum_{j=1}^{n-1} (\beta \epsilon_{m-1,j} + \delta \epsilon_{m-1,j-1}) D^m E^j.$$

Each of the terms in the third expression, $y_3(D, E)$, has four error coefficients corresponding to the middle of the error event

$$y_3(D, E) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} (\alpha \epsilon_{i,j} + \beta \epsilon_{i-1,j} + \gamma \epsilon_{i,j-1} + \delta \epsilon_{i-1,j-1}) D^i E^j.$$
The following lemma provides lower bounds for the squared distances of \( y_1(D, E) \), \( y_2(D, E) \).

**Lemma 4.2.** Depending on the size of the error event, the contributions to the squared distance from \( y_1(D, E) \) and \( y_2(D, E) \) are bounded from below as follows:

1. If \( m, n \geq 2 \),
   \[
   \|y_1(D, E) + y_2(D, E)\|^2 \geq 2(\alpha^2 + \beta^2 + \gamma^2 + \delta^2) + (p_0,*) - 1 \alpha,\gamma + (p_*,0 - 1) \beta,\gamma + (p_{m-1,*} - 1) \gamma,\delta + (p_*,n-1) \gamma,\delta - \alpha^2|\epsilon_{0,0}| - \beta^2|\epsilon_{m-1,0}| - \gamma^2|\epsilon_{0,n-1}| - \delta^2|\epsilon_{m-1,n-1}|
   \]
   where \( p_{f,g} \) is the length of a run of non-zero terms along the edge \( \epsilon_{f,g} \), and \( T_{q,r} \) is the minimum contribution to the squared distance from two non-zero adjacent entries on an edge:
   \[
   T_{q,r} = \min_{\epsilon_{i,j}, \epsilon_{i',j'} \neq 0} \left( q\epsilon_{i,j} + r\epsilon_{i',j'} \right)^2.
   \]
   where \( q, r \in \{\alpha, \beta, \gamma, \delta\} \); and \( \epsilon_{i,j} \) and \( \epsilon_{i',j'} \) are two adjacent entries on an edge.

2. If \( m = 1, n \geq 2 \),
   \[
   \|y(D, E)\|^2 \geq \alpha^2 + \beta^2 + \gamma^2 + \delta^2 + T_{\alpha,\gamma} + T_{\beta,\delta}.
   \]
   A similar result follows for the \( m \geq 2, n = 1 \) case.

3. If \( m = n = 1 \),
   \[
   \|y(D, E)\|^2 = \alpha^2 + \beta^2 + \gamma^2 + \delta^2.
   \]

**Proof.** Part 1. The proof has a symmetric characteristic that makes it sufficient to bound the contribution to the squared distance from one edge. Without loss of generality, we consider the terms related to the first row of the error event
   \[
   y'(D, E) = \alpha \epsilon_{0,0} + \sum_{j=1}^{n-1} (\alpha \epsilon_{0,j} + \gamma \epsilon_{0,j-1}) E^j + \gamma \epsilon_{0,l-1} E^n.
   \]

Let \( r = \{j_1, \ldots, j_2\} \) be a run of consecutive indices from the set \( \{0, \ldots, n-1\} \) such that \( \epsilon_{0,j_1-1} = \epsilon_{0,j_2+1} = 0 \) and \( \epsilon_{0,j} \neq 0 \) for \( j \in r \). From the definition of closed error events (4.3), there exists at least one such \( r \) for each edge. Rewriting \( y'(D, E) \) gives
   \[
   y'(D, E) = \alpha \epsilon_{0,j_1} E^{j_1} + \sum_{j=j_1+1}^{j_2} (\alpha \epsilon_{0,j} + \gamma \epsilon_{0,j-1}) E^j + \gamma \epsilon_{0,j_2+1} E^{j_2+1}.
   \]
The first and last terms contribute \( \alpha^2 + \gamma^2 \) to \( \|y'(D, E)\|^2 \). Each of the middle terms can be bounded by the least value of \((\alpha\epsilon_{0,j} + \gamma\epsilon_{0,j-1})\), which is

\[
T_{\alpha,\gamma} = \min_{\epsilon_{0,j}, \epsilon_{0,j-1} \neq 0} (\alpha\epsilon_{0,j} + \gamma\epsilon_{0,j-1})^2.
\]

Therefore, \( \|y'(D, E)\|^2 \geq \alpha^2 + \gamma^2 + (p_r - 1)T_{\alpha,\gamma} \) where \( p_r = j_2 - j_1 + 1 \).

The contributions to the distance from the other edges can be computed similarly. Since the terms at the corners are computed twice, their double contributions should be subtracted. For example, if \( \epsilon_{0,0} \neq 0 \), it contributes \( \alpha^2 \) along the two edges of the error event, \( \epsilon_{0,*} \) and \( \epsilon_{*,0} \). These contributions can be subtracted as stated in the lemma. Summing all terms concludes the proof of part 1.

Part 2. For \( m = 1 \) and \( n \geq 2 \), the channel output for the error event \( \epsilon(D, E) \) is given by

\[
y(D, E) = \alpha\epsilon_{0,0} + \sum_{j=1}^{n-1} (\alpha\epsilon_{0,j} + \gamma\epsilon_{0,j-1})E_j + \gamma\epsilon_{0,l-1}E^n
+ \beta\epsilon_{0,0}D + \sum_{j=1}^{n-1} (\beta\epsilon_{0,j} + \delta\epsilon_{0,j-1})DE_j + \delta\epsilon_{0,l-1}DE^n.
\]

Since \( \epsilon_{0,0}, \epsilon_{0,n-1} \neq 0 \), the lower bound on \( \|y(D, E)\|^2 \) is exactly as stated in the lemma. The case of \( m \geq 2, n = 1 \) is similar.

Part 3. If \( m = n = 1 \), then \( \|y(D, E)\|^2 = \alpha^2 + \beta^2 + \gamma^2 + \delta^2 \), which is the matched filter bound of the channel. \( \square \)

Proof of Proposition 4.3. This proof can be easily performed by using Lemma 4.2. By the definition of \( T_{q,r} \), \( T_{q,r} \geq 0 \) for all \( q, r \in \{\alpha, \beta, \gamma, \delta\} \). Even if all corners of the error event are non-zero, the squared distance is bounded from below by

\[
\|y_1(D, E) + y_2(D, E)\|^2 \geq \alpha^2 + \beta^2 + \gamma^2 + \delta^2.
\]

This lower bound is the same as the matched filter bound of the channel. Therefore, the minimum closed event distance of the channel is as stated in the proposition. \( \square \)

Remark 4.1. (a) The lower bound in Proposition 4.3 can be achieved for arbitrary \( m \) and \( n \) if \( T_{q,r} = 0 \) for all possible combinations of \( q \) and \( r \), the corners of the error events are not zero, and \( y_3(D, E) = 0 \). This is exactly the case for the 2-D PR1 channel since it is possible to achieve this by choosing alternating error coefficients + and −.
(b) For a general 2-D channel with impulse response, \( h = \{h_{i,j}\}_{i=0,j=0}^{u-1,v-1} \), the minimum closed event distance can be bounded from below by

\[
d_{<>} \geq \sqrt{h_{0,0}^2 + h_{u-1,0}^2 + h_{0,v-1}^2 + h_{u-1,v-1}^2}.
\]

In this case, the matched-filter-bound may not be achieved for some channels, as illustrated in the following example.

**Example 4.6.** Let \( h \) be a 2-D ISI channel with impulse response

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & a & 1 \\
1 & 1 & 1
\end{bmatrix}.
\]

If \( \epsilon(D, E) = 1 - D \), then the channel output becomes

\[
y = \begin{bmatrix}
1 & 1 & 1 \\
0 & a - 1 & 0 \\
0 & 1 - a & 0 \\
-1 & -1 & -1
\end{bmatrix}
\]

which has squared distance \( \|y\|^2 = 6 + 2(a - 1)^2 \). The range of values for \( a \) that lead to the channels which do not achieve the MFB is determined by

\[
\|h\|^2 - \|y\|^2 = -a(a - 4) > 0
\]

This inequality holds for \( 0 < a < 4 \). For instance, if \( a = 1 \), then \( \|y\|^2 = 6 \), but \( \|h\|^2 = 9 \).

### 4.5 The Probability of Error

If the entries of input arrays \( x \) (or \( u \) for the precoded case) are equally probable and independent, the probability of an error event under ML detection can be bounded from above by the union bound

\[
P(\text{event error}) \leq \sum_d K_d Q\left(\frac{d}{2\sigma}\right)
\]

where \( K_d \) is the average multiplicity of error events of distance \( d \), given by

\[
K_d = \sum_{\epsilon, d(\epsilon) = d} \left(\frac{1}{2}\right)^{w(\epsilon)},
\]
where \(w(\epsilon)\) is the Hamming weight of \(\epsilon\). Similarly, the bit error probability can be bounded from above as

\[
P(\text{bit error}) \leq \sum_d N_d Q\left(\frac{d}{2\sigma}\right)
\]

where \(N_d\) is the average multiplicity of bit errors resulting from error events of distance \(d\), given by

\[
N_d = \sum_{\epsilon, d(\epsilon) = d} w(\epsilon) \left(\frac{1}{2}\right)^{w(\epsilon)}.
\]

For a precoded system, \(N_d\) becomes

\[
N_d = \sum_{\epsilon, d(\epsilon) = d} w(\epsilon_u) \left(\frac{1}{2}\right)^{w(\epsilon)}.
\]

The error event multiplicity generating function is defined as

\[
ge_e(z) \triangleq \sum_{d \in D^*} K_d z^{d^2}.
\]

Likewise, the bit error multiplicity generating function is defined as

\[
ge_b(z) \triangleq \sum_{d \in D^*} N_d z^{d^2}
\]

where \(D^*\) is the set of all distances.

**Example 4.7.** For the 2-D PR1 channel without precoding, \(K_2\) and \(N_2\) can be computed as follows

\[
K_2 = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \left(\frac{1}{2}\right)^{mn} = \sum_{m=1}^{\infty} \frac{1}{2^m - 1} \approx 1.606 \quad (4.11)
\]

\[
N_2 = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} mn \left(\frac{1}{2}\right)^{mn} \approx 4.052 \ldots \quad (4.12)
\]

The exact expressions for \(K_2\) and \(N_2\) cannot be expressed in a closed form. The ratio test indicates that both sums converge. The former is called the Erdös-Borwein constant \([33]\) and an approximate value is shown in (4.11). The approximate value of the latter, shown in (4.12), was obtained numerically by summing over terms with \(m, n \leq 100\).

For the 2-D PR1 channel with precoding, \(K_2\) is the same as (4.11) and \(N_2\) can be computed by using (4.11)

\[
N_2 = 4K_2 \approx 6.4264.
\]

In this case, the coefficient \(N_2\) is larger for the precoded system. Therefore, precoding increases the bit-error multiplicity of the minimum distance error events.
4.6 A Bounded Depth-Search Algorithm

Error state diagrams for 1-D channels cannot be directly generalized to 2-D channels since there are no convenient graph-based descriptions of such channels. However, when the size of the error event is fixed in either of the dimensions, error state diagrams can be described as 1-D systems using a higher order alphabet. In this section, we propose a bounded depth-search algorithm using error-state diagrams for determining closed error events. To reduce the complexity of the algorithm, the following conditions are imposed on error events: (1) the edges of a error event contain at least one non-zero element and (2) error events are required to be connected, as defined below.

**Definition 4.1.** An error event $\epsilon$ is *connected* if the error event cannot be divided into two separate components $\epsilon_1$ and $\epsilon_2$ such that $d^2(\epsilon) = d^2(\epsilon_1) + d^2(\epsilon_2)$. The error events which are not connected are called *disconnected*.

In order to find a condition for an error event being connected, we define the notion of connected entries in the error event.

**Definition 4.2.** Let $h = \{h_{i,j}\}_{i=0,j=0}^{u-1,v-1}$ be the impulse response of a 2-D channel. Let $h'$ be a $u \times v$ indicator matrix whose entries are defined as

$$h'_{i,j} = \begin{cases} 1, & \text{if } h_{i,j} \neq 0 \\ 0, & \text{otherwise}. \end{cases}$$

For an error event $\epsilon$ of size $m \times n$, two non-zero entries $(i_1, j_1)$ and $(i_2, j_2)$ are said to be *connected* to each other if the following condition is true:

$$\|A \ast h'\|^2 < 2\|h'\|^2$$

where $A$ is an $m \times n$ matrix such that $A_{i_1,j_1} = 1$, $A_{i_2,j_2} = -1$ and other entries of $A$ are zero. If $\|A \ast h'\|^2 = 2\|h'\|^2$, then $(i_1, j_1)$ and $(i_2, j_2)$ are *disconnected*.

**Remark 4.2.** The entries $(i_1, j_1)$ and $(i_2, j_2)$ cannot be connected if $|i_2 - i_1| + 1 > u$ or $|j_2 - j_1| + 1 > v$.

**Definition 4.3.** A set of non-zero entries $\epsilon'$ in an error event $\epsilon$ is called a *connected component* of $\epsilon$ if every two non-zero entries of $\epsilon'$ are connected via other non-zero entries in $\epsilon'$.

Definitions 4.1, 4.2, and 4.3 directly imply the following result.
**Proposition 4.4.** An error event is connected if and only if it has one connected component.

**Example 4.8.** The following error events are two examples of connected and disconnected error events for the 2-D PR1 channel, respectively:

\[
\epsilon_1 = \begin{bmatrix} + & - & + \\ 0 & 0 & 0 \\ + & - & + \end{bmatrix}, \quad \epsilon_2 = \begin{bmatrix} + & 0 & + \\ 0 & - & 0 \\ + & 0 & + \end{bmatrix}.
\]

Error-state diagrams for 2-D channels can be constructed by considering error events as a 1-D sequence of symbols from a higher order alphabet. A state \(\sigma\) is a sequence of \(u - 1\) symbols, \((e_1, \ldots, e_{u-1})\), from the alphabet \(\Sigma\), which is the set of all row vectors of length \(n\) with entries \(\{0, +, -\}\). Note that the memory of the equivalent 1-D channel is \(u - 1\). Each state can be represented as a \((u - 1) \times n\) matrix whose row vectors are \((e_1, \ldots, e_{u-1})\). Therefore there are \(3^{(u-1)n}\) states in the error-state diagram. An edge \(e\) has the initial state \(\sigma(e) = (e_1, \ldots, e_{u-1})\) and the terminal state \(\tau(e) = (e_2, \ldots, e_u)\). A closed error event of size \(m \times n\) corresponds to the path \(e_0, \ldots, e_{m+u-1}\) in the error-state diagram that starts and ends at the all-zero state \(\sigma = (z, \ldots, z)\) without intermediate visit to that state, where \(z\) is the all-zero row vector of length \(n\). The algorithm searches for the connected error events of size \(m \times n\) whose distances are not larger than a specified limit \(d_{\text{max}}\).

Let \(\rho^{(l-1)}\) be a path of length \(l < m + u - 1\) ending with the state \(\sigma\). If a row vector \(e \in \Sigma\) is appended to this path, the algorithm checks the following conditions on the new path \(\rho^{(l)} = (\rho^{(l-1)}, e)\):

- The distance associated with the new path satisfies \(d^2(\rho^{(l)}) \leq d_{\text{max}}^2\).
- When \(l = m\), the error event associated with \(\rho^{(l)}\) contains at least one non-zero entry along its edges.
- \(\rho^{(l)}\) gives a connected error event.

If any of these checks fails, then the path \(\rho^{(l)}\) is called invalid and the algorithm will not extend the path \(\rho^{(l)}\). If all of these checks satisfy, then the path \(\rho^{(l)}\) is called a valid path.

Let \(C^{(l)}\) and \(V^{(l)}\) be the sets of candidate and valid paths at level \(l\), respectively. All candidate paths at level \(l\) are obtained by extending the valid paths at level \(l - 1\). The
algorithm performs three updates and checks on $C^{(l)}$ to obtain $V^{(l)}$ as discussed later in this section. The algorithm proceeds as follows:

- For level $l = 0$, the only candidate and valid path is the all-zero vector from $\Sigma$, i.e.,
  
  \[ C^{(0)} = V^{(0)} = \{ z \}. \]

- For level $l = 1$, the all-zero path in $V^{(0)}$ is extended by a non-zero row vector from $\Sigma$, i.e.,
  
  \[ C^{(1)} = \{ ze | e \in \Sigma \setminus \{ z \} \}. \]

  This guarantees that the first row of the error event contains at least one non-zero entry.

- For levels $1 < l < m$, $C^{(l)}$ is obtained by extending the valid paths from the previous level $V^{(l-1)}$. In this step, the appended row vector can be any vector from $\Sigma$ including $z$:
  
  \[ C^{(l)} = \{ \rho^{(l-1)} e | \rho^{(l-1)} \in V^{(l-1)} \text{ and } e \in \Sigma \}. \]

- For level $l = m$, the valid paths of the previous level are extended by non-zero row vectors from $\Sigma$, i.e.,
  
  \[ C^{(m)} = \{ \rho^{(m-1)} e | \rho^{(m-1)} \in V^{(m-1)} \text{ and } e \in \Sigma \setminus \{ z \} \}. \]

  This guarantees that the last row of the error event contains at least one non-zero entry.

- For levels $m < l < m + u$, since the error event has $m$ non-zero rows, the valid paths are extended by $z$, i.e.,
  
  \[ C^{(l)} = \{ \rho^{(l-1)} z | \rho^{(l-1)} \in V^{(l-1)} \}. \]

Let $\rho^{(l-1)}$ be a valid path of length $l - 1 < m + u - 1$ ending with the state $\sigma$. If a row vector $e \in \Sigma$ is appended to this path, the algorithm checks the following three conditions on the new path $\rho^{(l)} = (\rho^{(l-1)}, e)$.

4.6.1 Updating and Checking Squared Distance

Let $d(\rho^{(l)})$ and $d(\rho^{(l-1)})$ be the distances of paths $\rho^{(l-1)}$ and $\rho^{(l)}$, respectively. We assume that all entries outside the spans of the error events are simply zero. We can
compute the channel output by considering the augmented state \((\sigma(e), e)\), which is padded with zeros:

\[
\sigma' = \begin{pmatrix}
0^{(u-1)\times(v-1)} & \sigma(e) & 0^{(u-1), (v-1)} \\
0^{1\times(v-1)} & e & 0^{1, (v-1)}
\end{pmatrix}
\]

where \(0^{m\times n}\) be the all-zero matrix of size \(m \times n\). The channel output corresponding to the channel input \(e\) is given by

\[
\mu_l = \sum_{p=0}^{n-1} (h \ast \sigma')_p = \sum_{r=0}^{n-1}\sum_{i=0}^{u-1}\sum_{j=0}^{v-1} h_{i,j} \sigma'_{u-1-i,p+v-1-j}.
\]

Therefore the distance of the path \(\rho^{(l)}\) is updated as

\[
d^2(\rho^{(l)}) = d^2(\rho^{(l-1)}) + \mu_l^2.
\]

If \(d(\rho^{(l)}) > d_{max}\), then the distance check on \(\rho^{(l)}\) fails.

### 4.6.2 Updating and Checking Edge Indicator Functions

Let \(I_1(\rho^{(l)})\) be the left edge indicator function for the path \(\rho^{(l)}\) indicating whether the left edge of the error event contains a non-zero entry. The left edge indicator function can be updated as follows

\[
I_1(\rho^{(l)}) = \begin{cases} 
1, & \text{if } I_1(\rho^{(l-1)}) = 1 \text{ or } e_0 \neq 0 \\
0, & \text{otherwise}
\end{cases}
\]

where \(e_0\) is the first entry of the vector \(e\). The right edge indicator variable \(I_2(\rho^{(l)})\) can be defined and updated similarly. For a path \(\rho^{(m)}\) at level \(l = m\), if \(I_1(\rho^{(m)}) = 0\) or \(I_2(\rho^{(m)}) = 0\), then the check on \(\rho^{(m)}\) fails.

### 4.6.3 Updating and Checking Connection Map

A connected component of an error event is the set of connected entries in an error event. Connected components of an error events can be distinctly numbered with positive integers. We define the group number of a non-zero entry in a path as the connected component number to which that entry belongs. A connection map for a path \(\rho^{(l)}\) is an \((l+1, n)\) array whose entries are the group numbers of the non-zero entries. Zero entries in \(\rho^{(l)}\) are assumed to have group number zero. The maximum group number of \(\rho^{(l)}\) is denoted by \(\varphi(\rho^{(l)})\). Let \(M(\rho^{(l-1)})\) and \(M(\rho^{(l)})\) be the connection maps for paths \(\rho^{(l-1)}\) and
\( \rho^{(l)} \), respectively. For the overlapping rows of \( \rho^{(l-1)} \) and \( \rho^{(l)} \), the connection groups are the same, i.e, \( M(\rho^{(l)})_{i,s} = M(\rho^{(l-1)})_{i,s} \) for \( 0 \leq i < l \). For each non-zero entry \( j \) in the last row of \( M(\rho^{(l)}) \), we need to determine which groups are connected to the entry \( [M(\rho^{(l)})]_{i,j} \).

Let \( P_i \) be the set of entries in \( M(\rho^{(l)}) \) connected to the \((l,j)\)th entry and \( G_j \) be the set of group numbers of the entries in \( P_j \). For the \((l,j)\)th entry, the connection map \( M(\rho^{(l)}) \) is updated as follows:

- If \( G_j = \emptyset \), i.e, the \((l,j)\) entry of the path \( \rho^{(l)} \) is not connected to any connected components, then a new group number is assigned to this entry: \( [M(\rho^{(l)})]_{i,j} = \varphi(\rho^{(l-1)}) + 1 \) and \( \varphi(\rho^{(l)}) = \varphi(\rho^{(l-1)}) + 1 \).

- If \( G_j \neq \emptyset \), then \( [M(\rho^{(l)})]_{i,j} = \max\{G_j\} \) and for all non-zero entries \((a,b)\), set \( [M(\rho^{(l)})]_{a,b} = \max\{G_j\} \). In this way, all connected components whose numbers are in \( G_j \) are merged into the single connected component with number \( \max\{G_j\} \).

There are two different types of checks associated with connection maps.

- No group can terminate without merging with another group. That is, the rows \( [M(\rho^{(l)})]_{i,s}, 0 \leq i \leq l - u + 1 \), can not contain a group number which does not occur in the rows \( [M(\rho^{(l)})]_{i,s}, l - u + 1 < i < l \). This check is applied for the paths at level \( l \leq m \).

- All groups have to merge into a single group at level \( l = m \); otherwise the error event will be disconnected. That is, all non-zero entries of \( M(\rho^{(m)}) \) should be \( \varphi(\rho^{(m)}) \).

If any of these checks fails, then the algorithm will not expand the path \( \rho^{(l)} \).

**Example 4.9.** Figure 4.3 shows three paths and their connection maps for the 2-D PR1 channel. Two entries are connected for this channel if they are adjacent to each other horizontally, vertically, or diagonally. The path \( \rho^{(2)} \) is an extension of \( \rho^{(1)} \) and \( \rho^{(3)} \) is an extension of \( \rho^{(2)} \). The path \( \rho^{(1)} \) has two connected components, which are merged into one connected component in path \( \rho^{(2)} \). The path \( \rho^{(3)} \) introduces a new connected component and the old connected component terminates before merging with the new one. Therefore any path, which is an extension of \( \rho^{(3)} \) will give a disconnected error event.

### 4.6.4 Simulation Results

The number of distinct error events with a fixed distance varies from channel to channel. The checks discussed in previous subsections eliminate undesired repetition of error events.
Figure 4.3 An example of connection map.

Figure 4.4 shows the number of error events for the 2-D PR1 channel for different check configurations. Checks I, II, and III refer to the squared distance, edge indicator functions, and the connection map, respectively. Applying all three checks gives a considerably less number of error events for a fixed maximum distance $d_{max}$.

**Example 4.10.** Table 4.2 shows some error events for the 2-D PR1 channel. Table 4.3 shows the number of error events for $d^2 = 4, 6, 8, 10$ and $m, n \leq 6$. For $d^2 = 4$, the results of the search algorithm are consistent with the analytical results above. For $d^2 = 6$, the number of error events of the forms (4.7) and (4.8) are $4(m-1)(n-1)$ and $8(m-1)(n-1)$, respectively. The number of distinct error events that the algorithm produces is exactly $12(m-1)(n-1)$ for $m, n \leq 6$. This suggests that there may be no other error event forms for error events with squared distance 6.

Using the data given in Table 4.3, the error event multiplicity and the bit error multiplicity generating functions can be bounded from below by

$$g_e(z) \geq 1.575z^4 + 7.216z^6 + 45.74z^8 + 251.6z^{10},$$

$$g_b(z) \geq 3.800z^4 + 34.73z^6 + 311.1z^8 + 2049z^{10}.$$ 

Here “$\geq$” signifies that the coefficient of each term on the right is less than the corresponding one on the left. The computed values of $K_2$ and $N_2$ are close to the analytical values.
Figure 4.4 The number of error events of size $m \times n$ for the 2-D PR1 channel where $d^2 \leq 8$.

**Example 4.11.** Consider the channel with impulse response

$$h = \begin{bmatrix} 0 & .25 & 0 \\ .25 & 1 & .25 \\ 0 & .25 & 0 \end{bmatrix}.$$  

The eye diagram of this channel is closed but it can be opened by eliminating the following patterns from the channel input arrays

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}.$$  

All error events with squared distance $d^2 \leq 2.25$ are shown in Table 4.4. Note that reflections, rotations and negatives of these error events are also error events with the same distance. According to these results, the minimum closed event distance of this channel is 1.25, and only achieved by the single error [+]. Therefore, this channel may achieve the MFB. In Section 4.7, we will prove that this is indeed the case.

The error event and bit error multiplicity generating functions can be bounded from
Table 4.2 Some error events for the 2-D PR1 channel.

<table>
<thead>
<tr>
<th>$d^2(\epsilon)$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$[+]$, $[+ -]$, $[- +]$, $[+ -]$, $[- +]$, $[- +]$, $[+ -]$</td>
</tr>
<tr>
<td>6</td>
<td>$[+ 0]$, $[+ -]$, $[+ 0]$, $[0 0 +]$, $[+ -]$, $[+ -]$, $[0 0 +]$</td>
</tr>
<tr>
<td>8</td>
<td>$[+ - +]$, $[+ 0 +]$, $[0 0 +]$, $[+ - +]$, $[+ - +]$, $[0 0 +]$</td>
</tr>
<tr>
<td>10</td>
<td>$[+ 0]$, $[+ +]$, $[0 -]$, $[+ - +]$, $[+ +]$</td>
</tr>
</tbody>
</table>

Table 4.3 The number of error events for the 2-D PR1 channel.

<table>
<thead>
<tr>
<th>$d^2 = 4$</th>
<th>$d^2 = 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m \backslash n$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d^2 = 8$</th>
<th>$d^2 = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m \backslash n$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

below as

$$g_e(z) \geq 0.5z^{1.25} + 0.562z^{1.5} + 0.031z^{1.75} + 0.25z^{1.875} + 0.51z^2 + 0.25z^{2.125} + 0.643z^{2.25} + 1.125z^{2.375} + 1.528z^{2.5},$$

$$g_b(z) \geq 0.5z^{1.25} + 1.25z^{1.5} + 0.187z^{1.75} + 0.75z^{1.875} + 1.58z^2 + 1.25z^{2.125} + 1.65z^{2.25} + 3.875z^{2.375} + 5.612z^{2.5}.$$ 

Unlike the 2-D PR1 channel, the maximum size of an error event with a fixed distance seems to be bounded for this channel.
Table 4.4 Error events with squared distance $d^2 \leq 2.25$ for the channel given in Example 4.11.

<table>
<thead>
<tr>
<th>$d^2(\epsilon)$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25</td>
<td>$[+]$</td>
</tr>
<tr>
<td>1.5</td>
<td>$[+ - +]$, $[- + +]$</td>
</tr>
<tr>
<td>1.75</td>
<td>$[+ - +]$</td>
</tr>
<tr>
<td>1.875</td>
<td>$[+ - +]$</td>
</tr>
<tr>
<td>2</td>
<td>$[+ - + +]$, $[- + + -]$, $[+ - + +]$</td>
</tr>
<tr>
<td>2.125</td>
<td>$[+ - +]$</td>
</tr>
<tr>
<td>2.25</td>
<td>$[+ - + +]$, $[0 0 -]$, $[+ - + +]$</td>
</tr>
</tbody>
</table>

Example 4.12. Example 4.6 shows that the channel with impulse response

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

does not achieve the MFB. Table 4.5 shows some error events for this channel. These results suggest that the minimum squared distance of this channel is 4. Error events of sizes $1 \times 1$, $1 \times 2$ and $2 \times 1$ do not achieve the minimum distance. However, there is an infinite family of error events that achieve the minimum distance, consisting of error events of the form

$$e_{0^{2 \times 1}} e \cdots$$
$$0^{1 \times 2} 0 0^{1 \times 2}$$
$$e_{0^{2 \times 1}} e$$
$$\vdots$$

where $e = \begin{bmatrix} + & - \\ - & + \end{bmatrix}$ and $0^{m \times n}$ is a size $m \times n$ matrix of zeros.

All error events with size $m \times n$, $m, n \leq 6$, and $d^2(\epsilon) \leq 11$ are found by the bounded depth-search algorithm. The error event and the bit error multiplicity generating functions
Table 4.5 Some error events with squared distance $d^2 \leq 9$ for the channel given in Example 4.12.

<table>
<thead>
<tr>
<th>$d^2(\epsilon)$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$[+ -]$, ...</td>
</tr>
<tr>
<td>6</td>
<td>$[+ -]$, $[+ - +]$, ...</td>
</tr>
<tr>
<td>8</td>
<td>$[++ 0 +]$, $[+ - 0]$, $[+ - + -]$, ...</td>
</tr>
<tr>
<td>9</td>
<td>$[+]$, $[+ - +]$, $[+ - -]$, ...</td>
</tr>
</tbody>
</table>

can be bounded from below by

$$g_e(z) \geq 0.07z^4 + 0.733z^6 + 1.201z^8 + 1.393z^{10} + 9.246z^{10} + 8.505z^{11},$$

$$g_b(z) \geq 0.313z^4 + 2.193z^6 + 7.63z^8 + 3.539z^9 + 54.56z^{10} + 43.67z^{11}.$$

Similar to the 2-D PR1 channel, the error events with a fixed distance can be very large for this channel.

**Example 4.13.** Consider the channel with asymmetric impulse response

$$h = \begin{bmatrix} 1 & .5 \\ .5 & 0 \end{bmatrix}.$$  

The simulation results suggest that the minimum squared distance of this channel is 1.5 and the minimum distance error event is $[+]$. All error events satisfying $d^2(\epsilon) \leq 3$ are shown in Table 4.6. Note that if $\epsilon$ is an error event for this channel, then $\epsilon^T$ is also an error event. The error event and bit error error multiplicity generating functions can be bounded from below by

$$g_e(z) \geq 0.5z^{1.5} + 0.5z^2 + 0.812z^{2.5} + 1.406z^3 + 2.713z^{3.5} + 6.163z^4 + 12.11z^{4.5} + 24.48z^5,$$

$$g_b(z) \geq 0.5z^{1.5} + z^2 + 2.25z^{2.5} + 5.062z^3 + 11.57z^{3.5} + 28.83z^4 + 66.85z^{4.5} + 154.1z^5.$$
Table 4.6 Some error events for the channel given in Example 4.13.

<table>
<thead>
<tr>
<th>$d^2(\epsilon)$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>[+</td>
</tr>
<tr>
<td>2</td>
<td>+ −</td>
</tr>
<tr>
<td>2.5</td>
<td>[ + 0</td>
</tr>
<tr>
<td></td>
<td>[ + 0 0</td>
</tr>
<tr>
<td></td>
<td>[ + 0 +</td>
</tr>
<tr>
<td></td>
<td>[ + 0 −</td>
</tr>
<tr>
<td></td>
<td>[ + 0 − +</td>
</tr>
<tr>
<td></td>
<td>[ + 0 − + − ]</td>
</tr>
<tr>
<td>3</td>
<td>[ + − + − ]</td>
</tr>
<tr>
<td></td>
<td>[ + − + − ]</td>
</tr>
<tr>
<td></td>
<td>[ + − + − ]</td>
</tr>
<tr>
<td></td>
<td>[ + − + − ]</td>
</tr>
</tbody>
</table>

4.6.5 Reduction of Complexity

A measure for the complexity of the bounded depth-search algorithm can be the number of paths that are updated and checked. Implementing the checks for the edge indicator functions and connection map does not significantly reduce the complexity as shown in Fig. 4.5 for the 2-D PR1 channel. However, the complexity of the algorithm can be reduced by modifying the check corresponding to the squared distance. The minimum squared distance of an error event of size $m \times n$ is defined as the minimum contributions to the squared distance after the error event terminates, i.e.,

$$d_{min}^2 = \min_{\rho(m+u-1) \neq \rho(m), \tau} d^2(\rho(m+u-1)) - d^2(\rho(m))$$

where $\tau$ is the terminal state of the path $\rho(m+u-1)$. The squared distance of an error event of size $m \times n$ is bounded from below by $d_{min}$ for $m \geq u - 1$. The minimum squared termination distance can be found simply by enumerating all possible paths in question. The check related to the squared distance can be modified by either replacing $d_{max}$ with $d_{max}^2 - d_{min}^2$ for the paths at level $l \leq m$ or initializing the distance of the all-zero path at level $l = 0$ as $d_{min}$. In the latter case, $d_{min}$ has to be subtracted from the squared distances of the paths at level $l = m$. As shown in Fig. 4.5, this method reduces the complexity of the algorithm significantly. This method is also observed to be efficient for other channel and $d_{max}$ combinations.

4.7 A Lower Bound on Distances of Error Events

Consider an arbitrary 2-D ISI channel with impulse response $h = \{h_{i,j}\}_{i=0,j=0}^{u-1,v-1}$. In this section, we propose a computational lower bound to the distance of a closed error event
Figure 4.5 The complexity of the algorithm for the 2-D PR1 channel when $d_{\text{max}}^2 = 6$. The check IV refers to the modified squared distance with $d_{\text{min}}^2 = 2$ for all $m$ and $n$.

$\epsilon = \{\epsilon_{i,j}\}_{i=0,j=0}^{m-1,n-1}$ of size $m \times n$. The channel output error event corresponding to $\epsilon$ is given by

$$(\epsilon_y)_{i,j} = \sum_{i'=0}^{u-1} \sum_{j'=0}^{v-1} h_{i',j'} \epsilon_{i-i',j-j'}$$

(4.13)

where $\epsilon$ and $h$ are assumed to be zero outside of their spans. The squared distance of the error event can be written as

$$d^2(\epsilon) = \sum_{i=0}^{m+u-2} \sum_{j=0}^{n+v-2} |(\epsilon_y)_{i,j}|^2.$$ 

(4.14)

By fixing the row number $i = i_0$, we focus on finding a lower bound to the expression

$$\varphi_{i_0} = \sum_{j=0}^{n+v-2} |(\epsilon_y)_{i_0,j}|^2$$

where the distance of the error event is written as

$$d^2(\epsilon) = \sum_{i_0=0}^{m+u-2} \varphi_{i_0}.$$ 

The lower bounds to the expressions $\varphi_{i_0}$ represent three cases:
A. When $0 \leq i_0 < u - 1$, the channel impulse response is not captured in the error event while the convolution is computed. The rows $i_0 + 1$ to $u - 1$ of $h$ are not used in the convolution.

B. When $u - 1 \leq i_0 < m$, the channel impulse response is captured in the error event while the convolution is computed. All rows of $h$ are used in the convolution.

C. When $m \leq i_0 < m + u - 1$, the channel impulse response is not captured in the error event while the convolution is computed. The rows 0 to $i_0 - m$ of $h$ are not used in the convolution.

The lower bounds for these cases can be discussed in the same context, therefore we first start with the simplest of these cases, case B, and then we will apply the results of this case to the other cases.

### 4.7.1 Case A

Let $\epsilon(s)$, $u \leq s < m$, be a subarray of $\epsilon$ containing $u$ rows of $\epsilon$ from 1 to $s - u + 1$, i.e., $\epsilon_{i,j}^{(s)} = \epsilon_{i,j}$ for $s - u < m \leq s$ and $0 \leq j < n$; 0 otherwise. Also let $\epsilon(s,t)$ be a $u \times v$ subarray of $\epsilon$ defined as

$$
\epsilon_{i,j}^{(s,t)} = \begin{cases} 
\epsilon_{i+s-u+1,j+t-v+1}, & \text{if } 0 \leq i < u \text{ and } 0 \leq j < v, \\
0, & \text{otherwise.}
\end{cases}
$$

Note that $\epsilon_{0,0}^{(s,t)} = \epsilon_{s-u+1,t-v+1}$ and $\epsilon_{u-1,v-1}^{(s,t)} = \epsilon_{s,t}$. The channel output error event can be written in terms of the subarrays as follows

$$(\epsilon_y)_{i,j} = \sum_{i' = 0}^{u-1} \sum_{j' = 0}^{v-1} h_{i',j'} \epsilon_{u-i'-1,v-j'-1}^{(s,t)} \triangleq < h, \epsilon^{(s,t)} >.$$

For a fixed row $i_0$, we consider finding a lower bound to $\varphi_{i_0}$ which is given by

$$
\varphi_{i_0} = \sum_{j = 0}^{n+v-2} |(\epsilon_y)_{i_0,j}|^2 = \sum_{j = 0}^{n+v-1} |< h, \epsilon^{(i_0,j)} |^2.
$$

Note that all error coefficients involved in this expression are in the subarray $\epsilon^{(i_0)}$. Let $j_1$ and $j_2$ be the indices of the first and last non-zero column of $\epsilon^{(i_0)}$, respectively. There are two different cases for $\varphi_{i_0}$ depending on whether $\Delta j \triangleq j_2 - j_1$ is larger than $v$ or not.

1. ($\Delta j \geq v$) We propose lower bounds for two interval of $j$: $j_1 \leq j < j_1 + v$ and $j_2 \leq j < j_2 + v$. 


(a) \( j_1 \leq j < j_1 + v \) Let \( E_s, \ 0 \leq j < v \), be the set of all \( u \times v \) matrices of the following form
\[
\begin{bmatrix}
0_{u \times s} & w_{u \times 1} & A_{u \times (v-s-1)}
\end{bmatrix}
\]
whose entries are from the set \( \{+,-,0\} \). The vector \( w \) is a non-zero vector and \( A \) is an arbitrary array. The contribution to the squared distance from the term \( (\epsilon_y)_{i_0,j} \) can be written as
\[
\sum_{j=j_1}^{j_1+v-1} |(\epsilon_y)_{i_0,j}|^2 = \sum_{j=j_1}^{j_1+v-1} |< h, \epsilon^{(i_0,j)} >|^2. \tag{4.15}
\]
Note that \( \epsilon^{(i_0,j)} \in E_s \) where \( s \triangleq n_1 - n + v - 1 \). The subarray \( \epsilon^{(i_0,j)} \) in (4.15) can replaced by an array in \( E_s \) that gives the minimum contribution to the squared-distance. Therefore,
\[
|\epsilon^{(i_0,j)}|_2^2 \geq \min_{\epsilon \in E_s} \ |< h, \epsilon >|^2 \triangleq \theta_s.
\]
Note that \( \theta_s \) does not depend on the actual error event, but the location of non-zero vector \( w \). The total contribution from the terms in the interval \( j_1 \leq j < j_1 + v \) are bounded from below by
\[
\sum_{j=j_1}^{j_1+v-1} |(\epsilon_y)_{i_0,j}|^2 \geq \sum_{s=0}^{v-1} \theta_s.
\]

(b) \( j_2 \leq j < j_2 + v \) Let \( F_t, \ 0 \leq t < v \), be the set of all \( u \times v \) matrices of the following form
\[
\begin{bmatrix}
A_{u \times t} & w_{u \times 1} & 0_{u \times (v-t-1)}
\end{bmatrix}
\]
whose entries are from the set \( \{+,-,0\} \). The vector \( w \) is a non-zero vector and \( A \) is an arbitrary array. The contribution to the squared distance from terms in this interval can be bounded from below by
\[
\sum_{j=j_2}^{j_2+v-1} |(\epsilon_y)_{i_0,j}|^2 = \sum_{j=j_2}^{j_2+v-1} |< h, \epsilon^{(i_0,j)} >|^2 \tag{4.16}
\]
\[
\geq \sum_{j=j_2}^{j_2+v-1} \min_{\epsilon \in F_t} |< h, \epsilon >|^2 \tag{4.17}
\]
\[
\triangleq \sum_{t=0}^{v-1} \psi_t \tag{4.18}
\]
where \( \epsilon^{(i_0,j)} \in F_t \) and \( t \triangleq j_2 - j + v - 1 \).
2. $(\Delta j < v)$ We propose lower bounds for three non-overlapping intervals of \( j \): \( j_1 \leq j < j_2, \ j_2 \leq j < j_1 + v, \) and \( j_1 + v \leq j < j_2 + v. \)

(a) \( (j_1 \leq j < j_2) \) The contributions of the terms in this interval to the distance can be found similar to the case 1.(a):

\[
\sum_{j=j_1}^{j_2-1} |(\epsilon_y)_{i_0,j}|^2 = \sum_{j=j_1}^{j_2-1} |<h, \epsilon^{(i_0,j)}>|^2 \geq \sum_{s=v-\Delta j}^{v-1} \theta_s
\]

where \( \epsilon^{(i_0,j)} \in E_s \) and \( s \triangleq j_1 - j + v - 1. \)

(b) \( (j_2 \leq j < j_1 + v) \) Let \( H_{s,t}, 0 \leq s \leq t < v, \) be the set of all \( u \times v \) matrices of the following form

\[
\begin{bmatrix}
0_{u \times s} & w_{u \times 1} & A_{u \times (t-s-1)} & w'_{u \times 1} & 0_{u \times (v-t-1)}
\end{bmatrix}
\]

whose entries are from the set \{+, -, 0\}. The vectors \( w \) and \( w' \) are non-zero and \( A \) is an arbitrary array. The contribution to the squared distance from the term \( (\epsilon_y)_{i_0,j} \) can be bounded from below by

\[
|(\epsilon_y)_{i_0,j}|^2 = |<h, \epsilon^{(i_0,j)}>|^2 \geq \min_{\epsilon \in H_{s,t}} |<h, \epsilon>|^2 \triangleq \kappa_{s,t}
\]

where \( \epsilon^{(i_0,j)} \in H_{s,t} \) and \( s = j_1 - j + v - 1 \) and \( t = j_2 - j + v - 1. \) The total contribution from this interval is bounded from below by

\[
\sum_{j=j_2}^{j_1+v-1} |(\epsilon_y)_{i_0,j}|^2 \geq \sum_{s=0}^{v-\Delta j-1} \kappa_{s,s+\Delta j}
\]

where \( t = s + \Delta j. \)

(c) \( (j_1 + v \leq j < j_2 + v) \) The contribution of the terms in this interval to the distance can be found similar to the case 1 (b):

\[
\sum_{j=j_1+v}^{j_2+v-1} |(\epsilon_y)_{i_0,j}|^2 = \sum_{j=j_1+v}^{j_2+v-1} |<h, \epsilon^{(i_0,j)}>|^2 \geq \sum_{t=0}^{\Delta n-1} \psi_t
\]

where \( \epsilon^{(i_0,j)} \in F_t \) and \( t \triangleq j_2 - j + v - 1. \)

The lower bounds introduced until this point do not depend on the content of the error event \( \epsilon, \) however they depend on \( \Delta j \) which is not known. The value of \( \Delta j \) can be chosen to minimize the lower bounds as stated in the following lemma.
Lemma 4.3. For \( u - 1 \leq i_0 < m \), a lower bound to \( \varphi_{i_0} \) is given by

\[
\varphi_{i_0} \geq \lambda_{i_0}
\]

where \( \lambda_{i_0} \) is given by

\[
\lambda_{i_0} = \min_{0 < \Delta_j < n} \left\{ \begin{array}{ll}
\sum_{j=0}^{v-1} (\theta_j + \psi_j), & \text{if } \Delta_j \geq v, \\
\sum_{j=0}^{v-1} \theta_j + \sum_{j=0}^{v-\Delta_j-1} \kappa_{j,j+\Delta_j} + \sum_{j=0}^{\Delta_j-1} \psi_j, & \text{if } \Delta_j < v.
\end{array} \right. 
\]

For case A, \( \lambda_{i_0} \) does not change with \( i_0 \), however for the cases B and C, it does.

4.7.2 Case B and Case C

The same method mentioned for case A can be applied to the cases B and C with slight modifications. For case B, the convolution uses only the entries of \( h \) from rows \( 0 \) to \( i_0 \); the other rows are not overlapping with the error event. Let \( h' \) be an impulse response of size \( (i_0 + 1) \times v \) containing the rows \( 0 \) to \( i_0 \) of \( h \). Therefore, one can use \( h' \) instead of \( h \) and derive the corresponding lower bounds to \( \varphi_{i_0} \). In this case, \( u \) should be taken as \( i_0 + 1 \).

For case C, the convolution uses only the entries of \( h \) from rows \( i_0 - m \) to \( u - 1 \); the other rows are not overlapping with the error event. Let \( h'' \) be an impulse response of size \( (u - i_0 + m) \times v \) containing the rows \( i_0 \) to \( m \) of \( h \). Therefore, one can use \( h'' \) instead of \( h \) and derive the corresponding lower bounds to \( \varphi_{i_0} \). In this case, \( u \) should be taken as \( u - i_0 + m \).

Lemma 4.4. A lower bound to the distance of an error event \( \epsilon \) is given by

\[
d^2(\epsilon) \geq \sum_{i=0}^{u-2} \lambda_i + (m - u + 1)\lambda_{u-1} + \sum_{i=m}^{m+u-2} \lambda_i.
\]

An alternative bound to the squared distance can be obtained by considering the convolution column-wise instead of row wise. By fixing a column number \( j_0 \), (4.14) can be written as

\[
d^2(\epsilon) = \sum_{j_0=0}^{n+v-2} \bar{\phi}_{j_0}
\]

where

\[
\bar{\phi}_{j_0} = \sum_{i=0}^{m+u-2} |\langle \epsilon_y \rangle_{i,j_0}|^2.
\]

The following bound can be derived for this case similar to Lemma 4.4.
Table 4.7 Lower bounds to squared distances of error events.

<table>
<thead>
<tr>
<th>$h$</th>
<th>Lower Bound on $d^2(\epsilon)$</th>
<th>Lower Bound on $d^2_{&lt;&gt;}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\begin{bmatrix} 1 &amp; 1 \ 1 &amp; 1 \end{bmatrix}$</td>
<td>$d^2(\epsilon) \geq 4$, for $m, n \geq 1$.</td>
<td>$d^2_{&lt;&gt;} \geq 4$</td>
</tr>
<tr>
<td>$\begin{bmatrix} 0.25 &amp; 1 &amp; 0.25 \ 0.25 &amp; 1 &amp; 1 \end{bmatrix}$</td>
<td>$d^2(\epsilon) \geq 1.25$, for $m, n \geq 1$.</td>
<td>$d^2_{&lt;&gt;} \geq 1.25$</td>
</tr>
<tr>
<td>$\begin{bmatrix} 1 &amp; 1 &amp; 1 \ 1 &amp; 1 &amp; 1 \ 1 &amp; 1 &amp; 1 \end{bmatrix}$</td>
<td>$d^2(\epsilon) \geq \begin{cases} 6, &amp; \text{if } m = 1 \text{ or } n = 1, \ 4, &amp; \text{otherwise}. \end{cases}$</td>
<td>$d^2_{&lt;&gt;} \geq 4$</td>
</tr>
<tr>
<td>$\begin{bmatrix} 1 \ 0.5 \end{bmatrix}$</td>
<td>$d^2(\epsilon) \geq 1.5 + \frac{\max(m,n)-1}{4}$, for $m, n \geq 1$.</td>
<td>$d^2_{&lt;&gt;} \geq 1.5$</td>
</tr>
</tbody>
</table>

**Lemma 4.5.** A lower bound to the distance of a error event $\epsilon$ is given by

$$d^2(\epsilon) \geq \sum_{j=0}^{v-2} \tilde{\lambda}_j + (n - v + 1)\tilde{\lambda}_{v-1} + \sum_{j=n}^{n+v-2} \tilde{\lambda}_j$$

where $\tilde{\lambda}_j$ can be obtained similar to Lemma 4.3.

**Proposition 4.5.** A lower bound to the distance of an error event $\epsilon$ is given by

$$d^2(\epsilon) \geq \max\{\Lambda, \tilde{\Lambda}\}$$

where $\Lambda$ and $\tilde{\Lambda}$ are the lower bounds stated in Lemma 4.4 and Lemma 4.5.

**4.7.3 Results**

The lower bounds on the squared distance of error events are shown in Table 4.7 for the channels given in Examples 4.10, 4.11, 4.12, and 4.13. The minimum squared distances for a fixed error event size are shown in Table 4.8 for comparison. For the 2-D PR1 channel, the algorithm and lower bound gives the same results, while the results are loose for the other channels. However, in all four channels, the lower bound to the minimum closed event distances are consistent with the results of the bounded depth-search algorithm. In the first three channels, the lower bound is almost independent of the error event size, while in the fourth channel, the lower bound is a function of the error event size.

**4.8 Conclusion**

In this chapter, minimum distance and near minimum distance closed error events of the 2-D PR1 channel are characterized. Some open error events are also presented but
Table 4.8 The minimum squared distance for a fixed error event size.

<table>
<thead>
<tr>
<th>m \ n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.25</td>
<td>2</td>
<td>2.25</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>1.75</td>
<td>2</td>
<td>2.25</td>
<td>2.5</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.875</td>
<td>2.25</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2.25</td>
<td>2.25</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>2.25</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m \ n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>2.5</td>
<td>3</td>
<td>3.5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.5</td>
<td>3</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
</tr>
<tr>
<td>3</td>
<td>2.5</td>
<td>3</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>4.5</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

A complete characterization of the open error events is an unsolved problem. The effect of precoding on error events is also investigated. Some distance properties valid for the 2-D PR1 channel also apply to any 2-D ISI channel with impulse response of size $2 \times 2$. Error events for any 2-D channel can be generated by using the bounded-depth search algorithm developed here. The results of this algorithm is observed to be consistent with the analytical results. A computational bound on distances of error events is proposed for any 2-D ISI channel. Although this bound on the distance of an error event is often very loose, it is a good bound on the minimum closed event distance of the channel.

### 4.9 Acknowledgements

A portion of this chapter will be presented in the conference paper: I. Demirkhan, P. H. Siegel and J. K. Wolf, “Error event characterization on 2-D ISI Channels,” in *IEEE International Symposium on Information Theory (ISIT’06)*, July 2006. The dissertation author is the primary author in this paper.
Chapter 5

Block Codes for the Hard-Square Model

5.1 Introduction

Recent advances in optical storage technologies suggest the recording of data on two-dimensional (2-D) surfaces rather than on one-dimensional (1-D) data tracks (see [19] and [20]). Weeks and Blahut proposed several types of 2-D constraints that are useful in reducing the intersymbol interference in 2-D recording channels [8]. One of them is the diamond I constraint which is a binary array on a rectangular grid in which there cannot be neighboring pairs of 1’s either horizontally or vertically (see Fig. 5.1). This constraint is also known as the hard-square model [34].

The construction of block codes for 1-D runlength-limited (RLL) constraints has been the subject of much research [35]. However, the generalization to 2-D constraints seems to be difficult in many ways. First, the non-zero-capacities of 2-D constraints are not analytically known except for the hard-hexagon constraint [36]. Second, 2-D constraints lack descriptions in terms of finite-state-transition-diagrams (FSTD), which have proved to be extremely useful for 1-D constraints [37, 38]. Third, 2-D codewords can have different shapes such as rectangles or parallelograms (see Fig. 5.2(a)-(b)). These shapes can be tiled in different ways as shown in Fig. 5.2(c)-(f). In this chapter, it is assumed that the codewords are rectangular (see Fig. 5.2(a)) and they are tiled as a grid (see Fig. 5.2(c)).

A hard-square array is defined as a binary array where the hard-square constraint holds internally, but not necessarily on the edges when arrays are tiled. For instance, four hard-
square arrays of size $3 \times 3$ are shown in Fig. 5.3. As indicated by the dashed rectangles, there are violations of the hard-square constraint on the edges when these hard-square arrays are tiled. Thus, additional constraints will need to be imposed on the hard-square arrays in order that the hard-square constraint be satisfied at the edges.

Let $\Sigma^{m \times n}$ denote the set of all possible $m \times n$ hard-square arrays. Then, the capacity of the hard-square model is given by

$$ C = \lim_{m,n \to \infty} \log_2 |\Sigma^{m \times n}| / mn. $$

Although this limit exists, no exact calculation of the limit has been proposed. Very tight upper and lower bounds have been calculated such that the limit is known to be approximately given as $0.587891 \ldots$ (see [7, 8, 9, 10]).

For most 1-D constraints, finite-state encoders have been designed such that the resulting code sequence is sliding block decodable. As a special case, there are single-state block encoders, which produce sequences that can be decoded independently from block to block [39]. Finite-state encoders often result in block codes with larger coding rates than that achievable by single-state block codes. In this chapter, the discussion of block
codes for the 2-D hard-square model is divided into two parts: single-state block codes and finite-state block codes. In both cases, the codewords have fixed size $m \times n$.

A single-state (or finite-state) code, $C$, is a collection of $m \times n$ hard-square arrays such that the hard-square constraint is preserved when arrays are tiled. The rate of $C$ is defined as $R_C \triangleq (\log_2 |C|)/mn$. The cardinality of $C$ is not necessarily an integer power of 2. A single-state (or finite-state) block code, $C'$, is defined as a subset of $C$ where there are $2^k$ codewords following each state for an integer $k$. The rate of $C'$ is denoted as $\rho_{C'} \triangleq k/mn$.

### 5.1.1 Single-State Block Codes

A simple rate 0.5 single-state block code exists for the hard-square model called the checkerboard code [8]. Assume that $m \times n$ hard-square arrays are encoded as a checkerboard which is a grid of black and white squares where $m$ and $n$ are even. The codewords are generated by filling white squares arbitrarily with 1's or 0's and black squares with 0's. Because none of the white squares are adjacent to each other, there will be no violation of the hard-square constraint within the array or when the codewords are tiled. Such a code has $2^{mn/2}$ codewords, and yields a code rate of 0.5. Since, it is easy to obtain the checkerboard code with $m = n = 2$, in this chapter we are primarily interested in single-state (block) codes with coding rates larger than 0.5.

A single-state code with the largest codebook size is referred as an optimal single-state code. A method, which is applicable for small values of $m$ and $n$, will be discussed in Section 5.3 to find the optimal single-state codes.

### 5.1.2 Finite-State Block Codes

The ordering by which the codewords of a single-state code are tiled is not important since the constraint is satisfied at the edges for an arbitrary tiling. However, for finite-
state encoders, the ordering of the codewords determines how the state of the encoder is defined. In Fig. 5.4, two different ordering methods are shown for a 4 by 4 square grid of codewords. The number, which is shown on each codeword, is the time stamp of the corresponding codeword in the encoding process. To satisfy the constraint on the edges, the next codeword depends on two codewords that were encoded before. For instance, the codeword 5 in Fig. 5.4(a) only depends on codewords 2 and 3. Note that some of the codewords at the boundary of the 4 by 4 grid only depend on one past codeword. When the grid of codewords is large, it can be assumed that the state of the encoder is determined by two past codewords. For simplicity, in what follows it is assumed that the rectangular grid is large and that the ordering is row-wise as in Fig. 5.4(b).

The choice of the next codeword to encode depends only on the last column of the codeword to its left, \( a \), and the last row of the codeword above it, \( b \). The state of the encoder is denoted as \( (a, b) \) (see Fig. 5.5). Finite-state codes for the hard-square model can be designed by restricting the set of states. Section 5.4 introduces three algorithms \( A \), \( B \), and \( C \) for finding good finite-state codes.

Table 5.1 shows the coding rates for the optimal single-state codes, which are denoted as \( R^* \). The coding rates of finite-state codes are denoted as \( R^A \), \( R^B \), and \( R^C \) for the
Table 5.1 The rates of single-state and finite-state codes

| $m \times n$ | $|\mathcal{H}|$ | $R^*$ | $R^A$ | $R^B$ | $R^C$ |
|-------------|-------------|-------|-------|-------|-------|
| 3 × 3       | 4           | 0.4230| 0.4955| 0.4955| 0.4955|
| 3 × 4       | 4           | 0.4550| 0.5073| 0.5073| 0.5073|
| 3 × 6       | 4           | 0.4643| 0.5320| 0.5320| 0.5320|
| 3 × 8       | 4           | 0.4691| 0.5416| 0.5416| 0.5410|
| 3 × 10      | 4           | 0.4719|       | 0.5498| 0.5498|
| 4 × 4       | 2           | 0.5119| 0.5218| 0.5204| 0.5181|
| 4 × 6       | 2           | 0.5171| 0.5384| 0.5349| 0.5384|
| 4 × 8       | 2           | 0.5198| 0.5474| 0.5465| 0.5460|
| 4 × 10      | 2           | 0.5215|       | 0.5546| 0.5546|
| 5 × 5       | 4           | 0.4872| 0.5400| 0.5400| 0.5400|
| 6 × 6       | 2           | 0.5259| 0.5564| 0.5564| 0.5564|
| 6 × 8       | 2           | 0.5306| 0.5604| 0.5599| 0.5598|
| 6 × 10      | 2           | 0.5335|       | 0.5636| 0.5663|
| 7 × 7       | 8           | 0.5161| 0.5619| 0.5619| 0.5619|
| 8 × 8       | 2           | 0.5367|       | 0.5642| 0.5642|
| 8 × 10      | 2           | 0.5404|       | 0.5690| 0.5691|
| 10 × 10     | 2           | 0.5446|       | 0.5733| 0.5734|

three algorithms $A$, $B$, and $C$, respectively. The rates of the finite-state codes converge to the capacity considerably faster than $R^*$. The parameter $|\mathcal{H}|$ given in Table 5.1 will be addressed in Section 5.3.

This chapter is organized as follows. In Section 5.2, some notation and definitions are given. Single-state and finite-state codes are presented in Sections 5.3 and 5.4, respectively. A method for designing low complexity encoders and decoders for both single-state and finite-state block codes are discussed in Section 5.5. When the codeword size increases, the number of states for the finite-state block codes grows rapidly. A method for reducing the number of states is also presented in this section. A simple way to increase the rate of a single-state block code is presented in Section 5.6 via building large codewords using small codewords. The methods used for finding single-state and finite-state codes are applied to 2-D stripes in Section 5.7.

5.2 Notation and Definitions

A generating template, $T$, is defined as a ternary array whose components are from $\{0, 1, \phi\}$ such that it generates a set of hard-square arrays where the 0’s and 1’s remain fixed in place but the $\phi$’s are variables and can take on the values 0 or 1. There are two
types of generating templates. In one type, which is called an *unconstrained* generating template, the \( \phi \)'s can be filled in all possible ways by 0's and 1's. In the other type, which is called a *hard-square constrained* generating template, the \( \phi \)'s are replaced by 0's and 1's such that the resulting matrices all satisfy the hard-square constraint internally. In this chapter, we consider only hard-square constrained generating templates. For simplicity of notation, a hard-square constrained generating template will be referred to as a generating template.

**Example 5.1.** A \( 4 \times 4 \) generating template is

\[
T_1 = \begin{bmatrix}
0 & \phi & 0 & \phi \\
\phi & \phi & \phi & 0 \\
0 & \phi & \phi & \phi \\
\phi & 0 & \phi & 0
\end{bmatrix}.
\]

In the unconstrained case, \( T_1 \) produces \( 2^{10} = 1024 \) binary arrays. In the hard-square constrained case, it can generate 292 hard-square arrays such that each of them satisfies the hard-square constraint internally. Note that these arrays can be tiled as in Fig. 5.2.c without violating the hard-square constraint. Hence \( T_1 \) can generate a single-state code having 292 codewords, or by eliminating 36 of them, a rate 0.5 single-state block code can be obtained. The formula for calculating the number of hard-square arrays obtained from a generating template is given in (5.1) below.

For an \( m \times n \) generating template, \( T \), the set of hard-square arrays generated from \( T \) is denoted as \( \mathcal{G}(T) \subseteq \Sigma^{m \times n} \). The \( m \times n \) generating template with all \( \phi \)'s generates \( \Sigma^{m \times n} \).

The \( i^{th} \) row and \( j^{th} \) column of \( T \) are denoted as \( T_{i,*} \) and \( T_{*,j} \), respectively.

In the paper by Weeks and Blahut [8], adjacency matrices are used for computing upper and lower bounds to the capacity of the hard-square model. By a slight modification of this method, the cardinality of the generated set from a generating template can be computed. Let \( S_1 \) and \( S_2 \) be two sets of \( 1 \times n \) hard-square arrays. Suppose that \( S_1 \) and \( S_2 \) are lexicographically ordered. The *adjacency matrix* from \( S_1 \) to \( S_2 \) is denoted as \( \mathbf{A}(S_1 \rightarrow S_2) \). The \( (i,j)^{th} \) entry of \( \mathbf{A}(S_1 \rightarrow S_2) \) is defined as 1 if the \( i^{th} \) hard-square array of \( S_1 \) and the \( j^{th} \) hard-square array of \( S_2 \) can be concatenated vertically according to the hard-square model; 0 otherwise.

**Example 5.2.** For \( S_1 = \{[0000],[0100],[0101]\} \) and \( S_2 = \{[0001],[0100],[1000]\} \), the
adjacency matrix from $S_1$ to $S_2$ is given by
\[
A(S_1 \to S_2) = \begin{bmatrix}
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 0 & 1
\end{bmatrix}.
\]
Note that the orderings of $S_1$ and $S_2$ determine the row and column indexes of $A(S_1 \to S_2)$.

The cardinality of the generated set, $\mathcal{G}(T)$, can be computed using adjacency matrices that are defined over the rows of $T$. The cardinality of $\mathcal{G}(T)$ is given by
\[
|\mathcal{G}(T)| = S \left[ \prod_{i=1}^{m-1} A(\mathcal{G}(T_{i,*}) \to \mathcal{G}(T_{i+1,*})) \right]
\tag{5.1}
\]
where $S[\cdot] = 1^T[\cdot] 1$ is the sum of the entries over the product of the matrices in question. The proof of (5.1) is given in [8, Sec. III.A]. The cardinality of $\mathcal{G}(T)$ does not change if the orderings of $1 \times n$ hard-square arrays inside $\mathcal{G}(T_{i,*})$’s are different from the lexicographical ordering.

## 5.3 Single-State Codes

The codewords of a single-state code do not violate the hard-square model when tiled and can be decoded one at a time. For the $m \times n$ array size, since not every subset of $\Sigma^{m \times n}$ is a single-state code, the following is a necessary and sufficient condition for a single-state code.

**Condition 5.1.** Let $C$ be a set of $m \times n$ hard-square arrays. $C$ is a single-state code if and only if $C$ is a subset of $\mathcal{G}(T)$ such that $T$ satisfies the following two conditions:

1. Either $T_{i,1} = 0$ or $T_{i,n} = 0$, for every row $i$, $1 \leq i \leq m$.
2. Either $T_{1,j} = 0$ or $T_{m,j} = 0$, for every column $j$, $1 \leq j \leq n$.

The remaining entries of $T$ are set to $\phi$.

**Remark 5.1.** There are $2^{m+n-3}$ distinct $m \times n$ generating templates satisfying Condition 5.1. The set of these templates is denoted as $\mathcal{U}$. Observe that the generating template, $T_1$, given in Example 5.1 satisfies this condition.
5.3.1 The Optimal Generating Templates

A generating template, \( T \), is called *optimal* if \( G(T) \) has the largest cardinality among all \( G(T') \), \( T' \in \mathcal{U} \). Let \( \mathcal{H} \subseteq \mathcal{U} \) be the set of optimal generating templates, formally

\[
\mathcal{H} = \arg \max_{T \in \mathcal{U}} |G(T)|.
\]  

(5.2)

Condition 5.1 implies that the optimal single-state codes are generated by the optimal generating templates.

The computational complexity to find the optimal generating templates is reduced by means of the pruning algorithm, as discussed by Moison et al [40]. In order to use this algorithm, (5.2) will be simplified in various steps.

The first step is to recognize that the maximization over \( \mathcal{U} \) is equivalent to the maximization over different values of \( T_i* \)'s, \( 1 \leq i \leq m \), such that \( T \in \mathcal{U} \). Then, (5.2) becomes

\[
\mathcal{H} = \arg \max_{T_1*, ..., T_m*} \sum_{i=1}^{m-1} A(G(T_i*) \rightarrow G(T_{i+1}*)).
\]  

(5.3)

The second step is to determine the possible values of \( T_i* \)'s. Two \( 1 \times n \) generating templates, \( P \) and \( Q \), are defined such that \( P = [\phi_{n-1}^T 0] \) and \( Q = [0 \phi_{n-1}^T] \), where \( \phi_{n-1}^T \) is a row vector of \( n-1 \) \( \phi \)'s. In (5.3), \( T_1* \) and \( T_m* \) has to be chosen to satisfy both horizontal and vertical concatenation rules, which are parts 1 and 2 of Condition 5.1, respectively. However, for \( 2 \leq i \leq m-1 \), \( T_i* \) has to satisfy only part 1 of Condition 5.1. Therefore \( T_i* \) is either \( P \) or \( Q \). This restriction on \( T_i* \)'s determines the possible adjacency matrices in (5.3), which are summarized as follows:

- **Starting Matrix:** \( A(G(T_1*) \rightarrow G(T_2*)) \) is either \( A(G(T_1*) \rightarrow G(P)) \) or \( A(G(T_1*) \rightarrow G(Q)) \) according to two possible values of \( T_2* \).

- **Middle Matrices:** For \( 2 \leq i \leq m-2 \), \( A(G(T_i*) \rightarrow G(T_{i+1}*)) \) is one of the four possibilities: \( A(G(P) \rightarrow G(P)) \), \( A(G(P) \rightarrow G(Q)) \), \( A(G(Q) \rightarrow G(P)) \), or \( A(G(Q) \rightarrow G(Q)) \) according to the four possible values of \( T_i* \) and \( T_{i+1}* \).

- **Ending Matrix:** \( A(G(T_{m-1}* \rightarrow G(T_m*)) \) is either \( A(G(P) \rightarrow G(T_m*)) \) or \( A(G(Q) \rightarrow G(T_m*)) \) according to the two possible values of \( T_{m-1}* \).

Note that the starting and ending matrices depend on \( T_1* \) and \( T_m* \), respectively, and are further restricted by part 2 of Condition 5.1.
In the third step, the number of possible middle matrices will be reduced to two using appropriately permuted versions of $G(P)$ and $G(Q)$. However, in general, none of the four possible middle matrices are equal to each other. In the definition of adjacency matrices, the orderings of the $1 \times n$ hard-square arrays inside the sets are lexicographical. On the other hand, the cardinality of any generated set is independent of different orderings of the $1 \times n$ hard-square arrays. Therefore, there may be a different ordering so that some of the four possible middle matrices can be equal to each other.

For a $1 \times n$ binary array, $c = [c_1 \, c_2 \ldots \, c_n]$, a reversing operation on $c$ is defined as $\overline{c} \triangleq [c_n \, c_{n-1} \ldots \, c_1]$. Let $\overline{G(P)}$ be the reversed set of $G(P)$ such that the $i^{th}$ hard-square array of $G(P)$ is defined as $\overline{G(P)}^{(i)} \triangleq \overline{G(P)^{(i)}}$. The sets $\overline{G(P)}$ and $G(Q)$ are the same, but they have different orderings of the hard-square arrays. When $\overline{G(P)}$ is used instead of $G(Q)$, it is easy to obtain the following relations

$$ A(\overline{G(P)} \rightarrow \overline{G(P)}) = A(\overline{G(P)} \rightarrow G(P)) \triangleq M, $$

$$ A(G(P) \rightarrow G(P)) = A(\overline{G(P)} \rightarrow \overline{G(P)}) \triangleq N. $$

Some of the multiplications among the four middle matrices cannot occur in (5.3); e.g., $A(G(P) \rightarrow G(P))A(G(Q) \rightarrow G(P))$. However, the multiplications of $M$ and $N$ can occur in any pattern. Thus, (5.3) can be further simplified by defining $\Omega \triangleq \{M, N\}$. The set of products of $n$ matrices from $\Omega$ is defined as

$$ \Omega^n \triangleq \{ \prod_{i=1}^{n} B_i : B_i \in \Omega \}. $$

Rewriting (5.3) gives

$$ H = \arg \max_{Z \in \Omega^{m-3}} S[A_1ZA_2] \quad (5.4) $$

where

$$ A_1 = A(G(T_{1,*}) \rightarrow G(P)), $$

and

$$ A_2 = A(G(T_{m-1,*}) \rightarrow G(T_{m,*})). $$

In (5.4), we have assumed without loss of generality that $T_{2,*} = P$. In this case, $G(T_{m-1,*})$ is determined as

$$ G(T_{m-1,*}) = \begin{cases} G(P), & \text{if the number of } M \text{'s in } Z \text{ is even} \\ \overline{G(P)}, & \text{otherwise.} \end{cases} $$
5.3.2 The Pruning Algorithm

Before stating the algorithm, we need some preliminaries about matrix norms and
dominations, which are borrowed from [40]. The \( L_1 = \| \cdot \|_1 \) norm of a matrix, \( A \), is
defined as

\[
\|A\|_1 \triangleq \sum_{i,j} |A_{i,j}|.
\]
If every entry of a matrix, \( S \), is non-negative, we write \( S \geq 0 \). A matrix, \( A \), dominates \( B \)
with respect to \( L_1 \) norm if

\[
\|AS\|_1 \geq \|BS\|_1
\]
for all \( S \geq 0 \) and \( S \neq 0 \). Also, \( A \) strictly dominates \( B \) if (5.5) is satisfied without the
equality sign.

The sufficient condition for \( L_1 \) domination is given in [40, Lemma.7]. We write \( A \geq_C B \)
if every column-sum of \( A \) is greater than or equal to the corresponding column-sum of \( B \). Clearly,
if \( A \geq_C B \), then \( A \) dominates \( B \) according to \( L_1 \) norm. Similarly, we write \( A >_C B \)
if \( A \geq_C B \) holds and at least one column-sum of \( A \) is greater than the corresponding
column-sum of \( B \). Hence, if \( A >_C B \), then \( A \) strictly dominates \( B \) according to \( L_1 \) norm.

Let \( \Omega \) be a set of non-negative matrices. A subset \( \Psi \) of \( \Omega \) is dominating if every matrix
in \( \Omega \) is dominated by some matrix in \( \Psi \). A dominating subset of \( \Omega \), \( \Psi' \), is called prime if
every matrix in \( \Psi \) is not strictly dominated by any matrix in \( \Omega \). Because no two distinct
matrices can strictly dominate each other, the prime dominating subset of \( \Omega \) is unique.

Let \( \Psi_n \) be a dominating subset of \( \Omega^n \). Since every matrix of \( \Omega \) is non-negative, \( \Psi_n \Omega \)
is a dominating subset of \( \Omega^{n+1} \), i.e, \( \Psi_{n+1} \subseteq \Psi_n \Omega \). Using prime dominating subsets, one
can construct the following recursive algorithm to compute the prime dominating subset
of \( \Omega^n \), \( \Psi'_n \):

\begin{itemize}
  \item \( \Psi'_1 = \text{prime dominating subset of } \Omega \).
  \item \( \Psi'_n = \text{prime dominating subset of } \Psi_{n-1} \Omega \) for \( n \geq 2 \).
\end{itemize}

This algorithm is called the pruning algorithm [40]. Here, prime dominating subsets are
used for simplifying the computations, since they are unique.

The pruning algorithm will be modified to reduce the computational complexity of
solving (5.4). Suppose that \( \Psi_i \) is a subset of \( A(\mathcal{G}(T_{i,*}) \to \mathcal{G}(P)) \Omega^{i-1} \). Let \( \Psi^e_i \subseteq \Psi_i \) be a
set of matrices which contains an even number of \( M \)'s as multipliers. Every matrix in \( \Psi^e_i \)
has \( A(\mathcal{G}(T_{i,*}) \to \mathcal{G}(P)) \) as the last multiplier, where \( T_{i,*} \) is \( P \) or \( \tilde{P} \). The set of the next
possible multipliers are the same for every matrix in $\Psi^e_i$. Therefore they can be compared using $L_1$ domination. Similarly, let $\Psi^o_i$ be the set of matrices in $\Psi_i$ which contains an odd number of $M$’s as multipliers. Every matrix in $\Psi^o_i$ has $A(G(T_{i,*}) \rightarrow (\bar{P}))$ as the last multiplier. Therefore, the matrices in $\Psi^o_i$ can be compared in terms of $L_1$ dominance. However, we cannot compare matrices from $\Psi^e_i$ and $\Psi^o_i$, since the set of the next possible matrices are different. Thus, it is necessary to modify the pruning algorithm as follows:

- $\Psi^e_1 = A(G(T_1,*) \rightarrow G(P))$
- $\Psi^e_2 = A(G(T_1,*) \rightarrow G(P))N$ and $\Psi^o_2 = A(G(T_1,*) \rightarrow G(P))M$
- $\Psi^e_i$ is the p.d.s. of $[\Psi^o_{i-1}M] \cup [\Psi^e_{i-1}N]$
- $\Psi^o_i$ is the p.d.s. of $[\Psi^e_{i-1}M] \cup [\Psi^o_{i-1}N]$
- $\Psi_{m-1} = \cup_{i \geq 1} \{ [\Psi^e_{m-2}A(G(T_{m,*}) \rightarrow G(T_{m,*}))] \}$ and $[\Psi^o_{m-2}A(G(T_{m,*}) \rightarrow G(T_{m,*}))]$
- $Z = \arg \max_{Z \in \Psi_{m-1}} \|Z\|_1$.

Here p.d.s. stands for prime dominating subset. For every matrix in $Z$, an optimal generating template can be extracted from its multipliers. In this way, the set of optimal generating templates, $H$, is obtained from $Z$. The optimal generating templates in $H$ generate the optimal single-state codes, whose coding rates, $R^*$, are listed in Table 5.1 along with the number of the distinct optimal single-state codes, $|H|$. 

**Example 5.3.** One of the four optimal $3 \times 6$ generating templates is

$$T_2 = \begin{bmatrix} \phi & 0 & \phi & 0 & \phi & 0 \\ \phi & \phi & \phi & \phi & 0 \\ 0 & \phi & 0 & \phi & \phi \end{bmatrix}.$$ 

The other three generating templates are just reflections of $T_2$ along the horizontal or vertical axes or both. All of them yield 328 hard-square arrays, resulting in the coding rate $R^*_{3,6} \approx 0.4643$.

For $T_2$ given in Example 5.3, the difference between the number of 0’s and $\phi$’s along the edges is at most 1. The generating template, $T_1$, given in Example 5.1 has the property that the number of 0’s and $\phi$’s are the same along the edges. Let $w_x(E)$ be the number of $x$’s in an edge, $E$, of a generating template. We conjecture that the following inequality is true for the edges of an optimal generating template, $T$:

$$|w_0(E) - w_{\phi}(E)| \leq 1, \quad (5.6)$$
where $E$ can take the values $T_{1,s}$, $T_{m,s}$, $T_{s,1}$, and $T_{s,n}$. In fact, this claim is true for all optimal generating templates whose coding rates are listed in Table 5.1. Unfortunately, not every generating template satisfying (5.6) is optimal. The following are two instances of generating templates which are not optimal:

$$T_3 = \begin{bmatrix} 0 & \phi & 0 & \phi \\ 0 & \phi & \phi & \phi \\ \phi & \phi & \phi & 0 \\ \phi & 0 & \phi & 0 \end{bmatrix}, \quad T_4 = \begin{bmatrix} 0 & \phi & \phi & 0 & 0 & \phi \\ \phi & \phi & \phi & \phi & 0 \\ \phi & 0 & 0 & \phi & 0 \end{bmatrix}$$

where $|G(T_3)| = 145$ and $|G(T_4)| = 242$. However, the optimal $4 \times 4$ and $3 \times 6$ generating templates can generate 292 and 328 hard-square arrays, respectively.

Another property of even-length boundaries is that 0’s and φ’s are ordered in an interleaved fashion. For example, $T_1$ given in Example 5.1 satisfies this property. In fact, this property holds for all optimal generating templates whose coding rates are listed in Table 5.1 for even $m$ and $n$. Under this property and (5.6), if $m$ and $n$ are even, then there can be at most two different optimal generating templates such that one of them is the mirror image of the other.

The corresponding property for the odd-length edges is that there is only one run of two 0’s or two φ’s, while the other symbols are interleaved. All optimal generating templates in Example 5.3 satisfy this property. However, not every generating template that satisfies this property is optimal.

### 5.4 Finite-State Codes

In the 1-D case, for a rate $k/n$ block code, Franaszek introduced the concept of principal states which have the property that there are at least $2^k$ codewords of length $n$ starting from a principal state and terminating in another principal state [41]. Although, there is no obvious way to define states for 2-D constrained sequences which are imposed by the hard-square model, we can assume the definition which was illustrated in Fig. 5.5. Let $G$ be a labeled graph representation of the encoder. The encoder moves from state $(a, b)$ to $(c, d)$ by emitting the codeword $w \in \Sigma^{m \times n}$ as long as the hard-square constraint is satisfied. Since $d$ does not depend on $w$, $G$ is not deterministic. Hence, the block code condition described in [41] is not applicable to this case. However, a special case of this problem where the states are defined over $V \times H$, $V \subseteq \Sigma^{m \times 1}$, $H \subseteq \Sigma^{1 \times n}$ is easy to solve by algorithms which search for $(V, H)$ pairs that result in good finite-state codes.
Definition 5.1. Let $Z \subseteq \Sigma^{m \times 1}$ be a generating template. The complementary generating template of $Z$ is defined as

$$Z_{i,1} = \begin{cases} \phi, & \text{if } Z_{i,1} = 0 \\ 0, & \text{otherwise} \end{cases}$$

for all $i$. In other words, $\mathcal{G}(Z)$ is the largest subset of $\Sigma^{m \times 1}$ such that hard-square arrays of $\mathcal{G}(Z)$ can be vertically concatenated with every hard-square array of $\mathcal{G}(Z)$. Likewise, the generating template $\overline{W}$ can be defined from $\mathcal{G}(W) \subseteq \Sigma^{1 \times n}$. For hard-square arrays $a \in \Sigma^{m \times 1}$ and $b \in \Sigma^{1 \times n}$, their complementary templates, $\overline{a}$ and $\overline{b}$, can be similarly defined.

The hard-square function $h[a, b, V, H]$ is defined as the number of hard-square arrays given by the state $(a, b) \in V \times H$ and the boundary sets $V$ and $H$. The hard-square function is analytically unknown. If it were known for arbitrary $m$ and $n$, one could find the capacity of the hard-square model. Numerically, this function can be computed using (5.1) for small values of $m$ and $n$. Let $c \in V$ and $d \in H$ be two hard-square arrays as shown in Fig. 5.6. The generating template, $T(\overline{a}, \overline{b}, c, d)$, is defined as all $\phi$’s except those that satisfy $T_{*,1} = \overline{a}$, $T_{1,*} = \overline{b}$, $T_{*,n} = c$, $T_{m,*} = d$. The number of hard-square arrays following the state, $(a, b)$, is given by

$$h[a, b, V, H] = \sum_{c \in V, d \in H} |\mathcal{G}(T(\overline{a}, \overline{b}, c, d))|.$$

Example 5.4. For the $3 \times 3$ case, the value of the hard-square function for each state is shown in Table 5.2 where the boundary sets are $V_0 = \Sigma^{3 \times 1}$ and $H_0 = \Sigma^{1 \times 3}$. We refer to this table as the hard-square table of $V_0$ and $H_0$. The smallest entries are shown in boldface. Let $\delta(V_0, H_0)$ be the minimum value of the hard-square table of $V_0 \times H_0$. The elimination of
either \([010]^T\) or \([101]^T\) may be useful to avoid one of the smallest entries at \((3, 5)\). Similarly, the elimination of either \([101]^T\) or \([010]^T\) may be useful to remove the other smallest entry at \((5, 3)\). After eliminating \([101]^T\) and \([101]^T\), we are left with the sets \(V_1\) and \(H_1\). The hard-square table of \(V_1\) and \(H_1\) is shown in Table 5.3, giving \(\delta(V_1, H_1) > \delta(V_0, H_0)\). An immediate question is whether \(\delta(V_1, H_1)\) is the largest among all possible pairs of \((V, H)\), \(V \subseteq \Sigma^{m \times 1}\), \(H \subseteq \Sigma^{1 \times n}\) or not. In the next subsection, it is shown that \((V_1, H_1)\) is the only pair which attains the largest value for the \(3 \times 3\) case.

For any array size \(m \times n\), the problem is to find \((V^*, H^*)\) pairs such that \(\delta(V^*, H^*)\) is the largest among all possible pairs. Formally,

\[
(V^*, H^*) = \arg \max_{V \subseteq \Sigma^{m \times 1}, H \subseteq \Sigma^{1 \times n}} \delta(V, H)
\]

(5.8)

where

\[
\delta(V, H) = \min_{(a, b) \in V \times H} h[a, b, V, H]
\]

(5.9)

and superscripts denote the pairs which lead to the largest finite-state codes.

A measure of complexity for solving (5.8) is the number of required cardinality computations using (5.1). Therefore the complexity of \(h[a, b, V, H]\) is \(|V||H|\). The complexity for solving (5.8) is upper bounded by

\[
N_{BF} = f_m^2 f_n^2 2^{f_m + f_n}
\]

(5.10)
where \( f_m = |\Sigma^{m \times 1}|, f_n = |\Sigma^{n \times 1}| \) and \( N_{BF} \) is the complexity of a brute-force algorithm. Note that \( f_1 = 2, f_2 = 3 \) and \( f_m = f_{m-1} + f_{m-2} \) for \( n \geq 3 \). When \( m \) is sufficiently large,

\[
f_m \approx \frac{1}{\sqrt{5}} \left( \frac{1 + \sqrt{5}}{2} \right)^{m+2}.
\]

Thus, the complexity of the brute-force algorithm is doubly exponential. It is desirable to reduce this complexity by heuristic methods.

### 5.4.1 Algorithm A

In Table 5.2, for a fixed row \( a \in V_0 \), \( h[a, [000], V_0, H_0] \geq h[a, b, V_0, H_0] \) for all \( b \in H_0 \). The same is true for Table 5.3 although the values are different. A partial ordering needs to be defined over \( \Sigma^{m \times 1} \) and \( \Sigma^{1 \times n} \) to explain this.

**Definition 5.2.** For \( a, b \in \Sigma^{m \times 1} \), if every position of \( a \) is smaller than or equal to the corresponding position of \( b \), then \( a \) is smaller than or equal to \( b \). Formally,

\[
a \leq b \iff \{a_i, 1 \leq i \leq m \} \subseteq \{b_i, 1 \leq i \leq m \}.
\]

We write \( a < b \), if at least one position of \( b \) is larger than the corresponding position of \( a \).

For example, \([000] \leq b\) for every \( b \in \Sigma^{1 \times 3} \). Similarly the partial ordering can be defined over \( \Sigma^{1 \times n} \).

**Proposition 5.1.** Let \((a, b)\) and \((a', b')\) be two states of \( V \times H \) such that \( a \leq a' \). Then, for all \( b \in \Sigma^{1 \times n} \)

\[
h[a, b, V, H] \geq h[a', b, V, H].
\]

**Proof.** Let \( Q \) be the set of the hard-square arrays defined by the state \((a, b)\) and the boundary sets \( V \) and \( H \). Similarly, \( Q' \) is defined for \((a', b)\), \( V \) and \( H \). Since \( G(a') \subseteq G(a) \) and \( c \in Q' \), then \( c \in Q \). The proposition follows due to \( Q' \subseteq Q \). \(\square\)

**Remark 5.2.** In general, for \((a, b), (a', b') \in V \times H \) such that \( a \leq a' \) and \( b \leq b' \),

\[
h[a, b, V, H] \geq h[a', b', V, H]. \tag{5.11}
\]

Therefore, it is not necessary to compute all entries of the hard-square table in order to find the minimum value of it. Let \( V_S \subseteq V \) be a set of \( m \times 1 \) hard-square arrays such that
whenever \( a \in V_S \), there exists \( a' \in V \) and such that \( a < a' \). Also, \( V_L \triangleq V \setminus V_S \). Similar subsets can be defined for \( H \). Thus, (5.9) is simplified as

\[
\delta(V, H) = \min_{(a, b) \in V_L \times H_L} h[a, b, V, H].
\] (5.12)

It is also possible to eliminate some pairs that cannot be a solution to (5.8).

**Proposition 5.2.** Let \((V^*, H^*)\) be a solution to (5.8). Let \( a \in V^* \) and \( a' \in \Sigma^m \times 1 \) be such that \( a' \leq a \). If we extend \( V^* \) by \( a' \) to obtain \( Z = V^* \cup \{a'\} \), then \((Z, H^*)\) is also a solution to (5.8).

**Proof.** We need to prove that

\[
\min_{(p, q) \in Z \times H^*} h[p, q, Z, H^*] \geq \min_{(p, q) \in V^* \times H^*} h[p, q, V^*, H^*].
\] (5.13)

If \( p \in V^* \) and \( p \neq a' \), then the following expression is true for every \( q \in H^* \) due to the definition of the hard-square function (5.7):

\[
\min_{(p, q) \in V^* \times H^*} h[p, q, Z, H^*] \geq \min_{(p, q) \in V^* \times H^*} h[p, q, V^*, H^*].
\] (5.14)

If \( p = a' \), then for every \( q \in H^* \),

\[
\begin{align*}
h[a', q, Z, H^*] & \quad \overset{(a)}{\geq} \quad h[a', q, V^*, H^*] \\
& \quad \geq \quad h[a, q, V^*, H^*].
\end{align*}
\]

where \((a)\) follows from the definition of the hard-square function, (5.7), and \((b)\) follows from the assumption and (5.11). Thus,

\[
\min_{q \in H^*} h[a', q, Z, H^*] \geq \min_{q \in H^*} h[a, q, V^*, H^*].
\] (5.15)

Note that (5.13) follows from (5.14) and (5.15). \(\square\)

**Remark 5.3.** A similar result can be stated for \( H^* \). The extension of \( V^* \) and/or \( H^* \) with the condition of Proposition 5.2 usually increases the number of next codewords. Hence, it is sufficient to consider the pairs satisfying the following condition: If \((a, b) \in V \times H \) and \((a', b') \in \Sigma^m \times 1 \times \Sigma^1 \times n \) such that \( a' \leq a \) and \( b' \leq b \), then \((a', b') \in V \times H \).

Algorithm \( A \) searches over all pairs satisfying the condition in Remark 5.3. Finding the minimum value of the hard-square table is facilitated by using (5.12). The coding rates of the finite-state codes which result from algorithm \( A \) are shown in Table 5.1. Compared
with single-state codes, $R^A$ converges to the capacity faster than $R^*$. The complexities of algorithm $A$ and the brute-force algorithm are compared in Fig. 5.7 for $m = n$. The parameter $N_A$ denotes the number of cardinality computations required by algorithm $A$. The parameter $N_{BF}$ is computed directly from (5.10). Algorithm $A$ has significantly less complexity than that of the brute-force algorithm for large array sizes, such as $6 \times 6$ and $7 \times 7$.

**Example 5.5.** For the $4 \times 4$ case, algorithm $A$ yields the following pair


$$H^* = \{ [0000], [1000], [0100], [0010], [0001], [0101] \}.$$  

This solution is symmetric in the sense that if $a \in V^*$, then $a^T \in H^*$.

**Example 5.6.** For the $6 \times 6$ case, the pair $(V^*, H^*)$ has the special property that $V^*$ and $H^*$ contain all the hard-square arrays which have weight less than 3. Similar to Example 5.5, the solution is symmetric.

Also, the solution pair for the $3 \times 3$ case has the same property. In that case, the hard-square arrays with weight less than 1 are included in the pair. This property is often
satisfied for the pairs found by algorithm A. However, it is occasionally violated as was the case for Example 5.5.

5.4.2 Algorithm B

This algorithm is based on the intelligent guessing discussed in Example 5.4. Some hard-square arrays from V and/or H are eliminated based on a metric, which is defined from the hard-square table. To reduce the complexity, only a portion of the hard-square table, which is defined by \( V_L \times H_L \), is taken into account. For \( a \in V_L \), the metric \( \mu(a) \) is defined as

\[
\mu(a) = \frac{1}{|H_L|} \sum_{b \in H_L} h[a, b, V, H].
\]

Similarly, for \( b \in H_L \), \( \mu(b) \) is defined as

\[
\mu(b) = \frac{1}{|V_L|} \sum_{a \in V_L} h[a, b, V, H].
\]

Algorithm B is initialized with the pair \((V, H) = (\Sigma^{m \times 1}, \Sigma^{1 \times n})\). After computing the hard-square table for the states \( V_L \times H_L \), the metric for each state is obtained. The hard-square array with the smallest metric is eliminated. Then, the algorithm proceeds with the updated pair until there is no state remaining in the hard-square table.

The rates of the finite-state block codes using algorithm B are shown in Table 5.1. For most values of \( m \times n \), \( R^B = R^A \). The complexity of this algorithm is upper bounded by \((f_n + f_m)f_m^2f_n^2\). The simulated complexity of this algorithm is shown in Fig. 5.8 for \( m = n \). Although this algorithm does not provide solutions to (5.8), it is significantly simpler than algorithm A.

5.4.3 Algorithm C

This algorithm is an extension of the notion that the solutions to (5.8) have the proper weight properties such as in the 3 \( \times \) 3 and 6 \( \times \) 6 cases. The set \( \Sigma^{m \times 1} \) is partitioned as \( \{V_0, V_1, \ldots, V_{\lceil \frac{m}{2} \rceil}\} \) where \( V_i \) is the set of \( m \times 1 \) hard-square arrays having weight \( i \). Similarly, \( \Sigma^{1 \times n} \) is partitioned as \( \{H_0, H_1, \ldots, H_{\lceil \frac{n}{2} \rceil}\} \). Algorithm C searches over pairs \((V, H)\) where \( V = \cup_{i=0}^{\lceil \frac{m}{2} \rceil} V_i \) and \( H = \cup_{j=0}^{\lceil \frac{n}{2} \rceil} H_j \) for \( 0 \leq i \leq \lceil \frac{m}{2} \rceil \) and \( 0 \leq j \leq \lceil \frac{n}{2} \rceil \). Hence, the complexity of this algorithm is upper bounded by \((\lceil \frac{m}{2} \rceil + 1)(\lceil \frac{n}{2} \rceil + 1)f_m^2f_n^2\).
5.4.4 Comparison of Codes

The rates of the resulting finite-state block codes using these algorithms are shown in Table 5.1. Although algorithm $C$ is much less complex than algorithm $A$, $R^C$ is very close to $R^A$. Algorithm $C$ is slightly less complex than algorithm $B$ for large block sizes (see Fig. 5.8). For the same block sizes, the three algorithms often give the same coding rate, such as in the $7 \times 7$ case.

5.5 Low Complexity Encoders and Decoders

The Constraint-Tree algorithm, as discussed by Modha and Marcus [42], is used for obtaining low complexity encoders and decoders for 1-D constrained codes. This algorithm can be generalized to 2-D single-state and finite-state block codes. Using this algorithm, a generating template is decomposed into a set of simple generating templates. The simple generating templates having the largest cardinalities are chosen in the codebook design in order to simplify the complexity of the encoder and decoder.
5.5.1 Decomposition of Generating Templates

An \( m \times n \) generating template, \( T \), is called simple whenever \(|G(T)| = 2^{w_\phi(T)}\). If \( T \) is simple, the positions with \( \phi \)'s are called free, whereas the other positions with 0's or 1's are called fixed. For a non-simple generating template \( T' \), two positions \((i, j)\) and \((k, l)\) are called dependent if there are two possible assignments of pairs of bits which can be applied to these two positions, and these possible assignments are either \( \{(0, 0), (1, 1)\} \) or \( \{(0, 1), (1, 0)\} \).

Example 5.7. The \( 4 \times 4 \) generating template \( T_1 \) given in Example 5.1 is not simple, since \(|G(T_1)| = 292\) and \( w_\phi(T_1) = 10\). However,

\[
T_5 = \begin{bmatrix}
0 & \phi & 0 & \phi \\
\phi & 0 & \phi & 0 \\
0 & \phi & 0 & \phi \\
\phi & 0 & \phi & 0
\end{bmatrix}
\]

is simple since \(|G(T_5)| = 256\) and \( w_\phi(T_5) = 8\). An 8-bit binary input string can be easily inserted into the 8 free positions in \( T_5 \).

Proposition 5.3. For a generating template, \( T \), under the hard-square model, there are no pairs of dependent coordinates.

The proof follows from the fact that for any hard-square array, changing a value from a 1 to a 0 gives another hard-square array.

As indicated in Example 5.7, the mapping of the input stream into a simple generating template is easy. Thus, generating templates are decomposed into the minimum number of disjoint simple generating templates using the 2-D Constraint-Tree algorithm, which is an extension of the 1-D Constraint-Tree algorithm. The 2-D Constraint-Tree algorithm decomposes an \( m \times n \) generating template \( T \) as follows:

- **Step 1**: For every position of \( T \), check whether it is fixed or not. If it is fixed, assign that position to its fixed value.

- **Step 2**: If \(|G(T)| = 2^{w_\phi(T)}\), then \( T \) is a simple generating template. Terminate the algorithm.

- **Step 3**: If \( T \) is not simple, for every position \((i, j)\) with \( \phi \), compute \( Z_{i,j} \) as the number of hard-square arrays generated by \( T \) such that coordinate \((i, j)\) takes the
value 0. Then, find the most constrained position

\[(\hat{i}, \hat{j}) = \arg \min_{(i,j)} \min(Z_{i,j}, |T| - Z_{i,j})\]

Ties in ‘argmin’ are broken by selecting the smallest row. If there is still a tie, it is broken by selecting the column with the smallest index.

- **Step 4:** Partition \(T\) into two disjoint generating templates and recurse

  - \([T_0]_{i,j} = T_{i,j}\) for every position except that \([T_0]_{\hat{i},\hat{j}} = 0\), then go to the step 1 by taking \(T = T_0\).

  - \([T_1]_{i,j} = T_{i,j}\) for every position except that \([T_1]_{\hat{i},\hat{j}} = 1\), then go to the step 1 by taking \(T = T_1\).

**Example 5.8.** Let \(T_6(\pi, \bar{b}, c, d)\) be a \(4 \times 4\) generating template, where \(a = [1000]^T\), \(b = [0100]^T\), \(c = [0001]^T\), and \(d = [1001]^T\) (see Fig. 5.6):

\[
T_6 = \begin{bmatrix}
0 & 0 & \phi & 0 \\
\phi & \phi & \phi & 0 \\
0 & \phi & \phi & 0 \\
1 & 0 & 0 & 1
\end{bmatrix}.
\]

Since \(|G(T_6)| = 20\) and \(w_\phi(T_6) = 6\), \(T_6\) is not simple. The 2-D Constraint-Tree algorithm decomposes \(T_6\) into the following four simple \(4 \times 4\) generating templates:

\[
W_1 = \begin{bmatrix}
0 & 0 & \phi & 0 \\
\phi & 0 & 0 & 0 \\
0 & 0 & \phi & 0 \\
1 & 0 & 0 & 1
\end{bmatrix},
W_2 = \begin{bmatrix}
0 & 0 & \phi & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \phi & 0 \\
1 & 0 & 0 & 1
\end{bmatrix},
\]

\[
W_3 = \begin{bmatrix}
0 & 0 & 0 & 0 \\
\phi & 0 & 1 & 0 \\
0 & \phi & 0 & 0 \\
1 & 0 & 0 & 1
\end{bmatrix},
W_4 = \begin{bmatrix}
0 & 0 & \phi & 0 \\
\phi & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1
\end{bmatrix}.
\]

The sets, \(G(W_i)\)’s, are disjoint and \(\sum_{i=1}^{4} |G(W_i)| = |G(T_6)|\). We define the *simple generating template decomposition* of \(T_6\) as

\[
\mathcal{D}(T_6) \triangleq \{W_1, W_2, W_3, W_4\}.
\]
Table 5.4 Generating polynomials of optimal generating templates

| $m \times n$ | $|\mathcal{D}(T)|$ | $p_T(x)$ |
|----------------|-------------------|-----------|
| 3 × 5          | 8                 | $x^3 + 3x^4 + 2x^5 + 2x^2$ |
| 4 × 4          | 4                 | $x^3 + 2x^4 + x^2$ |
| 4 × 6          | 16                | $x^{12} + 4x^8 + 3x^6 + 2x^5 + 3x^4 + 2x^3 + x^2$ |
| 4 × 8          | 66                | $x^{16} + 6x^{12} + 5x^{10} + 4x^{09} + 8x^8 + 10x^7 + 11x^6 + 8x^5 + 8x^3 + 4x^2 + x$ |
| 6 × 6          | 143               | $x^{18} + 8x^{15} + 9x^{12} + 12x^{11} + 19x^{10} + 31x^9 + 26x^8 + 16x^7 + 14x^6 + 6x^5 + x^4$ |
| 6 × 8          | 1307              | $x^{24} + 12x^{20} + 15x^{18} + 22x^{17} + 43x^{16} + 87x^{15} + 120x^{14} + 136x^{13} + 206x^{12} + 211x^{11} + 194x^{10} + 113x^9 + 74x^8 + 40x^7 + 20x^6 + 10x^5 + 3x^4$ |

Let every simple generating template $W_i$ be represented as $x^{w_{\phi}(W_i)}$. The generating polynomial of $T_6$ is given by

$$p_{T_6}(x) = x^3 + 3x^2.$$ 

In general, for an arbitrary $m \times n$ generating template $T$, the generating polynomial is given by

$$p_T(x) = \sum_{i=0}^{mn} a_i x^i,$$

where $a_i$ represents the number of simple generating templates of $\mathcal{D}(T)$ having $i$ free coordinates. Note that $p_T(1) = |\mathcal{D}(T)|$ and $p_T(2) = |\mathcal{G}(T)|$. Table 5.4 shows the generating polynomials of some optimal generating templates discussed in Section 5.3. Note that $|\mathcal{D}(T)|$ grows very quickly with increasing array size. However, it is not necessary to use all of the simple generating templates in the codebook design. In the succeeding subsections, the descriptions of low complexity encoders and decoders will be discussed based on the paper by Modha and Marcus [42, Section III.A-C].

#### 5.5.2 Codebook Design

The codebook design procedure for single-state and finite-state block codes are similar to each other. We first discuss the codebook design for single-state block codes.

Suppose that $T$ is an optimal $m \times n$ generating template obtained by the pruning algorithm. Let $\mathcal{D}(T)$ be the decomposition of $T$ into the minimum number of simple generating templates. We want to design a single-state block code $C \subseteq \mathcal{G}(T)$ such that
\( |C| = 2^k \) for integer \( k \). Let the generating polynomial of \( T \) be
\[
p_T(x) = \sum_{i=0}^{mn} a_i x^i.
\]
The simple generating templates, which can generate the largest number of hard-square arrays, are included into the codebook so as to reduce the complexity. Since \( p_T(2) \geq 2^k \), there exist unique integers \( r \geq 0 \) and \( b > 0 \) such that
\[
\sum_{i=r+1}^{mn} a_i 2^i + b 2^r \geq 2^k > \sum_{i=r+1}^{mn} a_i 2^i + (b - 1)2^r
\]
where \( 0 < b \leq a_r \). In fact, in [42, Section III.A] it is proved that
\[
\sum_{i=r+1}^{mn} a_i 2^i + b 2^r = 2^k. \tag{5.16}
\]
Thus, the codebook includes all of the simple generating templates which have \( r + 1 \) or a higher number of free positions and \( b \) of the \( a_r \) simple generating templates with \( r \) free positions. From the left hand side of (5.16), the generating polynomial for \( C \) is defined as
\[
p_C(x) = \sum_{i=r+1}^{mn} a_i x^i + bx^r.
\]
**Example 5.9.** One of the optimal \( 6 \times 8 \) generating templates is
\[
T_7 = \begin{bmatrix}
0 & \phi & 0 & \phi & 0 & \phi & 0 & \phi \\
\phi & \phi & \phi & \phi & \phi & \phi & 0 \\
0 & \phi & \phi & \phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi & \phi & 0 \\
0 & \phi & \phi & \phi & \phi & \phi & \phi & \phi \\
\phi & 0 & \phi & 0 & \phi & 0 & \phi & 0 \\
\end{bmatrix},
\]
which can generate an optimal single-state code with rate \( R^*_6,8 \approx 0.5306 \). \( \mathcal{D}(T_7) \) consists of 1307 simple generating templates. The generating polynomial of \( T_7 \) is given in Table 5.4. A rate \( \frac{25}{48} \) single-state block code, \( C \), can be designed using \( T_7 \). From (5.16), we obtain \( r = 17 \) and \( b = 2 \). The generating polynomial for the codebook \( C \) is
\[
p_C(x) = x^{24} + 12x^{20} + 15x^{18} + 2x^{17}.
\]
Surprisingly, the number of simple generating templates used for designing \( C \) is only 30 out of 1307 templates. One can arbitrarily choose 2 out of the 22 simple generating templates, which have 17 free positions.
Table 5.5 Generating polynomials of some single-state block codes

<table>
<thead>
<tr>
<th>m x n</th>
<th>$\rho_C$</th>
<th>$p_C(1)$</th>
<th>$p_C(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 x 8</td>
<td>25/48</td>
<td>30</td>
<td>$x^{24} + 12x^{20} + 15x^{18} + 2x^{17}$</td>
</tr>
<tr>
<td>8 x 8</td>
<td>33/64</td>
<td>17</td>
<td>$x^{32} + 16x^{28}$</td>
</tr>
<tr>
<td>8 x 8</td>
<td>34/34</td>
<td>771</td>
<td>$x^{32} + 18x^{28} + 25x^{24} + 40x^{20} + 100x^{16} + 213x^{12} + 374x^8$</td>
</tr>
<tr>
<td>8 x 10</td>
<td>41/80</td>
<td>17</td>
<td>$x^{40} + 16x^{36}$</td>
</tr>
<tr>
<td>8 x 10</td>
<td>42/80</td>
<td>246</td>
<td>$x^{40} + 24x^{36} + 35x^{32} + 58x^{28} + 128x^{24}$</td>
</tr>
</tbody>
</table>

In Example 5.9, the coding rate 25/48 is in fact the largest coding rate that a 6 x 8 single-state block code can achieve. In general, the largest coding rate for an $m \times n$ single-state block code is given by

$$\rho_{m,n}^* = \left\lfloor \frac{mnR_{m,n}^*}{mn} \right\rfloor,$$

(5.17)

where $\lfloor \cdot \rfloor$ is the floor function, which prevents single-state block codes from achieving the rates of optimal single-state codes. In Section 5.6, a method for reaching the rates of the optimal single-state codes is discussed by introducing more dependency among codewords.

As it will be clarified in the encoder design, one of main parameters to measure the complexity of a single-state block code is the number of simple generating templates that it contains, or $p_C(1)$. The generating polynomials of some single-state block codes are shown in Table 5.5. There are two 8 x 8 single-state block codes having rates 33/64 and 34/64. Even though the difference of their rates is small, the one with the larger rate is several times more complex than the other.

For finite-state block codes with codeword size $m \times n$, let $(V, H)$ be a pair of boundary sets which is found by either algorithm $A$, $B$, or $C$. For a state $(a, b) \in V \times H$, the set of hard-square arrays that follow the state $(a, b)$ is denoted as $F(a, b)$. Formally,

$$F(a, b) = \bigcup_{(c, d) \in V \times H} T(\bar{a}, \bar{b}, c, d).$$

We want to design a finite-state block code such that $|F(a, b)| \geq 2^k$ for all states $(a, b) \in V \times H$. Since the codebook after each state is different, it is necessary to design $|V| \times |H|$ different codebooks. Let $C(a, b) \subseteq F(a, b)$ be the codebook for the state $(a, b)$. Then, the finite-state block code, C, can be defined as

$$C = \{C(a, b)|((a, b) \in (V, H))\}.$$ 

The design of $C(a, b)$ is the same as the design of a rate $k/mn$ single-state block code. Basically, $F(a, b)$ is decomposed into the set of simple generating templates $\mathcal{D}(F(a, b))$. 
Then the simple generating templates are added to $C(a, b)$ from $D(F(a, b))$ in descending order of their cardinalities. The largest coding rate for finite-state block codes is

$$\rho_C = \left\lfloor \frac{mnR(V,H)}{mn} \right\rfloor$$

(5.18)

where $R(V,H)$ is the rate associated with the pair $(V,H)$.

5.5.3 Encoder and Decoder Descriptions

The single-state block encoder is designed by using the decomposition of the codebook $D(C)$. Each template $T \in D(C)$ is assigned to a set of input words, which has the same cardinality as $G(T)$. Input words are chosen such that they constitute a 1-D generating template with $w_\phi(T)$ free coordinates. For finite-state block codes, since different codebooks are considered for different states, the encoder chooses the codebook $C(a, b)$ for the state $(a, b)$. Then, it proceeds in the same way as the single-state block encoder by using $D(C(a, b))$. Each input word is assigned to $|V| \times |H|$ codewords, one for each codebook.

Let $X = \{b_1, \ldots, b_k\}$ be an unconstrained input sequence. The encoder maps $X$ into the codeword $Y \in C$ in two steps as shown in Fig. 5.9(a).

- **The Simple Template Encoder** selects the simple generating template which is used as a layout for representing $X$. Let $D(C) = \{T_1, \ldots, T_p\}$ be the set of all simple generating templates in $D(C)$. The probability mass function over $D(C)$ is defined as

$$t(T_i) = \frac{2^{w_\phi(T_i)}}{2^k} = 2^{\ell_i},$$

where $\ell_i \triangleq k - w_\phi(T_i)$. Since $t(\cdot)$ is a dyadic distribution, the corresponding prefix-free code can be found by using the Huffman algorithm [43]. Let $L(T_i)$ be the corresponding prefix to $T_i$ in such a prefix-free code. The generating template encoder divides $X$ into two binary strings:

$$X_{prefix} = \{b_1, \ldots, b_{\ell_i}\},$$

$$X_{free} = \{b_{\ell_i+1}, \ldots, b_k\}.$$  

The $X_{prefix}$ part is used for selecting the generating template $T_i$ out of $D(C)$, i.e, $X_{prefix} = L(T_i)$. The $X_{free}$ part is used for filling in all free positions of $T_i$.

- **The Index Mapper** maps $X_{free}$ into free positions in $T_i$ to obtain $Y$ according to a 2-D mapping function $g(T_i)$, which is different for each template. This mapping has
Figure 5.9 (a) Encoder and (b) decoder block diagrams.

to be one-to-one and onto provided that $|X_{\text{free}}| = w_{\phi}(T_i)$. Index mapper functions, which have low encoding and decoding complexities, are discussed in [42, Sec. III.C].

The decoding process is simply the inverse of the encoding process. The hard decisions are made at the receiver so that $\hat{Y} \subseteq \{0, 1\}^m \times n$ is the received binary array. The decoder maps $\hat{Y}$ into the input word estimate $\hat{X}$ in two steps as shown in Fig. 5.9(b):

- **The Simple Template Decoder** compares the positions in $\hat{Y}$ with the corresponding fixed positions of $T_i$ for all templates in $D(C)$. If it does not match to any of simple generating templates in $C$, the decoder will generate an error and sets $\hat{X}$ to a random sequence. Let $\hat{T}$ be the generating template whose fixed coordinates matches $\hat{Y}$. Since, the simple generating templates of $C$ are disjoint, there can be no two distinct generating templates that match $\hat{Y}$.

- **The Inverse Index Mapper** reconstructs $\hat{X}$ using both $L(\hat{T})$ and the free positions in $\hat{T}$

$$\hat{X} = \{L(\hat{T}), g^{-1}(\hat{T})\}.$$  

Because $g$ is one-to-one and onto, $g^{-1}$ exists.

**Example 5.10.** Fig. 5.10 shows the simple template encoder designed for the rate $25/48$ $6 \times 8$ single-state block code in Example 5.9. A 25-bit input word is assigned a simple $6 \times 8$ generating template according to its prefix. For example if input word starts with 0, the encoder will choose the simple generating template, which has 24 free positions. The rest of the input sequence is copied by the index mapper into the free positions in the selected generating template.
Figure 5.10 The simple template encoder for Example 5.10.

Table 5.6 The hard-square table of \((V_1, H_1)\)

| \(a\) \(b\) \(000\) \(010\) \(001\) |
|\(\phi^T\) \(29\) \(27\) \(27\) |
|\(010\) \(27\) \(30\) \(22\) |
|\(001\) \(27\) \(22\) \(34\) |

5.5.4 Reduction of States

For finite-state codes, it is possible to decompose the set of states into a set of simple generating templates in order to reduce the number of states.

**Example 5.11.** For the \(3 \times 3\) array size, the pair that satisfies (5.8) given in Example 5.4 is

\[ V_1 = \{ [000]^T, [001]^T, [010]^T, [100]^T \}, \]
\[ H_1 = \{ [000], [001], [010], [100] \}. \]

The hard-square arrays \([000]^T\) and \([100]^T\) can be combined into the simple generating template \([\phi 00]^T\). Similarly, \([000]\) and \([100]\) can be combined into \([\phi 00]\). The minimum number of codewords following all states does not change as shown in Table 5.6. In this way, the number of states is reduced from 16 to 9.

In general, this method can be extended to any array size. Let \((V, H)\) define a finite-state code. The decomposition of \(V\) and \(H\) are obtained by using the 1-D Constraint-Tree algorithm. Let these decompositions be enumerated as \(D(V) = \{ V_1, V_2, \ldots, V_x \}\) and \(D(H) = \{ H_1, H_2, \ldots, H_y \}\). The sets of corresponding complementary generating templates are \(\{ \overline{V}_1, \overline{V}_2, \ldots, \overline{V}_x \}\) and \(\{ \overline{H}_1, \overline{H}_2, \ldots, \overline{H}_y \}\). In this case, the state of the encoder is defined.
Table 5.7 The number of states for finite-state codes by algorithm A

| $m \times n$ | $|V^*| \times |H^*|$ | $xy$ | $m \times n$ | $|V^*| \times |H^*|$ | $xy$ |
|-------------|-----------------|-----|-------------|-----------------|-----|
| $2 \times 2$ | 4 | 1 | $4 \times 4$ | 36 | 9 |
| $2 \times 4$ | 8 | 1 | $4 \times 6$ | 105 | 40 |
| $2 \times 6$ | 20 | 2 | $4 \times 8$ | 155 | 68 |
| $2 \times 8$ | 50 | 4 | $6 \times 6$ | 389 | 100 |
| $3 \times 3$ | 16 | 9 | $6 \times 8$ | 782 | 170 |
| $3 \times 6$ | 68 | 30 | $7 \times 7$ | 529 | 225 |
| $3 \times 8$ | 196 | 57 | | | |

as a pair of generating templates $(V_i, H_j)$. The set of codewords following the state, $(V_i, H_j)$, can be expressed as

$$F(V_i, H_j) = \bigcup_{k,l} T(V_i, H_j, V_k, H_l).$$

The number of codewords following the state, $(V_i, H_j)$, is given by

$$h[V_i, H_j, V, H] = \sum_{k,l} |G(T(V_i, H_j, V_k, H_l))|.$$ 

In this way, the number of states is reduced from $|V| \times |H|$ to $xy$. In the worst case, the 1-D Constraint-Tree algorithm results in $x = |V|$ and $y = |H|$. However, this method usually reduces the number of states significantly in practice. For the pairs $(V^*, H^*)$ that are the solutions to (5.8), the number of states and the reduced number of states are shown in Table 5.7.

The following proposition claims that the coding rate for single-state and finite-state block codes achieves the capacity when the codeword size approaches infinity.

**Proposition 5.4.** The rates of single-state and finite-state block codes asymptotically achieve the capacity of the hard-square model, i.e,

$$C = \lim_{m,n \to \infty} \frac{|mnR|}{mn},$$

where $R$ is either $R^*$, $R^A$, $R^B$, or $R^C$ that are evaluated for the $m \times n$ case.

The proof is based on the fact that the number of codewords for all four cases is lower bounded by the cardinality of the set $\Sigma^{(m-2)\times(n-2)}$. 
5.6 Increasing The Rate of Single-State Block Codes

The multiple codewords of an optimal single-state block code, $C$, can be used for designing a larger single-state block code, $C'$, such that $\rho_C \geq \rho_{C'}$.

**Example 5.12.** Let $C$ be an optimal $6 \times 4$ single-state code with the coding rate, $R^*_6,4 \approx 0.5171$. However, one of the largest $6 \times 4$ single-state block codes has the coding rate, $\rho^*_6,8 = 0.5$. Using four codewords of $C$ arranged as a $12 \times 8$ codeword, a single-state block code, $C'$, can be obtained with the coding rate, $\rho_{C'} = 49/96$. This code is referred to as a $2 \times 2 - 6 \times 4$ single-state block code.

One of the optimal $6 \times 4$ generating templates is

\[
T_8 = \begin{bmatrix}
0 & \phi & 0 & \phi \\
\phi & \phi & \phi & 0 \\
0 & \phi & \phi & \phi \\
\phi & \phi & \phi & 0 \\
0 & \phi & \phi & \phi \\
\phi & 0 & \phi & 0
\end{bmatrix}
\]

which has the generating polynomial

\[p_{T_8}(x) = x^{12} + 4x^8 + 3x^6 + 2x^5 + 3x^4 + 2x^3 + x^2.\]

There should be $2^{49}$ codewords chosen from the $(p_{T_8}(2))^4 \approx 2^{49.64}$ possible hard-square arrays. Using the first two terms of $p_{T_8}(x)$, there are

\[(2^{12} + 4 \cdot 2^8)^4 > 2^{49}\]

(5.19)
codewords. Hence, it is not necessary to use the simple generating templates that have less than 8 free positions. Let $A, B, C, D, E$ be an enumeration of five simple generating templates, where $A$ has 12 free positions and $B, C, D, E$ have 8 free positions. Also, let $a, b, c, d, e$ be their cardinalities, respectively. Rewriting the left hand side of (5.19) gives

\[(a + b + c + d + e)^4 = a^4 + 4a^3b + 4a^3c + 4a^3d + 4a^3e + g(a, b, c, d, e).\]

(5.20)
The first five terms in the right side of (5.20) add to $2^{49}$. Therefore, it is not necessary to use $g(a, b, c, d, e)$. The first term, $a^4$, in (5.20) corresponds to the use of four $A$’s as a $12 \times 8$ codeword (see Fig. 5.11(a)). The coefficient 4 in front of the remaining terms
except \( g(a, b, c, d, e) \) is obtained by combining \( B, C, D \) or \( E \) with three \( A \)'s as shown in Fig. 5.11(b). Hence there are 17 different simple generating templates which give the coding rate of 49/96.

This method can be generalized to a \( q \times s \times m \times n \) single-state block code, \( C \), provided that \( \rho_C = k/(qsmn) \leq R^*_m,n \). The codebook design procedure can be summarized in the following four steps:

- Define

  \[
  K \triangleq \sum_{i=r+1}^{mn} a_i 2^i + b 2^r,
  \]

  and find the unique integers \( r \geq 0 \) and \( b \) such that

  \[
  K^{qs} \geq 2^k > (K - 2^r)^{qs},
  \]

  where \( 0 < b \leq a_r \).

- Enumerate the generating templates that are represented by the polynomial

  \[
  \sum_{i=r+1}^{mn} a_i 2^i + b 2^r
  \]

  as \( A_1, A_2, \ldots, A_z \) where \( z = \sum_{i=r+1}^{mn} a_i + b \). Let \( w_1, w_2, \ldots, w_z \) be their cardinalities, respectively. The choice of \( b \) out of the \( a_r \) generating templates having \( r \) free positions is arbitrary.

- Expand the expression

  \[
  (w_1 + \cdots + w_z)^{qs}.
  \]  

  (5.21)
A particular term in this expansion is
\[ cw_1^{p_1} w_2^{p_2} \cdots w_z^{p_z}, \]
where \( \sum_{i=1}^{z} p_i = qs \) and \( c = (p_1^{qs}, p_2^{qs}, \ldots, p_z^{qs}) \). This term represents the use of \( p_i \) instances of \( A_i \), for all \( 1 \leq i \leq z \). Also, there have to be \( c \) different orientations of \( A_i \)'s. Suppose that all generating templates are distinguishable. There are \( (qs)! \) different combinations in an \((qm, sn)\) array. However, for all \( i \), \( p_i \) of them are alike, giving
\[ \frac{(qs)!}{p_1! p_2! \cdots p_z!} \]
different combinations, which is equal to \( c \). The terms in (5.21) are sorted in decreasing value of \( w_1^{p_1} \cdots w_z^{p_z} \).

- Take the terms from left to right until \( 2^k \) codewords are collected.

The \( q \times s - m \times n \) single-state block codes have larger coding rates than their constituent single-state block codes as shown in the following proposition.

**Proposition 5.5.** Let \( \rho \) and \( \rho' \) be the largest coding rates of \( m \times n \) and \( q \times s - m \times n \) single-state block codes. Then,
\[ 0 \leq \rho' - \rho < \frac{1}{mn}, \]
and
\[ \lim_{l \to \infty} \rho' = R^*_{m,n}, \]
where \( l = qs \).

### 5.7 Block Codes for 2-D Stripes

The methods discussed in the previous sections can be applied to 2-D stripes of width \( m \). In this case, the hard-square arrays are tiled only horizontally as shown in Fig. 5.12 so that the hard-square model can be considered as a 1-D constraint working on a larger alphabet. The number of states for the induced 1-D constraint of the hard-square model is \( f_m \), which grows exponentially with increasing \( m \). There are various 1-D coding schemes based on the FSTD's of 1-D constraints\[38\]. The methods discussed in this section may serve as an alternative to known methods for designing low complexity block encoders and decoders for 2-D stripes.
Table 5.8 The rates of single-state and finite-state codes for 2-D stripes

<table>
<thead>
<tr>
<th>$m \times n$</th>
<th>$R^s$</th>
<th>$R^{A}$</th>
<th>$m \times n$</th>
<th>$R^s$</th>
<th>$R^{A}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2 \times 2$</td>
<td>0.5000</td>
<td>0.5805</td>
<td>$4 \times 2$</td>
<td>0.5000</td>
<td>0.5490</td>
</tr>
<tr>
<td>$2 \times 4$</td>
<td>0.5490</td>
<td>0.6072</td>
<td>$4 \times 4$</td>
<td>0.5386</td>
<td>0.5799</td>
</tr>
<tr>
<td>$2 \times 6$</td>
<td>0.5756</td>
<td>0.6167</td>
<td>$4 \times 6$</td>
<td>0.5606</td>
<td>0.5906</td>
</tr>
<tr>
<td>$2 \times 8$</td>
<td>0.5903</td>
<td>0.6215</td>
<td>$4 \times 8$</td>
<td>0.5731</td>
<td>0.5960</td>
</tr>
<tr>
<td>$2 \times 10$</td>
<td>0.5994</td>
<td>0.6244</td>
<td>$4 \times 10$</td>
<td>0.5808</td>
<td>0.5992</td>
</tr>
</tbody>
</table>

$C_{2,\infty}$ | 0.6358 |

| $6 \times 2$  | 0.5000 | 0.5537 |
| $6 \times 4$  | 0.5351 | 0.5793 |
| $6 \times 6$  | 0.5555 | 0.5876 |
| $6 \times 8$  | 0.5672 | 0.5917 |
| $6 \times 10$ | 0.5745 | 0.5942 |

$C_{6,\infty}$ | 0.6040 |

$C_{4,\infty}$ | 0.5878 |

$C_{8,\infty}$ | 0.5913 |

Figure 5.12 Tiling hard-square arrays horizontally.

5.7.1 Single-State Codes

The condition for a single-state code for 2-D stripes of width $m$ is the same as Condition 5.1 in Section 5.3 except part 2. There are $2^m$ different generating templates satisfying part 1 of Condition 5.1. The optimal generating templates for this case are the solution of

$$H = \arg \max_{Z \in \Omega^{m-1}} S[Z],$$

(5.22)

where $\Omega = \{ M, N \}$, which is defined in Section 5.3.1. This maximization problem is significantly simpler than its 2-D equivalent, (5.4), therefore it can be efficiently solved by using the pruning algorithm. An optimal single-state code is generated by an optimal generating template of $H$.

The rates of optimal single-state codes, $R^s$, are shown in Table 5.8. Note that these rates are not symmetric with respect to $m$ and $n$. When the thickness of the stripe increases, the capacity of the induced constraint decreases from the capacity of 1-D $(1, \infty)$ RLL constraint, 0.6942, to the capacity of the hard-square model, 0.5878. The rates of optimal single-state codes seem to follow this behavior. For a fixed $n$, when $m$ increases,
$R^*$ decreases. The effect of $n$ on the coding rates is opposite to the effect of $m$.

**Example 5.13.** An optimal $4 \times 6$ generating template for 2-D stripes of width 4 is

$$T_9 = \begin{bmatrix}
0 & \phi & \phi & \phi & \phi & \\
\phi & \phi & \phi & \phi & 0 & \\
0 & \phi & \phi & \phi & \phi & \\
\phi & \phi & \phi & \phi & 0 & 
\end{bmatrix},$$

Using this template, a rate $13/24$ single-state block code, $C$, can be designed. The generating polynomials of $T_9$ and $C$ are

$$p_{T_9}(x) = x^{12} + 4x^9 + 4x^8 + 8x^7 + 21x^6 + 37x^5 + 26x^4 + 9x^3 + 3x^2,$$

$$p_C(x) = x^{12} + 4x^9 + 4x^8 + 8x^7.$$

### 5.7.2 Finite-State Codes

The algorithms for finding finite-state codes for 2-D stripes is a special case of what was discussed in Section 5.4. The state of the encoder for this case is defined as the last column of the previous codeword. Let $V$ be a set of states. The hard-square function for the state, $a \in V$, is computed by

$$h[a, V] = \sum_{b \in V} \left| G(T(\bar{a}, b)) \right|,$$

where $T(\bar{a}, b)$ is the generating template bounded by $\bar{a}$ and $b$ as shown in Fig. 5.13. Similar to the 2-D case, the goal is to find the sets $V^* \subseteq \Sigma^{m \times 1}$ such that the minimum number of following codewords for all states is the largest. Formally,

$$V^* = \arg \max_{V \subseteq \Sigma^{m \times 1}} \min_{a \in V} h[a, V]. \quad (5.23)$$

The complexity of a brute-force algorithm for solving (5.23) is $f_m^2 2^{f_m}$, which is doubly exponential as in the 2-D case.

Algorithm $A$ mentioned in Section 5.4.1 can be easily modified to solve (5.23). The coding rates of the finite-state codes designed by using this algorithm, $R^A$, are shown in Table 5.8 together with the corresponding capacity values. For a fixed $m$, $R^A$ converges to $C_{m,\infty}$ much faster than $R^*$. 
Figure 5.13 A generating template, $T(\pi, b)$, following the state, $a \in V$.

**Example 5.14.** A $4 \times 2$ finite-state code designed by using algorithm $A$ has the set of states, $V_{4,2}^*$, which is obtained by eliminating the state $[1001]^T$, i.e,

$$V_{4,2}^* = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$ 

The rate of this code is $R_{4,2}^A \approx 0.5490$.

### 5.8 Conclusion

In this chapter, single-state and finite-state block codes for the hard-square model are investigated. The methods and algorithms mentioned in this chapter can be applied to other first order constraints, such as the hard-hexagon constraint. The lack of a graphical representation for 2-D constraints is still the main difficulty in developing a general theory.

### 5.9 Acknowledgements

Chapter 6

Depth-First Algorithm for Designing 2-D Single-State Block Codes

6.1 Introduction

In Chapter 5, single-state and finite-state block codes have been proposed for the hard-square model. The methods introduced in Chapter 5 can be extended to certain constraints, such as the hard-hexagon model, where 1’s are isolated horizontally, vertically, and diagonally. However, in practice constrained codes are frequently used for eliminating undesired error patterns from the channel detector output. Since different channel targets often require different error patterns to be eliminated, general coding schemes working for most constraints are more desirable in practice. In this chapter, we propose a depth-first algorithm for finding the single-state block codes for 2-D constraints represented by a set of forbidden patterns.

A 2-D constrained system is a restriction on input arrays that are written onto the recording medium. There are some undesired input arrays that make the detector fail or introduce high ISI to the readback signal. The common pieces of these undesirable arrays usually can be reduced to a set of unwanted patterns, which are called forbidden patterns. Most 2-D constrained systems can be described in terms of sets of forbidden patterns. Let $S$ be a 2-D constrained system represented by a set of forbidden patterns $F(S)$. An example of such 2-D constrained system is the hard-square model whose set of forbidden
Table 6.1 Some 2-D constraints and their capacity values.

<table>
<thead>
<tr>
<th>$S$</th>
<th>$\mathcal{F}(S)$</th>
<th>$C(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-isolated-1 (n.i.1)</td>
<td>$\begin{bmatrix} 0 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>0.9617</td>
</tr>
<tr>
<td>Non-isolated-bit (n.i.b.)</td>
<td>$\begin{bmatrix} 0 &amp; 1 &amp; 0 \ 0 &amp; 1 &amp; 1 \end{bmatrix}$</td>
<td>0.9238</td>
</tr>
<tr>
<td>Square non-isolated-1 (s.n.i.1)</td>
<td>$\begin{bmatrix} 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>0.9973</td>
</tr>
<tr>
<td>Square non-isolated-bit (s.n.i.b.)</td>
<td>$\begin{bmatrix} 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
<td>0.9945</td>
</tr>
<tr>
<td>Hard-square (h.s.)</td>
<td>$\begin{bmatrix} 1 &amp; 1 \ 1 &amp; 1 \end{bmatrix}$</td>
<td>0.5878</td>
</tr>
<tr>
<td>Hard-hexagon (h.h.)</td>
<td>$\begin{bmatrix} 1 &amp; 1 \ 1 &amp; 1 \end{bmatrix}$</td>
<td>0.4808</td>
</tr>
</tbody>
</table>

patterns is

$$\mathcal{F}(S) = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$  

Table 6.1 shows the 2-D constraints considered in this chapter. The hard-square and hard-hexagon models have been studied in various papers [8, 7, 10]. Non-isolated bit (n.i.b.) and square n.i.b. (s.n.i.b.) constraints can be useful in practical applications (see Section 6.4). Non-isolated 1 (n.i.1) and square n.i.1 (s.n.i.1) constraints are also included since the complexity of the depth-first algorithm is small for them. The codes designed in this chapter have rectangular codewords and the codewords are tiled as a grid as shown in Fig. 5.2(c).

A **constrained array** is defined as a binary array where the constraint holds internally, but not necessarily on the edges when the arrays are tiled. Let $\Sigma(S)^{m \times n}$ denote the set of all possible $m \times n$ constrained arrays under the constrained system $S$. Then, the capacity of $S$ is given by

$$C(S) = \lim_{m,n \to \infty} \frac{\log_2 |\Sigma(S)^{m \times n}|}{mn}.$$  

This limit exists, but an exact calculation of this limit is not known for the constraints shown in Table 6.1, except the hard-hexagon model [36]. The other capacity values are obtained by using the fast-converging approximation method proposed by Marrow [5].
Table 6.2 The rates of the optimal single-state codes for various constraints.

<table>
<thead>
<tr>
<th>$m \times n$</th>
<th>n.i.1</th>
<th>n.i.b.</th>
<th>s.n.i.1</th>
<th>s.n.i.b.</th>
<th>h.s.</th>
<th>h.h.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3 \times 3$</td>
<td>.8851</td>
<td>.7753</td>
<td>.9596</td>
<td>.9055</td>
<td>.4230</td>
<td>.3522</td>
</tr>
<tr>
<td>$3 \times 4$</td>
<td>.8955</td>
<td>.8006</td>
<td>.9633</td>
<td>.9178</td>
<td>.4550</td>
<td>.3475</td>
</tr>
<tr>
<td>$3 \times 5$</td>
<td>.9033</td>
<td>.8147</td>
<td>.9678</td>
<td>.9281</td>
<td>.4467</td>
<td>.3524</td>
</tr>
<tr>
<td>$3 \times 6$</td>
<td>.9083</td>
<td>.8242</td>
<td>.9709</td>
<td>.9351</td>
<td>.4643</td>
<td>.3615</td>
</tr>
<tr>
<td>$4 \times 4$</td>
<td>.9054</td>
<td>.8205</td>
<td>.9671</td>
<td>.9286</td>
<td>.5119</td>
<td>.3509</td>
</tr>
<tr>
<td>$4 \times 5$</td>
<td>.9124</td>
<td>.8336</td>
<td>.9711</td>
<td>.9375</td>
<td>.4923</td>
<td>.3643</td>
</tr>
<tr>
<td>$4 \times 6$</td>
<td>.9170</td>
<td>.8421</td>
<td>.9740</td>
<td>.9438</td>
<td>.5171</td>
<td>.3701</td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>.9188</td>
<td>.8455</td>
<td>.9747</td>
<td>.9454</td>
<td>.4872</td>
<td>.3768</td>
</tr>
<tr>
<td>$5 \times 6$</td>
<td>.9231</td>
<td>.8534</td>
<td>.9773</td>
<td>.9510</td>
<td>.5058</td>
<td>.3826</td>
</tr>
<tr>
<td>$6 \times 6$</td>
<td>.9272</td>
<td>.8608</td>
<td>.9797</td>
<td>.9551</td>
<td>.5259</td>
<td>.3982</td>
</tr>
</tbody>
</table>

A single-state code $C$ is a collection of $m \times n$ constrained arrays such that the constraint is preserved when the arrays are tiled. The rate of $C$ is defined as $R_C \triangleq (\log_2 |C|)/mn$. The cardinality of $C$ is not necessarily an integer power of 2. A single-state block code $C'$ is defined as a subset of $C$ where $|C'| = 2^k$ for an integer $k$. The rate of $C'$ is denoted as $\rho_{C'} \triangleq k/mn$. A single-state code with largest codebook size is referred as an optimal single-state code. Table 6.2 shows the coding rates for the optimal single-state codes obtained by the depth-first algorithm. The parameter $N$ is the number of different optimal single-state codes, which will be discussed in Section 6.3.

This chapter is organized as follows. The depth-first algorithm for finding optimal single-state codes with rectangular codewords is presented in Section 6.2. The results of the depth-first algorithm are discussed in Section 6.3. For certain 2-D constraints, this algorithm has lower complexity compared with the brute-force algorithm. The coding rate asymptotically approaches the capacity of the 2-D constrained system when the codeword size increases. A simple coding example is demonstrated in Section 6.4.

### 6.2 The Depth-First Algorithm

Suppose that we want to find the optimal single-state codes for the n.i.1 constraint when the codeword size is $3 \times 3$. Let $p$ denote the forbidden pattern of the n.i.1 constraint. When $p$ is located at codeword boundaries, it can overlap with adjacent codewords. There are three situations of $p$ across the codeword window as shown in Fig. 6.1(a)-(c):
Figure 6.1 (a)-(c) Different situations of $p$ across the $3 \times 3$ codeword window, (d) the number forbidden pieces for each position in the codeword window.

1. $p$ is captured inside the codeword. In this case, the codewords having $p$ at that position have to be removed from the codebook.

2. $p$ is overlapping with two or three adjacent codewords. Each overlapping piece of $p$ is called a forbidden piece. In order to satisfy the constraint, at least one of the forbidden pieces should be eliminated for all codewords in the codebook.

If the smaller forbidden piece shown in Fig. 6.1(b) is selected, the number of removed codewords is at most $2^8$ by assuming that there is no constraint imposed on the other positions. In this case, the rate of the optimal single-state code is upper bounded by $R \leq 8/9 \approx .8889$. On the other hand, if the larger forbidden piece is selected, the number of removed codewords is at most $2^5$ and $R \leq \log_2(2^9 - 2^5) \approx .9897$. Therefore, it may be an efficient strategy to remove forbidden pieces with larger areas.

Suppose that the forbidden pattern $p$ is centered at the $(i, j)$th entry of the codeword. The set of forbidden pieces that $p$ introduces at position $(i, j)$ is called a decision set, $\lambda_{i,j}(p)$. Figure 6.1(d) shows the cardinalities of decision sets for different positions of the codeword. For each position, one forbidden piece has to be selected for elimination, therefore the brute-force algorithm searches $3^4 2^4 = 1296$ different situations to find the optimal single-state codes. In general for the n.i.1 constraint, the brute-force algorithm searches $3^4 2^2(m+n-4)$ number of different situations for the codeword size of $m \times n$.

Building a tree of decisions is useful to reduce the complexity of searching for the optimal single-state codes. We start from the upper-left corner of Fig. 6.1(d) to make decisions about which forbidden pieces will be eliminated from decision sets. Then we pass over all positions clockwise to generate a decision tree, which is shown in Fig. 6.2. The value of the root node shows the number of remaining codewords after eliminating the forbidden pattern captured inside of the codeword, i.e., the decision at the center position of Fig. 6.1(d) has already been made. The value of the root node can be obtained as
follows. Let $B_1$ be the set of all $3 \times 3$ binary arrays. The set of codewords having pattern $p$ inside is given by

$$B_2 = \left\{ \begin{bmatrix} x_1 & 0 & x_2 \\ 0 & 1 & 0 \\ x_3 & 0 & x_4 \end{bmatrix} \mid x_1, x_2, x_3, x_4 \in \{0, 1\} \right\}.$$ 

Clearly, $|B_1| = 512$ and $|B_2| = 16$. Therefore there are 496 remaining codewords. On the next level, the root node is expanded into three leaves depending on the selection of the forbidden piece from the decision set

$$\lambda_{1,1}(p) = \left\{ (0)_{1,3}, (0)_{3,1}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{1,1} \right\}$$

where the subscripts of the forbidden pieces denotes the position of their upper-left entry in the codeword. For instance, let $B_3$ be a subset of the root node, $B_1 \setminus B_2$, without having 0 at the $(1,3)$th entry of its codewords. There are 248 codewords in $B_3$ satisfying this property as shown in the right-most node of level $l = 1$. In this way, each node value in the decision tree becomes the number of remaining codewords after the decisions are made from the root node up to that node. Since the codewords are removed at each level, the decision tree is non-increasing.

When the decision tree is expanded to all levels by selecting the leaf node with the largest value, we obtain a node with value 250 at the bottom level, $l = 8$. The value of this node is a candidate to be the largest among all nodes at level $l = 8$. At level $l = 1$, 

Figure 6.2 Decision tree of $3 \times 3$ codewords for the n.i.1 constraint.
it is not necessary to expand the two right nodes with value 248, since the decision tree is non-increasing. In fact, in the other levels there is no unexpanded node with value larger than or equal to 250. Therefore, the node with the largest number of codewords is found only in 21 node computations. On the other hand, the brute-force algorithm computes the value of all of the 1296 leaf nodes at level $l = 8$.

The depth-first algorithm can be generalized to any 2-D constraints represented by a set of forbidden patterns. Let $\mathcal{F}(\mathcal{S}) = \{p_i\}_{i=1}^t$ be a set of forbidden patterns. Suppose that each pattern $p$ is confined to a rectangle where empty positions are filled with don’t care symbols, $\delta$. For instance, the pattern of n.i.1 constraint is represented as

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

The pattern $p$ is said to be located at the $(i,j)$th position of the codeword window, if the center of $p$ is $(i,j)$. If there is no center position in $p$, i.e. one of dimensions of $p$ is even, then one of the positions closest to the center is taken as the center of the pattern. Each forbidden pattern $p$ produces a set of forbidden pieces at position $(i,j)$, which is called the decision set $\lambda_{i,j}(p)$. The forbidden pieces containing only “don’t care” symbols are ignored and not included in decision sets.

In order to satisfy $\mathcal{S}$ for a particular location $(i,j)$, at least one forbidden piece from each pattern has to be eliminated. Let $\Lambda_{i,j}$ be the Cartesian product of the decision sets for the $(i,j)$th position, i.e.,

$$\Lambda_{i,j} = \{(f_1, \ldots, f_t) | f_l \in \lambda_{i,j}(p_l)\}.$$ 

The cardinality of $\Lambda_{i,j}$ is given by

$$|\Lambda_{i,j}| = \prod_{l=1}^t |\lambda_{i,j}(p_l)|.$$ 

If all forbidden patterns at the $(i,j)$th position are confined in the codeword window, then $|\Lambda_{i,j}| = 1$. Let $I_1$ be the set of positions such that $|\Lambda_{i,j}| = 1$ and $I \setminus I_1$ be the set of positions having more than 1 choice. The decision tree is constructed as follows:

- The root node $\nu^{(0)}$ is the set of remaining codewords after eliminating the codewords having forbidden pieces from $\Lambda_{I_1}$. The value of $\nu^{(0)}$ is its cardinality $|\nu^{(0)}|$. 
Let $I'$ be a permutation of $I \setminus I_1$. For a level $l$, $1 \leq l \leq |I'|$, a node $\nu^{(l)}_i$ is the set of remaining codewords after eliminating the codewords having forbidden pieces $(\Lambda_{I'})_i$ from the node $\nu^{(l-1)}$. Therefore, $|\nu^{(l)}_i| \leq |\nu^{(l-1)}|$ for all $i$.

The optimal single-state codes are a set of nodes with the largest value at the bottom level $l = |I'|$. The brute-force algorithm searches for all nodes at the bottom level $|I'|$. A measure for the complexity of the depth-first and the brute-force algorithms can be defined as the number of nodes that they travel. Therefore, the complexity of the brute-force algorithm is

$$\prod_{(i,j) \in I'} |\Lambda_{i,j}|$$

which is on the order of $O(2^{2(m+n)})$. The depth-first algorithm is designed to reduce this complexity, although in the worst case it may travel to all nodes of the decision tree. Let $Z$ be the set of nodes with the largest value at level $l = |I'|$. The depth-first algorithm operates on the decision tree as follows:

- **Step 1:** Compute the root node $\nu^{(0)}$. Set $Z = \emptyset$.

- **Step 2:** Find a node in level $l = 0$, which is not expanded. If there is no such node continue to the next level until there is such a node $\nu^{(l')}$. If there is no more nodes to expand, terminate the algorithm.

- **Step 3:** Expand $\nu^{(l')}$ into its leaves $\{\nu^{(l'+1)}_i\}$. Find the node $\nu^{(l'+1)}_{\max}$ with the largest cardinality among $\{\nu^{(l'+1)}_i\}$. Repeat this step for $\nu^{(l'+1)}_{\max}$ until the bottom level $l = |I'|$ is reached. The node with the largest cardinality at the bottom level, $\nu^{(|I'|)}_{\max}$, is a candidate to be an optimal single-state code. If $Z = \emptyset$, then $Z = \{\nu^{(|I'|)}_{\max}\}$; otherwise proceed as follows:
  - If $|\nu^{(|I'|)}_{\max}| > |\nu|$, for $\nu \in Z$, then remove past candidates and add $\nu^{(|I'|)}_{\max}$ to the set of candidates, i.e., $Z = \{\nu^{(|I'|)}_{\max}\}$.
  - If $|\nu^{(|I'|)}_{\max}| = |\nu|$, for $\nu \in Z$, then add $\nu^{(|I'|)}_{\max}$ to the set candidates, i.e., $Z \leftarrow Z \cup \{\nu^{(|I'|)}_{\max}\}$.
  - If $|\nu^{(|I'|)}_{\max}| < |\nu|$, for $\nu \in Z$, then remove $\nu^{(|I'|)}_{\max}$.

- **Step 4:** Remove all nodes $\nu^{(l)}$, $0 < l < |I'|$, from the decision tree satisfying $|\nu^{(l)}| < |\nu|$ for $\nu \in Z$, since their leaves cannot give an optimal single-state code. Repeat steps 2, 3, and 4.
The depth-first algorithm can proceed by using only the node values. Therefore it is not necessary to compute and store the set of codewords in a node. The cardinality of a node can be computed by using adjacency matrices [8]. Let \( r \) be the maximum number of rows of a pattern \( p \in \mathcal{F}(S) \). The decimal value of an \( m \times n \) binary array \( M \) is defined as
\[
\sum_{i=1}^{m} \sum_{j=1}^{n} M_{i,j} 2^{in+j}.
\]
Let \( U \) and \( V \) be two ordered sets of \((r - 1) \times n\) binary arrays in ascending order of their decimal values. The adjacency matrix from \( U \) to \( V \) is denoted as \( A(U \rightarrow V) \). The \((i, j)\)th entry of \( A(U \rightarrow V) \) is defined as 1 if the \( i \)th constrained array of \( U \), \( U^{(i)} \), and the \( j \)th constrained array of \( V \), \( V^{(j)} \), satisfy the following properties:
- \( U^{(i)}_{k,*} = V^{(j)}_{k-1,*} \) for \( 0 \leq k \leq r \).
- The matrix \[
\begin{bmatrix}
U^{(i)} \\
V^{(j)}
\end{bmatrix}
\]
does not contain any forbidden pieces from the constraint.
Otherwise \( [A(U \rightarrow V)]_{i,j} = 0 \).

Suppose that a node \( \nu^{(l)} \) is obtained after \( l \) decisions are made. Let \( f \) denote the set of forbidden pieces selected from \( \nu^{(0)} \) to \( \nu^{(l)} \). In order to compute the cardinality of \( \nu^{(l)} \), the ordered sets of size \((r - 1) \times n\) are defined corresponding to each subarray of the codeword window with the same size. Let \( W_i \), \( 1 \leq i \leq m - r + 1 \), be a ordered set of constrained arrays corresponding to the subarray whose top-left entry is \((i, 1)\) and bottom right entry is \((i + r - 1, n)\). These sets are required not to contain any forbidden piece from \( f \). The cardinality of \( \nu^{(l)} \) is given by
\[
|\nu^{(l)}| = S\left[ \prod_{i=1}^{m-r+2} A(W_i \rightarrow W_{i+1}) \right]
\]
where \( S[\cdot] = 1^T[\cdot]1 \) computes the sum of all entries over the product of the matrices in question.

### 6.3 Results

The coding rates of some optimal single-state codes are shown in Table 6.2. The parameter \( N \) is the number of distinct optimal single-state block codes. For the hard-
square model, the optimal single-state codes designed by the depth-first algorithm and the pruning algorithm are the same (see Table 5.1).

Let n.i.1, n.i.b., s.n.i.1, and s.n.i.b. be the first group of constraints, and the hard-square and the hard-hexagon models be the second group of constraints. For the first group of constraints, there is only one optimal single-state block code for each codeword size. These codes have two special properties:

- The forbidden pieces selected at each position of the codeword window have the largest area among their decision sets.
- The first candidate node at the bottom level is the optimal single-state block code.

The decision tree shown in Fig. 6.2 is an example of these properties. The optimal single-state code is found after the first depth-first search and contains the largest forbidden pieces from each decision set $\lambda_{i,j}$. Intuitively, larger forbidden pieces are contained in fewer codewords; therefore the algorithm selects those pieces. However, these properties are not true for the second group of constraints.

Figure 6.3 Complexity plots for the depth-first (DF) and the brute-force (BF) algorithms for the first group of constraints.
Figure 6.4 Complexity values for the depth-first (DF) and brute-force (BF) algorithms for the second group of constraints.

The complexity of the depth-first and brute-force algorithms are shown in Figures 6.3 and 6.4 for the first and second groups of constraints, respectively. There is a significant gap between the complexities of these algorithms for the first group of constraints. Therefore, the bias between forbidden pieces facilitates finding the optimal single-state codes. The complexities of both algorithms increase rapidly with the number of forbidden patterns.

However, for the second group of constraints, the depth-first algorithm has comparable complexity with the brute-force algorithm. Furthermore, when the codeword size is $3 \times 3$, the depth-first algorithm has a worse performance for the hard-square constraint. One reason for this is that the hard-square or hard-hexagon models produce equal-size forbidden pieces at codeword boundaries. It is much harder in this case to decide which forbidden piece should be eliminated so that the codebook size remains the largest. In principle this phenomenon may happen for an arbitrary constraint since the complexity of the depth-first algorithm is bounded from above by the number of nodes in the decision tree, which is larger than the complexity of the brute-force algorithm.
6.4 The Performance of a Coded System

In this section, the performance of a rate $13/16$ single-state block code for the n.i.b. constraint is evaluated for the channel with impulse response

$$h = \begin{bmatrix} \alpha \\ \alpha \\ 1 \\ \alpha \end{bmatrix}$$

where $\alpha$ varies from 0 to 0.5 with the step-size of 0.05. For the uncoded case, the eye of the channel is closed for $\alpha \geq 0.25$. However, the n.i.b. constraint can prevent the worst bipolar input patterns

$$\begin{bmatrix} -1 \\ -1 \\ +1 \\ -1 \end{bmatrix}, \begin{bmatrix} +1 \\ +1 \\ -1 \\ +1 \end{bmatrix}$$

that give zero signal energy at the channel output. Eliminating these input patterns opens the eye of the channel for $\alpha < 0.5$.

The simulation diagram is depicted in Fig. 6.5. There are four rate $\rho = 13/16$ encoders implementing single-state block codes for the n.i.b. constraint. The modulation code is designed by choosing $2^{13}$ codewords among 8950 possible codewords. The 52-bit binary inputs $u_{i,j}$ are mapped to $8 \times 8$ bipolar channel codewords $x_{i,j}$ where 0's and 1's are converted to -1 and +1, respectively. The channel output is distorted by the AWGN with zero-mean and variance $\sigma^2$.

The received array $r_{i,j}$ is detected by using a simple threshold detector, which gives
For each $4 \times 4$ detected array, the decoder searches for the matching codeword to the received array. If there is a match, then it outputs the corresponding 13-bit data sequence of the matched codeword. If there is no match, then it outputs a random 13-bit sequence.

In the uncoded case, the simulation diagram is the same as the coded system except that there are no modulation encoders and decoders. Binary input arrays are converted to bipolar arrays before they are fed into the channel.

The bit error rate performance of the uncoded and coded channels are shown in Figures 6.6 and 6.7, respectively. The rate loss due to the modulation code is about $-10 \log_{10} (13/16) \approx 0.9018$ dB. When $\alpha$ is close to 0, the performance of the coded channel is worse than the uncoded case, since there is not enough ISI so that the modulation...
code can be beneficial. When $\alpha$ is close to 0.25, the performance of the coded system is significantly better than the uncoded case. For $\alpha \geq 0.25$, the eye of the channel is opened as expected.

### 6.5 Conclusion

In this chapter, the depth-first algorithm for finding single-state block codes is proposed for 2-D constrained systems, which are represented by a set of forbidden patterns. The lack of labeled graph representations for 2-D constraints is the main difficulty for designing high-rate robust modulation codes.
6.6 Acknowledgements

A portion of this chapter will be presented in the conference: I. Demirkan, and J. K. Wolf, “The dept-first algorithm for designing 2-D single-state block codes,” in *IEEE International Conference on Communications (ICC’06)*, Jun. 2006. The dissertation author is the primary author in this paper.
Chapter 7

Combined Constraints for Perpendicular Recording Channels

7.1 Introduction

Runlength-limited (RLL) constraints are typically employed in magnetic recording systems in order to improve the write ability of the recording system and to enhance the readback performance. One class of the RLL constraints commonly used in practice are the \((d, k)\) constraints, where \(d\) and \(k\) represent the minimum and maximum number of consecutive 0’s between two adjacent 1’s, respectively. In peak-detection systems, 1 represents a signal location, whereas 0 represents void of signal. The minimum separation is imposed between adjacent 1’s to avoid intersymbol inference (ISI). However, this separation is limited to \(k\) to help gain and timing recovery loops in the readback process. The drawback of the parameter \(d\) is low capacity values, which is not desirable for high-density recording channels equalized to partial response (PR) shapes. The notion of the RLL constraints is extended to \((0, G/I)\) constraints for PR channels (see [39, 38, 37, 44]).

The channel symbols of contemporary magnetic recording systems are \(-1\) and \(+1\) that represents low and high signal amplitudes in readback signals, respectively. A commonly used mapping from constrained sequences to channel sequences is Non-Return-To-Zero (NRZ) convention where 0’s and 1’s are represented by \(-1\)’s and \(+1\)’s, respectively. Another frequently used mapping is Non-Return-To-Zero-Inverse (NRZI) convention where 1’s represent transitions from \(-1\) to \(+1\) or \(+1\) to \(-1\), and 0’s represent no transition. Under the NRZI convention, the RLL constraints determine the minimum and maximum runs of
no transitions in channel sequences.

The maximum transition run (MTR) constraints have a wide variety of applications in magnetic recording systems. The MTR \((j)\) constraint eliminates consecutive runs of transitions larger than \(j\) so that certain common error events can be avoided. The MTR \((j)\) constraints have much higher capacities than the RLL \((d,k)\) constraints when \(d > 0\). For instance, the capacity of the RLL \((1,\infty)\) constraint is 0.6942, whereas the capacity of the MTR \((2)\) constraint is 0.8791 [45].

The TMTR constraints are obtained by relaxing the \(j\) parameter of the MTR constraint to allow \(j+1\) consecutive transitions starting at predefined positions. The TMTR constraints allowing \(j\) and \(j+1\) consecutive transitions have larger capacities than the MTR \((j)\) constraints. For example, consider a TMTR constraint that allows 3 and 4 consecutive transitions to occur at even and odd positions, respectively. It has much higher code efficiency compared with the MTR \((3)\) constraint since their capacities are .9613 and .9468, respectively. The detailed description of MTR and TMTR constraints are given in Section 7.3.

Maximum likelihood sequence detectors are typically employed in contemporary magnetic recording systems. The error rate performance of the system is determined by the error events with minimum and near minimum squared-Euclidean distances when the signal-to-noise (SNR) ratio is high. For a pair of input sequences \(x(D)\) and \(x'(D)\), the input error sequence is defined as \(\varepsilon_x(D) = x(D) - x'(D)\) with the corresponding output error sequence \(\varepsilon_y(D) = h(D)\varepsilon_x(D)\), where \(h(D)\) is the \(D\)-transform of the channel impulse response. The input error sequences are called error events and denoted with symbols \((+,-,0)\). For example, an error event for the binary input \((+1,0,+1,-1)\) is denoted as \((+0+)\). For an channel output error sequence \(\varepsilon_y\), the squared-Euclidean distance is defined as

\[
d^2(E) = \|\varepsilon_y(D)\|^2 = \sum_{i=1}^{\infty} \varepsilon_{y,i}^2,
\]

where \(\varepsilon_y = \{\varepsilon_{y,i}\}_{i=1}^{\infty}\). For example, if the channel output sequence of \(+1,-1,+1\) is mistakenly detected as \(-1,+1,-1\), then \(\varepsilon_y = \{+2,-2,+2\}\) and the squared-Euclidean distance is 12.

For perpendicular recording channels, some typical error events with small distances are \((+), (+-), (+--), (+++), (00+), (000+)\), and their inverse correspondences. The error rate performance of perpendicular recording channels can be improved by combining the properties of MTR constraints and parity checking [46]. For example,
the MTR (3) constraint with 2-bit parity checking eliminates (+−−) and corrects (+), (+−), (+−+), and (+00+) error events. The TMTR constraints allowing 3 or 4 consecutive transitions can be utilized instead of the MTR (3) constraint if relaxation positions are separated by at least 1 bit interval, which guarantees that the error event (+−−−) is eliminated.

The property of an RDS constraint is desirable for achieving high-density recording for recording channels exhibiting low frequency components, such as perpendicular recording channels. The RDS constraints exhibit a null at DC frequency and suppress the low frequency content of the input signal in the NRZ domain. The running digital sum $z_n$ of an NRZ sequence $x = \{x_i\}_{i=-\infty}^{\infty}$, $x_i \in \{-1, +1\}$ is defined as

$$z_n = \sum_{i=-\infty}^{n} x_i.$$  

When $z_n$ is lower and upper bounded by $N_1$ and $N_2$ along the entire sequence, respectively, the digital-sum-variation (DSV) of $x$ is defined as $N = N_2 - N_1 + 1$. The RDS (N) constraint allows all bipolar sequences having DSV at most $N$. The details of RDS constraints are discussed in Section 7.3.

The properties of both RDS and MTR (or TMTR) constraints are desirable for recording channels exhibiting low frequency contents (Section 7.2). In order to take advantages of these constraints, their labeled graph representations are combined to obtain new classes of constraints: RDS-MTR and RDS-TMTR constraints (Section 7.4). The capacity of a combined constraint is observed to be slightly lower than that of its constituent constraints. The labeled graph representations of the combined constraints serve as a foundation for code selection and construction. The spectra of RDS-MTR and RDS-TMTR constraints show the suppression of low frequency contents compared to the conventional MTR and TMTR constraints (Section 7.5). In Section 7.6, a method for designing block codes is proposed to implement the properties of RDS, MTR (or TMTR), RLL (0,k), and twin constraints. The cardinalities of block codes are maximized over various situations of codeword boundaries. The code design methodology is demonstrated via a rate 100/108 RDS-TMTR block code based on a rate 20/21 RDS-TMTR mother code.
7.2 Channel Targets

7.2.1 Recording Channels Exhibiting Low Frequency Contents

Perpendicular recording channels exhibit low frequency contents unlike longitudinal recording channels (see Fig. 7.1). The channel detector targets may be categorized into three groups: DC-free, DC-full, and DC-attenuated targets. The DC-full targets are the ones that exhibit full DC-response or full low frequency energy matching recording systems, while DC-attenuated targets exhibit some low frequency energy that lie in between DC-full and DC-free targets. If noise at the input to the channel is white, then we cannot do anything better to reduce the noise power other than doing as little as possible on the equalization. As a result, the target will match the channel spectral response (DC-full) with little equalization needed. On the other hand, if the noise is not white, then we can further reduce the noise power by whitening the noise, which will not necessarily result in DC-full target. For example, the media noise present in modern recording systems is the low frequency noise whose spectral shape is similar to the channel spectral response. With pronounced noise at low frequencies, DC-attenuated targets may become the better choice in balancing signal energy and noise power at low frequencies. The DC-attenuated targets approach DC-free targets when increasingly suppressing DC and low frequency contents. DC-free targets sharply suppress low frequency noise, but also totally eliminate signal energy at DC and heavily suppress signal energy at low frequencies, resulting in losses in SNR. For uncoded recording systems exhibiting low frequency content, DC-attenuated targets become the favorable choice for improving SNR.

7.2.2 DC-Compensation Loops for DC-Full or DC-Attenuated Targets

Even though channel detector targets can be DC-full or DC-attenuated, in practice there are AC coupling filters in recording systems and signals entering the read-paths of recording channels are AC coupled. The impact of AC coupling is the removal of the low frequency content in readback signals, causing signals entering channel detectors to have perturbations at baselines, a phenomenon called baseline wander. Baseline wander has an adverse impact on the channel error rate performance. One way to compensate for the baseline wander is to apply DC-compensation loops into channels to restore the low frequency content of incoming signals. The effectiveness of DC-compensation is dependent on the high-pass pole frequencies in AC-coupling networks. The higher the high-pass
pole frequencies, the harder it is for DC-compensation loops to work effectively. DC-compensation loops work just like a decision-feedback equalizer system, which may be sensitive to error propagation.

7.2.3 Utilization of RDS Constraints for DC-Full or DC-Attenuated Targets

A different approach for solving baseline wander for DC-attenuated targets is to apply DC-free or RDS constraints to data sequences so that readback signals from recording channels have much attenuated low frequency components and the AC-coupling network will have no or much less impact on the readback signal entering the channel equalizer and detector. With the aid of DC-free codes, DC-compensation loops can be eliminated and channel detector targets can still be selected to include low-frequency energy to better match the recording system response. The main advantage of using RDS constraints is that DC-full or DC-attenuated targets can be applied without the need for DC-compensation loops.
7.2.4 Utilization of RDS Constraints for DC-Free Targets

DC-full or DC-attenuated targets may produce better error rate performance than DC-free targets under normal operational conditions, but they are vulnerable to any low frequency distortions or disturbances that may appear under special conditions. One example is the existence of thermal asperity (TA) disturbance that current magnetic recording systems experience. TA is a low frequency distortion, which has a catastrophic impact on channel detectors that respond to low frequencies. To eliminate the sensitivity of channel detectors to low frequency disturbances and also minimize the losses in SNR of detectors under normal conditions, RDS constraints are used in connection with DC-free targets. DC-free targets suppress low frequency noise and also signal energy at low frequencies. But since signals at low frequencies have little energy due to RDS constraint, DC-free targets may improve spectral SNR at low frequencies. The improvement of spectral SNR at low frequencies can be achieved by matching the spectrum of RDS constraints to that of DC-free targets at low frequencies.

7.2.5 High Rate RDS Constraints and Long DC-Free Targets

In practical uses in magnetic recording, code rates of RLL constraints are preferred to be high in order to achieve good overall error rate performance. The RDS constraints are also preferred to have high code rate efficiency for use in magnetic recording channels. However, the reduction on the low frequency energy by RDS codes depends on RDS code rates. For example, low rate RDS codes achieve much better reduction on the low frequency energy. For high rate RDS codes, this reduction is not so sharp, despite that all the RDS codes have a null at DC. In order to match the spectrum of high code rate codes at low frequencies, DC-free targets need to be longer than conventional DC-free targets so that more boost at low frequencies can be realized. One way to implement long DC-free targets is through post-processing.

7.3 Preliminaries

A constrained system is a restriction on sequences of symbols from an alphabet. Constrained systems are commonly represented by either sets of allowed symbols or labeled graphs. Let $S$ be a constrained system represented as a set of allowed symbols
\{s_1, s_2, \ldots, s_n\} with durations \{t_1, t_2, \ldots, t_n\}. The capacity of \(S\) is given by
\[ Cap(S) = \log z_0, \quad (7.1) \]
where \(z_0\) is the largest real root of the equation
\[ z^{-t_1} + z^{-t_2} + \cdots + z^{-t_n} = 1. \quad (7.2) \]
This result is mentioned in the Shannon’s well-known paper [11].

Alternatively, if \(S\) is represented as a labeled graph \(G\), then the capacity can be deduced by using symbolic dynamics and Perron-Frobenius theory [38]. If \(G\) has a finite local anticipation - in particular if \(G\) is deterministic, i.e., outgoing labels of each state are distinct - then
\[ Cap(S) = \log \lambda(D(G)) \quad (7.3) \]
where \(D(G)\) is the connection matrix of \(G\) and \(\lambda(D(G))\) is the largest eigenvalue of \(D(G)\). In fact these two methods are equivalent because (7.2) is the characteristic equation of \(D(G)\).

### 7.3.1 RDS Constraints

The labeled graph representation of the RDS constraint for \(N = 4\) is shown in Fig. 7.2(a). This graph is a Mealy-type finite state machine so that the generated sequences are in NRZ domain. State labels indicate the current RDS values provided that the initial RDS value is assumed to be 1 in state 1. The Moore-type equivalent of the same graph is shown in Fig. 7.2(b). The sequences generated by this graph are in NRZI domain. In order to generate an NRZ sequence using this graph, the initial polarity of the sequence needs to be defined. The number and polarity in state labels represent RDS values and the polarity of the sequence in NRZ domain after each transition, respectively. Note that the states 1+ and 4− are possible but they are unreachable by the other states of the graph.

The capacity of the RDS constraint is analytically known as
\[ C_N = \log_2 \left[ 2 \cos \frac{\pi}{N + 1} \right] \]
for \(N \geq 3\) [47, Chapter 9]. Note that \(C_N = 0\) for \(N = 1, 2\). When \(N \to \infty\), the capacity approaches 1 as expected. Some capacity values of RDS constraints are shown in the last column of Table 7.7. Since the capacity is very close to 1 for \(N > 20\), it is possible to design high-rate modulation codes in practice with reasonable suppression of DC-frequency.
7.3.2 The RLL \((0, k)\) Constraints

This class of constraints limit the number of consecutive 0’s to at most \(k\) in NRZI sequences. The graph representation of the RLL \((0, 4)\) constraint is shown in Fig. 7.3. State labels indicates the number consecutive 0’s on the tail of the sequence. The capacity can be obtained by using (7.1)

\[
C_k = \log z_0
\]

where \(z_0\) is the largest positive real root of the characteristic equation

\[
z^{k+2} - 2z^{k+1} + 1 = 0.
\]

Some capacity values of the RLL \((0, k)\) constraints are shown in the last column of Table 7.5. The capacity rapidly converges 1 as \(k \to \infty\). For practical values of \(k\), such as \(k = 20\), the rate loss due to this constraint is negligible.

7.3.3 MTR and TMTR Constraints

The graph of the MTR constraint for \(j = 4\) is shown in Fig. 7.4(a). The state labels are the number of consecutive 1’s on the tail of the generated sequence. The MTR constraints can be considered as the complements of the RLL \((0, k)\) constraints when \(j = k\). Therefore
the capacity of the MTR \((j)\) constraint is equal to that of the RLL \((0,k)\) constraint, when \(j = k\).

The graphs of TMTR constraints are not as obvious as that of MTR constraints. For example, the graph of the TMTR constraint with \(j = 2\) and \(j = 3\) for even and odd relaxation positions, respectively, is shown in 7.4(b). The circled and squared states are at even and odd time index, respectively. There can be at most three successive 1’s starting from an odd time index (the direct path between squared state 0 and circled state 3), whereas there are at most two consecutive 1’s starting from an even time index (the direct path between circled states 0 and 2).

The TMTR constraints can be generalized via imposing different sets of relaxation points to MTR constrained sequences. We define the pattern \(r\) of a TMTR constraint as a set of maximum number of consecutive transitions periodically imposed on code sequences. For instance, the TMTR constraint with \(r = (2, 3)\) implies that there can be at most 2 and 3 consecutive transitions starting from even and odd positions (or odd and even positions), respectively. In general, for a TMTR constraint with pattern \(r = (j_1, \ldots, j_\nu)\) and period \(\nu\), a constrained sequence \(x = \{x_i\}\) can have at most \(j_i\) consecutive 1’s starting from the positions \(x_k\) where \(k \equiv i \pmod{\nu}\). Under this generalization, the MTR \((j)\) constraint can be considered as a TMTR constraint with pattern \(r = (j)\). For a general TMTR constraint, the following rule has to be satisfied

\[
j_{i+1} \geq j_i - 1 \quad \text{for} \quad 1 < i \leq \nu,
\]

otherwise the TMTR constraint is violated.

The capacity of a MTR or TMTR constraint can be obtained by using (7.3). Let \(C_r\) denote the capacity of the TMTR constraint with pattern \(r\). The capacity values for the
MTR \((j)\) and TMTR \((j, j + 1)\) constraints are listed on the last row of Tables 7.8 and 7.9, respectively. The capacity values for TMTR constraints with pattern \((j, j + 1)\) are larger than that for MTR \((j)\) constraints, i.e., \(C_{(j)} < C_{(j,j+1)}\). In general, the following capacity relation is true for any TMTR constraint with pattern \(r\), which is an \(n\)-tuple of \(j\) and \(j + 1\),

\[
C_{(j)} \leq C_{r} \leq C_{(j+1)}.
\]

For practical values of \(j\), such as \(j = 2\) or \(3\), the gap between \(C_{(j)}\) and \(C_{(j,j+1)}\) is large so that the coding rates in this gap can be achieved by efficient TMTR code designs.

### 7.4 Combined Constraints

In designing practical codes for the magnetic recording channel, it is common to implement several constraints into the same code. Constraints having properties of more than one constraint are called as *combined constraints*. The RLL \((d,k)\) constraints and MTR \((j,k)\) constraints are well known examples of combined constraints in practice [45]. In this section, the graphs of RDS, MTR (or TMTR) and RLL \((0,k)\) constraints will be combined to derive the graphs of several classes of combined constraints via using the fiber product of the labeled graphs [38].

**Definition 7.1.** Let \(G\) and \(H\) be two labeled graphs. Let \(V_G\) and \(V_H\) denote the set of states in \(G\) and \(H\), respectively. The fiber product of \(G\) and \(H\) is defined as the labeled graph \(G \ast H\), whose states are defined as

\[
V_{G \ast H} = V_G \times V_H = \{(u, u')|u \in V_G, u' \in V_H\}.
\]

The edge \((u, u') \xrightarrow{e} (v, v')\) of \(G \ast H\) exists if and only if \(u \xrightarrow{e} v\) exists in \(G\) and \(u' \xrightarrow{e} v'\) exists in \(H\). In this way, the constraints systems represented by \(G\) and \(H\) are combined to give new constrained system that is represented by \(G \ast H\). The capacity of the combined constraints represented by labeled graphs can be computed by using \((7.3)\).

The MTR and TMTR constraints are commonly combined with the RLL \((0,k)\) constraints in practice in order to improve timing recovery loops in the readback process. We refer to these classes of combined constraints as the MTR \((j,k)\) and TMTR \((r,k)\) constraints. The capacity values of MTR \((j,k)\) and TMTR \((r,k)\) constraints are shown in Tables 7.5 and 7.6, respectively. The loss of capacity due to the RLL\((0,k)\) constraint is insignificant for large values of \(k\).
The RDS and RLL \((0, k)\) constraints can be combined to obtain a new class of constraints, namely the RDS \((N, k)\) constraints. Some capacity values for this class are tabulated in Table 7.7. Let \(C_{N,k}\) denote the capacity of the RDS \((N, k)\) constraint. When \(k > N - 2\), \(C_{N,k} = C_{N,N-2}\) since the graph of the RDS \((N)\) constraint can produce at most \(N - 2\) consecutive 0’s. For large values of \(k \geq 20\), the capacity loss due to the RLL \((0, k)\) constraints is still insignificant.

Similarly, the RDS and MTR (or TMTR) constraints can be combined to give the RDS-MTR \((N, j)\) (or RDS-TMTR \((N, r)\)) constraints. The capacity values for these classes are shown in Tables 7.8 and 7.9, respectively. For large values of \(N > 40\), the capacity of a RDS-MTR (or RDS-TMTR) constraint is very close to the capacity of the constituent MTR (or TMTR) constraint. Three constraints can be combined to give RDS-MTR \((N, j, k)\) and RDS-TMTR \((N, r, k)\) constraints. The capacities of RDS-MTR \((N, j, k)\) constraints are shown in Tables 7.10 and 7.11 for \(j = 3, 4\) and several values of \(N\) and \(k\), respectively. These capacity values indicate the achievable coding rates, which are mainly affected by \(j\) and \(N\) parameters when \(k\) is large.

### 7.5 Spectral Properties

The characteristics of the RDS-MTR and RDS-TMTR constraints are better understood in the spectral domain. A method for computing the power spectral density for constrained sequences is discussed by Immink based on the labeled graph representations of the constraints [47, Chapter 3]. In this section, we provide a brief overview of this analysis.

Let \(S\) be a constrained system represented by a graph \(G\). We assume that \(G\) is a Mealy-type graph, meaning that symbols are emitted when the states are visited. Spectral properties of magnetic recording channels are commonly discussed in terms of channel sequences in NRZ domain. However, most of the constraints discussed in Sections 7.3 and 7.4 are represented by Moore-type graphs, which generate NRZI sequences. A Moore-type graph can be converted to the equivalent Mealy-type graph as follows: Let \(G'\) be a Moore-type labeled graph with the set of states \(\Sigma'\). Let \(G\) be the equivalent Mealy-type graph of \(G'\) with the set of states \(\Sigma\). A state \(\sigma \in \Sigma'\) in \(G'\) is split into two states \([\sigma-], [\sigma+] \in \Sigma\) in \(G\) with the polarity information of the NRZ sequence. An edge \(\sigma \rightarrow \rho\) in \(G'\) is represented by a pair of edges in \(G\): \(([\sigma-] \rightarrow [\rho-], [\sigma+] \rightarrow [\rho+])\) and \(([\sigma+] \rightarrow [\rho-], [\sigma+] \rightarrow [\rho-])\) depending on whether the label of \(e\) is 0 or 1, respectively. An example of this conversion
is shown in Fig. 7.2 for the RDS (4) constraint.

Let $\Sigma = \{\sigma_1, \ldots, \sigma_N\}$ be the set of states in $G$. Let $\zeta = \{\zeta(\sigma_1), \ldots, \zeta(\sigma_N)\}$ be the set of symbols from a bipolar alphabet $\{-1, +1\}$ such that symbol $\zeta(\sigma_i)$ is emitted when state $\sigma_i$ is visited. Reading state labels of $G$ may be considered as a Markov chain $Z = \{\ldots, Z_{-1}, Z_0, Z_1, \ldots\}$ such that the random variable $Z_t$ takes values on the state alphabet $\Sigma$. The Markovity property implies that $Z_t$ only depends on the previous sample $Z_{t-1}$; not on $Z_{t-i}$ for $i > 1$. Therefore, the Markov chain $Z$ can be represented as a graph, in this case that graph is $G$. Any Markov chain can be described by a transition probability matrix $Q$ with entries

$$q_{i,j} = Pr(Z_t = \sigma_j | Z_t = \sigma_i), \quad 1 \leq i, j \leq N.$$ 

The connection matrix $D$ of $G$ is defined such that its $(i,j)$th entry is the number of edges going from state $\sigma_i$ to state $\sigma_j$. The state transition probabilities that maximize the entropy of the sequences generated by $G$ are

$$q_{i,j} = \frac{1}{\lambda_{\text{max}}} d_{i,j} \frac{p_j}{p_i},$$

where $\lambda_{\text{max}}$ is the largest real eigenvalue of $D$ and $p = (p_1, \ldots, p_N)^T$ is the eigenvector associated with $\lambda_{\text{max}}$. The steady state probability vector $\pi$ is found by $\pi Q = \pi$ provided that $\sum_{i=1}^N \pi_i = 1$. The transition probability matrix $Q$ has rank $N - 1$, therefore there is a unique solution to $\pi$.

The correlation function of the output process $X = \{X_i | X_i = \zeta(Z_i)\}$ is given by

$$R_X(k) = E[X_t X_{t+k}] = \zeta^T \Pi Q^k \zeta,$$

where $\Pi = \text{diag}\{\pi_1, \ldots, \pi_N\}$. Hence, the power spectral density of $X$ is given by

$$H_X(w) = \sum_{k=-\infty}^{\infty} R_X(k) e^{-jkw}.$$

In practice, $R_X(k)$ is truncated into a window of $[-K, K]$, expecting that $R_X(k)$ decays when $k$ is large.

Let $H(w)$ be the power spectral density of a constraint exhibiting the DC-free property. The cut-off frequency of $H(w)$ is defined as the smallest frequency $w_0 > 0$ such that $H(w_0) = 1/2$. The low-frequency suppression of a DC-free code can be assessed by the cut-off frequency of the underlying constraint.
Table 7.1 Cut-off frequencies of the RDS and RDS-TMTR constraints for $r = (3, 4)$ and various values of $N$. Values are shown as percentages of $2\pi$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>RDS ($N$)</th>
<th>RDS-TMTR ($N, (3, 4)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>16.66</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6.879</td>
<td>6.787</td>
</tr>
<tr>
<td>7</td>
<td>3.832</td>
<td>3.849</td>
</tr>
<tr>
<td>9</td>
<td>2.448</td>
<td>2.464</td>
</tr>
<tr>
<td>12</td>
<td>1.447</td>
<td>1.456</td>
</tr>
<tr>
<td>20</td>
<td>5.548×10^{-1}</td>
<td>5.567×10^{-1}</td>
</tr>
<tr>
<td>40</td>
<td>1.456×10^{-1}</td>
<td>1.459×10^{-1}</td>
</tr>
<tr>
<td>60</td>
<td>6.583×10^{-2}</td>
<td>6.592×10^{-2}</td>
</tr>
<tr>
<td>80</td>
<td>3.736×10^{-2}</td>
<td>3.741×10^{-2}</td>
</tr>
<tr>
<td>100</td>
<td>2.404×10^{-2}</td>
<td>2.406×10^{-2}</td>
</tr>
</tbody>
</table>

The power spectral densities of the RDS constraints are shown in Fig. 7.6. It is clear that the spectra of an RDS constraint has a null at DC frequency. As $N$ increases, the cut-off frequencies of the RDS constraints decrease so that the constraint become less effective to remove the low frequency content near DC. The spectra of TMTR $(r, k)$ constraints are shown in Fig. 7.7 for $r = (3, 4)$ and various values of $k$. For $k \geq 10$, the spectra of combined constraints become indistinguishable from their constituent TMTR constraints. Note that the spectra of TMTR $(r, k)$ constraints are flat near DC frequency. The same holds for the spectra of MTR $(j, k)$ constraints. Figure 7.8 shows the power spectral density of RDS-TMTR constraints for $r = (3, 4)$ and various values of $N$. These spectra have a null at DC frequency showing the effect of RDS constraints on TMTR constraints. The spectra of RDS-MTR $(N, j)$ constraints has similar properties for $j = 3$ and $j = 4$. As shown in Table 7.1, the cut-off frequencies of some RDS and RDS-TMTR constraints are almost the same for a fixed value of $N$, since the spectra of TMTR constraints are flat near DC frequency.

### 7.6 Coding Examples

There are several approaches in practice to design constrained codes for magnetic recording channels. One method is based on the labeled graph representations of the constrained systems, namely the state-splitting algorithm (or ACH algorithm) introduced by Adler, Coppersmith, and Hassner [48]. Constrained codes designed by this algorithm are generally state-dependent. Received codewords are decoded by using sliding-block
decoders designed with finite anticipation of future and past blocks. State-dependent codes require complex circuit implementations; hence they are usually avoided in practical design scenarios. Another design method is to search for codewords satisfying the desired constraint properties.

7.6.1 A Rate 20/21 Block Code

The capacity of the TMTR constraint with \( r = (3, 4) \) is about 0.9613. Therefore a rate 20/21 < 0.9613 block code satisfying this constraint may exist. There are four constraints to be embedded in this code: TMTR, RDS, RLL \((0, k)\) and twin constraints. Twin constraints will be discussed in Subsection 7.6.1.

The Implementation of the TMTR Constraint

The TMTR \((3, 4)\) constraint has period 2. When the codeword length is odd, two codebooks are required to design a block code, which increases the complexity of the encoder. Instead, this constraint can be modified to obtain a new TMTR constraint with period 21. The capacity of the new constraint should be close to that of TMTR \((3, 4)\) constraint. The following condition is imposed on the pattern of the new constraint in order to eliminate the error event \((+-++)\): 4’s should be separated by 3’s. Let \(n_3\) and \(n_4\) denote the number of 3’s and 4’s in the pattern of the new constraint, respectively. Since, a pattern starting and ending with 4 is not possible, \(n_4 \leq 10\); otherwise two consecutive 4’s can appear at the codeword boundary. We search all patterns with \(n_4 = 10\) or \(n_4 = 9\) to obtain the best patterns leading to the largest codebooks.

For a given codebook \(C\) satisfying a TMTR constraint with pattern \(r\), some codewords cannot be followed by some other codewords in the encoding process. The goal is to find the largest subset \(C'\) of \(C\), such that any two codewords of \(C'\) can follow each other, i.e., \(C'\) is a single-state block code. For TMTR constraints, this is accomplished by the optimization over possible suffix-prefix combinations at codeword boundaries.

Example 1: Suppose a code with length 5 satisfying the TMTR constraint with \(r = (4, 3, 4, 3, 3)\). Two codewords satisfying this constraint, \([10111]\) and \([11110]\), cannot follow each other; otherwise there will be seven consecutive 1’s. Therefore, the boundary violations are determined by the number of consecutive 1’s in the prefixes or suffixes of the codewords.

The goal is to find the sets of suffixes and prefixes which lead to the largest codebooks.
Table 7.2 Prefixes and suffixes at codeword boundaries.

<table>
<thead>
<tr>
<th>current codeword</th>
<th>next codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>s0</td>
</tr>
<tr>
<td>s1</td>
<td>0</td>
</tr>
<tr>
<td>s2</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>0</td>
</tr>
</tbody>
</table>

Let $s_i$ represent the suffix of a codeword starting with zero and followed by $i$ 1’s. Similarly, $p_i$ represents the prefix of a codeword starting with $i$ 1’s followed by a zero. The possible suffixes and prefixes for Example 7.6.1 are tabulated in Table 7.2. The set of suffixes $S$ and prefixes $P$ should be selected in a way that every suffix $s \in S$ may be followed by any prefix $p \in P$ at the codeword boundary. Let $P_i$ be the set of prefixes which $s_i$ can follow. For the constraint mentioned in Example 7.6.1, the set of prefixes that can follow $s_1 = (0, 1)$ is $P_1 = \{(0), (1, 0), (1, 1, 0)\}$. In general, $P_{i+1} \subseteq P_i$, since there is no prefix $p \in P_i$ which $s_{i+1}$ can follow but $s_i$ cannot. Therefore, the sets of suffixes and prefixes are in the form $S = \{s_0, \ldots s_u\}$ and $P = \{p_0, \ldots p_w\}$. The possible $(u, w)$ pairs for Example 7.6.1 are (0, 4), (1, 2), (2, 1) and (3, 1). Therefore, the complexity of the prefix-suffix optimization is just 4, which is independent of the codeword length. In general, the complexity this optimization is the number of possible suffixes, which is bounded by the largest number in the pattern of the TMTR constraint.

Imposing the suffix and prefix sets on codeword boundaries creates a new TMTR constraint. Consider the suffix $S = \{s_0, s_1\}$ and prefix sets $P = \{p_0, p_1, p_2\}$ for Example 7.6.1. Since there is no codeword ending with [1, 1], $j_1 \leq 1$. Due to similar reasons, $j_3 \leq 2$ and $j_2 \leq 3$. Since the longest run of 1’s in prefixes is 2, $j_1 \leq 2$. Hence, the set of codewords satisfying the TMTR constraint and boundary conditions conform to a new TMTR constraint with pattern $r' = (2, 3, 2, 1, 3)$. This constraint is called a reduced constraint of the original TMTR constraint.

In general, for a TMTR constraint with pattern $r = (j_1, \ldots, j_\nu)$, and a pair of suffix and prefix sets, $S = \{s_0, \ldots s_u\}$ and $P = \{p_0, \ldots p_w\}$, the reduced TMTR constraint with the pattern $r' = \{j'_1, \ldots, j'_\nu\}$ is given by

$$j'_i = \begin{cases} 
\min\{j_i, w, \nu - 1\}, & \text{if } i = 1 \\
\min\{j_i, \nu - i\}, & \text{if } 1 < i \leq \nu - u \\
\min\{j_i, \nu - i + w + 1\}, & \text{if } \nu - u < i \leq \nu
\end{cases}$$
where \( \nu \) is both the period of the constraint and the codeword length.

The number of patterns of length 21 having \( j_i = 3 \) or \( j_i = 4 \) under the rule (7.4) is given by

\[
\frac{(20 - n_4)!}{(n_4 - 1)! (21 - 2n_4)!} + \frac{(21 - n_4)!}{n_4! (21 - 2n_4)!}.
\]

For \( n_4 = 10 \) and \( n_4 = 9 \), there are 21 and 385 different patterns, respectively. The prefix-suffix combination associated with each pattern is optimized to maximize the codebook size. The following pattern yields the largest codebook:

\[
r = (3, 4, 3, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3, 4, 3, 4, 3, 3, 3, 4, 3)
\]

which has surprisingly \( n_4 = 9 \). The set of optimal suffixes and prefixes has \( u = 2 \) and \( w = 2 \). The reduced TMTR constraint has the following pattern

\[
r' = (2, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 2, 4, 3).
\]

The number of codewords satisfying this constraint is 1,072,089. There are 23,513 excessive codewords, which are helpful to implement the other three constraints.

**The Implementation of the RDS Constraint**

The TMTR constraints are localized in the sense that only certain neighbor bits are related to each other. However, the RDS constraints are not localized so that the RDS value of the whole sector has to be limited to some interval. Except for some low rate codes, such as the zero disparity code [47, Chapter 10], it is not possible to design a high rate RDS single-state block code that guarantees DC-free sequences. Hence, our focus is to design block codes whose encoders store the RDS information. Even though RDS encoded blocks may depend on each other, they can be decoded one at a time.

Let \( z \) be the running digital sum sequence of the data sequence for whole sector. This sequence is called the *global* RDS sequence. In this design, the changes of the global RDS sequence will be monitored from codeword to codeword. The codewords in the NRZI domain are converted to the NRZ domain by assuming the previous NRZ level is \(-1\). For example, the codeword \( c = [10000] \) is transformed to \( d = -1[+1 - 1 - 1 - 1 - 1] \), which changes the global RDS sequence by \( \Delta z = -3 \). In the case of the previous NRZ level being \(+1\), every digit in \( d \) is flipped to change \( z \) by \( \Delta z = 3 \). Therefore every codeword with non-zero \( \Delta z \) can change the global RDS value in both ways according to the previous NRZ level that cannot be controlled. We consider keeping \( |\Delta z| \) values as small as possible
Table 7.3 Histogram of codewords according to their $\Delta z$ values.

<table>
<thead>
<tr>
<th>$\Delta z$</th>
<th>$N$</th>
<th>$\Delta z$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>171,813</td>
<td>1</td>
<td>169,323</td>
</tr>
<tr>
<td>-3</td>
<td>148,387</td>
<td>3</td>
<td>142,113</td>
</tr>
<tr>
<td>-5</td>
<td>108,634</td>
<td>5</td>
<td>101,285</td>
</tr>
<tr>
<td>-7</td>
<td>66,869</td>
<td>7</td>
<td>60,887</td>
</tr>
<tr>
<td>-9</td>
<td>34,159</td>
<td>9</td>
<td>30,523</td>
</tr>
<tr>
<td>-11</td>
<td>14,196</td>
<td>11</td>
<td>12,536</td>
</tr>
<tr>
<td>-13</td>
<td>4,658</td>
<td>13</td>
<td>4,104</td>
</tr>
<tr>
<td>-15</td>
<td>1,152</td>
<td>15</td>
<td>1,026</td>
</tr>
<tr>
<td>-17</td>
<td>199</td>
<td>17</td>
<td>182</td>
</tr>
<tr>
<td>-19</td>
<td>21</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>-21</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
</tbody>
</table>

via removing particular codewords with larger $|\Delta z|$ values. Another way to minimize the global RDS value is to design two codebooks with opposite $\Delta z$ ranges. We refer these methods as RDS-I and RDS-II constructions, respectively.

**RDS-I Construction**

The codewords with $|\Delta z| > L$ can be discarded to obtain a code which has a DSV at most $2L + 1$. The histogram of $\Delta z$ values is tabulated Table 7.3 for the TMTR code designed in Section 7.6.1. Since there are 23,513 excessive codewords, the codewords having $|\Delta z| > 11$ can be discarded. In this way, the number of excessive codewords has dropped to 12149.

**RDS-II Construction**

A construction for a better performing code can be obtained using two different codebooks. For the histogram of codewords shown in Table 7.3, we can design two codebooks $C_1$ and $C_2$ having all codewords with $-L' \leq \Delta z \leq L'$ for $L' > 0$, respectively. The minimum value of $L'$ is 9 provided that $|C_1| = 1,051,862 > 2^{20}$ and $|C_2| = 1,054,220 > 2^{20}$.

The encoding of codewords designed with the RDS-I and RDS-II constructions are different. In the RDS-I construction, the codewords can be simply encoded and decoded by using a look-up table. In the RDS-II construction, two codebooks with opposite $\Delta z$ ranges promise a better overall RDS performance. Suppose that we want to encode an input sequence $u$ to a codeword in both $C_1$ or $C_2$. If $z < 0$ and the previous NRZ level is
\( x_{-1} = -1 \), the encoder selects a codeword from \( C_1 \) to make \( z \) be as close to 0 as possible. However, \( \Delta z \) may not be positive as desired but in this way \( z \) cannot smaller than \( z - 9 \). Therefore, the encoder selects a codeword from \( C_1 \) if \( \text{sgn}(z)x_{-1} \geq 0 \); otherwise it selects a codeword from \( C_2 \). It is assumed that \( \text{sgn}(0) = 0 \).

The encoding process in this construction depends on \( z \); therefore it generates state-dependent codewords. However, this sequence can be designed to be block decodable. Let \( C' \) be the set of codewords both \( C_1 \) and \( C_2 \) have, i.e., \( C' \) has all the codewords of the TMTR code with \(-9 \leq \Delta z \leq 9 \). The same input sequences can be mapped to \( C' \) regardless of whether \( C_1 \) or \( C_2 \) is selected by the encoder. The remaining input sequences can be mapped to \( C_1 \setminus C' \) or \( C_2 \setminus C' \) depending \( z \) and \( x_{-1} \). Let \( B \) be the set of all 20-bit input sequences and \( B' \) be the set of input words that will be mapped to \( C' \). The encoder and decoder operations can be summarized as

\[
\mathcal{E}(u) = \begin{cases} 
    c_1 \in C', & \text{if } u \in B' \\
    c_2 \in C_1 \setminus C', & \text{if } u \in B \setminus B' \text{ and } \text{sgn}(z)x_{-1} \geq 0 \\
    c_3 \in C_2 \setminus C', & \text{if } u \in B \setminus B' \text{ and } \text{sgn}(z)x_{-1} = -1
\end{cases}
\]

\[
\mathcal{D}(c) = \begin{cases} 
    u_1 \in B', & \text{if } c \in C' \\
    u_2 \in B \setminus B', & \text{if } c \in C_1 \setminus C' \text{ or } c \in C_2 \setminus C'
\end{cases}
\]

Contrary to the encoder, the decoder does not depend on \( z \) or \( x_{-1} \) since \( C_1 \setminus C' \) and \( C_2 \setminus C' \) are disjoint. Therefore there is no error-propagation in the decoding. The RDS-II construction provides a better RDS performance than the RDS-I construction since for a codeword block \( |\Delta z| \leq 9 \) and \( |\Delta z| \leq 11 \), respectively.

**The Implementation of the RLL \((0, k)\) Constraint**

Similar to the TMTR constraints, the RLL \((0, k)\) constraints are localized so that it is enough to satisfy the constraint along the boundaries and the inside of the codewords. Let \( a \) and \( b \) be the number of consecutive 0’s as a prefix and suffix for a codeword, respectively. Let \( c \) be the number of consecutive 0’s inside of a codeword. If a codeword contains only one 1, then \( c \) is defined to be 0. If the codeword is all-zero, then \( a = b = c = 21 \). In order to satisfy the RLL \((0, k)\) constraint at codeword boundaries, the following inequalities must hold for all codewords: \( a + b \leq k \) and \( c \leq k \). The codewords with \( c > k \) are first eliminated from the codebook designed in the previous section. For the remaining codewords, the best pairs of \( a \) and \( b \) satisfying \( a + b = k \) are searched to maximize the cardinality of the
Table 7.4 Optimized parameters for the RDS, RLL \((0,k)\), and twin constraints.

<table>
<thead>
<tr>
<th>(k)</th>
<th>(t)</th>
<th>(\Delta z) (RDS-I)</th>
<th>(\Delta z) (RDS-II)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>-15</td>
<td>15</td>
<td>-13,13</td>
</tr>
<tr>
<td>13</td>
<td>-13</td>
<td>13</td>
<td>-11,11</td>
</tr>
<tr>
<td>14-16</td>
<td>-11</td>
<td>11</td>
<td>-11,11</td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>-11,11</td>
<td>-9,9</td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>-11,11</td>
<td>-9,9</td>
</tr>
<tr>
<td>19-25</td>
<td>-11</td>
<td>11</td>
<td>-9,9</td>
</tr>
<tr>
<td>26-41</td>
<td>15</td>
<td>-11,11</td>
<td>-9,9</td>
</tr>
<tr>
<td>42,∞</td>
<td>15</td>
<td>-11,11</td>
<td>-9,9</td>
</tr>
</tbody>
</table>

codebook. The minimum values of \(k\) for block codes designed in the previous section are shown in Table 7.4 such that the coding rate is still 20/21.

The Implementation of the Twin Constraint

The twin constraint prevents long error sequences at Nyquist frequencies from occurring by removing supporting sequences in the NRZI format. The error sequence of \((+0+0\ldots)\) is one of the examples. These error sequences result from the null at the Nyquist frequency of channel target responses. One way to eliminate these error sequences is to limit the number of consecutive pairs of 0's or 1's (twins) in the NRZI domain [49]. We have to eliminate all possible \(t+1\) pairs of zeros or ones to limit the maximum length of the error event to \(2t\). Similar to the RLL \((0,k)\) constraint, twin constraints are also localized. Let \(a\) and \(b\) be the number of consecutive pairs of zeros and ones as a prefix and suffix of a codeword, respectively. Likewise, let \(c\) and \(d\) be the number of consecutive pairs of zeros and ones starting at the second position and ending at the second to last position of a codeword, respectively. Note that we are not interested in run of twins inside the codeword since \(t\) is expected to be at least 11. In order to prevent different situations in which an error event can propagate to the next codeword, the following conditions have to be ensured for all codewords: \(a+b \leq t\) and \(c+d \leq t-1\). Based on the designed code in the previous section, we optimize these parameters for the least value of \(t\) as long as the other parameters of the code stay the same. Table 7.4 shows the minimum values of \(t\) for the designed codes in the previous section. There is a trade-off between the RLL \((0,k)\) and twin constraints for a fixed codebook of TMTR code with the same RDS performance. Typical values of \(t\) range from 15 to 19.

We designed two 20/21 block codes based on the TMTR code discussed in Section
7.6.1. The first one implements the RDS-I construction with $-11 \leq \Delta z \leq 11$, whereas the second one implements the RDS-II construction with $-9 \leq \Delta z \leq 9$. As a design choice, both codes have parameters $t = 16$ and $k = 19$. We refer to these codes as code $A$ and code $B$, respectively.

### 7.6.2 A Rate 100/108 Block Code

Five blocks of code $A$ or $B$ are used in parallel to obtain the rate 100/105 block code as shown in Fig. 7.5. The global RDS value $z$ can vary from the beginning to the end of codewords by at most $5 \times (\pm 11) = \pm 55$ for code $A$ and $5 \times (\pm 9) = \pm 45$ for code $B$. Also, $z$ can vary by at most $\pm 11$ inside each 21-bit blocks. In general $z$ can vary from $\Delta z$ inside the codeword by at most $\pm \lfloor n/2 \rfloor$, where the $n$ is the length of the codeword. Therefore, the true range of variation is $\pm 66$ for code $A$ and $\pm 56$ for code $B$.

The TMTR code designed in Section 7.6.1 eliminates all error events with alternating structure with lengths of more than 3 or 4 depending on the bit position in the codeword. The error events with length less than or equal to 3 or 4 could still happen in data detection. In order to improve the error rate performance, those short error events, $(+),(+–)$, $(+–)$, and $(+00+)$, need to be detected and hopefully corrected. In order to eliminate these error events, two bit parity-check bits are embedded into the 105-bit codewords in such a way that the TMTR and RDS constraints are still valid after inclusion of the parity check bits. A method for eliminating short error events using parity check codes has been proposed in [50, 46]. However, the insertion of parity bits may violate the TMTR constraint at the codeword boundary. This problem is solved by inserting the parity bits at carefully chosen positions, which will not be addressed here.

The global RDS value has to be limited to a certain range along the entire sequence so as to obtain the DC-free property. This can be done by means of inserting a redundant
polarity bit before each 107-bit codeword. Inserting a 1 before any NRZI sequence causes all symbols to flip in the NRZ domain, whereas the insertion of a 0 has no effect on the sequence. Therefore, the $\Delta z$ of a 107-bit codeword can be controlled by a polarity bit inserted before the sequence. The encoder sets the polarity bit as 0, if $\text{sgn}(z)\text{sgn}(\Delta z)x_{-1} = +1$; 1 otherwise. For example, if $z < 0$, $x_{-1} = -1$ and $\Delta z \geq 0$, it is desirable to add a positive value to $z$ to make it close to zero. Since $x_{-1} = -1$ and $\Delta z \geq 0$, the polarity bit is set to 0. The polarity bit guarantees that the range of $z$ for the entire sequence is the same as the range of $\Delta z$ for 105-bit blocks.

Inserting a polarity bit may fail the TMTR constraint at the codeword boundaries. However, in that case the error events will be split into two separate codewords. They can be corrected by a post-processor. The polarity bit is not decoded and it only affects the polarity of the write current waveform, not the information bits in the NRZI domain.

7.7 Conclusions

This chapter has discussed the usage of combined constraints on perpendicular recording systems. Long error events with alternating structure, such as $(+-+-)$, can be eliminated by the help of MTR or TMTR constraints. Short error events can be removed by using 2 parity check bits inserted at certain positions of the codewords. The low frequency content of the perpendicular channels is suppressed by implementing a RDS constraint whose spectra have a null at DC-frequency. The concept of combining the RDS and MTR or TMTR constraints is proposed to examine the possible coding rates. A code-design methodology is presented by the step-by-step design of a rate 100/108 block code.

7.8 Acknowledgements

The author is grateful to Yuan Xing Lee for providing much information regarding perpendicular recording channels and possible channel targets. This work was supported by Hitachi Global Storage Technologies.

author in these papers.
7.A Capacity Tables

Table 7.5 Capacity values for MTR \((j,k)\) constraints.

<table>
<thead>
<tr>
<th>(k\backslash j)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>(C_k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
<td>0.4057</td>
<td>0.5515</td>
<td>0.6174</td>
<td>0.6509</td>
<td>0.6690</td>
<td>0.6942</td>
</tr>
<tr>
<td>2</td>
<td>0.4057</td>
<td>0.6942</td>
<td>0.7947</td>
<td>0.8376</td>
<td>0.8579</td>
<td>0.8680</td>
<td>0.8791</td>
</tr>
<tr>
<td>3</td>
<td>0.5515</td>
<td>0.7947</td>
<td>0.8791</td>
<td>0.9146</td>
<td>0.9309</td>
<td>0.9388</td>
<td>0.9468</td>
</tr>
<tr>
<td>4</td>
<td>0.6174</td>
<td>0.8376</td>
<td>0.9146</td>
<td>0.9468</td>
<td>0.9614</td>
<td>0.9684</td>
<td>0.9752</td>
</tr>
<tr>
<td>5</td>
<td>0.6509</td>
<td>0.8579</td>
<td>0.9309</td>
<td>0.9614</td>
<td>0.9752</td>
<td>0.9818</td>
<td>0.9881</td>
</tr>
<tr>
<td>6</td>
<td>0.6690</td>
<td>0.8680</td>
<td>0.9388</td>
<td>0.9684</td>
<td>0.9818</td>
<td>0.9881</td>
<td>0.9942</td>
</tr>
<tr>
<td>7</td>
<td>0.6793</td>
<td>0.8732</td>
<td>0.9427</td>
<td>0.9718</td>
<td>0.9850</td>
<td>0.9912</td>
<td>0.9971</td>
</tr>
<tr>
<td>8</td>
<td>0.6853</td>
<td>0.8760</td>
<td>0.9447</td>
<td>0.9735</td>
<td>0.9865</td>
<td>0.9927</td>
<td>0.9986</td>
</tr>
<tr>
<td>10</td>
<td>0.6909</td>
<td>0.8782</td>
<td>0.9462</td>
<td>0.9748</td>
<td>0.9877</td>
<td>0.9938</td>
<td>0.9996</td>
</tr>
<tr>
<td>12</td>
<td>0.6930</td>
<td>0.8789</td>
<td>0.9466</td>
<td>0.9751</td>
<td>0.9880</td>
<td>0.9941</td>
<td>0.9999</td>
</tr>
<tr>
<td>(C_{(j)})</td>
<td>0.6942</td>
<td>0.8791</td>
<td>0.9468</td>
<td>0.9752</td>
<td>0.9881</td>
<td>0.9942</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 7.6 Capacity values for TMTR \((r,k)\) constraints for \(r = (j, j+1)\).

<table>
<thead>
<tr>
<th>(k\backslash j)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>(C_k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
<td>0.5000</td>
<td>0.5840</td>
<td>0.6358</td>
<td>0.6600</td>
<td>0.6743</td>
<td>0.6942</td>
</tr>
<tr>
<td>2</td>
<td>0.5706</td>
<td>0.7507</td>
<td>0.8170</td>
<td>0.8482</td>
<td>0.8630</td>
<td>0.8707</td>
<td>0.8791</td>
</tr>
<tr>
<td>3</td>
<td>0.6804</td>
<td>0.8423</td>
<td>0.8974</td>
<td>0.9231</td>
<td>0.9349</td>
<td>0.9408</td>
<td>0.9468</td>
</tr>
<tr>
<td>4</td>
<td>0.7381</td>
<td>0.8802</td>
<td>0.9312</td>
<td>0.9543</td>
<td>0.9649</td>
<td>0.9701</td>
<td>0.9752</td>
</tr>
<tr>
<td>5</td>
<td>0.7619</td>
<td>0.8983</td>
<td>0.9466</td>
<td>0.9685</td>
<td>0.9785</td>
<td>0.9834</td>
<td>0.9881</td>
</tr>
<tr>
<td>6</td>
<td>0.7764</td>
<td>0.9070</td>
<td>0.9540</td>
<td>0.9753</td>
<td>0.9850</td>
<td>0.9897</td>
<td>0.9942</td>
</tr>
<tr>
<td>7</td>
<td>0.7831</td>
<td>0.9115</td>
<td>0.9576</td>
<td>0.9786</td>
<td>0.9881</td>
<td>0.9927</td>
<td>0.9971</td>
</tr>
<tr>
<td>8</td>
<td>0.7874</td>
<td>0.9137</td>
<td>0.9594</td>
<td>0.9802</td>
<td>0.9896</td>
<td>0.9941</td>
<td>0.9986</td>
</tr>
<tr>
<td>10</td>
<td>0.7908</td>
<td>0.9156</td>
<td>0.9608</td>
<td>0.9814</td>
<td>0.9908</td>
<td>0.9953</td>
<td>0.9996</td>
</tr>
<tr>
<td>12</td>
<td>0.7919</td>
<td>0.9161</td>
<td>0.9612</td>
<td>0.9817</td>
<td>0.9911</td>
<td>0.9956</td>
<td>0.9999</td>
</tr>
<tr>
<td>(C_{(j,j+1)})</td>
<td>0.7925</td>
<td>0.9163</td>
<td>0.9613</td>
<td>0.9818</td>
<td>0.9912</td>
<td>0.9957</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Table 7.7 Capacity values for RDS \((N,k)\) constraints.

<table>
<thead>
<tr>
<th>(N\backslash k)</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>(C_N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.6942</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.6942</td>
</tr>
<tr>
<td>6</td>
<td>0.8032</td>
<td>0.8495</td>
<td></td>
<td></td>
<td></td>
<td>0.8495</td>
</tr>
<tr>
<td>8</td>
<td>0.8378</td>
<td>0.9048</td>
<td>0.9103</td>
<td></td>
<td></td>
<td>0.9103</td>
</tr>
<tr>
<td>10</td>
<td>0.8532</td>
<td>0.9302</td>
<td>0.9396</td>
<td>0.9403</td>
<td></td>
<td>0.9403</td>
</tr>
<tr>
<td>12</td>
<td>0.8613</td>
<td>0.9440</td>
<td>0.9558</td>
<td>0.9573</td>
<td>0.9575</td>
<td>0.9575</td>
</tr>
<tr>
<td>16</td>
<td>0.8692</td>
<td>0.9576</td>
<td>0.9723</td>
<td>0.9748</td>
<td>0.9752</td>
<td>0.9752</td>
</tr>
<tr>
<td>20</td>
<td>0.8728</td>
<td>0.9639</td>
<td>0.9800</td>
<td>0.9831</td>
<td>0.9837</td>
<td>0.9838</td>
</tr>
<tr>
<td>24</td>
<td>0.8748</td>
<td>0.9674</td>
<td>0.9843</td>
<td>0.9877</td>
<td>0.9884</td>
<td>0.9886</td>
</tr>
<tr>
<td>32</td>
<td>0.8767</td>
<td>0.9708</td>
<td>0.9886</td>
<td>0.9924</td>
<td>0.9932</td>
<td>0.9935</td>
</tr>
<tr>
<td>40</td>
<td>0.8776</td>
<td>0.9724</td>
<td>0.9906</td>
<td>0.9946</td>
<td>0.9955</td>
<td>0.9958</td>
</tr>
<tr>
<td>(C_k)</td>
<td>0.6942</td>
<td>0.8791</td>
<td>0.9752</td>
<td>0.9942</td>
<td>0.9986</td>
<td>0.9996</td>
</tr>
</tbody>
</table>

Table 7.8 Capacity values for RDS-MTR \((N,j)\) constraints.

<table>
<thead>
<tr>
<th>(N\backslash j)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>(C_N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.2878</td>
<td>0.4057</td>
<td>0.5943</td>
<td>0.6174</td>
<td>0.6623</td>
<td>0.6690</td>
<td>0.6942</td>
</tr>
<tr>
<td>6</td>
<td>0.5018</td>
<td>0.6594</td>
<td>0.7777</td>
<td>0.8052</td>
<td>0.8303</td>
<td>0.8372</td>
<td>0.8495</td>
</tr>
<tr>
<td>8</td>
<td>0.5816</td>
<td>0.7499</td>
<td>0.8468</td>
<td>0.8748</td>
<td>0.8945</td>
<td>0.9010</td>
<td>0.9103</td>
</tr>
<tr>
<td>10</td>
<td>0.6202</td>
<td>0.7936</td>
<td>0.8807</td>
<td>0.9087</td>
<td>0.9260</td>
<td>0.9324</td>
<td>0.9403</td>
</tr>
<tr>
<td>12</td>
<td>0.6419</td>
<td>0.8182</td>
<td>0.8998</td>
<td>0.9279</td>
<td>0.9439</td>
<td>0.9502</td>
<td>0.9575</td>
</tr>
<tr>
<td>16</td>
<td>0.6641</td>
<td>0.8436</td>
<td>0.9195</td>
<td>0.9477</td>
<td>0.9624</td>
<td>0.9686</td>
<td>0.9752</td>
</tr>
<tr>
<td>20</td>
<td>0.6746</td>
<td>0.8558</td>
<td>0.9290</td>
<td>0.9572</td>
<td>0.9713</td>
<td>0.9775</td>
<td>0.9838</td>
</tr>
<tr>
<td>24</td>
<td>0.6805</td>
<td>0.8627</td>
<td>0.9342</td>
<td>0.9625</td>
<td>0.9763</td>
<td>0.9824</td>
<td>0.9886</td>
</tr>
<tr>
<td>32</td>
<td>0.6864</td>
<td>0.8697</td>
<td>0.9396</td>
<td>0.9680</td>
<td>0.9813</td>
<td>0.9874</td>
<td>0.9935</td>
</tr>
<tr>
<td>40</td>
<td>0.6892</td>
<td>0.8730</td>
<td>0.9421</td>
<td>0.9705</td>
<td>0.9837</td>
<td>0.9898</td>
<td>0.9958</td>
</tr>
<tr>
<td>(C_{(j)})</td>
<td>0.6942</td>
<td>0.8791</td>
<td>0.9468</td>
<td>0.9752</td>
<td>0.9881</td>
<td>0.9942</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Table 7.9 Capacity values for RDS-TMTR \((N,r)\) constraints for \(r = (j,j + 1)\).

<table>
<thead>
<tr>
<th>(N) (\backslash j)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>(C_N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.3471</td>
<td>0.5234</td>
<td>0.6060</td>
<td>0.6420</td>
<td>0.6657</td>
<td>0.6763</td>
<td>0.6942</td>
</tr>
<tr>
<td>6</td>
<td>0.5840</td>
<td>0.7284</td>
<td>0.7917</td>
<td>0.8184</td>
<td>0.8338</td>
<td>0.8406</td>
<td>0.8495</td>
</tr>
<tr>
<td>8</td>
<td>0.6702</td>
<td>0.8051</td>
<td>0.8611</td>
<td>0.8850</td>
<td>0.8978</td>
<td>0.9036</td>
<td>0.9103</td>
</tr>
<tr>
<td>10</td>
<td>0.7118</td>
<td>0.8426</td>
<td>0.8950</td>
<td>0.9177</td>
<td>0.9292</td>
<td>0.9345</td>
<td>0.9403</td>
</tr>
<tr>
<td>12</td>
<td>0.7352</td>
<td>0.8638</td>
<td>0.9142</td>
<td>0.9362</td>
<td>0.9471</td>
<td>0.9521</td>
<td>0.9575</td>
</tr>
<tr>
<td>16</td>
<td>0.7593</td>
<td>0.8857</td>
<td>0.9339</td>
<td>0.9553</td>
<td>0.9655</td>
<td>0.9703</td>
<td>0.9752</td>
</tr>
<tr>
<td>20</td>
<td>0.7708</td>
<td>0.8963</td>
<td>0.9434</td>
<td>0.9645</td>
<td>0.9744</td>
<td>0.9791</td>
<td>0.9838</td>
</tr>
<tr>
<td>24</td>
<td>0.7772</td>
<td>0.9022</td>
<td>0.9487</td>
<td>0.9696</td>
<td>0.9794</td>
<td>0.9840</td>
<td>0.9886</td>
</tr>
<tr>
<td>32</td>
<td>0.7837</td>
<td>0.9082</td>
<td>0.9541</td>
<td>0.9748</td>
<td>0.9844</td>
<td>0.9890</td>
<td>0.9935</td>
</tr>
<tr>
<td>40</td>
<td>0.7868</td>
<td>0.9110</td>
<td>0.9567</td>
<td>0.9773</td>
<td>0.9868</td>
<td>0.9913</td>
<td>0.9958</td>
</tr>
<tr>
<td>(C_{(j,j+1)})</td>
<td>0.7925</td>
<td>0.9163</td>
<td>0.9613</td>
<td>0.9818</td>
<td>0.9912</td>
<td>0.9957</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 7.10 Capacity values for RDS-MTR \((N,j,k)\) constraints for \(j = 3\).

<table>
<thead>
<tr>
<th>(N) (\backslash k)</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>(C_{N,j=3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.5943</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5943</td>
</tr>
<tr>
<td>6</td>
<td>0.7149</td>
<td>0.7777</td>
<td></td>
<td></td>
<td></td>
<td>0.7777</td>
</tr>
<tr>
<td>8</td>
<td>0.7518</td>
<td>0.8393</td>
<td>0.8468</td>
<td></td>
<td></td>
<td>0.8468</td>
</tr>
<tr>
<td>10</td>
<td>0.7679</td>
<td>0.8669</td>
<td>0.8795</td>
<td>0.8807</td>
<td></td>
<td>0.8807</td>
</tr>
<tr>
<td>12</td>
<td>0.7764</td>
<td>0.8816</td>
<td>0.8974</td>
<td>0.8996</td>
<td>0.8998</td>
<td>0.8998</td>
</tr>
<tr>
<td>16</td>
<td>0.7846</td>
<td>0.8962</td>
<td>0.9153</td>
<td>0.9188</td>
<td>0.9194</td>
<td>0.9195</td>
</tr>
<tr>
<td>20</td>
<td>0.7883</td>
<td>0.9028</td>
<td>0.9237</td>
<td>0.9279</td>
<td>0.9288</td>
<td>0.9290</td>
</tr>
<tr>
<td>24</td>
<td>0.7903</td>
<td>0.9064</td>
<td>0.9283</td>
<td>0.9330</td>
<td>0.9340</td>
<td>0.9342</td>
</tr>
<tr>
<td>32</td>
<td>0.7922</td>
<td>0.9100</td>
<td>0.9328</td>
<td>0.9380</td>
<td>0.9392</td>
<td>0.9396</td>
</tr>
<tr>
<td>40</td>
<td>0.7931</td>
<td>0.9117</td>
<td>0.9350</td>
<td>0.9404</td>
<td>0.9417</td>
<td>0.9421</td>
</tr>
<tr>
<td>(C_{j=3,k})</td>
<td>0.7947</td>
<td>0.9146</td>
<td>0.9388</td>
<td>0.9447</td>
<td>0.9462</td>
<td>0.9468</td>
</tr>
</tbody>
</table>
Table 7.11 Capacity values for RDS-MTR \((N, j, k)\) constraints for \(j = 4\).

<table>
<thead>
<tr>
<th>(N) (\backslash k)</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>(C_{N,j=4})</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.6174</td>
<td></td>
<td></td>
<td></td>
<td>0.6174</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.7496</td>
<td>0.8052</td>
<td></td>
<td></td>
<td>0.8052</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.7900</td>
<td>0.8683</td>
<td>0.8748</td>
<td></td>
<td>0.8748</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.8078</td>
<td>0.8968</td>
<td>0.9078</td>
<td>0.9087</td>
<td>0.9087</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.8172</td>
<td>0.9121</td>
<td>0.9259</td>
<td>0.9277</td>
<td>0.9279</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.8263</td>
<td>0.9273</td>
<td>0.9442</td>
<td>0.9472</td>
<td>0.9476</td>
<td>0.9477</td>
</tr>
<tr>
<td>20</td>
<td>0.8304</td>
<td>0.9343</td>
<td>0.9528</td>
<td>0.9564</td>
<td>0.9571</td>
<td>0.9572</td>
</tr>
<tr>
<td>24</td>
<td>0.8326</td>
<td>0.9381</td>
<td>0.9575</td>
<td>0.9615</td>
<td>0.9623</td>
<td>0.9625</td>
</tr>
<tr>
<td>32</td>
<td>0.8348</td>
<td>0.9419</td>
<td>0.9622</td>
<td>0.9667</td>
<td>0.9677</td>
<td>0.9680</td>
</tr>
<tr>
<td>40</td>
<td>0.8359</td>
<td>0.9436</td>
<td>0.9644</td>
<td>0.9691</td>
<td>0.9702</td>
<td>0.9705</td>
</tr>
<tr>
<td>(C_{j=4,k})</td>
<td>0.8376</td>
<td>0.9468</td>
<td>0.9684</td>
<td>0.9735</td>
<td>0.9748</td>
<td>0.9752</td>
</tr>
</tbody>
</table>

7.B Power Spectral Density Plots

Figure 7.6 Power spectral densities of the RDS constraints.
Figure 7.7 Power spectral densities of TMTR \((r, k)\) constraints for \(r = (3, 4)\).

Figure 7.8 Power spectral densities of RDS-TMTR \((N, r)\) constraints for \(r = (3, 4)\).
Chapter 8

Conclusions

This dissertation investigated some aspects of 2-D ISI channels for holographic data storage systems. Modelling, equalization and detection for these channels are described in Chapter 2. Misalignments between the data pages and the detector arrays can cause serious performance degradations in detectors. The method of oversampling appear to be a solution to this problem without any performance loss but at the cost of increased complexity (see Chapter 3). Chapter 4 discusses the performance assessment of an ML detector by generating common error events in the channel. One way to improve the channel performance is to eliminate common error patterns using a modulation code. Chapter 5 described efficient algorithms for finding single-state and finite-state block codes for the hard-square constraint. Chapter 6 proposed a depth-first algorithm to extend single-state block codes to any 2-D constraint represented by a set of forbidden patterns.

Chapter 2 described modelling, equalization and detection for 2-D ISI channels. Modelling of 2-D ISI channels are complex, in part due to the shortcomings of camera detectors, such as the ability of measuring intensity of the optical signal only, and the non-linear effects of the optical system. The LTI and weighted LTI channel models still seem to be the best option for modelling of these channels, since the methods for equalization and detection for non-linear channels are not widely known. The detectors for 2-D ISI channels, such as the IMS algorithm, has large computational complexity due to the lack of graph based descriptions of such channels. Low complexity detectors such as a threshold detector and the IMS algorithm using low-complexity SISO detectors can serve as better alternatives to the IMS algorithm in practice.

Chapter 3 addressed the misalignment problem in holographic data storage. The method of oversampling proposed to solve this problem is discussed with its pros and
cons. The oversampled channels are linear but are periodically time-varying. The implementation of the channel characterization and equalization for these channels pose no difficulty, but the design of an optimal detector is challenging. We proposed a MAP algorithm working on the time-varying periodic trellises. The complexity of this algorithm is not much different than the one working on a regular trellis, yet its implementation is tedious. Simulation results suggest that oversampling does not improve the MAP detector performance in terms of the bit-error-rate. However, the MAP detector performance for oversampled channels does not vary for different misalignments.

Chapter 4 described a method for computing the minimum distance and near minimum distance closed error events of the 2-D PR1 channel. Some open error events are also characterized but the complete description of the open error events is an open problem. The effect of a precoding scheme for the 2-D PR1 channel is also investigated. The main advantage of precoding for this channel is that it allows threshold detection to be used as a channel detector. Error events for any 2-D ISI channel can be generated by using the bounded-depth search algorithm working on error state diagrams. We also proposed a lower bound on distances of error events for any 2-D ISI channel. This bound is observed to be very loose for estimating the distance of error events, but it is a good bound on the minimum closed event distance of the 2-D ISI channels.

Chapter 5 described algorithms to find optimal single-state and finite-state block codes for the hard-square model. Encoding and decoding of constrained arrays can be performed easily using generating templates. The algorithms described in this chapter can be extended to other first-order constraints. However the extension to constraints with order larger than one is very complex and tedious. The lack of graph-based representations for 2-D constraints is still the main difficulty in developing a general theory.

Chapter 6 discussed the depth-first algorithm for finding single-state block codes for any 2-D constrained system represented by a set of forbidden patterns. The advantage of modulation codes are illustrated for a set of channels using a threshold detector. The modulation codes are observed to be useful when the ISI in the channel is large. The extension of this method to find finite-state block codes is an open problem.

Chapter 7 switched from 2-D storage systems to a 1-D recording problem. This chapter discussed a code design methodology for the design of high-rate modulation codes implementing several constraints for perpendicular recording channels. The RDS and TMTR constraints are combined to observe the design trade-off between the coding rate and the
constraint parameters. The rate of a combined constraint is observed to be slightly less than its constituent constraints. The spectra of a combined constraint shows the effect of its constituent constraints. We have illustrated the code design methodology via a rate 100/108 block code.
Bibliography


